Emílio
Gerardo Estrelinha

Tele-operação de um Robô Humanóide usando Háptica e Sensores de Força
Tele-operation of a humanoid robot using Haptics and Load Sensors

Emílio
Gerardo Estrelinha

Tele-operação de um Robô Humanóide usando Háptica e Sensores de Força
Tele-operation of a humanoid robot using Haptics and Load Sensors

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 Vítor 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 Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

O júri / The jury

Presidente / President

Prof. Doutor Jorge Augusto Fernandes Ferreira
Professor Auxiliar da Universidade de Aveiro

Vogais / Committee

Prof. Doutor António Manuel Ferreira Mendes Lopes
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

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

Prof. Doutor Filipe Miguel Teixeira Pereira da Silva
Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (co-orientador)

Agradecimentos / Acknowledgements

Quero aqui agradecer a algumas pessoas sem as quais não teria conseguido desenvolver o trabalho aqui apresentado. Assim, agradeço a todos aqui referidos a ajuda e o apoio que me deram.

Aos meus pais e irmãos pelo apoio que me deram durante a minha vida académica e por me suportarem todos estes anos.

Ao professor Vitor Santos pelo acompanhamento, dedicação e motivação contagiante ao longo dos últimos dois anos. Ao professor Filipe Silva pela incessante ajuda e concelho. Deixo aqui também os meus desejos para que o Projeto Humanóide continue a crescer e evoluir.

Ao futuro Doutor Jorge Almeida por toda a ajuda, crítica e concelho dado sobre programação e organização de código. Ao Doutor Ricardo Pascoal pelos concelhos na concepção da placa de aquisição. Ao Pedro Cruz pelo conhecimento passado e pelos desafios que deixou. A todos os meus colegas do terceiro piso pelas sessões de *brainstorming*, amizade e camaradagem.

Ao Engenheiro António Festas e ao Sr. Júlio Gonçalves pela ajuda e cooperação na fabricação da placa impressa.

Ao Prof. Carlos Relvas pela cedência do joystick háptico para a realização deste trabalho.

Palavras-chave

Háptica, interface háptica, células de carga, aquisição de sinal, teleoperação, robótica humanoide, aprendizagem robótica.

Resumo

O principal objetivo desta tese é criar uma plataforma eficaz e modular para preparar o Projeto Humanoide da Universidade de Aveiro (PHUA) para o ensino tele-cinestésico. Este novo conceito de aprendizagem robótica por demonstração é neste momento a base deste projeto, onde um usuário humano tele-opera o robô em vários movimentos e tarefas de equilíbrio. Os dados recolhidos durante estas demonstrações podem ser usados em algoritmos de aprendizagem de modo a que o robô se possa mover, equilibrar e caminhar sozinho. O robô utilizado neste trabalho é uma plataforma humanoide proprietária com 27 GDL desenvolvida completamente na Universidade de Aveiro. Várias demonstrações são realizadas neste trabalho usando dados sensoriais das células de carga instaladas nos pés do robô e um dispositivo háptico para interface com o utilizador. Quatro células são colocadas em cada um dos pés dando ao robô a capacidade para sentir o chão e estimar o centro de pressão. Uma unidade de aquisição de dados foi desenvolvida para obter os sinais das células de carga. Esta unidade é capaz de ler das células e transmitir essa informação a uma frequência superior a $1000\,\mathrm{Hz}$ permitindo um fluxo de informação praticamente contínuo. A informação de força dos pés é então usada para gerar a realimentação de força do dispositivo que é sentida pelo utilizador como o desequilíbrio do robô. A plataforma ROS é usada para controlar os diferentes módulos de *software*, utilizando o próprio sistema de mensagem para comunicar entre estes, dando a este projeto várias ferramentas para posterior desenvolvimento.

Keywords Haptics, haptic interface, load cells, signal acquisition, teleoperation, humanoid robotics, robot learning.

Abstract The main objective of this thesis is to create an effective and modular platform to prepare the Humanoid Project of the University of Aveiro (PHUA) for tele-kinesthethic teaching. This new concept of robot learning from demonstration is now the base of this project, where a human user tele-operates the robot in various motion and balance tasks. The data gathered during this demonstrations can be used in learning algorithms so the robot can move, balance and walk on it's own. The robot used in this work is a 27-DOF proprietary humanoid platform developed completely at the University of Aveiro. Several demonstrations are carried out using sensory data from the load cells installed in the robot's feet and an haptic device for user interface. Four cells are placed in each foot giving the robot the ability to sense the floor and to estimate the center of pressure. A unit for data acquisition was developed to measure the load cells signals. This unit is capable of reading the cells and transmitting that information at a frequency over 1000 Hz allowing for a nearly continuous stream of information. The force information from the feet is then used to generate the force feedback in the haptic device which is felt by the human as the robot's sense of balance. The ROS platform is used to control the different modules of software using it's message system to communicate among them, giving this project several tools for further development.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The idea of an autonomous human-like robot has been a dream of some for a long time. Introduced by science fiction, the idea that mankind can some day create a fully functional lookalike of humans to do any task too dangerous or too monotonous has inspired many people. It is thought that this achievement will give mankind a chance to pursue other endeavours otherwise not attainable. So far, no such robotic piece has come together despite all the scientific advances in the last 50 years.

The humanoid robotics field gives engineers a great challenge in an wide range of areas, such as biomechanics, physics, electronics and mechanical. With each area, comes an array of problems. One of the main problems is the motion control. Due to the parallel robotics, there no model for motion of a humanoid robot and the found solutions are based in pre-programmed routines. Contrasting with these, the robot learning from demonstration algorithms give the possibility to have robotic systems with learning capabilities and able to adapt to new environments. To study and implement this methods a structured framework is needed. The work here presented tries to introduce a solution for these problems with a haptic interface combined with a modular programming structure.

1.1 Motivation

This work is the preliminary stage of a long term goal to teach auto-balancing and walking motion with a learning from demonstration algorithm. The idea is to move the robot in different stages of equilibrium while using haptics to transmit its state back to the operator. The force feedback can be generated by different sensory data (inertial, center of pressure, tilting,etc.), just like humans. The robot's stability is deduced with this data and felt by the operator with the force feedback.

By recording all the robot's information (state of joints, center of pressure, haptic device kinematics, etc.) during those episodes, the PHUA project believes that it is possible to use a learning from demonstration algorithm. After which the robot knows how to perform such equilibrium and motion operations without the operator's control.

1.2 Objectives

The objectives of this work are to create a software and hardware basis to work on the PHUA robot's equilibrium and stability.

This is achieved by sensing the floor in with the robot is standing and with that information calculate the vertical projection of the center of gravity , also know as center of pressure (COP). The state of the COP is then transformed into force feedback to be used in the haptic device that also controls the movement of the robot.

The main objectives of this work are presented in the bullet points below.

- Implement the ROS (Robot Operating system) platform into the project
- Develop a data acquisition unit for the load cells
- Link the load cells data with the haptic force feedback
- Develop a control system for the robot with haptics
- Validate the methodology by performing balancing experiences
- Record all relevant data from the experiences

The diagram in Figure 1.1 shows the main problem for this project.

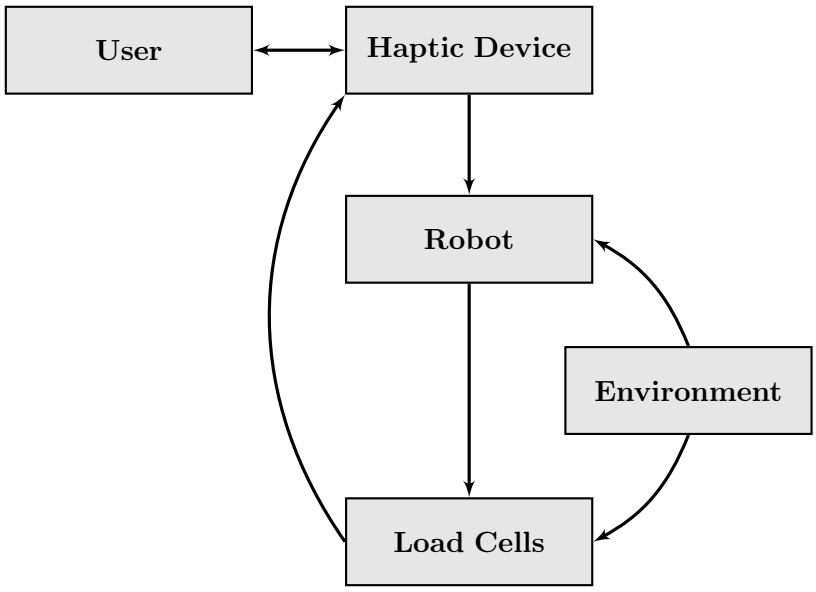


Figure 1.1: Schematic of the main parts for this work.

The user handles the haptic device to control the robot while is sensing the force feedback. The external interactions (gravity, friction, external forces, tilted planes, etc.) influence the robot and the load cells. These give the information about the robot's equilibrium and in turn activate the force feedback.

1.3 Thesis guide

The thesis is organized by chapters in the fallowing manner: the rest of this Chapter gives the reader an explanation about the past work done on the PHUA and its robot since the beginning of the project. It also shows how the haptic device is incorporated in that work. The Chapter 2 has two parts: the first describes the chosen approach for the signal acquisition from the pressure cells and all the other hardware requirements for this work. The second explains the structure for the software structure used in this work and its possibilities. The Chapter 3 explains how the modules created work and how they interact with the materials presented in Chapter 2. Chapter 4 presents the demonstrations done to validate this work and its results. The final chapter discuss the main points of this thesis, presents its conclusions and proposes future work hypothesis.

1.4 Framework

1.4.1 Haptic device

The haptic device this work uses is the PHANTOM OMNI by Sensable, shown is Figure 1.2. The OMNI device is a 6 degrees of freedom (DOF) impedance-based joystick able to render three-dimensional forces at high frequencies. This device uses a proprietary library provided by the manufacturer, explained in Section 2.2.2.

Even though the device has 6 DOF, due to it's mechanical construction only the first three joints are actuated. This means that it is only possible to render force vectors and not toques since the last three DOF's are not actuated. These last joints possess potentiometers that give out their positions resulting in the gimbals angles of the tip. The position is given by high precision optical encoders that are coupled at the base's actuators.

The specifications of the PHANTOM OMNI device are shown in Table 1.1. The workspace where it's capable of exerting its nominal maximum force is quite small, so when using the force feedback to command the robot a scale factor needs to be used. Outside of this workspace, the forces magnitude and directions are not guaranteed to be the ones sent to the device. The apparent mass at the tip is a useful data for gravity compensation.

Figure 1.2: PHANTOM OMNI by Sensable haptic device [SensAble, 2008c].

Table 1.1: PHANToM OMNI device specifications [SensAble, 2008c].

Force Feedback Workspace	160 W X 120 H X 70 D [mm]
Footprint (Physical area the base of the device occupies)	168 W x 203 D [mm]
Weight (device only)	3 lb 15 oz (aprox. 1.786 kg)
Range of Motion	Hand movement pivoting at wrist
Nominal Position Resolution	450 dpi (aprox. 0.055 mm)
Backdrive Friction	0.26 N
Maximum Exertable Force (at nominal position)	3.3 N
Continuous Exertable Force (24 hrs.)	0.88 N
Stiffness	X axis : 1.26 N/mm Y axis : 2.31 N/mm Z axis : 1.02 N/mm
Inertia (apparent mass at tip)	45 g
Force Feedback	x, y, z
Position Sensing (with stylus gimbal)	- x, y, z (digital encoders) - roll, pitch, yaw (±5% linearity potentiometers)
Interface	IEEE-1394 FireWire port (6-pin to 6-pin)

1.4.2 Humanoid Project of the University of Aveiro (PHUA)

This project was started in 2003/2004 in a joint effort between the Mechanical Engineering Department (DEM) and the Electronics, Telecommunications and Informatics Department (DETI) to create a humanoid platform suitable for research in control, perception, navigation and behaviour. It's main objective is to develop and integrate both software and hardware in a functional low-budget platform.

The first model was finish in 2008 with several thesis and publish papers during those 4 years of development. The platform consisted in a humanoid robot with 22 DOF weighting 6 Kg with its 9600 mA 7.4 V batteries and with an approximated height of 60 cm. Assisted with a PC-104 connected with a CAN network by RS-232 protocol, this platform was capable of identifying and following objects with its head and performing several exercises with its legs [Ferreira, 2007, Rodrigues, 2008, Silva, 2006].

In 2009, the project took another important step and a new platform was developed. The mechanical structure was redesigned and new actuators and sensors were chosen. The new platform has 27 DOF, visible in Figure 1.3. *HITEC®* digital and analogue servomotors actuate 25 of the joints which are arranged as in Figure 1.4 with the servomotors assembled as in Table 1.2. When assembled it weights about 6 kg and has an approximated height of 65 cm [Cruz, 2012, Godinho, 2011, Lage, 2011, Ribeiro, 2010, Sabino, 2009].

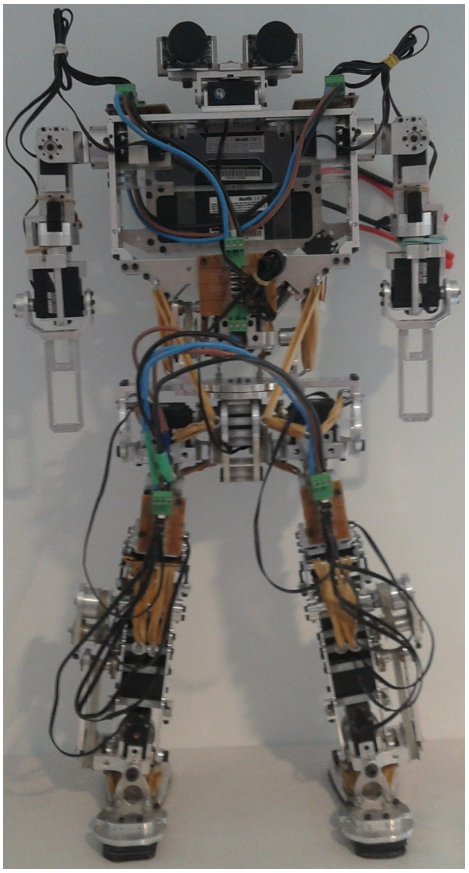


Figure 1.3: PHUA robot, front view [Cruz, 2012].

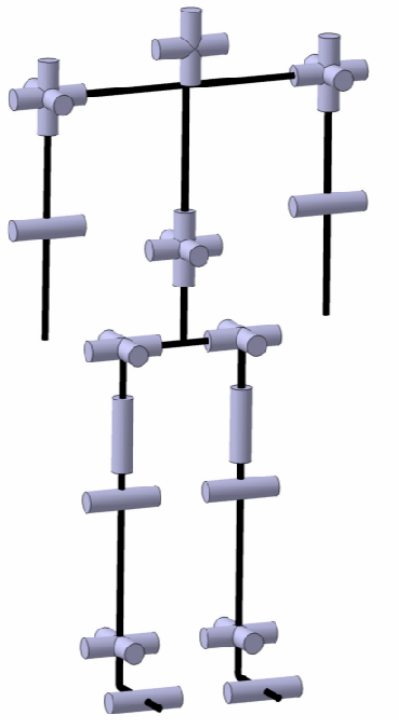


Figure 1.4: PHUA robot degree's of freedom [Cruz, 2012].

Table 1.2: PHUA platform's degrees-of-freedom and installed servomotors.

Joint	**Number of DOFs**	**Servomotor**
Toe	1(×2)	(none)
Ankle	2(×2)	*HSR-5980SG*
Knee	1(×2)	*HSR-5980SG*
Hip	3(×2)	*HSR-5980SG*
Trunk	3	*HSR-5980SG*(×2) *HSR-8498SG*(×1)
Shoulder	3(×2)	*HSR-5980SG* *HSR-5498SG* *HS-82MG*
Elbow	1(×2)	*HSR-5498SG*
Neck	2	*HSR-5498SG* *HS-5056MG*
Total	27	—

The servomotors possess a proprietary protocol that make use of RS-232 called *Bidirectional Serial Interface*. This works with a 7 byte frame, represented in the Table 1.3, where the first 5 are the command, parameter and checksum. The last 2 bytes are placed by the servomotor responding to the command [Cruz, 2012].

Table 1.3: The HITEC *BSI* RS-232 7 byte word [Ribeiro, 2010].

byte	1	2	3	4	5	6	7
Controller	Startbyte	Command	Param1	Param2	Checksum	0x00	0x00
Servomotor	High-Impedance					Return1	Return2

Each servomotor has a PD controller based in position, ranging from 600 to 2400, with speed input, integer values between 1 and 255. Any one of the servomotors is identified by a 2 digit number setted with a HITEC supplied software [Cruz, 2012].

Part of the locomotion system of the robot is the elastic elements present in the lower limbs and torso. These help store potential energy to be dispensed in a later phase of the gait. Rubber bands are placed in some of the lower limbs's joints. Although they keep the servomotors in strain these elastic elements ensure the stability and return to the initial pose [Cruz, 2012, Santos et al., 2010, 2012].

In order to provide the correct binary to the different joints and reduce the strain on the actuators's transmission axis belted transmission were used to transmit power to the joints of the legs and torso. Belt tensioners keep the transmission stable and allow for adjustments.

The PHUA platform is also equipped with 8 pressure sensors in its feet, that will be further analized in chapter 2, and 2 *FireFly®* *FireWire* cameras for the artificial vision system.

The last work done in this project introduced the concept of tele-kinesthethic teaching. The idea is junction between teleoperation and learning by demonstration.

The developed haptic interface used force feedback for the operation of the robot. This interface tries to improve kinesthethic teaching through an haptic environment of the robot's surroundings and it's own restrictions [Cruz, 2012].

The data recorded during such operations can be ran through a robot learning from demonstration algorithm, without the need to use complex dynamic models. This methods usually relies on a teleoperation system or a vision or motion sensors with the user's recording of the task. A haptic interface can change the paradigm on this issue. With the ability to 'feel' the robot's state, the user can dynamically adapt to changes in the environment or to the robot's restrictions. Using the user's reactions to the force feedback allows the creation model-free control system [Cruz et al., 2012]. Before the robot learning from demonstration can be applied, a number of steps have to be taken and this work is the start of those steps.

Chapter 2

Experimental setup

This chapter presents all the needed parts to complete the proposed solution.

The first section explains what steps were taken to ensure the hardware was functional. All the electronics are described in this part as well as the sensors and actuators. The software section is presented next where the libraries and programming platforms are explained.

2.1 Hardware

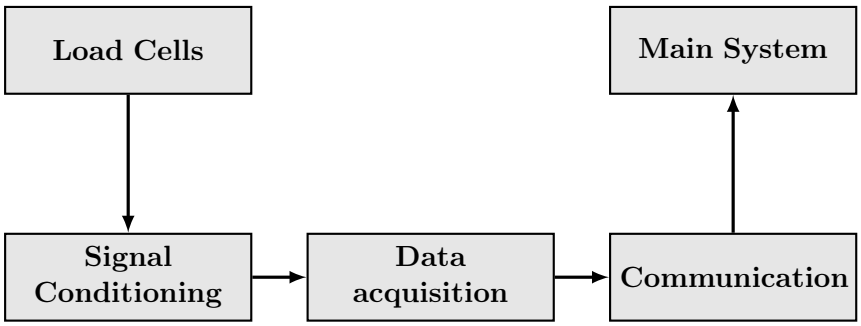


Figure 2.1: Interactions between the developed hardware.

In this section the hardware developed in this work is explained in depth. The schematic in Figure 2.1 shows how the hardware interacts. The load cells sense the reactions to the ground. This information needs to have a signal conditioning fit for this application. After this, it requires a data acquisition and data communication system to the main system.

2.1.1 Load cells

The load cells are Interface's LBS-5 and LBS-10, with a weight limit of 5 and 10 pounds of force respectively. This choice is due to its small size as seen in Figure 2.2. Table 2.1 presents the load cells characteristics where the 150% overload and the temperature compensation stand out as surplus to the solution.

Figure 2.2: Load cell LBS-5/10/25/50 model.

Table 2.1: LBS-5 and LBS-10's characteristics [INTERFACE, 2009].

Characteristics	**LBS-5**	**LBS-10**
Nominal Load [lbs]	5	10
[N]	22,2	44,4
Excitation, nom	5	5
max	7	7
Safe Overload	150%	150%
Nonlinearity–% FS	0,25	0,5
Compensated Temp. Range [ºF]	60 - 160	60 - 160
Bridge Resistance [Ω]	350	350
Rated Output [mV/V]	2.0	2.0

Due to the differential signal output from the cells, the preferred amplification would be a rail-to-rail output with low noise and voltage offset. Also needed to take into account is the rated output of the cells. The $OutputRange\,(O_{range})$ is determined by the $RatedOutput\,(O_{rated})$ and the $Excitation\,(VCC)$, as presented in the Equation 2.1.

$$O_{range} = O_{rated} \times VCC = 2,0 \times 5 = 10mV \tag{2.1}$$

Since both models have the same rated output, it is possible to make the amplification and conversion set up equal in hardware for both types of cells. This method will allow for interchangeable load cells as long as the calibration curves are updated in the software module.

As planed in the lower limbs's project [Sabino, 2009], the cells are located between two plates in the feet, placed as in Figure 2.3. Each cell is placed in a machinated circular box with their center in the coordinates of Table 2.2.

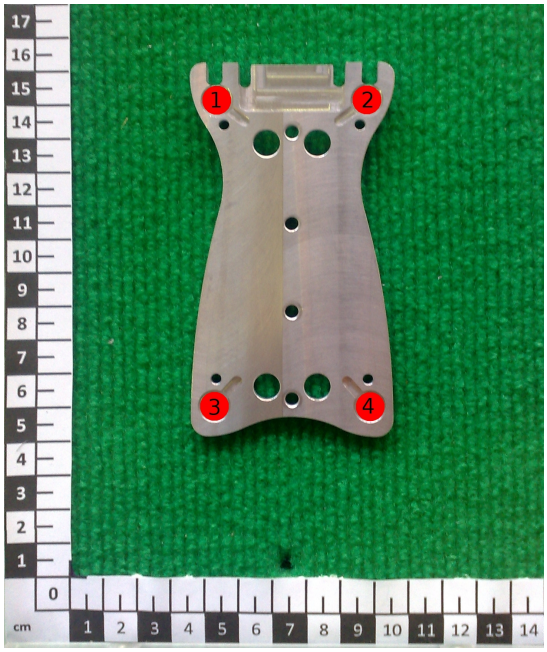


Figure 2.3: Location of the load cells in the bottom piece of the foot [Sabino, 2009].

Table 2.2: Load cells coordinates [Sabino, 2009].

Nº of the load cell	X [m]	Y [m]
1	0.044	0.022
2	0.044	-0.022
3	-0.046	-0.023
4	-0.046	0.023

2.1.2 Aquisition and Communication

To make the state of the load cells available at the main system an interface is needed. This needs to have analogue to digital conversion capabilities and be able to communicate with other systems. Two *ARDUINO®* boards are used for this purpose, one for each foot.

ARDUINO is a company, that in cooperation with other companies, releases open/free designs for prototyping and testing boards. These are typically composed of one *ATMEL®* microcontroller and supporting electrical conditioning. Most ARDUINO board's are expandable by stacking one or more *Shields* to them (motor control shield, Ethernet shield, *X-BEE®* shield, etc.).

In this work the Fio model is used, shown in Figure 2.4. The main relevant characteristics for this work are detailed in the Table 2.3.

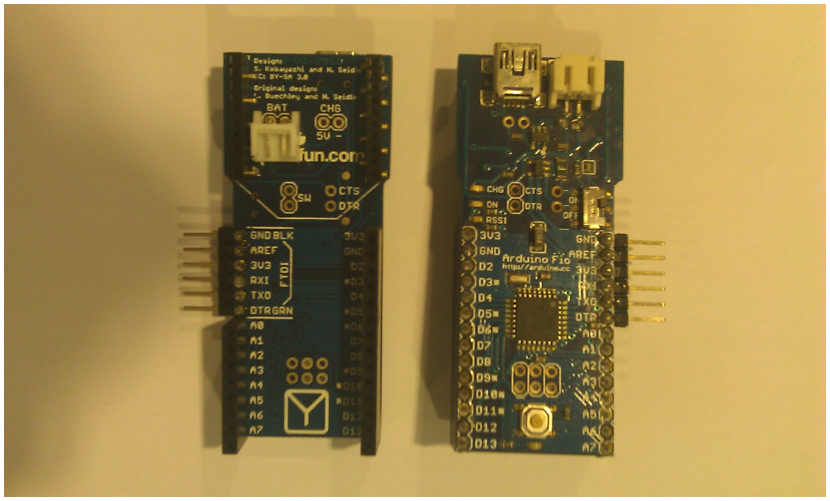


Figure 2.4: *ARDUINO®* Fio used in this work.

Table 2.3: ARDUINO Fio's main characteristics.

Fio's Characteristics	
Microcontroller	ATmega328P
Operating Voltage	3,3 V
Analog Input Pins	8
Clock Speed	8MHz
UART Port (serial)	1
Dimensions	27,94 x 66.04 mm
Analogue to Digital Converter	10-bit

The ARDUINO Fio was developed to work with the wireless communication platform X-BEE which operates at $3,3V$. That is why the Fio usually works at that voltage. A closer look at microcontroller's datasheet [ATMEL, 2009] shows the option of elevating the operating voltage to 5 volts. Since the X-BEE communication will not be used in this work it was decided to power the Fio with 5 volts. Which is provided by the USB to Serial connection used for programming the ARDUINO and for serial communication.

The structure of data acquisition program is shown in the Figure 2.5. An interrupt service callback occurs every 769μs (1300 Hz). This is where the analogue readings are made. In the main loop sequence three events take action if boolean variable is true. The boolean is set to true at the end of the interrupt sequence. The 4×10-bits are rearranged into a 5 bytes sequence. This process consists of taking the least 2 significant bits of each reading and place them in a fifth byte to be sent, as in Figure 2.6. These 5 bytes are placed in a array and is added another one, the ending message byte. All 6 bytes are sent through the serial port to the main system. The configurations of the serial communication are as shown in Table 2.4. At the end of this process the boolean variable set set to false. If the boolean is false at the beginning of the loop, nothing is done.

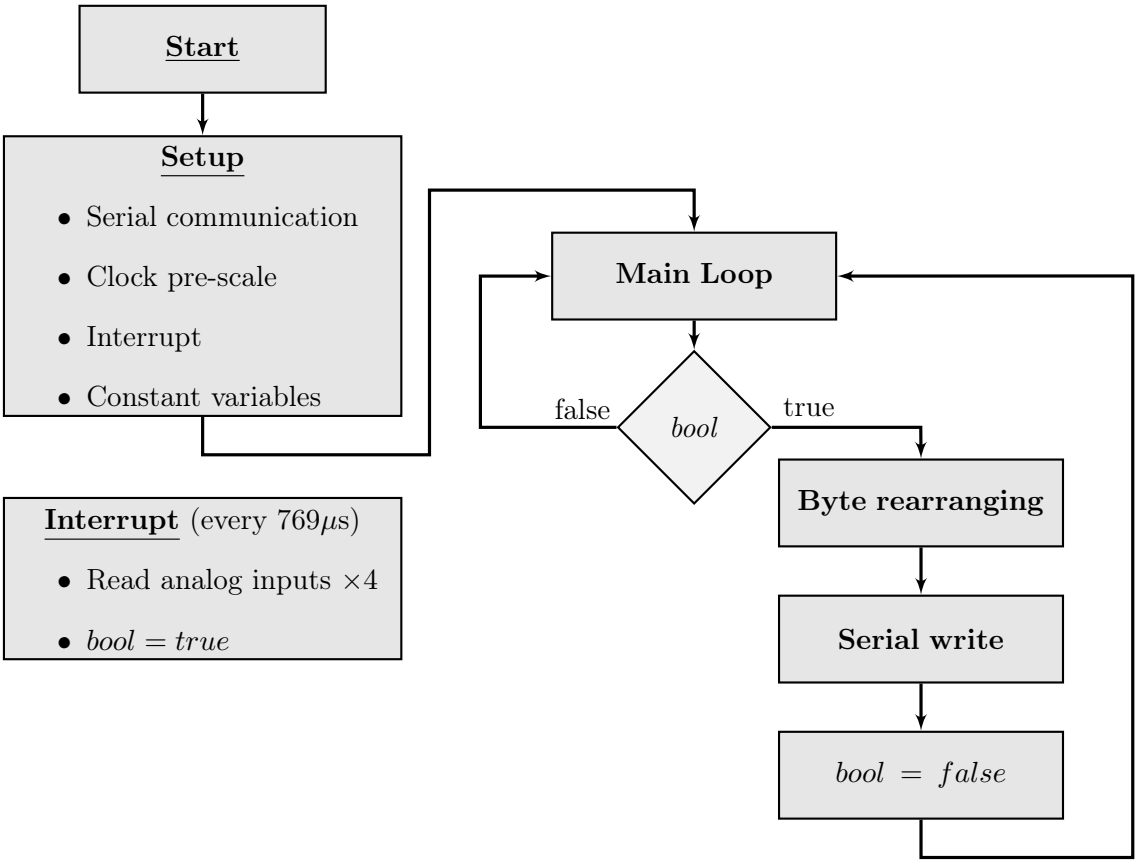


Figure 2.5: Schematic of Fio's program.

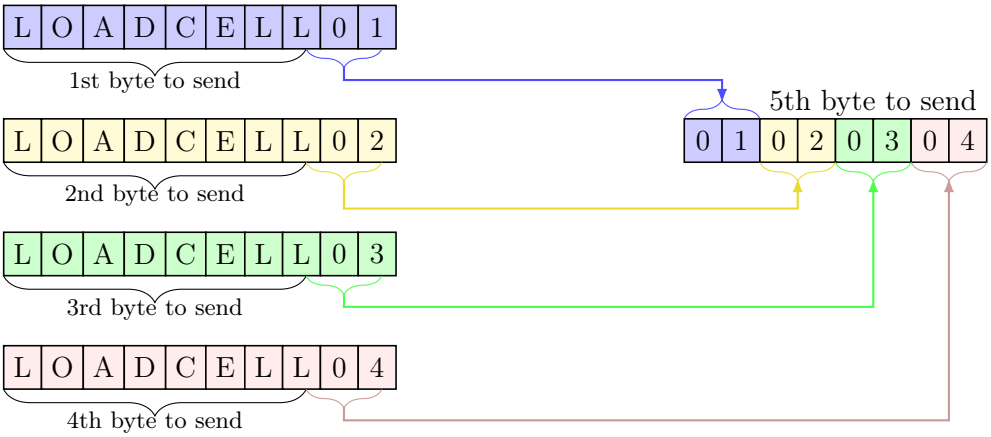


Figure 2.6: Byte rearrangement for the serial communication.

Table 2.4: Serial communication configuration used by ARDUINO Fio.

Parameter	**Value**
Baudrate	115200
Data Bits	8
Stopbits	1
Parity	(none)
Handshake	(none)

2.1.3 Signal conditioning

The maximum output of the load cells when powered at $5V$, is only $10mV$ (2.1); because of that it is necessary to amplify the signal. To ensure that the output is reliable, these conditions are needed for the amplification:

- Differential input
- High gain
- Low noise
- Low offsets
- Single power supply
- rail-to-rail output.

After a careful analysis of different amplifiers, the chosen one was the *Texas Instruments®*'s instrumentation amplifier INA 122. It complies with all the prerequisites specified above and it has a very low quiescent current. It is also indicated for industrial sensor amplifications of bridges, RTD and thermocouple [INSTRUMENTS, 1997].

To prevent the overload of the cells and the saturation of the amplifiers, the voltage output of the amplifier when the nominal loads are sensed, is $4,5V$. With this setting and the Rated Output, the needed gain is 450 (2.2).

$$G = \frac{4,5}{0,01} = 450 \tag{2.2}$$

Using (2.3) and (2.2) the gain resistor R_G for the amplifier is $449,43\,\Omega$ (2.4). The final resistor in use is $453\,\Omega$, with a 1% error.

$$G = 5 + \frac{200k}{R_G} = 450 \tag{2.3}$$

$$R_G = \frac{200k}{450 - 5} = 449,43\,\Omega \tag{2.4}$$

Working with Fio's pin layout, the schematic uses the Fio's power supply at $5V$ to power the load cells and the amplifiers. A capacitor, in each amplifier, is added to ensure

a stable power supply. To prevent aliasing, a first order low-pass filter is needed. The cutoff frequency is set to 500Hz (2.5) while expecting a sample frequency of 1 kHz or higher.

$$f_C = \frac{1}{2\pi \times RC} = 500\,\text{Hz} \tag{2.5}$$

The chosen pair of resistor and capacitor is $R = 1k43\,\Omega$ with $C = 220n\text{F}$ respectively. This results in a cuttoff frequency of 506.15 Hz (2.6) which is a fair result for this application.

$$\frac{1}{2\pi \times 1k43 \times 220n} = 506.15\,\text{Hz} \tag{2.6}$$

The circuit for the amplification and filtering for a single load cell is visible in Figure 2.7 where R_F and C_F are the chosen pair of resistor and capacitor and C_P is the capacitor for stabilizing the tension given to the amplifier.

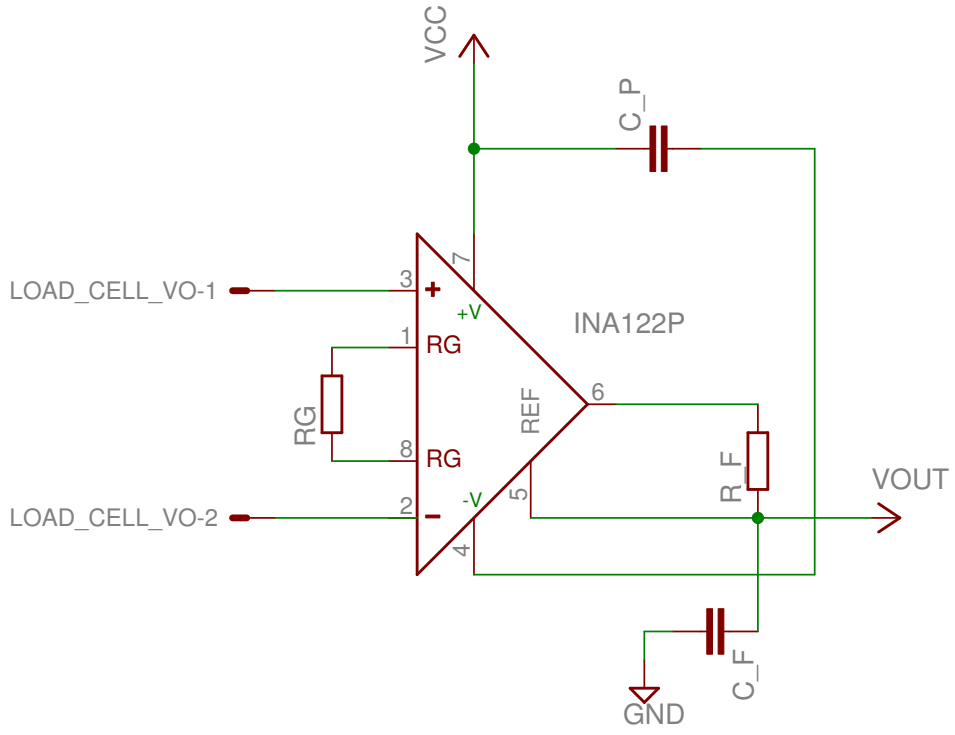


Figure 2.7: Amplification circuit for a single load cell.

Using the EAGLE PCB design software, a full scheme is design, appended in Appendix B. After this step, a 2 layer board is made to fit to the ARDUINO Fio's dimensions and pin layout (Appendix A). Most components are placed in the top side of the board, this side is facing the ARDUINO. On the bottom side is installed the connection pins for the load cells and the four capacitors for stabilizing the power source.

A ground plane is placed on the bottom layer of the board and on some part of the top layer. A power plane is also added to a part of the top layer for better distribution of the $5V$. These two sets of planes help reduce the electric noise and high frequencies interference. The outcome is a compact amplification shield as seen in Appendix C.

The board is manufactured by chemical etching and so it needs to be drilled. Because each board has over 100 drill holes, the board is drilled with a CNC. To guide both the CNC's grip and the etching 4 circular markings are added to the layers. The CNC that engineered the component's holes and via points is a 3-axis MIKRON VCE 500 present at the mechanical shop at the Mechanical Engineering department.

The assembling and soldering are done manually with the help of a professional soldering station and resulted in the presented board in Figure 2.8. After this the board is inspected, tested, cut to the correct proportions and covered in protective varnish.

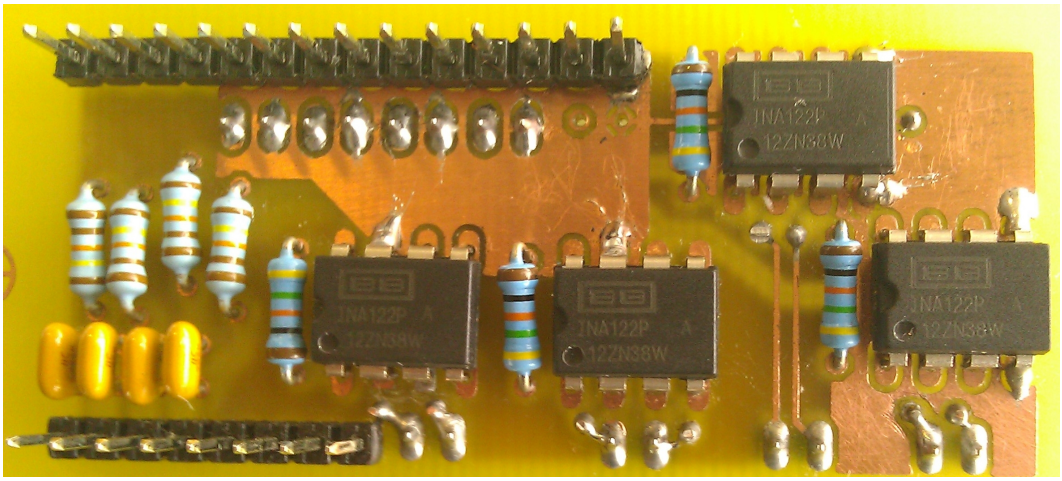


Figure 2.8: Load cells signal conditioning board.

2.1.4 Load cells's calibration

The main objective of using load cells is to know the force exerted in the different points of each foot. For this a calibration process is needed. A simple test was done to check if the cells responded in the same way. The cells were placed under a rectangular plate and different weights were placed in the center. The results showed that the increase of values shown is close but their starting points are very different. It is also noticeable that different load cells give different values when free of any weight.

Calibration curves are needed for each individual cell. For this reason, a universal testing machine is used, model AGS-X $10kN$ by Shimadzu Corporation shown in Figure 2.9. During this process all the cells were numbered and marked for future reference.

Figure 2.9: AGS-X by Shimadzu Corporation.

In the calibration process, each load cell is taken to its maximum nominal load while recording the values of force, voltage at ADC entry and 3 ADC values (10-bits each). As visible in Figure 2.10, the behaviour is mostly linear for all cells. With this data, a linear regression is done to get the calibration curves. The linear regression's coefficients for the ADC values are shown in Table 2.5, as well as the coefficient of determination for each cell. The first 4 cells are LBS-5 and the others are LBS-10.

Table 2.5: Load cells linear coefficients ($x = c_1 \times y - c_2$) and their coefficient of determination (r^2).

Nº of the load cell	c_1	c_2	r^2
1	0.0265769	-0.8356859	0.9999565
2	0.0262359	-4.3127300	0.9999443
3	0.0271632	-1.7507315	0.9999672
4	0.0270209	-3.4972975	0.9998818
5	0.0616719	-0.0399251	0.9998305
6	0.0593637	-0.9739984	0.9998724
7	0.0615672	-1.5177286	0.9999784
8	0.0627200	-0.8477938	0.9998996

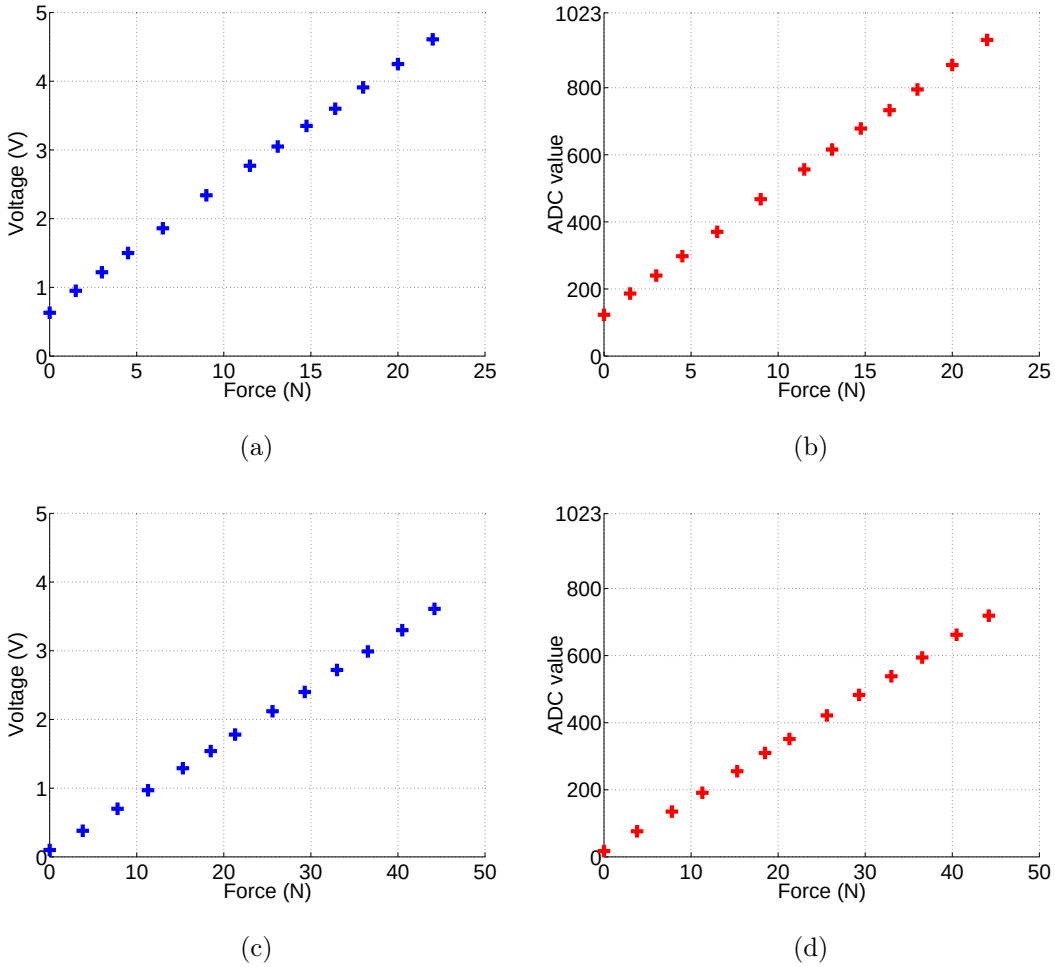


Figure 2.10: Calibration curves: (a) and (b) load cell nº 4 tension and ADC curves respectively; (c) and (d) load cell nº 8 tension and ADC curves respectively.

2.2 Software

This section presents the software platform and libraries that are used in this work and how they communicate among them. The ROS platform allows for great control over the processes and a direct connection with low-level hardware. The OpenHaptics API is needed for the interface with the haptic device. This Library gives control over the force feedback and the device's kinematics. Figure 2.11 shows the OpenHaptics API integrated into a ROS module. Because of these different systems, the different communication protocols are also explained in this Chapter.

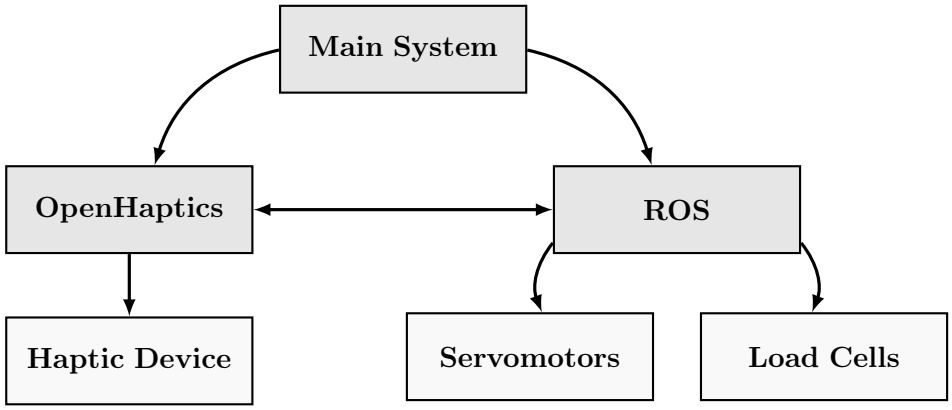


Figure 2.11: Interactions among the developed software.

All the software used in this work is present in the main system. The main specifications of this system are as fallow:

- Ubuntu 10.04 LTS
- Linux kernel 2.6.32-47-generic
- ROS Fuerte
- OpenHaptics API 3.0
- Intel Corp. Core(TM) 2 Duo CPU E8400 @ 3.00GHz
- Samsung HD103SJ 931GiB (1TB) SATA

2.2.1 ROS (Robot Operating system) platform

One of the objectives of this work is the implementation of the ROS platform. This is a open source software toolkit for the development of robots. Used for control of several robots, like Nao and Robonaut 2, ROS stands out as a modular tool with several utilities. Here are some instances of ROS that are needed for the PHUA project:

- Modular software development
- Free and open source platform
- Supports *C++* language
- Supply tools for code implementation and testing
- Allow *peer-to-peer* communication between different modules

The modular software development gives the chance to separate small pieces of a big problem. This is particularly important when working inside a bigger project, like this work in PHUA. This modularity is able by having different modules or *packages*. These are parts of software that can communicate with each other but still operate individually.

Each module has *nodes*, executables compiled with the written code; at the moment, ROS fully supports *C++* and *Python*. Any number of *nodes* can be created in each module.

To communicate among *nodes* two approaches are possible. The first is *message*; these possess a *topic*, a name that identifies it, and different fields, like headers (*frame*, time-stamp, etc.), standard variables and/or arrays of these for data storage. Any *node* can create and publish a *message*, ROS standard or user defined, and it is available for any *node* to subscribe. The second type of communication is a *service*. The concept of these is request and response. One *node* sends out a *service* request and waits for the response. The response comes from any *node* that subscribes to the *service*. Much like the messages, the services can have multiple standard variables and arrays on both the request and the response.

With this kind of interlinking, the PHUA can have a module for commanding the humanoid robot, one for the haptic device, one for the force generation algorithm, etc. Both methods of communication use persistent, stateful TCP/IP socket connections for the most part. This allows the expansion the main system. Using TPC/IP communications is possible to connect two or more systems via Ethernet and have different *nodes* running on them while *messages* run in the socket connections. A solution like this may be applied if a certain *node* uses a lot of computer power or if hardware restrictions exist.

ROS is designed to allow users to interact with low-level hardware control as well as complex algorithms and interactive interfaces. To help reduce the complexity that brings the interaction of different *nodes* working at the same time, ROS provides a utility called *roslaunch*. This allows the developer, with a single configuration file, to launch (execute) different *nodes* at once. A launch file can execute *nodes* from different modules and give each of them parameters and arguments. This way, setting variables names or values of variables can be done outside of the program's scope avoiding recompilation. The downside of this tool is that it doesn't guarantee the order of the *node* launching for one launch file. If there is a necessity of executing nodes in a certain order, multiple launch files are needed [Quigley et al., 2009].

The utilities like *RViz*, *rosbag* and *rosdep* give programmers extra tools for developing software. *RViz* is a visualization tool that allows for fames, point arrays and other geometries to be displayed and even interact with them. The *rosbag* utility gives the user the possibility to record messages and play them later on. This is particularity helpful when processing data off-line. These two utilities are very useful for the PHUA. *RViz* can represent the structure of the robot and the values of several sensors. The *rosbag* utility can be used for recording the messages that circulate among modules in *bags* (haptic device's state, robot's state); later, this data can be used in a learning by demonstration algorithm or a visualization by *RViz*. The *rosdep* and other *ros*-like tools give the user control the dependencies, compilation and directory usage of the modules.

ROS is distributed under a *BSD* license (permissive free software license) and has a large online community. A very well documented code and solutions help developers to create their code faster. This makes ROS a very flexible platform capable of being implemented in any robot.

2.2.2 OpenHaptics toolkit

The OpenHaptics toolkit is a proprietary programming solution provided by SensAble for building applications with haptics. Its latest version, OpenHaptics 3.0, is available for Windows, Mac OS X and Linux systems. Using the *C/C++* capabilities, this API has features like:

- Shapes
  - OpenGL primitives (polygons, points, and lines)
  - Custom/extension
- Force Effects
  - Constant (e.g. gravity)
  - Viscosity, 3D friction
  - Spring
  - Custom/extension
- Surface Material Properties
  - Friction
  - Stiffness and damping
  - Front/back faces
- Deformable Objects
  - Hooks for third-party integration

The toolkit has 4 main parts: the QuickHaptics micro API, the Haptic Library API (HLAPI), the Haptic Device API (HDAPI) and the PHANTOM Device drivers. These are depicted in Figure 2.12 by level of control over the haptic device. The QuickHaptics API gives the programmer control over the haptic rendering and events occurring at the device. The HLAPI gives the user a high level control of the force rendering and it targets OpenGL API programmers. Someone familiar with OpenGL API might have difficulties to implement haptics into their application but HLAPI allows it to be done easily.

The HDAPI is the low-level class that accesses and controls to the haptic device's capabilities. It is better used by programmers that are familiar with haptic rendering, frame transformations and the mechanics of the haptic device. It gives the possibility to control the rendered forces directly and change the drivers configurations in runtime. HDAPI is the main API used in this work because it allows direct access to the state of the haptic device.

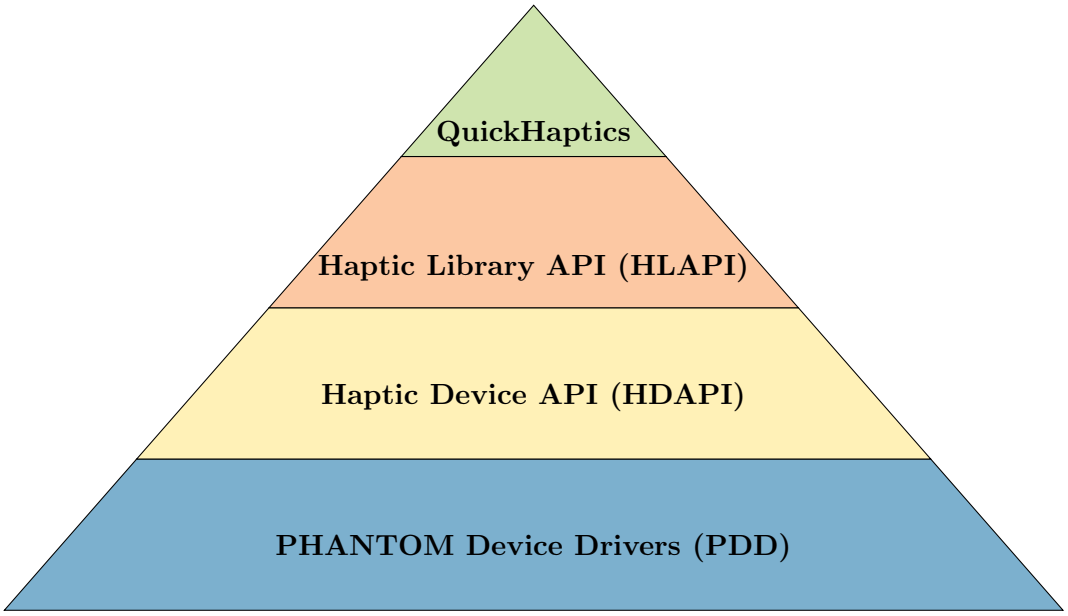


Figure 2.12: OpenHaptics toolkit structure.

The control loop used for calculating the forces and sending them to the haptic device is known as the servo loop. This is a tread-safe process. It is automatically done by HLAPI if necessary but it doesn't guaranty effectiveness of it. This one needs to have a 1 kHz or higher frequency to be kept stable [SensAble, 2008a,b]. Lower frequencies cause the device to tremor and kicking (rapid and uncontrollable force increase).

The communication between the OpenHaptics and the device in itself is done via *FireWire*. This, with the PHANTOM drivers, allows for more than one device to be used in the IEEE-1394 bus. Unfortunately, the OpenHaptics 3.0 does not support multiple devices. Another issue with this API is the necessity of the 2.6 Linux kernel for the PHANTOM drivers. Both problems are expected to be resolved in 4.0 version.

Chapter 3

Robot's Tele-operation

This chapter explains the developed software modules for this work and how they interact with each other; Figure 3.1 illustrates that interaction.

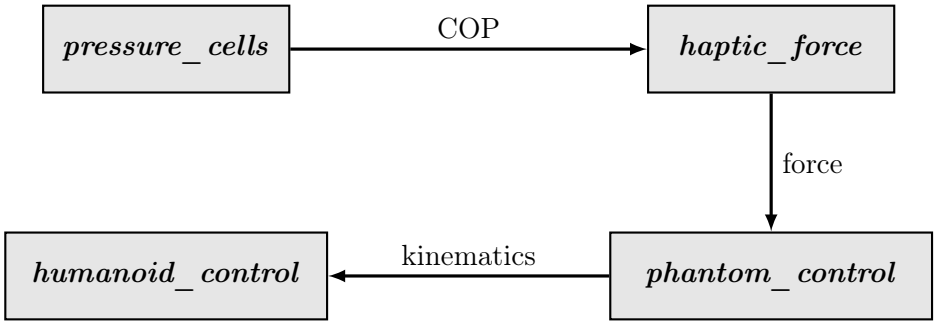


Figure 3.1: Interactions between the developed software modules.

The *pressure_ cells* module is in charge of receiving the ADC values from the serial port and convert them into forces and COP's. The COP's are then transmitted to the *haptic_ force* module. This module is one of the central ideas for this work. The idea is to have a module that translates sensor information into force feedback. Load cells are used in this work but other sensors like inertial ones can be adapted into this module. After this the force values are given to the *phantom_ control* module, which serves as a control and update service of the haptic device. The received forces are updated on the device and its kinematics and button status are retrieved to control the robot. The state of the device is published and the subscriber is the *humanoid_ control* module. The function of this module is to control the servomotors and update its state to a message.

All messages described in this Chapter are to be recorded into a *bag* (using *rosbag*) for later analysis. In future works, they can be used in learning algorithms.

3.1 Module *pressure_ cells*

The *pressure_ cells* module's main purpose is to retrieve the ARDUINO Fio's information and publish it in ROS format. It also has the function of calculating the *center of Pressure* (COP) of each foot and the total COP if the feet are coplanar. This module has 2 *nodes* that run in parallel: one for receiving the ARDUINO's information

and another one for calculating the COP's. Since there are 4 load cells for each foot, 2 ARDUINO boards are used.

As explained in Section 2.1.2, the four 10-bit ADC values (0 to 1023) are rearranged to to fit in 6 bytes and are sent by serial port. The first *node*, named *arduino*, is in charge of waiting for the termination character in the serial port and reverse the rearrangement bit process to get the original values.

The module accepts various launcher parameters for different configurations:

- the serial device port that each ARDUINO is connected to
- to show or not the visualization markers
- which feet does each ARDUINO corresponds to
- the identification of the load cells that are installed in the foot.

After checking the parameters given by the launcher, the *node* sets a publisher ready to send a message with the force values. In then enters the main loop where it continually waits for the termination byte. If found, the bytes received before are rearranged into force values and published in their message. The identification parameters for the cells trigger a calibration routine which uses the linear regression coefficients in Table 2.5. This way, the published message of each *node* is the 4 forces being felt by the load cells. The *node* runs a loop at higher frequency than the expected one of the ADC readings (1300 Hz). This way there is a continuous stream of information.

Due to the use of two data acquisition units, two *arduino nodes* are launched. Each with their own set of parameters and published messages.

The second *node* is responsible for calculating the COP of each foot and the global COP, called *force_ cop*. The global COP is only calculated when the parameter for coplanar feet is set; this parameter is set by the user in the launcher file *pressure_ cells.launch* present at the Appendix D. Much like the previous *node*, a series of parameters were set up for global control over the *force_ cop*:

- which message comes from which *presure_ cells* module
- set the correct frame identification for each foot
- show or not the visualization markers
- if the feet are coplanar.

By subscribing to the *pressure_ cells*'s force message, and using the measurements in the Table 2.2, the COP is calculated as in Equation 3.1.

$$C\vec{O}P = \frac{\sum_i \vec{r_i} \times f_i}{\sum_i f_i} \tag{3.1}$$

The $\vec{r_i}$ is the coordinates of the ith load cell in the feet and f_i is the correspondent force. This equation results in a $C\vec{O}P$ with the coordinates in (x, y) axis. A consequence is the impossibility to calculate the general COP from the COP's values of each feet.

This is why the general COP needs to be calculated with all forces and coordinates in the same referential.

The operation of this *node* is illustrated in Figure 3.2. It begins with the setting of the cells coordinates in each foot as in Table 2.2. Follow, the initialization of the ROS *node* and only after this it's possible to set the ROS variables (transformations, publishers, advertisers, etc.). The ROS parameters given to the *node* by its launcher are checked to see if they are at their default setting. If not, internal variables are set to the new values or names. These last variables ($ok1$, $ok2$, *visualization*, etc.) control the global operation of the *node* during the ROS loop. The setup ends with the setting of the advertising and subscribing variables the handle the ROS messages to be sent and received respectively. The advertisers are in charged of publishing the COP messages with the calculated vector. Three advertisers can exist in this *node*: one for each foot, identified by left or right, and the global COP if the launcher gives the correct parameter.

The loop will continue while the *ros::ok* variable is true, this happens while no order of shutdown is given to ROS. In this loop, 4 events take place: the first three are dependent on the global control variables.

The condition that is first checked is the *level* parameter, this would mean that both feet are at the same level. In parallel with this, the $ok1$ and $ok2$ are also verified; being true means that the correspondent callback ran in the last iteration of the loop. If these three conditions are met (*level*, $ok1$ and $ok2$), then the calculation for the general COP is activated. It begins with the lookup of the transformation from the right coordinate system to the left one. This way the coordinates of the right foot's load cells are transformed in to the left foot system. Now that all the cells coordinates are on the same system, using the Equation 3.1, the general COP is calculated and its coordinates are published in the correspondent message.

The second and third event is the publishing of the visualisation markers. The first to happen is the individual COP markers, one marker for each foot. This occurs if the *visualisation* parameter is set. The other one needs the verification of both the *visualisation* and *level* parameters. If both true, the general COP marker is published.

The last event of the main loop checks if there are new messages from the subscribed topics. If they are, the correspondent callbacks are activated. The callbacks present in this *node* are activated if a new force message is received. Since two messages are subscribed, two callbacks exist. They are almost identical, one is for the left foot and the other for the right one. Each of the callback uses Equation 3.1 to calculate the COP for the correspondent foot. Before exiting the callback, the COP is published in its message.

After these 4 events, the loop waits the necessary time to keep a stable frequency.

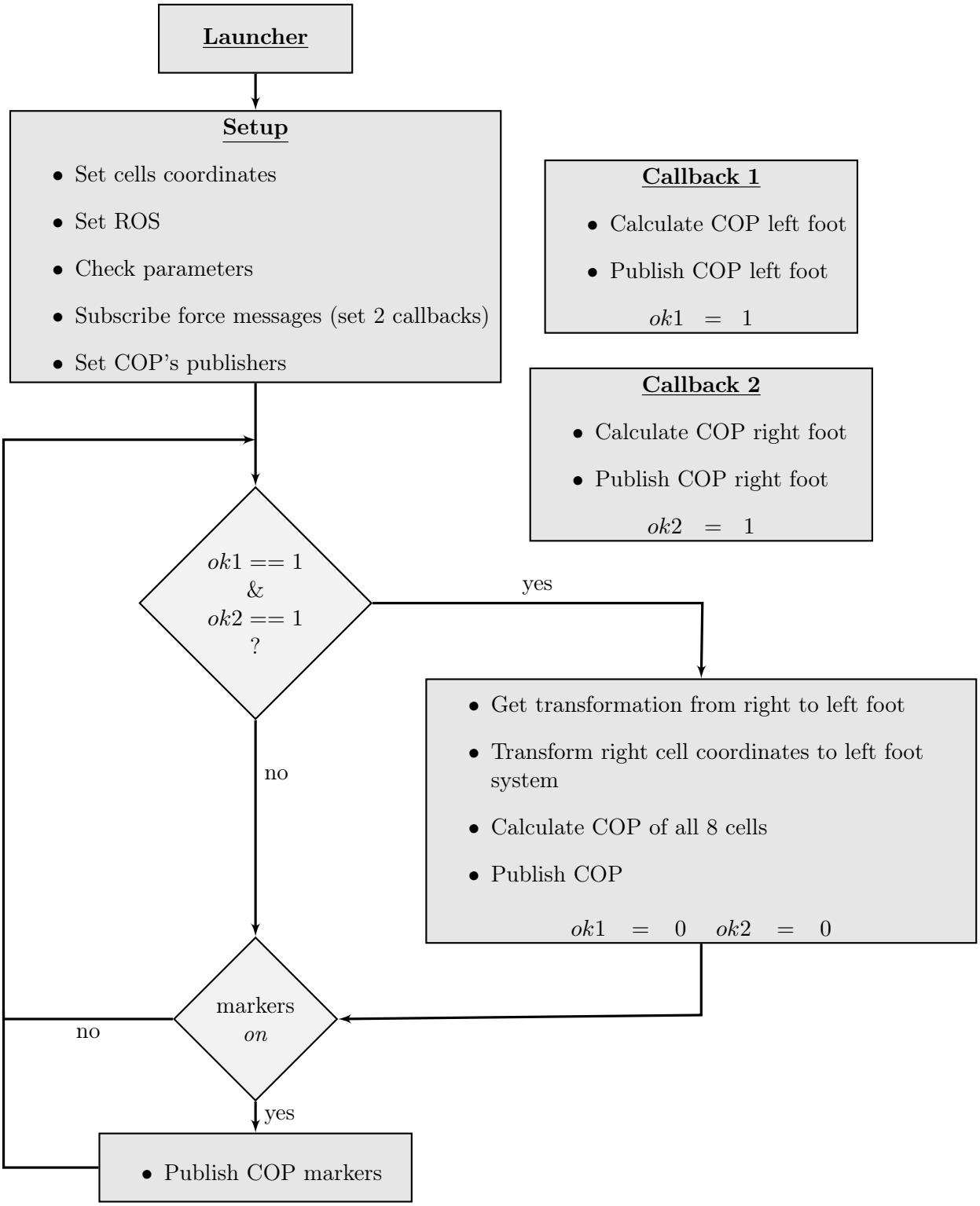


Figure 3.2: Simplified schematic of *force_ cop* node.

3.2 Module *haptic_force*

The idea of the *haptic_force* module is to translate sensory information into force feedback. In this case, the information comes from the load cells.

The used information is the calculated COP's. These are translated to force feedback with a simple linear equation represented in Equation 3.2. The equation is a direct correspondence from the limits of the COP, COP_u^{max} and COP_u^{min}, to the limits of the force feedback, $F_{Phantom}^{max}$ and $F_{Phantom}^{min}$. Where u indicates the axis ,x or y, F_u is the force exerted in the u axis of the haptic device and the COP_u is the current value of the u component of the COP. The used COP is the general one; using the information of the 8 load cells gives a wide dynamic range to the force feedback. The difference between the COP while the robot is in its home position (similar to the human's neutral anatomical position) and the current COP gives the direction and magnitude to the force feedback. The direction is from the home position COP to the current one. A dead zone is added to the generation of the haptic force to prevent positive feedback; if a component of the COP produces a force where its absolute value is lower than d_u N that force is reduced to zero. If is higher, the Equation 3.2 is applied. This gives the user the unbalance that the robot is in.

$$F_u = \frac{F_{Phantom}^{max} - F_{Phantom}^{min}}{COP_u^{max} - COP_u^{min}} * COP_u \mp d_u \tag{3.2}$$

The function of the user is to correct the unbalance back to a stable position. This happens by opposing the force being exerted in the haptic device.

A single *node* handles all the above functionalities, called *cells_force_gen* and its structure is simple. It starts with the ROS initialization and then it sets the COP message subscriber and force message advertiser. Like before, the subscriber has a callback trigger where the translation happens. The variable that holds the force values is updated in the callback, but is only sent inside the main loop. This allows a greater control over the setting of each force component. Before the main loop starts, 2 variables are set: the loop's frequency and the third component of the force. The frequency of the main loop, like in the previous module, is set higher than the origins of the message subscribed. Because the COP only sets 2 force components, the third is set to compensate gravity. Using the apparent weight of the tip, the vertical component of the force ensures that the user doesn't have to hold the device in place.

3.3 Module *phantom_control*

The module *phantom_control* uses both ROS and OpenHaptics. This is the module that controls the device's force feedback and obtains its joint values.

Due to the programming structure of a typical OpenHaptics program, all the commands for getting and setting the device's state need to be done within the same servo loop. This means that the control of the device and communication to the other modules needs to be done in the same *node*; because of this, the *phantom_control* has only one *node*.

The *node* starts by initializing the structure that holds the state of the haptic device, which is composed of the following fields:

- *position* ([1 × 3] vector)
- *rotation* ([1 × 3] vector)
- *gimbals* ([1 × 3] vector)
- *thetas* ([1 × 7] vector)
- *temp* ([1 × 3] vector)
- *buttons* ([1 × 2] vector)
- *home* (boolean)
- *home_position* ([1 × 3] vector)

Visible in Figure 3.3, the *position*, *rotation* and *gimbals* fields are taken directly from the device. The *position* gives the X, Y and Z position of the tip in millimetres. The *rotation* and *gimbals* fields gives the angles, the first gives the 3 base joints and the second the give the tip joints. All these angles are stored in the *thetas* vector; this one is used for the kinematics. The *temp* holds the temperature of the motors in the base of the device. These values are used for integrity and security purposes. The *buttons* hold the state of both buttons present in the device. The *home* and *home_position* are used for setting a home position for the device. The boolean is only set if the device is in the home position. All these variables are published in a custom ROS message.

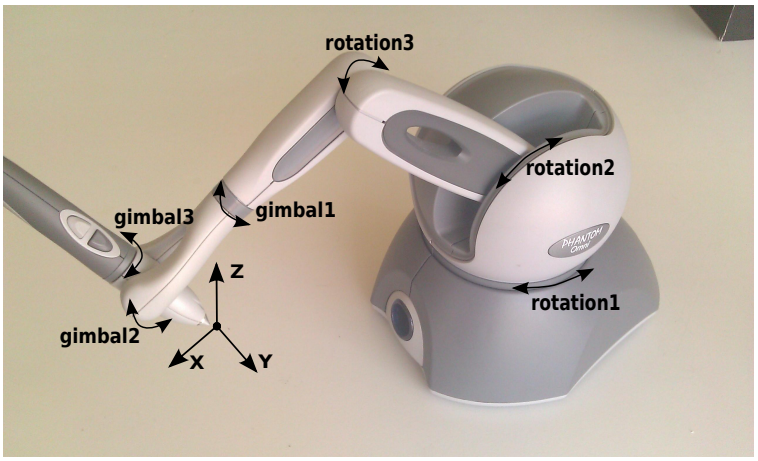


Figure 3.3: PHANTOM OMNI's kinematics.

The device is initialized by using the HDAPI is possible to start the communication with the device and control it at a low-level. Besides initializing the device itself, an asynchronous scheduler is set to manage the servo loop: it is a high frequency and high priority thread for sending the forces to the device and retrieving its state. To have a stable force feedback it needs to run at least at 1000 Hz. The scheduler begins by starting the communication with the device. It then gets the position, joint state, buttons state and motor temperature. It then checks if button 1 is pressed; if not, the process creates a force to take the tip to the home position. If the tip is already in the home position the

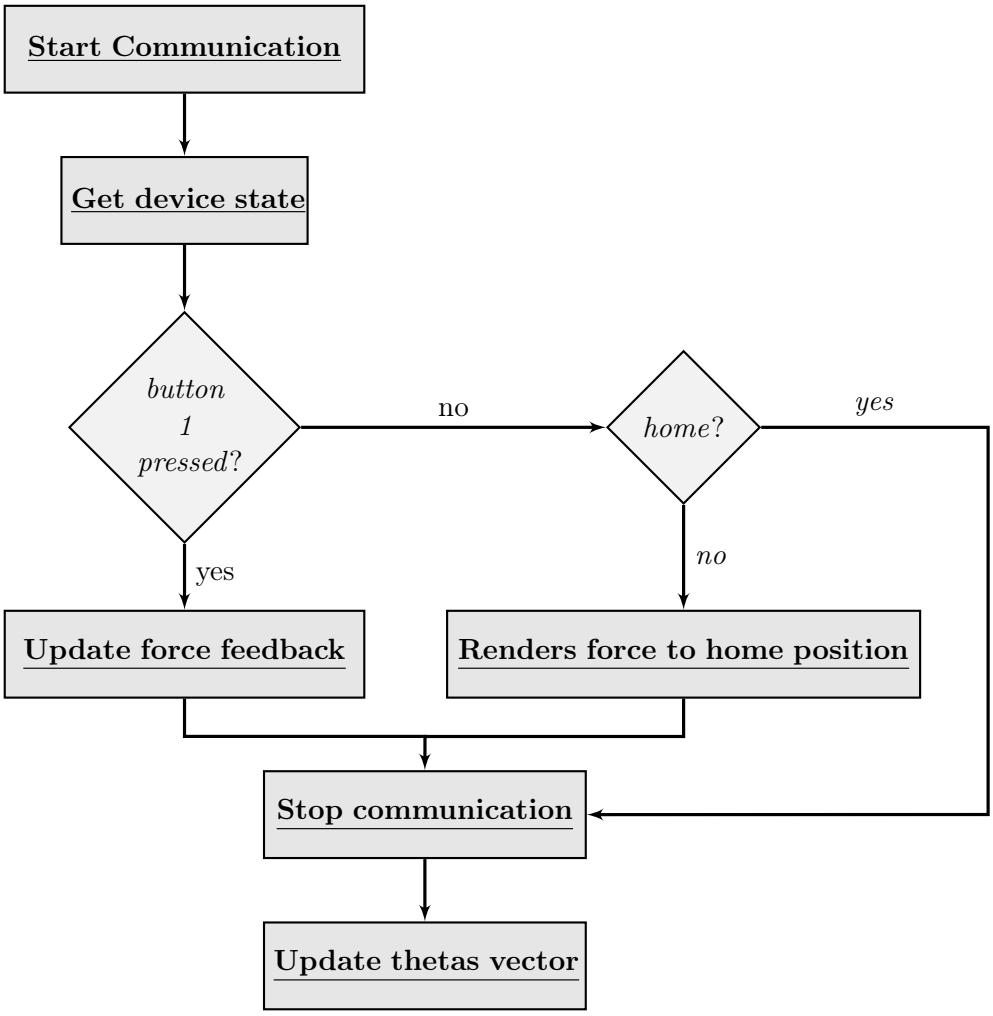


Figure 3.4: *phantom_ control* scheduler's callback scheme.

process continues. If button 1 is pressed, the force feedback is generated using the force message from the *haptic_force* module. After this the communication with the device is stopped. Before the callback exits the *thetas* vector is updated with the new values [SensAble, 2008a,b]. Figure 3.4 shows the schematic of the callback.

After the setting of the scheduler, the force feedback is enabled and the scheduler starts. If the *node* is unable to start due to an error, it stops and reports back to the launcher. The final step for the device settings is the calibration routine. This is an automatic calibration routine that checks if the device is already calibrated and if not, it takes actions to calibrate it. Usually the device is calibrated with a utility provided by OpenHapics before the *node* is launched.

The structure where the state is saved is then initialized with zeros. As soon as the main loop starts, the variables present at the structure are updated.

The next step is to initialize the ROS *nodes*. Like before, the subscriber, the publisher and the loop frequency are set. The published message is the state of the device. The *phantom_ control node* subscribes the force message that when received starts a callback. The callback consists of updating the structure with the required force. As in the previous

modules, the frequency is set so the *node* gets all the messages in the required time.

The main loop is dependent of the *ros::ok* variable. This allows for the *node* to exit correctly when the shutdown signal for ROS is given. The scheduler and the haptic device need to be properly stopped; this happens when the loop is stopped and before the *node* ends. The main loop is where the device's state is published. This only occurs after the device's callback is executed because a continuous message publication would consume too many resources. After this, button 2 state is checked; if pressed, the current position is set as the new home position. This allows for a more intuitive user experience where one can leave the command of the robot at any time. At the end of the loop, a ROS function checks if there are new messages and runs the callback. Figure 3.5 illustrates the *phantom_ control*'s program structure.

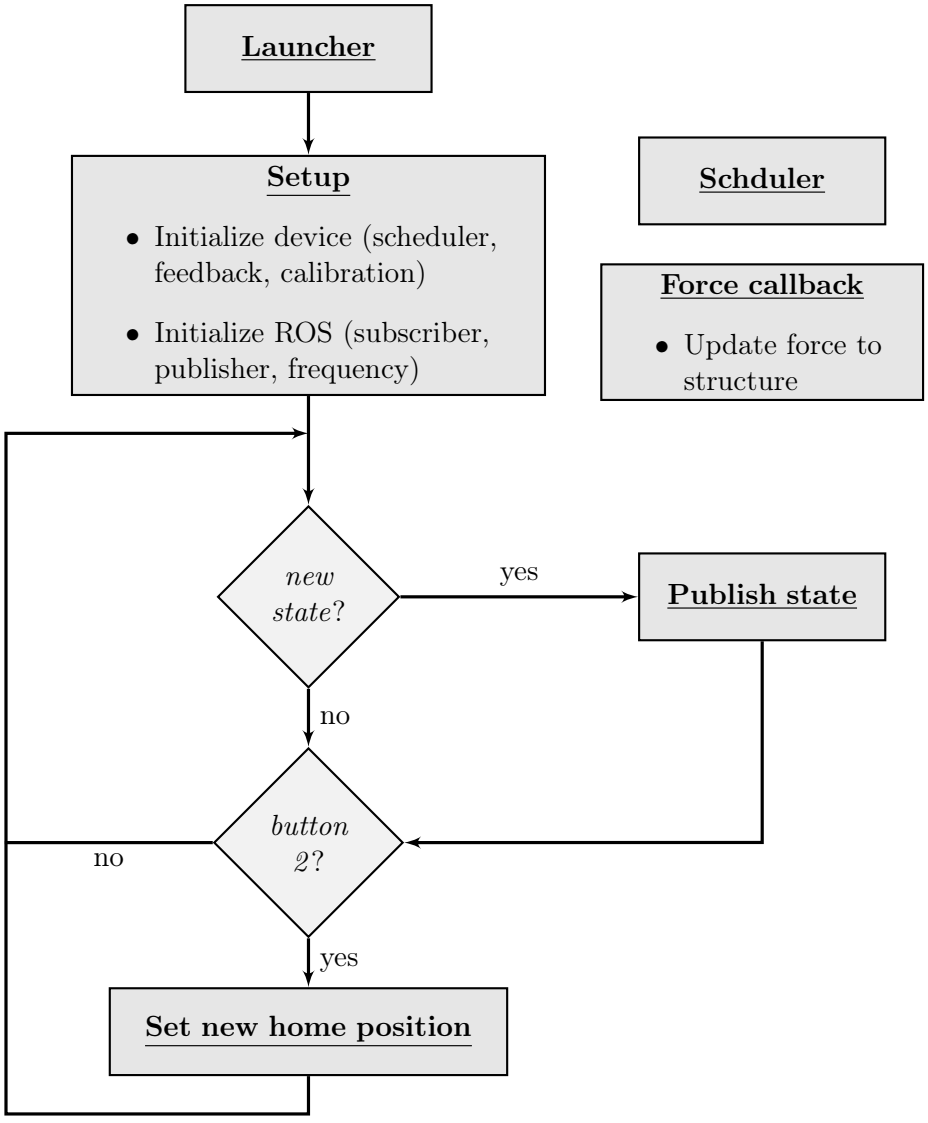


Figure 3.5: *phantom_ control* scheduler's callback scheme.

3.4 Module *humaniod_ control*

The module *humaniod_ control* has the function of commanding the PHUA robot's servomotors. The idea is to have a module that can communicate with the robot. Commanding the servomotors and getting their state are two prerequisites for this module and a *C++* class is developed for this purpose. This work is focused on the lower limbs and torso and, therefore, only those joints are used and commanded in this *node*. The rest of the joints can be implemented using the principles explained next.

Section 1.4.2 describes the *Bidirectional Serial Interface* and its protocol. In the work of P. Cruz [Cruz, 2012] some software was implemented to control the servomotors using this protocol. A low-level control was done as well as a joint-based control. Using the mentioned previous work, a new *C++* class was created to integrate the servomotor control with the modular idea of the present work. This class is integrated in the *humaniod_ control* module since it is not needed elsewhere. The class is capable of commanding each servomotor with just the identification number an its joint position or speed. The positions are given in degrees and converted to servomotor values before sending the byte word. The speed is set with an integer value between 0 and 255.

The main commands used in this class are the *MoveJoint* and *SetSpeedJoint*. They both have 2 input parameters: the identification of the joint and the value to be set. The first command is used to request a position change to a specific servomotor. The internal PD controller of the servomotor checks the current position and acts upon it. The servomotor response is a 2 byte word. In the case of *MoveJoint*, the response is the speed at with the movement is being done. If the communication isn't successful, the return is *0xFFFF* for both functions. The *SetSpeedJoint* sets the speed at which the servomotor must make its movement. This function does not move the joint, only sets how it will move. The return response is the current position of the servomotor. In both functions is possible to command all servomotors with the right identification. This is used to take all joints to the zero position or to set the same speed for all of them.

The structure of this *node* is illustrated in Figure 3.6. Like before, this *node* starts with the ROS initialization. A subscriber is set to receive the the message published by the *phantom_ control node* which contains the haptic devices's state. The publisher for the the humanoid is also defined, this state is published at the end of the main loop. The frequency is set at the same value of the *phantom_ control node*. Before entering the main loop, the class for controlling the servomotors is initialized with the corrected port for serial communication. During this process the speed of all the servomotors is set to 50. It is then used to set the robot to the home position; this is the correspondent position of the humans neutral position. This happens by setting all the joints to there zero value.

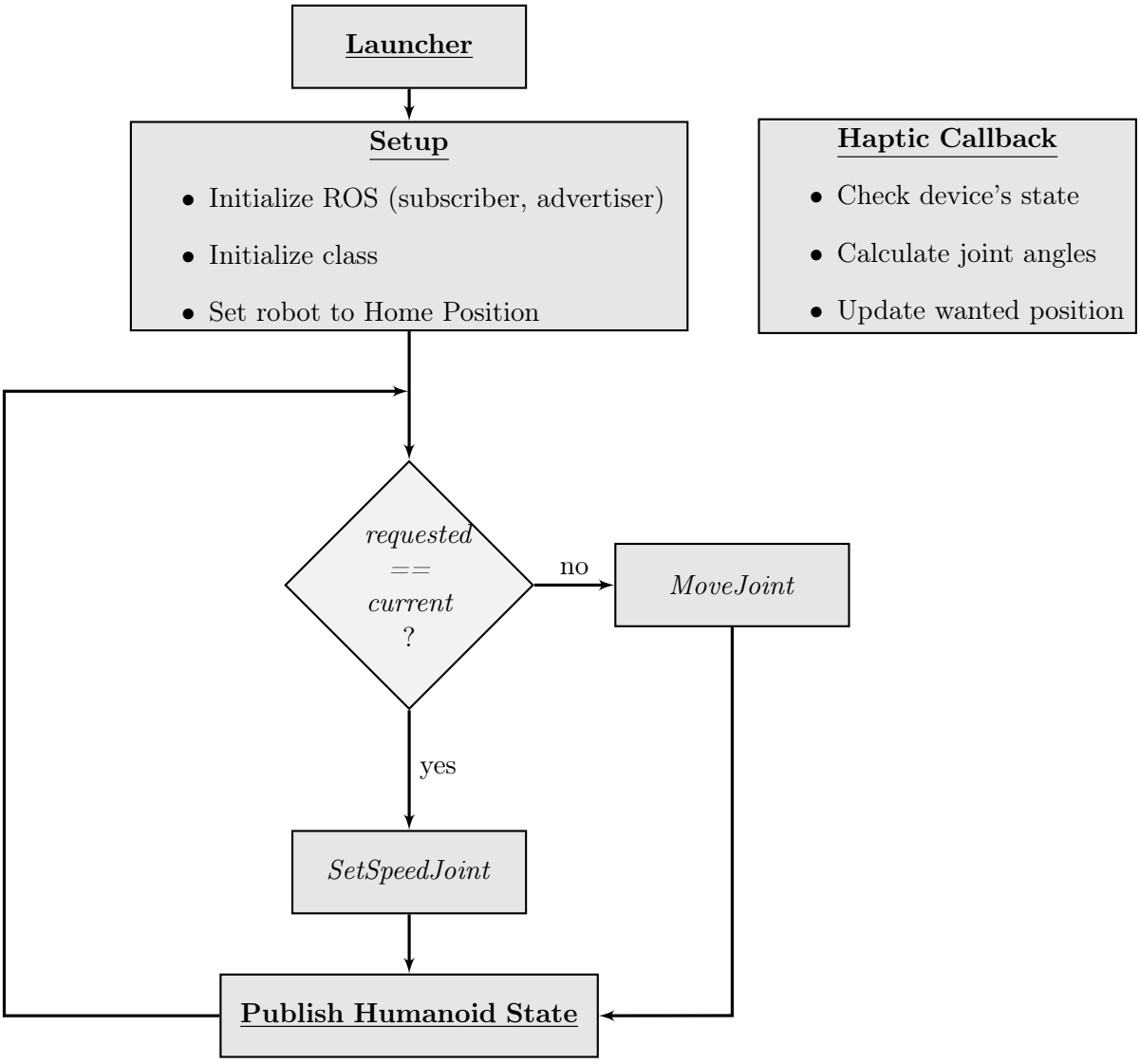


Figure 3.6: Program structure of *phantom_ control node*.

The main loop starts by checking all used joints to see if they are at the requested position, which is given by the haptic device callback that receives the state of the device and calculates the joint angels. These angles are the new requested position for the the joints. The calculation of the joints angles are explained further in Chapter 4. The requested position is compared against the current position. If they are the same, the *SetSpeedJoint* command is used to get the position and keep the speed updated. If not, the *MoveJoint* is used to give the move command to the servomotor. All the servomotors in the line go through this process.

After this, the state of the humanoid is published; all values, requested and current, speed and positions, are placed in the message. The main loop ends with the check for new received messages for the *phantom_ control* module.

Chapter 4

Robot's demonstrations

This Chapter describes the demonstrations that use the development detailed in Chapter 3. These consist of the tele-operation of the robot using the haptic device and in some cases, using the general COP to generate a force feedback for user interaction. In all the demonstrations, only the 5 DOF of each leg linked by the hip piece are used. During these demonstrations, the robot's feet are kept in constant contact with the floor at a distance of 14 cm from each other.

The command of the humanoid robot is done by the *humanoid_ control* module, holding the kinematics functions that control the movements. These functions are adaptations from the previous work [Cruz, 2012]. In the experiments where the haptic force feedback is active, as explained in Section 3.2, the force is generated using the difference between the COP in the home position and the current COP. In each of the demonstrations, a single axis from the haptic device is monitored; using the value of that axis and applying a scale factor to use the maximum of the effective force feedback workspace, a kinematic function calculates the joint values according to a parameter set by the launcher. This parameter's value is set by the user and it corresponds to the type of movement (demonstration) of the robot. The movements of the robot are controled by the haptic device's state, more precisely, it's position. Because the robot's movements are done solely in one plane the control is done in position depending on a single axis of the device's position.

All the data from the experiments is recorded into *bags* using the *rosbag* utility. This allows for offline analysis and visualization of the recorded informations; the results presented in following Sections are created from these *bags*.

Since the PHUA robot won't be performing walking motion, only balancing, a bench power source was used. It consists of a AC-DC power supply and a DC-DC power converter.The power supply is MASCOT Type 9821 with an output of $12V$ and a maximum current of $20A$. The power converter is a CC BEC Pro by Castle Creations capable of an adjustable output between $4.8V$ and $12.5V$ with a maximum peak current of $20A$. For the demonstrations the converter was set to the lower working voltage of the servomotors, $6V$.

The axis system for the robot used in this demonstration is depicted in Figure 4.1; this system is centred in the geometrical center of the right foot so that both the COP and the position of the robot are represented in the same referential. All the kinematics are done for the right leg and because all the movements performed in these experiments are symmetrical, the calculated kinematics can be applied to both legs. In all of the

demonstrations, the used arrangement of the load cells in the robor's feet is depicted in Figure 4.2 where the red cells correspond to the LSB-10 and the blue cells to the LBS-5.

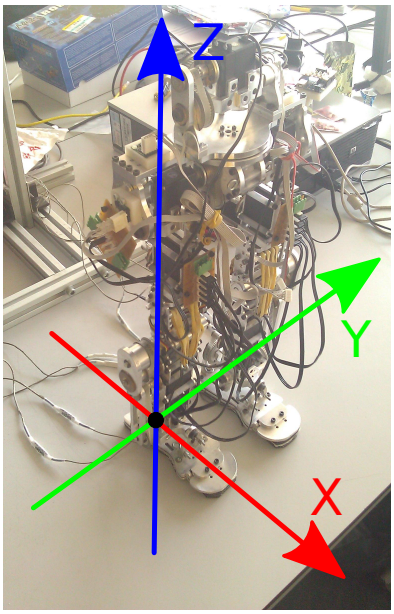


Figure 4.1: PHUA robot's axis system.

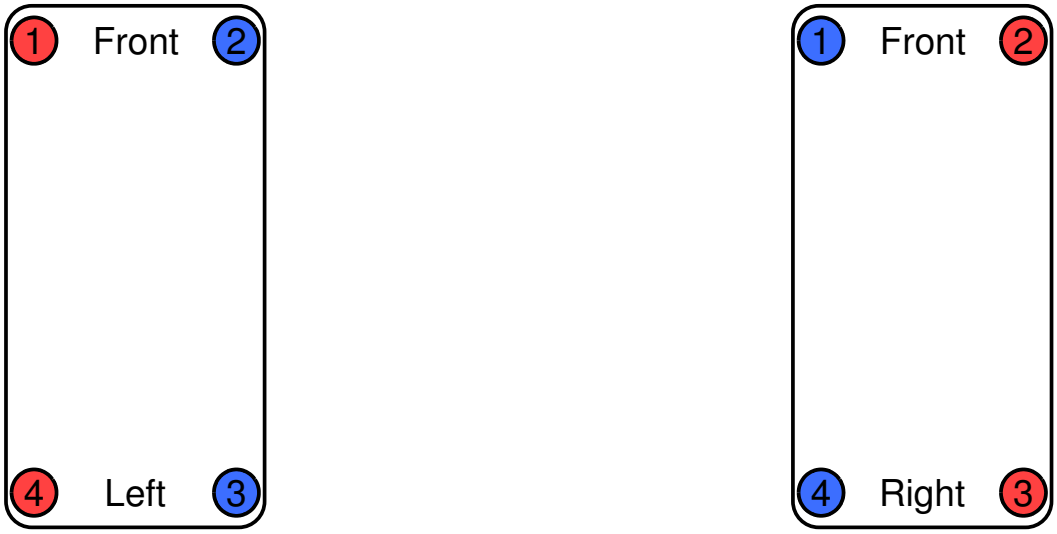


Figure 4.2: PHUA robot's axis system.

Two main launch files are created and used to keep the experiments simple to perform and to replicate. The file that is launched first has the purpose of finding the used ports automatically and publishes the transformations between each foot and their relation the the world. The second file calls other launch files that in turn launch each of the nodes necessary for the experiments. All this files are in the Appendix D.

4.1 Hip's vertical movement

This demonstration consists of the vertical movement of the hip; to ensure this, the legs only move in the XOZ plane. This happens by controlling three joints: Flexion/Extension of the ankle, knee and hip, where the knee joint is the only one to move along the X axis. The COP variations are expected to be minimal so the force feedback originating from the COP is not active. This experience aims to validate the control of the PHUA robot using the motion of the haptic device. The Z axis of the device is monitored and with its values commands the kinematics of the robot.

4.1.1 Inverse kinematics

Each joint is controlled according to the inverse kinematics presented below. The used kinematics is similar to the three links planar robot depicted in Figure 4.3 where the foot corresponds to the origin of the X and Y axis and the hip is represented by the P_W point.

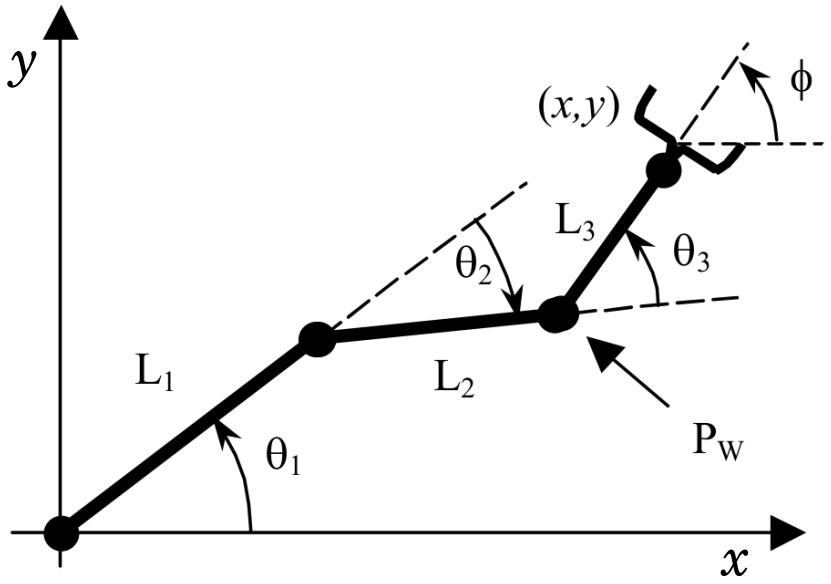


Figure 4.3: Three rotary joints planar robot [Santos, 2003].

The x and y coordinates of the P_W point can be expressed by the Equations 4.1 and 4.2 respectively.

$$P_{Wx} = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \tag{4.1}$$

$$P_{Wy} = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) \tag{4.2}$$

Developing these equations gives the solution for the inverse kinematics of the θ_2 joint angle given in Equation 4.3.

$$\theta_2 = \pm \arccos \frac{P_{Wx}^2 + P_{Wy}^2 - L_1^2 - L_2^2}{2L_1L_2} \tag{4.3}$$

Due to the resemblance to the human body's knee, the robot's knee joint was mechanically blocked and the servomotor assembled so this joint only bends backwards; according to the kinematics presented above and the physical limitations the redundancy of the θ_2 joint is eliminated resulting only in positive angle joints. The inverse kinematics of the θ_1 joint is obtained by further development of the Equations 4.1, 4.2 and 4.3 as shown in Equation 4.4

$$\theta_1 = \arctan \frac{P_{Wy}(L_1 + L_2 \cos\theta_2) - P_{Wx}L_2 \sin\theta_2}{P_{Wx}(L_1 + L_2 \cos\theta_2) + P_{Wy}L_2 \sin\theta_2} \tag{4.4}$$

To keep the hip always horizontal to the ground, the orientation of the grip present in Figure 4.3, ϕ, is needed and it depends only on the θ_1, θ_2 and θ_3 angles. This dependency results in the final part of the inverse kinematics for this demonstration and is given by the Equation 4.5.

$$\theta_3 = \phi - \theta_1 - \theta_2 \tag{4.5}$$

4.1.2 Haptic device and robot's motion

In the hip's vertical movement, the Z axis of the haptic device is used to control the height of the PHUA robot's hip. The control is done by making a correspondence between the Z axis values from the device to the P_{Wy} variable in the inverse kinematics detailed in the Subsection 4.1.1. An offset is given to the kinematics so that the *zero* of the haptic device's Z axis coincides with the home position of the robot. This way, the robot only starts to crouch if the user moves the tip of the device to negative values of the Z axis. Figure 4.4 illustrates the values of the Z axis during the experiment. Using the inverse kinematics function in the *humanoid_ control* module, the robot's actuated joints (ankle, knee and hip's Flexion/Extension) performed the path present in Figure 4.5 following the device's vertical movement; since both legs follow the same kinematics, only the right leg joints are represented. By using the time stamp present in every ROS message it is possible to represent multiples sources of information in the same graph, e.g. joint values and movement of the haptic device.

During the experiments, 2 movements of lowering and rising are performed; the first is done by moving the haptic device's tip down and up at a low speed. This movement corresponds to the first 20 seconds represented in Figures 4.4 and 4.5. Between the lowering and rising movements, the joints saturate because one of them, the ankle, reaches one of its limits stopping the motion of all the other joints involved in the kinematic. In both downward and upward movements, the joints behave at a constant speed with some steps near the beginning and end of each movement. These steps are also visually observable in the robot during the demonstrations. The reason for this phenomenon is that the servomotors reach the requested position before the new position reaches the *humanoid_ control node*.

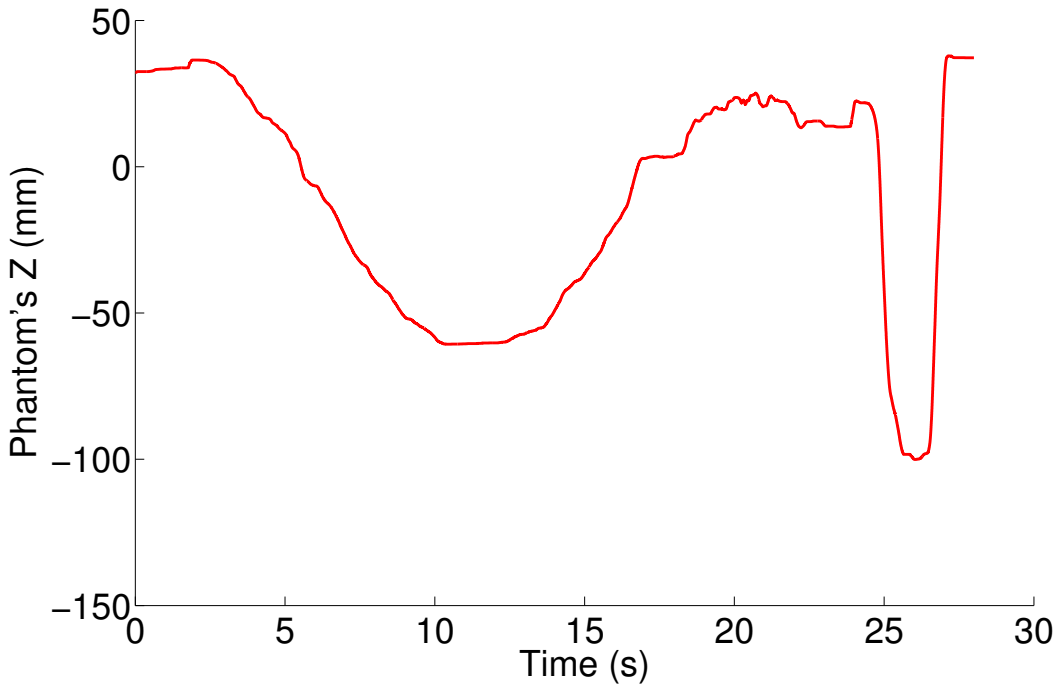


Figure 4.4: Motion of haptic device's Z axis during the vertical movement demonstration.

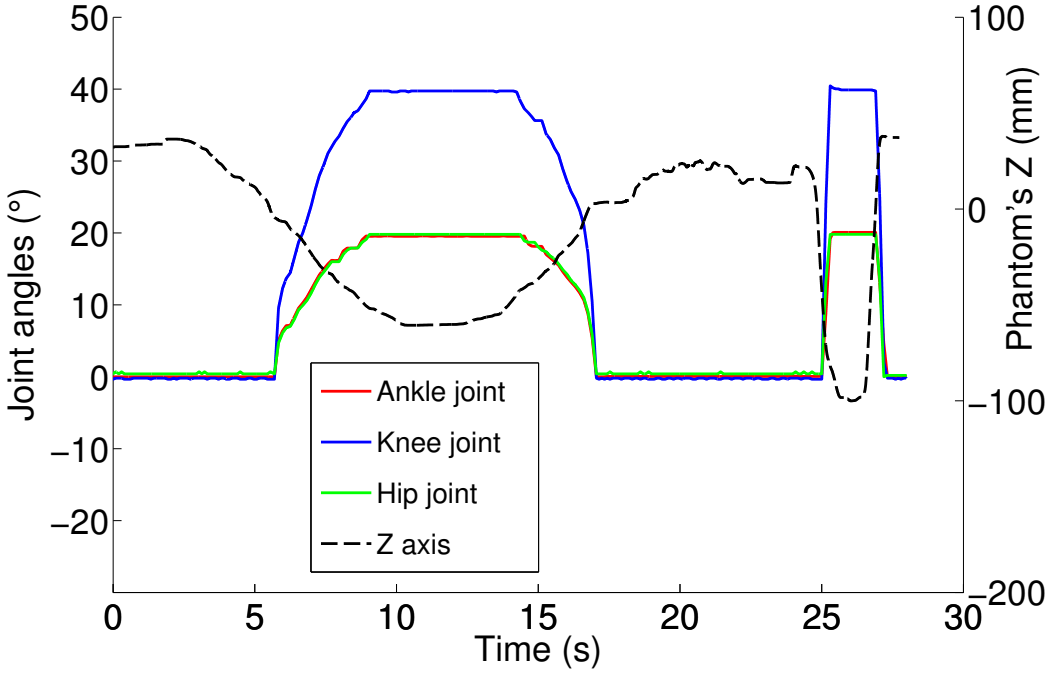


Figure 4.5: Motion of the actuated joints of the PHUA robot during the vertical movement demonstration with the Z axis movement of the haptic device.

The second movements are done using the same motion but at a faster speed on the haptic device's side. The servomotors quickly respond to the orders in a stable movement using the set speed for each joint. A small overshoot is perceivable on the knee joint but it has a fast stabilization. In both cases, fast and slow movements, the only joint that reaches its physical limits is the ankle Flexion/Extension joint, all other joints still have more range but the kinematics function prevents further movements once one of the actuated joints reach its limits.

In Figure 4.5, it is possible to see a delay between the motion of the haptic device and the servomotors during the fast movements. This may indicate either a slow control loop of the servomotors or a slow communication line between the main system and the servomotors. During the experiments, the CPU and RAM usage was monitored; with all *nodes* necessary to this demonstration both CPU cores were at 100% and over 80% of the RAM has being used. This issue is addressed in Section 5.2 where suggestions are made to prevent this problem.

4.1.3 Forces and COP

Using the information of the load cells stored in this demonstration's *bag*, it is possible to obtain the graphs in Figures 4.6 and 4.7. For a better understanding of the dynamic shifts of the forces, the movement of the haptic device is also represented in the graphs.

The oscillations in the transitions from the home position to the lowest position stand out as an uncertainty of the measurements but, when the robot's joints stop in any of the 2 positions, the measurements are stable and constant. This indicates that the system developed for the acquisition of the load cells values is capable of accurately reading their values in static positions of the robot. The hight and low values in the transition phases are explained by the dynamics of the movement, slacks in the belt transmission and elastic effects caused by the rubber soles can be the sources for some of the oscillations. Figure 4.17 illustrates the sum of all the cells in each; here is possible to see the full acceleration and inertial effects on the robot's feet.

An interesting observation can be done in Figures 4.6 and 4.7; most variations only occur in three of the four load cells, and the fourth cell value is caused by the compression from the bolts that hold the bottom and top parts of the feet; this last cell shows very little variations throughout the demonstration. This is most likely the redundancy of four contact points in a rigid plane like the PHUA robot's foot. Since a plane only needs three points of contact the fourth load cells doesn't sense more pressure unless the COP moves close to it. Even though this happens, there is no need to change the mathematical formulation of the COP (Equation 3.1) since it's weighted on the value of the force; a small force has little or none influence on the final value of the COP.

Figure 4.8 shows the sum of all the forces in each foot where is possible to see a stable measurement of the total normal force. Except for the oscillations caused by the inertial component of the normal force, the graphs show that the weight measured by the load cells is constant and accurate.

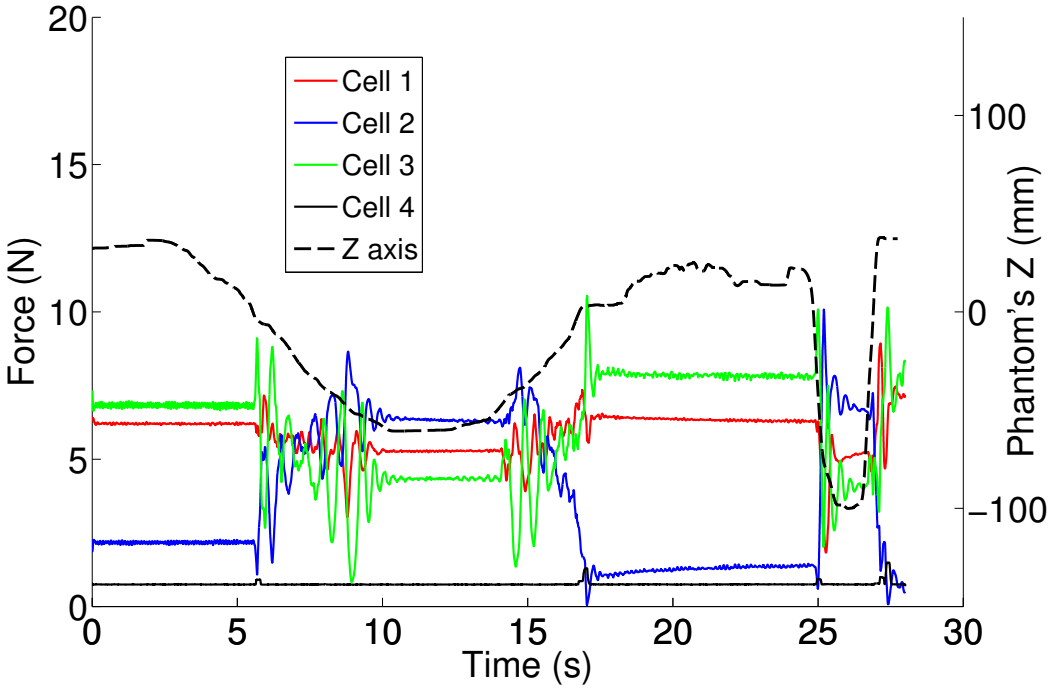


Figure 4.6: Forces normal to the right foot of the robot during the vertical movement demonstration with the Z axis movement of the haptic device.

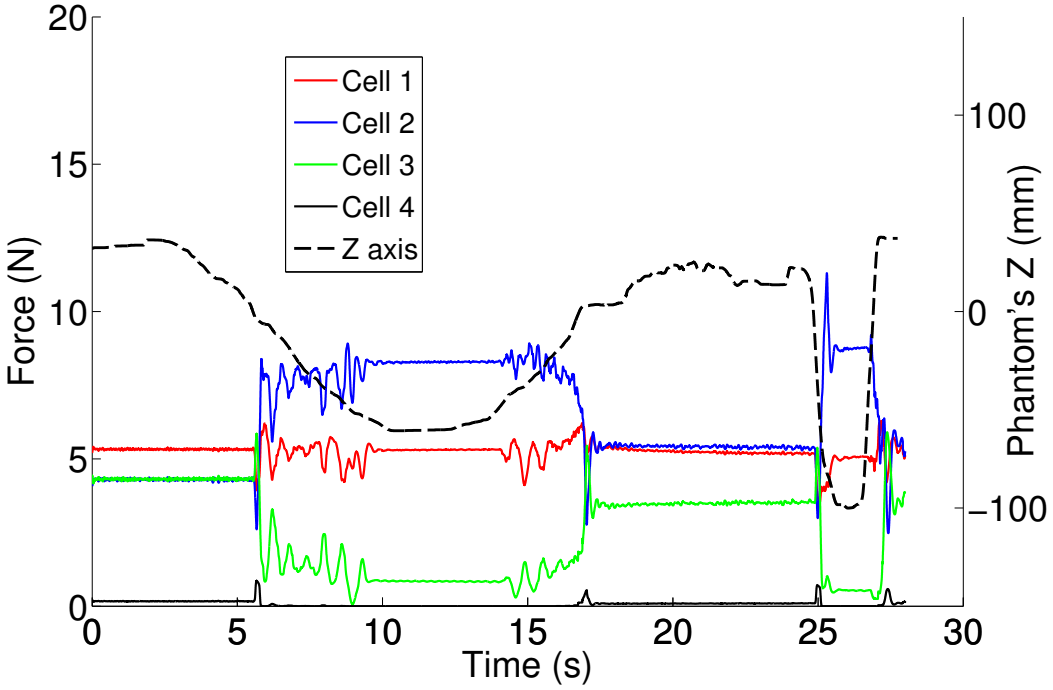


Figure 4.7: Forces normal to the left foot of the robot during the vertical movement demonstration with the Z axis movement of the haptic device.

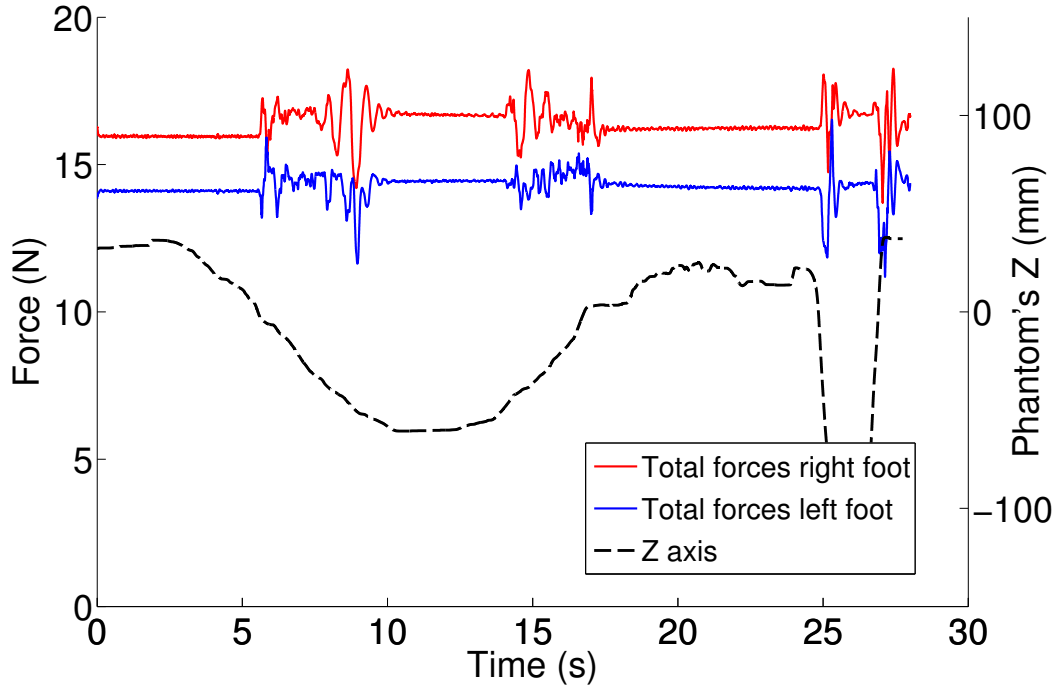


Figure 4.8: Total of the forces in each foot of the robot during the vertical movement demonstration with the Z axis movement of the haptic device.

The calculated COP for the whole experiment is shown in Figure 4.9. The path of the COP shows a region of stability. For a better visualization, a shorter time range of the COP is represented in Figure 4.10; the time range corresponds to the fast lowering movement of this experiment. It shows clearly 2 zones where the bigger one corresponds to the home position part of the movement and the smaller one happens when the robot is in the lowest crouching position.

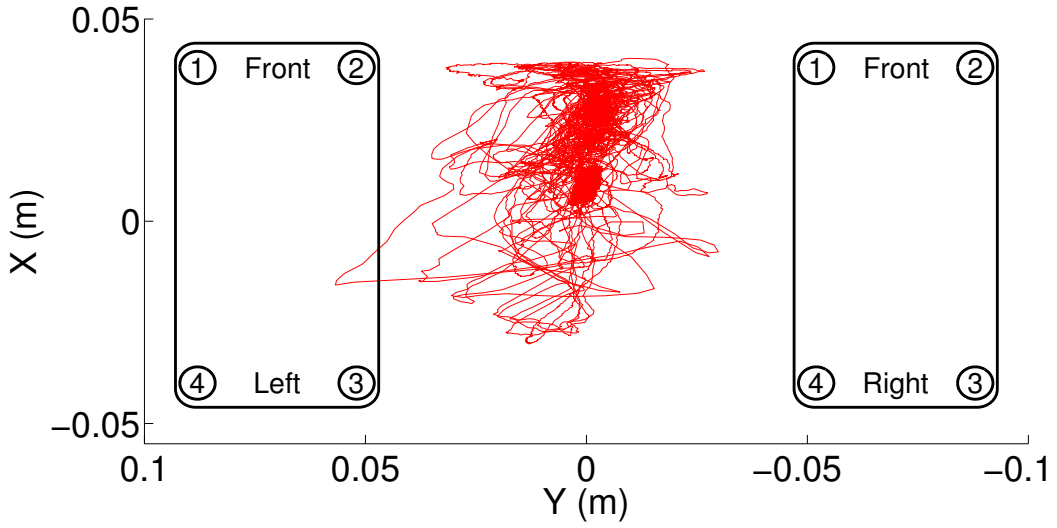


Figure 4.9: Complete motion of the COP during the vertical movement demonstration.

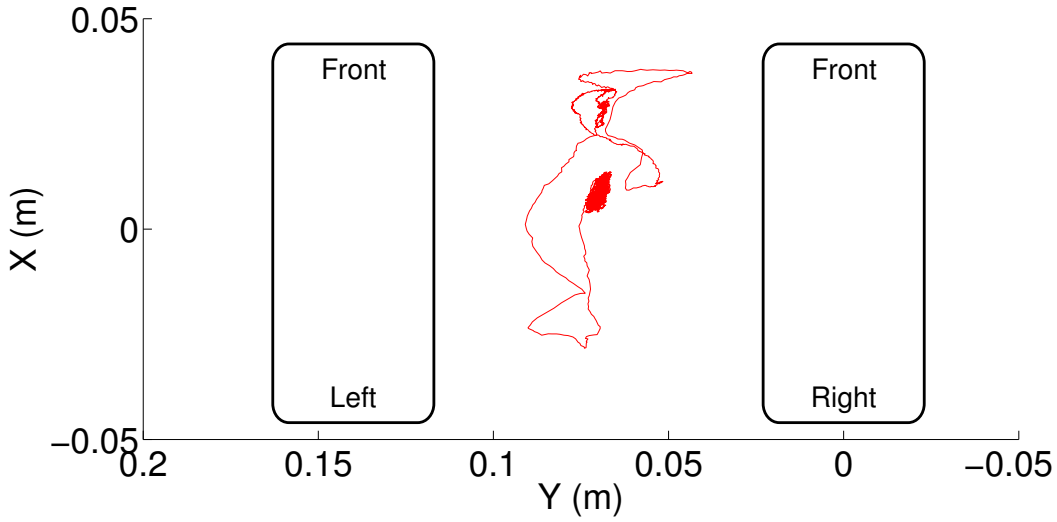


Figure 4.10: Part of the motion of the COP during the fast downward movement.

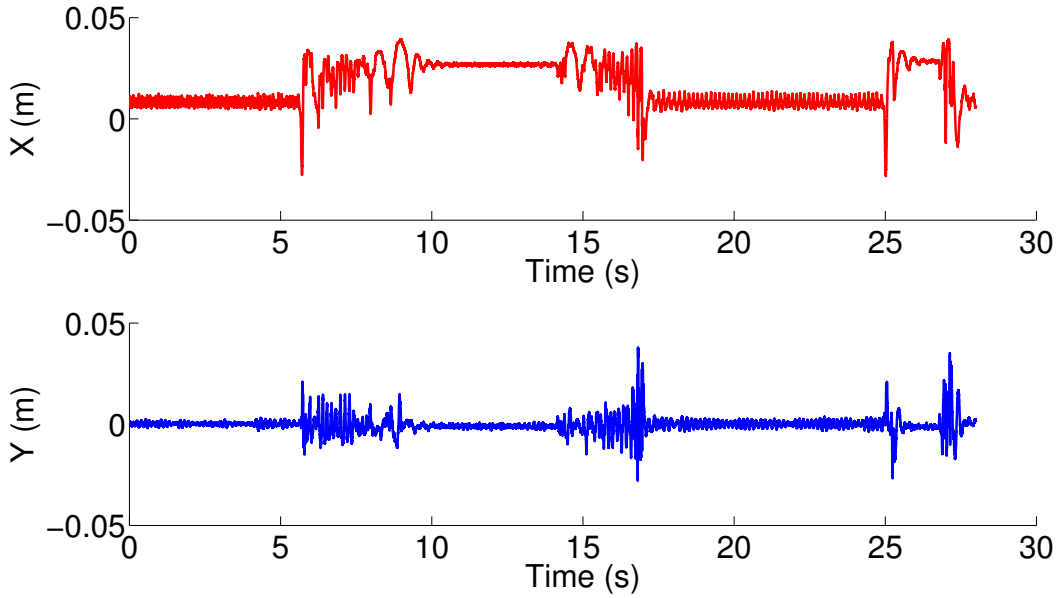


Figure 4.11: Components of the COP robot during the vertical movement demonstration.

Figure 4.11 represents the X and Y components of the COP during the full time range of the experience. The Y does not have a large variation during the experiment because most of the mass moves in the X axis direction. In X component, 2 levels of coordinates are visible; these correspond to the 2 zones of stability in the 2 static stages of movement, home position and full crouching.

4.2 Hip's lateral movement

For this demonstration, the hip moves sideways with the ankle's Inversion/Eversion and the hip's Abduction/Adduction joints performing the relevant movements. These are done in the YOZ plane where the vertical movements are done only when needed for a wider range of lateral movement. The COP is expected to move considerably so the force feedback is active. Like explained in Section 3.2, this forces are the direct result of the difference of the resting position's COP and the COP of the current position of the robot.

4.2.1 Inverse kinematics

The kinematics used in this demonstration are based a two links planar robot like the one shown in Figure 4.12. Like in the previous experiment, the inverse kinematics is done directly from the trigonometrical relations present in this kind of robots. In this case the PHUA robot's hip corresponds to the point A in Figure 4.12 and it's ankle to the axis's origin.

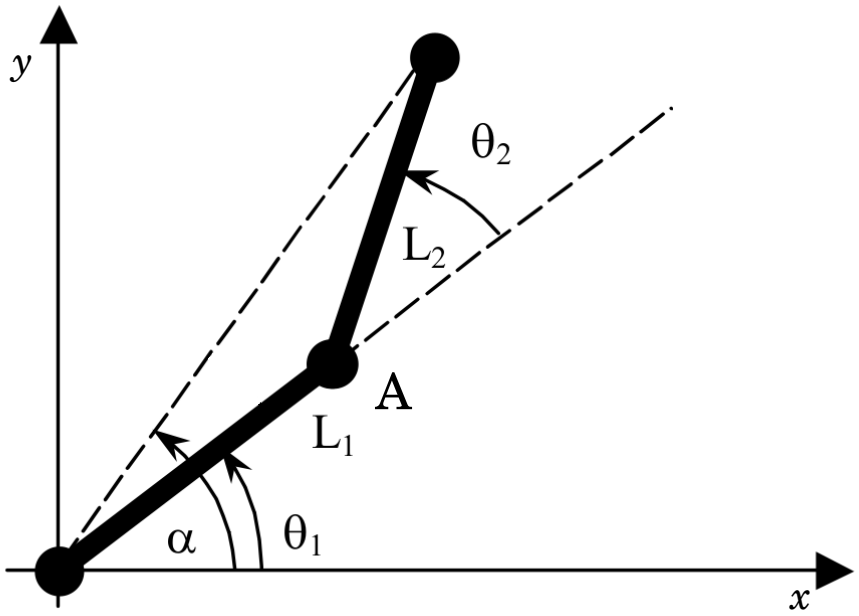


Figure 4.12: Two links planar robot [Santos, 2003].

Since the movement is now done sideways, the parameters for the inverse kinematics are the value of X component of the position of the point A and the direction of the hip. The position of A is given by the Equations 4.6 and 4.7.

$$A_x = L_1 \cos\theta_1 \tag{4.6}$$

$$A_y = L_1 \sin\theta_1 \tag{4.7}$$

By solving the A_x equation to θ_1 is possible to get the first part for the inverse kinematics as seen in Equation 4.8. The θ_2 angle can be obtained by guaranteeing the verticality of the hip, α, as in Equation 4.9.

$$\theta_1 = \arcsin \frac{A_y}{L_1} \tag{4.8}$$

$$\theta_2 = \alpha - \theta_1 \tag{4.9}$$

Due to mechanical restrictions the ankle's Inversion maximum range is 30°; to keep the servomotors from overstraining the applied kinematics take this limitation into account restricting the PHUA robot's movement

4.2.2 Haptic device and robot's motion

In this demonstration the monitored axis of the haptic device is the Y where its zero corresponds to the home position of the PHUA robot. Figure 4.13 illustrates the movement performed by the user during the experiment.

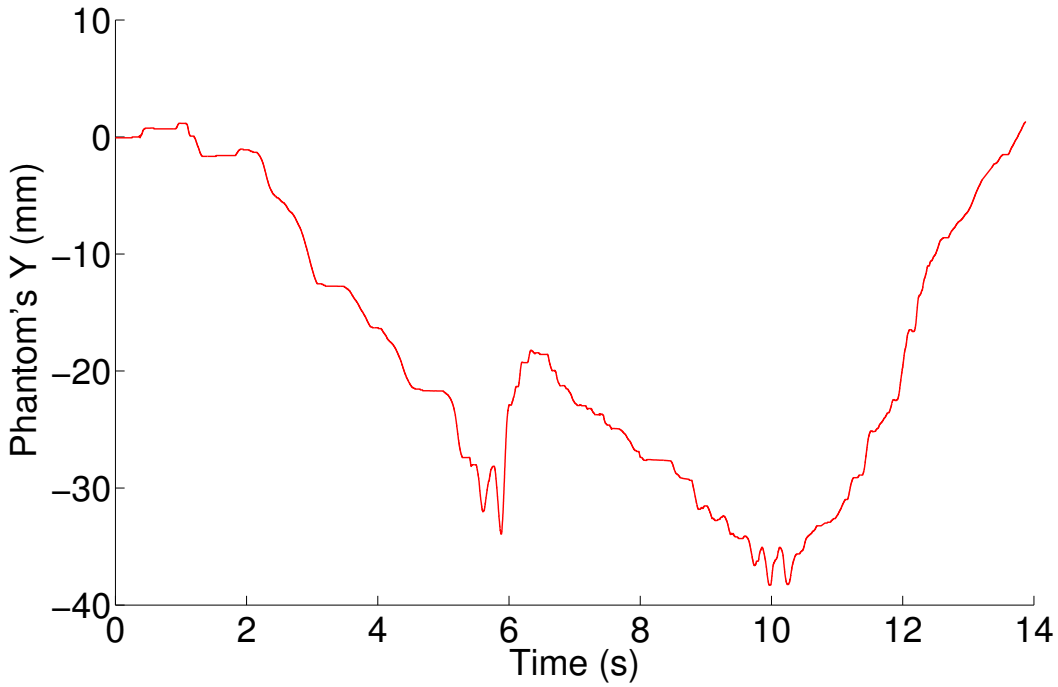


Figure 4.13: Motion of haptic device's Y axis during the lateral movement demonstration.

The movement started with a motion to the left side by the haptic device where the robot followed as seen by the joint values of the right ankle and hip in Figure 4.14. When the user felt a substantial force feedback given the haptic device, a movement to the center position was done to return the robot to its home position; but an unexpected event took place as the ankle furthest from the hip, the left one, skipped some of its belt transmission's teeth. This event causes the *humanoid_ control node* to work as the

servomotor is in the correct position when it is not the case; the joint and the servomotor do not have a correspondent position. Although all the joints are correctly dimensioned to perform the required torques some of them, like the ankle, have a low contact ratio in the transmission; this is caused by very few teeth of the belted transmission are in contact with the pinion. The event happens around the 51st second, just before the experience is terminated.

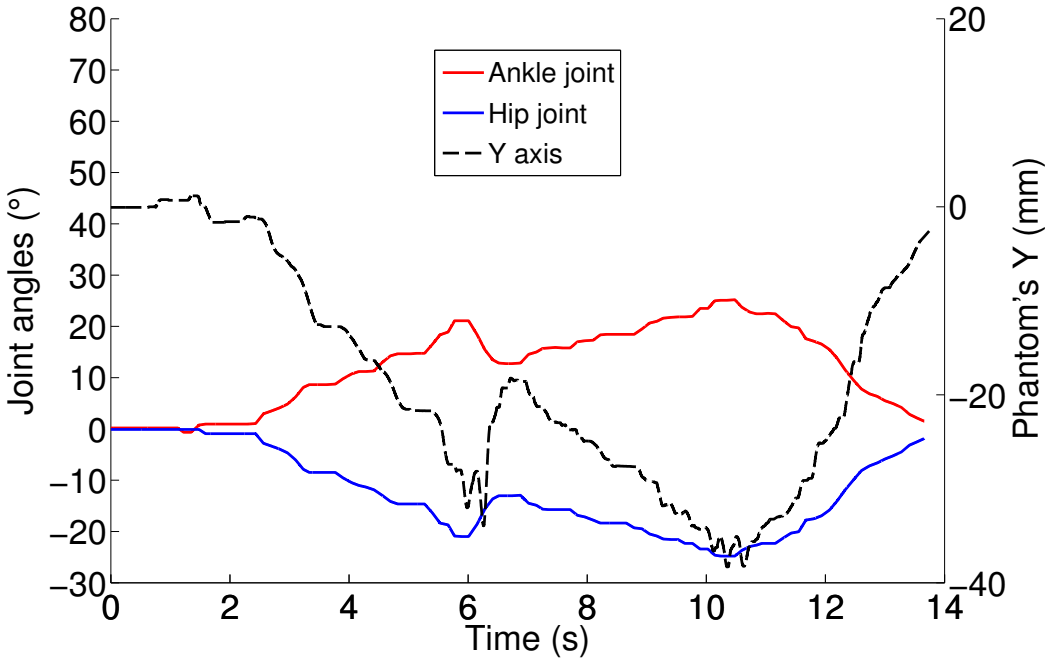


Figure 4.14: Motion of the actuated joints of the PHUA robot during the movement demonstration with the Y axis movement of the haptic device.

The stepping issue also was noticed in this demonstration; now, with all of the developed modules, the frequencies of the ROS messages are checked for discrepancies. Without recording the experiment with *rosbag*, the frequencies are similar to the expected ones except the one holding the state of the humanoid robot which is being published at around 5 Hz. A control loop this slow can be related to the step movement problems in both the first experiment as well this one. During the recording of this experiment's *bag* a loss of ROS messages was noticed; this may be due to excessive CPU usage or a *bug* in the *rosbag* utility.

4.2.3 Forces, COP and haptic generation

Figures 4.15 and 4.16 show the values of the forces in each load cell with the movement of the haptic device also represented.

Similar to the previous experiment, only three load cells at a time register variations but in this demonstration is visible an exchange of the cell that is reduced to the pre-compression from the bolts between the cells 2 and 4. What happens in this demonstration is due to the hip's side movement: the COP moves close enough to affect the cell that at the beginning had no extra strain on it. When this happens, another cell

losses the pressure from the robot's weight reducing the force down to the pre-compression value.

Figure 4.17 represents the total values of the force in each foot. In it is visible a change in weight from the left foot to the right one as the COP moves closer to it.

Figure 4.18 shows the path of the COP during the demonstration where it is clearly visible the dislocation of it to the right as expected. In Figure 4.19 is possible to visualize the X and Y components of the COP during the experience. The X component has very little variation during the demonstration as opposed to the Y component that has a wide range of variations.

It was expected for the COP to have a wide range of movement during this demonstration so the force feedback on the haptic device is activated. Using the linear relation described in Section 3.2 the user feels a force in the direction of the unbalance of the robot. This method requires the user to fully control the haptic device with fine and precise movements so the maximum force of the device, 3.3 N, is not used instead it is used a maximum force of 2.5 N. Figures 4.20 and 4.21 show the force feedback on the X and Y axis of the haptic device in accordance with the X and Y components of the COP; in these, the space between the black lines represents the dead zone. In the first, where the X components are represented, the COP has very small peaks above the dead zone line, thus creating no sensible force for the user. As expected, the generated force is bigger on the Y component since the COP clearly passes the dead zone lines. In Figure 4.21 is possible to observe that the generated force follows the oscillations of the COP.

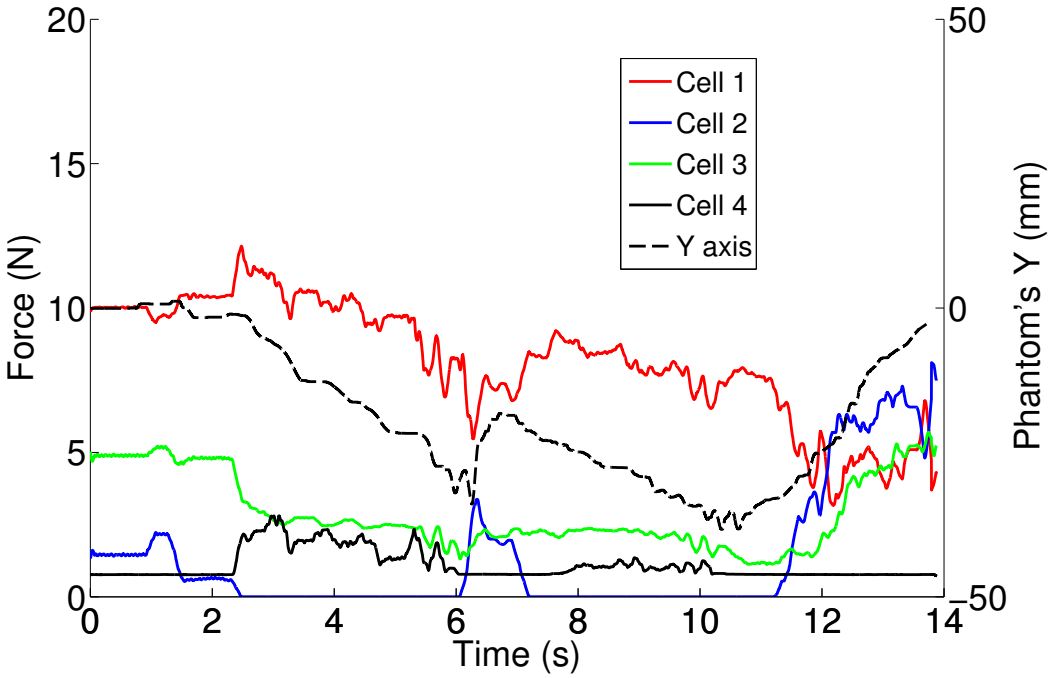


Figure 4.15: Forces normal to the right foot of the robot during the lateral movement demonstration with the Y axis movement of the haptic device.

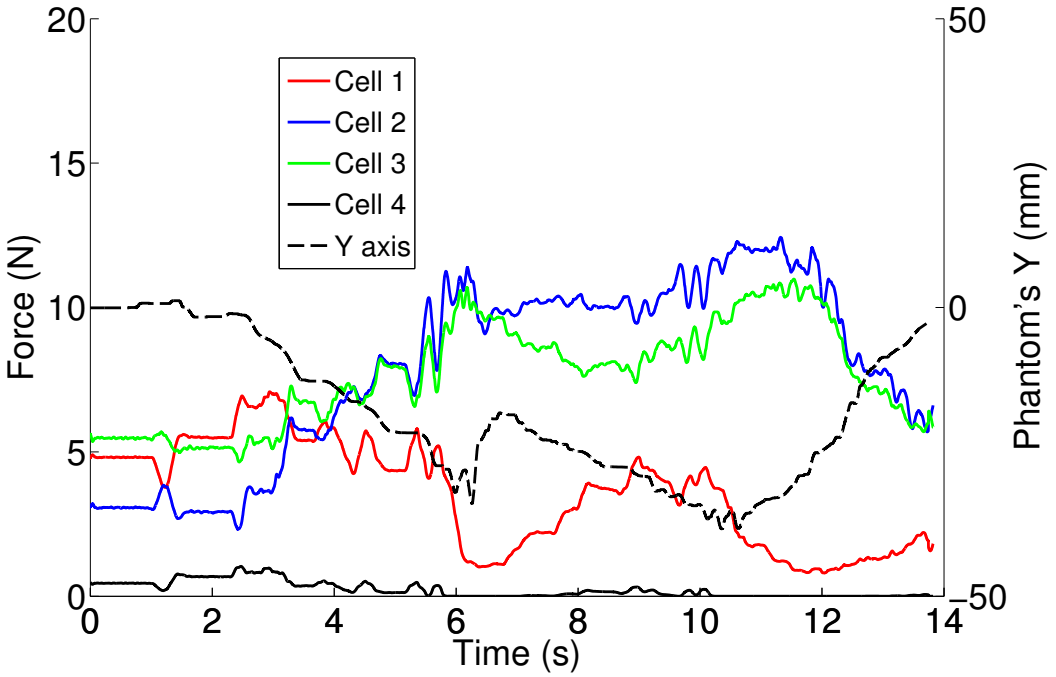


Figure 4.16: Forces normal to the left foot of the robot during the lateral movement demonstration with the Y axis movement of the haptic device.

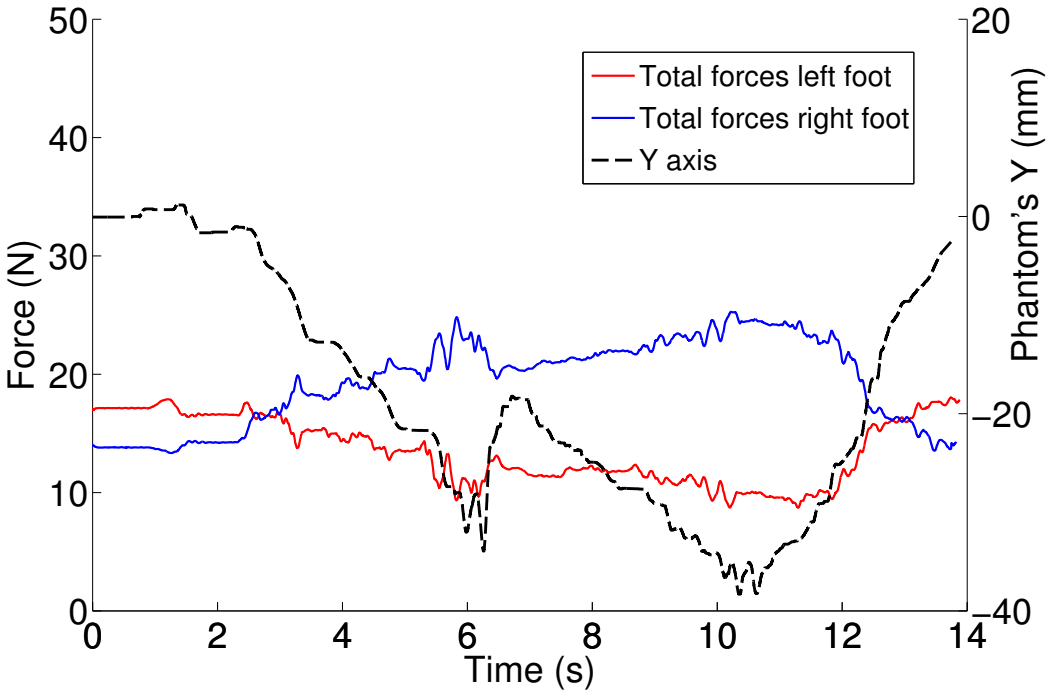


Figure 4.17: Total of the forces in each foot of the robot during the lateral movement demonstration with the Y axis movement of the haptic device.

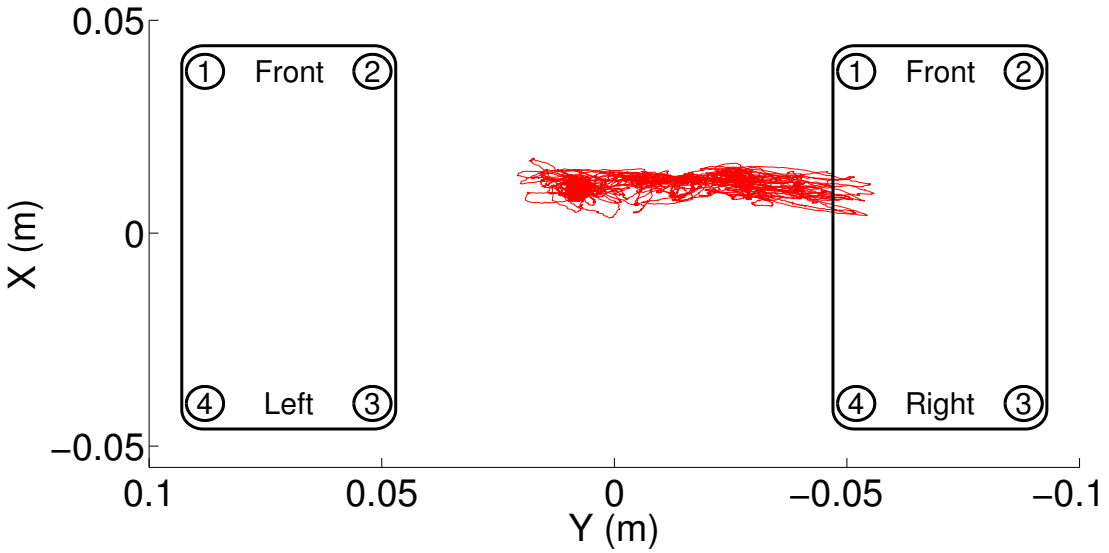


Figure 4.18: Complete motion of the COP during the demonstration.

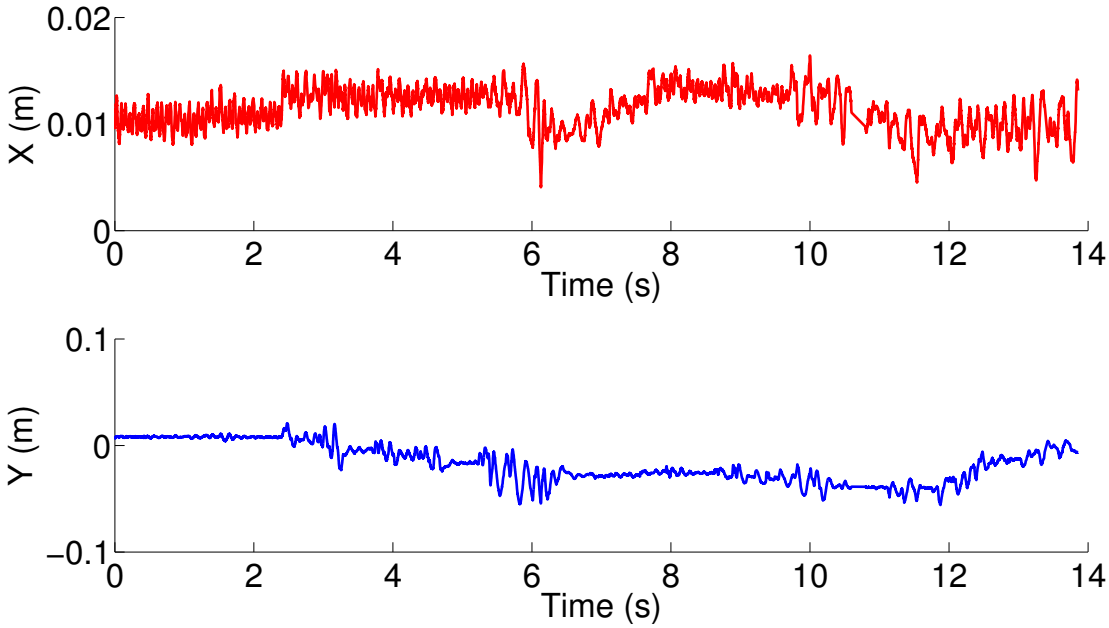


Figure 4.19: Components of the COP robot during the lateral movement demonstration.

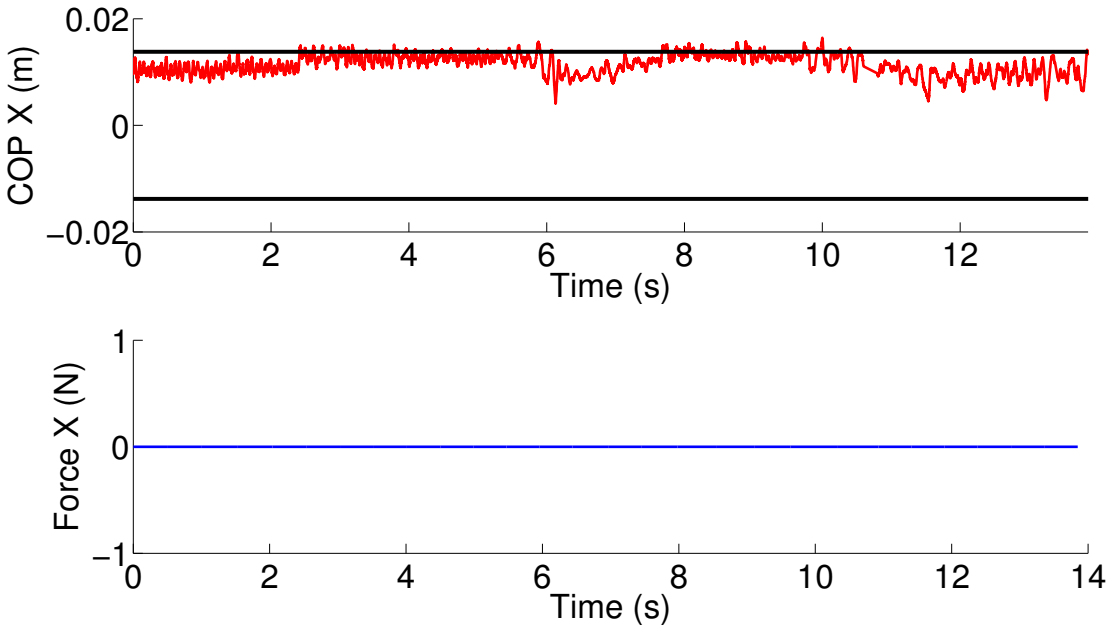


Figure 4.20: X's Components of the COP and force feedback with the dead zone lines.

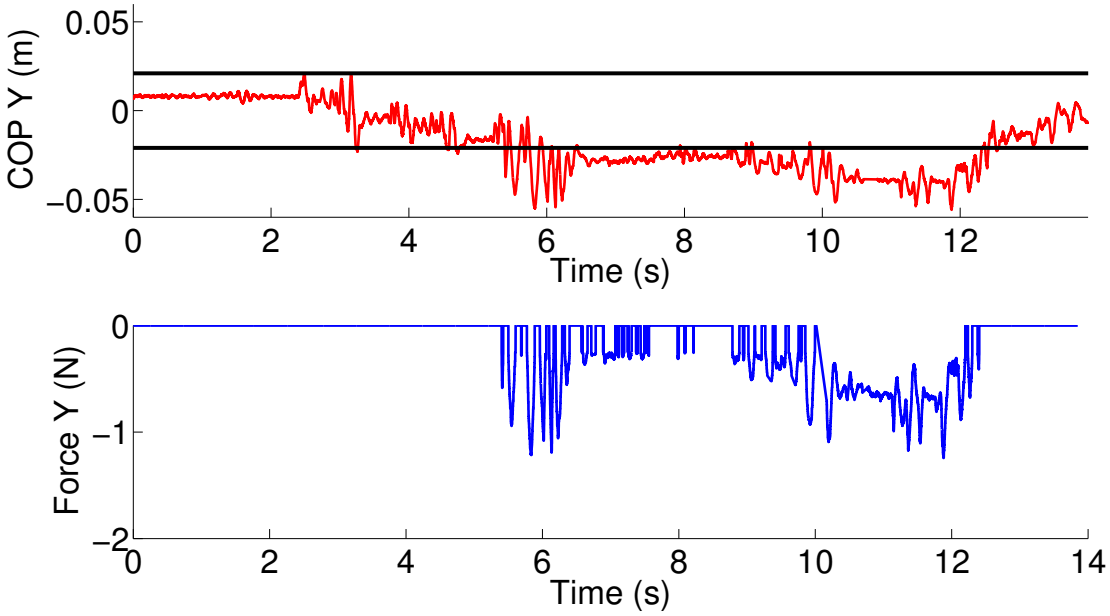


Figure 4.21: Y's Components of the COP and force feedback with the dead zone lines.

4.3 Hip's sagittal movement

In this demonstration, the robot's hip moves back and forward with most movements in the X axis. For back and forward movements the hip needs to lower its height so the knee joint can have a greater influence in this demonstration. This movement is particularly important for the walking gait as the forward balancing of the body greatly contributes to the activity. Similarly to the last experience, the force feedback is active so the user can feel the unbalance of the robot. The data that this experience produces is invaluable to the future work for robot learning from demonstration.

4.3.1 Inverse kinematics

The kinematics used in this demonstration is based on the generic planar robot shown in Figure 4.3 similar to the first demonstration. The main difference is the necessity to have the hip at a lower height; for this, the ankle joints are placed in their maximum Flexion range, 20°, and the knee and hip's joints make the longitudinal movement of the hip while keeping the hip vertical.

Using the kinematics from the other demonstrations and the constrains described before, the Equation 4.10 gives the value for the θ_2 where it depends on θ_1 that now is not a variable but a constant. Like before, the last joint depends on the required orientation of the hip as in Equation 4.11. Similar to the other experiments, the kinematics is restrained by the first joint to reach one of its maximum ranges.

$$\theta_2 = \arccos\left(\frac{x}{L_2} - \frac{L_1}{L_2}\cos\theta_1\right) - \theta_1 \tag{4.10}$$

$$\theta_3 = \phi - \theta_2 - \theta_1 \tag{4.11}$$

4.3.2 Haptic device and robot's motion

The monitored axis of the haptic device in the demonstration is the X; its zero corresponds to the full crouching position. Figure 4.22 shows the movement performed by the haptic device's tip during the experiment.

The experience started with the hip near the home position moving then to the front. The user, feeling the force feedback, stopped the front motion before the robot lost its balance. Passing through the home position, the user tele-operated the robot to take its hip back as far as possible. Once reached the joint's limits, the robot preforms a repetition of the movement. The motion of the hip and knee's joints are visible in Figure 4.23 as is the X axis's values of the device.

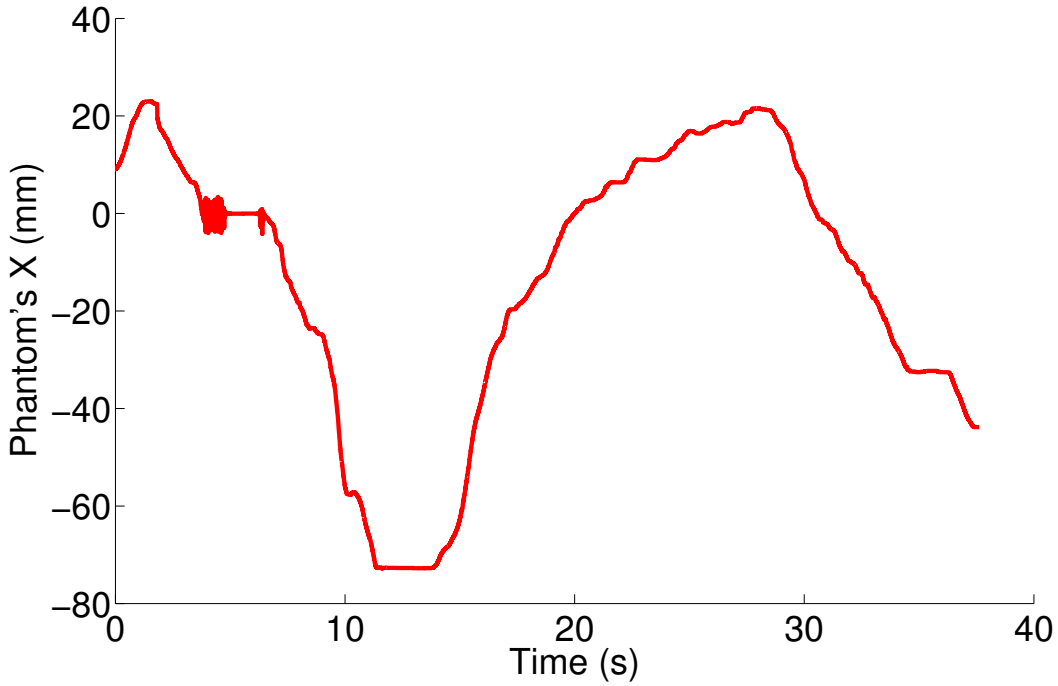


Figure 4.22: Motion of haptic device's X axis during the sagittal movement demonstration.

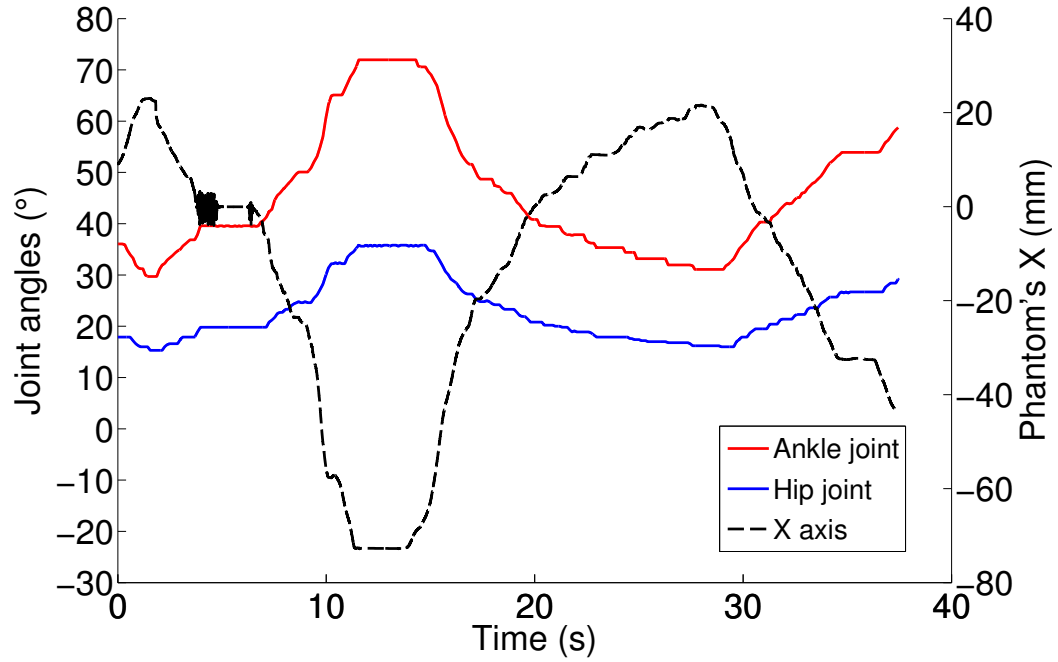


Figure 4.23: Motion of the actuated joints of the PHUA robot during the movement demonstration with the Z axis movement of the haptic device.

4.3.3 Forces, COP and haptic generation

In Figures 4.24 and 4.25 it is possible to visualize the forces sensed by the load cells during the demonstration. In some parts of this experiment only 2 cells of a foot are feeling the floor under the foot. Around the 20th second only the front cells of the left foot are feeling the normal forces from the robot's weight; the duration of this state is approximately 10 seconds and it corresponds to the movement of the hip beyond the home position for this experiment.

Figure 4.26 illustrates the total values of the force in each foot; in this graph a constant shift between the two feet is seen. This may be caused if the feet are not in the same plane or if there is a joint with the belted transmission not tight enough.

Figure 4.27 represents the COP during the experience where it shows a back and forward movement with a slight inclination to the right; the analyses of this and the graphs in Figures 4.24 and 4.25 indicate that the right foot is inclined in relation to the floor. Figure 4.28 illustrates the X and Y components of the COP during the experience.

The generated forces in this experience are shown in Figures 4.29 and 4.30 with the dead zone lines in black. In the X axis, the bigger forces are generated when the user takes the hip to the front causing the COP to move to the front zone. The graph corresponding to the Y axis shows that a great force was generated due to the deviation of the Y component of the COP; this resulted from the inclined foot.

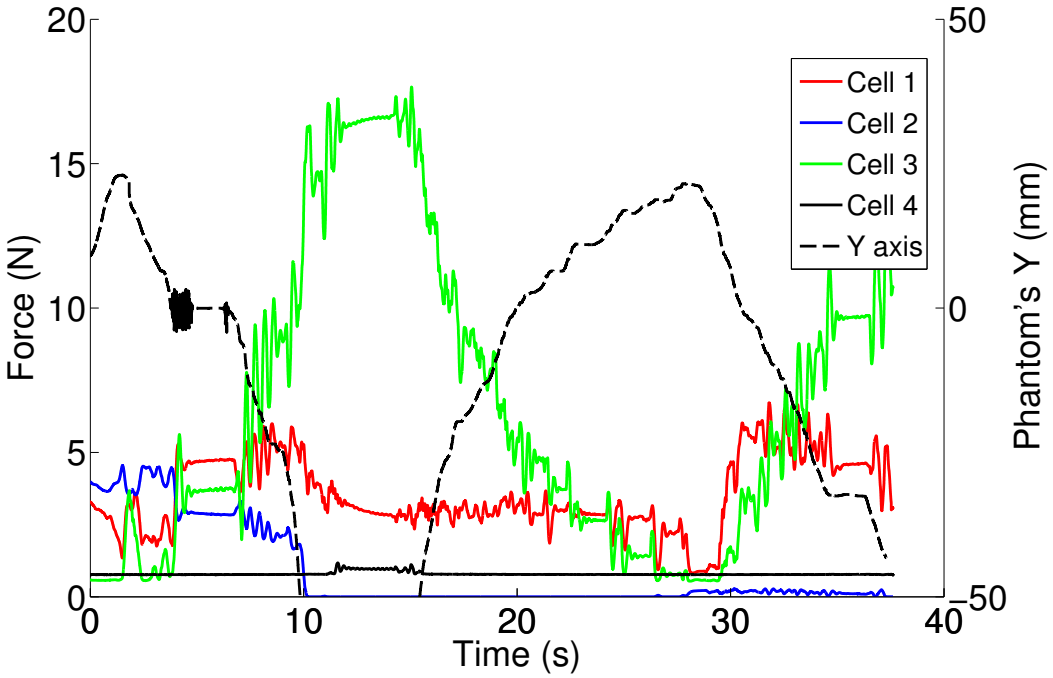


Figure 4.24: Forces normal to the right foot of the robot during the sagittal movement demonstration with the Z axis movement of the haptic device.

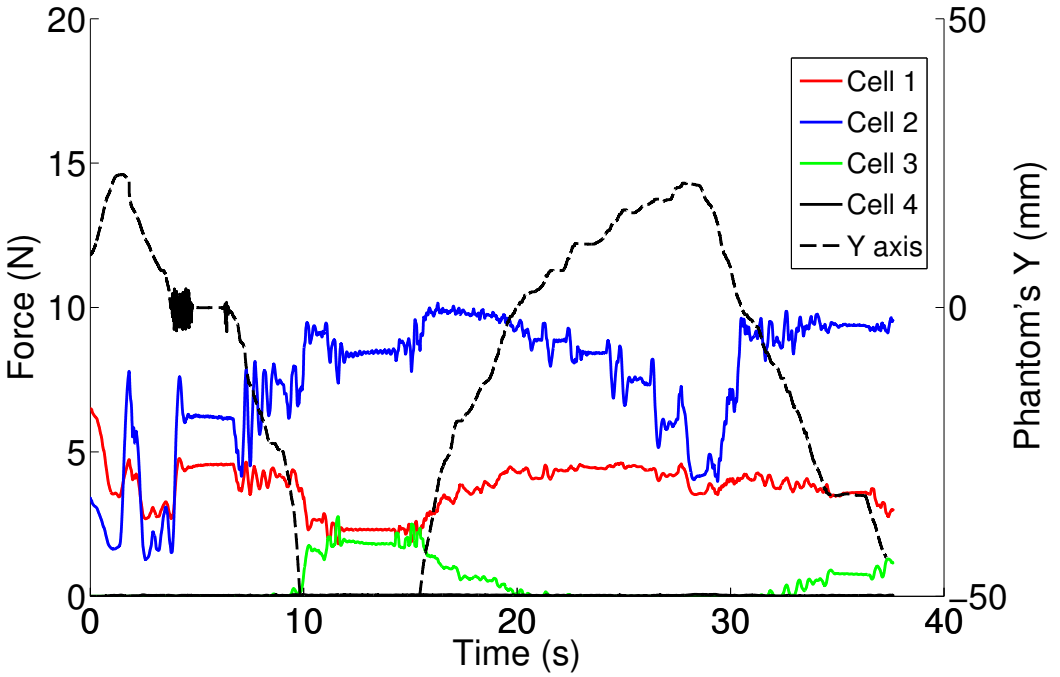


Figure 4.25: Forces normal to the left foot of the robot during the sagittal movement demonstration with the Z axis movement of the haptic device.

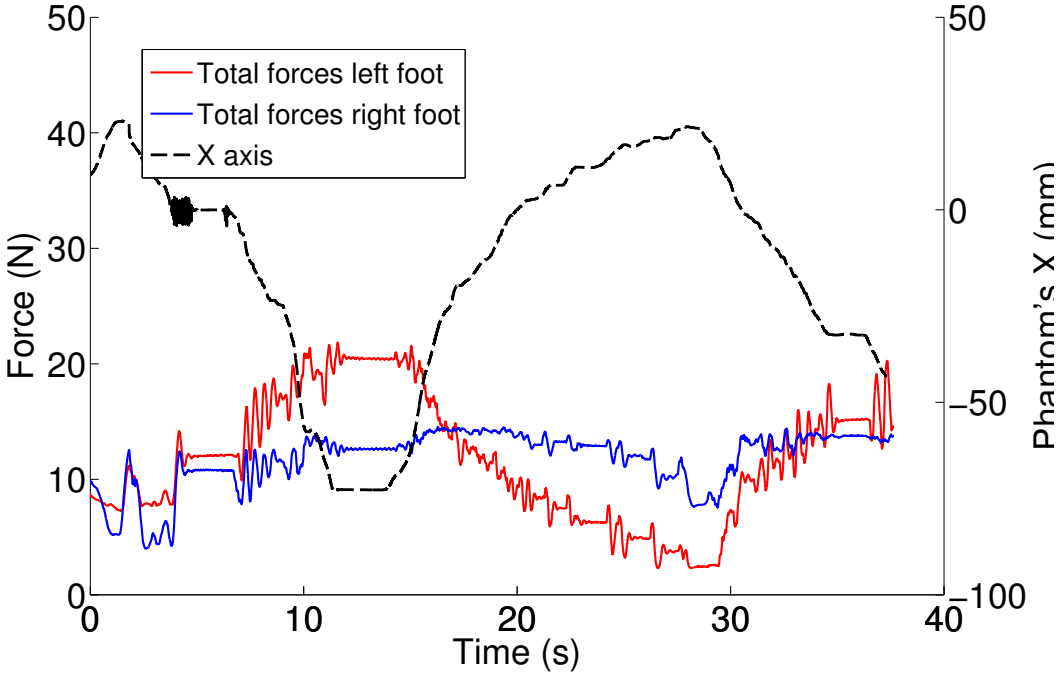


Figure 4.26: Total of the forces in each foot of the robot during the sagittal movement demonstration with the Z axis movement of the haptic device.

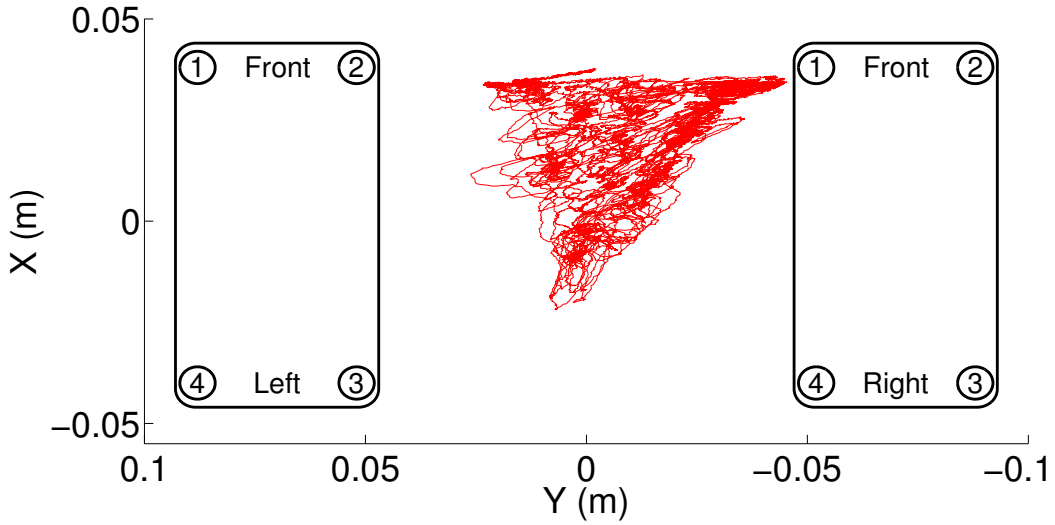


Figure 4.27: Complete motion of the COP during the demonstration.

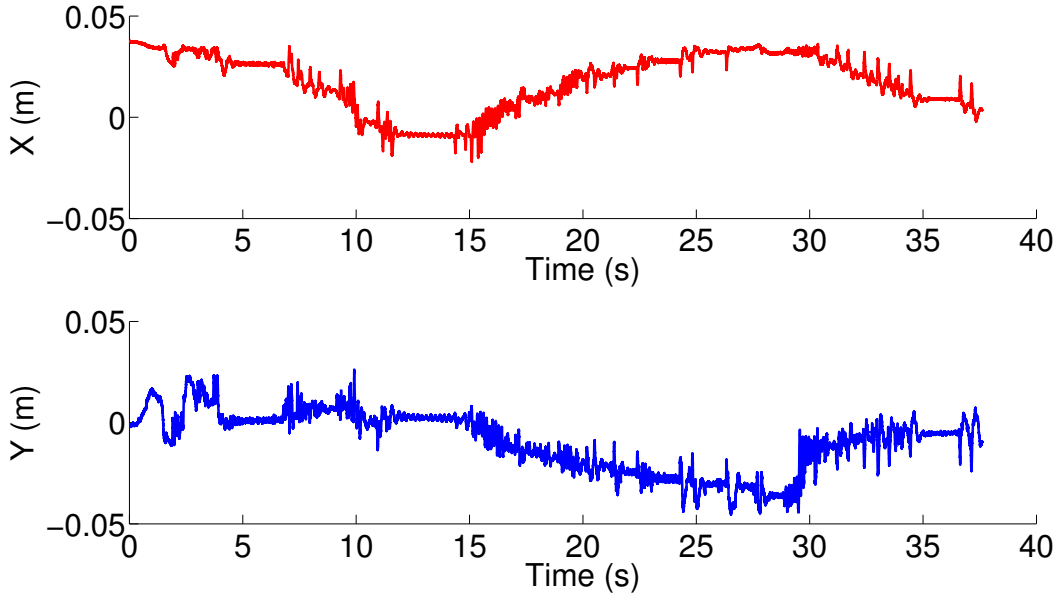


Figure 4.28: Components of the COP robot during the sagittal movement demonstration.

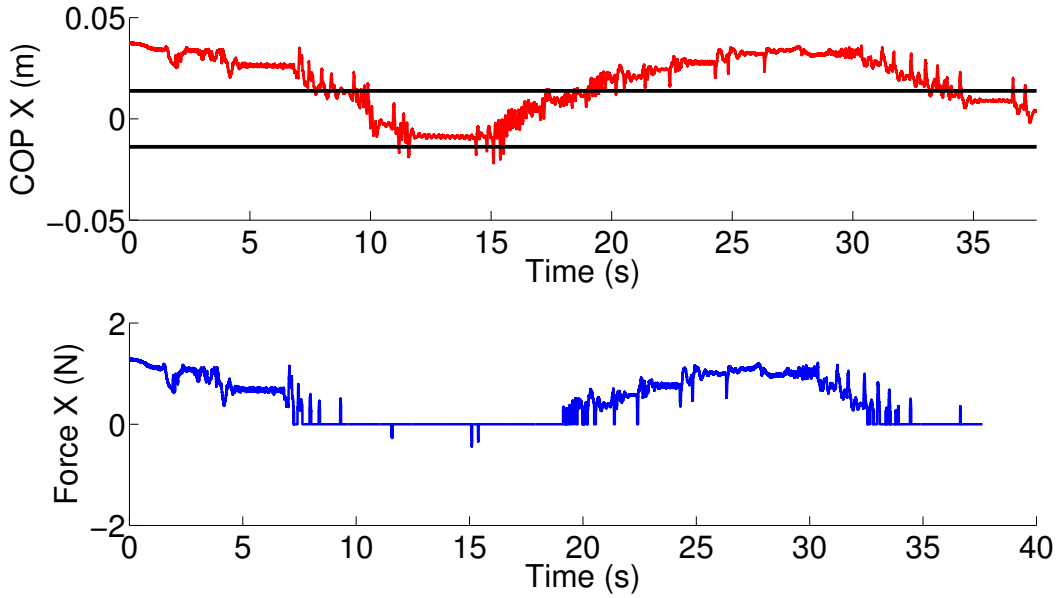


Figure 4.29: X's Components of the COP and force feedback with the dead zone lines.

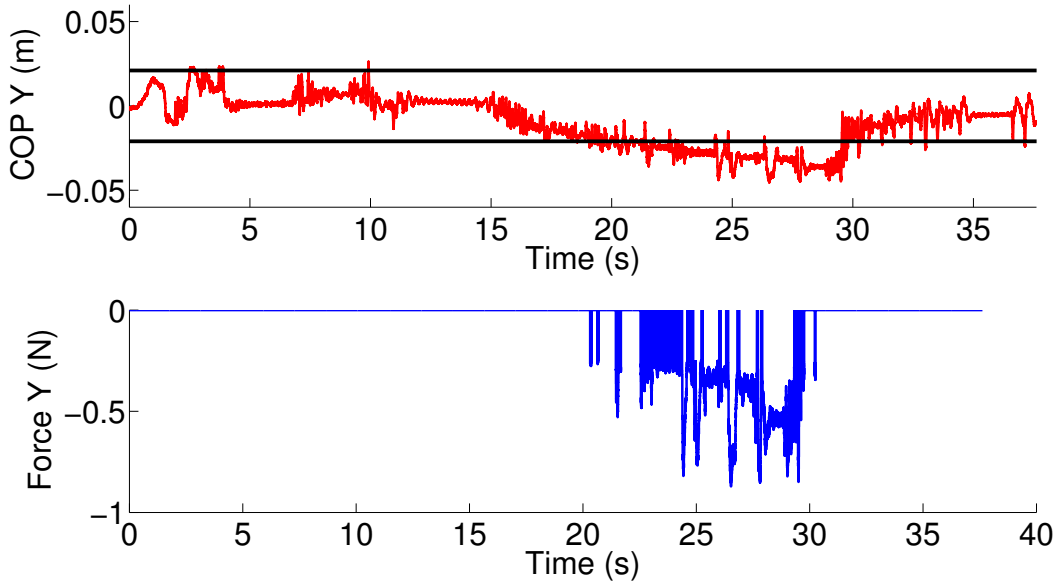


Figure 4.30: Y's Components of the COP and force feedback with the dead zone lines.

Chapter 5

Conclusions and Future Work

The conclusions of this work are detailed the next Section, evaluating the success of the proposed objectives. The Future Work Section gives proposals for the continuation of this work and future of this project.

5.1 Conclusions

The first objective of this work was to implement the ROS platform in the PHUA. This has been done successfully; the modular capabilities of this platform are very useful for this project, where treating each problem separately has shown immense potential. The message communication system of ROS works well with the problems presented here. It allowed for a easy data transmission mechanism and an easy visualization. The usage of the *rosbag* utility needs to be rethought; the loss of messages is problematic for future work of robot teaching from demonstration algorithms.

The data acquisition unit for the load cells was successfully developed and tested. The retrieval of the load cells signal is solid and stable and it gives the project a reliable source for environment sensing. There is still space for improvement now that this design was tested and proven to work, more is explained in Section 5.2.4.

Thanks to the modular software development and message communication, the link between the load cells information and the haptic force feedback was successful. Although the concept is proven, it still needs improvement in the force generation functions for a better sense of feel by the operator. A multiphase polynomial formulation may be worth to explore [Cruz, 2012]. The formulation presented in this work (and future ones) should be tested with multiple users after which they give testimony in order to pinpoint the optimal solution.

The control of the PHUA robot's legs by the PHANTOM haptic device was successful; the control in position is capable of correcting the position of the robot. A speed based control is the next step for this project as is referred in Subsection 5.2.3.

Even though the used kinematics worked well in the demonstrations, there is a lot space for improvement in this part of the project. The continuation of the work here developed needs a full mathematical formulation for the control of both legs simultaneously with torso integration, direct, inverse and differential kinematics. This will allow for three dimensional control in both position and velocity.

The demonstrations were successful in different levels. The first one proved the

concept of tele-operation of the robot by the haptic device where servomotors responded with little delay to the implemented kinematics. The load cells showed a quick and stable response to the movements of the robot giving the calculated COP a great accuracy. In the second experiment, even with the transmission slip, it was proven that the user can feel enough force on the haptic device to keep the robot in a stable position. The fact that only three load cells sense more than the pre-compression did not interfere with the behaviour of the COP; even then, for all the cells to be in contact with the floor and have a complete knowledge of the COP a proposal in given in Subsection 5.2.4. The force feedback's dead zone proved to be an asset to the tele-operation system as it prevented instability due to positive feedback and straining on the user's manipulation of the device. The last demonstration aimed to gather further knowledge for the preparation of the walking gait. Although more experiments with sophisticated kinematics are needed, it proves that the robot can perform the forward balancing motion that is needed for the gait.

5.2 Future work

5.2.1 Software

The software developed in this work can now be considered the base for further development in this project. With the ROS platform, the project can evolve each module separately with little or no repercussion to the rest of it. More software modules can be added depending on the needs of the project.

The path for growth in this project is the refinement of the haptic force generation and the robot control. The force generation can be further developed in terms of mathematical formulation to increase the feel of the robot's unbalance by the user. More on this subject in Section 5.2.2.

The robot control needs more refining. Due to the small force workspace, a velocity control instead of a position control may bring improvements. This might be difficult though because of the servomotors communication. These particular servomotors can only handle position or speed commands making this type of control complex.

5.2.2 Haptics

The haptic interface has a lot to be developed in this project. It needs more improvement in terms of force generation and possibly more sources of information for a better feel for the user.

Different approaches should be taken to the force generation. In this work only two dimensional forces were created, a third direction may improve the user experience by creating a more intuitive interface. A possible use for the third force component is the effect the inertial forces at the end of the movements; the data for this effect can be accounted for by the total forces of each foot. In the generation itself, more complex mathematical formulations should be attempted, polynomial, exponential or sets of both can be experimented on different users for a more practical interface.

There is also space for more sensory information. The addition of other types of sensors, like inertial or tilt, can bring additional information and complexity to the force feedback generation. Each type of sensor needs a module to read/communicate with the

sensor and then a new *node* in the *haptic_force* module can be written to translate the new type of information into force feedback. Only after these steps can all the sensory data be merged into a single force generation routine.

5.2.3 Command

The command of the servomotors present in the PHUA robot needs a refinement. A more comprehensive study of both the haptic device's and the robot's kinematics needs to be done. Another possible solution for the command issue is the use of two haptic devices, one for each leg. Although this can bring a lot of control problems, it represents the possibility to execute more complex movements, like one leg balancing or a human-like walking gait. When implemented, the torso's joints can work in function of the COP's deviation to help with the robot's balance.

To prevent saturation of the serial communication line for the servomotors, the usage of multiple serial lines where one line commands one member of the robot; this may require a rearrangement of the software control with a *node* for each line/member and a central one to control them all. This will bring more the need for computer power for this project, possible solutions for this issue are addressed in the next Subsection.

5.2.4 Hardware

The continuation of this work in terms of hardware passes through the reduction of size of the data acquisition unit for the load cells. A smaller unit capable of being installed in the robot's shin is the ideal solution. The ARDUINO Nano board is similar and smaller to the Fio and can easily be placed in the robot. The signal acquisition board can be adapted with smaller component packages (SOIC or QFP) to fit the Nano board footprint and pin layout.

One of the problems encountered during the experiments was the spiking of the belted transmissions teeth's due to the low contact ratio between the servomotor's pinion and the belt. This requires an intervention on the ankle's Inversion/Eversion and the hip's Abduction/Adduction that has a similar reduction; a stretcher for the belts or a change in the reduction may solve the problem.

A solution for the over usage of the CPU and RAM can pass through the application of the *peer-to-peer* capabilities of ROS; this allows for the ROS *nodes* to be launched in different machines while keeping the ROS messages socket system working between them. This way a distributed system can be used form controlling the PHUA robot increasing the computational power.

References

ATMEL. *ATmega48PA/88PA/168PA/328P Datasheet*. ATMEL, October 2009.

Pedro Cruz. Haptic interface data acquisition system. Master's thesis, University of Aveiro, 2012.

Pedro Cruz, Vítor Santos, and Filipe Silva. Tele-kinesthetic teaching of a humanoid robot with haptic and data acquisition. In *Proceedings of the IROS'2012 Workshop on Learning and Interaction in Haptic Robots*, 2012.

Pedro Ferreira. Desenvolvimento de algoritmos de controlo para locomoção de um robot humanóide. Master's thesis, University of Aveiro, July 2007.

Ricardo Godinho. Desenvolvimento do tronco e braços de uma plataforma humanóide híbrida. Master's thesis, University of Aveiro, 2011.

TEXAS INSTRUMENTS. *INA 129 Datasheet*. TEXAS INSTRUMENTS, October 1997.

INTERFACE. *Model LBS Miniature Compression Load Button Datasheet*. INTERFACE, March 2009.

William Lage. Algoritmos de controlo do movimento para um robô humanóide. Master's thesis, University of Aveiro, 2011.

Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. *ROS: an open-source Robot Operating System*. International Conference on Robotics and Automation (ICRA), 2009.

Miguel Ribeiro. Desenvolvimento dos sistemas sensorial e motor para um robô humanóide. Master's thesis, University of Aveiro, 2010.

Mauro Rodrigues. Unidade de processamento e sistema de visão para um robô humanóide. Master's thesis, University of Aveiro, 2008.

Rémi Sabino. Estrutura híbrida de locomoção para um robô humanóide. Master's thesis, University of Aveiro, 2009.

Vítor Santos. *Sebenta da cadeira de Robótica Industrial*. 2003.

Vítor Santos, R.A.S. Moreira, Miguel Ribeiro, and Filipe Silva. Development of a hybrid humanoid platform and incorporation of the passive actuators. Tianjin, China, December 2010.

Vítor Santos, Rui Moreira, and Filipe Silva. Mechatronic design of a new humanoid robot with hybrid parallel actuation. *International Journal of Advanced Robotic Systems*, Vol. 9, 2012. doi: 10.5772/51535.

SensAble. *OpenHaptics API Reference Manual.* USA: SensAble Technelogies, 2008a.

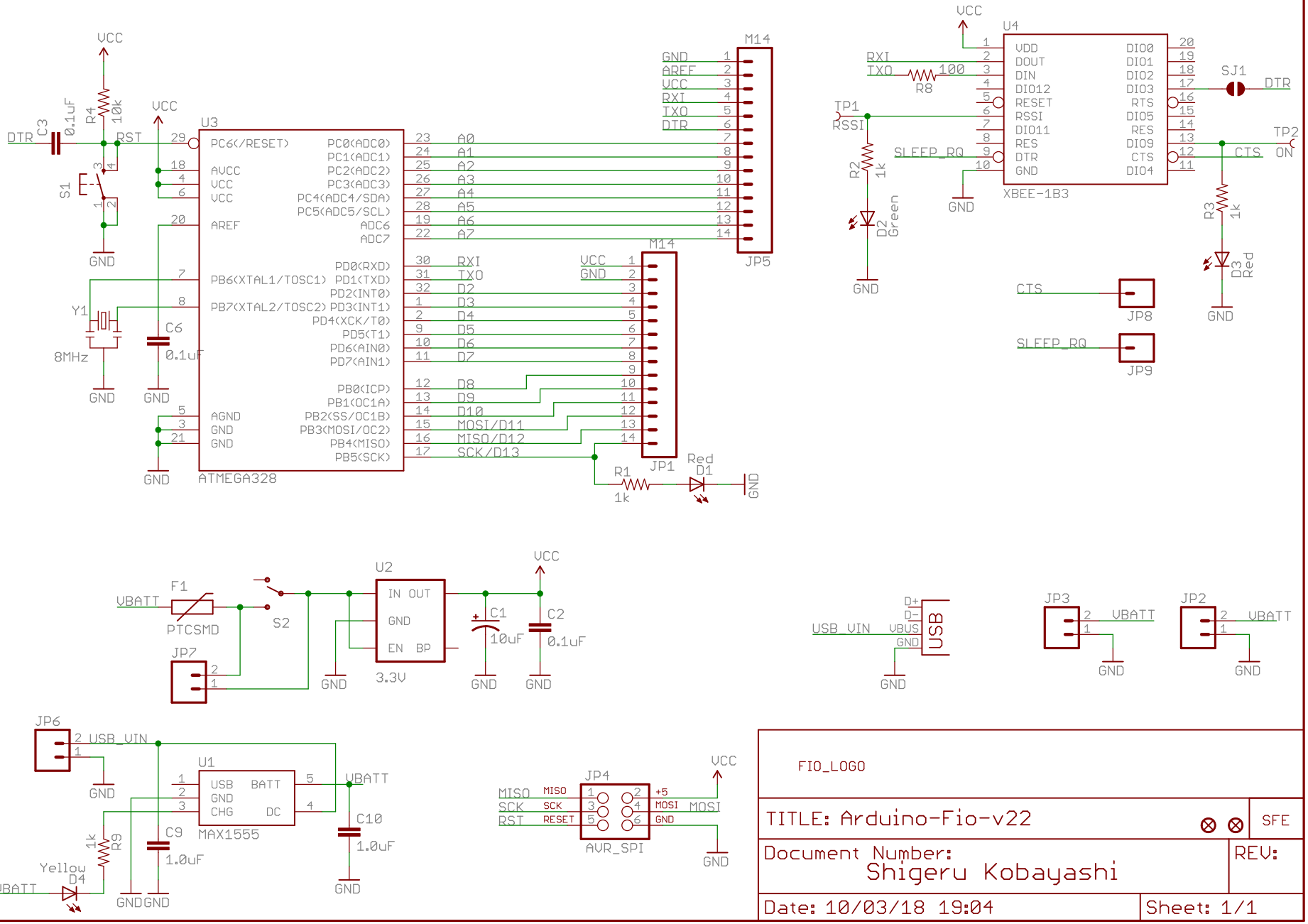
SensAble. *OpenHaptics Toolkit Programmer's Guide.* USA: SensAble Technelogies, 2008b.

SensAble. *PHANTOM Omni User's Guide.* USA: SensAble Technelogies, 2008c.

Milton Silva. Desenvolvimento de algoritmos de controlo para locomoção de um robot humanóide. Thesis, University of Aveiro, July 2006.

Appendix A

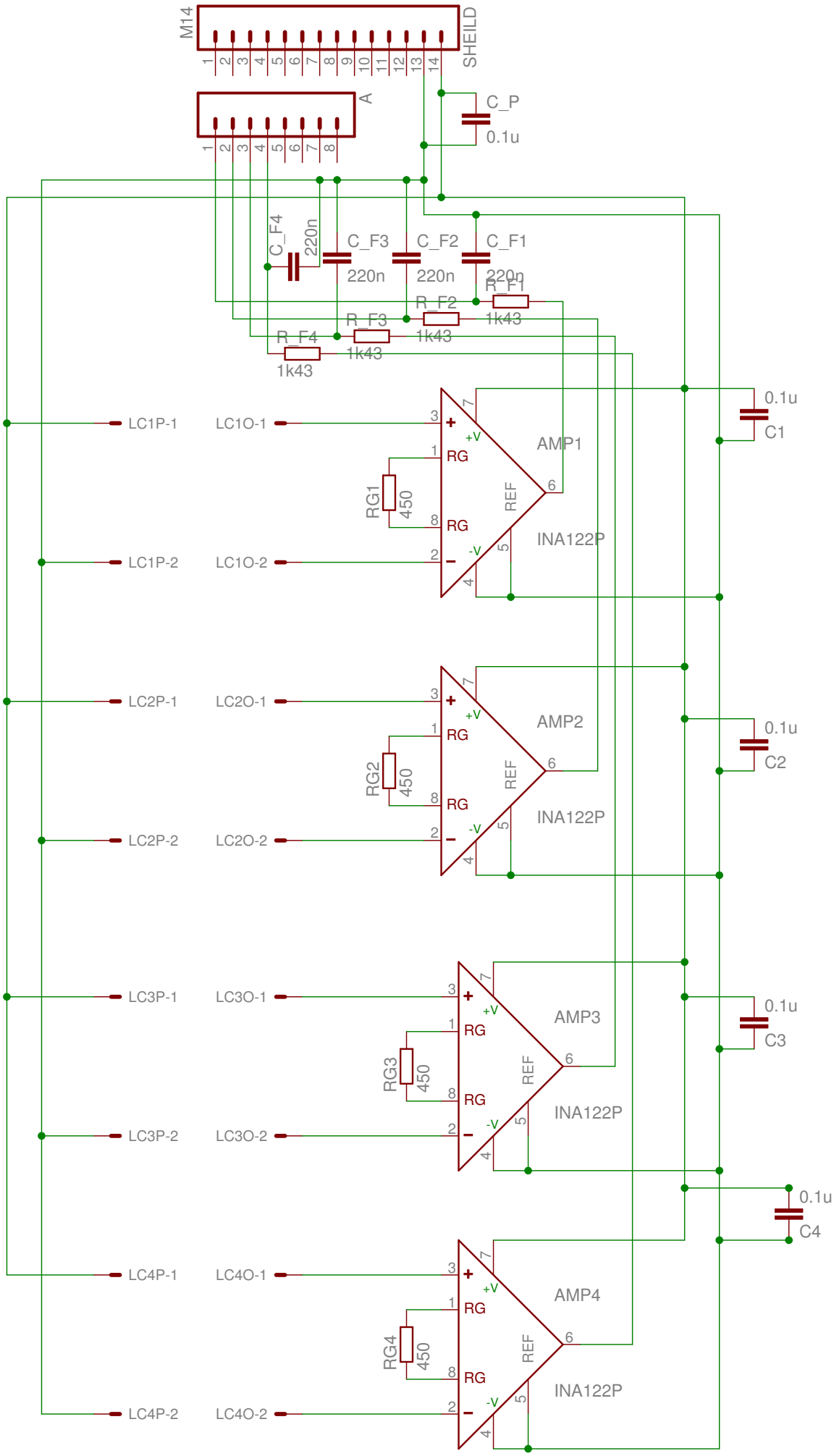
Arduino Fio's Schematic



TITLE: Arduino-Fio-v22	SFE
Document Number: Shigeru Kobayashi	REV:
Date: 10/03/18 19:04	Sheet: 1/1

Appendix B

Circuit for the amplification shield


M14
SHEILD
A
C_P
0.1u
C_F4
220n
C_F3
220n
C_F2
220n
C_F1
220n
R_F1
1k43
R_F2
1k43
R_F3
1k43
R_F4
1k43
LC1P-1
LC1O-1
LC1P-2
LC1O-2
AMP1
RG1
450
INA122P
0.1u
C1
LC2P-1
LC2O-1
LC2P-2
LC2O-2
AMP2
RG2
450
INA122P
0.1u
C2
LC3P-1
LC3O-1
LC3P-2
LC3O-2
AMP3
RG3
450
INA122P
0.1u
C3
LC4P-1
LC4O-1
LC4P-2
LC4O-2
AMP4
RG4
450
INA122P
0.1u
C4

Appendix C

Layers for the etching of the amplification shield

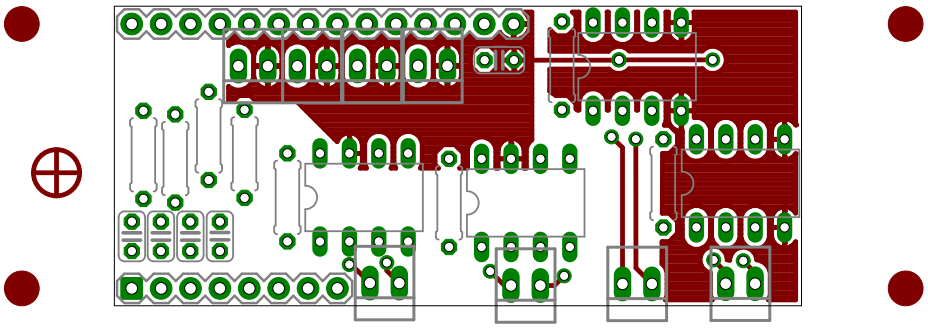


Figure C.1: Top layer of the amplification shield.

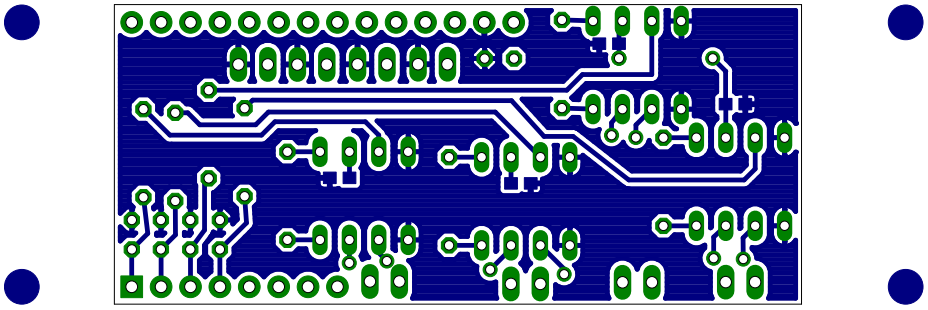


Figure C.2: Bottom layer of the amplification shield.

Appendix D

Launch files used in demonstrations

Launch file *pre_pressure_cells.launch*:

```
<?xml version="1.0"?>
<launch>
<!-- This is the first file to be launched -->

       <!-- This launches a device_mapper node to identify the used ports by the arduino
       <node name="device_mapper" pkg="device_mapper" type="device_mapper">
         <param name="robot_urdf_path" value="$(find pressure_cells)/urdf/humanoid_urdf.
    </node>

    <!-- This launches a node to publish the transformations between each foot and the
    <node name="control_test" pkg="humanoid_control" type="control_test"/>

</launch>
```

Launch file *humanoid_full.launch*:

```
<?xml version="1.0"?>
<launch>

     <!-- Launch humanoid_ph.launch file -->
     <include file="$(find humanoid_control)/launch/humanoid_ph.launch"/>

    <!-- Launch pressure_cells.launch file -->
     <include file="$(find pressure_cells)/launch/pressure_cells.launch"/>

    <!--  Launch haptic_force.launch file -->
     <include file="$(find haptic_force)/launch/haptic_force.launch"/>

</launch>
```

Launch file *humanoid_ph.launch*:

```
<?xml version="1.0"?>
<launch>

    <!-- This launches a Humanoid node-->
    <node name="humanoid_node" pkg="humanoid_control" type="humanoide_state">
        <param name="op_mode" value="1"/>
    </node>

    <!-- This launches a PHANTOM node-->
    <node name="phantom_node" pkg="phantom_control" type="phantom_control" />
</launch>
```

Launch file *pressure_cells.launch*:

```
<?xml version="1.0"?>
<launch>
   <!-- This is a launcher for pressure_cells pack-->

    <!-- This is a launches arduino node 1-->
    <node name="arduino1_node" pkg="pressure_cells" type="pressure_cells">
        <remap from="/device" to ="/device_mapper/arduino_1"/>
        <remap from="/values" to="/arduino_1_values"/>
        <param name="/tf_feet" value="/tf_left_feet"/>

        <param name="/markers" value="default_markers"/>

        <!--Cell Calibration-->
        <param name="cell1" value="5"/>
        <param name="cell2" value="3"/>
        <param name="cell3" value="2"/>
        <param name="cell4" value="8"/>
    </node>

    <!-- This is a launches arduino node 2-->
    <node name="arduino2_node" pkg="pressure_cells" type="pressure_cells">
        <remap from="/device" to ="/device_mapper/arduino_2"/>
        <remap from="/values" to="/arduino_2_values"/>
        <param name="/tf_feet" value="/tf_right_feet"/>

        <param name="/markers" value="default_markers"/>

        <!--Cell Calibration-->
        <param name="cell1" value="1"/>
        <param name="cell2" value="7"/>
        <param name="cell3" value="6"/>
        <param name="cell4" value="4"/>
    </node>

    <!-- This is a launches force_cop node-->
    <node name="cop_node" pkg="pressure_cells" type="force_cop">
        <remap from="/msg1" to="/arduino_1_values"/>
        <remap from="/msg2" to="/arduino_2_values"/>
        <remap from="/cop" to="/cop_left_right"/>
        <remap from="/cop1" to="/cop_left"/>
        <remap from="/cop2" to="/cop_right"/>
        <param name="/tf_feet_l" value="/tf_left_feet"/>
```

```
        <param name="/tf_feet_r" value="/tf_right_feet"/>

        <param name="/markers" value="/markers_cop"/>

        <param name="/feet_level" value="/feet_level_ok"/>

    </node>

</launch>
```

Launch file *haptic_force.launch*:

```
<?xml version="1.0"?>
<launch>
    <!-- This launches an haptic_force node-->
    <node name="cell_force" pkg="haptic_force" type="cell_force"/>

</launch>
```