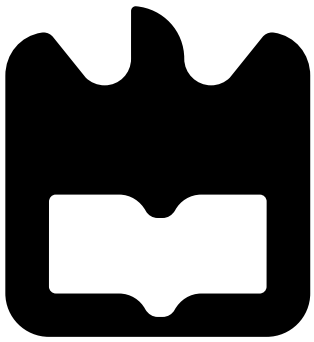
Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2017

Bruno Daniel Almeida Teixeira

Development of an anthropomorphic arm for manipulation

Desenvolvimento de um braço antropomórfico para um robô de serviço



Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2017

Bruno Daniel Almeida Teixeira

Development of an anthropomorphic arm for manipulation

Desenvolvimento de um braço antropomórfico para um robô de serviço

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Eletrónica e Telecomunicações, realizada sob a orientação científica dos Professores:
Professor Manuel Bernardo Salvador Cunha Professor Auxiliar da Universidade de Aveiro do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro
Professor José Luis Costa Pinto de Azevedo Professor Auxiliar da Universidade de Aveiro do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri / the jury

presidente / president	**Professor Doutor Filipe Miguel Teixeira Pereira da Silva** Professor Auxiliar, Universidade de Aveiro
Arguente / Arguente	**Professor Doutor Luís Miguel Pinho de Almeida** Professor Associado da Faculdade de Engenharia da Universidade do Porto, Departamento de Engenharia Electrotécnica e Computadores
vogais / examiners committee	**Professor Doutor Manuel Bernardo Salvador Cunha** Professor Auxiliar da Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

São algumas pessoas a quem gostaria de agradecer por me terem ajudado nesta fase da minha vida e no decorrer do curso de EET. As quais não posso deixar de mencionar as pessoas mais próximas nesta fase, dado que estiveram presentes nos bons e maus momentos, e que ate hoje se mantiveram ao meu lado aturando o meu mau feitio, e no entanto continuam a me dar apoio. Algumas dessas pessoas são Marta Lima, Cristiana Costa, Daniel David, irmão e os meus pais. O tempo que passei em Aveiro e as pessoas que conheci realmente compensaram o meu desenvolver como pessoa e como futuro engenheiro. Peço desculpa ás pessoas as quais não irei agradecer mas se mencionasse as pessoas todas a lista iria ser interminável e poderia me esquecer de alguém. Aos meus orientadores Bernardo Cunha e José Luís Azevedo agradeço pela oportunidade de me facultarem este projeto, aproveitando para pedir desculpas pelos incómodos causados. O projeto levou a acabo reflete o conhecimento adquirido ao longo do percurso académico bem como ainda outras técnicas necessárias para usar nesta área da robótica. Não podendo esquecer de agradecer à equipa de trabalho que se encontra no IRIS lab, assim como ao David Simões e ao Eurico Pedrosa pela sua ajuda pelo bom ambiente de trabalho.

"Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world."
- Albert Einstein

Resumo

Nos dias que correm, os robôs são usados na investigação, para uso privado, como nos programas espaciais para explorarem planetas, ou para um projeto Universitário. Este projeto terá como foco o desenvolvimento de um braço antropomórfico, e possível instalação na plataforma móvel CAMBADA@Home, para este poder participar na competição do RoboCup e conseguir alcançar mais objetivos do que aqueles que faz atualmente. Nesta dissertação irá ser explicado como os braços para o CAMBADA@Home serão desenvolvidos, explicando os motores que foram usados, motor BLDC e os servo motores, e como foram aplicados, o desenvolvimento de um modelo virtual a partir um software CAD e a construção do modelo físico. Irá também conter a informação de como a placa de controlo dos motores foi desenvolvida, o software usado assim como as ferramentas necessárias para este trabalho, incluindo o simulador usado para testar o modelo.

palavras chave: ROS, anthropomorphic arm, BLDC motor, Dynamixel servo motor, GAZEBO, MoveIt!

Abstract

Nowadays, service robots are used for private or research usage, like in space programs to explore planets, or in an University project. This dissertation is mainly focused on the development of an anthropomorphic arm and its assemble on the platform CAMBADA@Home to participate and achieve better results in the RoboCup competition. In this dissertation it will be explained how the robotic arm for CAMBADA@Home was developed. This will include the motors used, BLDC motor and Dynamixel servo motors, as well as the construction of a model of the arm with the use of a CAD software, which involves the use of the motors designed and the creation of some parts to connect them, as well as the assembly of the physical model. Afterwards the hardware and software control were developed. It was also used a simulator in order to safely test the model. Then a few test were run which consisted on the use of no load an with a load of 263 grams. Additional test were run to verify if the planning sequence matched the physical trajectory.

key words: ROS, anthropomorphic arm, BLDC motor, Dynamixel servo motor, GAZEBO, MoveIt!

Contents

List of Figures

List of Tables

Glossary

AMIGO	Autonomous Mate for InteliGent Operations. 10, 11
CAD	computer-aided design. 3, 11, 16, 17
CAMBADA	Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture. 1, 2, 10, 15, 21, 27, 30, 35, 45, 56
DFKI	DeutschesForschungszentrum für Künstliche Intelligenz. 9
DOF	Degrees of Freedom. 1, 2, 5, 12, 15
GUI	Graphical User Interface. 41
HARM	Humanoid Arm For Manoeuvring. iv, 4, 15, 16, 21, 28, 38, 43, 45, 46, 52
IFR	International Federation of Robotics. 10
IRIS	Intelligent Robotics and Intelligent Systems. 2, 10, 35
OSRF	Open Source Robotics Foundation. 45
PERA	Philips Experimental Robotic Arms. 11
ROS	Robot Operating System. 1–4, 10–12, 35–37, 45, 46, 55, 56
SCARA	Selective Compliant Articulated Robot for Assembly. 5
SRDF	Semantic Robot Description Format. 39
TIAGo	Take It And Go. 10, 12
URDF	The Unified Robot Description Format. 37, 46

Chapter 1

Introduction

Nowadays our households are equipped with machines that get smarter everyday, in the sense that they make some of our daily tasks easier and even create new needs and habits. With the increase of the capabilities in processing units and the decrease in power consumption, the creation of mobile service robots can be facilitated due to their improved autonomy and their complex systems, which allows them the ability to understand some of our requests.

A Service Robot is able to assist some of the human beings' needs, therefore it should be able to interact and communicate with humans (human-robot interaction), as well as with its surrounding environment. This kind of robots are typically autonomous and are operated by a built-in control system.

This dissertation will describe the work developed for the CAMBADA[1]@Home, focusing mainly on the development of an anthropomorphic arm, which was required for the project @Home. It will also take into consideration some of the previous work done by a previous master student [1], who also dedicated some effort into the development of such a project. It should also be mentioned that, in the global scope of these projects, human interaction, computer vision as well as navigation and mapping for CAMBADA@Home have already been developed. All of which have been integrated in the Robot Operating System (ROS)[2], an Open source community that have been giving a boost into developing robots.

1.1 Motivation

This specific robot, CAMBADA@HOME, requires the manipulation of objects which in the present state does not do so. In order to accomplish this objective, it was needed to develop a robotic arm with a gripper to allow the agent to perform such task. Tasks that would involve picking up objects, opening bottles, manipulating other kind of instruments' interface, among others. Making the focus of this project the development and implementation of two identical anthropomorphic arms with a total of 6 Degrees of Freedom (DOF) plus the gripper to be able to perform the required tasks proper of functional human limbs.

[1]CAMBADA is an acronym of Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture

1.2 Objectives

The purpose of this dissertation should be described as the understanding and the setting up of the features of an anthropomorphic arm and its development, as well as the development of a simulation environment in order to test it.

This work aims to create a usable arm with 6 DOF plus the gripper and the solution providing the control and manipulation of the arm. The model and control of the arm within ROS will also be developed for future merge within the rest of the already existing project.

1.3 CAMBADA@Home

The CAMBADA@Home was created in 2011 following a past experience from CAMBADA robotic soccer team, which competes in the Middle Size League. CAMBADA@Home is aimed to compete in the RoboCup@Home league, representing the University of Aveiro. It has already participated in the RoboCup 2011, DutchOpen 2012 and reached the second stage of the competition in RoboCup2012. The CAMBADA@Home platform, which can be seen in Figure 1.1, was designed, developed and assembled in the IRIS (Intelligent Robotics and Intelligent Systems) Laboratory. The platform consists of three layers.

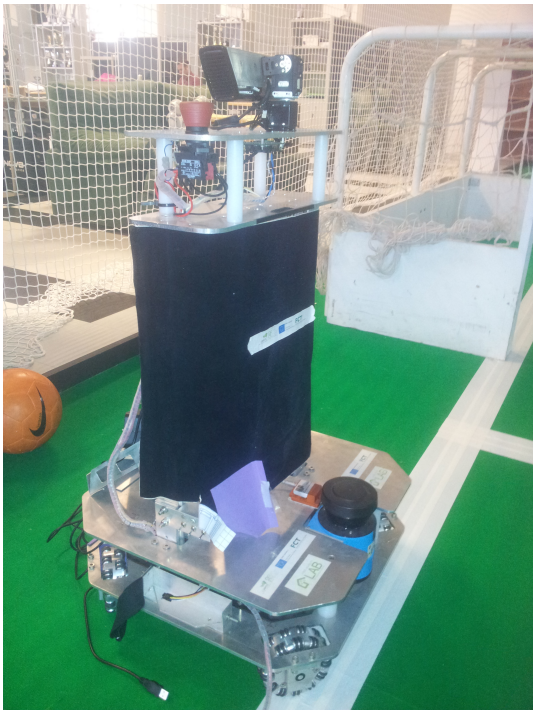


Figure 1.1: Current CAMBADA@Home robot.

The bottom layer consists of a stable four wheels configuration, by using fixed swedish wheels positioned in a 45º angle compared to its normal configuration, which allows the robot to have an omnidirectional drive, allowing it to move in every direction. This layer also contains three 4 cells Lithium Polymer (Li-Po) batteries, a SICK LMS100 laser range finder and a support for all the control hardware. This bottom layer consists on tree sub-layers, whose top is used to carry a standard laptop. By using this approach, the robot's center of mass was lowered, which increases the stability of the agent. Figure 1.2 represents the CAD model for this layer.

The second layer is represented as the torso of the robot, whose height is designed to be variable between 95 cm and 140 cm. This layer also contains a speaker and a Voice Tracker multi directional microphone array allowing Human-Robot Interaction by voice.

The top layer has the robot's vision system. Currently the robot uses a single Microsoft Kinect camera placed on top of the robot in a pand&tilt configuration, composed of two Dynamixel servos (RX-28), which allows the Kinect to capture the surrounding environment. Figure 1.3 represents the CAD model of the top layer where it is possible to see how the servo motors and the Kinect are assembled.

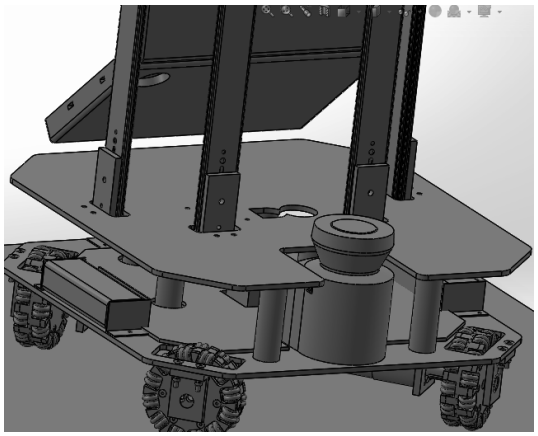


Figure 1.2: Representation of the first layer on the CAD Model.

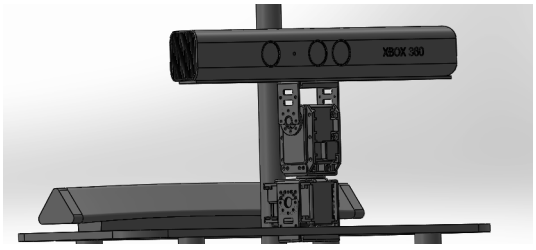


Figure 1.3: Representation of the Top layer on the CAD Model.

At software level, the Robotic Operating System (ROS) framework was adopted.

1.3.1 Robocup@Home

The Robocup@Home league [3] is aimed for the development of service and assertive robots for future personal domestic applications. It evaluates the abilities and performance of the agent in a realistic home environment, focused in Human-Robot-Interaction and cooperation, navigation and mapping of the environments. Computer Vision and Object Recognition under natural light conditions, object manipulation, adaptive behaviour, behaviour integration, ambient intelligence, standardization and system integration, must be part of the robots' skill which aim to participate in the competition.

The competition consists of challenges which the robots have to meet. The criteria for the test consists of: having a human machine interaction, be socially relevant, be application oriented, easy to set up and low in cost, and finally, take a small amount of time when performing the given tasks.

1.4 Dissertation Structure

Chapter 2: Concepts - This chapter describes some of the different classifications of service robots, as well as the different types of grippers that are available in the market. Then there will be a brief section to give some examples of robotic arms application in the medical field which requires great precision and an underwater applications to show some versatility of a robotic arm.

Since this project is dedicated to the development of a robotic arm aimed to be implemented on a service robot, some examples are given, ranging from a robotic arm to the integration of robotic arms in moving platforms and its capability for human interaction.

Chapter 3: HARM Development - This chapter presents the proposed solution, for the robotic arm denominated by HARM (Humanoid Arm For Manoeuvring): the first section describes how the arm was planned, taking into consideration some of the available motors in the laboratory, as well as the required force and weight, in order to make the robotic arm as light weighed and robust as possible.

The following section describes the used servos and why they were chosen, as well as their communication protocol.

The third section follows up with the control architecture, by explaining the development of the control board, the hardware and the communication system used. The final section will describe the BLDC motor, how it was used, its requirements, as well as the development of its control board.

Chapter 4: ROS Integration - This chapter describes the tools that were used to integrate this project with ROS. It explains the provided tools in order to develop the software. It starts with an overall description of ROS to understand how it works, then the use of the planner *MoveIt!* which uses the inverted kinematic, and the use of *Rviz* so it is possible to preview the movement before executing it. After that, the software implementation to provide an overall view of the robot's functionalities. Finally, *Gazebo*, a simulation for the physical model, will be introduced.

Chapter 5: Validation - This chapter describes the tests performed with HARM in order to validate the performance of the project and help to build a base line of the capabilities of HARM.

Once the basic tests are done to evaluate the performance, the planner integrated within this project will be used, in order to compare the outcome of what was expected, (the movement that the planner wishes the HARM to perform), with the actual outcome of the physical model, and so conclude if it performs as expected.

Chapter 6: Conclusion - In this chapter, the overall conclusion of the project is described. It mentions some possible implementations where HARM can be used and also making a few references in order to improve the work done.

Chapter 2

Concepts

When one thinks about robotic arms it usually comes to mind something similar to a human arm which is capable of operating and performing functions similar to human limbs. This tells us that the arm possesses articulated joints and links that are in between these joints, as well as an end effector, which can be represented by the gripper.

Nowadays, robotic arms are used in many fields such as, medicine, automotive assembly lines, bomb defusal and in space programs. The type of robotic arm depends on the application which is aimed at, therefore, the number of joints may vary, the payload that each one can handle as well as the precision, control, feasibility and its robustness.

2.1 Types of robotic arms

There are many types of configurations of robotic arms. However, the most important ones in the industry are: cartesian, Selective Compliant Articulated Robot for Assembly (SCARA), cylindrical, delta, polar and vertically articulated.

Each of these types have a different joint configuration, as described below.

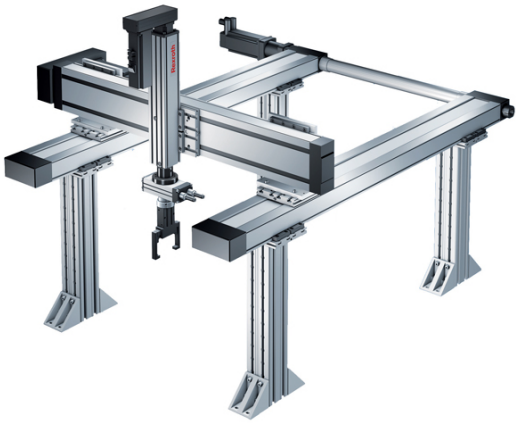
- `Cartesian`: these robotic arms have three linear joints that use the cartesian system. They can also have an attached wrist to allow rotational movement.
- `SCARA`: is commonly used in assembly application, due to their stiffness and the two parallel joints that provide compliance in one selected plane.
- `Polar`: in this configuration, the arm is connected to the base with a twisting joint. With a combination of two rotary joints and one linear joint, the axes form a polar coordinate system, creating a spherical-shaped work envelope. Some applications of this type of arm can be for loading and unloading tasks.
- `Articulated`: this design features rotary joints in which the range of joint structure can go from two to ten or more. The arm is connected to the base with a twisting joint. Each link in the arm is connected by a rotary joint, each of these joints will provide a new DOF. Industrial robots commonly have four or six axes. This configuration is commonly used for spray painting, fettling, gas welding, etc.

Some examples of these configurations can be seen in Figure 2.1.

(a) [4]

(b) [5]

(c) [6]



(d) [7]

Figure 2.1: Example of some configurations where a) is a polar configurarion b) is a SCARA configuration, c) represents an articulated configuration and d) represents a cartesian configuration.

For these types of robotic arms, the number of joints varies depending on their configuration and system needs.

2.2 Types of grippers

As there is a wide number of joints available to a robotic arm, there are also a wide number of type of grippers available. For example:

- `Vacuum`: This kind of gripper is frequently used for grasping non-ferrous objects. It uses vacuum cups as the gripping device, it is also commonly known as suction cups. This type of grippers will provide good handling if the objects are smooth, flat, and clean. With this gripper there is only one surface for gripping the objects.
- `Mechanical`: This kind of gripper is used as an end effector in a robot for grasping objects with mechanical fingers. Two fingers are enough for holding purposes, but more than three can be also used depending on the application. The actuator system can be either electrical, hydraulic or pneumatic, since it is able to produce power for the input signal.
- `Magnetic`: Magnetic grippers are most commonly used in a robot as an end effector for grasping ferrous materials, with the use of electromagnets or permanent magnets.

This kind of grippers only require one surface to grasp the material, it does not require different designs to handle different size materials and the material can have holes.

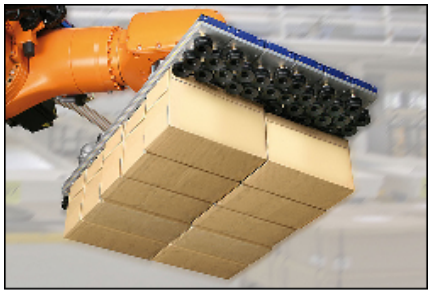


Figure 2.2: Example of a robotic arm with a vacuum gripper.

Figure 2.3: Example of a pneumatic gripper.

Figure 2.4: Example of a three-finger servo gripper.

Figure 2.5: Example of an electromagnetic gripper used to lift ferromagnetic objects.

It is worth noting that mechanical grippers can have different kinds of fingers, which can be seen in Figure 2.3 and 2.4.

Another example would be the Fin Ray from Festo [8]. It consists of two flexible bands, which meet at the top featuring a triangular format. The bands are connected by ribs spaced in regular intervals which make these fingers flexible and sturdy and allows them to adapt to the objects. In Figure 2.6, it is possible to see these kinds of gripper when no force is being applied to it and, in Figure 2.7, it is also possible to observe how it adapt themselves when holding an egg.

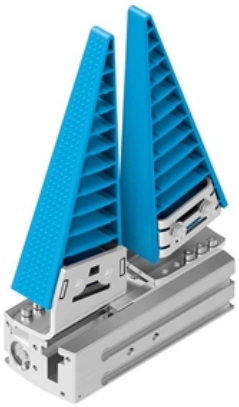


Figure 2.6: FinRay gripper in resting position with no force applied.

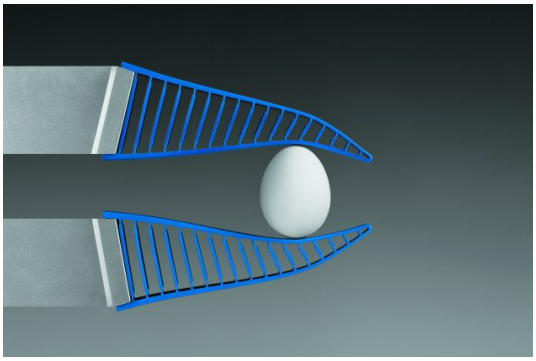


Figure 2.7: FinRay gripper holding an egg.

This was just an example of fingers which may be used in mechanical joints. While there are many other kinds available, these in particular were the most suitable, regarding the function which they are used for.

2.3 Application of robotic arms

As it was mentioned previously there are many areas where robotic arms can be applied, namely, medicine, underwater exploration, space programs, among others.

2.3.1 Robotic arms on medical field

As an example in the medical field, it is worth mentioning the robot *da Vinci Surgical System*[9] [10] , (Figure 2.8).

This kind of system needs to be stable and have high precision in order to perform the smallest movements. The controller of the robot is able to see where each arm is looking at due to a camera placed on each one. It is also possible to control each arm individually. Since it requires a high precision on each of them, they can be used for holding objects to act as scalpels, scissors, etc.

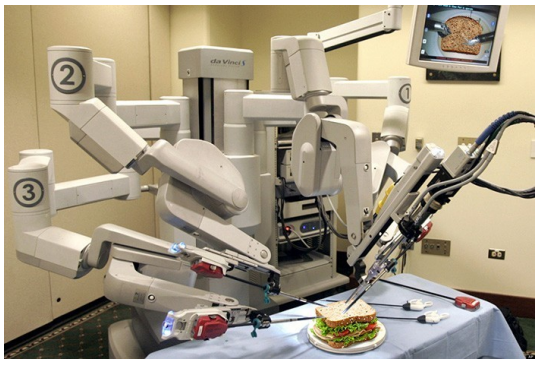


Figure 2.8: Representation of the da Vinci Surgical System.

With the stable motion of the *da Vinci Surgical System*, it is possible to use it in situations where the surgeon must look into a 2D video monitor, to observe an image of the target anatomy and have the camera in the right position, while holding the surgical tools. While most of these surgical interventions are done while the surgeon is standing, with the da Vinci System, it is possible to do all this while the surgeon is seated in front of the console.

2.3.2 Robotic arm for underwater

When it comes to robots operating underwater, the main topics that one usually focuses on are: developing a system that supports remote control, designing methods for autonomous manipulation and mission planning of robot arms in underwater applications, image processing and object recognition.

The CManipulator-Project[11] is a project focused on methods for visual detection of objects underwater and for autonomous manipulator control. This system is planned for underwater inspection and maintenance tasks which include autonomous picking, placing and connecting certain objects (e.g. an underwater plug).

Figure 2.9 is an example of a test of the arm in an underwater tested at the DFKI (Deutsches Forschungszentrum für Künstliche Intelligenz) robotics lab.

Figure 2.9: Experimental setup of the CManipulator system.

This setup is able to grasp a range of different objects up to 30 kg in water. The gripper is designed to grasp objects that are cylindrical for underwater transport and objects with a handle that enables them to be grasped by CManipulator.

2.4 State of the art

There is a necessity of comparing the service robot, CAMBADA@HOME, to the ones already existing in the market. This specific robot needs to be placed withing the exiting classes and compare it with them.

2.4.1 Service robot

According to the International Federation of Robotics (IFR), a service robot can be classified in the following classes.

- `service robot`: service robot is a robot which performs useful tasks for humans or equipment, excluding industrial automation application. Note: the classification of a robot into industrial robot or service robot is done according to its intended application [12].
- `personal service robot`: a personal service robot is a service robot used for a non-commercial task, usually by disable people. For example, domestic servant robot, automated wheelchairs[12].
- `professional service robot`: a professional service robot is a service robot used for a commercial task, usually operated by a properly trained operator, for example, a cleaning robot for public places[12].

With this information, it is possible to say that CAMBADA@Home belongs to the category of a personal service robot. For that reason, even though this project is of an academic studies level, it is possible to mention some similar projects that follow in this category, starting with Kinova JACO [13], a robotic arm with similar functionalities to the one covered in the main topic of this dissertation. Other similar projects include robotic arms integrated in a moving platform, like TIAGo[14], PR2 [15] and AMIGO [16], mainly because the main objective for the development of the robotic arm in this dissertation is to incorporate it in the platform @Home.

Kinova JACO

JACO[13] enables the user to interact with the environment. The arm moves smoothly and silently with unlimited rotation on each axis. The axes are of aluminum compact actuator discs and contain elements that are easily replaceable and maintained.

Its main structure is entirely made of carbon fibre, which makes it robust, durable and light. The arm is mounted on a standard aluminum support structure that can be attached to almost any surface, as long as it is stable and strong enough to handle its weight. The gripper consists of 2 or 3 fingers which can be individually controlled, allowing it to hold an egg, as well as a jar firmly. JACO can be controlled through software, some of which is being developed in ROS, or with a controller *Kinova's 3-axis*, and a 7-button joystick.

A representation of the JACO arm can be seen in Figure 2.10, as well as a representation of the arm mounted on a wheel chair on Figure 2.11.

Some applications involving the JACO arm have been done in the IRIS lab some of which, involve playing *tick tack toe* with the help of a Microsoft Kinect against a human player, and picking an object and placing it on a desired location.

Figure 2.10: Kinova JACO.

Figure 2.11: Kinova JACO attached to a wheelchair.

AMIGO

AMIGO[16] is an acronym for Autonomous Mate for InteliGent Operations. AMIGO is a service and care taking robot developed by Eindhoven University of Technology. It is used as a demonstrator in several projects and also competes in the RoboCup@Home league. The CAD model of AMIGO can be seen in figure 2.12 it consists of a head with a laser range finder in order to visualise the environment. Two Philips Experimental Robotic Arms (PERA) interact with its surroundings, a torso containing a spindle which enables AMIGO to translate the upper body in vertical direction and a base containing the batteries, electronics and four omni wheels. It is also worth to mention that the software is developed using the ROS framework.

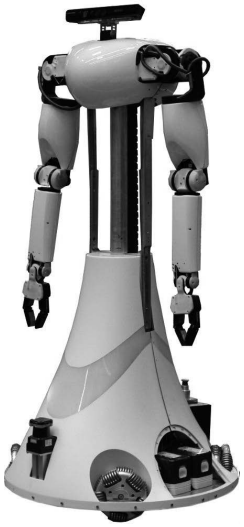


Figure 2.12: Representation of AMIGO robot.

TIAGo robot

TIAGo (Take It And Go) robot[14], was developed by PAL Robotics Team. TIAGo is a mobile research platform enabled for perception, manipulation and interaction tasks. The robot comprises a sensory pan and tilt head, a lifting torso and arms. It was designed to have a versatile hand for manipulation. TIAGo is an advanced robot to boost research areas from low level control to high level applications and service robotics. It is fully compatible with ROS and comes with multiple functionalities like, multi sensor navigation, collision free motion planning, detection of people, faces and objects and speech recognition and synthesis.

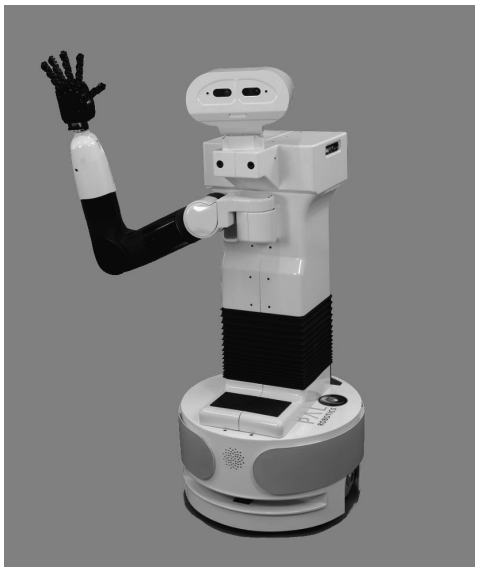


Figure 2.13: Representation of the TIAGo robot.

PR2

PR2[15] is a personal service robot. It is a combination of mobility in order to navigate in human environments as well as it is capable to grasp and manipulate objects. Its software system development is written entirely in ROS, giving access to all of its capabilities available via ROS interface. The PR2 has 4 DOF on the arm 3 on the wrist and 1 on the gripper making a total of 8 DOF. It has a maximum payload of 1.8kg. For the sensors it includes a 5-megapixel camera, a tilting laser range finder and an inertial measurement unit. Its structure is composed of a pan and tilt head, a telescoping spine so it is capable of changing its total height an omni directional base and two on-board servers.

PR2 software has been developed within ROS which provides a wide variety of libraries that can be used for the vision, navigation, grasping, etc. It is also possible to test the robot within Gazebo [17] in order to understand its behaviour and develop new applications.

Figure 2.14: Representation of the PR2 robot.

2.4.2 Discussion

The project that is proposed for this dissertation, compared with the other robotic arms, is aimed to be more affordable and lightweight in a way that anyone can build it at home and assemble it in an appropriate platform. Even though this is most desired when compared with other technology available in the market, it may have a reduced precision and lower maximum payload when compared with JACO and the arms that are present in TIAGo and AMIGO.

Chapter 3

Development of HARM

HARM is an acronym for Humanoid Arm For Manoeuvring, that was developed with the intention of implementing it on CAMBADA@Home. It would be simpler to just implement an existent robotic arm on the platform, for example, JACO, this would bring up a number of main issues: each JACO arm weights about 5.5 kg, which would change the center of mass of the platform. Also, the arms are extremely expensive, increasing the funds of the project CAMBADA@Home. For that reason, the focus of this dissertation was to develop a robotic arm which fills in the basic requirements, and try to reduce its cost and its total weight, by also reducing some performance or even precision.

The basic requirements for the robotic arm, developed in this project, consists on being as lightweight as possible, so it does not affect the stability of the current platform, be able to lift 500 grams when fully extended and have seven DOF.

3.1 Development of the arm

For a better understanding on how it was possible to connect each joint, so they would have the desired performance and resemblance to a human arm, the characteristics of the motors were taken into consideration. In order to classify the movement of each joint, in order to define them as roll or pitch movement, it was used a coordination system as shown in Figure 3.1, where the x-axis is aligned with the motor shaft.

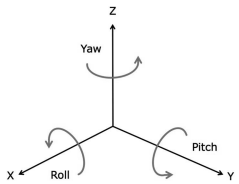


Figure 3.1: Coordination system [18].

3.1.1 Motor configuration

As mentioned previously, there was a need to understand how the motors could be assembled together, before trying to create a basic design, so, this section started by explaining most of the configurations studied with the Dynamixel servo motors.

ROBOTIS [19], already provides some of the possible combinations that are possible to use with Dynamixel servo motors. Here, it will only be discussed the ones that were used in this project.

In figure 3.2, it is possible to see the motor whose function is to rotate over its own axis. For a pitch movement it was used the configuration shown in Figure 3.3. It is also possible, to assemble two motors together, Figure 3.4, to perform a pitch movement, improving the total output force.

As it is shown in Figure 3.5, it is possible to use these configurations assembled together, in a pitch and rotational movement. These configurations were the ones used in this work.

Figure 3.2: Representation of the servo motor with its basic operation.

Figure 3.3: Representation of the servo as a pitch movement.

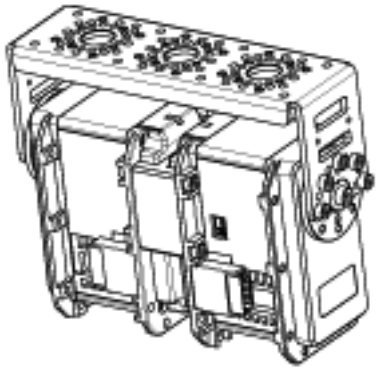


Figure 3.4: Representation of two servo motors working as one for a pitch movement.

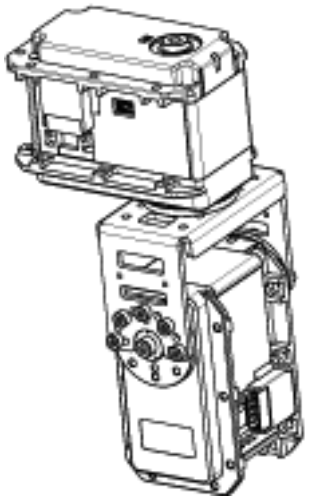


Figure 3.5: Representation of two servo motors, one that performs a pitch movement attached to one that rotates.

With these configurations, it is possible to build the CAD model and have a first look of the HARM model. The following section, will explain in detail, the CAD model's development, along with a description of the parts that were needed, in order to establish a connection between joints.

3.1.2 CAD model

To provide a general idea of the behaviour of each joint, as well as their location, it was used a general representation of an anthropomorphic arm, Figure 3.6.

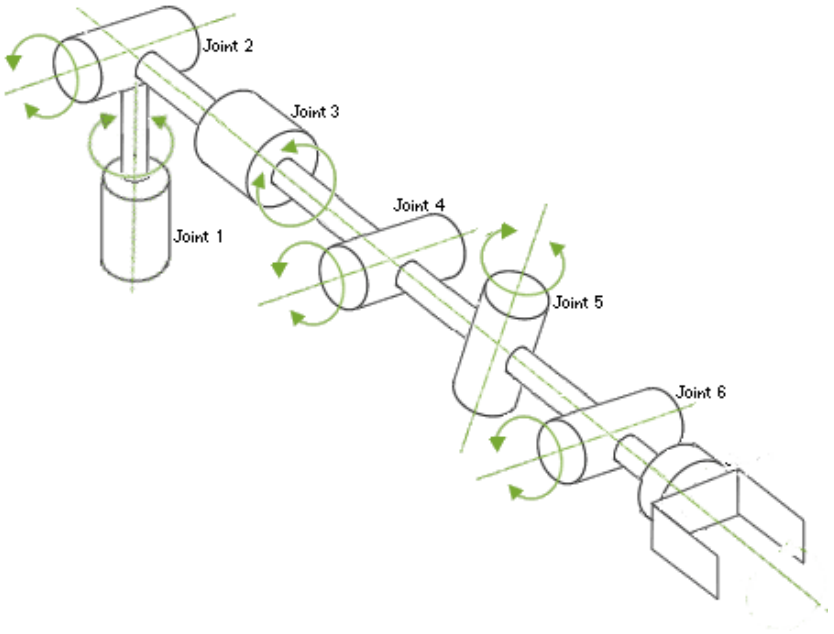


Figure 3.6: General representation of an anthropomorphic arm with 6 degrees of freedom [20].

With the basic aspect in mind and with the help of SolidWorks[1], it was possible to create a CAD model for the robotic arm, Figure 3.7. In this section, only the model will be discussed.

Within the CAD model, it was defined the terms short-link and a long-link. The term short-link is used when two motors are assembled together, in a pitch and rotating movements, as it is shown in Figure 3.5. While the term long-link is applied when it is used a carbon fibre rod or a gearbox, serving the purpose of distancing the joints.

[1] http://www.solidworks.com/sw/183_ENU_HTML.htm

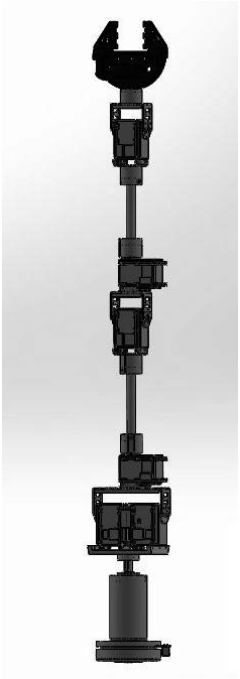


Figure 3.7: CAD model for the arm created with SolidWorks.

In the following close up, Figures 3.8 ,3.9, 3.10, it is possible to have a better understanding on how the motors were assembled.

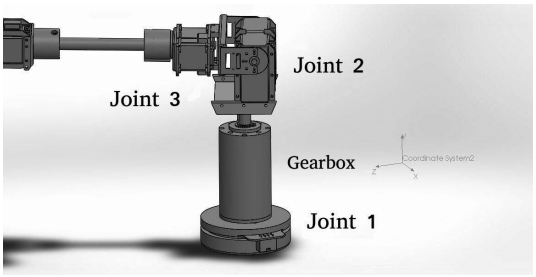


Figure 3.8: Representation of Joint 1, 2 and 3 in the CAD model.

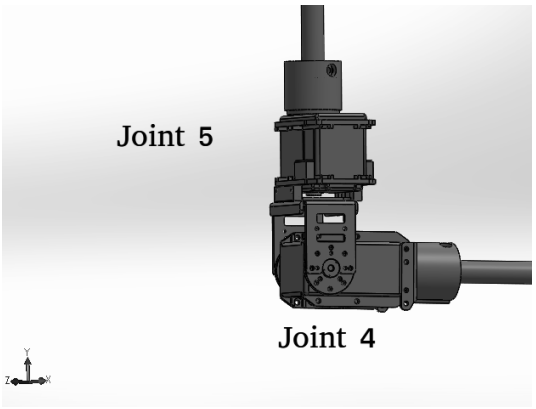


Figure 3.9: Representation of Joint 4 and 5 in the CAD model.

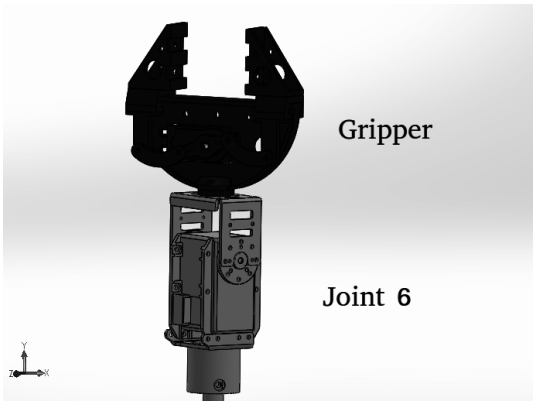
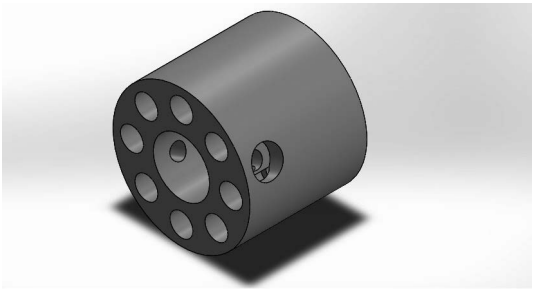


Figure 3.10: Representation of Joint 6 and Gripper in the CAD model.

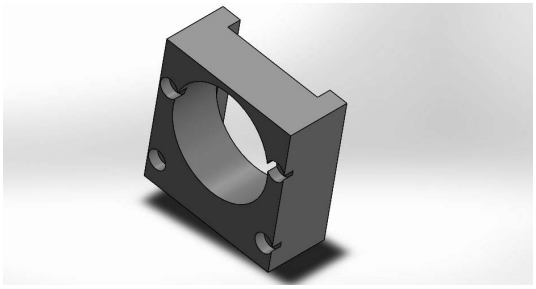
In the previous Figures 3.8, 3.9 and 3.10 ,it is possible to observe some of the parts that were developed, where most of them have been used to perform the connection between the servo motor, with the carbon fibre rod. This component can be seen in Figure 3.11a.

Since it was possible that the weight of the arm could cause the shaft, from joint 3 and 5, to bend against its axis direction. The parts represented in Figure 3.11b, were designed and placed in order to prevent possible malfunctions and make it sturdier.

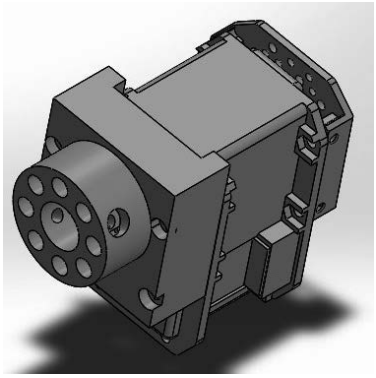
In Figure 3.11c, it is possible to see how they were assembled with the servo motor.



(a)



(b)



(c)

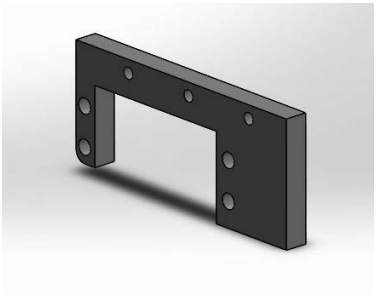
Figure 3.11: Designed CAD parts to help preventing the horn of the motor to twist (a) and b). c) is the assembly of a) and b) with the motor.

In order to use the gripper shown in Figure 3.10, there was a necessity to adapt it, since it had been designed to be used with a hobby servo, represented in Figure 3.12, instead of a Dynamixel servo motor.

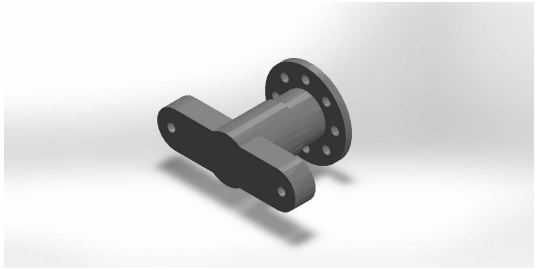
For that reason, the following parts, seen in Figures 3.13a and 3.13b, were designed. With these two combined, it is possible to hold the servo in place. Then, to control the gripper itself, an extension for the shaft was designed as shown in Figure 3.13c. In Figure 3.13d, it is possible to see all the components assembled together to make the gripper operational.

Figure 3.12: Example of a hobby servo.

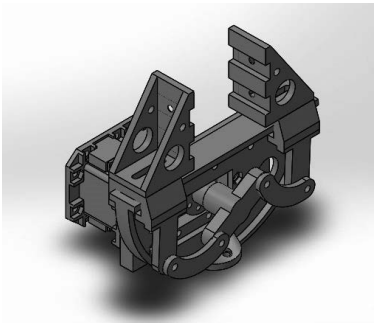
(a)



(b)



(c)



(d) CAD model of the complete gripper assembled.

Figure 3.13: Designed CAD parts to work with the gripper.

The adapter shown in Figure 3.14 was created, to connect the gearbox with joint 2.

(a) (b)

Figure 3.14: CAD model of the designed mechanical structure used to connect the gearhead shaft to the base of the Dynamixel servo, where a) is used to establish the connection between the shaft and b) and b) is used to hold the Dynamixel servo motors at joint 2.

Although this connection is used in the CAD model, it was not used with the physical model, since the hardware which provided the control for the BLDC motor was not available at the time. To test the physical model, it was created the part that can be seen in Figure 3.15, making it possible to hold the HARM at joint 2 and attach it to a solid surface.

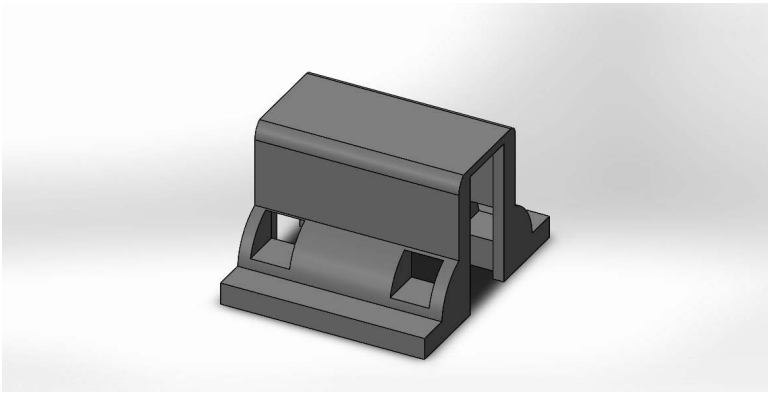


Figure 3.15: Representation of mechanical drawing to hold HARM at joint 2.

In Figure 3.16 it is possible to see this part, Figure 3.15, assembled with two servo motors working together in a configuration of a pitch movement.

After the discussion of the reasons that led to the above-mentioned connections, it is now possible to focus on the size of each link. Knowing that the CAMBADA@Home platform has 95 cm height, and the location of the shaft of the main motor is going to be located at 75 cm height, the total length of the robotic arm was planned to be 70 cm from joint 2 to the gripper, preventing it from dragging.

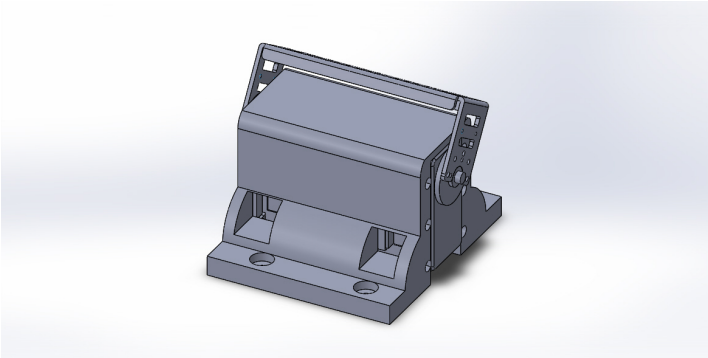


Figure 3.16: Mechanical part with the motors assembled.

As a reminder of each joint and link location, of the robotic arm Figure 3.17 was created, were it was establish the length of each link. Where short-link 1 and 2 have 8 cm of length, long-link 1 with 7.85 cm, long-link 2 have 20 cm and long-link 2 have 18 cm. The gripper has a total length of 15 cm.

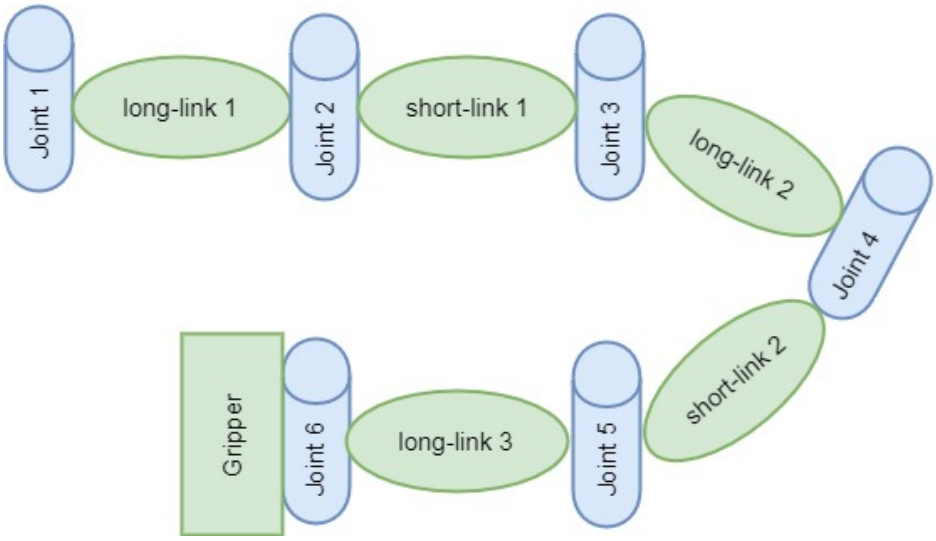


Figure 3.17: Joints and links of the robotic arm.

With the lengths chosen, it is possible to select the appropriate Dynamixel servo motors according to their specification, which it will be discussed next. Also, it will be explained why a reducer was planned to be used with the BLDC motor.

3.1.3 Motor Selection

In this project, two types of motors were used, the BLDC and the Dynamixel servo motors. The BLDC was discussed firstly, followed by the Dynamixel servo motors.

BLDC motor

In this work, it was planned to use a single BLDC motor, which would be applied at joint 1. However it was not used or tested, due to the control board not being ready at the time being. The motor used was the EC-90 series 323772[21], from maxon motors, Figure 3.18, whose characteristics can be seen in Table 3.1.

Figure 3.18: Representation of the BLDC motor [21].

Table 3.1: Motor BLDC maxon-EC-90 24V 90W general description.

EC-90	
Operation Voltage	24V
Nominal Power	90W
Stall Torque	50.373 kg∗cm
Nominal current	6.06A
Nominal Torque (max. - continuous torque)	4.527 kg*cm
Max. efficiency	84%

Since there was no information available, about the starting torque required for the motor to start moving, it was considered that it would take half of the stall torque value.

By applying the basic design of the robotic arm, it was possible to calculate how much force it would have, with the use of the torque Equation 3.1, where τ represents the maximum torque, **F** being the force and **d** the distance from the shaft. Since the maximum distance from the shaft to the tip of the robotic arm is 70 cm, the motor provides a total force of 0.359 kg (considering no losses).

$$\tau = F * d \tag{3.1}$$

Then there is still a need to consider the nominal torque, to have a continuous movement without overheating the motor. In which the value for the maximum force at a distance of 70 cm would be of 64 grams. The obtained values are still far from the main objective of being

able to handle 0.5 kg and there is still a need to consider the total weight of the robotic arm, which is 1kg. Although the total weight will be driven along the arm, for our calculations it will be considered that all the weight will be located at the end effector.

Taking this into consideration the motor needs to surpass a total force of 1.5 kg, at a distance of 70 cm. For that reason the BLDC motor was already assembled with a planetary Geardhead (maxon Planetary Gearhead GP 52 series 223091 [22]) seen in Figure 3.19, with the specifications present in table 3.2.

Figure 3.19: Representation of the Planetary Gearhead [23].

Table 3.2: Planetary Gearhead GP 52 series 223091 [22].

Gearhead Data	
Reduction	66:1
Absolute reduction	1183/18
Max. continuous torque	305.91kg*cm
Max. intermittent torque	458.87kg*cm
Max. transmittable power (continuous)	290W
Max. efficiency	75%

With the applied gearhead the output starting torque would be improved to 1034 kg*cm (with the maximum efficiency already taken into consideration). Unfortunately, the value would surpass the maximum intermittent torque, allowed by the gearhead, which would shorten the life of the component, since it will be used for position control.

Then there is still a need to verify the conditions for the nominal torque, to guarantee that the motor will not overheat, taking the maximum efficiency into consideration the nominal torque obtained at the output of the geardhead would be of 188.2 kg*cm, fortunately this value will not reduce the life time of the component, since the value obtained is within the parameters.

With the use of the torques equation 3.1 and taking into consideration the total length of 70 cm, it would give a total continuous force of 2.68 kg, at the end effector, meaning that even though it was considered a weight of 1.5 kg, it is possible for the arm itself to have a little more weight than what was considered, to maintain a continuous movement.

Even though the hall-sensors from the BLDC motor are used to drive its half bridge, it would not be wise to use those them, to detect the position of the gearhead's output shaft, since it would be bound to errors, due to the energy transfer between the motor and the gearhead, the reduction, among others.

For the reasons mentioned, an absolute encoder AEAT-6012 [24] and a magnetic encoder with 12-bit resolution (ideal for angular detection within 360°) were attached to the output shaft of the gearhead.

For the remaider of the joints only Dynamixel servo motors were used. Since ROBOTIS [19] has a wide variety of Dynamixel servo motors, which can be used in different combinations, the next section will be devoted them. It will be also explained the chosen motors for each joint.

Dynamixel servos

The Dynamixel servo motor, is an actuator system developed for connecting joints on a robot or in a mechanical structure. They are designed to be modular and daisy chained (since they have a simple connection structure), in order to obtain fluid movements on most mechanical designs. The Dynamixel is a model actuator, which incorporates most of the functions that are required on the robot joints, as it can be seen in Figure 3.20.

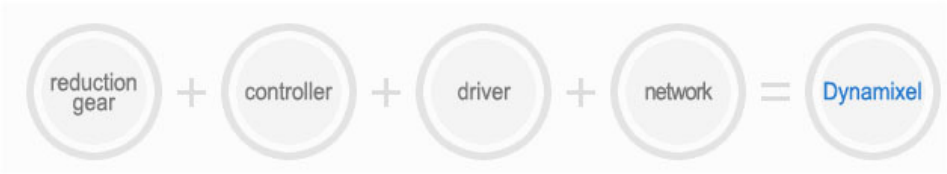


Figure 3.20: Actuator model which is incorporated in Dynamixel, image taken from [19].

The control network is able to access the actuator information, so it can be read and written through a data packet stream. The Dynamixel servo motors have an unique ID between them, and it is referred in the data packet, allowing to communicate with multiple servos at once. Depending on the model used, the physical network can be TTL or RS 485.

In Figure 3.21, it is possible to observe some of the Dynamixel servo motors available in the market.

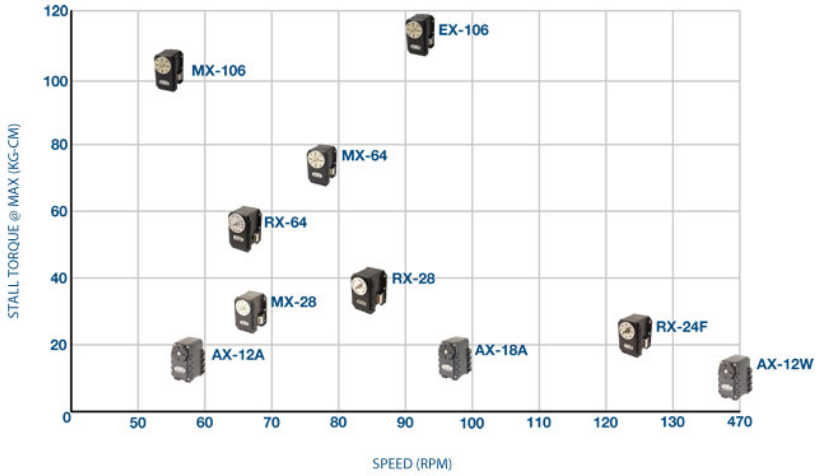


Figure 3.21: Chart of different Dynamixels servos, with stall torque as a function of speed [19].

In analysis of the previous Figure, it is possible to have a basic idea of each servo motors stall torque as a function of their speed. By only taking into consideration the stall torque, it is possible to have an idea of which servo motor can be used, at each joint.

Selection according to the specifications

When choosing the Dynamixel servo motors, there was a trade-off between the maximum torque as well as their weight. By analyzing the available motors in the laboratory, it was possible to choose those who have the necessary requirements, Table 3.3.

Table 3.3: Servo Dynamixel General Description.

Servo	MX-106R			RX-64R		RX-28R	
Operation Voltage (V)	14.8	12	11.1	18	14.8	16	12
Stall Torque (kg*cm)	102	85.6	81.5	64	53	37.7	28.3
Weight (grams)	153			125		72	
Operating Angle	360° or Continuous Turn			300° or Continuous Turn			
Protocol	RS 485 asynchronous Serial						
Dual Joint*	Yes			No			
Control	PID			Slope			

* Dual joint is a master-slave configuration between two motors, with the use of a synchronisation cable, allowing 2 motors to work simultaneously and allowing for certain configurations to double the output torque.

During the prototype phase, it was used a voltage of 14.8 V for all Dynamixels, since all the servos mentioned, in table 3.3, can operate at that voltage. This would avoid the use of DC/DC converters, which require extra hardware during the prototyping phase. For this reason, the RX series will not be able to work at full capacity. It is also worth to mention, that the communication protocol used by these motors, is the RS 485.

By taking into consideration their stall torque, as well as the location of each joint, Table 3.4 was created. In this Table it was taken into consideration, the maximum distance (dmax) that each motor would have to handle and it was considered that all the weight is located at the end effector. The total weight that it is going to be considered is the weight of the motors, as well as the maximum payload of 500 g.

Table 3.4: Minimum torque required for each joint considering the maximum payload of 500 g plus the highest weight of the servo motor available (153g), times the number of joints it is going to support.

	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
dmax (cm)	65	38	38	11	11
Minimum torque (kg*cm)	71,724	36,442	30,628	7,183	5.5

It is to note, that the typical stall torque only provides an estimate for the maximum payload, the minimum load torque required to stall a spinning motor. The minimum load torque required to prevent a motor from starting is expected to be lower, due to the inertia

of the system. ROBOTIS advise to use 1/5 or less of the stall torque in order to guarantee stable motions[19].

Table 3.5 was created to have an idea of the torque of each motor, in order to maintain stable fluid motions.

Table 3.5: Stable fluid movements guaranteed according to ROBOTIS.

	MX-106R	RX-64R	RX-28
Torque (kg*cm)	20.4	12.8	7.54

Table 3.5 establishes the choice's for the several joints. Joint 2, to use two servos MX-106R, with a dual joint configuration. Joint 3 and 4 used a MX-106R. Joint 5 and 6 it was used a RX-64R. The gripper, a single RX-28 servo.

Assuming that each motor is working at full capacity, the maximum payload that each one can handle, with a stable and fluid motion at the end effector will be: joint 2 - 0.627 kg, joint 3 and 4 - 0.536 kg, joint 5 and 6 - 1.09 kg. These values were obtained using the Equation 3.1, the information in Table 3.5 and the distance present at Table 3.4.

Even with this selection, it is not guaranteed that the arm is going to have a stable fluid motion when dealing with 500 grams, since there is still a need of handling its own weight. In order to improve the movement in joint 3 and 4, it could be used a dual joint configuration. This would increase the weight of the arm and at the same time, degrad the behaviour of joint 2. For those reasons it was chosen to use the selection of motors already establish.

With the motors selected the physical model was assembled, with the parts that have been designed in this work, section 3.1.2, the selected motors, as well as the parts provided by ROBOTIS. Figure 3.22 is an actual picture of the physical model, and as previously mentioned, the arm was assembled without the BLDC motor and it was only used the Dynamixel servo motors. For safety reasons, the servo motors were configured to have their limits at +90 and -90 degrees. The origin position of the arm (when they read 0 degrees) is the same as it is represented in Figure 3.22.

This concludes the selection of motors, which were used in the robotic arm. With the known communication protocols required for each motor, it is now possible to start the development of the communication board and its implementation.

The next section will describe the communication protocol used in this project, the RS 485 which is needed for the Dynamixel servo motors, the CAN protocol, which will be needed in order to integrate this project with the current CAMBADA@Home platform and it will also be discussed the selection of the controller used for the BLDC motor.

3.2 Communication architecture

In order to propose a communication architecture, there was a need to understand the current architecture implemented on the CAMBADA@Home, so it would be possible to integrate this work with the existing one.

The current architecture possesses a CAN network, in which it is already implemented the motion wheels with odometry and pan&tilt nodes. The CAN network is responsible to

Figure 3.22: Anthropomorphic arm mounted vertically on plywood.

act as the nervous system of the robot, in the sense that each node can control a different low-level sensing and actuation system, where the communication with the processing unit is made through a CAN bus.

For those reasons the HARM node will need to be integrated in the same CAN network. In figure 3.23, it is possible to view the CAN network for the @Home platform. The HARM node needs to be able to interact with different devices, Dynamixel servo motors, BLDC motor, encoder, have a debugging feature and all of the communication protocols required. In Figure 3.24 it is possible to see the representation of the HARM node.

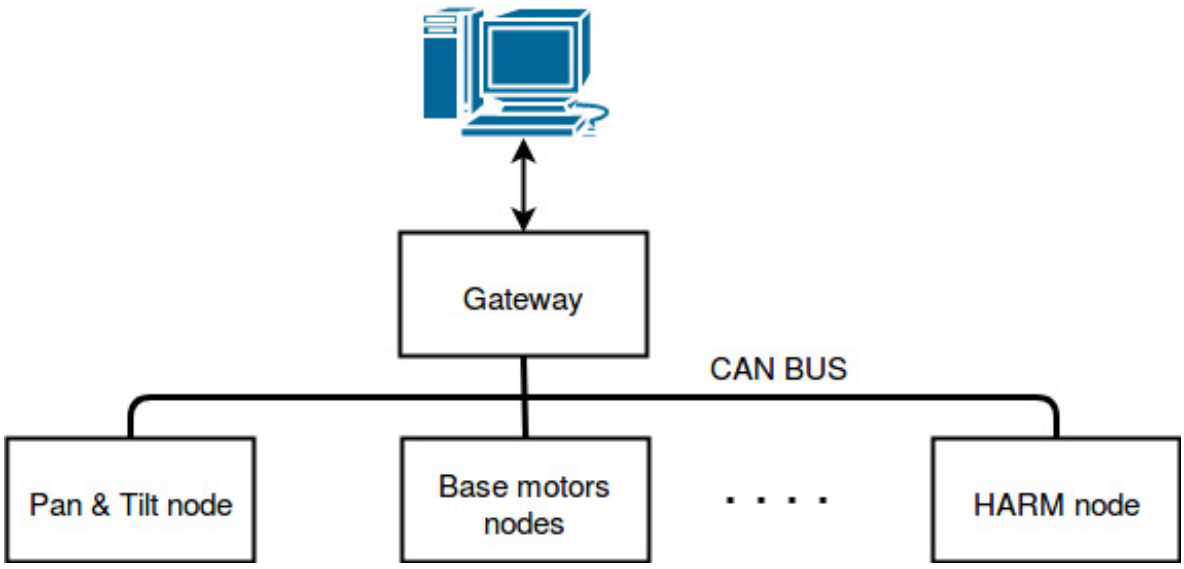


Figure 3.23: Representation of the big picture of the control architecture implemented.

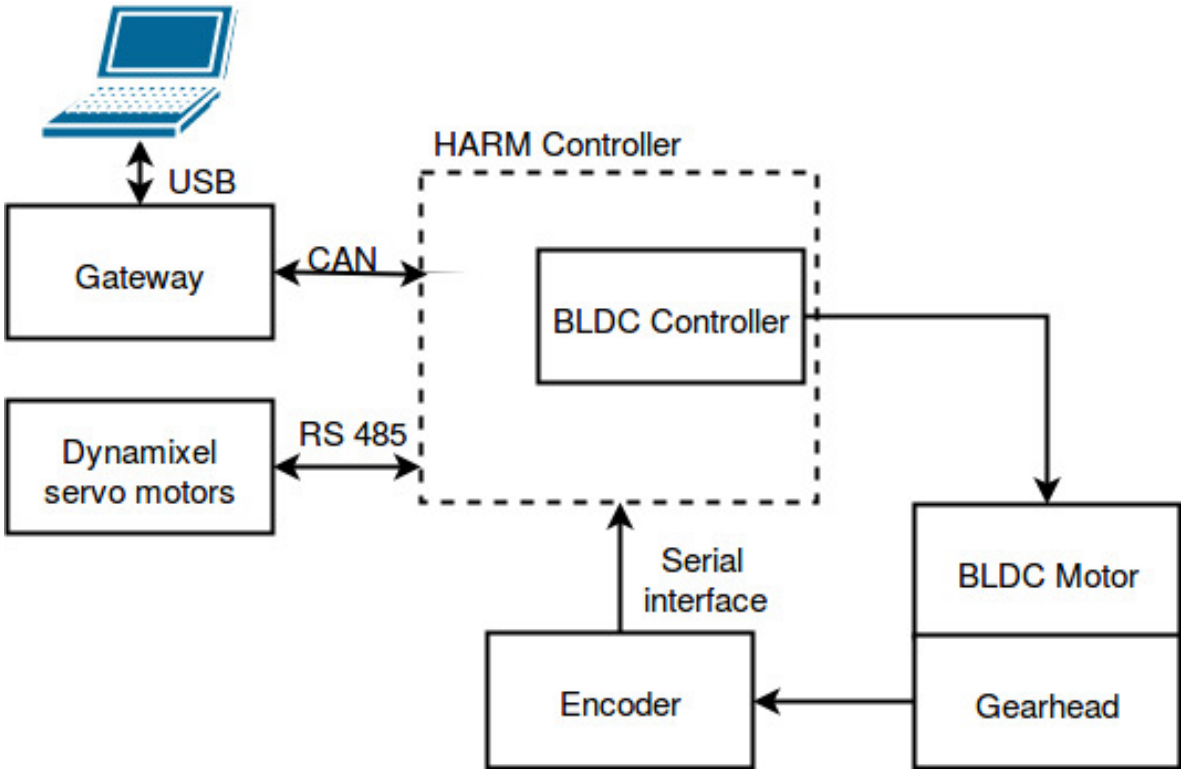


Figure 3.24: HARM node communication diagram.

3.2.1 Development of the Controller board

A previous project[1] has already shown some work when it comes to the development of the communication board. After an attentive analysis of the work, it was clear that the communication part with the servo motors was possible to reuse, but the part that was able to control the BLDC motor had to be remade, due to the fact that the controller used, FCM8201 [25], has since become obsolete.

As is was mentioned in chapter 1.3, the power supply is composed of three four cells LiPo batteries which supply all the logic and power devices on the CAMBADA@Home platform. When a battery is fully charged it supplies 16.8 V, more than enough to supply all the logical circuits. For those reasons the CAMBADA@Home platform supplies 7.5 V for all the logical signals, by using a step-down DC to DC converter. For the HARM control board, it is required to have a supply voltage of 5 V and 3.3 V, this voltages can be obtained with two linear regulators, KF50B [26] and MCP1703-330 [27] respectively.

In order to supply the motors of the platform, two batteries in series were used. By doing so, a voltage of of 30 V is made available to them. The BLDC motor from the robotic arm as well as the base motors can work directly with 30 V. For the Dynamixels servo motors it is required to use voltage regulators to obtain 18 V, to be used with the RX series and 14.8 V for the MX series.

The actuators in this project are all rotational ones, where the Dynamixel servo motors have an inner control which enables an easy way to interact with them, in which they provide access to the position, velocity, torque, among others.

For the BLDC motor, there was a need to find a controller for it, in which the controller. The controller needs be able to drive the half bridge, according to our requirements, as well as being able to detect errors, such as low voltage, over current, among others. It was also used an encoder to detect the position of the gearhead's output shaft. The development of the controller for this motor will be later discussed in section 3.2.2.

The following section, will reintroduce the elements that have been previously selected for communicating with the servo motors, as well as the chosen μC, that will deal with the communication (CAN, RS 485, TTL, etc.), actuators and sensors.

Previous work

This section, will start by explaining the chosen components that have been selected in a previous dissertation [1]. Since these components were still available on the market and still fulfills the requirements, that were necessary for the implementation of the communication protocol needed for the system and the Dynamixel servo motors. The μC used was the PIC32MX795F512H [28], whose description can be seen in table 3.6.

This μC provides all the I/O and PWM pins, as well as UART, SPI, CAN, I^2C communication models needed for this project. The UART protocol helps with the implementation of TTL and RS-485 protocol, which are required to communicate with the Dynamixel servo motors, that either use TTL or RS-485 protocol depending on the series.

Table 3.6: PIC32MX795F512H Features.

Microchip PIC32MX795F512H	
Recommended Voltage	3.3 V
Maximum Frequency	80 MHz
Flash Memory	512 KB
RAM	128 KB
Timers 16-bit	5
Timers 32-bit	2
UART	6
SPI	3
I^2C	4
CAN models	2

The RS-485 protocol is a half-duplex asynchronous serial communication protocol. It uses a UART of the μC to generate data with 8 data bits, 1 stop bit and no parity. In order to implement this physical protocol, it is necessary to add an extra component, the transceiver MAX3362 [29]. This device is composed by a differential transceiver, that consists on a line driver and receiver, which allows to select the direction of the data flow. Here the μC acts as a master and the Dynamixel actuators as slaves.

The TTL protocol, a half-duplex asynchronous serial communication protocol, it uses a UART of the μC to generate data with 8 data bits and 1 stop bit. The protocol does not allow to transmit and receive data at the same time. Therefore, one device is transmitting while the other ones needs to be listening. In order to implement this protocol in this project it was used an internal UART of the μC and a quad buffer MC74LCX125 [30] to control the direction of the data flow.

For the CAN protocol it is used a transceiver, MAX3051 [31], responsible for the establishment of the communication between the CAN controller and the CAN bus lines.

This concludes the overview communication system which was possible to reuse. The next section will explain how the control board for the BLDC motor was developed.

3.2.2 Control board for the BLDC motor

The controller that was previously chosen, in the last dissertation [1] was the FCM8201 3.7, but since then it became obsolete. With some research of the available controllers in the market, there were three that satisfied the requirements. One of them can be provided by maxon motors, but the price for this controller [32] was too high when compared with the other ones available, A4910 [33] and A3931 [34]. On Table 3.7 it is possible to see the general description of these controllers. In this table it is also possible to see the FCM8201 [25] description.

Table 3.7: Difference between controller A4910, A3930-1 and FCM8201.

	A4910	A3930-1	FCM8201
communication	Serial	Parallel	SPI
Function	Controller - Commutation, Direction Management		
Output Configuration	Pre-Driver - Half Bridge (3)		
Input Logic	inverted	direct	direct
Read Hall Sensors	No	Yes	Yes
Integrated charge pumps	Yes	Yes	No

The controller A3931 was chosen over the A4910, for being able to read the hall sensors directly. While the A4910 required that the μC processed the information gathered from the hall sensors and then communicate with the controller, to be able to drive the half-bridge.

The controller A3931, has charge pumps incorporated, avoiding any extra hardware to drive the mosfets that compose the half-bridge. This controller works by providing a logical input signal for the direction, break, reset and a PWM for control. The controller also provides two fault flags, (FF1 and FF2, logical output signals). These flags are used to detect possible errors, for example logical fault, undervoltage, overtemperature, among others. All logical signals are connected to the μC via optocouplers.

3.3 Summary

This chapter starts with the introduction on the composition that the robotic arm should have. Afterwards it explains how it is possible to combine the Dynamixel servo motors together.

It was also mentioned the CAD model and how it was created, by explaining some of the components that were required, to perform some connections between joints. After presenting the final design, it was discussed the selection of the Dynamixel servo motors, based on the requirements which were needed at each joint. In the end, it is possible to see how the communication board was created and developed, in order to communicate with the Dynamixel servo motors using the RS-485 protocol, as well as the controller chosen in order to drive the BLDC motor.

In Figure 3.25 is the representation of the overall communication board. Here it is possible to see the connections used, as well as their names.

In the Anex it is possible to see the schematic of the developed board in page 62 and 63. In page 61 of the Anex it is represented the CAD view of the PCB.

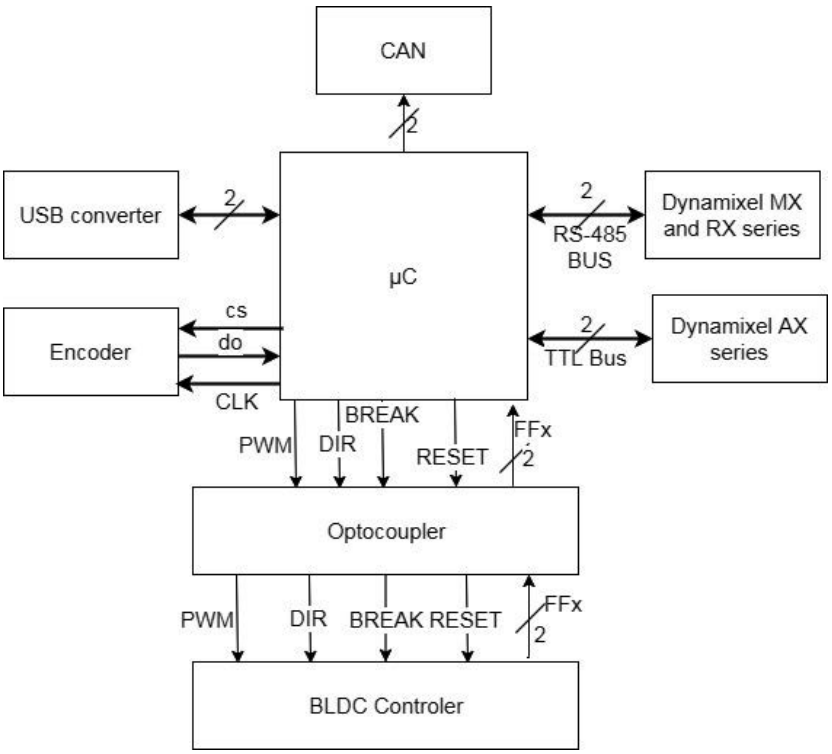


Figure 3.25: Communication done and pinage usage seen by micro controller.

Chapter 4

ROS Integration

To develop an interface between the user and the robotic arm, two large software applications were used. Those being the following: MoveIt! [35] and GAZEBO [17]. It was also used the Robot Operation System (ROS) [2] framework. These applications and framework will be discussed in detail along this chapter.

4.1 Robot Operation System

ROS is a flexible framework for writing robot software. It has a collection of tools like live plotting a 3D visualisation and processing the sensor data. There is also a collection of libraries that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms.

This framework is completely free and has already been used in the IRIS lab for projects like ROTA and CAMBADA@Home. For the CAMBADA@Home project, it has already been developed, within ROS, the computer vision, navigation and mapping . For these reasons, it would be wise to develop the application, which is going to be used for this project, in ROS, so it would allow a possible merge with the existing software.

In order to develop the software, there was a need to understand how ROS works.

4.1.1 Computation graph level

ROS computational graph level is based on a peer to peer network of ROS processes, in which the basic elements that it uses are: nodes, messages, topics, services, master, namespace.

- `Nodes`: Nodes are processes which perform computation. Each ROS node is written with ROS client libraries such as roscpp and rospy. By using the client libraries API, it is possible to implement different types of communication methods [36].

  A robot controller can have multiple nodes to perform different kinds of tasks, for example controlling motors, processing image from the kinect etc. It is also possible to use multiple nodes, in order to build multiple but simple processes, instead of only using one to process all the information. This would make it easier to understand their behaviour and isolate possible errors.

  Nodes are represented as an ellipse with its given name inside it, as it is represented in Figure 4.1.

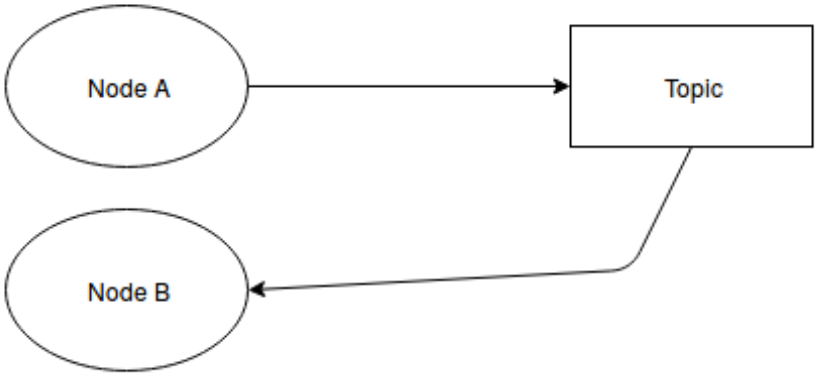


Figure 4.1: Example of "Node A" publishing in "Topic" and "Node B" is subscribing from "Topic".

- `Messages`: Nodes communicate with each other with the use of messages, a simple data structure known to both nodes. These messages can have various types like integer, floating point, boolean, among others, which are supported by ROS messages. It is also possible to build different types of message structures using different combinations of standard types [36].

- `Topics`: Each message in ROS is transported using named buses called topics. When a node sends a message through a topic, is publishing a topic. When a node receives a message through a topic, is subscribing to a topic. The publishing node and subscribing node are not aware of each other's existence. It is also possible to subscribe a node into a topic that might not have any publisher. Each topic has a unique name, and any node can access this topic and send data through it, if it possesses the right message type [36].

  Figure 4.1 represents two nodes and one topic, Node A is publishing into Topic and Node B is subscribing from Topic

- `Services`: Services are a well defined pair of message structures, containing a request and a reply. When sending a message, one expects a confirmation of its acceptance. [36].

- `Master`: The ROS Master provides the name registration and lookup for all ROS services. Without it nodes would not be able to find each other or exchange messages or even invoking services [36].

- `Namespace`: A namespace is used to organise topics as well as nodes which belong to a structure. This process establishes the difference between nodes and services which might share a similar name, as the example bellow represents

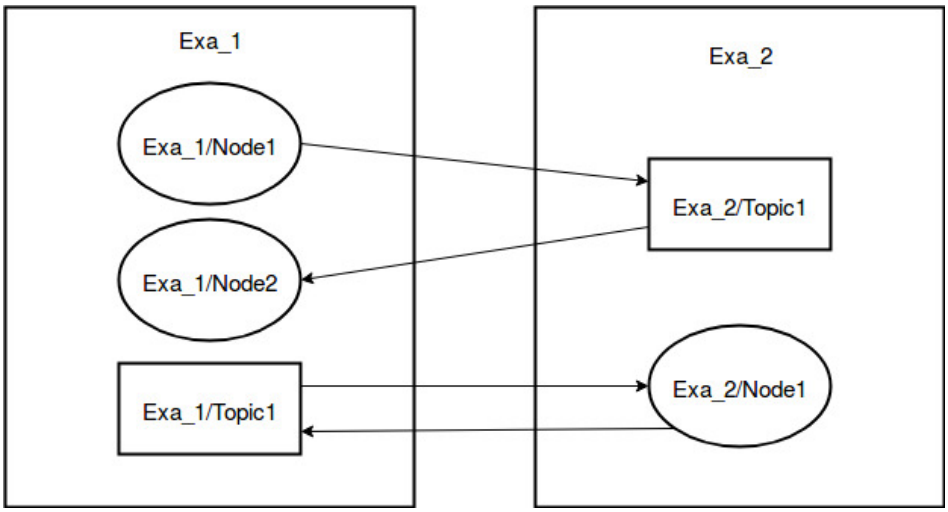


Figure 4.2: Representation of nodes and topics publishing and subscribing between different namespaces.

This is the basic communication graph level, which is required to understand in order to develop the software for this project.

4.1.2 Unified Robot Description Format

The Unified Robot Description Format (URDF) is an XML dialect used in ROS, which describe the model of a robot. As an example it is used to define the visual model representing the robot in Gazebo[17]. To apply it on Gazebo it is possible to join specific elements to the simulation environment with the use of an extension. The URDF lets the user describe the physical structure of the robot, and correlate it with the model collisions and define some of its physical properties, like the friction coefficient between links, the dynamics of the joints, etc. Even though it is possible to build a model of the robot, it is not possible to add sensors to it and some relevant physical components like kinetic friction and static friction. In order to do so, it is needed to add the Gazebo extension to have access to those elements.

In anex 7.4 it is possible to see the structure of the model used in this work.

4.1.3 Coordinate Frames and Transformations

Normally robotic systems need to track spatial relationships for a wide variety of reasons: between a moving part of the robot given a fixed frame for reference localization, between various sensors and manipulator frames, or for simple pick and place objects for control purposes.

To simplify the treatment of spatial frames ROS provides a transformation system/package called tf. The tf package keeps track of all the transformations and by doing so it is able to construct a dynamic transformation tree, which correlates all the frames of the system. As information streams in form of subsystems of the robot, (this includes joint encoders, localization algorithms, etc), the tf package produces streams of transformation between nodes on the tree, constructs a path between the desired nodes and performs the necessary calculations. An example that can be used thanks to the tf package is the possibility to perform forward

kinematic and know the location of end effector of the robot by knowing the relative movement at each joint and the dimension of each link.

4.2 HARM integrated with ROS

Figure 4.3 represents the topics and nodes used in this work, in order to communicate with the robotic arm. These nodes and topics are the communication that the anthropomorphic arm creates when launched.

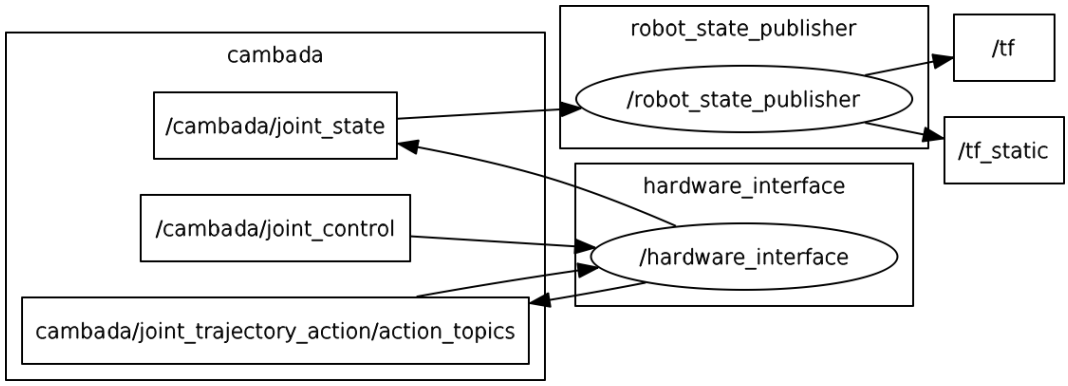


Figure 4.3: Representation of Topics and Nodes needed for a base communication with HARM.

Each Topic that is present in the namespace "cambada" contains the following information:

- `joint_state`: Contains the information of the current position of each joint. The message structure of this topic is the following:
  - `string[] name`: Name of each joint available
  - `float64[] position`: Current angle read by each joint
  - `float64[] velocity`: Current angular speed of each joint
  - `float64[] effort`: Current effort of each joint
- `joint_control`: This topic exists for the debugging mode to control each joint individually, to verify if everything is in order and if the angle we sent corresponds to the actual angle in the arm, this topic contains the following information:
  - `string[] name`: Name of each joint to be controlled
  - `float64[] position`: Goal angle for each joint
  - `float64[] velocity`: Angular speed that each joint will use to reach their goal position
- `action_topics`: This topic contains all the information of the trajectory of each joint, as well as their velocity and accelerations. It also contains feedback from the hardware interface. The action_topic will be used by the planner which will be described on section 4.3. The action_topic contains a series of subtopics such as, feedback, goal, result, status and cancel. Among them it will be only focused the goal subtopic, since it is the one that contains the information provided by the planner which contain the trajectory to be performed. The message structure of this topic contains the following information:

- `joint_names[]`: contains the name of each joint in an array.
- `JointTrajectoryPoint[] points`: Array that contains the information of all the points calculated for the whole trajectory.

  float64[] positions: the angular position that each joint should be at a given time

  float64[] velocities: angular speed that should be applied at each joint

  float64[] accelerations: accelerations that each joint should have

  float64[] effort: the possible effort present.

  duration time_from_start: time that occurred since the movement has started

The node hardware_interface is responsible for processing all the information that needs to be published and subscribed from the namespace cambada. The hardware_interface is responsible to communicate with HARM, so it can provide the information from the motor controller and publish it in the correct topic, as well as subscribing from the correct topic and send the commands to the motors.

4.3 MoveIt!

MoveIt! provides the capability which allows it to work in a human presence environment, since the robots needs to avoid the collision with humans, as well with other obstacles. Since MoveIt! [35], is capable of performing such tasks and has been integrated with ROS, it allows the robots to have a representation of the environment around them, using data fused from a three-dimensional and other types of sensors and is capable of generating motion plans. For these reasons it was chosen to use MoveIt! as the planner for this dissertation.

In order to fully understand how MoveIt! works, a representation of its high level system communication can be seen in Figure 4.4. Here it can be seen the communications provided by MoveIt (represented as the move_group), this node serves as an integrator that pulls all the components together to provide the actions and services present in ROS, allowing the user to operate with it.

The "move_group" communicates with the robot through ROS topics and actions. In this way it is possible to get the current state information from the encoders of each joint, as well as communicating with the controller of the robot.

4.3.1 The move_group node

The "move_group" node is able to integrate various components of the robot, working as the heart of MoveIt!. From the architecture present in Figure 4.4, the "move_group" gathers the information from the "Robot Sensors" and "Robot State Publisher". It collects the robot kinematic data, such as the URDF and the configurations from the parameter server. A Semantic Robot Description Format (SRDF) file, and the configurations files are generated once the MoveIt! package of the robot is created. The file contains the information about the joint limits, maximum velocity and acceleration, which link represents the end effector, etc. Once MoveIt! is able to collect all the information about the robot, it is possible to start commanding it from the user interface.

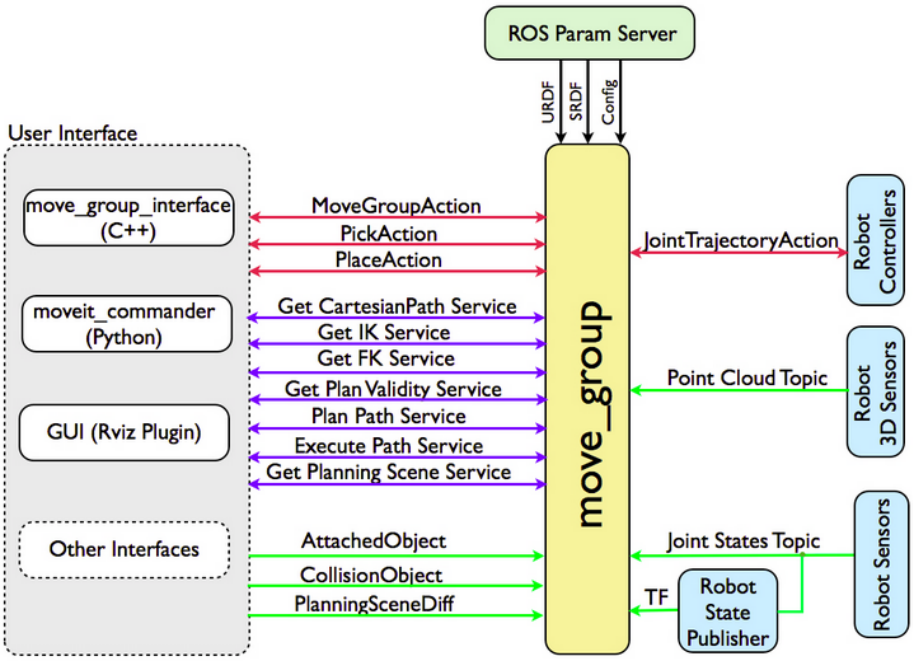


Figure 4.4: MoveIt! architecture diagram [37].

4.3.2 Motion planning

The motion planning plugin interface allows MoveIt! to communicate with different motion planners' libraries, which allow the planning of a trajectory and the avoidance of obstacles. In a typical application, regarding the topic of this project, it would be provided a new location for the end effector. One can also attach an object to the end effector, to perform a pick and place action, allowing the planner to consider the motion of that object, while planning trajectories.

The motion planning result will generate a desired trajectory according to the planning request. This will lead the robot to the desired location, by providing the trajectory which it should take at each moment, as well as its velocities and acceleration.

With the use of this tool and the current arm model, created with the CAD software , that had been previously used, the model was load and it was possible to start planning movements by providing the right Cartesian coordinates and orientation of the end effector.

When a valid goal position is provided for the end effector to reach, MoveIt! will start planing a possible trajectory.

MoveIt! calculates multiple trajectories, but it will only provide one of them. The provided trajectory might not be the best one when considering the work done by each joint, since the current planner is optimised to take the shortest trajectory. Once a trajectory is available it will start the movement.

Since providing a valid goal position by hand has proven to be somewhat difficult a visualizer (Rviz) was used, this lets the user to choose the target location and it also provides a visualisation of the trajectory before executing it.

MoveIt! can be configured by running the command *"roslaunch moveit_setup_assistant setup_assistant.launch"*. With the use of this tool, it is possible to properly configure the robot, as long as the URDF file is provided.

4.4 RViz

This section will start by describing RViz, and how it was used with MoveIt!

4.4.1 RViz concept

RViz is a tool for visualisation of sensor data that can be extended with plugins. With this tool it is possible to visualise the robot and its surrounding environment and can also be adapted with different configuration files.

RViz comes with a Graphical User Interface (GUI) that provides more than just pure data visualisation. This offers ways for creating maps, defining navigation points and the capability for monitoring the navigation, also enabling the user to test objects for objects recognition and to monitor and define commands for human-robot interaction. It is also possible to have basic state behaviour and task execution that can be defined in the GUI.

4.4.2 RViz motion planing using MoveIt!

MoveIt! provides a plugin for RViz that allows to create new planning scenes in the robot workspace, create motion generation plans, add new objects and visualize the planning result. It also allows direct interaction with the visualized robot.

4.4.3 RViz motion planning plugin

The plugin provided by MoveIt! which works with RViz, Figure 4.5 and 4.6. In which they provide several tabs like, Planning, Manipulation, among others. In the Context tab it is possible to change the Planning Library that is currently in use. The Planning tab is used to define the starting and goal state, plan the trajectory and execute it, as it is shown in Figure 4.6. Under the *Query* panel it is located the *Start State and Goal state* of the robot, and with *Plan* button located under *Commands*. By which the planned trajectory from the Start state to the Goal state and its executions can be foreseen.

It is also possible to set the goal position of the end effector of the HARM, with the use of an interactive marker attached on the gripper. In Figure 4.7 it is possible to see a target location for the HARM represented in orange, the marker in blue and in grey is the current state of the HARM.

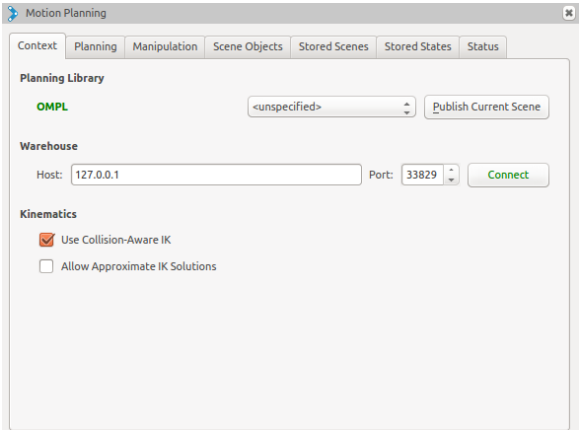


Figure 4.5: MoveIt! plugin for RViz.

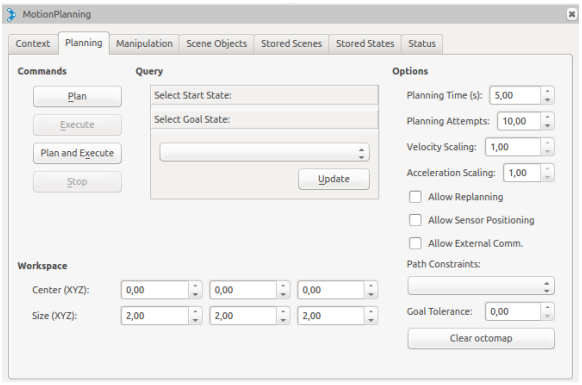


Figure 4.6: MoveIt! plugin for RViz with Planning tab open.

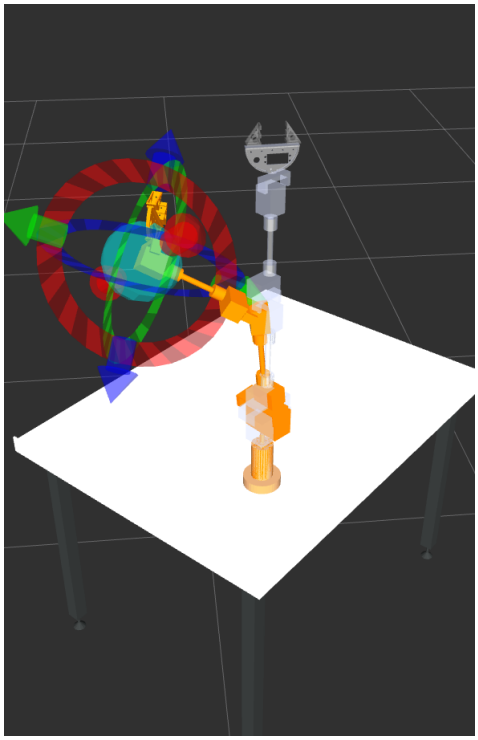


Figure 4.7: Image taken from RViz. In grey it is represented the state read from the physical module and in orange the goal position.

4.4.4 Software view

This section will be dedicated to an overview of the software implementation. It will start with an overview of the software that is implemented for the HARM with MoveIt! and then how to test each joint individually.

Figure 4.8 represents the HARM flow diagram. Once the process starts the HARM will move automatically to its starting state (where all joints read 0 degrees) and maintain that position, then it will check if a new position is provided. This information can be provided with simple communication with MoveIt! where it is possible to specify the cartesian coordinates and the yaw, pitch, and roll rotations for the end effector, or as previously mentioned, or the plugin for RViz could be used to set the target location. Then the process will verify if the position is valid. If the position is valid it will start executing the planned trajectory, if the provided coordinates are no valid it will report an error while maintaining the current position.

The HARM had to be tested to ensure the movements of each joint individually. Even though the project at the moment has a fixed velocity, when in debug mod, for every movement it is recommended to perform a few tests first, in order to see the motors behaviour. Figure 4.9 represents the joint state publisher used to test each joint. This tool refers to each HARM joint as cambada_joint and the number correspond to the actual joint number of the HARM. The interface lets the user to move the joints freely from -90 to 90 degrees. It is also possible to press the *Randomize* button to go to a random position, or just place the HARM in the original position with the *Center* button and it will go to the center/origin state. AS a side note when using this tool, the movement is automatically executed.

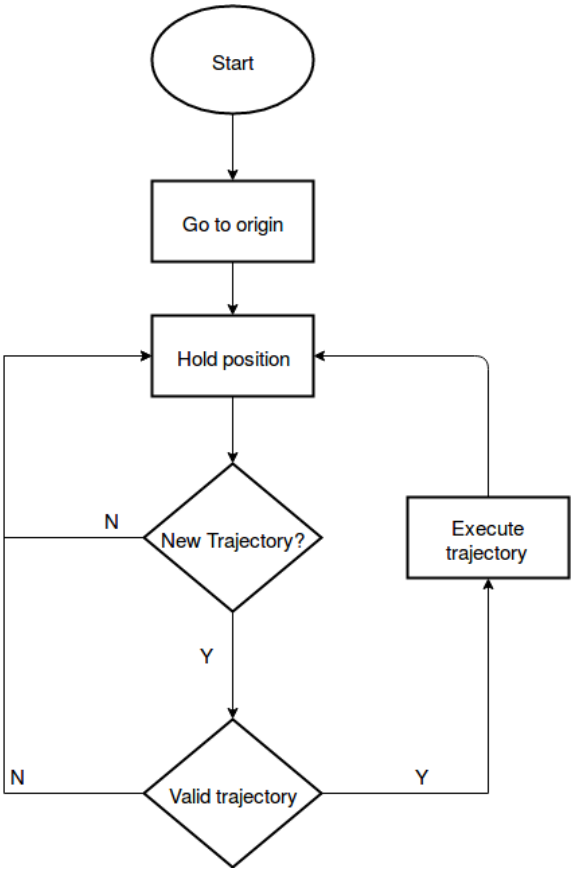


Figure 4.8: HARM's flow diagram.

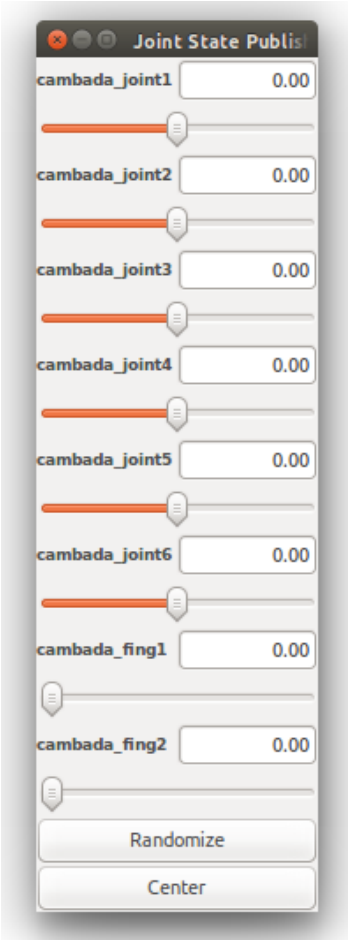


Figure 4.9: HARM joint state publisher.

4.5 Simulation

The HARM structure underwent through a necessary simulation environment.

The simulation is a technique used to test the dynamics of a system. The simulation can have a wide range of objectives, in which one of those goals can be to test the behaviour of the model within the simulation that will be as similar as possible to the performance of the physical model.

Simulators help to develop virtual models of physical systems and its surrounding environment, which allows to perform virtual actions that correspond to the real ones. This allows to test ideas, and test the software without harming the hardware, causing major cost in case of accident.

Since the physical model has already been assembled it was possible to compare both performances, the simulation and the physical model and verify if the virtual model actions, correspond to the real ones and if there is a discrepancy in the results. It is possible to tune the parameters of the virtual model to make it more accurate. Even though these parameters can be tuned by performing various tests, it is not possible to predict the material wear, battery level, etc, meaning that the parameters might have to change over time.

In this dissertation the choice for the simulator was based on some previous work that has been done for the CAMBADA@Home platform, and since the software has been developed for ROS, the simulator chosen was Gazebo.

4.5.1 Gazebo

Gazebo is a multi robot 3D open source simulator developed in C++. The Open Source Robotics Foundation (OSRF) is the current responsible for keeping the development of Gazebo. That lets us model the systems with multiple sensors, actuators and objects.

Gazebo provides an API inside the simulation, that allows to access the actuators and data about the world through the libraries provided by the community. Since all objects have physics associated to them apart from the light, it is possible to obtain a real behaviour from the system.

Gazebo provides some simulation models of popular robots, sensors, and a variety of 3D objects, making it easier to use these models directly without having to create them.

Gazebo interface is connected with ROS, in order to control the robot, which allows the controller developed in ROS to work exactly in the same way with the simulation running in Gazebo and in the actual system.

4.5.2 HARM simulation with Gazebo

When using Gazebo there is a need to use the exact same topics and nodes that the physical model uses. This made it possible to have the same controller that was developed for HARM, work with both models, physical and virtual, in the exact same way.

In the representation on Figure 4.10 it is possible to verify that the simulation also uses the same topics as our physical arm, guaranteeing similar information and behaviour results. The planner used would not distinguish a simulation from a real environment.

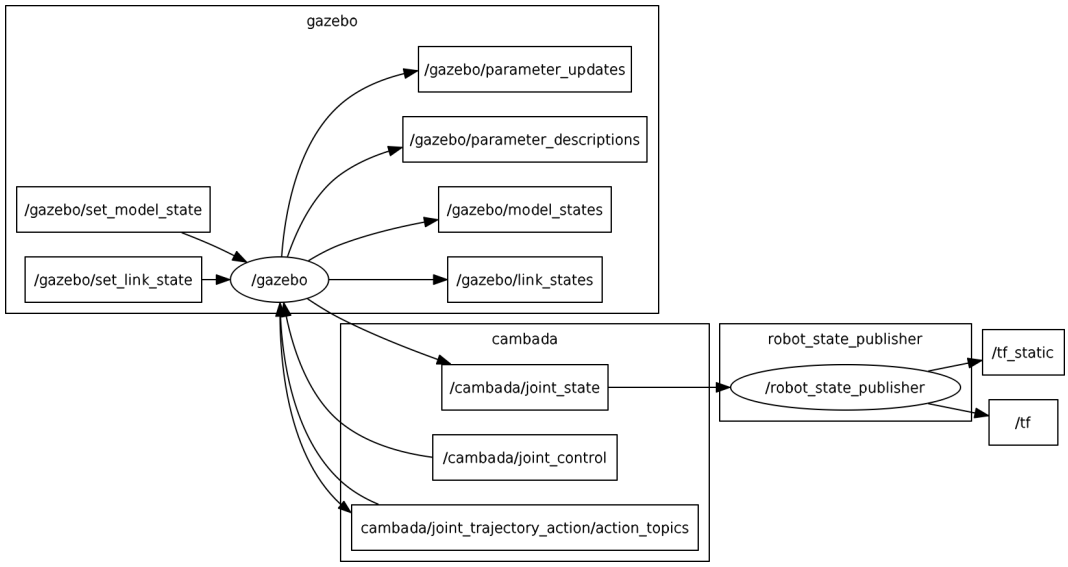


Figure 4.10: Published nodes and topics created by gazebo in order to replicate the real arm topics.

Even though the simulation (running in Gazebo) has created more nodes then HARM, the main topics associated to it are the same as well as the communication between them, which makes it possible to run the same program with both. The extra topics that are present in the gazebo namespace are the resources that it needs to properly simulate the robot and its surrounding environment.

It is to note that the current simulation still has no noise associated to it, and that the values for the friction coefficients and components mass still need to be correctly tuned, meaning that its movements will be exactly as the planner provided. This should not invalidate the simulation, since it still represents the behaviour that the physical model would take, making it still valid to use the simulation first to test new algorithms, as well as to introduce it to new users, etc and then proceeding to the physical one.

4.6 Summary

This chapter focus on understanding the framework used to control the HARM. Starting with the introduction of the framework, how it works, its computation graph level, URDF and what it does, as well as the coordinate frames and transformations, it also explains how the HARM interacts with ROS and the messages required to establish a proper communication. The needs to have a planner, for that the MoveIt! was used, by command or even with the interaction with RViz.

Then a presentation of the software implemented within ROS will be explained. The chapter ends with the simulation environment.

Chapter 5

Results

To properly test the performance of the robotic arm, it was taken into consideration all the worst case scenarios. The first couple of tests were performed without any load, to see how the robotic arm would handle with its own weight. The first test consists on only using joint 2 and with the arm fully extended, since it would create a downward movement and a upward movement of the robotic arm. These tests can be seen in Figure 5.1 and 5.2 respectively.

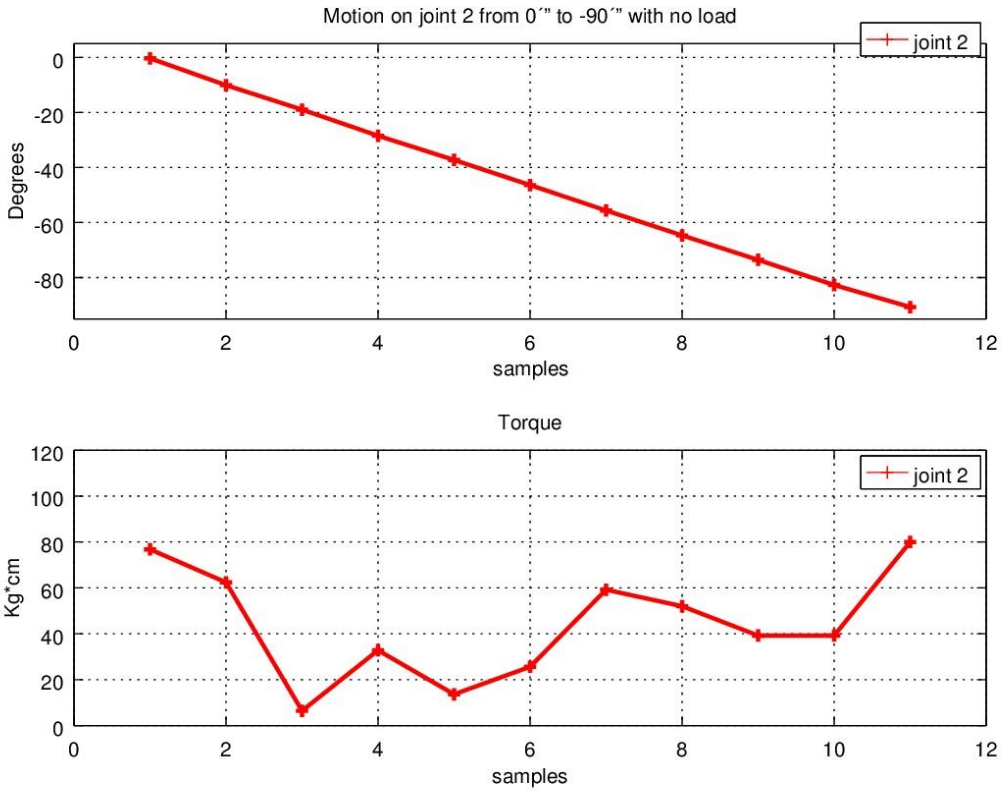


Figure 5.1: Joint 2 moving from 0 degrees to -90 degrees, creating a downward movement on the robotic arm ,with the respective force being applied.

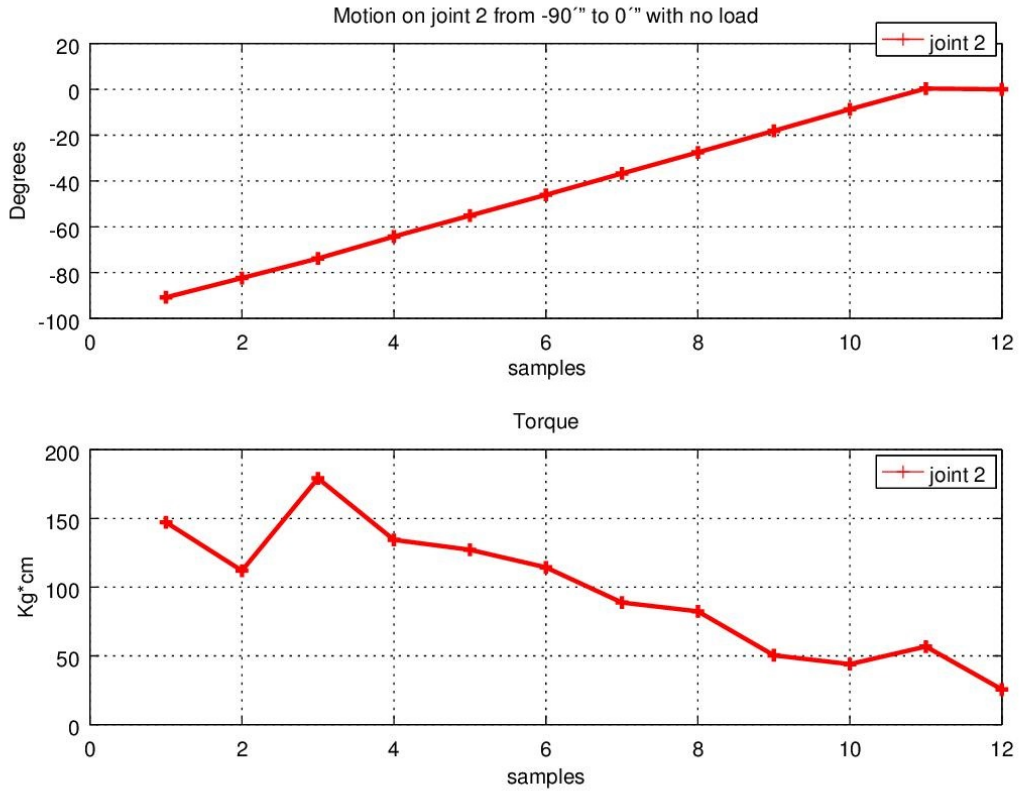


Figure 5.2: Joint 2 moving from -90 to 0 degrees, creating an upward movement of the robotic arm, with the respective force being applied.

In the test shown in Figure 5.1, it is possible to observe that the force that the motor is applying floats during its movement. The starting force can be explained due the starting movement, then the force required to be made starts to decrease. Then it just needs to adjust the force applied, to guarantee that it flows at a constant velocity. Once the robotic arm reaches the -45º mark (at sample 7), the motor increases the force applied, due to the gravity starting to have more influence on the arm.

Even though the movement are linear on both tests, the second test where the arm is moving upwards uses more force at the beginning of the movement. Such is expected since it needs to counter the force of gravity.

Since joint 3 and 4 shared the same type of motor and the distance from their axis to the end effector are similar, only the test performed with joint 4 will be shown. The test can be seen in Figure 5.3, where the motor is performing a rotation from -90 to 90 degrees and it is also possible to see the force that it applies at each moment. In this test the position of joint 2 was left at 0 degrees.

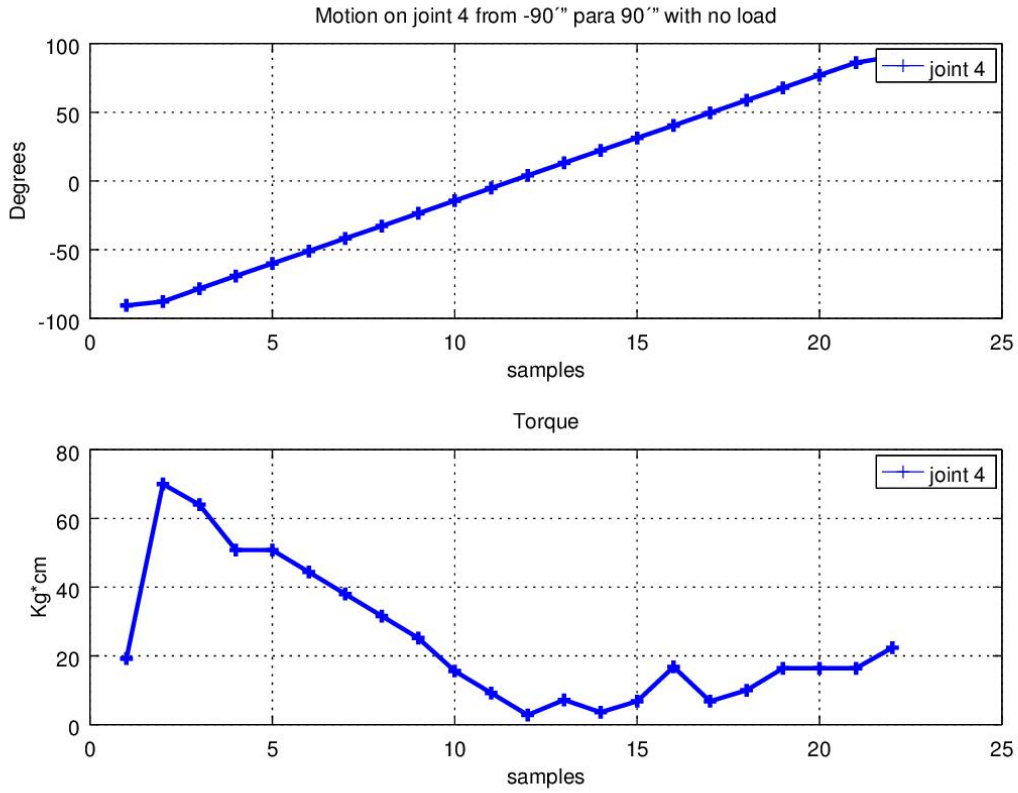


Figure 5.3: joint 4 moving from -90 degrees to 90 degrees with the respective force being applied.

In this movement it is possible to see some acceleration at the beginning of the movement, as the force required increases. Then when the velocity of the movement is constant, the force required starts to decrease. This peak in the force can be due to the static friction being higher than the dynamic friction, another reason could be that the movement started against the force of gravity.

Once the movements of the robotic arm were stable and fluid, a higher load of 263 grams was put to the test. Since the movements of the robotic arm were stable and fluid, with no load attached to the end effector, it was performed the same tests as before, this time it will used a load of 263 grams at the end effector.

In Figure 5.4 and 5.5 its represented the movement of the arm when its fully extended and as the first test. Only joint 2 will be operating all the other motors will be fixed at 0 degrees. Although joint 4 should be fixed in this test, it was also taken into consideration its performance. Since it might be affected, due to the starting movement of joint 2 and the counterweight caused by the load.

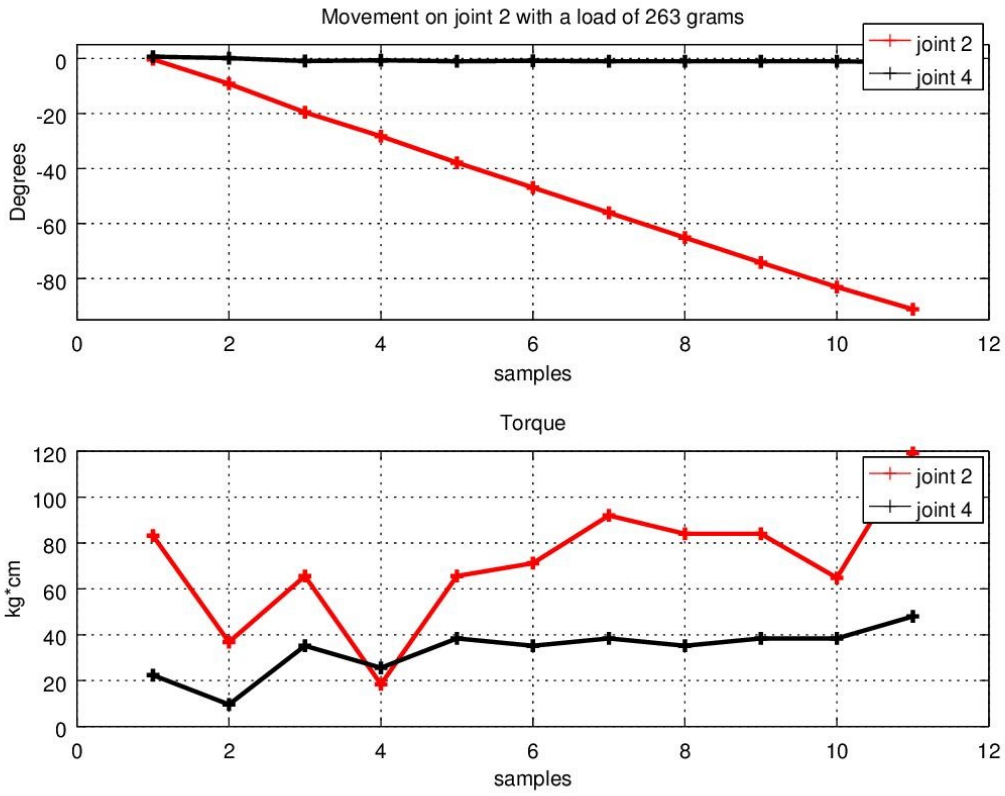


Figure 5.4: Arm joint 2 moving from 0 degrees to -90 degrees while holding a load.

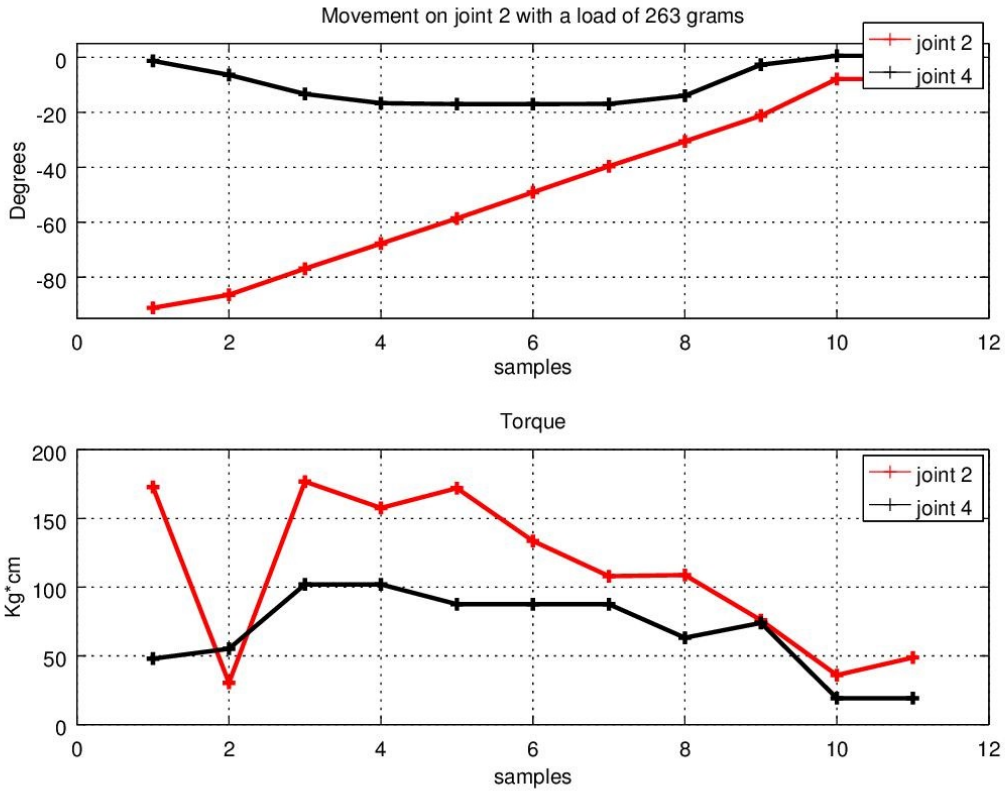


Figure 5.5: Arm joint 2 moving from -90 degrees to 0 degrees while holding a load.

In analyses of the previous Figures it is possible to verify that joint 2 acts as expected and can deal with the load, and once again it is possible to verify that the force required when performing an upward movement, Figure 5.5, is almost the double as the downward movement. It also possesses an error of 8.77% since it should stop at set point given, 0 degrees, and instead it finished the movement at -7.9 degrees. It should be possible to reduce the error with a better tuning of the internal PID of the motor.

As it was predicted the motor at joint 4 was affected, as it is shown in Figure 5.5. Once the movement of the arm started, the motor did not had the required force to maintain its position and had a maximum deviation of -17 degrees, at the same time it used its maximum force to guarantee that it would not fall over that value. This effect could be reduced, by ensuring that the movement of joint 4 is not in favour of the one being made at joint 2.

Since joint 4 was tested previously with no load and it has shown to be problematic in the previous test, it was performed a test considering the worst case scenario, but with joint 2 fixed at 0 degrees, to ensure that it would not affect the performance of joint 4. Figure 5.6 shows the graphics of the movement performed, from -90 to 90 degrees, as well as the force it needs to perform the movement while holding the load.

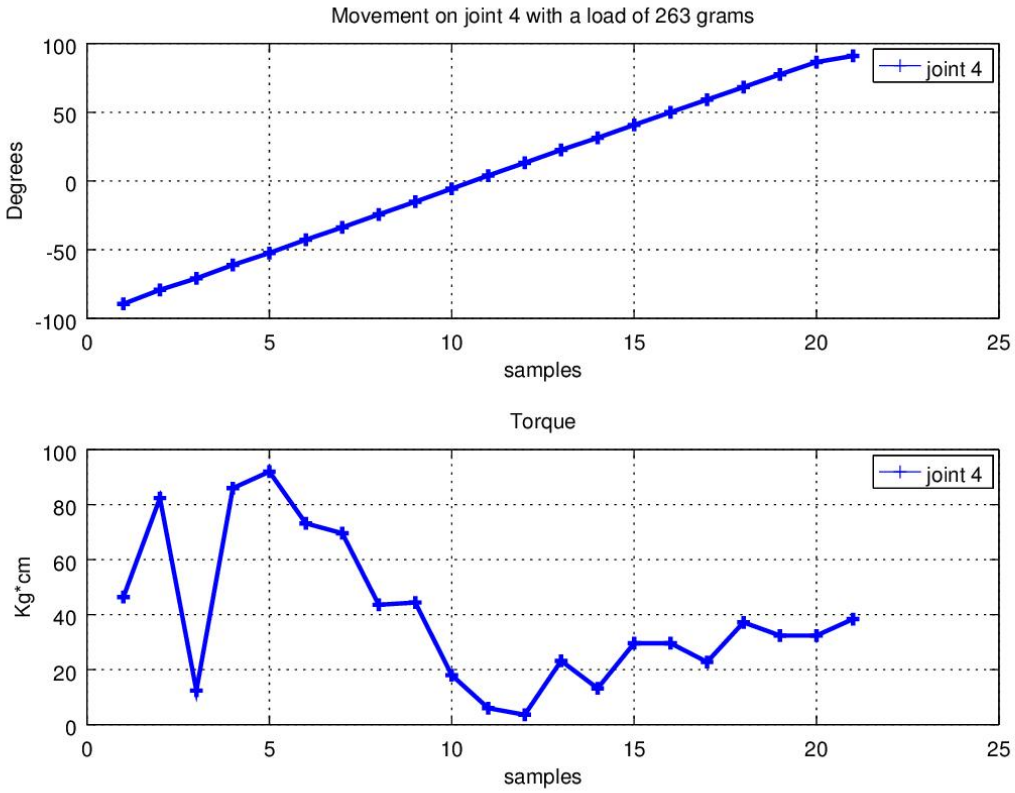


Figure 5.6: Joint 4 moving from -90 degrees to 90 with a load of 236 grams.

Here it is possible to verify that the behaviour presented in joint 4 is linear and that it performs most of the force when lifting the object. Once the joint starts approaching the 0 degrees mark, the force required is greatly reduced. After the 0 degrees mark, the force that it needs to apply is reduced, since it only needs to counter the gravity to ensure that it moves with the correct velocity.

With the basic tests performed and with some notions of the HARM capabilities, it was time to test it with the planner. The planner provides different velocities for each joint, as well as the trajectory that the arm should take. In Figure 5.7 it is shown a planned trajectory for each joint and the result of their physical movement.

In the previous test the movements performed were close to the ones that were planned, so in order to ensure repeatability, multiple test were performed. Each with a different trajectory to ensure that each joint would move more then just a few degrees as it is shown in figure 5.8.

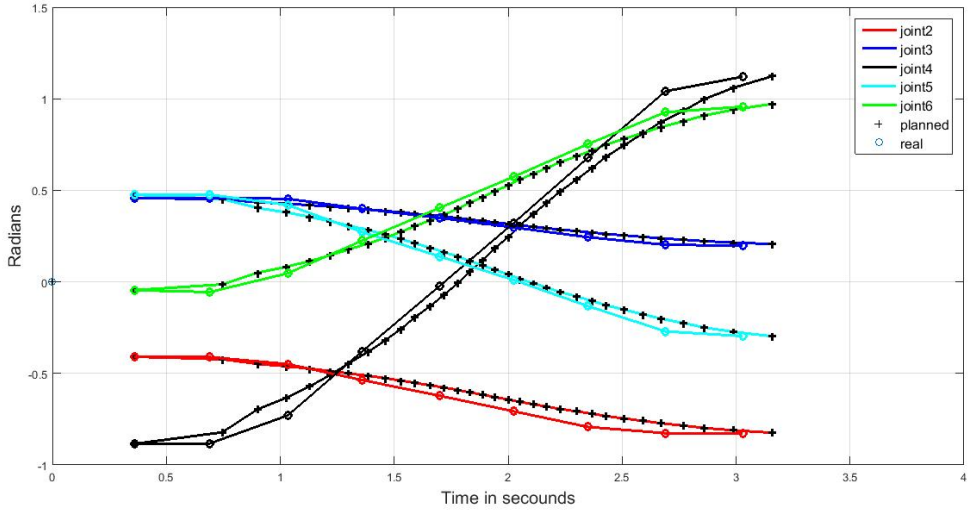


Figure 5.7: Movement provided by the planner and the movement performed by HARM.

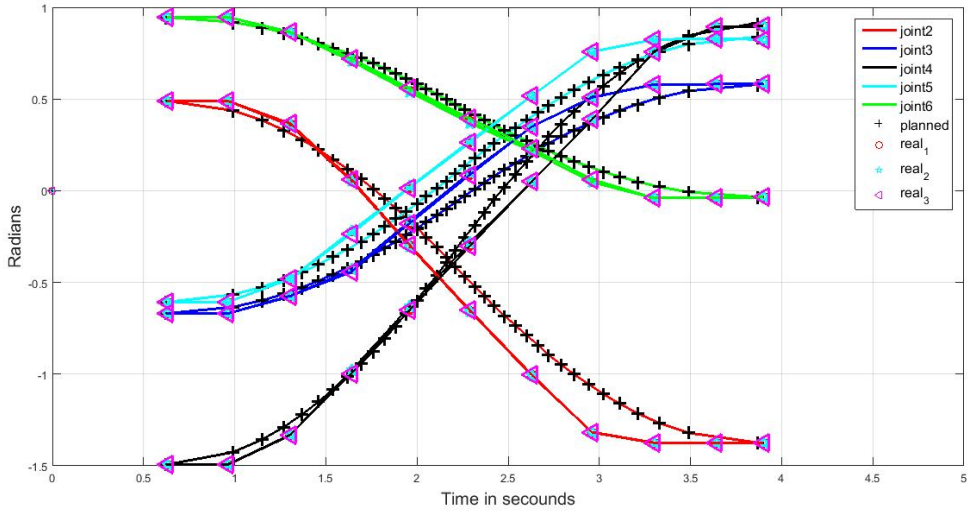


Figure 5.8: Movement provided by the planner and the movement performed by HARM, with repetition to guarantee repeatability.

In this final test it is possible to verify a gap between the planner and the real trajectory, this could be related to the mass of the robotic arm, kinetic friction, among others physical details that were not focused in this work. It would be also possible to adjust the velocity provided by the planner which is applied to the motors.

Chapter 6

Conclusions

The first step taken towards this work was the familiarisation with the Dynamixel servos and their communication protocol, as well as the Maxon EC-90(BLDC motor). It was also required to develop an interface in order to control the said motors, which led to an extensive study of the components, (microcontrolers, integrated circuits) needed to develop the hardware and the circuit board.

Afterwards, in order to make a 3D model and the eventual manufacturing of some needed mechanical parts, the research of the mechanical joints and the nature of the arm was performed. Finally, an integration of this system with ROS framework was performed in order to merge it with the current existence of the CAMBADA@Home model.

Even though one of the objectives was being able to have stable and smooth motion with a payload of 500 grams with the current configuration, this requirements could not be fulfilled. The movement at joint 3 and 4, could be improved by using two motors in a master slave configuration. This would rise the mass of the arm and it would deteriorate the movement at joint 2 and in order to improve it, there would be a need to select a different motor. It is also possible to improve the arm behaviour with the maximum payload by reducing the length of each link.

The Dynamixel servo motors have proven to be very versatile in terms of operation and communication. The MX-series allows us to adjust the controller with the PID and the RX and AX series can be tuned by adjusting the slop in order to improve the precision that can be adjusted in order to improve their precision.

The 3D model that has been developed in this dissertation, using the CAD software, SolidWorks, made a decisive contribution by allowing to perform an analysis of the mechanical structure, as well as understanding the safe operating range of some motors. With the model, it was also possible to dimension the links for the HARM, choose the motors depending on their location and the design of some mechanical components to be assembled with the arm. This model was also used to export its properties, into an URDF file to work with ROS.

ROS has proven to be useful for the development of the project, even considering the initial difficulties that it brought. With the use of ROS it was possible to integrate the planner, MoveIt! and the visualiser RViz, which provided a way to observe the trajectory provided by the planner, before giving the command to execute the movement.

With the use of ROS it was also possible to take advantage off the simulation, with the

use of Gazebo, to represent a virtual structure of HARM and test the algorithms in order to see if they would work on the physical model.

In this work there is still room for improvement that should be done in the future, and even though this project was aimed to be applied on a moving platform, CAMBADA@Home it might have other applications.

6.1 Future work and possible applications

Some of the ways to improve this project would be to perform a fine tuning of the PID controller of the MX series from Dynamixel servor motors and the slop control of the RX series. Once the control board, that was developed during this project, is available it should be possible to test the BLDC controller. Then test if it is possible tp install it with the rest of the HARM and associate with the controller, that has been already develop with ROS. Once this is done all the board should be tested and see if all the communications are working properly and then implement this project with the rest of the CAMBADA@Home platform, since it is one of the objectives for this work.

Since the mechanical structures which have been developed in this dissertation were only prototypes, they were made with a 3D printer. There for it would be recommended the usage of a stronger material.

As mentioned, the URDF file contains the physical properties of the model of the robotic arm, which is important for the simulation since it contains the basic parameters like mass, friction, etc, which can be and should be calibrated.

Some of the future applications, regarding the one that it was made for, can be place on a wheel chair, and be control by a joystick or even use the robotic arm as it is at its present state.

References

[1] Luís Filipe PereiraAlmeida Santos. Project of an anthropomorphic manipulator for the cambada@home robot. Master's thesis, Universidade de Aveiro, 2013.

[2] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009.

[3] RoboCup. Available: `http://www.robocupathome.org/`. Last access 2017-06-16.

[4] ST Robotics. Available: `https://www.prnewswire.com/news-releases/st-robotics-offers-new-super-fast-robot-arm-300412383.html`, Last seen 2017-07-14.

[5] yamaha. Available: `https://global.yamaha-motor.com/business/robot/lineup/ykxg/middle/img/index/img_l.jpg`, Last seen 2017-07-14.

[6] Global Robots LTD. Available: `http://www.globalrobots.com/product.aspx?product=24914`, Last seen 2017-07-14.

[7] Machine Design. Available: `http://www.machinedesign.com/industrial-automation/reshoring-boost-american-manufacturing`, Last seen 2017-07-14.

[8] Festo. Available: `https://www.festo.com/group/en/cms/index.htm`. Last access 2017-07-14, 2017.

[9] American company Intuitive Surgical. Da vinci surgical system. Available: `http://www.davincisurgery.com/safety/`. Last access: 2017-07-14, 2017.

[10] Intuitive surgical. Da vinci surgical system. Available: `https://www.intuitivesurgical.com/products/davinci_surgical_system/`. Last access: 2017-07-14, 2017.

[11] Dirk Spenneberg, Jan Albiez, Frank Kirchner, Jochen Kerdels, and Sascha Fechner. C-manipulator: An autonomous dual manipulator project for underwater inspection and maintenance. In *Proceedings of OMAE*, 2007.

[12] Intl. Federation of Robotics (IFR). Service robots - definition and classification. Available: `https://ifr.org/service-robots/`. Last access 2017-07-14, 2016.

[13] Kinova Robotics. Kinova jaco^2. Available: `http://www.kinovarobotics.com/`. Last access 2017-07-14, 2017.

[14] PAL robotics. Available: http://tiago.pal-robotics.com/. Last seen 2017-07-14.

[15] Willow Garage. Available at: http://www.willowgarage.com/pages/pr2/overview. Last access 2017-07-17.

[16] Robotic Open Platform. Available: http://roboticopenplatform.org/wiki/AMIGO. Last access 2017-07-14, 2017.

[17] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.

[18] Katherine Ellis, Suneeta Godbole, Simon Marshall, Gert Lanckriet, John Staudenmayer, and Jacqueline Kerr. Identifying active travel behaviors in challenging environments using gps, accelerometers, and machine learning algorithms. 2:36, 04 2014.

[19] ROBOTIS. Available: http://en.robotis.com/index/index.php Last seen 2017-07-17, 2017.

[20] Industrial Robot/Research Development Open Source Ecology. Available: http://opensourceecology.org/wiki/Industrial_Robot/Research_Development. Last seen 2017-06-15.

[21] Maxon Motors. BLDC motor especifications available at: http://pdf.directindustry.com/pdf/maxon-motor/program-2014-15/7173-600371.html, Last seen 2017-07-14.

[22] Maxon Motors. Planetary Gearhead GP 52 specifications available at: https://www.maxonmotor.com/medias/sys_master/root/8825548144670/17-EN-350-351.pdf, Last seen 2017-07-14.

[23] Maxon Motors. Image of a Planetary Gearhead https://www.maxonmotor.com/maxon/view/product/gear/planetary/gp52/223080, Last seen 2017-07-14.

[24] Avago. Available: https://www.broadcom.com/site-search?q=aeat-6012, Last seen 2017-07-14.

[25] Fairchild FCM8201 datasheet. Available: https://media.digikey.com/pdf/Data%20Sheets/Fairchild%20PDFs/FCM8201_Rev_1.0.4.pdf, Last seen 2017-07-14.

[26] STMicroelectronics. data sheet available at: http://www.st.com/content/ccc/resource/technical/document/datasheet/d7/80/b5/a2/a2/93/49/59/CD00000970.pdf/files/CD00000970.pdf/jcr:content/translations/en.CD00000970.pdf. Last access 2017-06-16.

[27] Microchip. datasheet available: http://ww1.microchip.com/downloads/en/DeviceDoc/22049f.pdf, Last seen 2017-07-14.

[28] Microchip Technology Inc. Available: http://www.microchip.com/wwwproducts/en/pic32mx795f512h. Last access 2017-07-14, 2017.

[29] Maxim Integrated Products Inc. 3.3v, high-speed, rs-485/rs-422 transceiver in sot. *Package, March 2007. 3 rd revision.*, 2007.

[30] MC74LCX125 Datasheet ON Semiconductor. Available: https://www.onsemi.com/pub/Collateral/MC74LCX125-D.PDF, Last seen 2017-07-14.

[31] Maxim Integrated Products Inc. +3.3v, 1mbps, low-supply-current can transceiver. *datasheet available at :https://datasheets.maximintegrated.com/en/ds/MAX3051.pdf*, 2007.

[32] Maxon Motors. Available: http://www.maxonmotor.pt/maxon/view/product/control/Positionierung/EPOS-4/504383. Last access 2017-07-14.

[33] Allegro A4910 datasheet. Available: https://www.digikey.pt/product-detail/en/allegro-microsystems-llc/A4910KJPTR-T/620-1534-1-ND/4448874. Last access 2017-07-14.

[34] Allegro A3930-31 datasheet. Available: https://www.digikey.pt/product-detail/en/allegro-microsystems-llc/A3930KJPTR-T/620-1289-2-ND/1972824. Last access 2017-07-14.

[35] Ioan A Sucan and Sachin Chitta. Moveit! *Online at http://moveit.ros.org*, 2013.

[36] Lentin Joseph. *Mastering ROS for robotics programming*. Packt Publishing Ltd, 2015.

[37] MoveIt! Image Available at: http://moveit.ros.org/documentation/concepts/, Last seen 2017-07-14.

Chapter 7

Anex

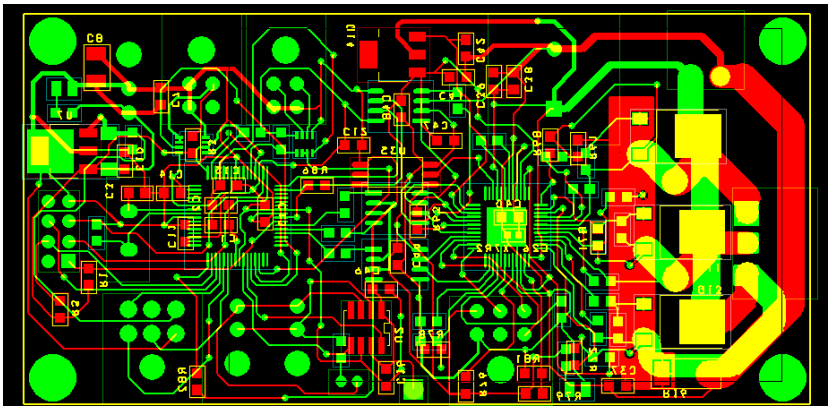


Figure 7.1: View of the PCB design in green the top layer and in red the bottom layer.

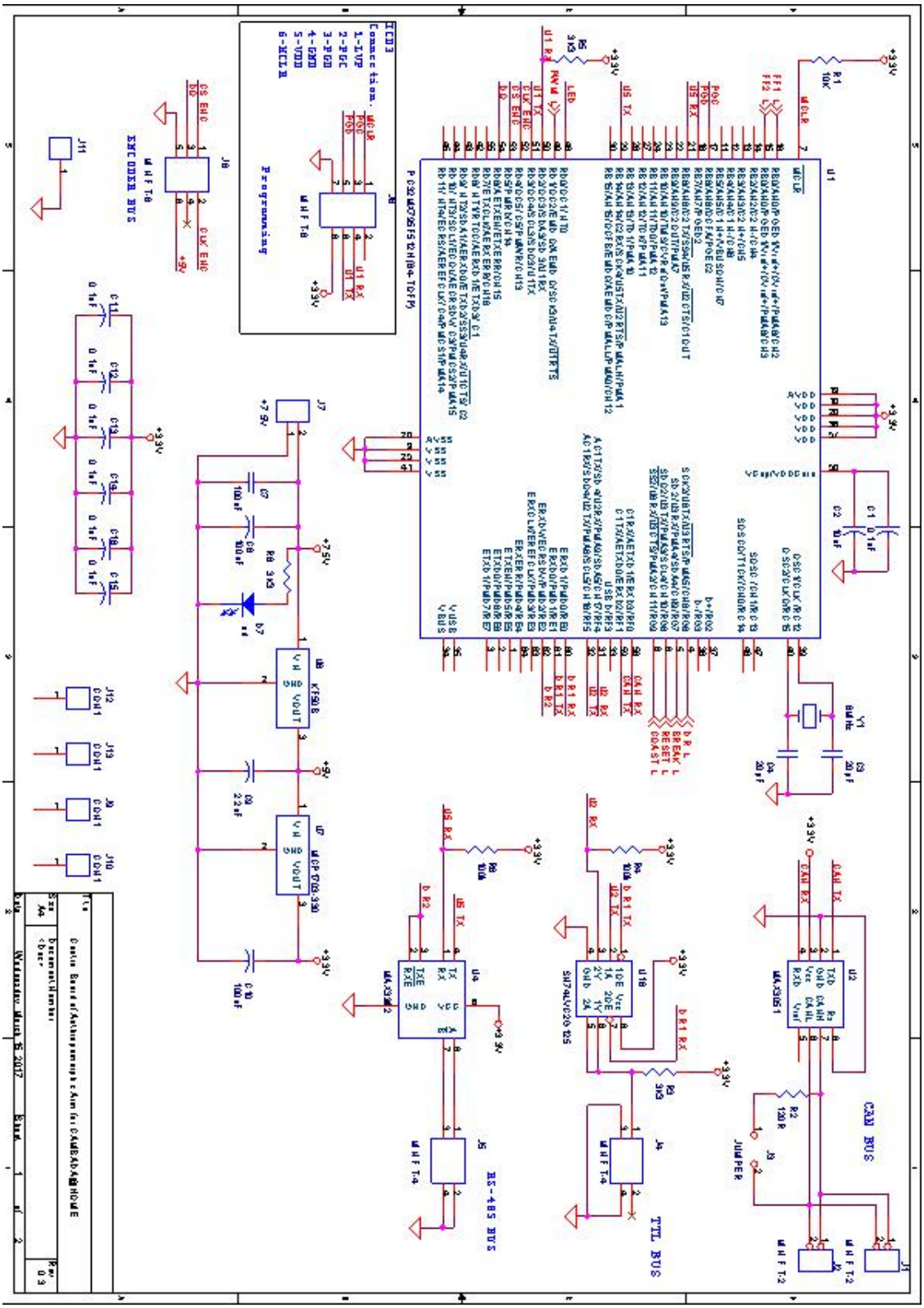


Figure 7.2: schematic for the digital and communication part.

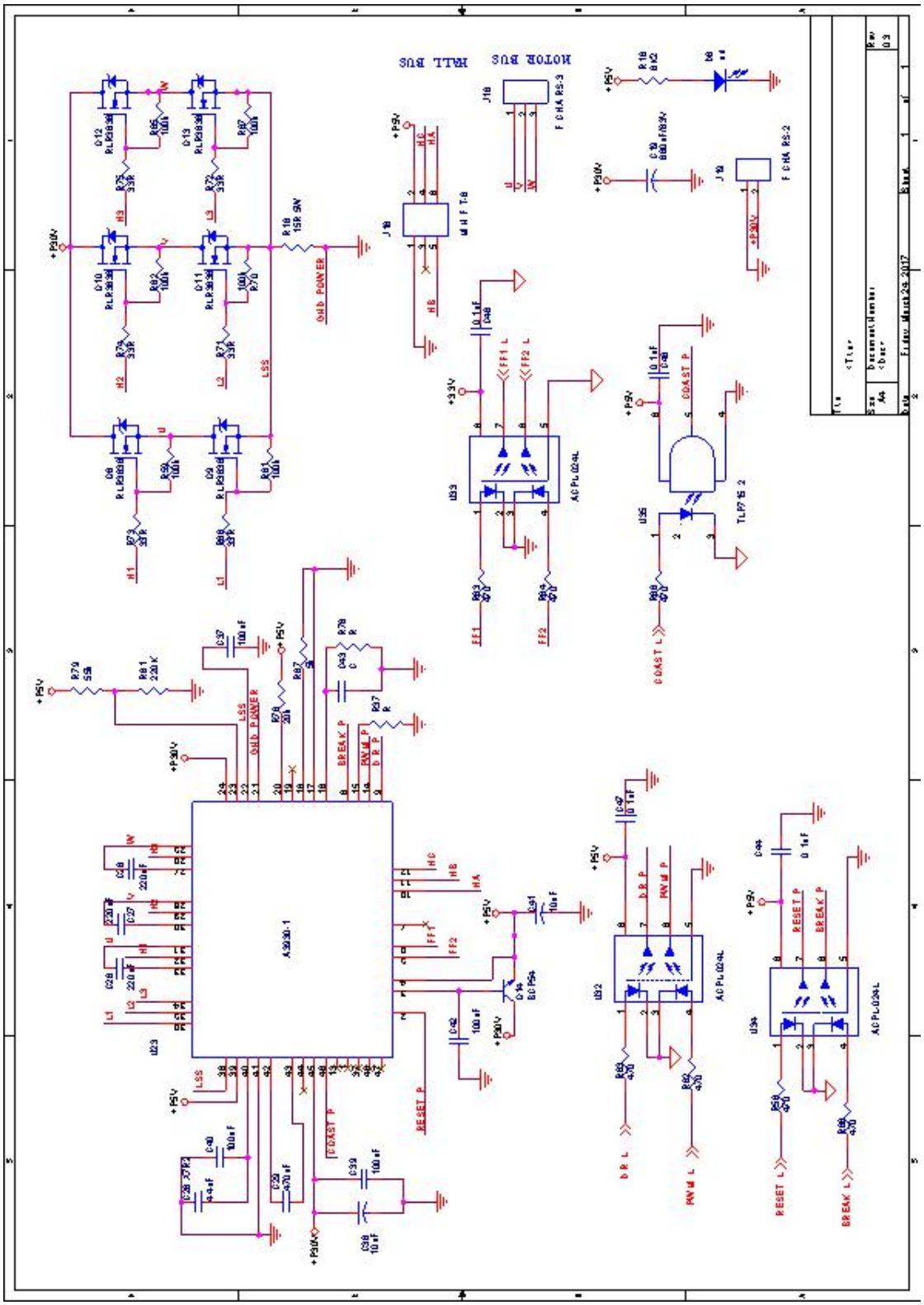


Figure 7.3: schematic for the power controller.

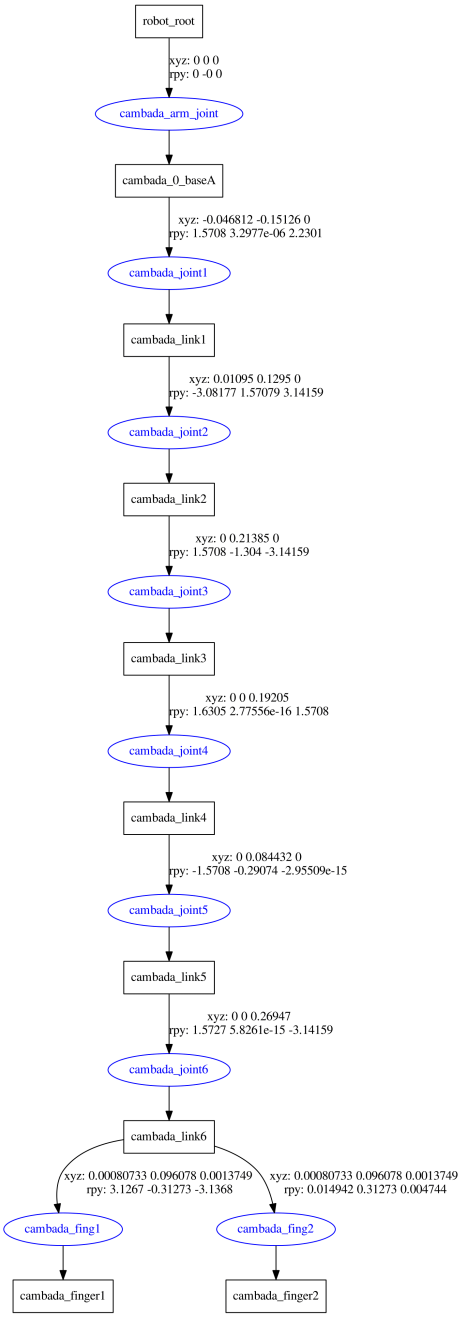


Figure 7.4: Representation of the model from the URDF file with the use of the tool urdf_to_graphiz.

Side notes

Here it will be introduced and how to operate HARM, commands that one needs to use to start controlling it.

The first thing to do before connecting the arm, it to make sure that no joint is facing the -90º mark (downwards), to ensure that once it starts it wont miss behave or even fall due to the motors not being in a lock position. Once the arm is connected, it is possible to start the communication. In a ROS environment run the command *rosrun cambadacontrol_test cambadacontrol_test_node*, this will connect the hardware with ROS. To operate the arm freely it is possible to run the command *roslaunch cambada_test aver.launch* this will bring the GUI with all the joint possitions, like its represented in FIGURE 4.9 in page 44.

In order to connect the planner with the system one should run the command *roslaunch cambada_test_moveit cambada_test_moveit.launch* in a ROS environment, make sure that the GUI for the free joint control is off otherwise once the movement provided by the planner finish executing it will go back to the position that the GUI has. In order to run RViz run the command *roslaunch cambada_test_moveit cambada_test_rviz.launch* and it will bring the RViz window up letting the user see the planned trajectory and decide to execute it or not.

If the user does not want to run the physical HARM it can run the simulation for that instead of using the *rosrun cambadacontrol_test cambadacontrol_test_node*, just run the command *roslaunch cambada_test cambada_test_gazebo_controlled.launch* and it will bring up the gazebo with HARM on it that performs the same way that the physical model does, it is also possible to run this command and bring the state publisher right away by adding load_joint_state_publisher:=true in front of the last command like so *roslaunch cambada_test cambada_test_gazebo_controlled.launch* load_joint_state_publisher:=true