Universidade de Aveiro Departamento de Engenharia Mecânica
2017

Joana Gil Távora de Almeida Ferreira

Interface háptica distribuída para aplicações de teleoperação robótica e reabilitação
Distributed haptic interface for applications in robotic teleoperation and rehabilitation

Universidade de Aveiro Departamento de Engenharia Mecânica
2017

Joana Gil Távora de Almeida Ferreira

Interface háptica distribuída para aplicações de teleoperação robótica e reabilitação
Distributed haptic interface for applications in robotic teleoperation and rehabilitation

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de Vitor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro e de Filipe Miguel Teixeira Pereira da Silva, Professor Auxiliar do Departamento de Eletronica e Telecomunicações da Universidade de Aveiro.

Apoio financeiro dos projetos UID/EMS/00481/2013-FCT e CENTRO-01-0145-FEDER-022032

O júri / The jury

Presidente / President

Prof. Doutor José Paulo Oliveira Santos
Professor Auxiliar da Universidade de Aveiro

Vogais / Committee

Prof. Doutor Jorge Manuel Mateus Martins
Professor Associado do Departamento de Engenharia Mecânica do Instituto Superior Técnico

Prof. Doutor Vitor Manuel Ferreira dos Santos
Professor Associado da Universidade de Aveiro (orientador)

Agradecimentos / Acknowledgements

Em primeiro lugar, um agradecimento aos professores Vitor Santos e Filipe Silva pela sua orientação e apoio dados durante todo este projecto.

Um obrigada ao Professor Miguel Oliveira e ao Heber Sobreira pela partilha do seu conhecimento e disponibilidade.

Um obrigada ao Professor António Amaro e ao Doutor Paulo Margalho pela sua ajuda e colaboração .

Um obrigada aos meus colegas do Laboratório de Automação e Robótica por toda a ajuda.

À minha tia Fátima, um enorme obrigada. Sem ela este projecto nunca teria ganho vida. Literalmente!

Por ultimo, um muito obrigada aos meus pais. Sem eles, nunca teria sido possivel. Ao meu pai, pela referência que foi e por toda a ajuda e conselhos que me foi dando ao longo dos anos. À minha mãe, um obrigada por todo o apoio e por não me ter expulsado de casa depois de não me ver por uns quantos dias.

A todos, muito obrigada!

Palavras-chave

Háptica; interface háptica; háptica distribuída; reabilitação; teleoperação robótica; motores vibratórios.

Resumo

A tecnologia háptica está cada vez mais presente na sociedade em áreas tão diversas como a saúde, a robótica e o lazer. Nesta dissertação desenvolveu-se um dispositivo háptico com o objetivo de ser utilizado na teleoperação robótica e na reabilitação. O dispositivo desenvolvido permite ao utilizador, através de uma interface gráfica, controlar uma rede de mini-motores vibratórios individualmente ou utilizando estímulos previamente definidos. O utilizador consegue assim definir que motor, ou que conjunto de motores, pretende activar e a intensidade dos mesmos.

No desenvolvimento do sistema utilizou-se a arquitetura ROS e a arquitetura Qt para a implementação do sistema de comunicação e da interface gráfica, e a plataforma Arduino para o comando dos motores vibratórios. Para a funcionalidade do dispositivo na área da reabilitação foi desenvolvida uma luva/manga composta por dezasseis motores vibratórios dispostos em pontos estratégicos tendo em conta os nervos do ser humano.

Todo o sistema foi testado e avaliado por pessoal médico e por uma amostra de voluntários com vários *backgrounds*.

Uma análise realizada aos dados recolhidos mostrou que o método proposto foi bem sucedido.

Keywords Haptic; haptic interface; distributed haptic; rehabilitation; robotic teleoperation; vibration motors.

Abstract The haptic technology is increasingly present in society in diverse areas such as health, robotics and recreation. In the present work, a haptic device with the objective of being used in robotic teleoperation and rehabilitation was developed. The developed device allows the user, through the use of a graphical interface, to control a network of mini vibration motors, either individually or using predefined stimuli. Therefore, the user can define which motor, or which group of motors, to turn on and its intensity.
In the development of the system the platform ROS and platform Qt were used in the implementation of the communication system and the graphical interface, and the platform Arduino was used to do the command of the vibration motors. To the functionality of the device, a glove/sleeve composed by sixteen vibration motors displayed in strategical points according the human being dermatomes was developed.
The entire system was tested and evaluated by clinicians and by a sample of volunteers with different backgrounds.
An analysis to the collected data showed that the proposed method was successful.

Contents

List of Tables

List of Figures

List of Acronyms

μC Microcontroller. 15, 23, 25, 63

3D Three-Dimensional. 79

BLDC Brush-less Direct Current. 16

BMP Bitmap image file. 29

CAD Computer Aided Design. 33, 79

CNS Central Nervous System. 9

CPU Central Processing Unit. 23

DC Direct Current. 16, 18, 63

DEM Mechanical Engineering Department. 80

DETI Department of Electronics, Telecommunications and Informatics. 33

DKKI German Research Center for Artificial Intelligence. 6

DoF Degrees of Freedom. 1

ERM Eccentric Rotating Mass. 16–18

FAT File Allocation Table. 29, 71

FTS Flight Telerobotic Servicer. 6

GUI Graphical User Interface. 24, 25, 61

IDE Integrated Development Environment. 22, 24

ISR Interrupt Service Routine. 28, 29

IST Technical University of Lisbon. 7

LAR Laboratory for Automation and Robotics of the University of Aveiro. 29

LRA Linear Resonant Actuator. 16

MOC Meta-Object Compiler. 24

NASA National Aeronautics and Space Administration. 6

NASREM NASA/NBS Standard Reference Model for Telerobot Control System Architecture. 6

PCB Printed Circuit Board. 33

PNS Peripheral Nervous System. 9

PWM Pulse Width Modulation. 19, 20, 23, 27–29, 63, 71

ROS Robot Operating System. 20–23, 25, 27, 42, 43, 50, 63

SAIL Stanford Artificial Intelligence Laboratory. 20

UART Universal Asynchronous Receiver/Transmitter. 22

US United States. 6

USB Universal Serial Bus. 15, 29, 71, 79

Chapter 1

Introduction

1.1 Motivation and background

The perception of the state of complex systems by an operator can be a huge challenge, either by the number of Degrees of Freedom (DoF) or by the way that information is conveyed to the operator senses. That is particularly relevant in robotic teleoperation of robots with several DoF, but also in the evaluation of effort states, pain or discomfort of human beings in physical exercise like, for example, in medical rehabilitation.

To allow the operator a richer perception of a complex system state (like a robotic system or some electromyographic parameters of a human being in a therapeutic rehabilitation process), the development of an equipment that allows the perception by contact in an area of the operator's body was thought. This contact should somehow reflect one or more extern quantities that can come either from real or simulated sensors, connected to a robotic system (or a human patient) like humanoid robots. These sensations in the operator can transmit a big wealth of information and can complement other traditional perception processes like vision or hearing.

In Robotics (for instance in the case of humanoid robots) these sensations can translate mechanical efforts, motion resistance, current, force sensors in robots, etc. In the human being they can translate position states, cutaneous pressure or muscular effort (by electromyography, body temperature, etc), or just simply create sensations that can emulate states or perception contexts for virtual reality or immersive multimedia experiences like multidimensional cinema.

When force feedback is used, it requires a counteract by the operator. However, if instead of a force feedback a vibration feedback is used, the user does not have to oppose the force, reducing the physical effort the user has to do. For this reason, and for an easier comprehension on what will be done in this project, it is proposed for haptic to be divided as Figure 1.1 shows. On the left side of the figure is the force and motion part and on the right side is the vibration part.

Since not every person (operator) feels touch the same way, a system calibration is necessary. That calibration can be done by applying stimuli in the operator just like in a diagnostic exam and then comparing the applied stimulus to what the operator really feels.

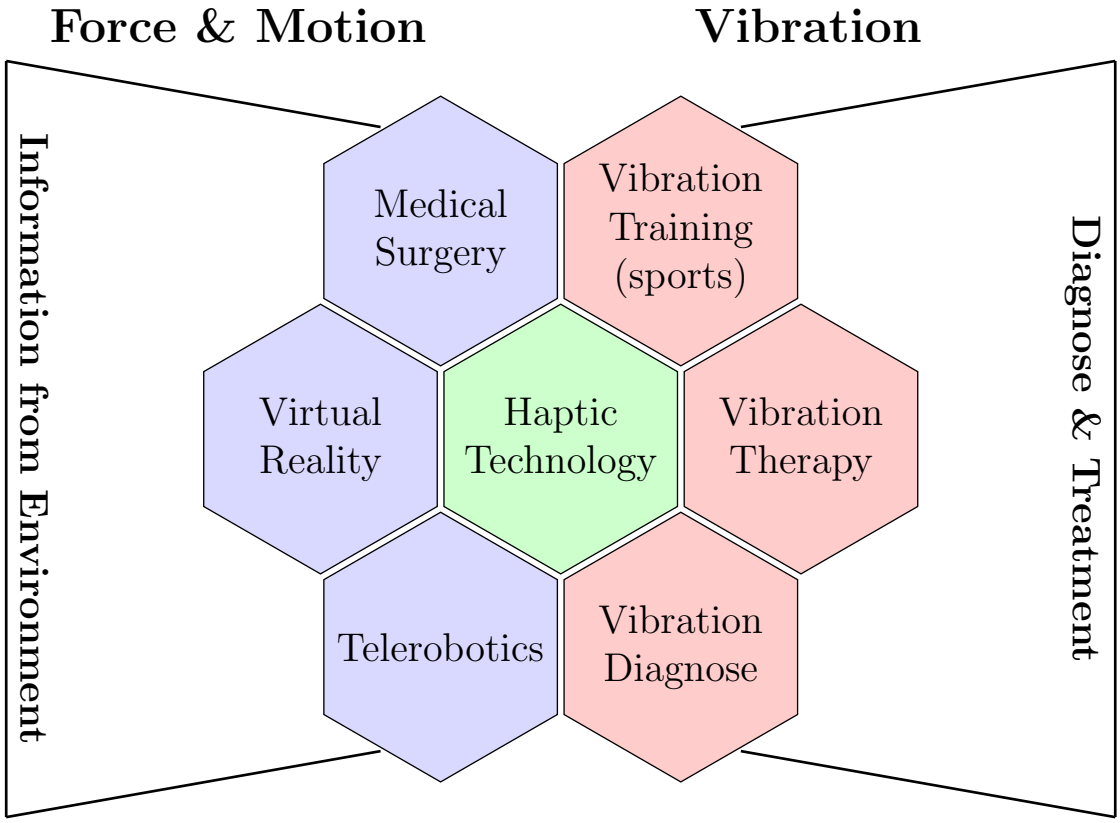

Figure 1.1: Classification based on goal of the stimulation

The stimulus can have several types like pressure, vibration, sound, temperature or electro-stimulation. In this project, it is planned to explore essentially a contact form based in vibrations with adjustable intensity on the creation of distributed stimuli, therefore this project is going to be focused in the right side of the scheme in Figure 1.1. This vibration stimulation can be achieved with the use of vibration motors.

1.2 Objectives

To summarize what was initialized before, the main objectives for this project can be defined as the following:

- Development of a command system for a network of vibration motors;
- Development of a usable prototype so that the haptic interface parameters can be tested in a human being;
- Definition and test of variable stimulation patterns.

1.3 State of the Art

1.3.1 Principles of Haptic

Traditionally, there are five perception methods, the famous *five senses*: palate, vision, smell, hearing and touch. With the absence of some of these senses, very simple tasks like standing, walking, seating, grabbing an object and writing can become very tricky. It is known that patients with total loss of their tactile senses are unable to drink out of soft plastic cups since they are unable to grasp and hold to the cup. Furthermore, when we place an object on a surface we combine information from our memory regarding the weight, compliance of the object and the friction of the surface [1] with the information we receive from our tactile sense and our visual sense (for example the distance between the object and surface) [2].

The touch is the first sense to develop in infants [3] and Thomas Aquina said in *De Anima* that, without the possibility of touch, no other sense could exist: touch is «the first sense, the root and ground, as it were, of the other senses» [4], defining touch as the most important sense of all.

Even though that in modern times vision is considered the dominant sense, the touch continues to be important to the human behavior[5]. To this sense of touch the concept of haptic is associated.

The term *haptic* comes from the Greek word *haptikos* which means being able to lay a hold of, be able to perceive, to grasp [6]. Since the early 20th century, psychologists have been using this term to label the subfield of their studies that addressed human touch-based perception and manipulation [7].

In 1892 Max Dessoire introduced the concept of haptic as "the science of the human touch" [8]. In 1966 Gibson defined it as "the sensibility of the individual to the world adjacent to his body by use of his body" [9].

In the 70s and 80s, the robotic community also began to focus on manipulation, perception and interaction by touch [10].

Haptic also includes kinesthesia, also known as proprioception, which is the ability to perceive one's body position, movement and weight, therefore becoming common to designate the sensory motor components of haptic as the haptic channel [11].

Currently, there are already some gadgets that use the tactile perception channel as they send out vibratory signals. For example, the vibratory mode of mobile phones and rumble packs in gaming controls have been in our daily routines since a long time ago [12].

Haptic is also used for example in the military, not only for military training and safety enhancement, but also to analyze military maneuvers and operate vehicles. More, in cars and flight simulators the operator can also control the vehicle using for example a joystick and feel the feedback that it gives to the user.

Another example occurs in the medical training. Due to the haptic contact, it is possible to simulate the contact with organs, which allows medicine students (and doctors) to practice for surgeries with an hologram and the touch sensation.

More advanced haptic interfaces make use of force feedback devices that can not only measure the position and contact forces of the user's hand, but also feedback position and force signals to the user[13]. Thus, haptic interfaces allows human-machine interaction through touch and mostly in response to user movements.

1.3.2 Virtual Reality

In the 90s the technology had improved and a new type of haptic began to emerge: the virtualized haptic (computer haptic). As the name indicates, this is no more than a virtual environment where the user can feel by applying forces, vibrations and/or motion on the user. This mechanical simulation may be used to assist in the creation of computed virtual objects that can be physically palpated and controlled and also to allow remote control of machines and devices.

This new sensory display modality presents information by exerting controlled forces on the human hand through a haptic interface, rather than, as in computer graphics, in the form of visual and auditory signals [14]. Unlike computer graphics where audio and visual channels feature unidirectional information and energy flow (system to user), haptic interaction is bidirectional as humans send and receive haptic signals[15].

Figure 1.2 illustrates the subsystems and information flow related to the interactions between users and haptic interfaces. On the right side there is the dynamical system of the haptic interface with a computer, and on the left side there is a dynamical system of the user with the central nervous system [16].

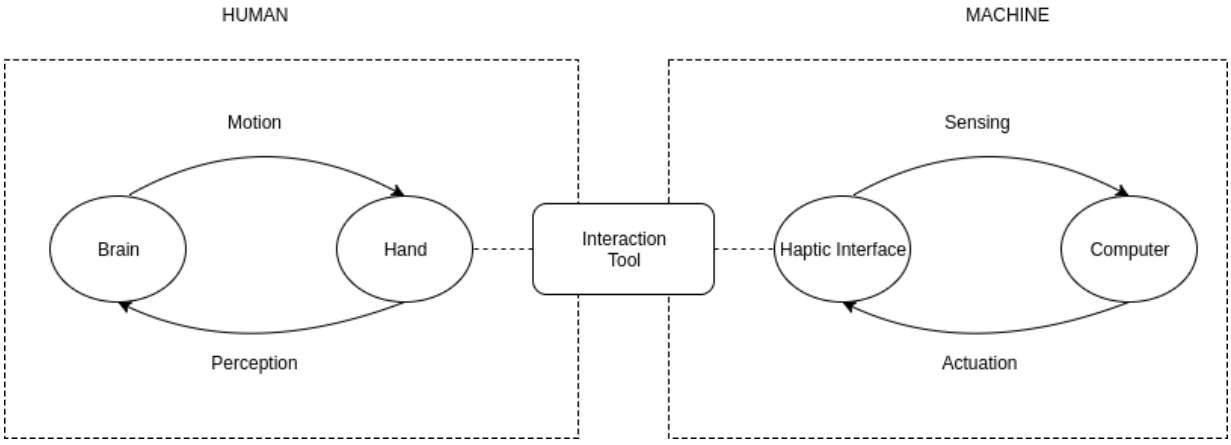

Figure 1.2: Haptic interaction between humans and machines (adapted [16])

- *Human sensorimotor loop:* when a human user touches a real or virtual object, forces are imposed on the skin. The associated sensory information is conveyed to the brain and leads to perception. After interpreting the environment, the brain motor commands activate the muscles and result in hand and arm motion.

- *Machine sensorimotor loop:* when the human user manipulates the end-effector of the haptic interface device, the position sensors on the device convey its tip position to the computer. The models of objects in the computer calculate in real-time the torque commands to the actuators on the haptic interface, so that appropriate reaction forces are applied on the user, leading to tactual perception of virtual objects.

Virtual reality technology allows the user to interact with computer simulated objects in real time and under real or imaginary conditions, and it is one of the technologies where

haptic is the most useful. It is used in gaming but also in medical activity(Section 1.3.4), robotics (Section 1.3.3) and graphical art applications [17]. Figure 1.3 shows the basic architecture for a virtual reality application incorporating visual, auditory and haptic feedback.

Application's main elements include:

- *simulation engine*, responsible for computing the virtual environment's behavior over time;
- *visual, auditory and haptic rendering algorithms*, which compute the virtual environment's graphic, sound and force responses toward the user;
- *transducers*, which convert visual, audio, and force signals from the computer into a form the operator can perceive;
- *human operator*, who typically interacts (bidirectional) with the haptic interface device and perceives audio and visual feedback , from computer, headphones, visual displays, etc.

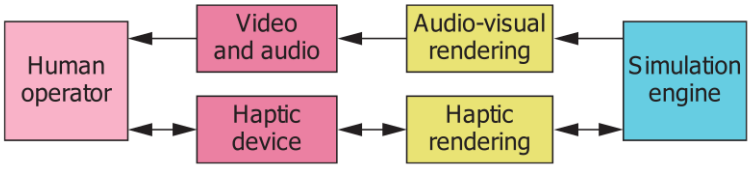

Figure 1.3: Basic architecture for a virtual reality application incorporating visual, auditory and haptic feedback[14])

Being able to touch and manipulate objects in an environment, in addition to just seeing or hearing them, provides a sense of immersion in the environment to the user, otherwise not possible, making haptic very important to the concept of virtual reality.

1.3.3 Robotic Teleoperation

Teleoperation is the technical term for the remote control of a robot at a distance [18]. A teleoperation system allows humans to perform tasks that can be in an inaccessible environment without putting themselves at risk.

Usually, a single operator is in control of the robot and feels some level of immersion in the remote environment [19].

Figure 1.4 sketches a teleoperation system set-up. The core idea of teleoperation is the connection between two distinct spatial domains which can be referred as the operator environment (where the human user is) and the teleoperator environment (where the robot/manipulator is)[12].

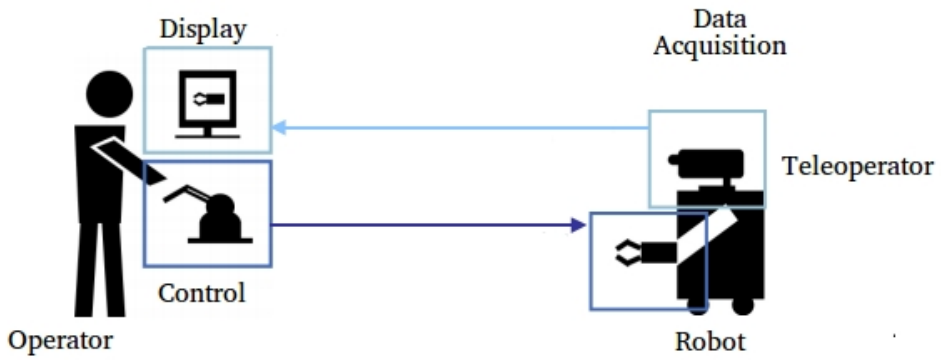

Figure 1.4: Teleoperation System (adapted [20])

As can be seen in Figure 1.4, on the left, in the operator environment, a operator (human) interacts with the controls and displays which define the human-machine interface of the operator device. A communication channel links the operator to the teleoperator device in the remote environment, where it controls the teleoperator's manipulator via actuators. This actuator usually has clamps or grippers as an end-effector that interacts physically with the objects in the remote environment [12].

The industrial teleoperation systems have been around since the 50s when the Argonne National Laboratory developed a manipulation system with the aim of handling radioactive materials safely[21]. Later, in the 60s, a teleoperation system was also used in the United States (US) Army in the decommissioning of a nuclear warhead [22]. Both of these examples show how important robotic teleoperation is, in the sense that it allows the human being to do something that could be dangerous for its safety.

For instance, in the space industry the robotic teleoperation is also frequently used. There are several contributions of the US National Aeronautics and Space Administration (NASA) in that area, such as the Flight Telerobotic Servicer (FTS) program [23] and the NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM) [24].

Although that in these past projects the robotic teleoperation was made without haptic feedback, currently there are already a few ones that use this technology.

One of those examples is the AILA robot (Figure 1.5). The German Research Center for Artificial Intelligence (DKKI) developed a system that allows to control a robot using an interface called CAPIO that simulates an exoskeleton at a distance of around 4000 km. As the operator is in Magnitogorsk in Russia, the AILA robot is in Bremen in Germany.

As the user moves, the CAPIO interface transfers those movements to the robot and returns haptic feedback of the robot movement [25].

Figure 1.5: AILA with Operator [26]

Another example of a haptic system is the OCEAN ONE (Figure 1.6) developed by the University of Stanford. The OCEAN ONE is a underwater humanoid robot with haptic feedback that allows the researchers access to the deepness of the ocean, which use to be impossible for the human beings.

This robot is equipped with force sensors in each hand and, being controlled by an operator, thanks to the haptic feedback, this operator can feel the weight and shape of what it finds under the sea [27].

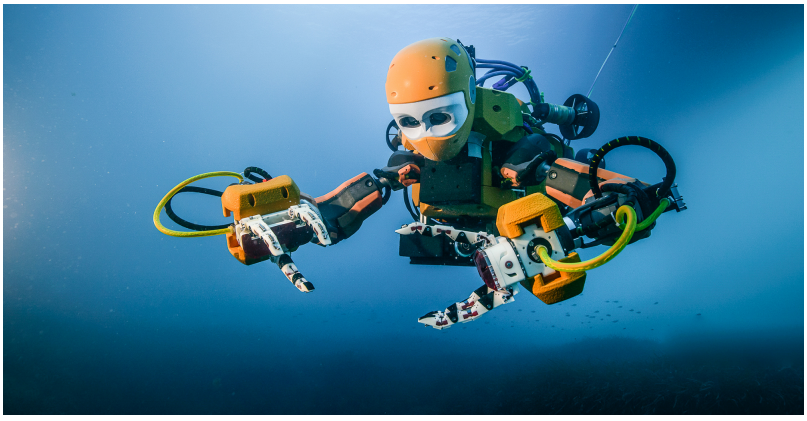

Figure 1.6: OCEAN ONE [28]

The examples previously mentioned all work under the force feedback method. An example of a teleoperation system using vibration feedback can be found in a study carried out by the Technical University of Lisbon (IST). In this study the teleoperation of an unmanned vehicle using different kinds of haptic feedback was tested, such as standard force feedback, stiffness feedback and vibration feedback. Unlike the first two methods that require the user to counteract the feedback force (which may increase the workload of the task), vibration feedback allows the information regarding the unmanned vehicle

surroundings to be intuitively transmitted without demanding high physical efforts from the operator [29].

On the medical department, with the increasing of the preference for non-invasive surgeries, the surgeon stops having the perception and palpation that previously had with the conventional surgery.

Thankfully, there are already some tele-tactile feedback systems in which a sensor present in the equipment that the doctor is using, allows the recording of every aspect of the interactions with the tissues. Then, those details are processed and sent to the tactile display of a master robot.

One example of robotic teleoperation in medicine is the very well known Da Vinci robots (Figure 1.7). However, this system does not have haptic feedback yet.

Figure 1.7: Da Vinci Robotic Surgery System [30]

1.3.4 Rehabilitation

Physical rehabilitation is the process of restoring and regaining physical strength and function [31]. It is used to return (or give) the maximum of functionality to a person that has an acquired condition (for instance, to help treat a simple ankle sprain or to help a stroke survivor walk, talk and eat again). In rehabilitation, what the doctor tries to do is to restore, in an anatomic way, but more important, give the patient autonomy and functionality executing daily routines with the maximum normality. Figure 1.8 is an example of what rehabilitation does, allowing the patient to regain the capability of walking normally.

It can be used in orthopedic cases, musculoskeletal, neurological, cardiorespiratory, palliative and so on. It is applied since birth till death.

Figure 1.8: Example of rehabilitation where the patient regains the capability of walking normally [32]

Sometimes, only one part of the spinal cord gets injured and depending on which part a person can feel different symptoms. This happens because the information regarding vibration and position/proprioception sensitivity is sent to a different part of the spinal cord than the pain/temperature sensitivity. For instance, a person can be able to feel pain but not know how their foot is positioned or feel vibration.

Due to the fact that the vibration felt with this motors is considered a superficial vibration (like the superficial pressure and slight touch) the information received by this sense of vibration is transmitted through the same channels as the pain and temperature (spinothalamic tract or anterolateral system of the spinal cord).

1.3.5 Nervous System

The nervous system is basically a collection of nerves (cylindrical bundles of fibers that start at the brain and central cord) and specialized cells (neurons) that transmit signals between different parts of the body[33].

In vertebrates, it consists of two main parts: the Central Nervous System (CNS) that consists of the brain and spinal cord and the Peripheral Nervous System (PNS) that consists mainly of nerves that connect the CNS to every other part of the body.

The nerves that transmit data from the body to the brain are called sensory nerves. On the other hand, the motor nerves work the other way around. The Spinal nerves can operate both to receive and send data from the body to the brain, so they are called mixed nerves.

Neurons send signals to other cells through thin fibers called axons that cause chemicals known as neurotransmitters to be release at junctions called synapses [34]. Then the synapse gives a command to the cell. This entire communication process usually only takes a fraction of a millisecond.

Sensory nerves react to physical stimulus and send feedback to the CNS and the motor nerves transmit signals to active the muscles or glands [33].

A stimulus is an external or internal signal capable of causing a reaction in a cell or

body. Every stimulus has four important features:

- Type (sensory way);
- Intensity;
- Location;
- Duration.

The type of stimulus can be classified as pressure, vibration, temperature, electro-stimulation, among others.

One of the most important part of the nervous system is the spinal cord (Figure 1.9). It is a long structure with a cylindrical shape that begins at the end of the brain stem and continues down to the upper lumbar region.

Emerging from the spinal cord between the vertebrae are 31 pairs of spinal nerves [35] [36]:

- 8 cervical (C);
- 12 thoracic (T);
- 5 lumbar (L);
- 5 sacral (S);
- 1 coccygeal (Co) - mainly vestigial.

Each nerve emerges in two short branches (roots):

- One at the front (motor or anterior/ventral root) of the spinal cord;
- One at the back (sensory or posterior/dorsal root) of the spinal cord.

The spinal nerves consist of the sensory nerve roots (dorsal), which enter the spinal cord at each level, and the motor roots (ventral), which emerge from the cord at each level [35].

The surface of the skin is divided into specific areas called dermatomes (Figure 1.10) in which sensory nerves derive from a single spinal nerve root. Dermatomes are useful to help localize neurological levels [38] because when information is detected in a dermatome the data is then sent to one of the peripheral nerves.

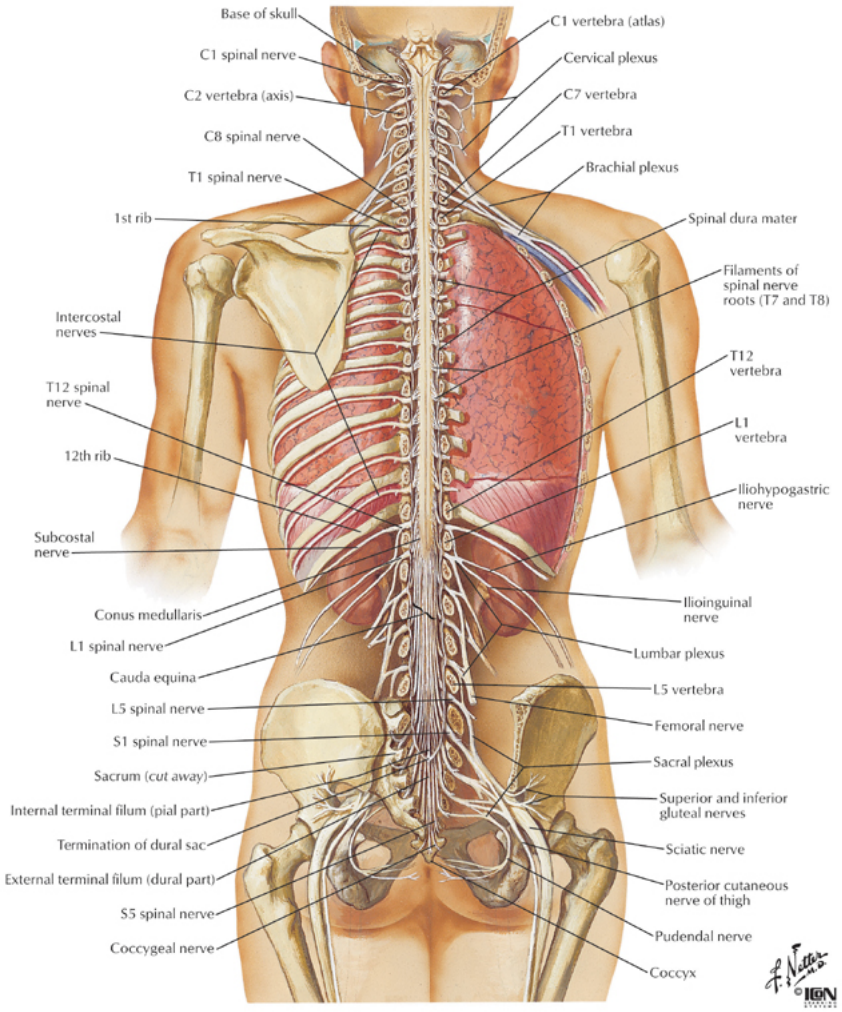

Figure 1.9: Spinal Cord and Peripheral Nerves [37]

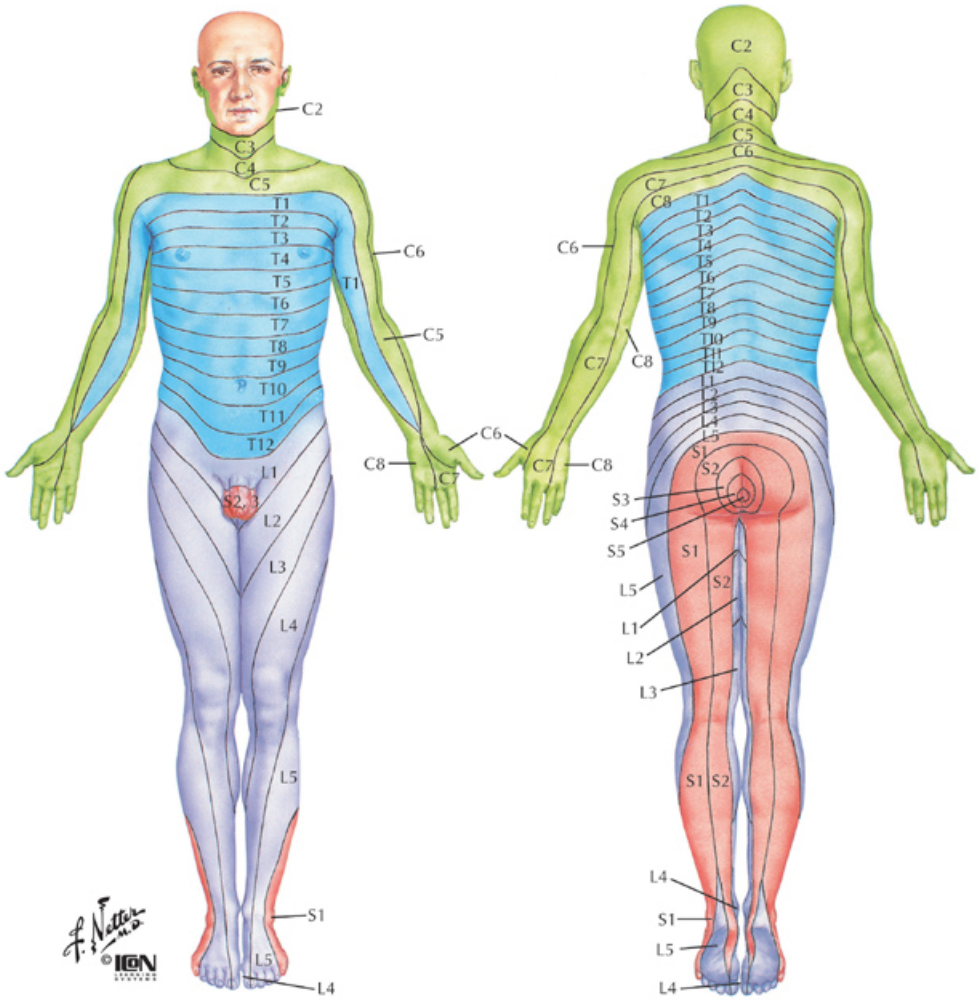

Figure 1.10: Schematic demarcation of dermatomes shown as distinc segments [37]

1.4 Thesis Guide

The stages of this work are presented in the following chapters, organized as follows:

- Chapter 1 briefly explained the context and objectives of the current project. A state of the art review on the principles of haptic, robotic teleoperation, nervous system and rehabilitation were also exposed;
- Chapter 2 consists of the presentation and explanation of the hardware and software used;

- Chapter 3 explains the development of the application both in the hardware and software part;
- Chapter 4 presents the experiments undertaken to test and validate the methodologies described in this project;
- Chapter 5 discusses the conclusions of this project and also presents some proposals for future works.

Chapter 2

Hardware and Software of the Infrastructure

The idea around the solution is to build a motor network that can be placed in a prototype and be tested. To do so, a micro-controller, vibration motors and a graphical interface are needed. The micro-controller used is an Arduino, the interface was developed using Qt and ROS is responsible for the communication between the components. This chapter describes all those components and their inter-connections.

2.1 Hardware

2.1.1 Arduino Mega

To control the motors a micro-controller is needed. The choice made was to use an Arduino. Arduino is an Italian open source computer hardware and software company that produces prototyping and testing boards. Arduino board designs use a variety of microprocessors and controllers, the most usual being ATMEL micro-controller. The boards have both digital and analog input/output (I/O) pins that may be expandable by stacking shields and other circuits to them. These boards feature serial communications interfaces, including Universal Serial Bus (USB) which are also used for loading programs from personal computers. It is a very affordable solution and also easily expandable[39].

There are several Arduino models available, such as Arduino UNO, Arduino Micro, Arduino Nano, Arduino Mega (Figure 2.1), Arduino DUE, Arduino Leonardo and others.

Table 2.1 shows some important characteristics of Arduino Mega 2560.

The biggest reason to chose Mega over other Arduino is the fact that, besides Arduino DUE, Mega is the Microcontroller (μC) with the most digital IO pins (both with 54). However, since the Mega operating voltage is 5V and the DUE is 3.3V [41], Arduino Mega was selected.

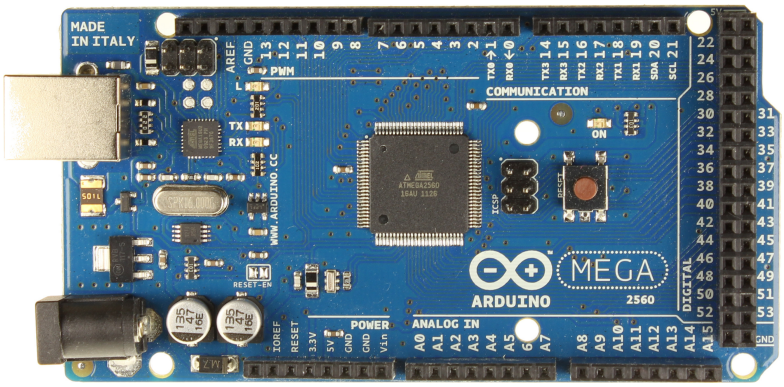

Figure 2.1: Arduino Mega 2560 [40]

Table 2.1: Arduino Mega 2560 characteristics [40]

CPU	ATMega2560
Operation Voltage	5V
Input Voltage	7-12V
Digital I/O Pins	54 (15 PWM output)
Analog Input Pins	16
Flash Memory	256KB (8KB used by bootloader)
EEPROM	4KB
SRAM	8KB
Clock Speed	16 MHz
Dimensions	101.52x53.3mm
Weight	37g

2.1.2 Vibration Motors

As mentioned in Chapter 1, one of the main goals of this project is to make a control system of a network of vibration motors.

There are several vibration motors in the market, such as Eccentric Rotating Mass (ERM) motors (in which the weight found on the rotating motors are what produces the vibration), Brush-less Direct Current (BLDC) motors (this motor does not have brushes) and Linear Resonant Actuator (LRA) motors (a unique type of vibration motor with a performance similar to a speaker) [42].

The motors used in this project are ERM motors, more specifically Direct Current (DC) brush motors (permanent magnet and brushes).

Brushed DC motors are widely used in applications ranging from toys to push-button adjustable seats. A standard DC brush motor consists of: [43]

- Stator;
- Rotor;
- Brushes;
- Commutator.

The conceptual construction of a standard brushed motor is shown in Figure 2.2.

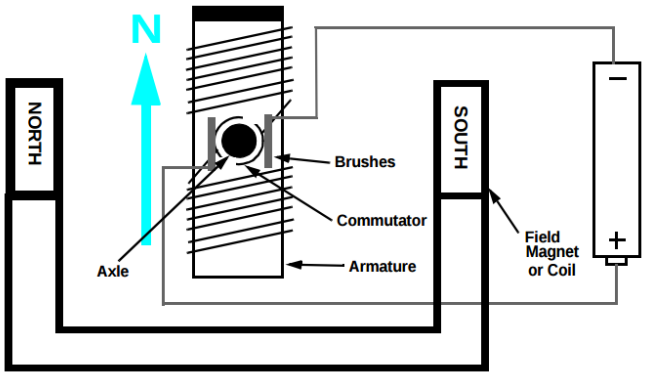

Figure 2.2: Simple Two-Pole Brushed DC Motor [43]

The ERM motors appear in different form factors, as shown in Figure 2.3.

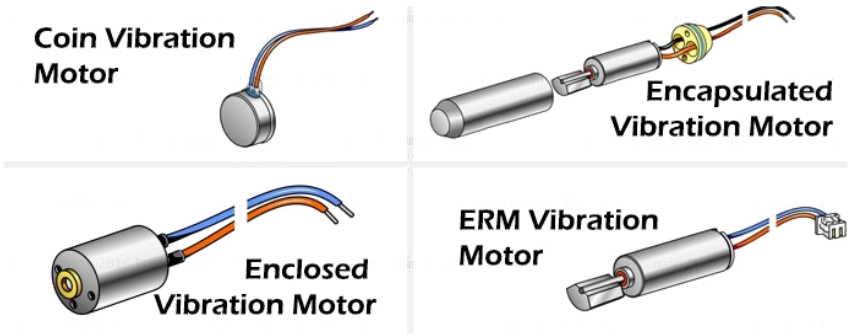

Figure 2.3: Vibration Motor Form Factors [42]

Since the "Coin Vibration Motor" form (top left on Figure 2.3 and motor on Figure 2.4) is compact and easy to use it was the chosen for this project. Table 2.2 shows some important characteristics of these motors.

Table 2.2: Mini DC Motor Characteristics [44]

Voltage	2V-5V
Diameter	10mm
Thickness	2.7mm
5V current draw	100mA
4V current draw	80mA
3V current draw	60mA
2V current draw	40mA
Weight	0.9g

Figure 2.4: Mini DC Motor [44]

Motor Operation

Figure 2.5 shows a disassembled motor.

Typically, a ERM motor is a DC motor with an offset (non-symmetric) mass attached to the shaft. In the case of the coin vibration motor, the non-symmetric mass is inside the motor capsule.

The stator inside the motor generates a stationary magnetic field that surrounds the rotor. The rotor (or armature) is made up of the windings. When these windings are energized they produce a magnetic field, and when the magnetic poles of the rotor field start to be attracted to the opposite poles generated by the stator, the rotor starts to rotate [43].

Figure 2.5: Motor Disassembled where 1 and 3 are the motor housing (3 having the brushes), 2 is the rotor and 4 is the magnet (stator)

As the motor rotates, due to the fact that the center of mass of the non-symmetric mass is not in the geometric centroid, there is a net centrifugal force, which causes a displacement of the motor. As the motor is constantly being displaced, that is perceived as vibration.

On Figure 2.6 there is a scheme by Precision Microdrives that explains how the motor is built.

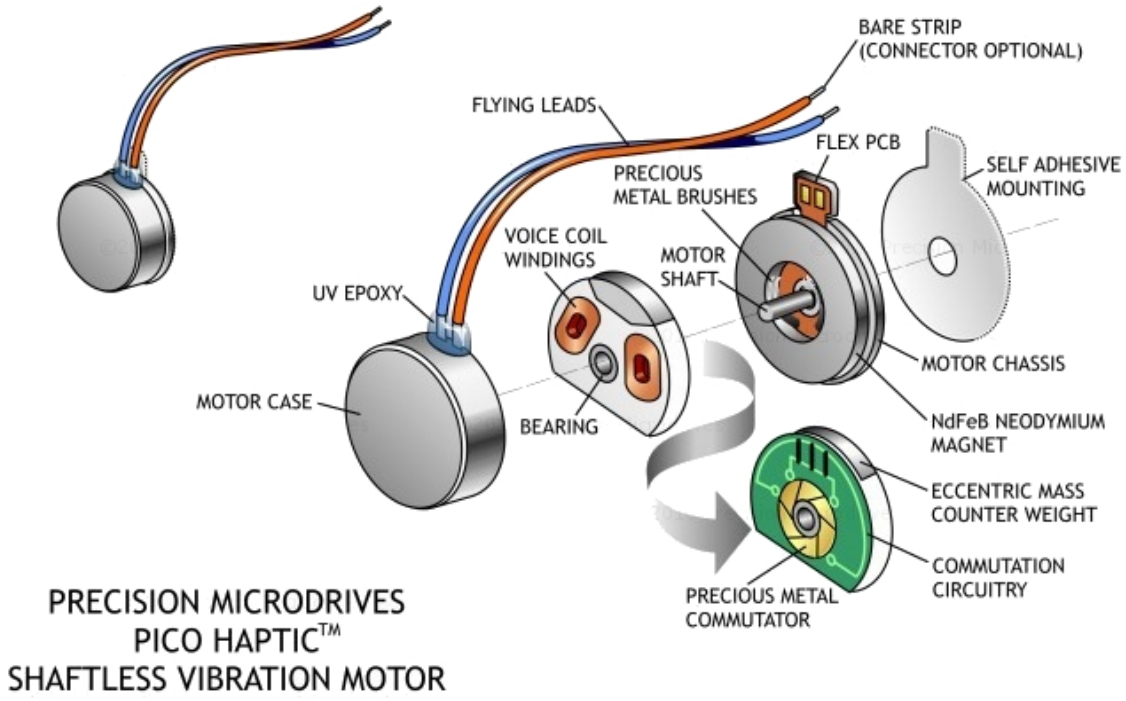

Figure 2.6: Scheme of the vibration Motor [45]

Voltage Control

A Pulse Width Modulation (PWM) signal is a type of digital waveform. It alternates between bursts of high (on/1) and low (off/0) at a fixed frequency. The difference between PWM signal and other digital signals is the fact that the time that the signal is high (and therefore low) can be varied [46].

When the pulse width is changed, what is actually being changed is the average voltage allowing any value between zero and the maximum voltage to be represented by increasing or decreasing the ON pulse width.

The PWM signal has three properties:

- A voltage amplitude, V_{PWM} - the value of the ON or high voltage level;
- A Frequency - the inverse of a period of one clock cycle (one high pulse and one low pulse);

- A duty cycle - the ratio of ON/period time.

The average voltage of a PWM wave can be calculated using equation (2.1).

$$V_{avg} = V_{PWM} \times DutyCycle \tag{2.1}$$

For instance, for a 5V PWM signal (Arduino Mega has a 5V output signal) with a duty cycle of 50%, the average output voltage is calculated according to equation (2.2).

$$\begin{aligned} V_{out} = V_{avg} = V_{PWM} \times DutyCycle \\ = 5V \times 50\% \\ = 2.5V \end{aligned} \tag{2.2}$$

This is the voltage that the motors will perceive on average if the frequency is large enough.

2.2 Software

2.2.1 ROS

Robot Operating System (ROS) is an open source software used for the development of robot applications. Its philosophy is to make a piece of software that could work in other robots by making little changes in the code. ROS was originally developed in 2007 by the Stanford Artificial Intelligence Laboratory (SAIL) and in 2008 Willow Garage continued its development. Currently a lot of companies and research institutions use ROS for the development of their projects (such as the NAO robot in Figure 2.7) [47].

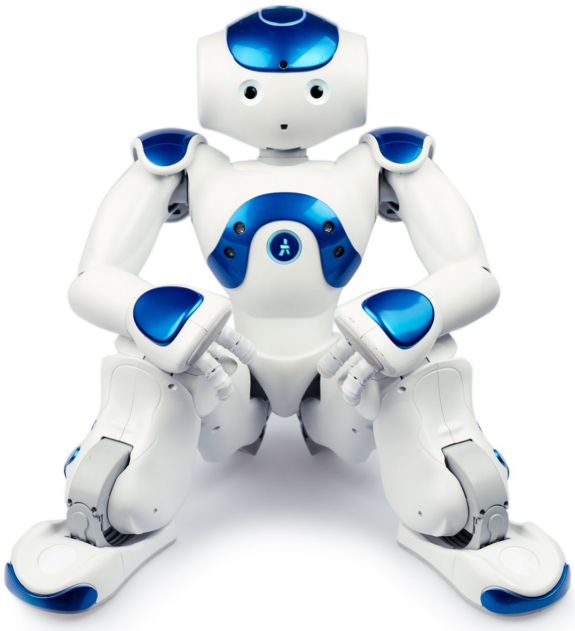

Figure 2.7: NAO Robot [48]

The sensors and actuators used in robotics have also been adapted to be used with ROS, and each day, there is an increasing number of devices that are supported by this framework.

ROS provides standard operating system facilities such as hardware abstraction, low-level device control, implementation of commonly used functionalities, message passing between processes, and package management [47].

There are several fundamental ROS concepts. The most important concepts for this project are the following:

- Nodes;
- Master;
- Messages;
- Topics.

A **node** is a process that performs computation or other operations. Nodes are combined together and communicate with each other. A robot control system usually comprises many nodes [49]. For instance, if there is a camera in a robot and the goal is to have the image from the camera both on the robot itself and on a computer screen, then there should be at least three nodes: A Camera node that deals with the communication with the camera, a Image Processing node on the robot that processes image data, and a Image Display Node that displays images on a screen [50]. A ROS node is an executable compiled with the written code. This code can be written in C++ or Python [7].

The ROS **Master** provides naming and registration services to the rest of the nodes in the ROS system. The role of the Master is to enable individual ROS nodes to locate one another so they can communicate with each other peer-to-peer [51].

Figure 2.8 shows how these two concepts are related.

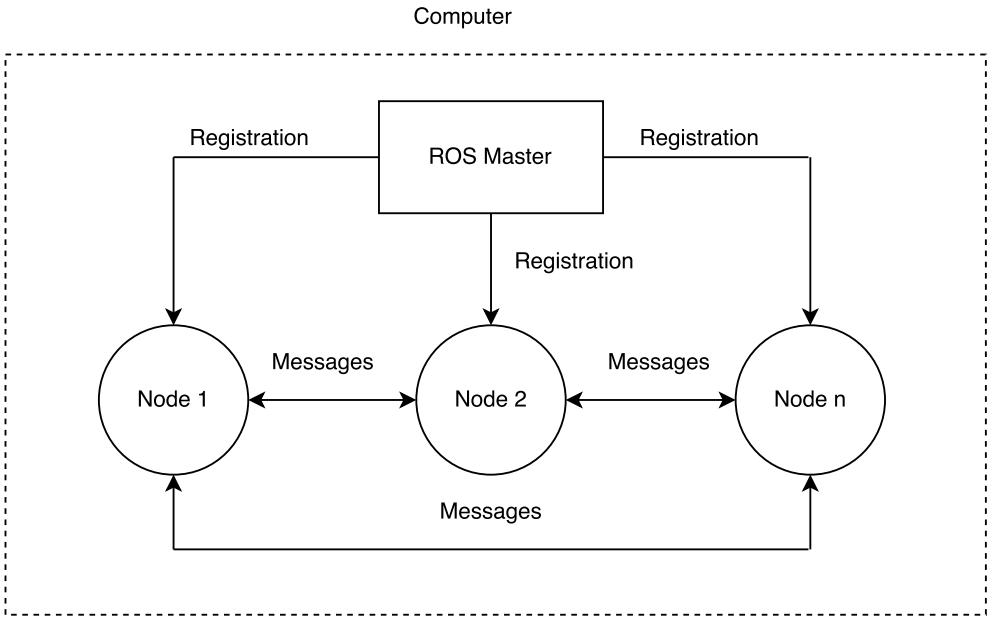

Figure 2.8: ROS Master and ROS Nodes relationship (adapted from [50])

To communicate with each other, nodes publish **messages**. A message is a simple data structure, comprising typed fields. Standard primitive types (such as integer, floating point, boolean, etc) are supported, as well as arrays of primitive types. Custom messages defined by the user are also allowed [52].

To send a message, the ROS node publishes it to a given **topic**. The topic is a name that is used to identify the content of the message [7]. Nodes that are interested in data subscribe to a relevant topic, while nodes that generate data publish it. There can be multiple publishers and subscribers to a topic [53].

Figure 2.9: Two ROS nodes communicating with each other over a ROS Topic

Figure 2.9 illustrates how two nodes communicate with each other over a ROS topic (Topic 1). The Node 1 is publishing information on Topic 1, while Node 2 subscribes to it to receive that information.

The main advantage of using ROS is that nodes in ROS do not have to be on the same system (multiple computers) or written on the same language. This makes ROS very flexible and adaptable to the needs of the user [54]. That is the main reason why ROS was chosen over, for example, the creation of a shared memory. The communication between nodes can be done through messages or services. In this work, the method used is based on messages. For this, a publisher and a subscriber to a topic must be created, in which the first publishes the message as the second receives it and acts according to it.

2.2.2 Arduino Development Environment

The Arduino controller comes with its own Integrated Development Environment (IDE): the Arduino Integrated Development Environment. It contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them [55].

The programming language of the Arduino is C/C++.

2.2.3 ROS and Arduino

To make the connection between ROS and the Arduino, Rosserial was used. Rosserial is a protocol for wrapping standard ROS serialized messages and multiplexing multiple topics and services over a character device such as a serial port or network socket [56].

The reason why Rosserial was chosen over a simple serial connection is because it provides a communication protocol to work over the Arduino's Universal Asynchronous Receiver/Transmitter (UART), allowing the Arduino to work as a ROS node that can publish and subscribe to ROS messages.

The Rosserial protocol is compiled just like any other ROS package and it contains several Client Libraries. The one used in this project is the `rosserial_Arduino`.

To be able to make the Arduino communicate with ROS, one must run the `rosserial_python` package and the `serial_node.py` node, mentioning the correct port where the Arduino is connected, just like the following, where `ACM0` is the port where the Arduino is connected [57]:

```
rosrun rosserial_python serial_node.py /dev/ttyACM0
```

2.2.4 Timers library in Arduino

A timer is an internal counter of the μC that controls the sequence of an event by counting in fixed intervals of time. It is used for producing precise time delay and measuring time events [58]. It can be programmed by some special registers.

For the purpose which is going to be explained in Section 3.1, for this project the use of timers and their features was necessary.

The Arduino Mega 2560 Central Processing Unit (CPU) is the Atmel AVR ATmega2560. As it can be seen in its datasheet [59], Arduino Mega has 6 timers: Timer 0, Timer 1, Timer 2, Timer 3, Timer 4 and Timer 5, with all of them being 16 bits except Timer 2 which works with 8 bits. The biggest difference between a 8 bit and 16 bit timer is the timer resolution, as 8 bit means 256 values while 16 bits means 65536 values for higher resolution or longer counter [60].

Since timer 0 is used for the software sketch functions like `delay()`, `milis()`, `micros()` and timer 2 only has 8 bits, those could not be used in this project.

For this project two timers were needed. For reasons that will be explained in Section 3.1, timer 1 and timer 4 were selected and configured.

Obviously that configuration could be done by changing the timer registers by hand. However, there are already some libraries that do that. For this project, those libraries are *TimerOne* [61] and *Timer4* [62].

This libraries have the following functions [61]:

- `initialize(period)`: it needs to be called before any other. It initializes the timer with a desired period in μs (by default it is set at 1s);
- `setPeriod(period)`: sets the period in μs;
- `pwm(pin, duty, period)`: generates a PWM waveform on the specified pin;
- `setPwmDuty(pin,duty)`: sets the PWM duty cycle for a given pin if it was already set by calling *pwm*() earlier;
- `attachInterrupt(function,period)`: calls a function at the specified interval in microseconds;
- `detachInterrupt()`: disables the attached interrupt;
- `disablePwm(pin)`: turns PWM off for the specified pin;
- `read()`: reads the time since last rollover in microseconds;
- `start()`;
- `stop()`:

- `restart()`.

In this project, the most used functions were `initialize()` and `attachInterrupt()`. The minimum period or highest frequency this library supports is 1μs or 1MHz.

2.2.5 Qt

To develop the Graphical User Interface (GUI) Qt was chosen. Qt is a cross-platform application development framework for desktop, embedded and mobile[63] [64]. It is not a programming language but instead a framework written in C++. A preprocessor (Meta-Object Compiler (MOC)) is used to extend the C++ language with features like signals and slots (mechanism used for communication between objects).

Before the compilation, the MOC parses the source files written in Qt-extended C++ and generates standard compliant C++ sources from them.

Although any build system can be used with Qt, it brings its own `qmake` (that automates the generation of Makefiles). Even though `qmake` is the most used build system, `CMake` is also a popular alternative to build Qt projects.

Qt comes with its own IDE named Qt Creator.

With Qt, GUI can be written directly in C++ using its Widgets module. Qt also comes with an interactive graphical tool (Qt Designer) which works as a code generator for Widgets based GUI. It can be used stand-alone but is also integrated into Qt Creator.

The fact that Qt has its own IDE and that it comes with the Qt Designer tool was the biggest factor to choose Qt over other GUI softwares.

2.3 Components Integration

Figure 2.10 shows how the components that were previously mentioned in this chapter integrate the solution.

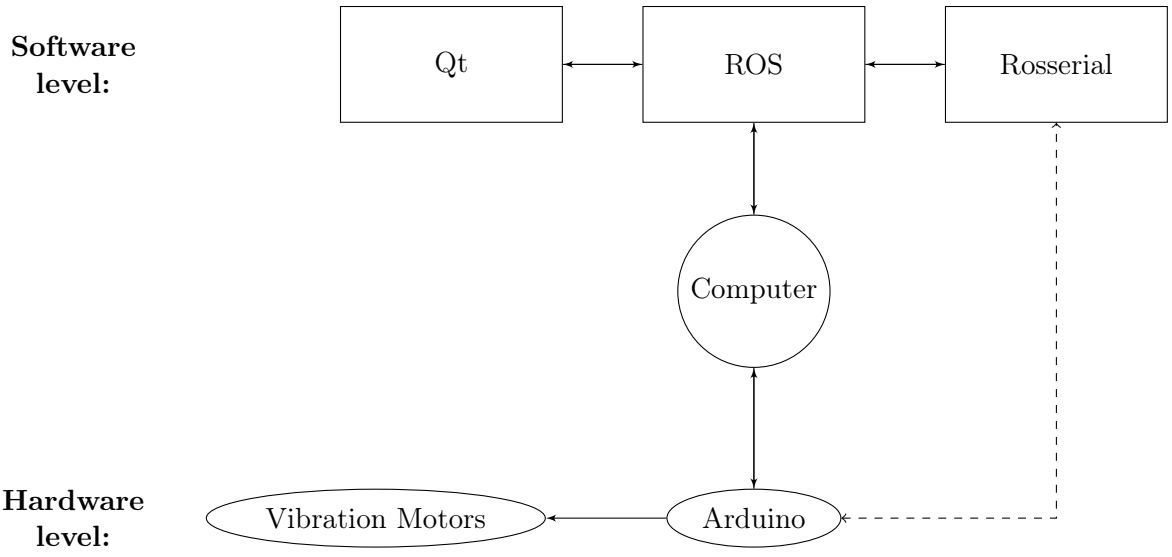

Figure 2.10: Diagram of all the components at the current development phase.

In this project the user can control the motors using a computer a graphical interface designed with Qt. Using ROS, the information given to the interface by the user is sent to the Arduino using the Rosserial protocol. The Arduino then proceeds to activate the vibration motors.

However, the system must be able to operate independently of other hardware/software. For instance, the μC doesn't necessary need to be Arduino, the communications can be done using bluetooth, the GUI can be designed using another software and the user can, for example, control the GUI via a smartphone and not a computer.

Chapter 3

Development of a Wearable Haptic Device

This chapter describes all the stages needed to develop a wearable haptic device. It explains the chosen location for the motors as well as the process of building a prototype. In this project the intensity, the location and the duration of the stimulus can be controlled and the type of stimulus can be classified as vibration. The first thing done was the generation of PWM signals using the Arduino and its timers/interrupts. Secondly, the graphical interface for the motors actuation and the programming of the stimuli was created. At last, the wearable device — a sleeve — with all the components integrated was developed.

Figure 3.1 shows a flowchart of the three ROS nodes that were created to achieve that goal.

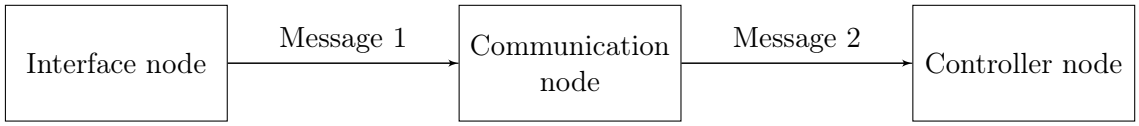

Figure 3.1: ROS nodes flowchart

The Interface node is a publisher node that sends information to the Communication node. This information is a message that contains the number of motors to turn on, their duty-cycle and their location (Message 1). In this node, the user can interact with the computer through a graphical interface and choose which motor to turn on and its intensity.

The Communication node connects the Interface and Controller node, being both a publisher and a subscriber. It receives the message from the Interface node, converts the variables (more will be explained in detail in Section 3.4.2), thereafter sending the duty-cycle and the location of the motors to the Controller node (Message 2).

At last, the Controller node acts as a subscriber. It receives Message 2 from the Communication node and then proceeds to activate the correct motors with the correct duty-cycle.

Figure 3.2 shows the ROS nodes layout made with the `rqt_graph` software where `qtguinode` is the interface node, `publish_arduino` is the communication node, `serial_node` is the controller node, `guichattergui` is the topic where interface node is publishing in

and `chatter2` is the topic where the communication node is publishing.

Figure 3.2: ROS nodes layout made with `rqt_graph`

3.1 PWM Motor Control Using Arduino

As stated in Section 2.1.1, Arduino Mega has only 15 PWM outputs. Since in this project there are 16 vibration motors, those 15 outputs were not enough to control all the motors.

To control all the motors individually, several solutions were thought about such as the use of multiplexers or decoders. Multiplexers has the limitation of only being able to turn ON one motor at a time or all of the motors together. The use of the decoders would oblige the use of several cables, having always the problem of finding a way to control the supply voltage of every motor to change their intensities. The use of a band pass filter was considered, however a simpler solution was found: to manually generate a PWM wave using the Arduino timers and their interrupts. Using a PWM wave the supply voltage of each motor could be controlled using the PWM duty cycle.

3.1.1 PWM manually generated

The idea to generate the PWM wave is to have an interrupt generating the HIGH-/LOW signal. An interrupt is a periodic event that interrupts the main program code and runs a special function known as Interrupt Service Routine (ISR). After the ISR has been finished, the running program continues [60]. According to the Arduino Mega data-sheet, the priority of the different timers is different [59]. The priority order of the timers is expressed below.

1. Timer 2 (bigger priority);
2. Timer 1;
3. Timer 0;
4. Timer 3;
5. Timer 4;
6. Timer 5 (smaller priority).

Having in mind that two timers were needed, one with bigger priority to turn ON the motors and the other to turn OFF the motors, and due to the fact that timer 2 is a 8 bit timer so it should not be used, timer 1 and timer 4 were selected.

- Timer 1: bigger priority and slower timer with a frequency of 1kHz (period of 1ms);
- Timer 4: faster timer with a frequency of 10kHz (period of 0.1ms).

The slowest timer turns ON the desired motors and the faster timer will turn OFF the motors according to the duty cycle previously decided by the user. Figure 3.3 shows the timing diagram for 3 outputs. For example, if the duty cycle percentage of motor 1 is 50%, motor 2 is 10% and motor 3 is 90%, the slowest timer will turn ON the 3 motors. Since the slowest timer is 10 times slower than the fastest timer, in the first call of the fastest timer ISR, motor 2 (10% duty cycle) turns OFF, in the fifth time slot of the ISR motor 1 (50% duty cycle) turns OFF and finally, on the ninth time slot of the fastest ISR motor 3 (90% duty cycle) turns OFF.

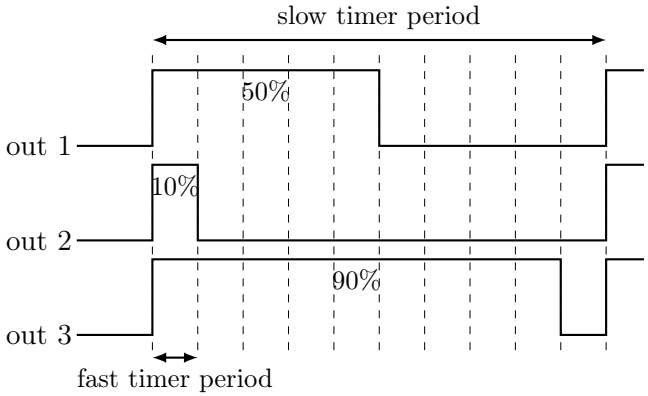

Figure 3.3: Timing diagram for 3 PWM outputs

As mentioned before, two ISR were created to enable the creation of the PWM wave manually and turn the motors on and off with the correct intensity. Every time timer 1 reaches its timeout, the interrupt function checks whether or not the motor should be on. If the motor is supposed to be on the value HIGH is given to the correspondent motor pin. Figure 3.4 illustrates the Interrupt Subroutine related to timer 1.

When timer 4 reaches its deadline, it verifies if the desired duty cycle (intensity) has been reached or not, turning the motors off if it is supposed to (giving the motor pin the value LOW). Figure 3.5 shows the ISR related to timer 4.

Using this method there can be as many motor connected as the number of output Arduino has. In this case, since Arduino Mega has 54 digital outputs, there can be 54 motors connected.

Due to the fact that timer 1 is 10 times slower than timer 4, the PWM duty cycle can assume values from 0 (0%) to 10 (100%).

Figure 3.6 shows the comparison between a PWM with a duty cycle of 50% using a standard PWM from Arduino Mega (function *analogWrite()*) in blue and a PWM wave with a duty cycle of 50% created with this timer/interrupts method in yellow.

As Figure 3.6 shows, the generated wave is similar to a standard Arduino PWM wave, which proves that this timer/interrupt method can be used.

Figure 3.6 was taken using an UNI-T Digital Storage Oscilloscope, model UT3102C available at the Laboratory for Automation and Robotics of the University of Aveiro (LAR). This oscilloscope has the capability of taking a picture of the screen, saving the image on a USB stick formated in File Allocation Table (FAT). This image is then saved in a Bitmap image file (BMP) file. More information regarding this process can be found in Appendix A.

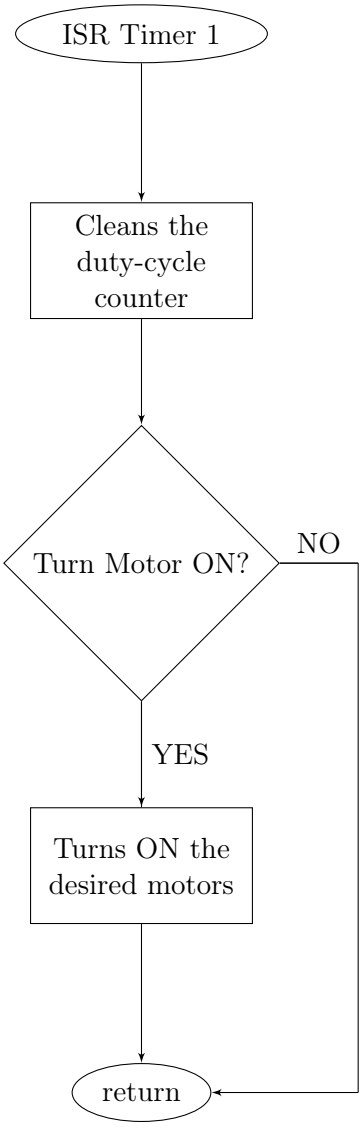

Figure 3.4: Scheme of the Interrupt Subroutine Timer 1

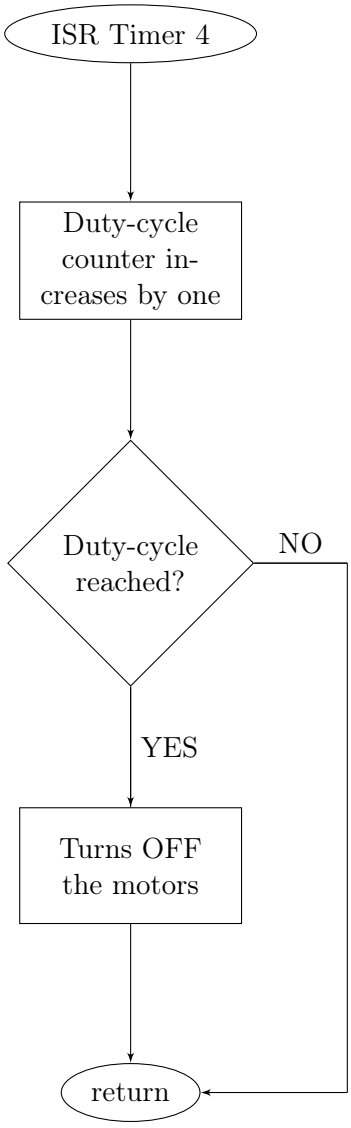

Figure 3.5: Interrupt Subroutine: Motors OFF

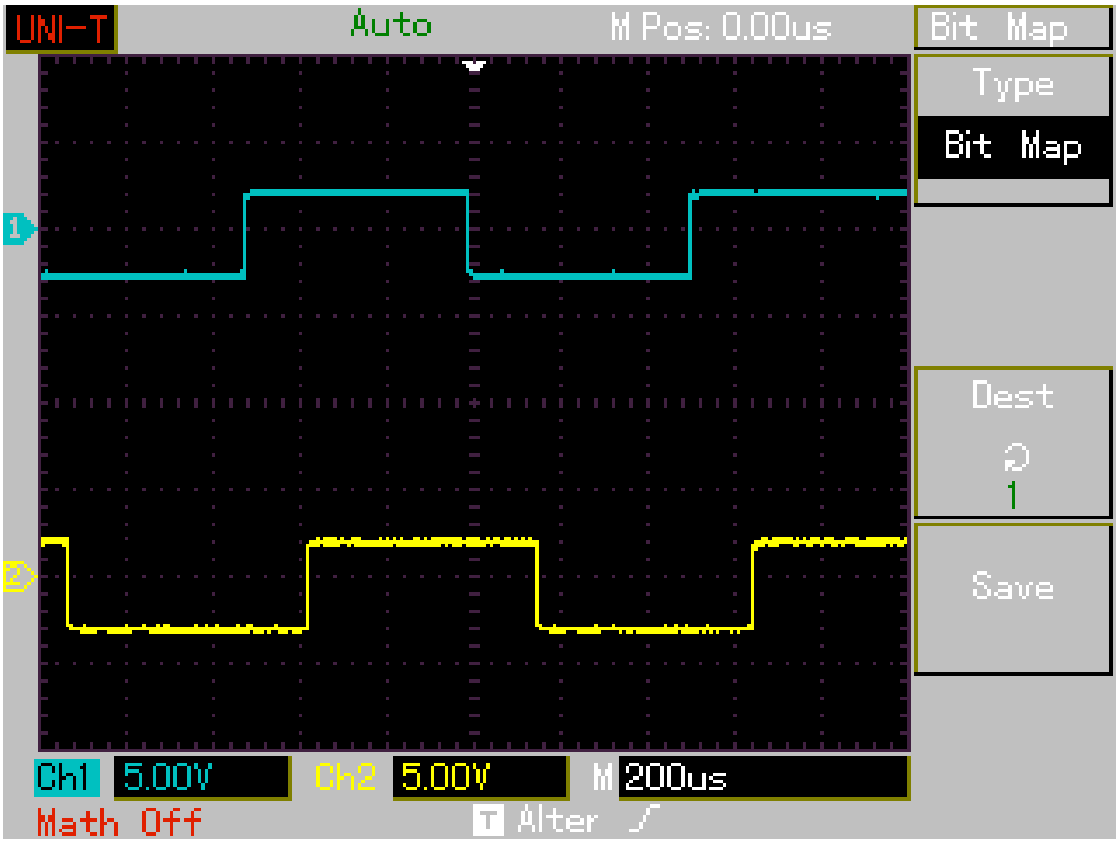

Figure 3.6: PWM wave with 50% duty cycle comparison. In blue is the PWM generated using the standard Arduino method and in yellow is the PWM generated using the described method above

3.1.2 Connection to the multiple motors

In this project the 16 motors are connected to the Arduino as the schematics in Figure 3.7 demonstrates.

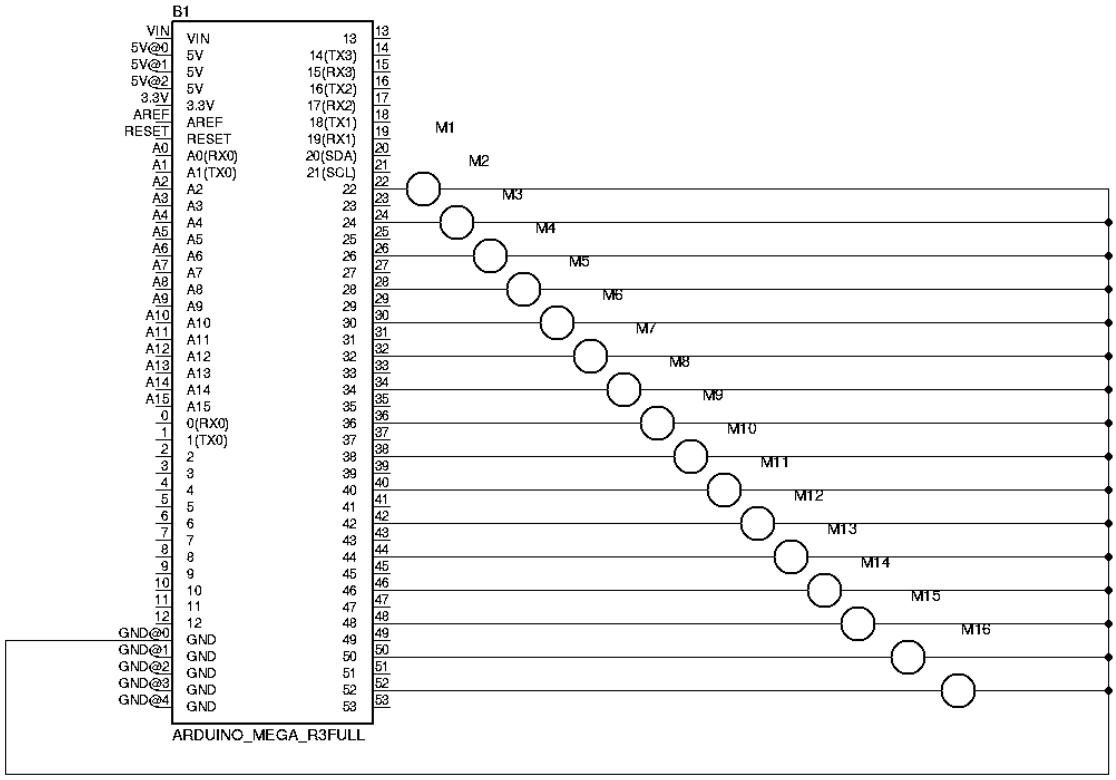

Figure 3.7: Electrical Schematics with the connections of μC to the 16 motors

To allow an easier connection of the motors to the controller a shield was developed using the Eagle Computer Aided Design (CAD) software as shown in Figure 3.8. In this shield, 16 Arduino digital outputs are connected to one side of a 20 pins pinhead and to the other side of the pinhead there is a line of grounds as Figure 3.9 illustrates.

The Printed Circuit Board (PCB) (Figure 3.10) was then printed at the Department of Electronics, Telecommunications and Informatics (DETI) of the University of Aveiro. The final result of this shield already with the soldered components can be seen in Figure 3.11.

Connected to the shield is a flat cable with a connector that fits into the pinhead (Figure 3.12). The motors are then connected to the other side of the flat cable as Figure 3.13 illustrates. One of the advantages of the vibration motors used is that their polarity does not matter. In other words,the way they are connected to the pinhead is not important because there is no positive/negative side on the motor.

Since the current consumed by these mini vibration motors is so small, no power drivers were used in this project. However, in a more advanced system the use of transistors or MOSFETs should be considered.

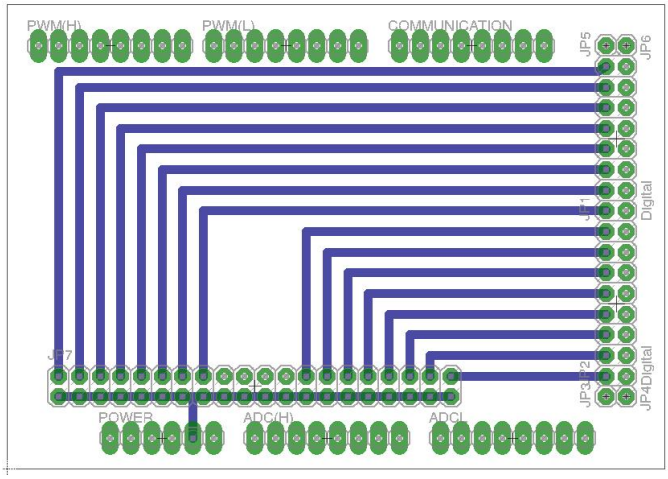

Figure 3.8: Eagle board design

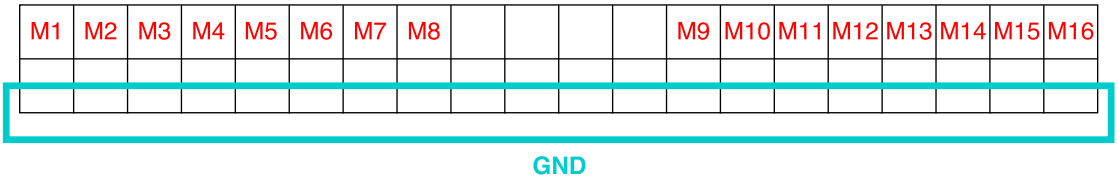

Figure 3.9: Pinhead connections

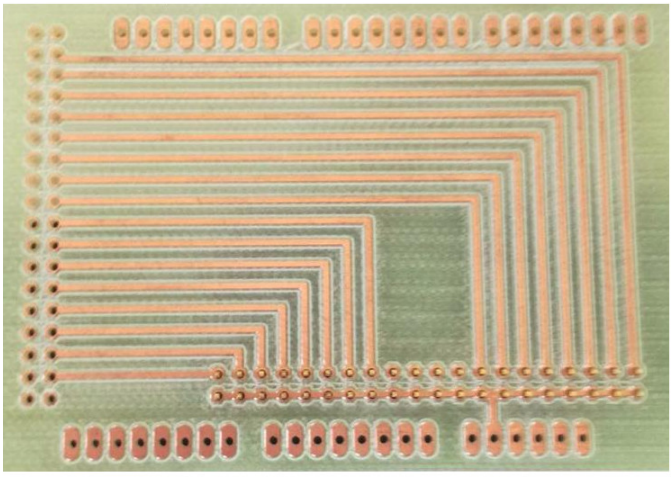

Figure 3.10: Printed shield

Figure 3.11: Shield installed to the Arduino Controller

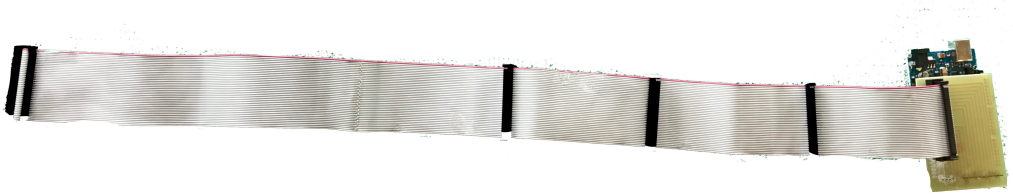

Figure 3.12: Flat cable with shield attached to the right

Figure 3.13: Connection of the motor cables to the pinhead of the flatcable

3.2 Control System for Multiple Motors

In Section 2.1.1 Arduino Mega was defined as the controller for this project and the controller node was identified as a subscriber of the communication node. The message received is the intensity and the motors number (location), meaning, what motors to turn ON.

The location of the motors can be considered a set of objects. There are two forms to represent which objects to pick:

- Map (array of values);
- Bitmask.

The Map associates with each object a boolean value, indicating whether the object is picked or not. However, this method takes up a lot of memory and it can be slower to transmit. Therefore a bitmask is more efficient since there is less overhead [65].

A mask defines which bits to keep and which bits to clear and it is used for bitwise operations. Masking (the act of applying a mask to a value) can be accomplished by [66]:

- Bitwise ORing in order to set a subset of the bits in the value;
- Bitwise ANDing in order to extract a subset of the bits in the value;
- Bitwise XORing in order to toggle a subset of the bits in the value.

The mask that defines which motors to turn ON or OFF is created in the Communication Node and it is called `motorMask`. This mask is generated using the following code (Figure 3.14) where the `motorsNumber` array is an array of 1s and 0s (1 if the motor is selected, 0 if the motor is not) and the `TOTALMOTORS` is 16.

```
for(int totalMotors=0;totalMotors<TOTALMOTORS;totalMotors++)
{
    uint8_t motorMask |= motorsNumber[totalMotors] << totalMotors;
}
```

Figure 3.14: Creation of the bitmask that represents the motors to turn ON

For instance, if the motors to turn on are motor 1, motor 3 and motor 5, the bit mask has the value 21.

Figure 3.15 shows the main structure of the controller program.

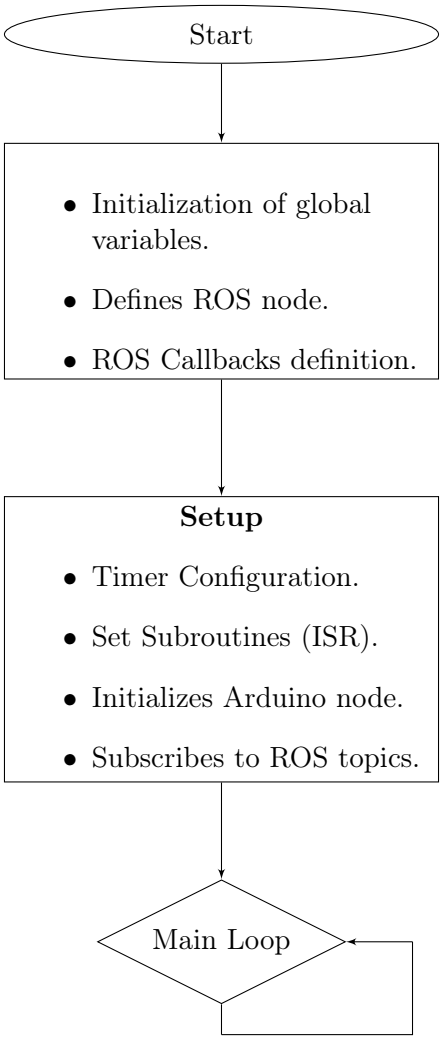

Figure 3.15: Flowchart of the program running in te Arduino unit

3.3 Vibration Motors Sleeve

3.3.1 Placement of the vibration motors

The prototype built in this project was developed to be used in the arm/hand. After medical advice, the motors were placed in strategical areas to affect the dermatomes C5, C6, C7 and C8 as shown in Figure 3.16. In the placement of the motors on those places, it was also considered that the motors should have, at least, around 2 cm of distance between them, or else their effect is hard to distinguish.

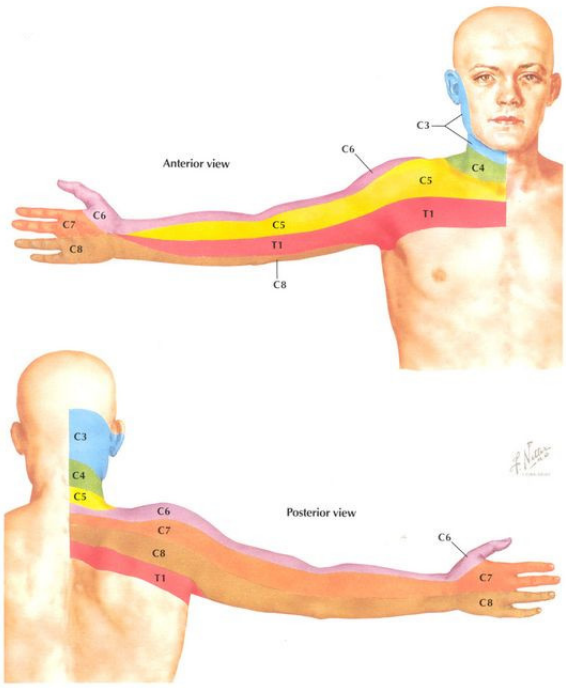

Figure 3.16: Schematic demarcation of dermatomes in upper body [37]

Considering this information, Figure 3.17 exhibits the proposed layout of the motors used in this project.

There are nine motors placed in strategical dermatomes areas in the lower part of the arm/hand. On the upper part, the other seven motors were placed in the joints for diagnose purposes, also due to medical advice.

3.3.2 Creation of the arm sleeve

The created sleeve can be seen in Figure 3.18. The sleeve was made with Lycra and it has a fastener and holes for the user fingers to facilitate the wearing.

The motors were placed as it was stated in Section 3.3.1 and then sewn into the sleeve. To ensure a more robust prototype and also for hygienic reasons (so the motors are not directly touching the user skin), nylon was used as lining as sketched in Figure 3.19. The final result of this process can be seen in Figure 3.20.

Figure 3.21 represents the final prototype.

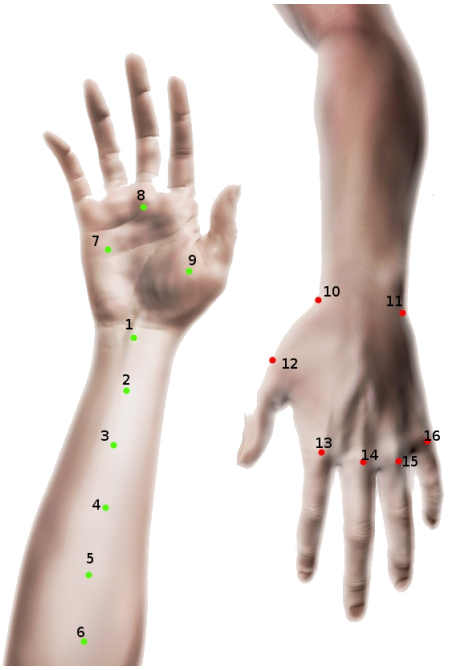

Figure 3.17: Placement of the motors (image adapted from [67])

Figure 3.18: Prototype of the Sleeve

This arm sleeve was developed for applications in robotics teleoperation or rehabilitation.

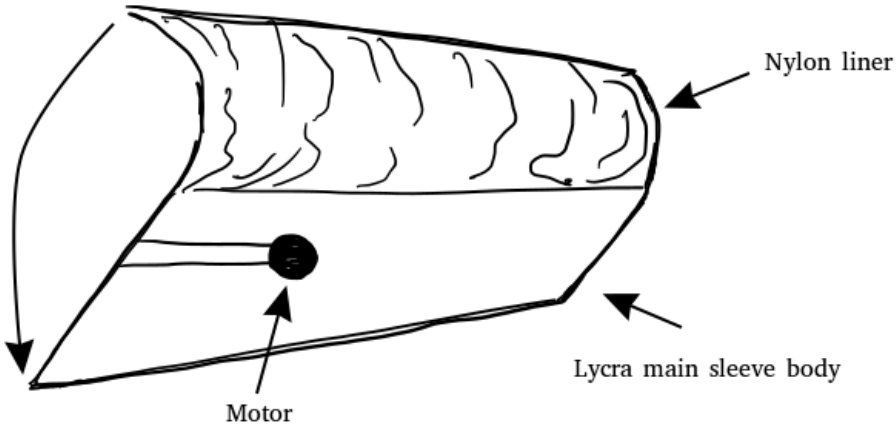

Figure 3.19: Sketch of the sleeve main components

Figure 3.20: Inside of the Sleeve

The interface that is described in the following section allows the activation of a group of motors from the sleeve and the variation of the stimulus intensity. In the current development phase, the identity of the motors and their intensity comes from the user decision, however, in the future this information can come from external factors (such as the use of force sensors positioned in the same place as the motors but in someone else's arm).

Figure 3.21: Final Prototype

3.4 Graphical user interface

Previously, the location of the motors (the motors number) and the intensity of their action were identified as the variables that the user would be able to control. To allow an easier control of these variables a graphical interface was developed.

3.4.1 Interface Program

Due to the fact that the interface is in a ROS node (`qtguinode`) a ROS package was created. This package is called `qtgui` and it is there that all the interface was built and the message was defined.

The design of the interface is made using a class that `QtDesigner` creates on its own: the `MainWindow` class. This is the class associated to the user interface.

ROS requires a large amount of information to work properly. It needs to be initialized, the ROS node needs to be named, a handler to that node needs to be created, among other things. To do so, the class `QtPublisher` was conceived inside the `qtgui` package. This class is responsible for the creation of the handler and to tell the master what type of message is going to be published and where. The topic `guichattergui` was defined as the topic where the message is published. Furthermore, the function `SendMessage` was also established. This function is responsible for broadcasting the message to any node that is connected to the topic previously defined.

Besides this class, the `qt_ros_interface` package was also created. In here, the ROS initialization (`ros::init()`) is made and the node is named.

Figure 3.22 shows the organization of the interface node.

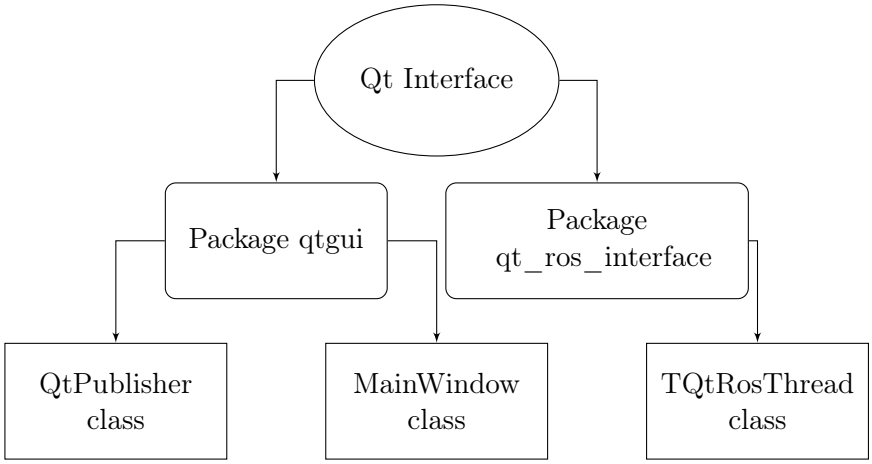

Figure 3.22: Interface organization

The message created consists of the variables that the user can control, intensity and location. This message was named `GUIDados.msg`.

```
uint8[ ] intensity
uint8[ ] location
```

Figure 3.23: Message *GUIDados.msg*

Figure 3.23 illustrates the created message. The type of variable *uint8* (unsigned 8-bit int) was used due to the fact of being the smallest variable that ROS messages works with. It is the equivalent of the variable type *char* in ROS. Both the intensity and location are an array of numbers. Since 16 motors are being used, the size given to the array is 16. However, if more motors were to be added, the size of the array should also be increased.

Figure 3.24 shows how the package *qtgui* is organized.

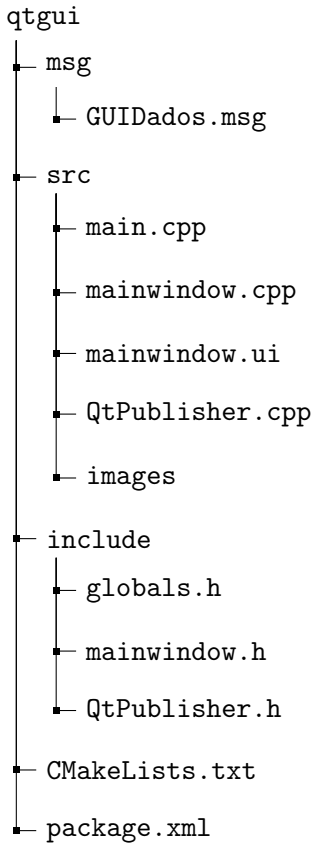

Figure 3.24: Organization of the qtgui package

ROS is built using `catkin_make` and Qt with `qmake`. The first thing done was to change the built option from `qmake` to `cmake`, so it would be easier to mix the Qt `CMakeLists` and the ROS `CMakeLists`.

Since ROS is being used in the interface node, some important things such as the `catkin_libraries` and `QT_libraries` had to be added in the `CMakeLists` file. Important packages like `Qt4` and `qt_ros_interface` were also added. The final `CMakeLists` can be found in Appendix B.

3.4.2 Communication

Being both a subscriber and a publisher, the Communication node receives the message from the Interface node and processes that information to pass it over to the Controller node.

The Communication node receives two *uint8* arrays with size 16 (intensity and location).

The intensity array is composed by numbers from 0(none) to 10(higher) and the location array is an array of 1s and 0s (1 if the motor is on and 0 if the motor is off). To make the communication and the processing faster, the size of the message is reduced. To achieve that reduction a bitmask that defines the motor behavior is created. The creation process of this mask was demonstrated previously on Section 3.2.

Regarding the location variable, the array received is an array of 16 *uint8* numbers. After the conversion (the creation of the mask), the location variable becomes an *uint8* number with 16-bits instead of an array.

The new message file created was named *Dados.msg* and it contains the same intensity array as before but a new location variable (Figure 3.25).

```
uint8[ ] intensity
uint8 location
```

Figure 3.25: Message *Dados.msg*

After the conversion these two variables are sent to the Controller node. The Communication node package organization is represented in Figure 3.26.

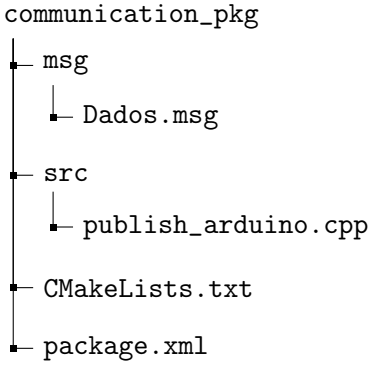

Figure 3.26: Organization of the communication package

3.4.3 User Interface

The interface offers the user two working modes:

- Motor control in which the user can select which motor (or group of motors) to turn ON and the intensity;
- Pattern control, in which the user can select from 3 standard stimulation pattern.

When the interface is launched, the operating mode that appears on the screen is the Motor control (Figure 3.27). In this mode the user can select which motor to turn on and its intensity. It allows more than one motor functioning at the same time and it also has the option to activate and deactivate all the motors.

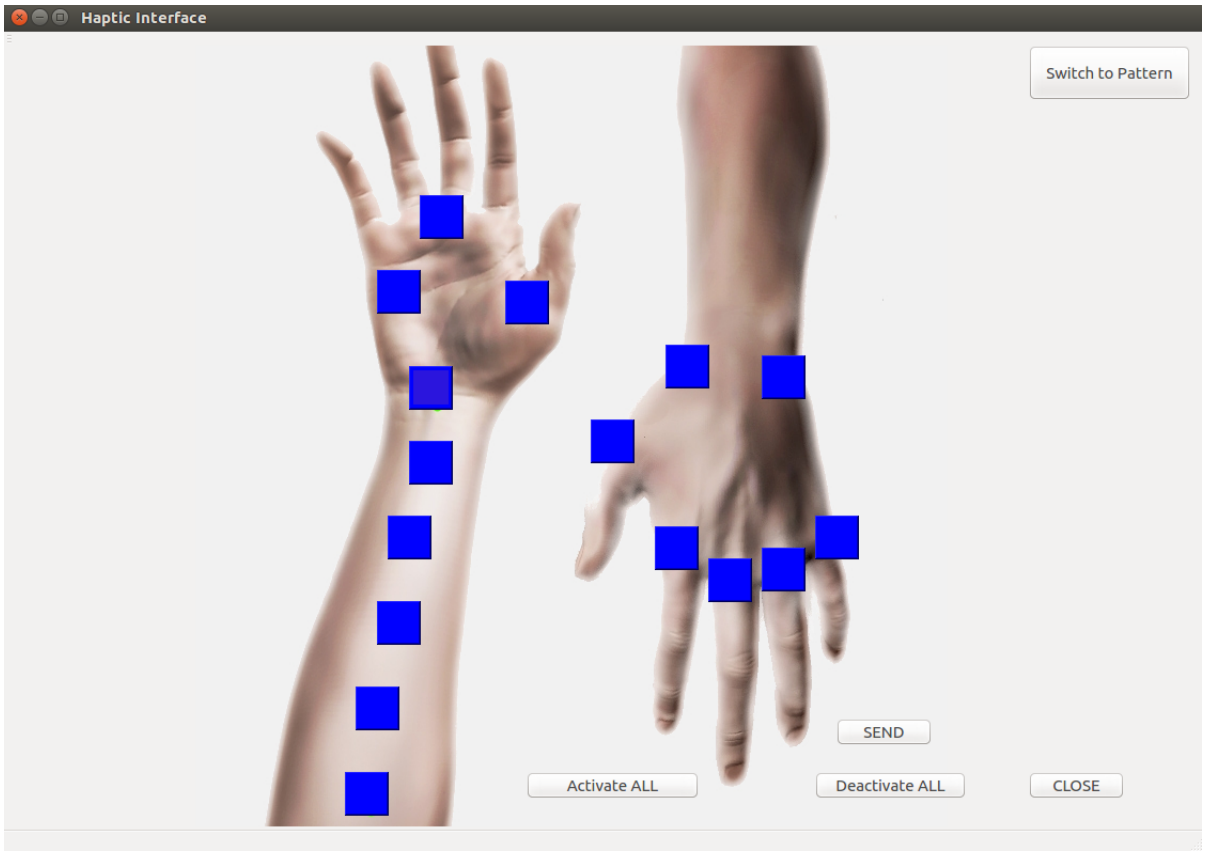

Figure 3.27: Interface: Motor Control

Initially the intensity selection box is not visible, however, every time a motor is selected (button is pressed) that option appears as Figure 3.28 illustrates. The same happens with the activate all option. When the "Activate ALL" button is clicked, all the motor's intensity selection box disappears and one general one emerges (therefore the chosen intensity is the same for every motor) as Figure 3.29 shows.

Another aspect in the motor control mode is the fact that, when the motors are OFF (buttons unselected) the buttons are blue and when the motors are ON the buttons turn red. This gives the user a visual help and was done to make the interface more user friendly.

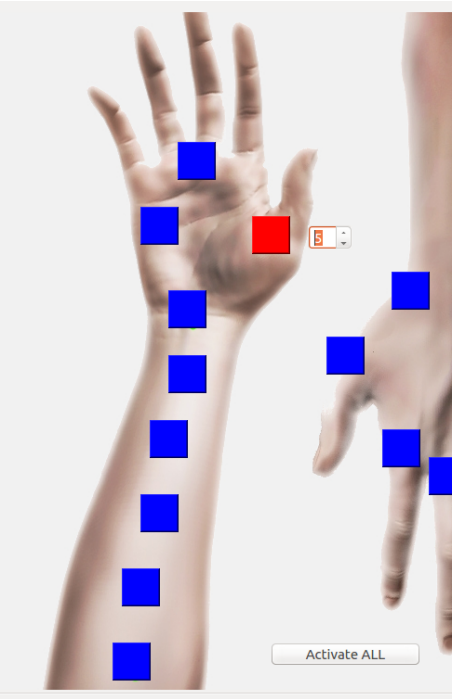

Figure 3.28: Intensity selection box of motor 9

Figure 3.29: Intensity selection box of all motors

To confirm which motors to turn on and their intensity the button "SEND" must be pressed. This will send the created message to the created topic as explained before.

The "Deactivate ALL" button, just like the name indicates, turns OFF all the motors.

When the button "Switch to Pattern" is pressed the window changes and so does the operation mode.

For an easier comprehension, Table 3.1 shows all the buttons available in the motor control operation mode (as well as the selection boxes) and their functionality.

Table 3.1: Motor Control Buttons

Button	**Action**
Motor n	Shows the intensity selection box corresponding to motor n
SEND	Sends the intensity and the location array to the controller
Activate ALL	Shows the intensity selection box corresponding to all the motors
Deactivate ALL	Turns OFF all the motors
Motor n selection box	Allows the user to select the intensity (0-10) for motor n
Activate ALL selection box	Allows the user to select a general intensity for all the motors
Close	Closes the GUI window
Switch to Pattern	Changes the GUI window to the Pattern Control

In the pattern option (Figure 3.30) the user can choose from 3 standard stimulation pattern:

- Shiver;
- Tickle;
- Diagnose.

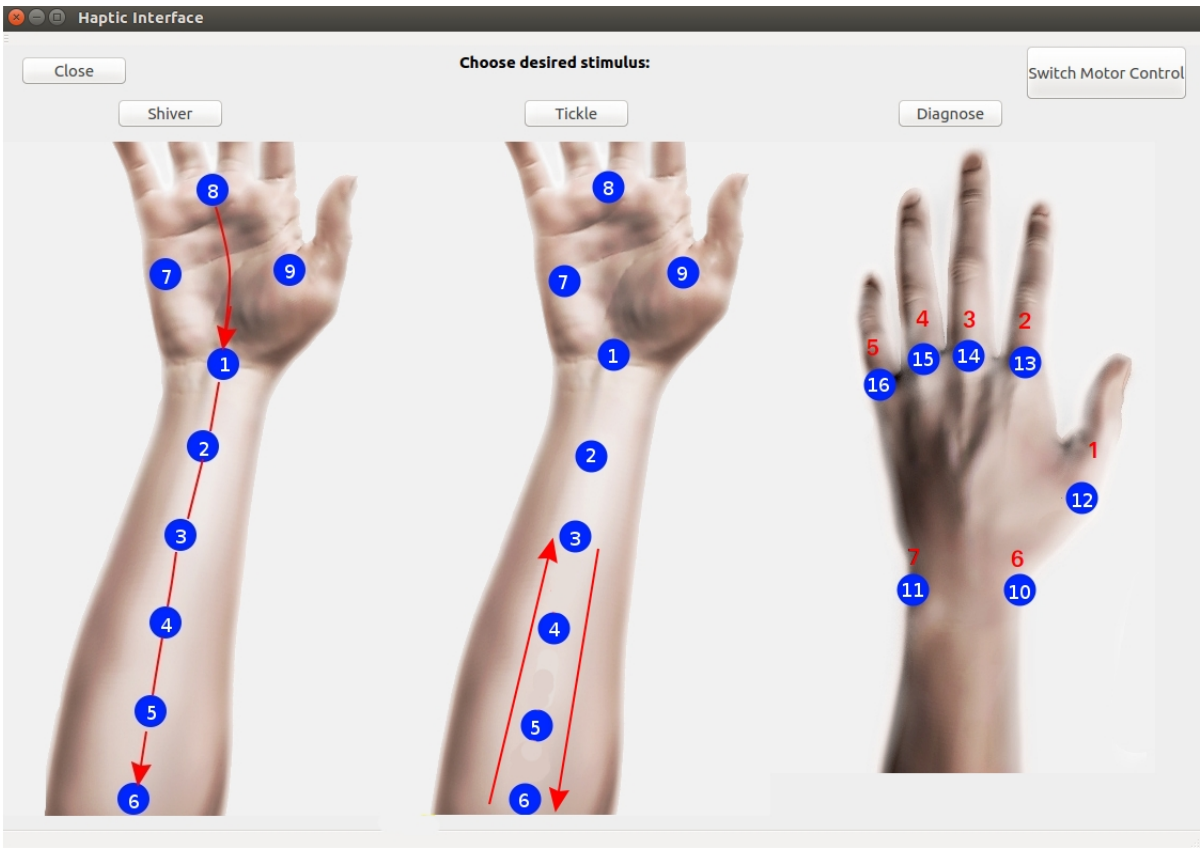

Figure 3.30: Interface: Pattern

The shiver stimulus consists on turning the motors ON from the bottom to the top. That is done turning motor 8 on, followed by motor 1, 2, 3, 4, 5 and 6 like Table 3.4 shows. The activation of the motors have an interval of 0.5 seconds. After turning one motor on, the previously activated motor is turned off so the shiver sensation is produced, like someone is moving a finger through the user arm.

Table 3.2: Shiver pattern table

Motor Number	Intensity	Time ON
8	6	0.5s
1	6	0.5s
2	6	0.5s
3	6	0.5s
4	6	0.5s
5	6	0.5s
6	6	0.5s

The tickle stimulus (Table 3.3) dwells on the back and front movement. It starts on motor 6, goes to motor 5, 4 and 3 and then goes back to 4, 5 and 6 with a delay of 0.5s as well.

Table 3.3: Tickle pattern table

Motor Number	Intensity	Time ON
6	6	0.5s
5	6	0.5s
4	6	0.5s
3	6	0.5s
4	6	0.5s
5	6	0.5s
6	6	0.5s

The last pattern stimulus created is the diagnose. For this stimulus, the motors used are the motors placed in the user joints. Those are motor 10, 11, 12, 13, 14, 15 and 16.

When the user selects the "Diagnose" button an option for the delay between the motors and the intensity pops up. That delay can go from 1s up to 5s and the intensity can be either LOW (intensity has the value 3), MEDIUM (intensity has the value 6) or HIGH (intensity has the value 10). After pressing the "OK" button, motor 12 is activated, followed by motor 13, 14, 15, 16, 10 and 11. The delay and intensity selection boxes were added in the interface because, for medical diagnose, the time between the motors activation is important, and so is the different intensities. For instance, a higher intensity is easily felt by everyone yet a lower intensity is not.

Table 3.4: Shiver pattern table

Motor Number	**Intensity**	**Time ON**
12	Chosen by the user	Chosen by the user
13	Chosen by the user	Chosen by the user
14	Chosen by the user	Chosen by the user
15	Chosen by the user	Chosen by the user
16	Chosen by the user	Chosen by the user
10	Chosen by the user	Chosen by the user
11	Chosen by the user	Chosen by the user

To go back to the motor control option the user has to press the "Switch to Motor Control" button.

Table 3.5 shows all the buttons available in the pattern control operation mode and their functionality.

Table 3.5: Pattern Control Buttons

Button	**Action**
Shiver	Sends the variables corresponding to the shiver pattern to the controller
Tickle	Sends the variables corresponding to the tickle pattern to the controller
Diagnose	Shows the delay selection box and the intensity buttons
Delay selection box	Allows the user to select the desired delay (1s-5s)
Intensity selection box	Allows the user to select the desired intensity (LOW/MEDIUM/HIGH)
OK button	Sends the variables corresponding to the diagnose pattern to the controller
Close	Closes the GUI window
Switch Motor Control	Changes the GUI window to the Motor Control

The process of associating the buttons and selection boxes of the interface and the message to be sent is done using callbacks. Every time a button in the interface is clicked a callback is called giving values to the variables in the message.

If a given motor is selected, then in the location array its value is going to be 1. If a motor is not selected then its value is 0. The same is done for the intensity array. The value of the intensity selection box is given to the intensity array in the motor position. For instance, if motor 1 button is clicked and an intensity of 5 is selected, then the variables are $intensity[0] = 5$ and $location[0] = 1$. When the "Send" button is pressed the `SendMessage()` explained previously is called and the message is then sent to the defined topic.

3.5 Alternative Approach

Sometimes, due to the fact that a lot of information is being sent between the nodes, Rosserial loses the connection and the system also has some delay. With the aim of solving this problem, the Communication node was eliminated, leaving only the Interface and the Controller node, as shown in Figure 3.31.

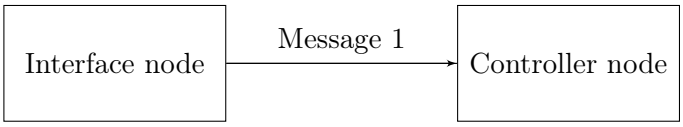

Figure 3.31: New ROS nodes flowchart

In this alternative approach the information from the interface goes straight to the controller, and the mask is created in the controller itself instead of in the Communication node. The message sent from the Interface node to the Controller node is only the duty cycle and the location of the motors (instead of the motor mask), and this message in sent only when some change on the interface occurs. Besides making the system faster, this also allows for, in the future, the ROS platform to be dropped.

Figure 3.32 shows the new ROS nodes layout made with the `rqt_graph` software, where `qtguinode` is the interface node, `serial_node` is the controller node and `guichattergui` is the topic where the message is published.

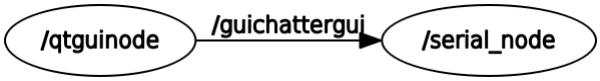

Figure 3.32: New ROS nodes layout made with *rqt_graph*

Figure 3.33 illustrates the new main structure of the controller program.

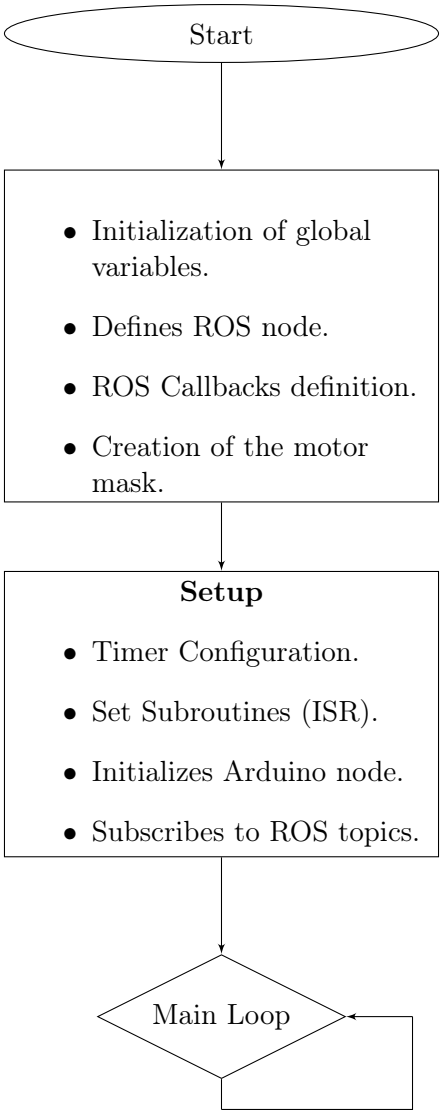

Figure 3.33: New flowchart of the program running in te Arduino unit

Chapter 4

Experiments and Results

4.1 Plan of Experiments

To evaluate the success of the prototype experiments were made and a group of people was invited to participate in the trials of the haptic sleeve.

The test is divided in three parts: Test of the motor control, of the pattern control and of the different intensities. The following plan was executed.

4.1.1 Test of the Motor Control

After the subject is comfortably sitting down with the sleeve in their arm, the first thing to test is the Motor Control part of the interface.

1. Activate motor 8 with an intensity of 8. Ask the user if anything is felt and where. Take note if the user said the correct place. This will show if the user can feel the vibration and introduce the user to the feeling.

2. Deactivate motor 8 and activate motor 7 and 9 simultaneously with the different intensities, such as 4 and 9, respectively. Ask if the user can differentiate the motors and their intensities, and if yes, which one does the user think is more intense?

3. Deactivate all the motors and activate motor 2, 5 and 6 with the same intensity such as 6. Can the user spot the different motors? If the user can not, which ones are complicated to distinguish?

4. Deactivate all the motors and activate motor 13 and 15 with the same intensity. Can the user locate them?

5. Deactivate motor 13 and 15 and activate motor 8. Can the user distinguish and say which motors are on?

For an easier comprehension, Figure 4.1 is a state diagram that illustrates the test.

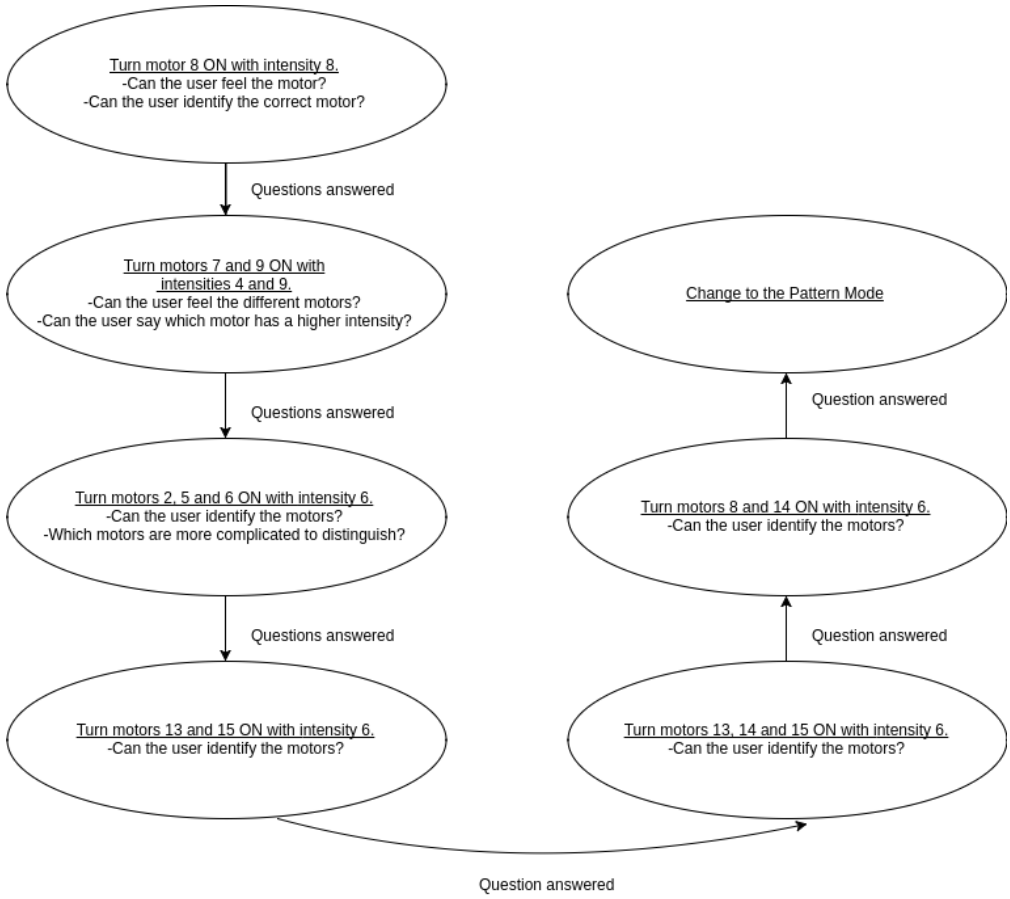

Figure 4.1: State Diagram of Motor Control Test

4.1.2 Test of the Pattern Control

The second test is on the pattern control section of the interface. This aims to show if the user is receiving the desired stimulus.

1. Deactivate all the motors that were previously activated to check the motor control. Run the shiver pattern. Can the user replicate in the other arm the stimulus using a finger? This will show if the stimulus was well received.

2. Run the tickle pattern. Can the user replicate in the other arm the stimuli using a finger?

3. Run the diagnose pattern with a delay of 3s. Ask the user to tell when the motors change.

Figure 4.2 demonstrates the test in the form of a state diagram.

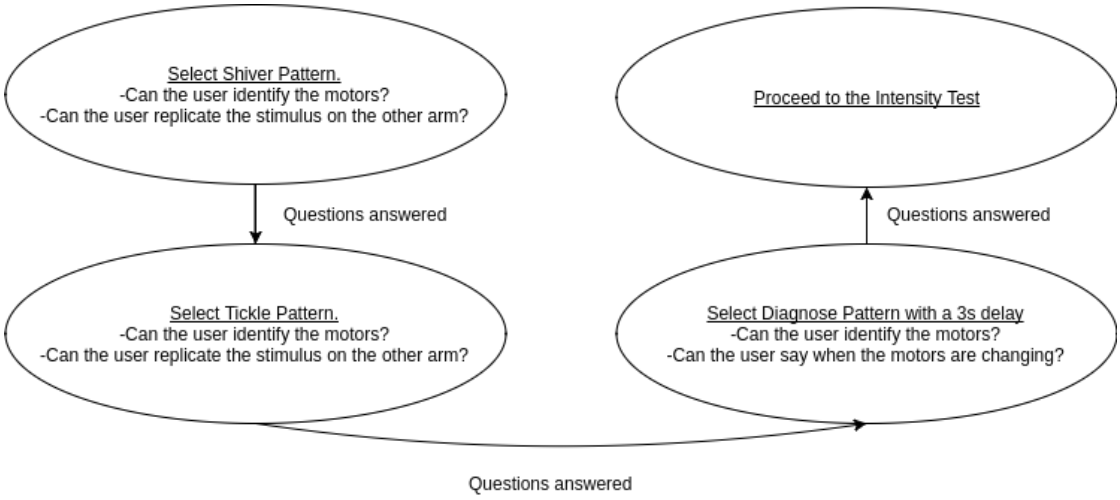

Figure 4.2: State Diagram of Pattern Control Test

4.1.3 Test of different intensities

The last test (Figure 4.3) serves to confirm that the intensity of the motor can be controlled. It also shows if the user can differentiate the contrasting intensities (which one is higher).

1. Activate motor 14 with an intensity of 3.

2. Activate the same motor with an intensity of 10. Ask the user if difference is felt. Which one is more intense?

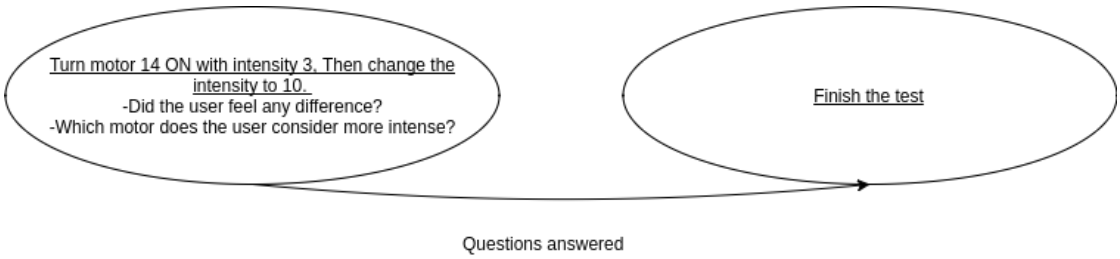

Figure 4.3: State Diagram of the Intensity Test

4.2 Questionnaire

After the test the subjects are asked to answer a small questionnaire created using Google forms. The questions can be found next.

General questions:

1. **What is your age?**
2. **What is your sex?**
   - ○ Female
   - ○ Male
3. **What is your main professional activity?**
   - ○ Student
   - ○ Teacher
   - ○ Doctor
   - ○ Other

Individual Motor Control questions:

1. **Could you localize all the motors?**
   - ○ Yes
   - ○ No
2. **If not, why do you think you couldn't?**

   (more than one answer allowed)

   - ☐ Motors were too close
   - ☐ Intensity was the same
   - ☐ Other
3. **Could you feel the different intensities?**
   - ○ Yes
   - ○ No
4. **Where do you think it's easier to detect the motors?**
   - ○ Upper hand
   - ○ Lower hand
   - ○ Arm
   - ○ None

Pattern Control questions:

1. **Did you try the shiver pattern?**
   - ○ Yes
   - ○ No
2. **What do you think about the shiver stimuli?**

   (more than one answer allowed)

   - ☐ Too fast
   - ☐ Too slow
   - ☐ Speed was ok
   - ☐ Too intense
   - ☐ Not intense enough
   - ☐ Could sense all the motors
   - ☐ Other
3. **Did you try the tickle pattern?**
   - ○ Yes
   - ○ No
4. **What do you think about the tickle stimuli?**

   (more than one answer allowed)

   - ☐ Too fast
   - ☐ Too slow
   - ☐ Speed was ok
   - ☐ Too intense
   - ☐ Not intense enough
   - ☐ Could sense all the motors
   - ☐ Other

Comments and Suggestions section:

1. Do you have any comment/suggestion related to the individual motor control?
2. Do you have any comment/suggestion related to the interface?
3. Do you have any comment/suggestion related to the prototype?
4. Other comments/suggestions

4.3 Results

To test the prototype and the validity of the method, 12 people were invited to participate in the experiments (Figure 4.4).

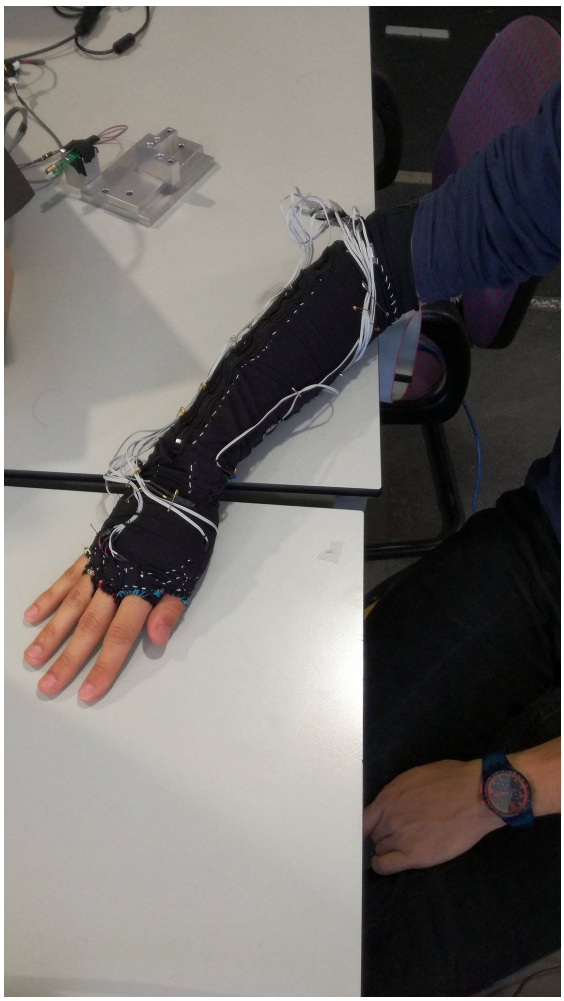

Figure 4.4: Test of the wearable device

50% of the people interviewed were women and the other 50% were men and, from those 12 people, 16.7% were teachers and 83.3% were students as Figure 4.5 illustrates.

As it was previously mentioned in Section 4.1, the test was divided in three parts: initially it was tested the individual motor control, secondly the pattern control, and lastly the different intensities.

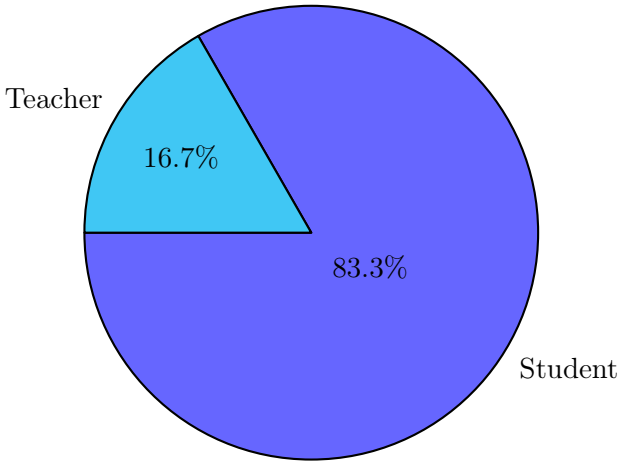

Figure 4.5: What is your main professional activity?

Regarding the individual motor control, it was asked to the subjects if they could localize all the motors that were active. Figure 4.6 shows that 33.3% could not localize all the motors and 66.7% could.

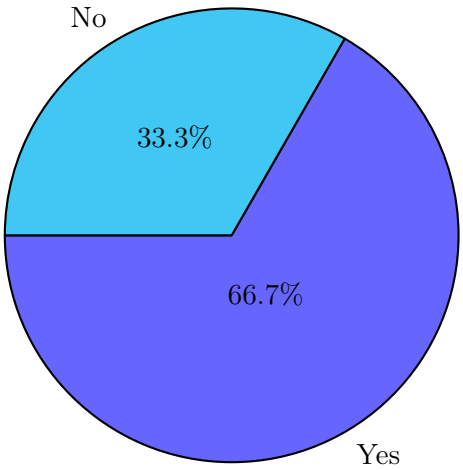

Figure 4.6: Individual Motor Control: Could you localize all the motors?

The main reason why the subjects could not localize/differentiate all the motors was because the motors were too close or due to the fact that the motors were not always touching their skin, as shown in Figure 4.7.

In the individual motor control test, several motors were activated in the upper hand, lower hand and arm. After the test, the subjects were asked where was it easier to detect the motors. Figure 4.8 shows that, 41.7% of the people interviewed think it is easier to detect the motors in the arm, 33.3% in the upper hand and 16.7% in the lower hand.

Regarding the shiver stimuli there were several opinions. Most of the subjects said that the speed was ok (91.7%) and that the intensity was also good (66.7%). 66.7% also said that they could sense all the motors and identified correctly the stimuli. More responses can be seen in Figure 4.9.

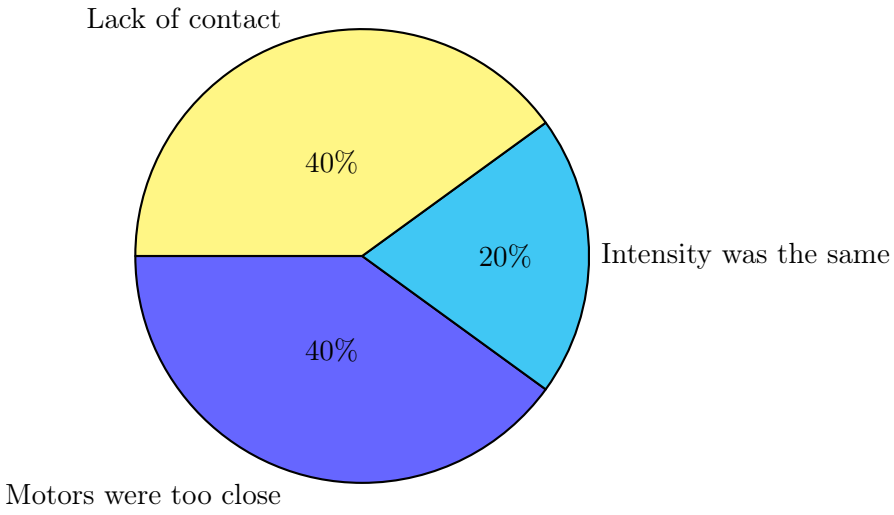

Figure 4.7: Individual Motor Control: Why do you think you could not localize the motors?

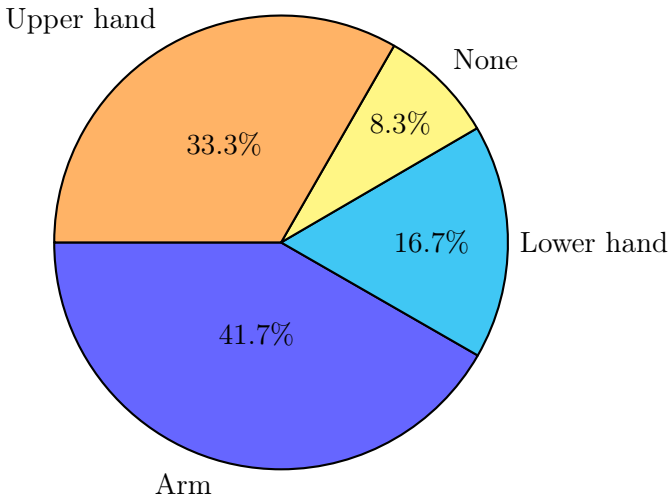

Figure 4.8: Individual Motor Control: Where do you think it's easier to detect the motors?

As for the tickle pattern the majority thought the speed was ok (66.7%), that the intensity was also ok (75%) and that they could sense all the motors (66.7%). More responses are available in Figure 4.10.

Finally, in the intensity test the same motor was activated with different intensities (a duty cycle of 30% versus a duty cycle of 100%). In this part of the test, everyone could feel the different intensities and guess correctly which stimulus was more intense.

In Appendix C a more complete view of the results of the experience can be seen.

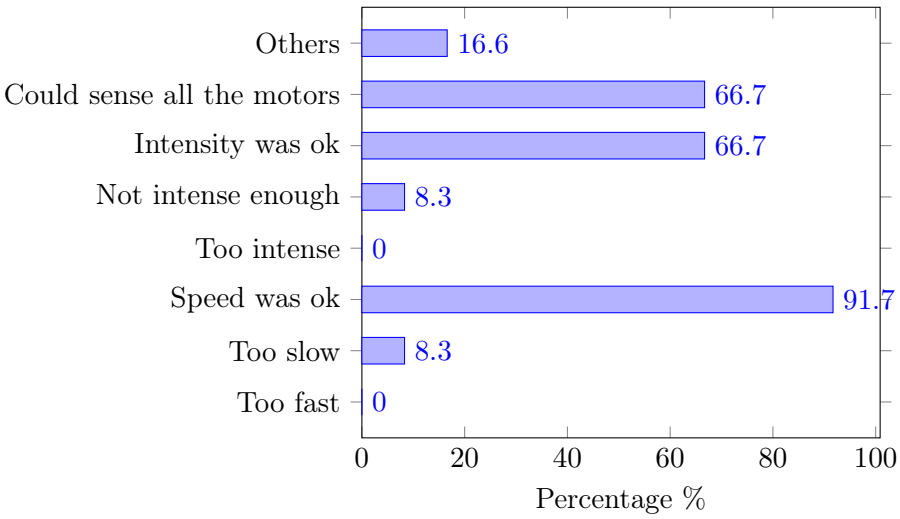

Figure 4.9: What do you think about the shiver stimuli?

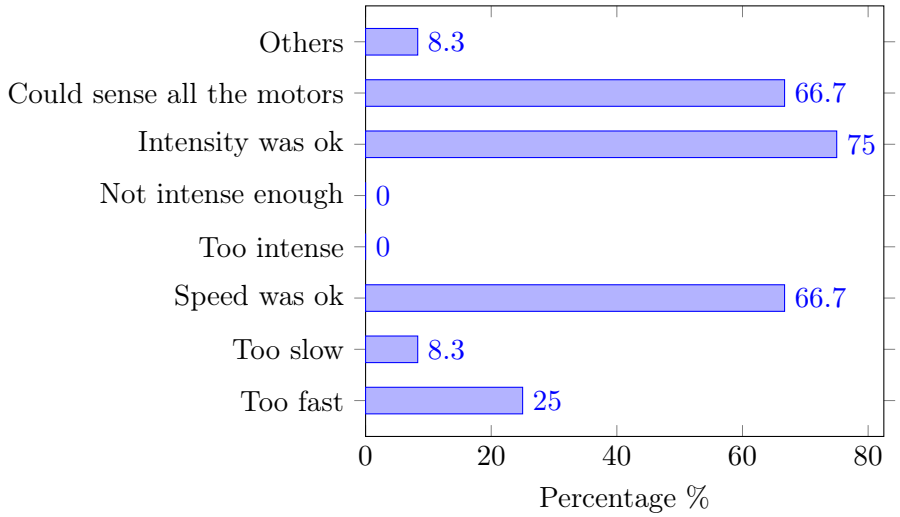

Figure 4.10: What do you think about the tickle stimuli?

In the comments/suggestions section, some interesting ideas were given by the subjects. Respecting the motor control, the possibility of the definition of a pattern in the interface could be very useful, for example in medical rehabilitation/diagnosis. This would allow, for instance, the doctor to define a specific pattern (that the doctor thinks might be useful for the patient) and to have the possibility of repeatability. Another interesting suggestion is to create a random stimulus in the GUI. Regarding the interface, the comments mentioned that the interface was user friendly but maybe not at a consumer-level yet. As for the prototype, most of what was said is that the prototype should be more robust/adjustable.

In analysis, since the motors could be activated correctly from the interface with the correct intensity, and that the subjects could feel that intensity change, the experimental tests proved that the concept worked.

Chapter 5

Conclusions and Future Work

The final conclusions of this project are explained on this chapter, evaluating the success of the proposed objectives. Some ideas for future works are also mentioned.

5.1 Conclusions

The main objectives for this project were to develop and test a prototype of a network of mini vibration motors that would allow to create sensation due to haptic feedback.

For the development of this motor network, a controller, vibration motors and a graphical interface were needed. Both the ROS and the Qt software proved that they can be used for this type of project even though that some changes can be made. The mini DC vibration motors also showed to be perfect for this project due to their size and current consumption. The Arduino also proved to be a valid option to work as a μC since the PWM method using timers and interrupts works well.

The first prototype was made using only a flat cable and with the motors connected along that cable. However, it was noted that this utilization was not practical and the prototype was changed to a sewn Lycra sleeve.

After medical analysis it was recognized that this prototype can be used in the rehabilitation medicine, such as sensitive reeducation and diagnose.

Some experimental tests were made and they proved that the proposed concept was successful and that applications in robotic teleoperation and rehabilitation have potential.

Within the framework of this project a provisional patent request is going to be made.

5.2 Future Work

In the future, the prototype should be made differently. For this project the sleeve was sewn with Lycra and Nylon. However, for a future project it is recommended that the sleeve is previously bought in an elastic material so the assembly of the motors is easier and more robust. That would also give users with different sizes a more accessible way to dress the sleeve.

For future applications, a more portable device can also be made. The ROS software can be dropped and instead bluetooth or wireless communication can be used, and the interface can also be made so that it can be used in android or iOS. Some work regarding this situation already started to being make. More can be seen in Appendix D.

Right now, this project depends on the user control of the motors, where the user decides which motor to turn on or which pattern to activate. One suggestion for future work is the use of sensors to control the location and intensity of the motors. That would allow, for instance, the user to apply pressure in someone else's arm and feel the vibration on its own body, in the same place the pressure was applied.

Another suggestion for future work is the creation of a data structure of the motors and the possibility for the user to create its own stimulation pattern instead of having to use the 3 standard patterns. Also, the possibility of sending a random stimuli could be considered.

References

All the online references were visited between February 2017 and July 2017

[1] S Ballesteros. "Implicit and explicit memory effects in haptic perception." In: *Human Haptic Perception: Basics and Applications*. M. Grundwald. Basel Boston Berlin: Birkhauser Verlag., 2008, pp. 183–197.

[2] S. Lederman and R. Klatzky. "Haptic perception: A tutorial." In: *Attention, Perception & Psychophysics*. 2009, pp. 1439–1459.

[3] J. Atkinson and O. Braddick. "Sensory and perceptual capacities of the neonate." In: *Psychobiology of the human newborn*. P. Stratton. London: John Wiley, 1982, pp. 191–22.

[4] T. Aquinas. *Commentary on Aristotles De anima*. Trans. by Foster K. and Humphries S. Dumb Ox Books, 1994.

[5] P. Bertelson and B. De Gelder. "The psychology of multi-modal perception." In: *Crossmodal space and crossmodal attention*. C. Spence and J. Driver. Oxford New York: Oxford University Press., 2004, pp. 141–177.

[6] Minerva Brooks Memorial Library Inc. *Haptic Guidance*. 2012.

[7] Joao Barros. *Cooperative haptics for humanoid robot teleoperation*. Universidade de Aveiro, Departamento de Engenharia Mecanica, 2014.

[8] R. Jutte. "Haptic perception: an historical approach." In: *Human Haptic Perception: Basics and Applications*. M. Grundwald. Basel Boston Berlin: Birkhauser Verlag, 2008, pp. 3–13.

[9] J. J. Gibson. *The senses considered as perceptual systems*. Boston: Houghton Mifflin., 1966.

[10] Pedro Cruz. *Haptic interface data acquisition system*. Universidade de Aveiro, Departamento de Engenharia Mecanica, 2012.

[11] V. Hayward, O. Astley, M. Cruz, D. Grant, and G. Robles. "Haptic interfaces and devices". In: *Sensor Review* 24.1 (2004), pp. 16–29.

[12] Verena Nitsch. "Haptic human-machine interaction in teleoperation systems and its implications for the design and effective use of haptic interfaces". In: *Universitat der Bundeswehr Munchen* (2012).

[13] H. Z. Tan, B. Eberman, M. A. Srinivasan, and B. Cheng. "Human factors for the design of force-reflecting haptic interfaces." In: *Dynamic Systems and Control*. Radcliffe, C. Vol. 55. The American Society of Mechanical Engineers, 1994, pp. 353–359.

[14] K. Salisbury, F. Conti, and F. Barbagli. *Haptic rendering: introductory concepts.* Vol. 24. IEEE Computer Graphics and Applications, 2004, pp. 24–32.

[15] H. Z. Tan. "Haptic Interfaces." In: *Communications of the ACM.* M. Grundwald. Vol. 43. 2000, pp. 40–41.

[16] M. A. Srinivasan and C. Basdogan. *Haptics in virtual environments: Taxonomy, research status, and challenges.* Vol. 21. 4. Computers and Graphics (Pergamon), 1997, pp. 393–404.

[17] V. Fulkar, M. Shivramwar, and A. Alkari. "Applications of haptics technology in advance robotics." In: *2010 International Conference on Emerging Trends in Robotics and Communication Technologies (INTERACT).* 2010, pp. 273–277.

[18] *What is Teleoperation-Telerobotics.* URL: `http://whatis.techtarget.com/definition/teleoperation-telerobotics`.

[19] *What is Teleoperation.* URL: `http://www.wisegeek.com/what-is-teleoperation.htm`.

[20] H. Pongrac. "Gestaltung und Evaluation von virtuellen und Teleprasenzsystemen an Hand von Aufgabenleistung und Prasenzempfinden." In: *Neubiberg: Universitat der Bundeswehr Munchen.* (2008).

[21] R. Goertz and R. Thompson. *Electronically controlled manipulator.* Vol. 12. Nucleonics, 1954, pp. 46–47.

[22] R. Aracil, M. Buss, S. Cobos, M. Ferre, S. Hirche, M. Kuschel, and et al. "The human role in telerobotics." In: *Advances in Telerobotics.* Ferre, M. and Buss, M. and Aracil, R. and Melchiorri, C. and Balaguer, C. Berlin Heidelberg: Springer Verlag, 2007, pp. 1–7.

[23] H. G. McCain, J. F. Andary, D. R. Hewitt, and D. C. Haley. *The space station freedom flight telerobotic servicer: The design and evolution of a dexterous space robot.* Vol. 24. Acta Astronautica, 1991, pp. 45–54.

[24] J. S. Albus, H. G. McCain, and R. Lurnia. "NASA/NBS standard reference model for telerobot control system architecture (NASREM)." In: *NIST Technical Note 1235.* National Bureau of Standards., 1989.

[25] "Robot In Germany Controlled From Russia With Unique Interface". In: *Robot Magazine* (March/April 2017), p. 11.

[26] *AILA Robot.* URL: `http://robotik.dfki-bremen.de/en/research/robot-systems/aila.html`.

[27] "Stanford's OCEAN ONE: First Deep Sea Misson". In: *Robot Magazine* (November/December 2016), pp. 10–11.

[28] *OCEAN ONE Robot.* URL: `http://news.stanford.edu/2016/04/27/robotic-diver-recovers-treasures/`.

[29] A. Casqueiro, D. Ruivo, A. Moutinho, and J. Martins. "Improving Teleoperation with Vibration Force Feedback and Anti-collision Methods". In: *Robot 2015: Second Iberian Robotics Conference. Advances in Intelligent Systems and Computing* 417 (2015), pp. 269–281.

[30] *Da Vinci Robot.* URL: http://www.davincisurgery.com/da-vinci-surgery/da-vinci-surgical-system/.

[31] *What is physical therapy.* URL: http://www.wcpt.org/what-is-physical-therapy.

[32] *Rehabilitation.* URL: http://walk-again.com/hal-therapy/.

[33] *Nervous system.* URL: http://www.livescience.com/22665-nervous-system.html.

[34] *Neurologic Diseases.* URL: http://www.nlm.nih.gov/medlineplus/neurologicdiseases.html.

[35] *Topographic and Functional Anatomy of the Spinal Cord.* URL: http://emedicine.medscape.com/article/1148570-overview.

[36] *Spinal Cord Anatomy and Injuries.* URL: https://www.slideshare.net/UtkarsshWayal/spinal-cord-anatomy-and-injuries.

[37] Frank H. Netter MD. *Atlas of Human Anatomy.* 6th ed. Saunders Elsevier, 2014.

[38] *Dermatomes.* URL: http://emedicine.medscape.com/article/1878388-overview.

[39] *What is Arduino.* URL: http://www.arduino.org/learning/getting-started/what-is-arduino.

[40] *Arduino MEGA 2560.* URL: https://www.arduino.cc/en/main/arduinoBoardMega2560.

[41] *Comparison Arduino Boards.* URL: https://www.arduino.cc/en/Products/Compare.

[42] PRECISION MICRODRIVES. *Vibration Motors Comparison Guide.* URL: https://www.precisionmicrodrives.com/application-notes/ab-028-vibration-motor-comparison-guide.

[43] MICROCHIP. *Brushed DC Motor Fundamentals.* URL: http://ww1.microchip.com/downloads/en/AppNotes/00905B.pdf.

[44] Adafruit. *Mini DC Motor Characterisitics.* URL: https://www.adafruit.com/product/1201.

[45] PRECISION MICRODRIVES. *COIN VIBRATION MOTORS.* URL: https://www.precisionmicrodrives.com/vibration-motors/coin-vibration-motors.

[46] PRECISION MICRODRIVES. *DRIVING VIBRATION MOTORS WITH PULSE WIDTH MODULATION.* URL: https://www.precisionmicrodrives.com/application-notes/ab-012-driving-vibration-motors-pulse-width-modulation.

[47] Jason M. O'Kane. *A Gentle Introduction to ROS.* University of South Carolina, Department of Computer Science and Engineering, 2014.

[48] *NAO Robot.* URL: http://sensilab.monash.edu/asset/nao/.

[49] *ROS Nodes.* URL: http://wiki.ros.org/Nodes.

[50] *ROS intro.* URL: http://robohub.org/ros-101-intro-to-the-robot-operating-system/.

[51] *ROS Master*. URL: http://wiki.ros.org/Master.

[52] *ROS Messages*. URL: http://wiki.ros.org/Messages.

[53] *ROS Topics*. URL: http://wiki.ros.org/Topics.

[54] ROS. *ROS Introduction*. URL: http://wiki.ros.org/ROS/Introduction.

[55] *Arduino IDE*. URL: https://www.arduino.cc/en/Guide/Environment.

[56] *Rosserial package*. URL: http://wiki.ros.org/rosserial.

[57] *Rosserial Arduino package*. URL: http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup.

[58] *What is a timer*. URL: https://www.engineersgarage.com/definitions/what-is-timer.

[59] *ATMEL 2560 Datasheet*. URL: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf.

[60] *Timers and Interrupts*. URL: https://arduino-info.wikispaces.com/Timers-Arduino.

[61] *Arduino Playground - Timer1*. URL: http://playground.arduino.cc/Code/Timer1.

[62] *Arduino Forum - Timer 4*. URL: http://forum.arduino.cc/index.php?topic=220775.0.

[63] Qt. *Qt Introduction*. URL: https://wiki.qt.io/About_Qt.

[64] Johan Thelin. *Foundations of Qt Developmnet*. 2007.

[65] *What is bitmask*. URL: http://codeforces.com/blog/entry/18169.

[66] *Stackoverflow: What is bitmask*. URL: https://stackoverflow.com/questions/10493411/what-is-bit-masking.

[67] *Hand and Arm image*. URL: http://www.zbrushcentral.com/showthread.php?31645-First-Model-Hand-and-Arm.

Appendices

Appendix A

Digital Oscilloscope Screenshot Process

In this process the UNIT-T UT3102C Digital Storage Oscilloscope was used (Figure A.1).

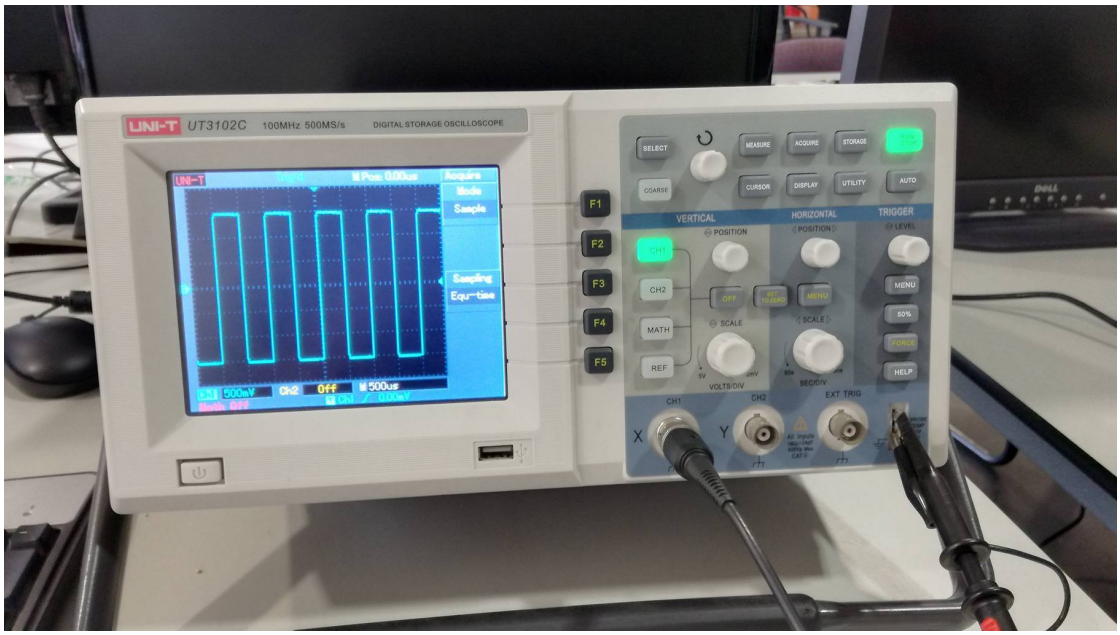

Figure A.1: UNIT-T UT3102C Digital Storage Oscilloscope

After turning the oscilloscope on, go to the "Storage" menu as shown in Figure A.2 and insert the previously formated USB stick. The USB must be formated in FAT.

In the storage menu, change the type to "Bit Map" and make sure that the "Dest" (destiny) is USB. To save the screenshot select "Save". Be aware that only one picture can be saved at a time as the previous picture is replaced by the new one every time "Save" is pressed.

Figure A.3 illustrates the process of comparing the two PWM waves using this Digital Oscilloscope.

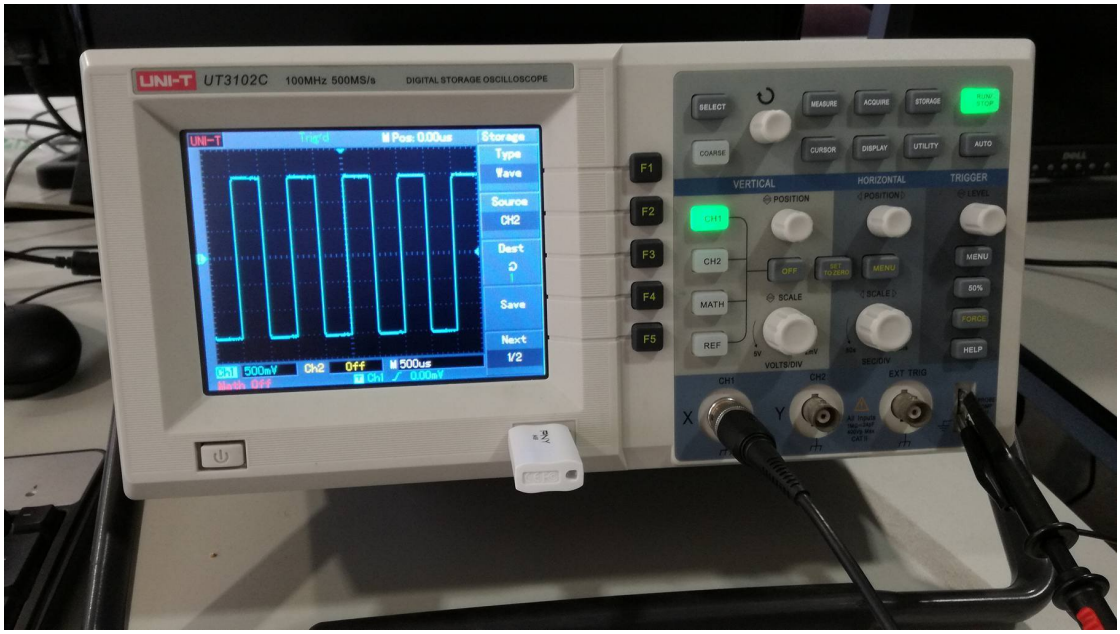

Figure A.2: UNIT-T UT3102C Digital Storage Oscilloscope: Storage Menu

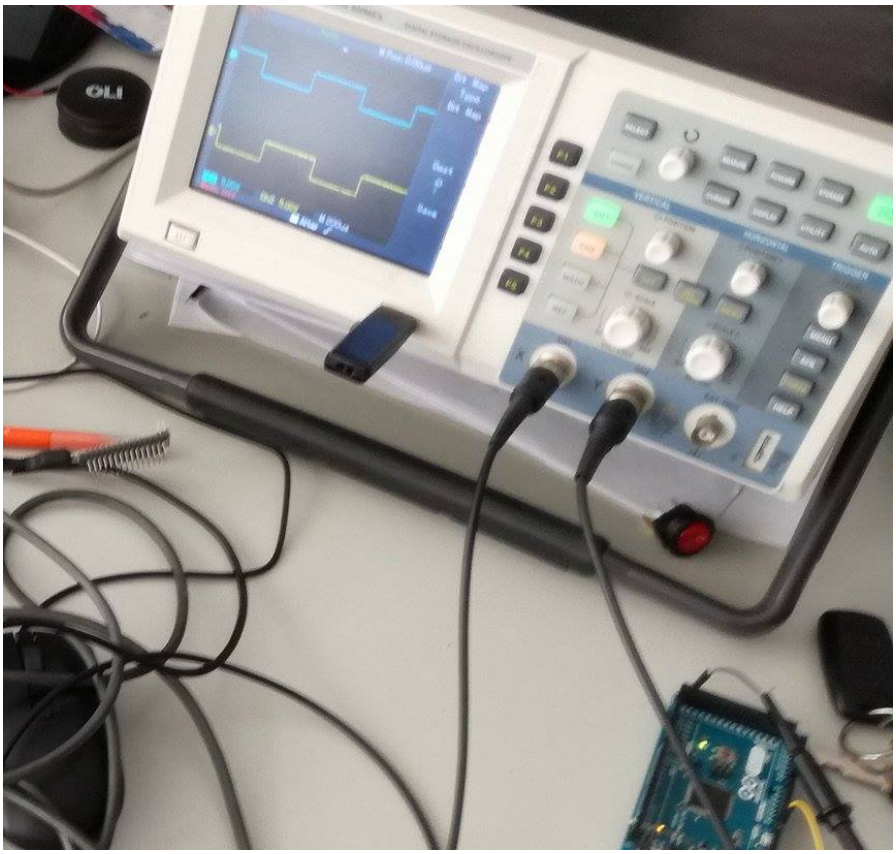

Figure A.3: Comparison of the two PWM waves process

Appendix B

Interface Node CMakeLists

```
#My CMakeLists.txt

cmake_minimum_required (VERSION 2.8)
project (qtgui)

#---------------------------------
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
 message_generation
  message_runtime
qt_ros_interface
)

FIND_PACKAGE(Qt4 REQUIRED)
INCLUDE(${QT_USE_FILE})
ADD_DEFINITIONS(${QT_DEFINITIONS})

SET(qtgui_SOURCES src/main.cpp src/mainwindow.cpp src/QtPublisher.cpp)
SET(qtgui_FORMS src/mainwindow.ui)
SET(qtgui_HEADERS src/mainwindow.h src/globals.h src/QtPublisher.h)
set(QRC_RESOURCES src/qtgui.qrc)
set(CMAKE_AUTOUIC ON)
set(CMAKE_AUTOMOC ON)

QT4_WRAP_CPP(qtgui_HEADERS_MOC ${SimpleProject_HEADERS})
QT4_WRAP_UI(qtgui_FORMS_HEADERS ${SimpleProject_FORMS})
QT4_ADD_RESOURCES(RESOURCES_RCC ${RESOURCE})

## Generate messages in the 'msg' folder
 add_message_files(
   FILES
```

```
   GUIDados.msg
 )

## Generate added messages and services with any dependencies listed here
 generate_messages(
  DEPENDENCIES
   std_msgs # Or other packages containing msgs
 )

## Declare a catkin package

catkin_package(CATKIN_DEPENDS roscpp rospy std_msgs message_runtime)
include_directories(${catkin_INCLUDE_DIRS})
INCLUDE_DIRECTORIES(${CMAKE_CURRENT_SOURCE_DIR} ${CMAKE_CURRENT_BINARY_DIR})

ADD_EXECUTABLE(qtgui
    ${qtgui_SOURCES}
    ${qtgui_HEADERS_MOC}
    ${qtgui_FORMS_HEADERS}
    ${QRC_RESOURCES}
)

TARGET_LINK_LIBRARIES(qtgui ${QT_LIBRARIES} ${catkin_LIBRARIES})
```

Appendix C

Table of Results of the Experimental Tests

General Questions

Table C.1: Questionnaire: What is your age?

Age	**Percentage (%)**
22	33.3%
23	25%
18	8.3%
20	8.3%
24	8.3%
48	8.3%
52	8.3%

Table C.2: Questionnaire: What is your sex?

Genre	**Percentage (%)**
Female	50%
Male	50%

Table C.3: Questionnaire: What is your main professional activity?

Professional Activity	**Percentage (%)**
Student	83.3%
Teacher	16.7%
Doctor	0%
Other	0%

Individual Motor Control questions

Table C.4: Questionnaire: Could you localize all the motors?

Answer	**Percentage (%)**
Yes	66.7%
No	33.3%

Table C.5: Questionnaire: If not, why do you think you couldn't?

Reason	**Percentage (%)**
Motors were too close	40%
Lack of contact	40%
Intensity was the same	20%

Table C.6: Questionnaire: Could you feel the different intensities?

Answer	**Percentage (%)**
Yes	100%
No	0%

Table C.7: Questionnaire: Where do you think it's easier to detect the motors?

Local	**Percentage (%)**
Arm	41.7%
Upper hand	33.3%
Lower hand	16.7%
None	8.3%

Pattern Control questions

Table C.8: Questionnaire: Did you try the shiver pattern?

Answer	**Percentage (%)**
Yes	100%
No	0%

Table C.9: Questionnaire: What do you think about the shiver stimuli?

Answer	**Percentage (%)**
Speed was ok	91.7%
Intensity was ok	66.7%
Could sense all the motors	66.7%
Too slow	8.3%
Not intense enough	8.3%
Just noticed it at second time	8.3%
Felt a little artificial	8.3%
Too fast	0%
Too intense	0%

Table C.10: Questionnaire: Did you try the tickle pattern?

Answer	**Percentage (%)**
Yes	100%
No	0%

Table C.11: Questionnaire: What do you think about the tickle stimuli?

Answer	**Percentage (%)**
Intensity was ok	75%
Speed was ok	66.7%
Could sense all the motors	66.7%
Too fast	25%
Too slow	8.3%
Felt a little artificial	8.3%
Too intense	0%
Not intense enough	0%

Appendix D

Hands-Free device

As it was mentioned in Section 5.2, the communication is meant to be bluetooth or wireless in the future. Since the communication is going to be hands-free, it makes no sense to have the Arduino connected to the computer to power it.

To power the Arduino, a rechargeable 3.7V lithium battery (with 5V output) circuit was made. The idea of a rechargeable battery came from the fact that, this way, there is no need to change the battery from time to time.

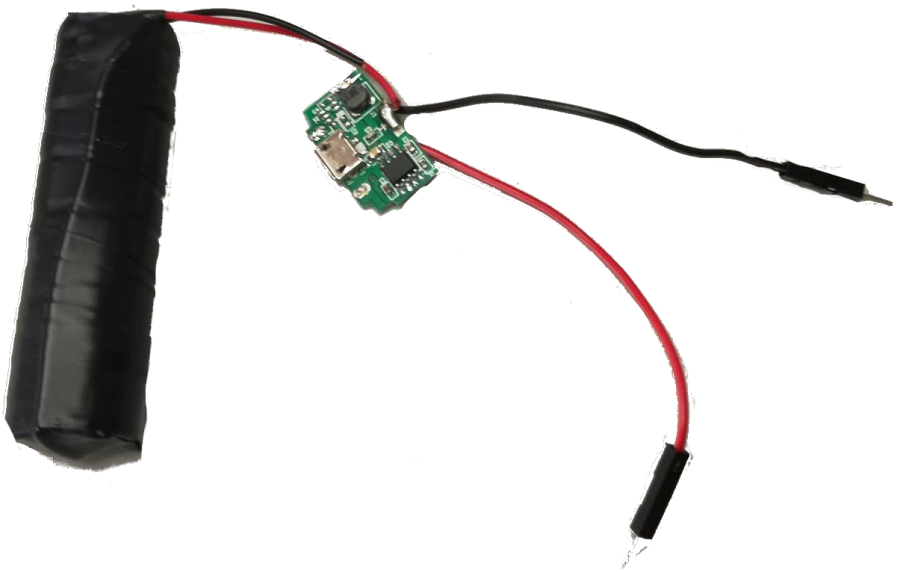

Figure D.1: Recharging circuit

To enclosure these components, a box was modeled using the CAD software *SolidWorks*. This box has a place to put the battery and its charging circuit, the bluetooth and the Arduino. It also offers the possibility of connecting the battery to a Micro-USB cable to recharge it, a ON/OFF button to turn on the controller and the access to the Arduino serial cable in case some changes on the code need to be made.

Figure D.2 and Figure D.3 show the interior of the box and an assembly of the box, respectively. This box was printed using a Three-Dimensional (3D) printer at the

Mechanical Engineering Department (DEM) of the University of Aveiro.

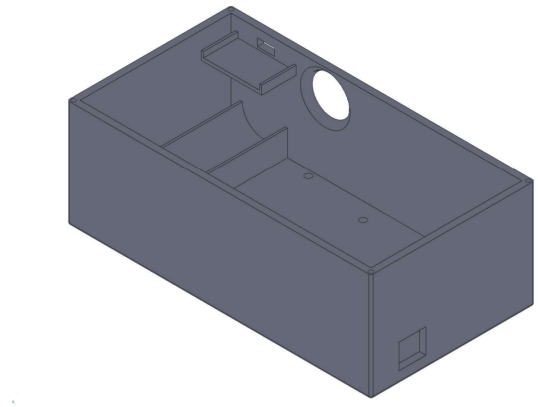

Figure D.2: Interior of the box

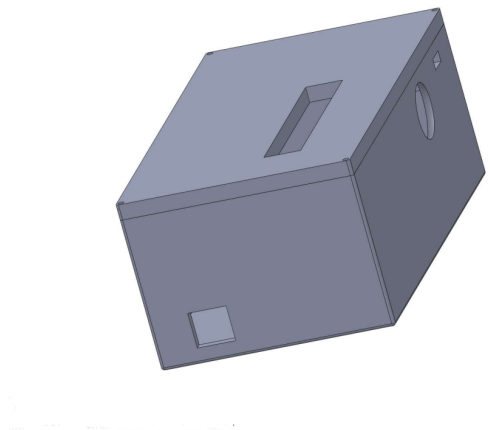

Figure D.3: Assembly of the Box