Text

Description automatically generated

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE ROBÓTICA SOFTWARE

**TRABAJO FIN DE GRADO**

**CONSTRUCTION AND DEPLOYMENT OF A SIMPLIFIED REACHY ROBOT**

Autor: Alberto Delgado del Cerro

Tutor: José Miguel Guerrero Hernández

Academic course 2021/2022

Index

[Index 2](#_Toc113876276)

[Summary 4](#_Toc113876277)

[Introduction 5](#_Toc113876278)

[Robotics: General Introduction to Robotics 5](#_Toc113876279)

[Social robotics: What it is and what it is used for 6](#_Toc113876280)

[Social robots 6](#_Toc113876281)

[Pepper 6](#_Toc113876282)

[Nao 7](#_Toc113876283)

[Voice assistants 8](#_Toc113876284)

[Work structure 9](#_Toc113876285)

[Objectives and methodology 10](#_Toc113876286)

[Objectives 10](#_Toc113876287)

[Methodology 11](#_Toc113876288)

[Work plan 12](#_Toc113876289)

[Choice of software 12](#_Toc113876290)

[Machine vision. 12](#_Toc113876291)

[Communication 13](#_Toc113876292)

[Movement 13](#_Toc113876293)

[3D Printing 13](#_Toc113876294)

[Coordination of behaviors 14](#_Toc113876295)

[Description of the work 15](#_Toc113876296)

[Operating system 15](#_Toc113876297)

[ROS 15](#_Toc113876298)

[Dialogue 17](#_Toc113876299)

[Artificial vision 18](#_Toc113876300)

[Follow cubes 19](#_Toc113876301)

[Movement 21](#_Toc113876302)

[Neck movement 22](#_Toc113876303)

[Antenna movement 23](#_Toc113876304)

[3D Printing 24](#_Toc113876305)

[Antenna support 24](#_Toc113876306)

[Face 25](#_Toc113876307)

[Neck 27](#_Toc113876308)

[Reachy Implementation 30](#_Toc113876309)

[User Manual 31](#_Toc113876310)

[Conclusions and future work 32](#_Toc113876311)

[Bibliography 34](#_Toc113876312)

# Summary

Since the first appearance of the term "robot" in *Rossum's Universal Robots,* humans have dream of interactions between them and machines. Throughout history, a multitude of humanoid robots have been created, allowing comfortable interaction with people, from Joseph Barnett's precarious Elektro to the innovative Sophia, created by the company Hanson Robotics.

This[[1]](#footnote-1) project aims to create a simplified version of the Reachy robot, a humanoid open source robot created by the company Pollen Robotics precisely to interact with people. This version will use ROS (*Robot Operating System*) to control the functions and a Raspberry Pi as a processor, so its cost will be much lower than that of the original project.

This project is based on three main (or essential) parts: movement, vision and human machine interaction, which will be developed separately to and the joined, obtaining a robot that can perform the three tasks in a coordinated way.

As a result of this work, a functional robot manufactured from scratch has been obtained, designing and 3D printing the appropriate parts, and programming the behaviors and actions necessary to interact with a person in different ways.

In conclusion, despite the problems caused by the incompatibility of the hardware, a robot has been created with a high autonomy, modular and that can be easily modified, either to add or eliminate functionalities.

# Introduction

## Robotics: General Introduction to Robotics

The Oxford Dictionary defines robotics as the technique used in the design and construction of robots and apparatus that perform operations or jobs, usually in industrial facilities and replacing human labor.

Despite this, the term robot has evolved since its creation, going from being used in industrial facilities to encompassing everything from communication assistants to humanoid machines capable of moving and interacting with their environment better than most people.

[](https://www.youtube.com/embed/tF4DML7FIWk?feature=oembed)

1 Video demonstration of movement of the Atlas robot

At present, we are living through the rise of domestic robots. In contrast to industrial ones, these robots are lighter, smaller, and less powerful, allowing them to share space with people without posing a danger to them. Within these domestic robots we find those that help in domestic tasks such as automatic vacuum cleaners, which use a map of the house to clean as efficiently as possible, or robots that dose pet food and entertain them to make their day to day more bearable. Also included in domestic robots are those that communicate directly with people, such as voice assistants or educational robots. The latter, which exist to interact with humans in the most natural way possible, are known as social robots.

## Social robotics: What it is and what it is used for

Social robotic can be defined as the science that uses an autonomous robot that interacts and communicates with humans or other autonomous physical agents following social behaviors and rules associated with its function.

Currently these robots are increasingly common, they are used in educational fields such as Nao or Pepper robots from the company SoftBank Robotics or pets such as the Paro seal, but they all have one problem in common: their prices. The cheapest of these three robots’ costs about $6,000, and the most expensive about $20,000, even Reachy's project, which is open source and allows you to print your own parts, has an estimated budget of about $2,000 counting only the materials.

## Social robots

### Pepper

Pepper is a humanoid robot that measures about 1 meter and 20 centimeters in height. Due to its attractive and dynamic design, it has been used in different establishments facing the public such as hotels or shopping centers and is able to interact with people thanks to its intelligence level and the sensors it has, as the camera that is on its front or the touch screen of its chest.

Thanks to his speaking ability he can give lectures or classes in up to 15 languages, and his arms and chest tablet allow him to be interactive, attracting visitors to the places where he is.

It can also be used as an educational tool at different levels, as it can be programmed using Choregraphe (a programming language which uses blocks, and is very simple and intuitive), Python or C++ (classic and more powerful text-based programming languages).

Despite its fame, the Pepper robot ceased production in 2021 due to its low sales and high cost ($44,000).[[2]](#footnote-2)

A picture containing toy, doll, automaton

Description automatically generated

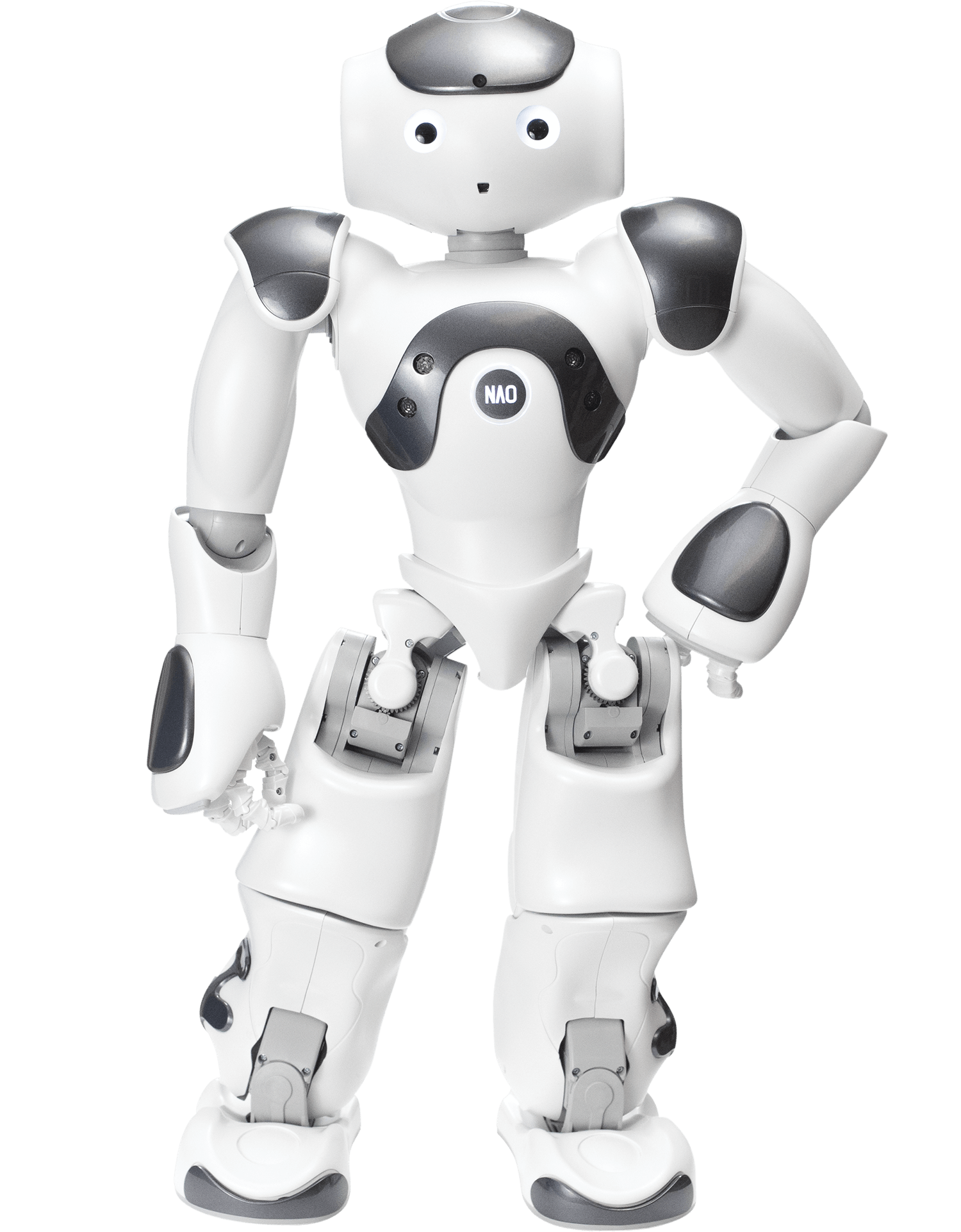
2 Image of the Pepper robot.

### Nao

NAO was the first robot created by *SoftBank* Robotic*[[3]](#footnote-3)* in 2004, and updated in the years 2006, 2011, 2012, 2014 and 2018. This humanoid robot has been used in various fields, from education (thanks to the ease of use provided by visual programming methods such as NaoQi), to nursing homes, where the humanoid robot was used to entertain residents.

In 2007 this robot replaced Aibo, Sony's robot dog as a standard platform for the Robocup, a project founded in 1997 to promote research on artificial intelligence using competitions between autonomous robots. In this competition, 6 of these robots are used in two teams to play football matches autonomously.[[4]](#footnote-4)

It has also been used in nursing homes and hospitals to help children with autism since, due to its shape and interactivity, it allows patients who are not able to communicate with people.



3 Image of the Nao robot.

Despite having been created in part as an educational robot, its price excludes it from most of the centers in which it could be used, so it is relegated to its use in universities or private centers.

### Voice assistants

Sticking to the definition, there is one more type of social robots: voice assistants. They are software programs based on artificial intelligence capable of recognizing spoken language and responding to voice commands to execute a series of tasks such as controlling lights, answering questions, making lists, etc. so that they allow users to interact with different platforms and hardware through voice.[[5]](#footnote-5)

Unlike the rest of the social robots, the problem with voice assistants is not their price, but their form. Most of these voice assistants use a smartphone as hardware, so their cost is almost zero, others use smart speakers in which the presence of a virtual assistant is more an addition than a main element. In both cases the problems are the same, little to nonphysical interaction and an environment too closed to investigate.

Imagen que contiene libro, tabla, cerca, puesto

Descripción generada automáticamente

4 Google nest mini voice assistant

## Work structure

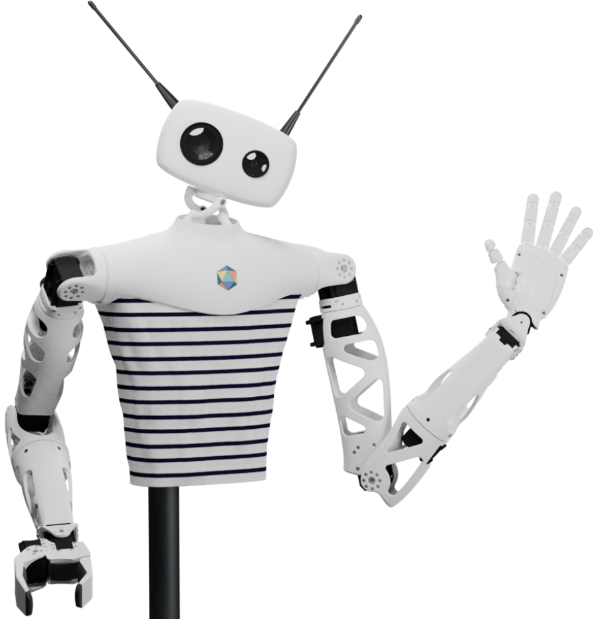
This document will be divided into 4 main blocks which in turn will be divided into subsections to give information as clearly as possible. These blocks will be:

1. **Objectives and methodology**, which will focus on explaining the objectives of this work, the methodology that will be followed to achieve them and the work plan, which specifies the necessary steps to follow the methodology.
2. **Reachy design**, which will explain how the parts needed to adapt the Reachy model to a simplified version have been designed, the hardware chosen to make it work and the software used to control and coordinate different Reachy functions.
3. **Implementation of Reachy,** where it will be explained how the integration of all the behaviors and functionalities of Reachy has been carried out, as well as a user manual to shows how to interact with Reachy.

# Objectives and methodology

## Objectives

The main goal of this work is to recreate a version of the Reachy robot with limited capabilities and an affordable budget. The possible alternatives for the motors will be investigated, as well as the controllers and the rest of the parts. To make the project affordable, a Raspberry pi 4 with 4 gigabytes of RAM will be used as a processor, with Ubuntu as the operating system and the ROS middleware, both open sources.



5 Complete Reachy robot.

The project will recreate Reachy's neck and head, both simplified but with similar functionalities. This robot must be able to obey a person's commands and hold a conversation with them, moving to look at them when they speak and responding physically and verbally.

## Methodology

To carry out this work, staged development[[6]](#footnote-6) (Mena, González, & Galván, 1999) will be used, a model generally used in software engineering but in this case will also be applied to hardware, such as in 3D printing

This development model is characterized by showing the client the software in different successive stages of development, and consists of five phases:

1. Concept evaluation phase.
2. Planning phase and product specification.
3. Development phase.
4. Testing and evaluation phase.
5. Launch phase.

Imagen que contiene Diagrama

Descripción generada automáticamente

6 Schematic representation of development by stages.

## Work plan

To follow the development explained in the previous section, the project has been divided into six different parts: Choice of software, artificial vision, communication, movement, 3D printing and coordination of behaviors. Each of these parts will go through the five stages of the chosen development.

### Choice of software

* **Concept evaluation phase**. Evaluation of different programs to control Reachy, such as Ubuntu, ROS or ROS2 and their versions.
* **Planning phase and product specification.** Choice of operating system and ROS version to be used.
* **Development phase.** Configuration and installation of ROS and Ubuntu.
* **Testing and evaluation phase.** Testing on both Ubuntu and Ubuntu server.
* **Launch phase.** The development of this part of Reachy is terminated.

### Machine vision.

* **Concept evaluation phase.**  Proposal of different modes of use of the cameras, tracking of objects, selection of objects according to specifications ... and choice of which elements will be used, and which ones will be discarded.
* **Planning phase and product specification.**  Planning how to make the vision work, what programs to use, how to implement the functionalities and what hardware will be needed.
* **Development phase.**  Programming of the tracking algorithms and the implementation that will be used to control the camera, as well as the necessary settings for both of them to work properly.
* **Testing and evaluation phase.** Performing tests to verify that the algorithms are working correctly and adjusting parameters such as the necessary limits for color filters if the tests fail. After this, it will be evaluated if the detection is rigid enough to terminate the development.
* **Launch phase.** The development of this part of Reachy is terminated.

### Communication

* **Concept evaluation phase**. Choice of the program used to carry out the communication.
* **Planning phase and product specification.** Assessment of forms of communication, as well as the commands that Reachy will obey and his reaction to them.
* **Development phase.** Programming of commands and their reactions.
* **Testing and evaluation phase.** Test of dialogue with different voices and evaluation of their operation.
* **Launch phase.** The development of this part of Reachy is terminated.

### Movement

* **Concept evaluation phase**. Assessment of the movement, its modes and whether they are feasible considering budget constraints.
* **Planning phase and product specification.** Choice of motors and mode of movement using a simplified neck instead of the original orbit neck.
* **Development phase.** Programming the movement of the motors and their combinations to make Reachy as expressive as possible.
* **Testing and evaluation phase.** Motion tests such as the speed of movement of the motors and how it affects the rest of the elements, or the power of the motors that must be able to move the entire head.
* **Launch phase.** The development of this part of Reachy is terminated.

### 3D Printing

* **Concept evaluation phase**. Choosing which parts of the Reachy model to use, which to modify, and which to create from scratch.
* **Planning phase and product specification.** Assembling the parts in a Reachy 3D model and checking the tolerances, to choose which parts to modify and which to create from scratch.
* **Development phase.** Modification and creation of the necessary parts using modeling programs such as Blender or freeCAD.
* **Testing and evaluation phase.** Printing the parts, checking the tolerances and making the necessary changes for the robot to function properly.
* **Launch phase.** The development and assembly of the rest of the elements is finished to obtain Reachy complete.

### Coordination of behaviors

* **Concept evaluation phase**. Assessment of possible types of communication.
* **Planning phase and product specification.** Choice of use of messages in ROS *topics* as a form of communication and the messages to be used.
* **Development phase.** Programming of the communication of the different nodes as well as the messages and parameters that will be used.
* **Testing and evaluation phase.** Communication tests, such as reaction speed or the fields that are needed between messages.
* **Launch phase.** The development of Reachy is terminated.

# Description of the work

The work description will be done in four parts: the operating system, the dialogue, the vision and the movement.

## Operating system

At the beginning of the project, the plan was to use the latest versions of the programs and the operating system, Ubuntu 20.04, and ROS 2 Foxy Fitzroy, but after various problems which will be explained below it has been decided to use ROS Noetic Ninjemys instead of ROS 2.

### ROS

ROS or [[7]](#footnote-7)*Robot Operating System* is an open-source middleware formed by a set of software libraries and tools that help create robotic applications. From controllers to state-of-the-art algorithms and powerful development tools, ROS contains everything needed to create robotics projects. It is currently divided into 2 branches: ROS and ROS 2, which develop in parallel in different distributions.

ROS works using processes called **nodes**. Each node performs a different task: for example, one node could read the computer's time of day, while another could receive the time and communicate it by voice. To allow asynchronous communication between nodes, topics are used, channels in which the nodes **publish** information if their purpose is to give information to others, or to those who **subscribe** if they are going to read certain information and process it, these *topics* have names like */time* or */sector*, which are used to identify them, subscribe, and publish in them. Each node can be at the same time a subscriber of one or more nodes and a publisher to one or more nodes, and they can use different messages: such as *int64* if you only want to publish a number, or *string* if you want to publish a phrase or messages of your own with several parameters for more specific uses.

There is also another form of communication between nodes: **services**. These allow synchronous communication between nodes using two messages: one **request**, which asks for certain data, and one **response**, which sends them. Being a synchronous communication, when a node uses a service asking for a request, it is blocked until it receives a response.

The initial intention was to use the most modern LTS (*Long Term Support*) version of ROS 2, *Foxy Fitzroy*, because of the advantages offered by ROS2, such as its greater security. Unfortunately, due to the novelty of ROS 2, it does not have as many add-ons as ROS.

After choosing this distribution, it was time to investigate ways to implement the functions necessary for the project: dialogue, movement of the engines and artificial vision.

There are codes created that allow the use of the camera by USB and the control of the motors. Unlike support for machine vision, dialogue support is scarce. There are options, such as Jaco, which run entirely on the host machine, but due to the power and storage limitations of the chosen processor, it has been preferred to use an implementation that works in the cloud. In this case, you could use the lex-ros2 implementation that uses Amazon's AWS (Amazon Web Services) to process the dialog and send a response.

Despite the potential of implementation using Amazon's AWS, problems have arisen such as its incompatibility with the distribution of ROS 2 Foxy Fitzroy, since it is only compatible with an earlier version of ROS 2 (Dashing Diademata, whose EOL *(End Of Life*) was in May 2022). Also, this version is only compatible with Ubuntu 18.

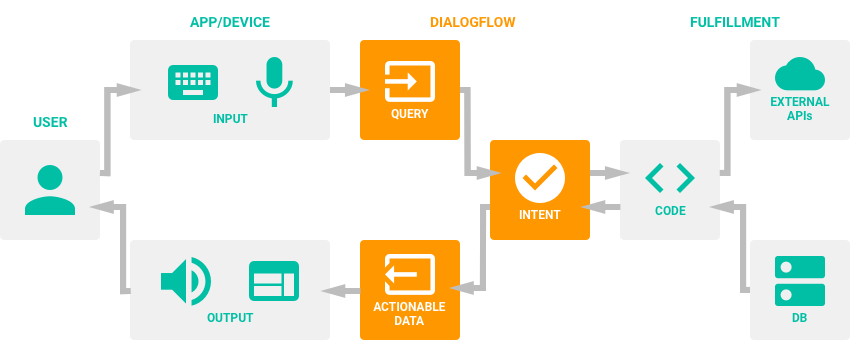
For this reason and given that the objective of the project does not contemplate the migration of third-party code to different versions, it has been decided to use ROS Noetic Ninjemys which, although it is not as efficient as ROS 2, will remain active until 2025 and works with Ubuntu 20.04.

## Dialogue

Dialogue is the main way to interact with Reachy: it allows you to communicate with him, give him orders or help him when he does not know what to do. To integrate this function, Dialogflow has been used[[8]](#footnote-8), an API (Application Programming Interface) from Google that can understand natural language and that provides infrastructure to recreate conversations and build dialogues in order to interact with the user in a fluid way.

To integrate this API into ROS, the implementation of Inteligent Robotics Lab for speech detection and Festival has been used to control sound and translate text to speech. The self-created *Reachy\_speech* node and the gb\_dialog launcher *gb\_dialog\_services\_soundplay* are used.

The *node Reachy\_speech* publishes to the *action* node whatever action has been requested by voice.



7 Diagram of operation of DialogFlow.

The phrases to interact with Reachy may vary slightly and you can still interact correctly with it. For example, to ask him to follow you, you can use phrases such as: *Follow me, Can you follow me?* o *Follow a person*.

## Artificial vision

A Logitech webcam model brio ultra-HD pro with a resolution of 4k and with USB connection has been used to implement vision capabilities. This model has been used beause it was the most accessible when doing the work, but it is replaceable by any other webcam as long as it has a resolution greater than or equal to 480 by 640.

Initially, when using ROS2, the driver implemented in ros-drivers was going to be used, but after changing versions, the official ROS implementation was used.

Visual detection is done using two nodes, one that captures the images from the camera and publishes them, called *usb\_cam camera\_publisher* and another one, *usb\_cam camera\_subscriber,* which subscribes to these images and the topic */action*. In this way *camera\_subscriber* receives the information of what to do, analyzes the images and using a map of 9 sectors, with 3 columns on the X axis and 3 rows on the Y axis, detects in which the objective to follow is located and communicates it using a *MotorMsg* message published the topic */motor\_msg.*  This message will then be read to the motor controller and contains 5 elements. Dependingon the message you receive from the dialog, this process may:

Un joven con una playera de color blanco

Descripción generada automáticamente con confianza media

8 Sector division visualization and face recognition.

### Follow cubes

The first step to track the cube is using a color filter. The filtered color that is the one chosen in the dialog, which comes in the *data* parameter of the *ActionMsg* message. If no color is specified, the green cube will be followed.

After performing the color filter, the contours are detected and filtered to eliminate those that are too small, such as errors that can be detected in the shadows, or too large as a wall. In this way we are left with only the outline that is most likely to belong to a cube, and if there are two equal cubes it follows the one that is closest.

By default, a person will be followed if asked to just follow something. Activating the "follow cube mode" without specifying a color will follow the green cube. After this, if you ask, for example, to follow the red cube, follow a person, and finally follow a cube without specifying the color, it will follow the last color that has been requested, in this case the red cube.

In case of using the "tracking mode", if Reachy does not find any target, he can be told if the target is on his right or left.

Pantalla de un celular con la imagen de una persona con la boca abierta

Descripción generada automáticamente con confianza baja

9 Image with yellow cube detected (left) and color filter (right)

Un hombre en una pantalla de un celular

Descripción generada automáticamente con confianza media

10 Image with blue cube detected (left) and color filter (right)

[](https://www.youtube.com/embed/KngwEZfFcAk?feature=oembed)

11 Video demonstration of color filter

## Movement

To allow the robot to move, two types of motors have been used: two Dynamixel AX18-A servo motors (analogous to the AX12 model of the same company, but more modern) for neck movement, and two Dynamixel XL-320 for the movement of the antennas.



12 Dynamixel AX18 (left) y XL-320 (right)).

These servo motors work using 3 pins: Voltage, ground, and a data pin, which uses a Duplex Serial asynchronous communication, so an external controller will be needed to translate the data. In this case, the U2D2 adapter and its PCB set have been used, along with a 12-volt, 5-amp power supply.

To control the engines using ROS we will use the implementation of Dynamixel and a program that is in [[9]](#footnote-9)[*dynamixel\_sdk\_examples*](https://github.com/Alberto-D/Reachy-TFG/tree/master/Code/DynamixelSDK/ros/dynamixel_sdk_examples) *motor\_controller* which subscribes to the topic */motor\_msg*. In this way the program reads the messages of the topic that tell you where to move, regardless of whether you are following a person or a cube.

The Dynamixel Wizard 2.0 application has also been used to configure the identifier and speed of movement of the motors, limiting it to prevent speeding from damaging the robot's neck due to the inertia produced by the weight of the head.

### Neck movement

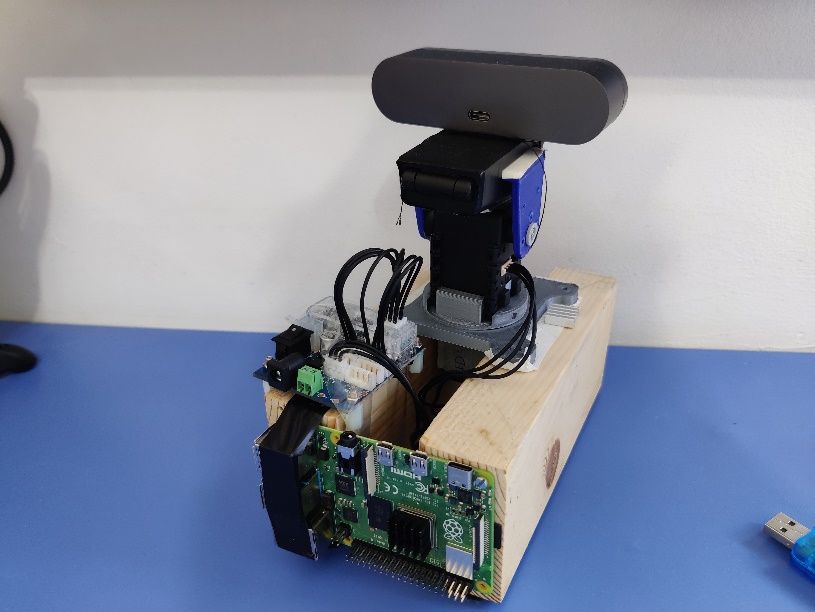
To move the motors, the node *dynamixel\_sdk\_examples read\_write\_node* must be launched*,* which reads whatever is published in the topic */set\_position* and moves the motors accordingly. This message must contain:

* The motor ID, in this case 1 for the X-axis motor and 2 for the Y-axis motor
* The position of the engine, a number between 0 and 1000 that indicates to the engine in which position to be placed, by default both are in the position 500.

This program also allows you to use the *get\_position service (ID)* which gives us the position of the motor with the specified ID.

For tracking, the program takes the current position of both motors. If the target is at its right or at its left (sectors 1 and 3 on the X axis) it publishes positions to gradually change in the motor with ID 1 until the target is in the central sector. Similarly, if the target is in sectors 1 or 3 of the Y axis, it publishes positions for the motor with ID 2 until it is centered.

For the rest of the actions, the field ischecked *motor\_action* and depending on the message, different actions are taken. For example, if the action is to deny, the engine with ID 1 moves to one side and to the other to simulate that it is shaking its head, if the message is to nod the engine with ID 2 moves up and down.



13 Prototype of the head to test the tracking.

### Antenna movement

Unlike neck motors which use 12 volts, antenna servo motors require only 7 volts, so various ways have been tried to adapt the motor controller to be able to use both models.

Unfortunately, after trying to use a voltage regulator, a variable resistor, and a voltage divider circuit, it has not been possible to move the antenna motors, so Dynamixel AX12 motors available in the laboratory have been used. Due to the larger size of these motors compared to the XL-320, the Reachy head has had to be modified to make it possible to hold these motors and the movement of the antennas.

[](https://www.youtube.com/embed/5C3LvWXiNrA?feature=oembed)

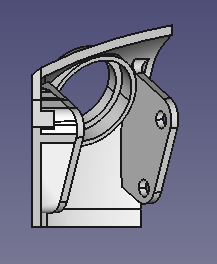
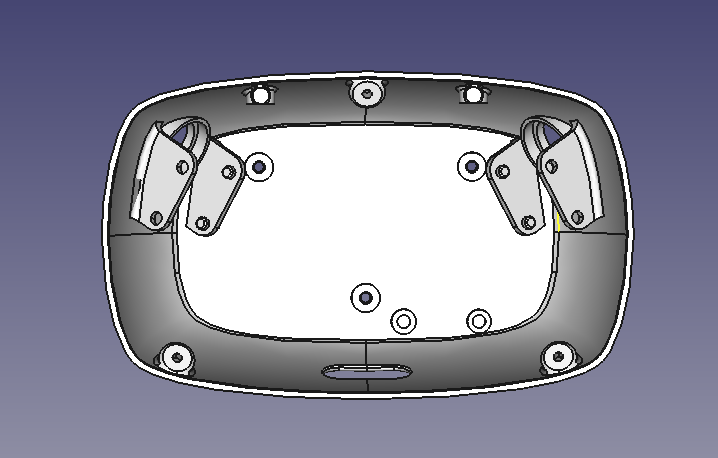
14 Video demonstration of movement.

## 3D Printing

Despite starting from the basis of the Reachy model created by Pollen robotics, it has been necessary to make modifications due to the simplification of the project.

For the 3D printing of the parts, the Sigma D25 model printer of the company BCN3D has been used accompanied by a spiral development. This development consists, in this case, in printing small pieces to check if they fit, and after this, modify them if necessary to print larger and larger pieces once it is certain that the tolerances are correct. Several iterations of each piece have been produced, but only the final ones have been included in the document.

16 Complete piece (left) and first printed part (right).



For the modification and creation of parts, Blender and FreeCAD have been used, both free of charge. The pieces that have been printed for this project can be divided into:

### Antenna support

In this case, instead of using magnets and bearings to attach the antennas to the head, a 3D modeled piece will be used to create a fixed connection between the motors and the piece wich holds the antenna.

Imagen que contiene agua, azul

Descripción generada automáticamente

16 First two iterations of the piece (left) and the piece holding the antenna (right).

It has also been necessary to modify the interface that connects the motor to the antenna to adapt it to the AX-12 motors.

Imagen que contiene tabla, pastel, azul, nieve

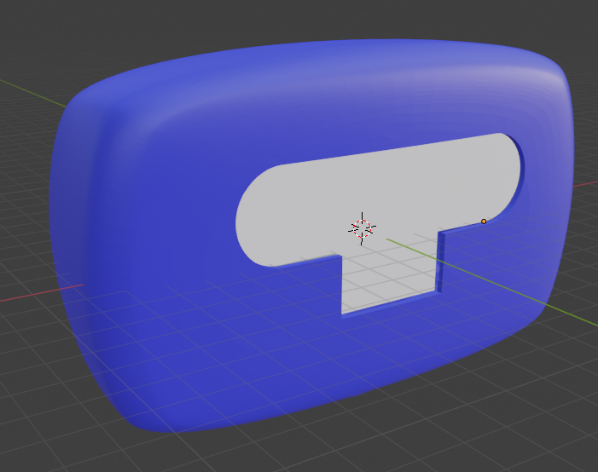
Descripción generada automáticamente

17 Final motor-antenna interface.

### Face

Reachy's original face is created to hold two cameras which act as eyes, antenna motors and a processing system. In this case, a single camera has been used, so the front of the head has been modified to use the appropriate webcam.

19 Different prototypes of Reachy's face.



After modifying the front, the interior has also been retouched to eliminate the existing camera mounts, and to add a support for the used webcam.

Icono

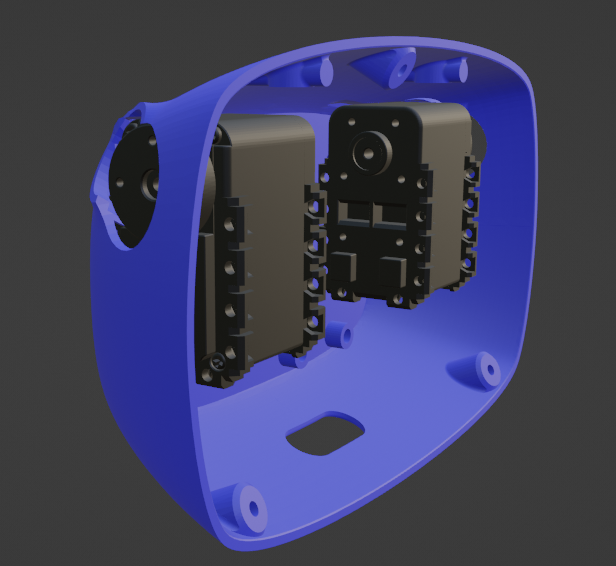
Descripción generada automáticamente

19 Back of the face modified for single camera use.

21 3D printed Reachy head



The rear of the head has also been modified to use the Dynamixel AX-12 motors, eliminating the motor mounts, as the Dynamixel AX12 can be screwed directly to the piece. In turn, the holes in the antennas and their supports have been modified to allow movement after changing the position of the motors.



21 Modeled back of the head.

Imagen que contiene azul, tabla, pastel, corte

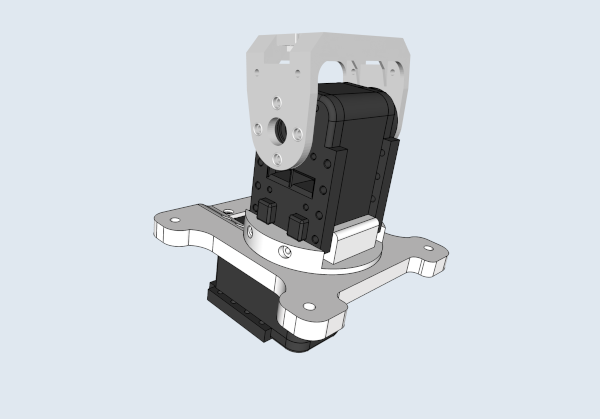
Descripción generada automáticamente

22 Rear part of the head printed with motors

### Neck

Reachy's original neck uses a mechanism created by Pollen robotics called Orbita, which uses three joints and three motors to endow Reachy with a fluid and expressive movement using 3 degrees of freedom.

To simplify this mechanism a system with two motors and two simple joints has been used, one at the base of the neck is a motor, which rotates from side to side, and one mounted in the first motor that moves from top to bottom and on which is the rest of the head rests, thus allowing the head of our simplified Reachy to move in two directions, allowing him to follow an object wich moves in front of it.



23 Orbit joint (left) and modified neck (right).

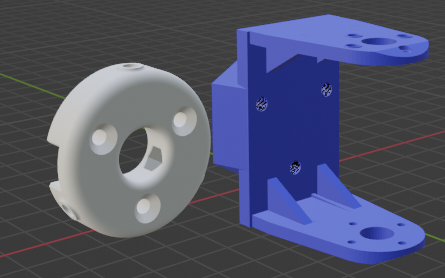
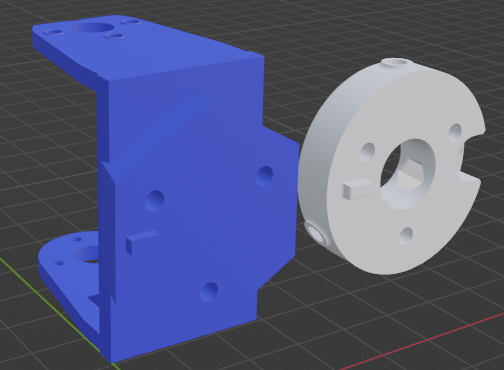
Imagen que contiene interior, tabla, cama, pequeño

Descripción generada automáticamente

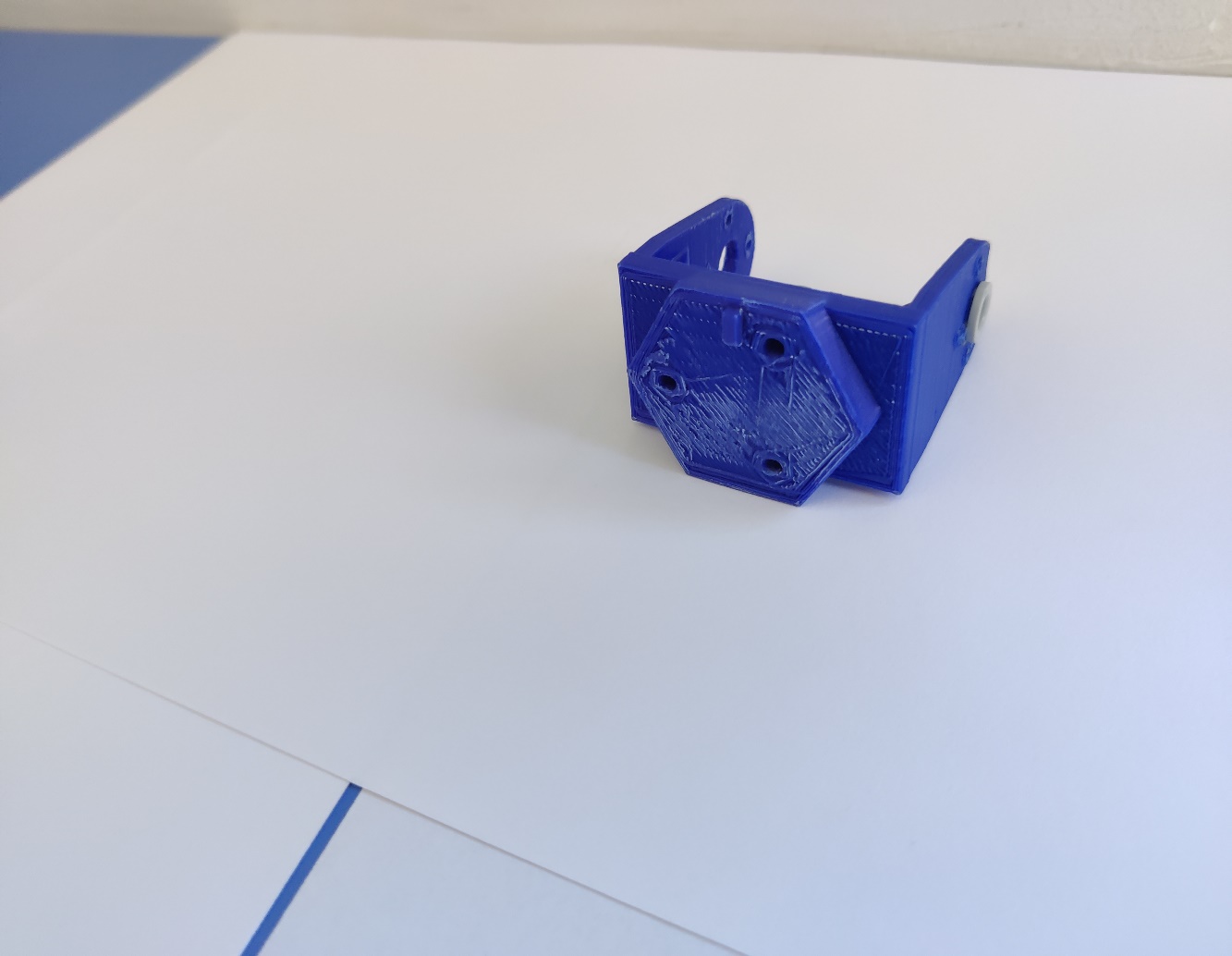
24 Printed modified neck

The neck which connects with the head has also been adapted, thus allowing the connection of the motors to the head to be printed in one piece.

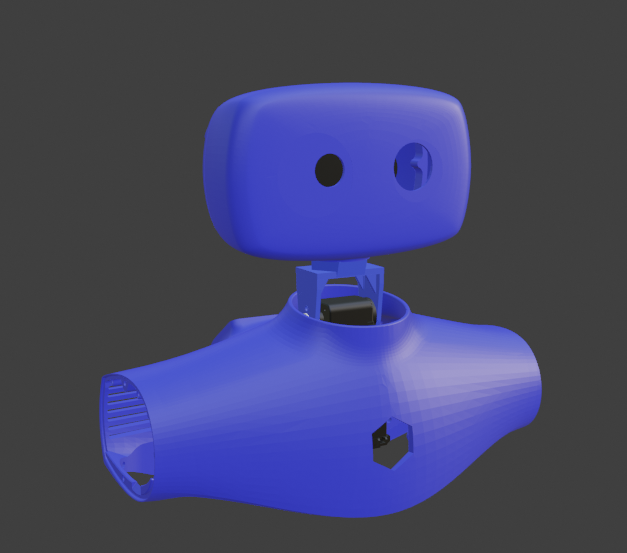
26 Modified (blue) and original (white) connection.



27 Printed modified head and neck connection



After designing all the parts, a simulation of the Reachy robot can see in Blender, and once all the parts are printed, and once all the parts have been printed and assembled the complete of the robot can be shown.



27 Complete Reachy structure in Blender.

Imagen que contiene interior, tabla, escritorio, computadora

Descripción generada automáticamente

28 Printed and assembled Reachy

# Reachy Implementation

To carry out the implementation of Reachy, two main topics have been used to allow communication between the 3 most important processes: dialogue, vision and movement. Two different messages are used to publish to these topics: ActionMsg and MotorMsg.

The main node is *Reachy\_speech*, which using DialogFlow and the sound driver located in [*gb\_dialog\_services\_soundplay.launch*](https://github.com/Alberto-D/Reachy-TFG/blob/master/Code/dialog/gb_dialog/gb_dialog/launch/gb_dialog_services_soundplay.launch) interprets voice commands, responds if necessary and publishes a message of type ActionMsg in the topic */action*. This message consists of:

* *int64 mode*, which determines behavior, following person, following cube, or speaking.
* *string action*, which is used to specify certain actions, such as shaking or nodding, or thinking before speaking.
* *string data*, used to give additional information, for example the color of the cube to be followed.

To detect objects and people, the *camera\_publisher* and *camera\_subscriber* nodes are used. The first takes the images from the camera and publishes them in the topic */usb\_cam/image\_raw,* the second subscribes to this topic and the topic /*action*. With the information from the two topics*,* the image camera\_subscriber processed according to what the action message asks for and publishes a MotorMsg message in the *topic /motor\_msg*. This message consists of:

* *int64 x\_sector*, which indicates the sector on the x-axis where the searched object is located.
* *int64 y\_sector*, which indicates the sector on the y-axis where the searched object is located.
* *int64 x*, the coordinate on the x-axis.
* *int64 y,* the coordinate on the y-axis.
* *string motor\_action*, a field used to give commands to motors outside of tracking. This way Reachy can nod, wake up or rejoice.

Finally, the *motor\_controller* node subscribes to the topic */motor\_msg* and uses the *set\_position* and *get\_position* services to control and calculate the position of the motors respectively. This node reads the message and activates motion control by following an object or performing the necessary actions depending on the motor\_action field of the message.

## User Manual

In order to use the robot, the *launcher of gb\_dialog Reachy\_launch.launch*, which launches the nodes that control all the behaviors of Reachy, and *gb\_dialog\_services\_soundplay.launch*, which will allow the use of sound, will be used.

Once the necessary nodes have been launched, the main talked commands used are:

*Hello*, which functions as a "power button", activating Reachy and moving it to the starting position, with both motors perpendicular to each other.

*Follow the COLOR cube*, where *COLOR* can be *green, yellow, blue,* or *red*, and activate the cube tracking mode of the color you specify.

*Follow me*, which activates a person's tracking mode.

*What time is it?* which makes Reachy tell us the time.

*On your left/right*, which serve to help Reachy in case he gets lost. After using this command, Reachy will move to the appropriate side and continue to search for its target.

*Tell me a joke*, which makes Reachy think of a joke and tell it.

*Tell me a fact*, which makes Reachy tell an interesting fact.

*Goodbye*, which "turns off" Reachy, causing him to look down imitating sleep.

*I am sad/happy*, which makes Reachy empathize and show him moving the antennas.

There are also other phrases that allow you to interact with Reachy. For example, it is able to answer different questions such as: *Where are you?* *How are you?* or *Are you a robot?* to which he responds by affirming or shaking his head.

# Conclusions and future work

The biggest conclusion drawn from the project is that, despite the difficulties, working with hardware is rewarding. While there are complicated parts, such as two motors which are listed as compatible with the same communication protocol not working with the same controller because they use different voltages, it is worth seeing the code materialize physically.

This work could be easily expandable. Due to the modularity of the project, the neck could be redesigned to use the Orbita mechanism by buying a single more engine, without having to reprint the head or modify the behaviors of certain nodes. The robot could be expanded by manufacturing one or two robotic arms that in conjunction with the camera and microphone allow to pick up and leave objects where specified, or even connect Reachy to a personal assistant and turn him into a kind of secretary who takes notes and helps in an office.

I chose this final degree project despite the difficulty for the same reason that I chose this career: the possibility of creating a robot from scratch, and despite the difficulties and concessions that have had to be made throughout the project, I consider that both decisions have been the right ones.

This work has required a multitude of skills both learned in the race and learned on their own. All the knowledge of ROS and ROS2 of the subject of Computer Architecture has been necessary to coordinate the behavior of the robot using different nodes that have a clear use, as well as to know which version of ROS to use and when to change; the skills obtained in Modeling and simulation of robots and Mechatronics to be able to modify and print the pieces, knowing which program to use at all times; the subject of Artificial Vision has also been necessary to be able to process the images captured by the camera and detect the objectives to follow. Other knowledge has also been used, such as that learned in Physics to avoid frying the motors or how to use a Raspberry, learned in Sensors and actuators.

Doing this project I also learned that hardware is not as easy as it seems, it requires a lot of information to be able to assemble a robot that works, which motor models are compatible with which controllers, the power needed to perform certain actions and how these limit the rest of the robot, or to look for information and check that it is not only reliable but that it is updated and functional with the necessary versions of other programs.

<https://github.com/Alberto-D/Reachy-TFG>

# Bibliography

Mena, A. A., González, F. J., & Galván, R. S. (1999). *DESARROLLO POR ETAPAS: UNA NUEVA GENERACIÓN DE PROCESOS DE DESARROLLO.*

1. Discover Reachy, a robotic platform based on AI – Reachy by Pollen Robotics, an open-source programmable humanoid robot. (s/f). Pollen-robotics.com. Recuperado el 7 de septiembre de 2022, de <https://www.pollen-robotics.com/reachy/> [↑](#footnote-ref-1)
2. anonymous. (2021, July 1). The production of the famous Pepper robot was suspended. infobae. <https://www.infobae.com/america/tecno/2021/07/01/se-suspendio-la-produccion-del-famoso-robot-pepper/> [↑](#footnote-ref-2)
3. RdR. (2020, February 3). 🥇 NAO Humanoid Robot; features and price in 2020. ROBOT MAGAZINE. <https://revistaderobots.com/robots-y-robotica/robot-nao-caracteristicas-y-precio/> [↑](#footnote-ref-3)
4. About: Nao (robot). (s/f). DBpedia. Retrieved April 12, 2022, from <https://dbpedia.org/page/Nao_(robot)> [↑](#footnote-ref-4)
5. Virtual voice assistants. (2019, October 11). Marketing-xxi.com. <https://www.marketing-xxi.com/voice-search-asistentes-voz-altavoces-inteligentes-seo-sem/asistentes-virtuales-voz> [↑](#footnote-ref-5)
6. <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjP3L73iMv5AhVL2xoKHXslAdAQFnoECAYQAQ&url=https%3A%2F%2Fdialnet.unirioja.es%2Fdescarga%2Farticulo%2F565251.pdf&usg=AOvVaw1YeEbZ-3PSHqsLE9PB5RTt> [↑](#footnote-ref-6)
7. <https://www.ros.org/> [↑](#footnote-ref-7)
8. <https://dialogflow.cloud.google.com/> [↑](#footnote-ref-8)
9. <http://wiki.ros.org/dynamixel> [↑](#footnote-ref-9)