Text

Description automatically generated

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE ROBÓTICA SOFTWARE

**TRABAJO FIN DE GRADO**

**CONSTRUCCIÓN Y DESPLIEGUE DEL ROBOT REACHY SIMPLIFICADO**

Autor: Alberto Delgado del Cerro

Tutor: José Miguel Guerrero Hernández

Curso académico 2021/2022

# Resumen

Desde la primera aparición del término “robot” en la obra *Rossum’s Universal Robots,* los humanos han imaginado máquinas con las que interactuar. A lo largo de la historia se han creado multitud de robots humanoides que permitan una interacción cómoda con personas, desde el precario Elektro de Joseph Barnett hasta la innovadora Sophia, creada por la compañía Hanson Robotics.

Este proyecto pretende crear una versión simplificada del robot Reachy[[1]](#footnote-1), un robot humanoide de acceso libre de la empresa Pollen Robotics creado precisamente para interactuar con personas. Esta versión utilizará ROS (*Robot Operating System*) para controlar las funciones y una Raspberry Pi como procesador, por lo que su coste será mucho menor que el del proyecto original.

La realización de este proyecto se fundamenta en tres partes principales: movimiento, visión e interacción con el humano, que se desarrollarán de forma separada para unirse finalmente obteniendo un robot que pueda realizar las tres tareas de forma coordinada.

Como resultado de este trabajo, se ha obtenido un robot funcional fabricado desde cero, diseñando e imprimiendo en 3D las piezas oportunas, y programando los comportamientos y las acciones necesarias para interactuar con una persona de distintas formas.

En conclusión, pese a los problemas producidos por la incompatibilidad del hardware, se ha creado un robot con una autonomía elevada, modular y que se puede modificar fácilmente, ya sea para añadir o eliminar funcionalidades.

Índice

[Resumen 2](#_Toc113434506)

[Tabla de contenido 5](#_Toc113434507)

[Introducción 6](#_Toc113434508)

[Robótica: Introducción general a la robótica 6](#_Toc113434509)

[Robótica social: Qué es y para qué se usa 7](#_Toc113434510)

[Robots sociales 7](#_Toc113434511)

[Pepper 7](#_Toc113434512)

[Nao 8](#_Toc113434513)

[Asistentes de voz 9](#_Toc113434514)

[Estructura del trabajo 10](#_Toc113434515)

[Objetivos y metodología 11](#_Toc113434516)

[Objetivos 11](#_Toc113434517)

[Metodología 12](#_Toc113434518)

[Plan de trabajo 13](#_Toc113434519)

[Elección de software 13](#_Toc113434520)

[Visión artificial. 13](#_Toc113434521)

[Comunicación 14](#_Toc113434522)

[Movimiento 14](#_Toc113434523)

[Impresión 3D 14](#_Toc113434524)

[Coordinación de comportamientos 15](#_Toc113434525)

[Descripción del trabajo desarrollado 16](#_Toc113434526)

[Sistema operativo 16](#_Toc113434527)

[ROS 16](#_Toc113434528)

[Diálogo 18](#_Toc113434529)

[Visión artificial 19](#_Toc113434530)

[Seguir caras 19](#_Toc113434531)

[Seguir cubos 20](#_Toc113434532)

[Movimiento 22](#_Toc113434533)

[Movimiento del cuello 23](#_Toc113434534)

[Movimiento de las antenas 24](#_Toc113434535)

[Impresión 3D 25](#_Toc113434536)

[Soporte de las antenas 25](#_Toc113434537)

[Cara 26](#_Toc113434538)

[Cuello 28](#_Toc113434539)

[Implementación del Reachy 31](#_Toc113434540)

[Manual de usuario 32](#_Toc113434541)

[Conclusiones y trabajos futuros 33](#_Toc113434542)

[Bibliografía 35](#_Toc113434543)

# Tabla de contenido

[1 Video demostración de movimiento del robot Atlas 6](#_Toc113434544)

[2 Imagen del robot Pepper. 8](#_Toc113434545)

[3 Imagen del robot Nao. 8](#_Toc113434546)

[4 Asistente de voz Google nest mini 9](#_Toc113434547)

[5 Robot Reachy completo. 11](#_Toc113434548)

[6 Representación esquemática del desarrollo por etapas. 12](#_Toc113434549)

[7 Esquema de funcionamiento de DialogFlow. 18](#_Toc113434550)

[8 Visualización de división en sectores y reconocimiento de cara. 20](#_Toc113434551)

[9 Imagen con cubo amarillo detectado (izq.) y filtro de color(der.) 21](#_Toc113434552)

[10 Imagen con cubo azul detectado (izq.) y filtro de color(der.) 21](#_Toc113434553)

[11 Video demostración del filtro de color 21](#_Toc113434554)

[12 Dynamixel AX18 (izq.) y XL-320 (der.). 22](#_Toc113434555)

[13 Prototipo de la cabeza para probar el seguimiento. 23](#_Toc113434556)

[14 Video demostracion de movimiento 24](#_Toc113434557)

[16 Pieza completa (izq.) y primera parte impresa (der.). 25](file:///C:\Users\alber\Downloads\TFGV3%20corregido.docx#_Toc113434558)

[16 Dos primeras iteraciones de la pieza (izq.) y pieza que sujeta la antena (der.). 26](#_Toc113434559)

[17 Interfaz motor-antena final. 26](#_Toc113434560)

[19 Diferentes prototipos de la cara de Reachy. 26](file:///C:\Users\alber\Downloads\TFGV3%20corregido.docx#_Toc113434561)

[19 Parte posterior de la cara modificada para el uso de una sola cámara. 27](#_Toc113434562)

[21 Parte delantera de la cabeza de Reachy impresa en 3D 27](file:///C:\Users\alber\Downloads\TFGV3%20corregido.docx#_Toc113434563)

[21 Parte trasera de la cabeza. 27](#_Toc113434564)

[22 Pieza trasera de la cabeza impresa con motores 28](#_Toc113434565)

[23 Orbita joint (izq.) y cuello modificado (der.). 28](#_Toc113434566)

[24 Cuello modificado impreso 29](#_Toc113434567)

[26 Conexión modificada (azul) y original (blanco). 29](file:///C:\Users\alber\Downloads\TFGV3%20corregido.docx#_Toc113434568)

[27 Conexión de la cabeza y el cuello modificada impresa 29](file:///C:\Users\alber\Downloads\TFGV3%20corregido.docx#_Toc113434569)

[27 Estructura de Reachy completa. 30](#_Toc113434570)

[28 Reachy completo 30](#_Toc113434571)

# Introducción

## Robótica: Introducción general a la robótica

El diccionario de Oxford define la robótica como la técnica que se utiliza en el diseño y la construcción de robots y aparatos que realizan operaciones o trabajos, generalmente en instalaciones industriales y en sustitución de la mano de obra humana.

A pesar de esto, el término robot ha evolucionado desde su creación, pasando de ser usado en instalaciones industriales hasta a llegar a abarcar desde asistentes de comunicación hasta máquinas de forma humanoide capaces de moverse e interactuar con su entorno mejor que la gran mayoría de las personas.

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

Video demostración de movimiento del robot Atlas

En la actualidad se está viviendo el auge de los robots domésticos. En contraste con los industriales, estos robots son más ligeros, más pequeños y menos potentes, lo que les permite compartir espacio con personas sin suponer un peligro para estas. Dentro de estos robots domésticos encontramos aquellos que ayudan en tareas domésticas como aspiradoras automáticas, que usan un mapa de la casa para limpiar de la manera más eficiente posible, o robots que dosifican la comida de las mascotas y las entretienen para hacer su día a día más llevadero. También se incluyen en los robots domésticos los que se comunican de forma directa con las personas, como los asistentes de voz o robots educativos. Estos últimos, que existen para interactuar con humanos de la forma más natural posible, se conocen como robots sociales.

## Robótica social: Qué es y para qué se usa

La robótica social se puede definir como la ciencia que usa un robot autónomo que interactúa y se comunica con humanos u otros agentes físicos autónomos siguiendo comportamientos sociales y reglas asociadas a su función.

Actualmente estos robots son cada vez más comunes, existen desde campos educativos como los robots Nao o Pepper de la empresa SoftBank Robotics hasta mascotas como la foca Paro, pero todos ellos tienen un problema en común: sus precios. El más barato de estos tres robots cuesta alrededor de unos 6.000 dólares, y el más caro unos 20.000, incluso el proyecto de Reachy, que es abierto y permite imprimir tus partes, cuenta con un presupuesto estimado de unos 2.000 dólares contando solo los materiales.

## Robots sociales

### Pepper

Pepper es un robot humanoide que mide aproximadamente 1 metro y 20 centímetros de altura. Debido a su diseño atractivo y dinámico, este robot se ha usado en distintos establecimientos de cara al público como hoteles o centros comerciales, y es capaz de interactuar con las personas gracias a su nivel de inteligencia y los sensores con los que cuenta, como la cámara que se encuentra en su frente o la pantalla táctil de su pecho.

Gracias a su capacidad de habla es capaz de dar conferencias o clases en hasta 15 idiomas, y sus brazos y la tableta de su pecho permiten que sea interactivo, atrayendo a visitantes a los lugares en los que se encuentra.

También puede usarse como herramienta educativa a diferentes niveles, pues se puede programar usando Choregraphe (una programación que se realiza usando bloques sencillos e intuitivos), Python o C++ (lenguajes de programación clásicos basados en texto y más potentes)

A pesar de su fama, el robot Pepper cesó su fabricación[[2]](#footnote-2) en el año 2021 debido a sus bajas ventas y su alto coste (44000$).

A picture containing toy, doll, automaton

Description automatically generated

Imagen del robot Pepper.

### Nao

NAO fue el primer robot creado por *SoftBank* Robotic*[[3]](#footnote-3)* en el año 2004, y actualizado en los años 2006, 2011, 2012 2014 y 2018. Este robot humanoide se ha usado en diversos ámbitos, desde la educación (gracias a la facilidad de programarlo usando métodos visuales como NaoQi), hasta en residencias de ancianos, donde el robot humanoide se usaba para entretener y dar compañía a los residentes.

Este robot sustituyo[[4]](#footnote-4) en 2007 a Aibo, el perro robot de Sony como plataforma estándar para la Robocup, un proyecto fundado en 1997 para promover la investigación sobre inteligencia artificial usando competiciones entre robots autónomos. En esta competición se usan 6 de estos robots en dos equipos para jugar partidos futbol de forma autónoma.

También se ha usado en residencias de ancianos y hospitales para ayudar a niños con autismo ya que, debido a su forma y su interactividad, permite comunicarse a pacientes que no son capaces de hacerlo con personas.

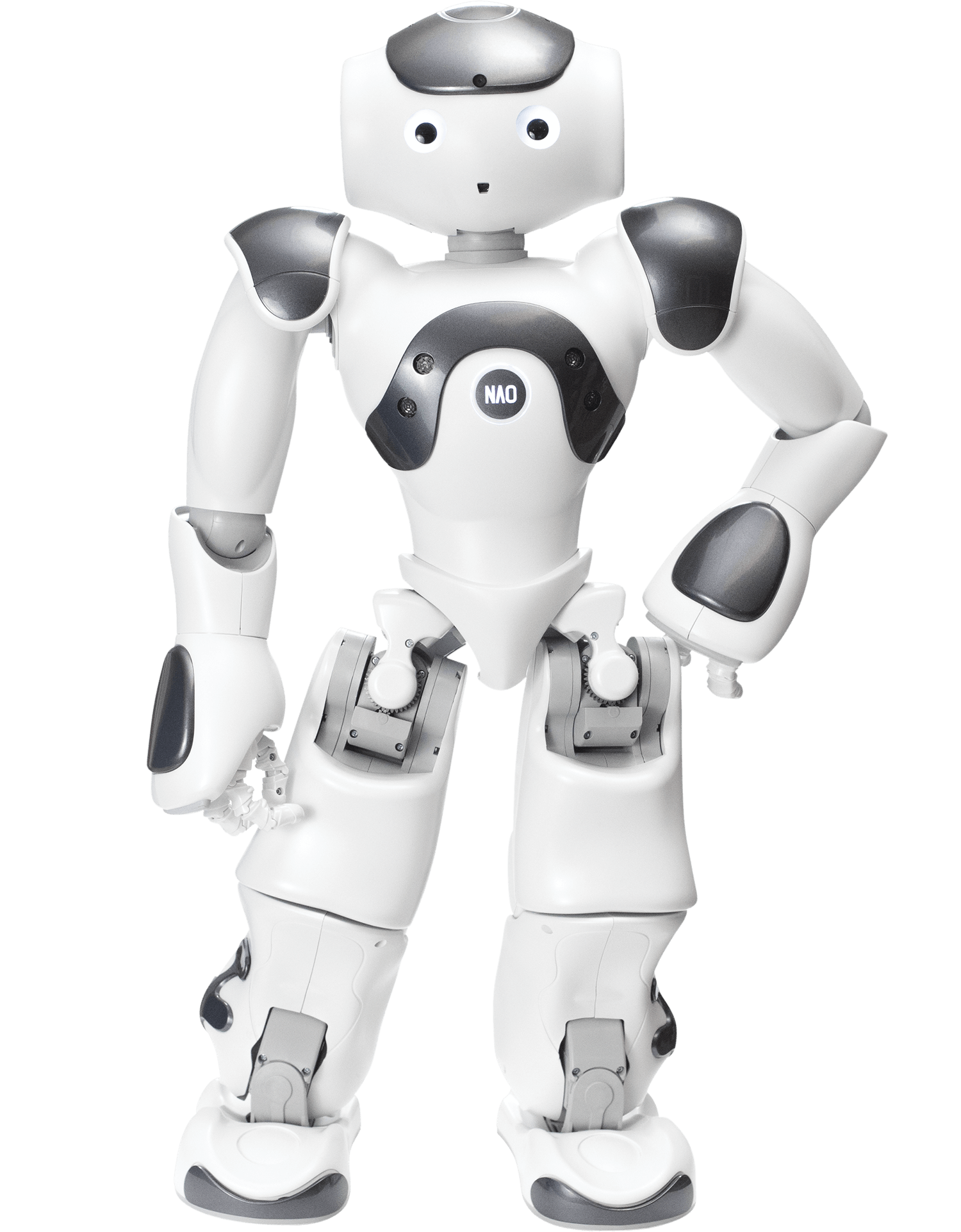


Imagen del robot Nao.

A pesar de haber sido creado en parte como robot educativo, su precio lo excluye de la mayoría de los centros en los que podría usarse, por lo que se ve relegado a su uso en universidades o centros privados.

### Asistentes de voz

Ajustándose a la definición, existen también otro tipo de robots sociales: los asistentes de voz. Son programas[[5]](#footnote-5) de software basados en la inteligencia artificial capaces de reconocer el lenguaje hablado y responder a comandos de voz para ejecutar una serie de tareas como controlar luces, responder preguntas, realizar listas, etc. de modo que posibilitan a los usuarios interactuar con diferentes plataformas y hardware mediante la voz.

Al contrario que el resto de los robots sociales, el problema de los asistentes de voz no es su precio, sino su forma. La mayoría de estos asistentes de voz utilizan un smartphone como hardware, por lo que su coste es casi nulo, y otros utilizan altavoces inteligentes en los que la presencia de un asistente virtual es más un añadido que un elemento principal. En ambos casos los problemas son el mismo, una forma poco o nada interactiva y un entorno demasiado cerrado como para investigar.

Imagen que contiene libro, tabla, cerca, puesto

Descripción generada automáticamente

Asistente de voz Google nest mini

## Estructura del trabajo

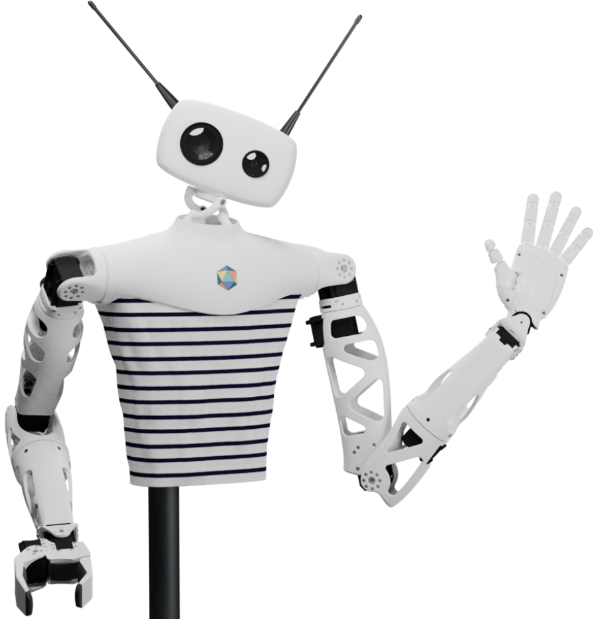
Este documento se dividirá en 4 bloques principales que a su vez se dividirán en subsecciones para dar información de la forma más clara posible. Estos bloques serán:

* **Objetivos y metodología**, que se centrará en explicar los objetivos de este trabajo, la metodología que se seguirá para conseguirlos y el plan de trabajo, que especifica los pasos necesarios para seguir la metodología.
* **Diseño de Reachy**, en el que se explicará cómo se han diseñado las piezas necesarias para adaptar el modelo de Reachy a una versión simplificada, el hardware elegido para hacerlo funcionar y el software utilizado para controlar y coordinar las diferentes funciones de Reachy.
* **Implementación de Reachy,** donde seexplicará cómo se ha realizado la integración de todos los comportamientos y funcionalidades de Reachy, así como un manual de usuario que muestre como interactuar con Reachy.

# Objetivos y metodología

## Objetivos

El objetivo principal de este trabajo es recrear una versión del robot Reachy con capacidades limitadas y un presupuesto asequible. Para ello se investigarán las posibles alternativas para los motores, así como los controladores y el resto de las piezas. Para que el proyecto sea realmente asequible, se usará como procesador una Raspberry pi 4 con 4 gigas de RAM, con Ubuntu como sistema operativo y el middleware ROS, ambos de acceso libre.



Robot Reachy completo.

En el proyecto se recreará el cuello y la cabeza de Reachy, ambos simplificados para obtener un funcionamiento similar. Este robot deberá ser capaz de obedecer a las órdenes de una persona y mantener una conversación con esta, moviéndose para mirarle cuando hable y respondiendo de forma física y verbal.

## Metodología

Para realizar este trabajo se va a utilizar el desarrollo por etapas[[6]](#footnote-6) (Mena, González, & Galván, 1999), un modelo usado generalmente en la ingeniería de software pero que en este caso se aplicará también al hardware, como en la impresión 3D.

Este modelo de desarrollo se caracteriza por mostrar al cliente el software en diferentes estados sucesivos de desarrollo, y consta de cinco fases:

* Fase de evaluación de conceptos.
* Fase de planificación y especificación del producto.
* Fase de desarrollo.
* Fase de pruebas y evaluación.
* Fase de lanzamiento.

Imagen que contiene Diagrama

Descripción generada automáticamente

Representación esquemática del desarrollo por etapas.

## Plan de trabajo

Para seguir el desarrollo explicado en el apartado anterior se ha dividido el proyecto en seis partes diferenciadas: Elección de software, visión artificial, comunicación, movimiento, impresión 3D y coordinación de comportamientos. Cada una de estas partes pasará por las cinco etapas del desarrollo elegido.

### Elección de software

* **Fase de evaluación de conceptos**. Valoración de distintos programas para controlar a Reachy, como Ubuntu, ROS o ROS2 y sus versiones.
* **Fase de planificación y especificación del producto.** Elección del sistema operativo y versión de ROS que se usará.
* **Fase de desarrollo.** Configuración e instalación de ROS y Ubuntu.
* **Fase de pruebas y evaluación.** Pruebas de funcionamiento tanto en Ubuntu como en Ubuntu server.
* **Fase de lanzamiento.** Se da por acabado el desarrollo de esta parte de Reachy.

### Visión artificial.

* **Fase de evaluación de conceptos.** Proposición de diferentes modos de uso de las cámaras, seguimiento de objetos, selección de objetos según especificaciones… y elección de qué elementos se usarán y cuáles se descartarán.
* **Fase de planificación y especificación del producto.** Planificación de cómo hacer funcionar la visión, que programas usar, cómo implementar las funcionalidades y qué hardware será necesario.
* **Fase de desarrollo.** Programación de los algoritmos de seguimiento y la implementación que se usará para controlar la cámara, así como las configuraciones necesarias para que ambos funcionen correctamente.
* **Fase de pruebas y evaluación.** Realización de pruebas para comprobar que los algoritmos funcionan correctamente y ajuste de parámetros como los límites necesarios para los filtros de color si es que las pruebas fallan. Tras esto se evaluará si la detección es lo suficientemente rígida para dar por terminado el desarrollo.
* **Fase de lanzamiento.** Se da por acabado el desarrollo de esta parte de Reachy.

### Comunicación

* **Fase de evaluación de conceptos**. Elección del programa usado para realizar la comunicación.
* **Fase de planificación y especificación del producto.** Valoración de formas de comunicación, así como los comandos que obedecerá Reachy y su reacción ante estos.
* **Fase de desarrollo.** Programación de los comandos y sus reacciones.
* **Fase de pruebas y evaluación.** Prueba de diálogo con distintas voces y evaluación de su funcionamiento.
* **Fase de lanzamiento.** Se da por acabado el desarrollo de esta parte de Reachy.

### Movimiento

* **Fase de evaluación de conceptos**. Valoración del movimiento, sus modos y si son factibles teniendo en cuenta las limitaciones de presupuesto.
* **Fase de planificación y especificación del producto.** Elección de los motores y el modo de movimiento usando un cuello simplificado en lugar del cuello órbita original.
* **Fase de desarrollo.** Programación del movimiento de los motores y sus combinaciones para hacer a Reachy lo más expresivo posible.
* **Fase de pruebas y evaluación.** Pruebas de movimiento como la velocidad de movimiento de los motores y cómo esta afecta al resto de elementos, o la potencia de los motores que deben ser capaz de mover toda la cabeza.
* **Fase de lanzamiento.** Se da por acabado el desarrollo de esta parte de Reachy.

### Impresión 3D

* **Fase de evaluación de conceptos**. Elección de qué piezas del modelo de Reachy usar, cuáles modificar y cuáles crear desde cero.
* **Fase de planificación y especificación del producto.** Montaje de las piezas en un modelo 3D de Reachy y comprobación de las tolerancias, para elegir qué piezas modificar y cuáles crear desde cero.
* **Fase de desarrollo.** Modificación y creación de las piezas necesarias usando programas de modelado como Blender o freeCAD.
* **Fase de pruebas y evaluación.** Impresión de las piezas, comprobación de las tolerancias y realización de los cambios necesarios para que el robot funcione correctamente.
* **Fase de lanzamiento.** Se da por acabado el desarrollo y montaje del resto de los elementos para obtener a Reachy completo.

### Coordinación de comportamientos

* **Fase de evaluación de conceptos**. Valoración de tipos de comunicación posibles.
* **Fase de planificación y especificación del producto.** Elección de uso de mensajes en *topics* de ROS como forma de comunicación y los mensajes que se usarán.
* **Fase de desarrollo.** Programación de la comunicación de los distintos nodos así como los mensajes y parámetros que se usarán.
* **Fase de pruebas y evaluación.** Pruebas de comunicación, como velocidad de reacción o los campos que se necesitan entre mensajes.
* **Fase de lanzamiento.** Se da por acabado el desarrollo de Reachy.

# Descripción del trabajo desarrollado

La descripción del trabajo se realizará en cuatro partes, el sistema operativo, el diálogo, la visión y el movimiento.

## Sistema operativo

Al comenzar el proyecto se pensaba usar las últimas versiones de los programas y el sistema operativo, Ubuntu 20.04 y ROS 2 Foxy Fitzroy, pero tras diversos problemas que se explicarán a continuación se ha decidido usar ROS Noetic Ninjemys en lugar de ROS 2.

### ROS

ROS[[7]](#footnote-7) o *Robot Operating System* es un middleware de código abierto formado por un conjunto de bibliotecas de software y herramientas que ayudan a crear aplicaciones robóticas. Desde controladores hasta algoritmos de última generación y con potentes herramientas de desarrollo, ROS contiene todo lo necesario para crear proyectos de robótica. Actualmente se divide en 2 ramas: ROS y ROS 2, que se desarrollan de forma paralela en diferentes distribuciones.

ROS funciona usando procesos llamados **nodos**. Cada nodo realiza una tarea distinta: por ejemplo, un nodo podría leer la hora del día del ordenador, mientras que otro podría recibir la hora y comunicarla por voz. Para permitir la comunicación asíncrona entre nodos se usan ***topics***, canales en los que los nodos **publican** información si su propósito es dar información a otros, o a los que se **suscriben** si van a leer cierta información y procesarla, estos *topics* tienen nombre como */hora* o */sector*, que se utilizan para identificarlos, suscribirse y publicar. Cada nodo puede ser a la vez suscriptor de uno o más nodos y publicar en uno o varios nodos, y pueden usar distintos mensajes: como *int64* si solo se quiere publicar un número, o *string* si se quiere publicar una palabra, una frase o mensajes propios con varios parámetros para usos más específicos.

Existe también otra forma de comunicación entre nodos: los **servicios**. Estos permiten una comunicación síncrona entre nodos usando dos mensajes: uno de **solicitud**, que pide ciertos datos, y uno de **respuesta**, que los envía. Al ser una comunicación síncrona, cuando un nodo usa un servicio pidiendo una solicitud, se bloquea hasta recibir una respuesta.

La intención inicial era usar la versión LTS (*Long Term Support*) más moderna de ROS 2, *Foxy Fitzroy*, por las ventajas que ofrece ROS2, como su mayor seguridad. Lamentablemente, debido a la novedad de ROS 2, este no cuenta con tantos complementos como ROS.

Tras elegir esta distribución, se comenzaron a investigar formas de implementar las funciones necesarias para el proyecto: diálogo, movimiento de los motores y visión artificial.

Existen códigos creados que permiten el uso de la cámara por USB[[8]](#footnote-8) y el control de los motores. Al contrario que con el soporte para la visión artificial, el soporte del diálogo es escaso. Existen opciones, como Jaco[[9]](#footnote-9) , que se ejecutan por completo en la maquina anfitriona, pero debido a las limitaciones de potencia y almacenamiento del procesador elegido, se ha preferido usar una implementación que funcione en la nube. En este caso se podría usar la implementación de lex-ros2[[10]](#footnote-10) que se vale de los AWS (Amazon Web Services) de Amazon para procesar el dialogo y enviar una respuesta.

A pesar del del potencial de la implementación usando el AWS de Amazon, han surgido problemas como su incompatibilidad con la distribución de ROS 2 Foxy Fitzroy, pues solo es compatible con una versión anterior de ROS 2 (Dashing Diademata, cuya EOL *(End Of Life*) fue en mayo de 2022). Además, esta versión solo es compatible con Ubuntu 18.

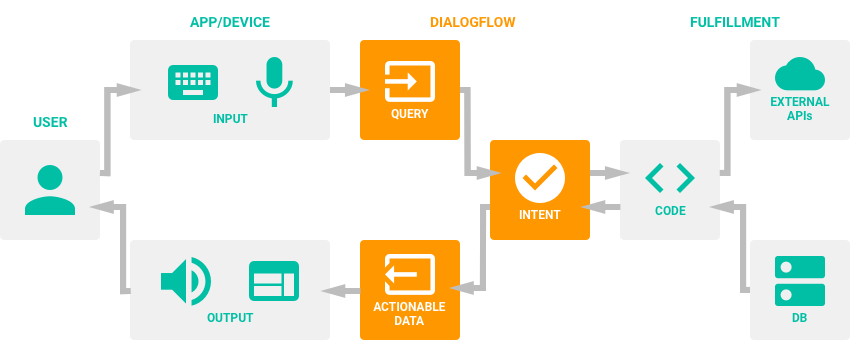
Por esta razón, y dado que el objetivo del proyecto no contempla la migración de código de terceros a distintas versiones, se ha decidido utilizar ROS Noetic Ninjemys que, si bien no es tan eficiente como ROS 2, se mantendrá en activo hasta 2025 y funciona con Ubuntu 20.04.

## Diálogo

El diálogo es la forma principal de interactuar con Reachy: permite comunicarse con él, darle órdenes o ayudarle cuando no sepa qué hacer. Para integrar esta función se ha utilizado Dialogflow[[11]](#footnote-11), una API (Application Programming Interface) de Google que es capaz de entender el lenguaje natural y que provee infraestructura para recrear conversaciones y construir diálogos con el fin de interactuar con el usuario de manera fluida.

Para integrar esta API en ROS se ha usado la implementación de Inteligent Robotics Lab[[12]](#footnote-12) para la detección del habla y Festival[[13]](#footnote-13)para controlar el sonido y traducir el texto a sonido (*text to speech)*. Se usan el nodo *Reachy\_speech* de creación propia y el launcher *gb\_dialog\_services\_soundplay* de gb\_dialog.

El nodo *Reachy\_speech* publica en el nodo *action* qué acción se ha pedido por voz.



Esquema de funcionamiento de DialogFlow.

Toda la comunicación hablada se realizará en inglés, pues la detección de frases en ese idioma es mucho más robusta que en español. Las frases para interactuar con Reachy pueden variar ligeramente y aun así se puede interactuar correctamente con él. Por ejemplo, para pedirle que te siga se pueden usar frases como: *Follow me, ¿Can you follow me?* o *Follow a person*.

## Visión artificial

Para la visión se ha usado la cámara web Logitech modelo brio ultra HD pro con una resolución de 4k y con conexión por USB. Se ha utilizado este modelo por ser el más accesible a la hora de realizar el trabajo, pero es sustituible por cualquier otra webcam siempre que tenga una resolución mayor o igual a 480 por 640.

En un principio al usar ROS2 se iba a usar el controlador implementado en ros-drivers[[14]](#footnote-14), pero tras cambiar de versión se ha usado la implementación oficial de ROS[[15]](#footnote-15).

La detección visual se realiza usando dos nodos, uno que captura las imágenes de la cámara y las publica llamado *usb\_cam camera\_publisher* y otro que se suscribe, *usb\_cam camera\_subscriber* a estas imágenes y al topic */action*. De esta forma recibe la información de qué debe hacer, analiza las imágenes y usando un mapa de 9 sectores, con 3 columnas en el eje X y 3 filas en el eje Y, detecta en cual se encuentra el objetivo a seguir y lo comunica usando un mensaje *MotorMsg* publicado el topic */motor\_msg.* Este mensajeluego será leído el controlador de los motores, y contiene 5 elementos. Dependiendo del mensaje que reciba desde el diálogo, este proceso podrá:

### Seguir caras

Para realizar el seguimiento de caras se usa la función faceCascade.detectMultiScale de openCV, que proporciona las coordenadas de todas las caras que se detectan en pantalla. Tras esto se filtran los datos para eliminar caras que sean demasiado pequeñas o grandes para evitar errores, y se elige seguir a la persona que se encuentre más cerca del robot. Si dos personas se encuentran a la misma distancia se sigue a la primera que se ha detectado.

Una vez se tienen las coordenadas de la persona, se saca el sector en el que se encuentra y se publica un mensaje en */motor\_msg* con la información necesaria.

Un joven con una playera de color blanco

Descripción generada automáticamente con confianza media

Visualización de división en sectores y reconocimiento de cara.

### Seguir cubos

Para realizar el seguimiento de los cubos se usa primero un filtro de color. El color que se filtra es el elegido en el diálogo, que viene en el parámetro *data* del mensaje *ActionMsg*. Si no se especifica ningún color, se seguirá al cubo verde.

Tras realizar el filtro de color se detectan los contornos y se filtran para eliminar los que son demasiado pequeños, como errores que se pueden detectar en las sombras, o demasiado grandes como una pared. De esta forma nos quedamos solo con el contorno que es más probable que pertenezca a un cubo, y si hay dos cubos iguales se sigue al que esté más cerca.

Por defecto, se seguirá a una persona si se pide tan solo seguir a algo. Si se activa el “modo seguir cubo” sin especificar un color se seguirá al cubo verde. Tras esto, si se pide, por ejemplo, seguir al cubo rojo, seguir a una persona, y finalmente seguir a un cubo sin especificar el color, se seguirá al último color que se haya pedido, en este caso al cubo rojo.

En caso usar el “modo de seguimiento”, si Reachy no encuentra ningún objetivo, pedirá ayuda, tras lo que se le podrá decir si el objetivo se encuentra a su derecha o a su izquierda.

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

Descripción generada automáticamente con confianza baja

Imagen con cubo amarillo detectado (izq.) y filtro de color(der.)

Un hombre en una pantalla de un celular

Descripción generada automáticamente con confianza media

Imagen con cubo azul detectado (izq.) y filtro de color(der.)

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

Video demostración del filtro de color

## Movimiento

Para permitir que el robot se mueva se han usado dos tipos de motores: dos servomotores Dynamixel AX18-A (análogo al modelo AX12 de la misma compañía, pero más moderno) para el movimiento del cuello, y dos Dynamixel XL-320 para el movimiento de las antenas.



Dynamixel AX18 (izq.) y XL-320 (der.).

Estos servomotores funcionan usando 3 pines: Voltaje, tierra y un pin de datos, que usa una comunicación asíncrona Duplex Serial, por lo que será necesario un controlador externo para traducir los datos. En este caso se ha usado el adaptador U2D2 y su set de PCB, junto con una fuente de alimentación de 12 voltios y 5 amperios.

Para controlar los motores usando ROS se ha usado la implementación de Dynamixel[[16]](#footnote-16) y un programa que se encuentra en [*dynamixel\_sdk\_examples*](https://github.com/Alberto-D/Reachy-TFG/tree/master/Code/DynamixelSDK/ros/dynamixel_sdk_examples) *motor\_controller* que se suscribe al topic */motor\_msg*. De esta forma el programa lee los mensajes del topic que le dicen dónde debe moverse, sin importar si está siguiendo a una persona o a un cubo.

Se ha usado además la aplicación Dynamixel Wizard 2.0 para configurar el identificador y la velocidad de movimiento de los motores, limitándola para evitar que el exceso de velocidad dañe el cuello del robot debido a la inercia producida por el peso de la cabeza.

### Movimiento del cuello

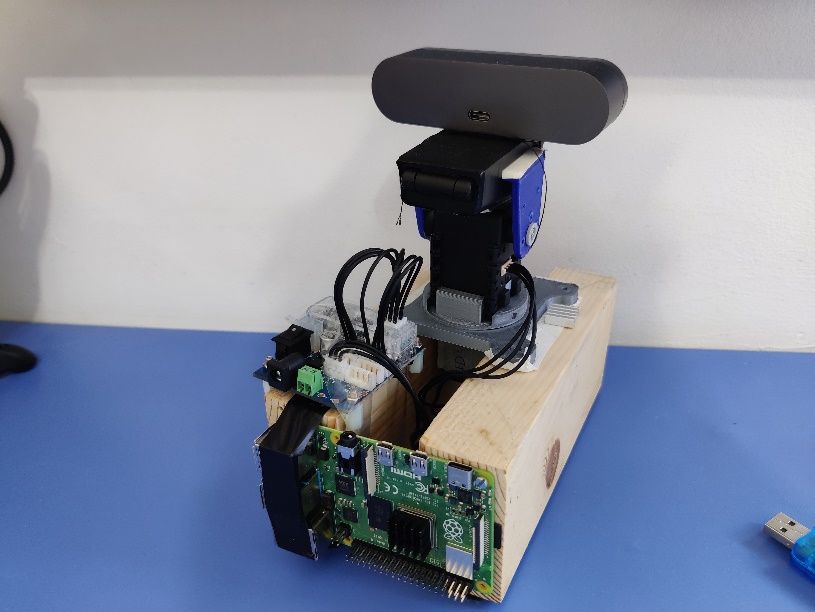
Para mover los motores se debe lanzar *dynamixel\_sdk\_examples read\_write\_node,* quelee lo publicado en el topic */set\_position* y mueve los motores de forma acorde. Este mensaje debe contener:

* El ID del motor, en este caso 1 para el motor del eje X y 2 para el motor del eje Y
* La posición del motor, un numero ente 0 y 1000 que indica al motor en que posición colocarse, por defecto ambos se encuentran en la posición 500.

Este programa también permite usar el servicio *get\_position (ID)* que nos da la posición del motor con el ID especificado.

Para el seguimiento, el programa toma la posición actual de ambos motores. Si el objetivo está a su derecha o a su izquierda (sectores 1 y 3 en el eje X) se publican posiciones que cambian gradualmente en el motor con ID 1 hasta que el objetivo esté en el sector central. De forma similar, si el objetivo está en los sectores 1 o 3 del eje Y, se publican posiciones para el motor con ID 2 hasta centrarlo.

Para el resto de las acciones se comprueba el campo *motor\_action* y dependiendo del mensaje se toman diferentes acciones. Por ejemplo, si la acción es negar se mueve el motor con ID 1 hacia un lado y hacia el otro para simular que está negando con la cabeza, si el mensaje es asentir se mueve el motor con ID 2 hacia arriba y hacia abajo.



Prototipo de la cabeza para probar el seguimiento.

### Movimiento de las antenas

Al contrario que los motores del cuello que usan 12 voltios, los servomotores de las antenas requieren solo 7, por lo que se ha intentado de diversas formas adaptar el controlador de los motores para poder usar ambos modelos.

Desgraciadamente, tras intentar usar un regulador de voltaje, una resistencia variable y un circuito divisor de tensión, no se ha conseguido mover los motores de las antenas, por lo que se han usado motores Dynamixel AX12 disponibles en el laboratorio. Debido al mayor tamaño de estos motores respecto a los XL-320, se ha tenido que modificar la cabeza de Reachy para hacer posible la sujeción de estos motores y el movimiento de las antenas.

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

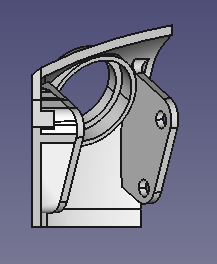
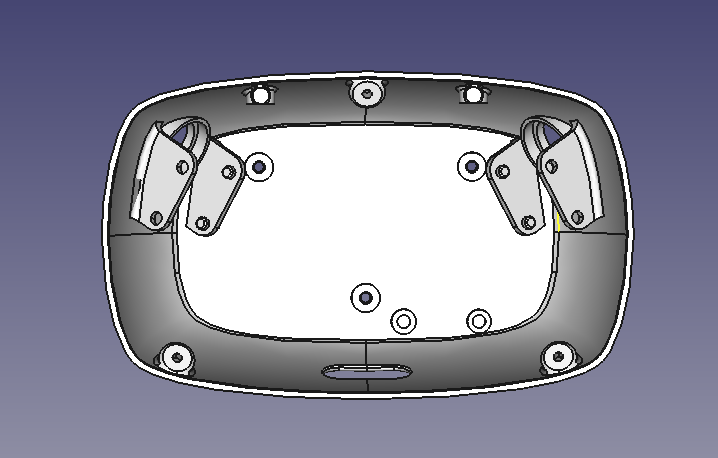
Video demostración de movimiento

## Impresión 3D

A pesar de partir de la base del modelo de Reachy creado por Pollen robotics[[17]](#footnote-17), ha sido necesario realizar modificaciones debido a la simplificación del proyecto.

Para la impresión 3D de las piezas se ha usado la impresora modelo Sigma D25 de la empresa BCN3D y un desarrollo en espiral. Este desarrollo consiste, en este caso, en imprimir piezas pequeñas para comprobar que encajan, y tras esto ir modificándolas si fuera necesario para imprimir piezas cada vez más grandes una vez es seguro que las tolerancias son correctas. Se han producido varias iteraciones de cada pieza, pero solo se han incluido en el documento las finales.

Pieza completa (izq.) y primera parte impresa (der.).



Para la modificación de piezas o creación de piezas nuevas se han usado los programas Blender y FreeCAD, ambos gratuitos. Las piezas que se han impreso para este proyecto se pueden dividir en:

### Soporte de las antenas

En este caso, en vez de usar imanes y rodamientos para unir las antenas a la cabeza se usará una pieza modelada en 3D que cree una conexión fija entre los motores y la pieza que sujeta la antena.

Imagen que contiene agua, azul

Descripción generada automáticamente

Dos primeras iteraciones de la pieza (izq.) y pieza que sujeta la antena (der.).

También ha sido necesario modificar la interfaz que une el motor a la antena para adaptarlo a los motores AX-12.

Imagen que contiene tabla, pastel, azul, nieve

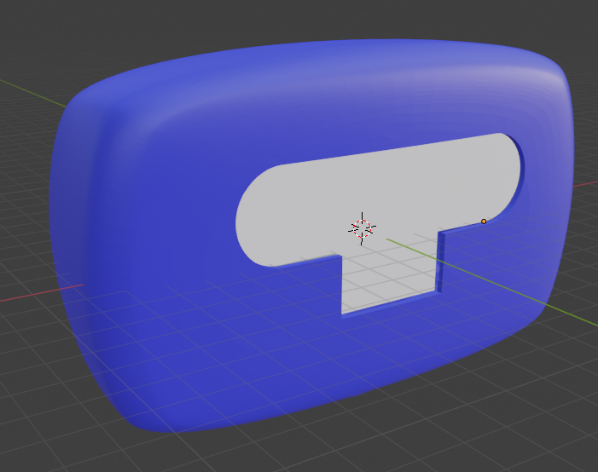
Descripción generada automáticamente

Interfaz motor-antena final.

### Cara

La cara original de Reachy está creada para sujetar dos cámaras que actúan de ojos, los motores de las antenas y un sistema de procesamiento. En este caso se ha usado una sola cámara, por lo que se ha modificado la parte frontal de la cabeza para usar la webcam adecuada.

Diferentes prototipos de la cara de Reachy.



Tras modificar la parte frontal, se ha retocado también el interior para eliminar los soportes para cámaras existentes, y se ha añadido un soporte para la cámara web usada.

Icono

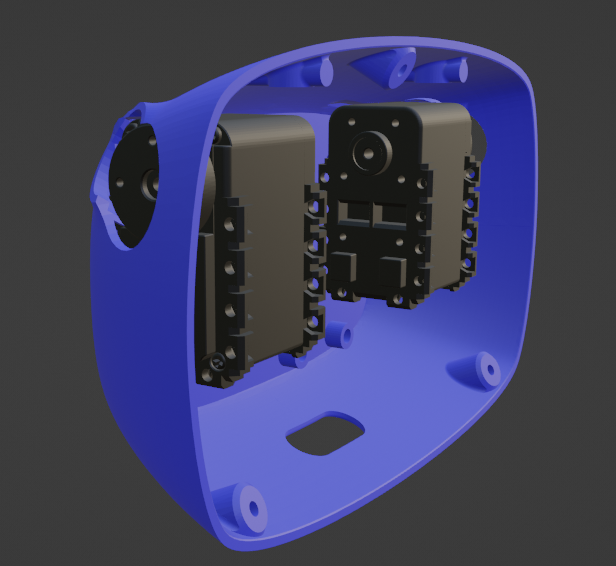
Descripción generada automáticamente

Parte posterior de la cara modificada para el uso de una sola cámara.

Parte delantera de la cabeza de Reachy impresa en 3D



Se ha modificado también la parte trasera de la cabeza para usar los motores Dynamixel AX-12, eliminando los soportes de los motores, pues los Dynamixel AX12 pueden atornillarse directamente a la pieza. Se han modificado a su vez los agujeros de las antenas y los soportes de estas para permitir el movimiento tras haber cambiado la posición de los motores.



Parte trasera de la cabeza.

Imagen que contiene azul, tabla, pastel, corte

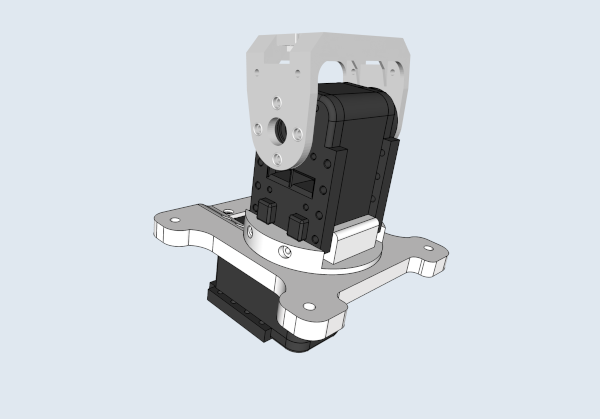
Descripción generada automáticamente

Pieza trasera de la cabeza impresa con motores

### Cuello

El cuello original de Reachy utiliza un mecanismo creado por Pollen robotics llamado Orbita, que usa tres articulaciones y tres motores para dotar a Reachy de un movimiento fluido y expresivo con 3 grados de libertad.

Para simplificar este mecanismo se ha usado un sistema con dos motores y dos articulaciones simples, en la base del cuello se encuentra un motor, que gira de lado a lado, y sobre este se ha montado otro motor que se mueve de arriba abajo y sobre el que se encuentra el resto de la cabeza, permitiendo así que la cabeza de nuestro Reachy simplificado se mueva en dos direcciones., de esta forma es capaz seguir a un objeto que se mueva frente a él.



Orbita joint (izq.) y cuello modificado (der.).

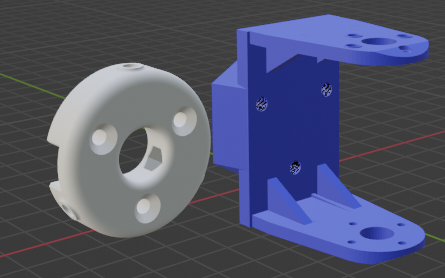
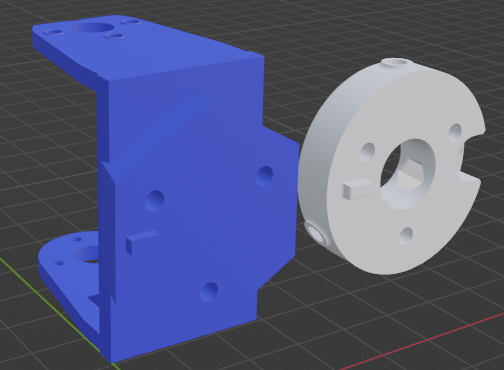
Imagen que contiene interior, tabla, cama, pequeño

Descripción generada automáticamente

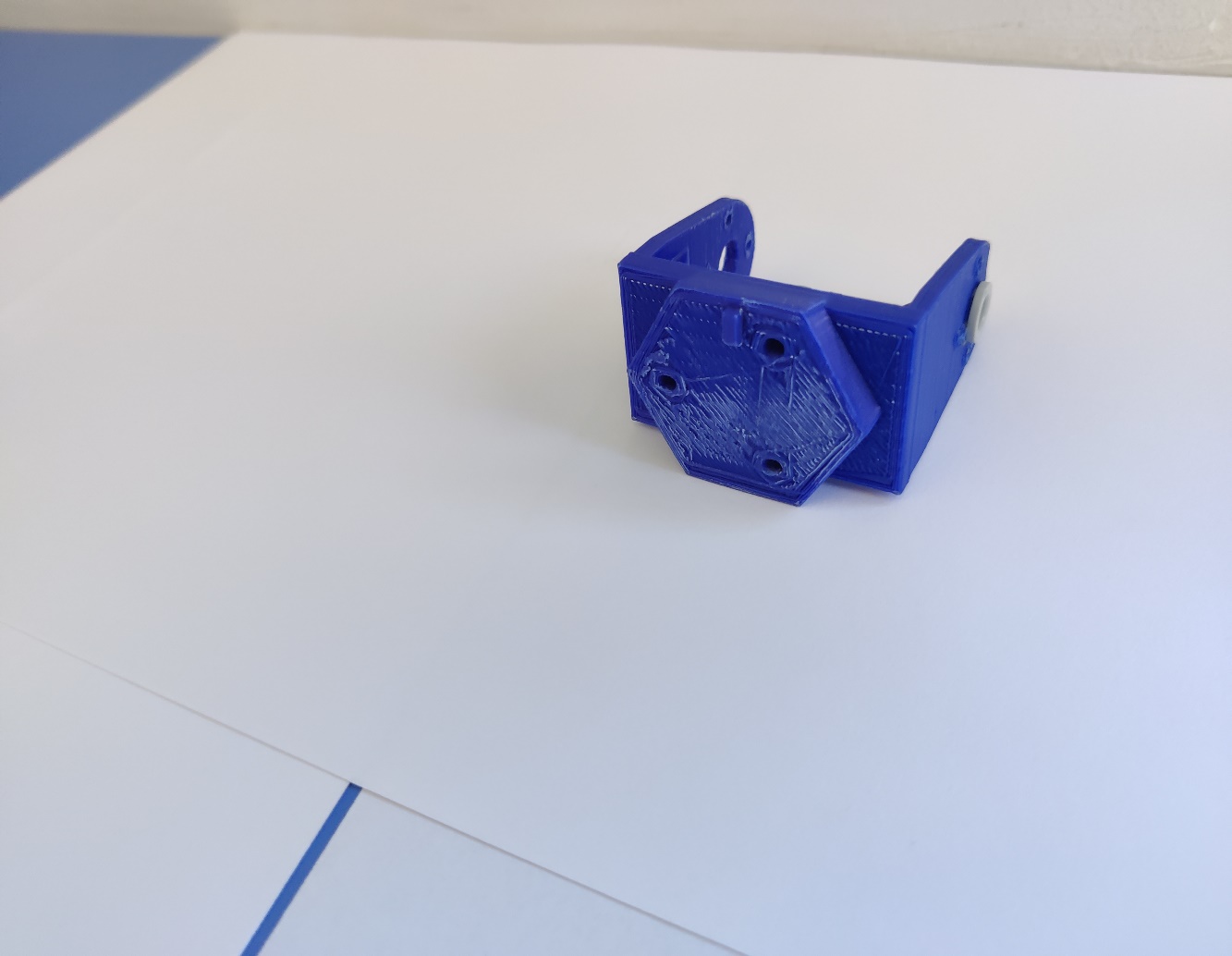
Cuello modificado impreso

También se ha adaptado el cuello que conecta con la cabeza permitiendo así imprimir en una sola pieza la conexión de los motores a la cabeza.

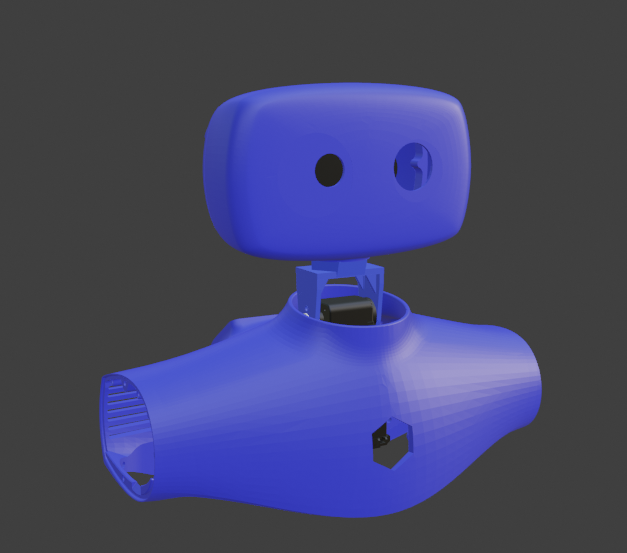
Conexión modificada (azul) y original (blanco).



Conexión de la cabeza y el cuello modificada impresa



Tras diseñar todas las piezas, se puede observar en Blender una simulación del robot Reachy, y una vez impresas todas las piezas, el montaje completo del robot.



Estructura de Reachy completa.

Imagen que contiene interior, tabla, escritorio, computadora

Descripción generada automáticamente

Reachy completo

# Implementación del Reachy

Para realizar la implementación de Reachy se han usado dos topics principales que permiten la comunicación entre los 3 procesos más importantes: el diálogo, la visión y el movimiento. Para publicar en esos topics se usan dos mensajes distintos: ActionMsg y MotorMsg.

El nodo principal es *Reachy\_speech*, que usando DialogFlow y el controlador de sonido ubicado en[*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) interpreta las ordenes de voz, responde si es necesario y publica un mensaje del tipo ActionMsg en el topic */action*. Este mensaje se compone de:

* *int64 mode*, que determina el comportamiento, seguir persona, seguir cubo, o hablar.
* *string action*, que se utiliza para especificar ciertas acciones, como negar o asentir con la cabeza, o pensar antes de hablar.
* *string data*, utilizado para dar información adicional, por ejemplo el color del cubo que se debe seguir.

Para poder detectar objetos y personas se utilizan los nodos *camera\_publisher* y *camera\_subscriber*. El primero toma las imágenes de la cámara y las publica en el topic */usb\_cam/image\_raw,* el segundo se suscribe a este topic y al topic /*action*. Con la información de los dos topics *camera\_subscriber* procesa la imagen conforme a lo que pide el mensaje de action y publica un mensaje MotorMsg en el topic */motor\_msg*. Este mensaje está formado por:

* *int64 x\_sector*, que indica el sector en el eje x en el que se encuentra el objeto buscado.
* *int64 y\_sector*, que indica el sector en el eje y en el que se encuentra el objeto buscado.
* *int64 x*, la coordenada en el eje x.
* *int64 y,* la coordenada en el eje y.
* *string motor\_action*, un campo usando para dar órdenes a los motores fuera del seguimiento. De esta manera Reachy puede asentir, despertarse o alegrarse.

Por último, el nodo *motor\_controller* se suscribe al topic */motor\_msg* y utiliza los servicios *set\_position* y *get\_position* para controlar y calcular la posición de los motores respectivamente. Este nodo lee el mensaje y activa el control de movimiento siguiendo a un objeto o realizando las acciones necesarias dependiendo del campo *motor\_action* del mensaje.

## Manual de usuario

Para poder comprobar el funcionamiento del robot se usarán los launcher de *gb\_dialog Reachy\_launch.launch*, que lanza los nodos que controlan todos los comportamientos de Reachy, y *gb\_dialog\_services\_soundplay.launch*, que permitirá el uso de sonido.

Una vez lanzados los nodos necesarios, los principales comandos usados son:

*Hello*, que funciona como “botón de encendido”, activando a Reachy y moviéndolo a la posición inicial, con ambos motores perpendiculares entre ellos.

*Follow the COLOR cube*, donde *COLOR* puede ser *green, yellow, blue* o *red*, y activa el modo de seguimiento del cubo del color que se especifique.

*Follow me*, que activa el modo de seguimiento de una persona.

*What time is it?,* que hace que Reachy nos diga la hora.

*On your left/right*, que sirven para ayudar a Reachy en caso de que se pierda. Tras usar este comando, Reachy se moverá hacia el lado correspondiente y seguirá buscando a su objetivo.

*Tell me a joke*, que hace que Reachy piense en una broma y la cuente.

*Tell me a fact*, que hace que Reachy cuente un dato interesante.

*Goodbye*, que “apaga” a Reachy, haciendo que mire hacia abajo imitando que duerme.

*I am sad/happy*, que hace que Reachy empatice y lo muestre moviendo las antenas.

Existen también otras frases que permiten interactuar con Reachy. Por ejemplo, es capaz de responder a diferentes preguntas como: *Where are you?*, *How are you?* o *Are you a robot?,* a la que responde afirmando o negando con la cabeza.

# Conclusiones y trabajos futuros

La conclusión que se ha sacado del proyecto es que, a pesar de las dificultades, trabajar con hardware es gratificante. Si bien existen partes complicadas, como que dos motores que se encuentran listados como compatibles con el mismo protocolo de comunicación no funcionen con el mismo controlador porque usan voltajes distintos, vale la pena por ver el código materializarse de forma física.

Este trabajo sería fácilmente expandible. Debido al modularidad del proyecto se podría rediseñar el cuello para usar el mecanismo Orbita comprando un solo motor más, sin ser necesario volver a imprimir la cabeza ni modificar los comportamientos de ciertos nodos. Podría ampliarse el robot fabricando uno o dos brazos robóticos que en conjunción con la cámara y el micrófono permitan coger y dejar objetos donde se especifique, o incluso conectar a Reachy a un asistente personal y convertirlo en una especie de secretario que tome notas y ayude en una oficina.

Elegí este trabajo de fin de grado a pesar de la dificultad por lo mismo que elegí está carrera: la posibilidad de crear un robot desde cero, y a pesar de las dificultades y las concesiones que se han tenido que hacer a lo largo del proyecto, considero que ambas decisiones han sido las correctas.

Este trabajo ha requerido multitud de competencias tanto aprendidas en la carrera como aprendidas por cuenta propia. Han sido necesarios todos los conocimientos de ROS y ROS2 de la asignatura de Arquitectura de computadores para coordinar el comportamiento del robot usando diferentes nodos que tengan un uso claro, así como saber qué versión de ROS usar y cuando cambiar; las habilidades obtenidas en Modelado y simulación de robots y Mecatrónica para poder modificar e imprimir las piezas, sabiendo qué programa usar en cada momento; también ha sido necesaria la asignatura de Visión Artificial para poder procesar las imágenes capturadas por la cámara y detectar los objetivos a seguir. Se han usado también otros muchos conocimientos como los aprendidos en Física para evitar freír los motores o cómo usar una Raspberry, aprendido en Sensores y actuadores.

También he aprendido haciendo este proyecto, que el hardware no es tan fácil como parece, se requiere bastante información para ser capaces de montar un robot que funcione, qué modelos de motor son compatibles con qué controladores, la potencia necesaria para realizar ciertas acciones y cómo estas limitan al resto del robot, o a buscar información y comprobar que no solo sea fiable sino que esté actualizada y funcional con las versiones necesarias de otros

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

# Bibliografía

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. anónimo. (2021, julio 1). Se suspendió la producción del famoso robot Pepper. infobae. <https://www.infobae.com/america/tecno/2021/07/01/se-suspendio-la-produccion-del-famoso-robot-pepper/> [↑](#footnote-ref-2)
3. RdR. (2020, febrero 3). 🥇 Robot Humanoide NAO; características y precio en 2020. REVISTA DE ROBOTS. <https://revistaderobots.com/robots-y-robotica/robot-nao-caracteristicas-y-precio/> [↑](#footnote-ref-3)
4. About: Nao (robot). (s/f). DBpedia. Recuperado el 12 de abril de 2022, de <https://dbpedia.org/page/Nao_(robot)> [↑](#footnote-ref-4)
5. Asistentes virtuales de voz. (2019, octubre 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://github.com/ros-drivers/usb_cam/tree/ros2> [↑](#footnote-ref-8)
9. <https://gitlab.com/Jaco-Assistant/Jaco-Master> [↑](#footnote-ref-9)
10. <https://github.com/aws-robotics/lex-ros2> [↑](#footnote-ref-10)
11. <https://dialogflow.cloud.google.com/> [↑](#footnote-ref-11)
12. <https://github.com/IntelligentRoboticsLabs/gb_dialog> [↑](#footnote-ref-12)
13. <https://www.cstr.ed.ac.uk/projects/festival/> [↑](#footnote-ref-13)
14. <https://github.com/ros-drivers/usb_cam/tree/ros2> [↑](#footnote-ref-14)
15. <http://wiki.ros.org/usb_cam> [↑](#footnote-ref-15)
16. <http://wiki.ros.org/dynamixel> [↑](#footnote-ref-16)
17. <https://cad.onshape.com/documents/5a9e7c88f6e1068df540bf7a/w/a0a08cc5a2a1d3ad8b96785c/e/a661d6a7df21bf59ff5454f5> [↑](#footnote-ref-17)