Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

DESARROLLO DE DISPOSITIVOS DE CONTROL REMOTO CONTROLADOS POR ROBOT CON CAPACIDAD DE INTERACCIÓN HUMANA

Autor: MIGUEL ESCRIBANO GUMIEL

Tutor: JULIO PASTOR MENDOZA

2017

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

DESARROLLO DE DISPOSITIVOS DE CONTROL REMOTO CONTROLADOS POR ROBOT CON CAPACIDAD DE INTERACCIÓN HUMANA

Autor: MIGUEL ESCRIBANO GUMIEL

Director: JULIO PASTOR MENDOZA

Tribunal:

Presidente: ELENA LÓPEZ GUILLÉN

Vocal 1º: MIGUEL ÁNGEL GARCÍA GARRIDO

Vocal 2º: JULIO PASTOR MENDOZA

Calificación: .....

Fecha: .....

A mi familia y amigos...

"1ª. Un robot no hará daño a un ser humano o, por inacción, permitir que un ser humano sufra daño."
"2ª. Un robot debe hacer o realizar las órdenes dadas por los seres humanos, excepto si estas órdenes entrasen en conflicto con la 1ª Ley."
"3ª. Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la 1ª o la 2ª Ley."

Isaac Asimov

Agradecimientos

La inteligencia es la capacidad de adaptarse al cambio.

Stephen Hawking

Tras muchos meses razonando, aprendiendo y diseñando este proyecto, muchas personas son dignas de mencionar y agradecer en este apartado. Ya que sin todos ellos no tengo la menor idea de como se podría haber desarrollado.

En primer lugar lugar, agradecer a la Universidad de Alcalá (UAH) y a todos los docentes que son dignos de llamar "profesores", ya que estas personas han demostrado lo importante que son los alumnos y alumnas de esta Universidad para ellos. Agradecer especialmente al profesor Julio Pastor, tutor de este proyecto, por su colavoración y consejo. Sin estos, este proyecto no tendría sentido alguno. Para completar este párrafo decentemente, he de mencionar a una persona que, a pesar de ser su trabajo, siempre lo ha realizado de manera excelente, por ello agradecer a mi profesora de la academia Maxwel, Victoria, todo lo que he conseguido aprender gracias a ella.

En segundo lugar, agradecer a mis padres Ángel y Paloma por su completo apoyo durante toda la carrera, a pesar de los momentos de crisis y dudas que han invadido nuestras más temibles pesadillas. Al final es posible ver la luz del camino, y es cuando se demuestra quién ha confiado en uno durante todo el camino, solo me queda deciros muchísimas gracias por todo. No puedo continuar este párrafo sin agradecer también el apoyo y colaboración de mi hermana Laura, mis abuelos paternos Ángel y Francisca que siempre han creído en mí. También agradecer especialmente la gran colaboración de mis abuelos maternos Victoria y Emilio, no solo por su eterno apoyo, sino también por permitirme compartir una gran parte de mi vida universitaria con ellos. Agradecer por último a todos mis tíos y primos, siempre han estado ahí cuando lo he necesitado.

En tercer lugar, agradecer a mis compañeros de carrera y sobre todo aquellos que han compartido conmigo todos los trabajos realizados durante esta aventura. Muchas gracias Sergio, José, Miriam, Ángel, Miguel Ángel, Víctor, Guillermo V., Carlos V., Adrián de la Torre, Borja, Guillermo P. y Adrián R.; ha sido un placer y un honor haber compartido con vosotros esta gran experiencia y amistad.

También quiero agradecer a mis amigos de siempre Fernando, Nicolás, Julio, José Ramón, Celia, Rubén P. y Rubén M.; que siempre han ofrecido su apoyo y consejo en los peores momentos. Agradecer especialmente a Diego, al que considero mucho más que un simple amigo, le considero alguien de mi familia (de hecho este agradecimiento debería estar unos párrafos más arriba), muchas gracias por permitirme compartir contigo todos los problemas que nos han ido surgiendo durante todos estos

años. No puedo acabar este agradecimiento a mis amigos sin mencionar a una persona, que aunque no haya compartido conmigo nada de mi vida universitaria, siempre le mantendré en mi memoria, Daniel.

Por último lugar, y no por ello menos importe, agradecer a la empresa Juguetrónica SL por haberme permitido realizar el proyecto proporcionando el material y las herramientas necesarias para obtener este objetivo. Quiero destacar la ayuda recibida por mis compañeros y compañeras de trabajo, que siempre me han apoyado y ayudado lo mejor que han podido, por ello muchas gracias Inés, Álvaro, Victoria, Joan, Sara, Yaiza, Manuel, Gonzalo y Rubén. Muchas gracias a todos por demostrar que ser compañeros de trabajo y amigos, no es una desventaja, sino que es un privilegio.

Para finalizar estos agradecimientos quien mejor que Carlos, maestro, compañero y amigo. Debo agradecerte todo lo que me has enseñado, la paciencia que has tenido y el apoyo incondicional durante este trabajo. Tu consejo y conocimientos han resultado ser los pilares más importantes para lograr estos objetivos, y si debo agradecer a alguien el poder estar escribiendo estas líneas, es a ti ¡Muchas gracias por todo!¡Amigo!

Índice general

Índice de figuras

Índice de tablas

Resumen

El proyecto **DESARROLLO DE DISPOSITIVOS DE CONTROL REMOTO CONTROLADOS POR ROBOT CON CAPACIDAD DE INTERACCIÓN HUMANA** realizado por **MIGUEL ESCRIBANO GUMIEL**, realiza el estudio y desarrollo de una aplicación sobre el robot diseñado y fabricado por Softbank Robotics, Pepper. Debido a la falta de elemento hardware del robot para poder integrarlo útilmente en un entorno domestico, este proyecto trata de facilitar una solución desarrollando un dispositivo, utilizando para ello una Raspberry Pi Zero W, que al estar conectado por red inalámbrica con el robot, permita agregar al robot emitir y recibir comandos mediante IR. Todo esto permitirá al usuario controlar los aparatos, tales como el televisor, la cadena de música y el reproductor de DVD, utilizando al robot Pepper.

Palabras clave: Robot Pepper, Raspberry Pi Zero W, IoT, protocolos de comunicación websocket, desarrollo de aplicacciones.

Abstract

That project developed by MIGUEL ESCRIBANO GUMIEL, carries out the study and development of an application on the robot designed and manufactured by Softbank Robotics, Pepper. Due to the lack of hardaware elements of this robot to be able to integrate it usefully to a domestic enviroment, this project tries to find a solution developing a device, using a Raspberry Pi Zero W. The hardware expansion that will be carried out on the robot consists in the development and control of a remote control emitter and receiver of infrared, which in a home environment will allow to control all devices that are managed by this communication method. For example: the television, the stereo and the DVD player.

Keywords: Pepper the robot, Raspberry Pi Zero W, IoT, WebSocket communication protocol, applications development.

Parte I

RESUMEN EXTENDIDO

Resumen extendido

El proyecto **DESARROLLO DE DISPOSITIVOS DE CONTROL REMOTO CONTROLADOS POR ROBOT CON CAPACIDAD DE INTERACCIÓN HUMANA** realizado por **MIGUEL ESCRIBANO GUMIEL**, tiene como objetivo el estudio y programación de una aplicación sobre el robot desarrollado por Softbank Robotics, Pepper[1], el cual permita al usuario utilizar las funciones del robot en un entorno doméstico. Debido a la falta de módulos hardware que posee el robot, es necesario un método que, sin dañar el robot, permita expandir sus funcionalidades y así conseguir extender las funciones y elementos del robot. En la figura 2, se puede apreciar el sistema desarrollado en este proyecto.

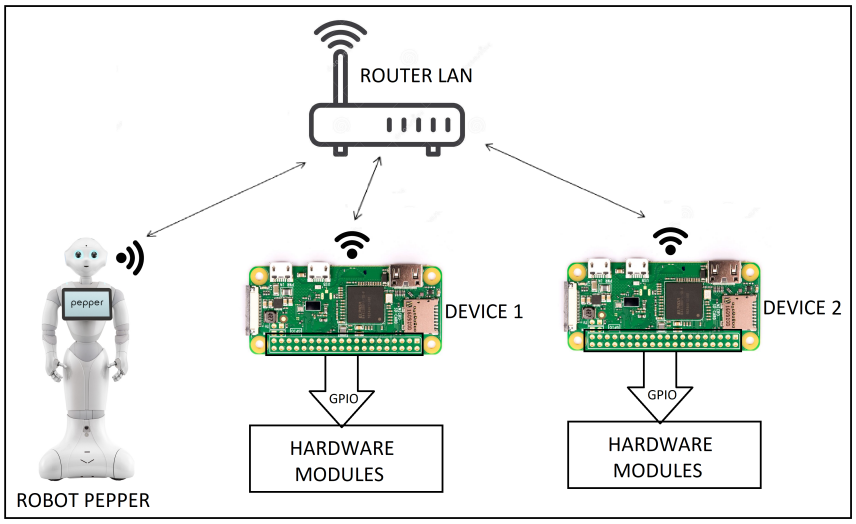


Figura 2: Diagrama general del proyecto.

Para ello ha sido necesario el desarrollo de un método de comunicación entre el robot y un dispositivo basado en una placa Raspberry Pi Zero W, que mediante conexión websocket, permite comunicar al robot con el dispositivo desarrollado. El cual esta diseñado para que el robot se comporte como cliente en la conexión, y la placa Raspberry Pi Zero W pueda comportarse como servidor. Esta configuración permite realizar al robot consultas de baja duración que no pondrán en peligro la comunicación entre ambos extremos.

Por ello este proyecto está dividido en tres partes, una correspondiente al estudio y documentación de las características técnicas y al desarrollo de los módulos hardware de expansión. Otra parte corresponde al estudio de las herramientas software empleadas para el desarrollo de las aplicaciones. Por último se explicara como se ha desarrollado todo el proyecto, tanto la parte hardware como software.

Estudio de las características del robot y de lo módulos hardware adicionales

El robot Pepper, es un robot desarrollado por las empresas Aldebaran y Softbank Robotics[1], el cual se muestra en la figura 3, el cual posee unas características tales como la capacidad de comunicarse tanto por voz, permitiendo incluso mantener una conversación, como mediante la ayuda de la *tablet* en su pecho. Este robot también tiene una serie de sensores táctiles en manos y cabeza, similar a su hermano pequeño NAO, así como sensores de distancia por ultrasonidos y por lasers.

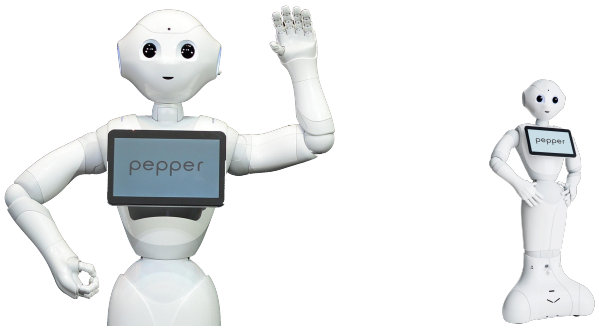


Figura 3: Robot Pepper.[1]

Estos sensores le permiten realizar los movimientos de forma segura, además de ser utilizados también a la hora de desplazarse. También posee una serie de cámaras con las que analiza el entorno, dos cámaras 2D y una cámara 3D, giroscópios, brújula, microfonos direccionales y altavoces. Pepper es capaz de conectarse a la red, aunque no posee otro tipo de comunicaciones como bluetooth o ZigBee.

Para poder realizar el desarrollo del dispositivo de expansión de hardware, se ha decidido realizarlo con una placa Raspberry Pi Zero W[2], mostrada en la figura 4, que debido a su bajo coste, reducido tamaño y gran potencia, permite realizar no solo las funciones que se necesiten para expandir el hardware, sino que también es capaz realizar las gestiones oportunas como servidor en la comunicación. El acceso a sus GPIO posibilitan la incorporación de un emisor IR, un receptor IR, una pantalla LCD y una serie de módulos para gestionar la energía del dispositivo.

Figura 4: Raspberry Pi modelo Zero W.[2]

Estudio de las herramientas software empleadas

Para la realización y desarrollo de este proyecto, es necesario conocer las herramientas de las que se dispone para obtener los resultados requeridos. Por ello, en este proyecto se deben conocer tres herramientas software imprescindibles para el correcto funcionamiento del sistema.

Como se ha especificado anteriormente, para poder manejar y desarrollar aplicaciones sobre el robot Pepper y su *tablet*, es necesario conocer y dominar el framework de NAOqi, el cual mediante su SDK, es posible controlar todos los aspectos del robot, como su *tablet*, los diálogos necesarios, y el envío y gestión de información referente a la aplicación.

También es preciso conocer el sistema operativo Raspbian, el cual es un SO capaz de controlar la placa Raspberri Pi Zero W permitiendo la gestión de los GPIO.

Por último, este proyecto se basa en la comunicación entre Pepper y la Raspberry Pi Zero W, por lo que la manera elegida para su conexión es mediante Tornado Websocket, el cual es un framework capaz de establecer conexiones entre cliente y servidor de manera continuada, obteniendo una gestión completa de la conexión. Hay que añadir, que el sistema operativo de Pepper basado en Linux, no es capaz de realizar tareas de superusuario, por lo tanto, es necesario utilizar las librerías de Tornado Websocket, ya que son las que el robot trae por defecto instaladas.

Desarrollo de las aplicaciones y dispositivos necesarios

El desarrollo de la aplicación principal del robot, ha consistido en crear un nuevo servicio en el robot, el cuan mediante por una serie de métodos, es capaz de realizar las tareas oportunas. Al realizar la aplicación por este proceso, es posible acceder a todos los métodos desde cualquier equipo conectado y sincronizado con el robot mediante NAOqi. Por ello es más cómodo y seguro realizar tareas desde la tablet y el script de diálogos.

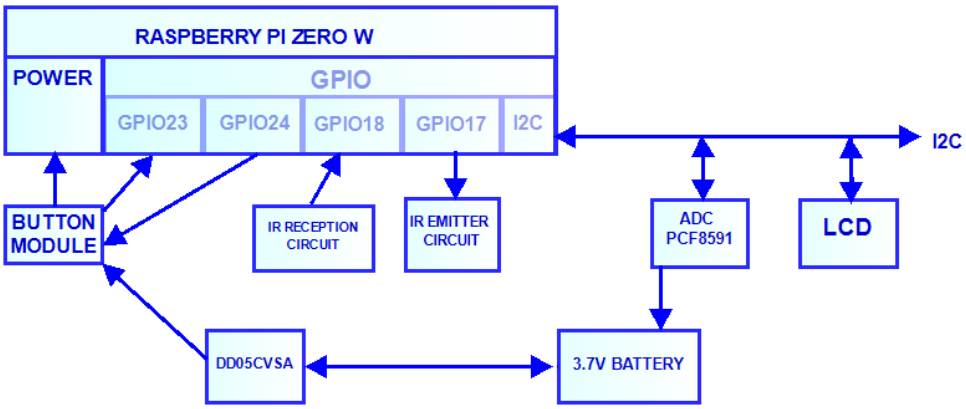


Figura 5: Diagrama de bloques del dispositivo desarrollado.

En cuanto al montaje del módulo de expansión hardware del robot, cuyo principal componente es el mini ordenador desarrollado por la fundación Raspberry Pi, el cual se muestra su diagrama general en

la figura 5. El dispositivo posee una pantalla LCD que permite al usuario conocer el estado de la batería y además obtener información de su conexión a la red. También cuenta con un módulo que se encarga de realizar las lecturas de voltaje de las baterías de ión-litio. En cuanto a la gestión del arranque y el apagado del dispositivo, se ha incluido un módulo que realiza estas tareas de forma segura. Por último, se han desarrollado los circuitos necesarios para construir un emisor y un receptor de IR.

Parte II

LISTA DE ACRÓNIMOS

$\frac{1}{2}$

Lista de acrónimos

Abreviatura	Descripción
3D	Tridimensional (Three-dimensional space)
AC	Corriente Alterna (Alternating Current)
ADC	Convertidor Analógico Digital (Analog-to-Digital Converter)
API	Interfaz de Programación de Aplicaciones (Application Programming Interface)
ASR	Reconocimiento Automático de Voz (Automatic Speech Recognition)
CPU	Unidad Central de Procesamiento (Central Processing Unit)
CSI	Interfaz Serie para Cámaras (Camera Serial Interface)
DAC	Convertidor Digital Analógico (Digital-to-Analog Converter)
DC	Corriente Continua (Direct Current)
DVD	Disco de Vídeo Digital (Digital Video Disc)
GPIO	Entrada/Salida de Propósito General (General Purpose Input/Output)
HAT	Hardware Conectado en la Parte Superior (Hardware Attached on Top)
HDMI	Interfaz Multimedia de Alta Definición (High-Definition Multimedia Interface)
HTML	Lenguaje de Marcas de Hipertexto (HyperText Markup Language)
I2C	Circuito Interintegrado (Inter-Integrated Circuit)
IMU	Unidad de Medición Inercial (Inertial Measurement Unit)
IoT	Internet de las Cosas (Internet of Things)
IP	Protocolo de Internet (Internet Protocol)
IR	Radiación Infrarroja (Infrared Radiation)
ISP	Programación en el Sistema (In-System Programming)
JTAG	Grupo de Acción de Prueba Conjunta (Joint Test Action Group)
LAN	Red de Área Local (Local Area Network)
LCD	Pantalla de Cristal Líquido (Liquid Crystal Display)
LED	Diodo Emisor de Luz (Light-Emitting Diode)
LIRC	Control Remoto por Infrarrojos de Linux (Linux Infrared Remote Control)
NEC	Nippon Electric Company
PC	Ordenador Personal (Personal Computer)
PCM	Modulación por Impulsos Codificados (Pulse Code Modulation)
PDM	Modulación de señal por Densidad de Impulsos (Pulse Density Modulation)
PWM	Modulación por Ancho de Pulsos (Pulse-Width Modulation)
RAM	Memoria de Acceso Aleatorio (Random Access Memory)
ROS	Sistema Operativo Robótico (Robot Operating System)
SD	Seguridad Digital (Secure Digital)
SDIO	Entrada y Salida con Seguridad Digital (Secure Digital Input Output)
SDK	Kit de Desarrollo de Software (Software Development Kit)
TCP	Protocolo de Control de Transmisión (Transmission Control Protocol)

USB	Bus Universal en Serie (Universal Serial Bus)

Parte III

LISTA DE SÍMBOLOS

Lista de símbolos

Símbolo	Descripción
Ω	Unidad de resistencia eléctrica en Ohmios.
m	Unidad de distancia en metros.
Hz	Unidad de frecuencia en Hercios.
B	Unidad de capacidad de almacenamiento digital en Bytes.
g	Unidad de medida de aceleración referenciada a la aceleración grabitacional terrestre.
A	Unidad de flujo de corriente eléctrica en Amperios.
W	Unidad de potencia eléctrica en Wattios.
V	Unidad de diferencia de potencial en Voltios.

Parte IV

MEMORIA

Capítulo 1

Introducción

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Alan Turing, Computing machinery and intelligence

1.1 Presentación

Actualmente no existen muchos robots capaces de interactuar con los seres humanos. Sin ninguna duda, el que posee un diseño más sofisticado y humano es el robot diseñado por Aldebaran, una empresa de Softbank Robotics. Este robot, no solo cuenta con un diseño muy apacible, sino que también cuenta con unas dotes de comunicación, movimiento y oratoria únicas en el mercado. Por ello le hace un robot perfecto para servir tanto en comercios como en el hogar, permitiendole ser una herramienta perfecta que se integra en el mundo del Internet de las Cosas (IoT).

En este proyecto, se desarrollará un sistema de domótica que utilizará al robot como intermediario entre los elementos IoT de un hogar corriente y los usuarios. El robot, mediante conexión a una red local, podrá manejar todos los electrodomésticos y aparatos por medio de un emisor de infrarrojos.

Para controlar todos los aparatos por control remoto como la televisión, el reproductor de DVD, el aire acondicionado, etc; será necesario disañar un dispositivo que pueda estar conectado en a la red y así realizar las funciones de servidor. Permitiendo al robot Pepper consultar y enviar información al dispositivo sin la necesidad de realizar conexiones de larga duración, haciendo la aplicación más estable.

Una vez diseñado el prototipo de los dispositivos, se realizará el desarrollo software en el robot, permitiendo controlar los aparatos por médio de comandos por voz y utilizando en su defecto la tablet incorporada en el robot.

Por lo tanto, en este proyecto se pretende demostrar los conocimientos acerca del estudio del funcionamiento de redes de comunicación, que permitan interconectar sistemas robotizados con el framework de NAOqi y cualquier dispositivo que se desarrolle para el mundo del IoT.

1.1.1 Motivación y objetivos

Debido a su elevado precio de venta al publico, y la necesidad de desarrollar una aplicación para poder obtener algún servicio del robot humanoide Pepper, es necesario en el mercado el desarrollo de algún método de conexión con otros dispositivos que permitan realizar ampliaciones de hardware sobre el robot, como por ejemplo en el caso de este proyecto, la utilización de un módulo de emisión y recepción de comandos por control remoto. La mayor de la motivaciones para realizar este proyecto, es conseguir un método para realizar dichas ampliaciones sobre el robot, que permitan ajustarse a las necesidades de los diferentes usuarios, sin que el usuario pierda la garantía sobre este producto, permitiendo ampliar sus sensores sin la necesidad de abrir físicamente al robot.

Para el desarrollo de una aplicación que permita al robot conectarse con un entorno IoT, es necesario utilizar alguna de las tecnologías desarrolladas para la conexión de objetos en Internet, que garantice la conectividad, velocidad y seguridad necesarias para realizar una conectividad cómoda para el usuario.

El desarrollo de la aplicación sobre el robot, no solo debe garantizar una buena conectividad con el entorno, sino que es necesario que realice correctamente las gestiones sobre los elementos del robot, garantizando al usuario una aplicación suficientemente estable y fácil de utilizar.

Por otro lado, es necesario desarrollar un dispositivo lo suficientemente potente para que consiga la conectividad necesaria con el robot en una red local, y además, permita la gestión de sus puertos GPIO, los cuales serán los encargados de realizar las funciones que se desean añadir al robot.

Para completar este proyecto, es necesario dotar al dispositivo de la posibilidad de ser controlado desde cualquier otro PC conectado a la misma red local, permitiendo así al usuario la posibilidad de utilizar el dispositivo sin la necesidad de poseer un robot Pepper.

1.1.2 Organización de la memoria

Para poder cumplir los objetivos descritos anteriormente, se va a realizar en primer lugar un estudio completo del hardware y software empleados y posteriormente se describirán los desarrollos, tanto hardware como software, utilizados para poder cumplir los objetivos anteriores.

Esta memoria está organizada en cinco capitulos, los cuales corresponden a la introducción del proyecto,al estudio del hardware utilizado, estudio del software utilizado, desarrollo del proyecto y para finalizar una conclusion.

Existen dos partes diferenciadas en este proyecto, una es todo lo correspondiente al estudio del hardware y desarrollo software que permita realizar aplicaciones sobre el robot.

Por otro lado, se encuentra el desarrollo hardware y software de un módulo emisor/receptor de infrarrojos que será el encargado de recibir órdenes del robot y controlar los diferentes aparatos por medio de control remoto.

Capítulo 2

Descripción de la plataforma robótica y los módulos hardware adicionales

2.1 Robot humanoide Pepper

El robot humanoide Pepper es un robot desarrollado por Aldebaran Softbank [1]. Pepper supone un gran paso evolutivo hacia la inserción de los robots en entornos comerciales, ya que actualmente la mayoría de los robots autónomos no se encuentran disponibles para su venta.

Este robot presenta un diseño muy cuidado, se encuentra enfocado en la interacción con las personas, este detalle se puede apreciar en la estatura del robot, que es de 120cm, que favorece mucho la iteración sobre todo teniendo en cuenta si lo comparamos con NAO, el otro robot humanoide desarrollado por la misma compañía, que tiene una estatura de 58cm. Además cuenta con una tablet en su pecho que puede servir de apoyo durante la iteración.

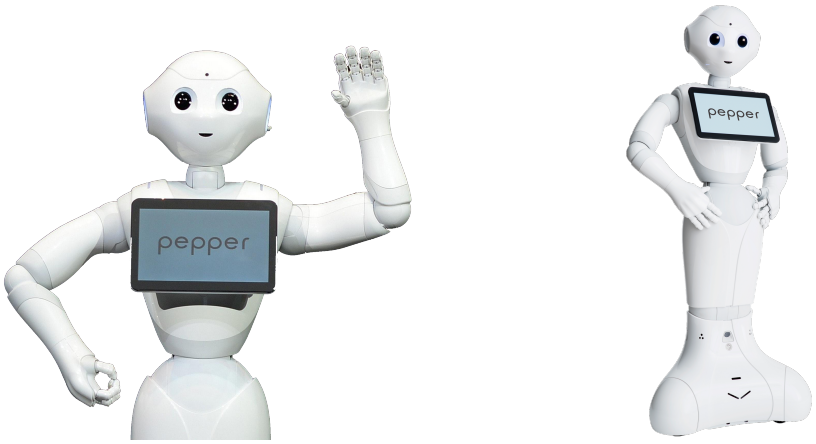


Figura 2.1: Robot Pepper.[1]

2.1.1 Características Técnicas.

Las características técnicas de este robot son resumidas a continuación, en base a la documentación oficial proporcionada por el fabricante [1].

2.1.1.1 Dimensiones

A continuación se mostrarán las dimensiones del robot con las extremidades tanto en posición reposo como extendida en cada uno de los ejes.

Altura (mm)	Ancho (mm)	Profundidad (mm)
1210	480	425

Tabla 2.1: Dimensiones de Pepper con las extremidades en posición reposo.[1]

Altura (mm)	Ancho (mm)	Profundidad (mm)
1355	1196.9	648.2

Tabla 2.2: Dimensiones de Pepper con las extremidades extendidas.[1]

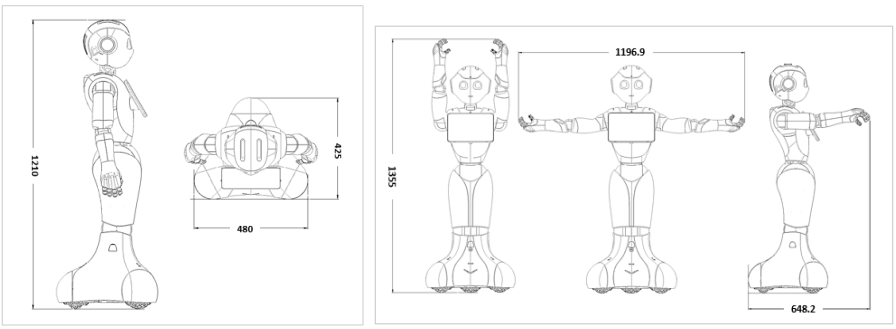


Figura 2.2: Dimensiones de Pepper.[1]

2.1.1.2 CPU

EL ordenador del humanoide se encuentra ubicado dentro de la cabeza y cuenta con las especificaciones mostradas en la tabla 2.3.

Procesador	Atom E3845
CPU	Quad core
Frecuencia CPU	1.91 GHz
Memoria RAM	4 GB DDR3

Tabla 2.3: Características de la CPU.[1]

2.1.1.3 Energía

Pepper cuenta con una batería ubicada en la base del robot y que ofrece una autonomía de entre 10 y 12 horas dependiendo del tipo de uso. Esta gran autonomía permite al robot soportar una jornada de trabajo. En la tabla 2.4 se muestran especificaciones de la batería.

Voltaje de entrada del cargador	240 v AC
Voltaje de salida del cargador	29.2 V DC - 8 A
Tipo de batería	ION-LITIO
Voltaje nominal de carga	26.46 V
Voltaje máximo de carga	29.4 V
Corriente máxima de carga	8 A
Capacidad típica	29.5 Ah
Energía	795 Wh
Corriente consumida en modo activo	Máx. 3.5 mA
Corriente consumida en modo reposo	Máx. 850 μA
Corriente consumida en modo apagado	Máx. 50 μA

Tabla 2.4: Características de la batería.[1]

Existen dos formas de cargar el robot, mediante un cargador o mediante la base de carga desarrollada por Aldebaran-Softbank (figura: 2.3). Esta base de carga es detectada por el robot, lo que permite localizarla para desplazarse hasta ella.

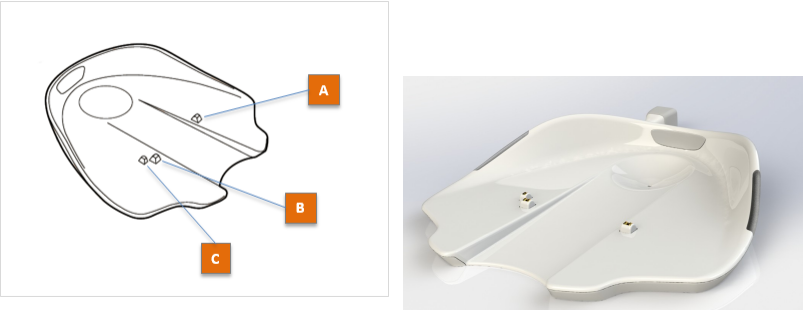


Figura 2.3: Base de Carga.[1]

2.1.1.4 Conectividad

En cuanto al apartado de conectividad solo cuenta con conectividad Ethernet y WiFi (especificaciones en la tabla: 2.5).

Ethernet	1xRJ45 - 10/100/1000 base T
WiFi	IEEE 802.11 a/b/g/n

Tabla 2.5: Características de los sistemas de conectividad.[1]

A pesar de que el robot cuenta con una conectividad que le permite acceder a la red, no es un sistema completo, ya que faltan módulos que permitan al robot conectarse con otros dispositivos mediante Bluetooth o Zigbee, tecnologías muy usadas sobretodo en posicionamiento, transmisión de audio y transmisión de datos.

2.1.1.5 Tablet del pecho

Una de los mejores apoyos de la comunicación de Pepper con los usuarios es, a parte de la voz y los gestos, la tablet que está colocada en su pecho. Según la documentación proporcionada por el fabricante

[1], esta tablet posee las características técnicas, correspondientes a la versión hardware V1.8a del robot, reflejadas en la tabla 2.6.

Dimensiones	246 x 175 x 14.5 mm
CPU	1.3 GHz quad-core ARM Cortex-A7 Cache 512 KB L2 Wireless radio technologies (Wi-Fi) 1.6G pixel/sec @416MHz
DDR3 SDRAM	1GB (512MB * 2)
DDR3 SDRAM	1GB (512MB * 2)
Memoria Flash	32GB (eMMC)
Display	Tipo: IPS Resolución real: 1280 * 800 Color: 24bit true color
Panel táctil	capacidad Multi-touch (5 puntos)
WiFi	802.11 a/b/g/n

Tabla 2.6: Características generales de la tablet en la versión 1.8a.[1]

Esta tablet, posee un sistema operativo Android, aunque no se puede instalar ninguna aplicación ni usarse como tal, ya que la tablet está programada para mostrar un aplicación web, que se comunica con NAOqi por JavaScript.

2.1.1.6 Cámaras

Pepper cuenta con dos cámaras 2D, una mirando al frente y otra apuntando hacia abajo para poder evaluar el terreno (figura:). También cuenta con una cámara 3D (de profundidad), concretamente tiene incorporado la ASUS Xtion 3D (ver figura:), que se encuentra ubicada dentro su ojo izquierdo.

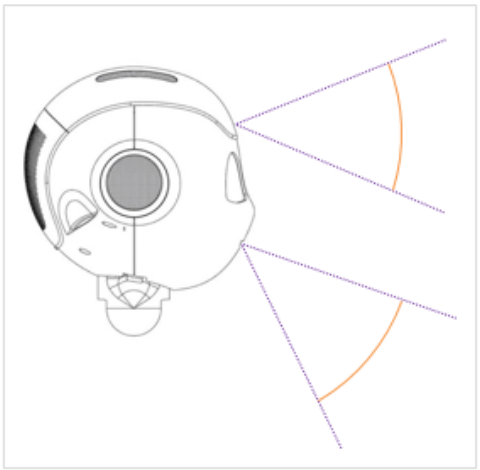


Figura 2.4: Localización de las cámaras.[1]

Cámara 2D superior	2592x1944
Cámara 2D inferior	2592x1944
Cámara 3D	1280x1024

Tabla 2.7: Resoluciones máximas de cada una de las cámaras.[1]

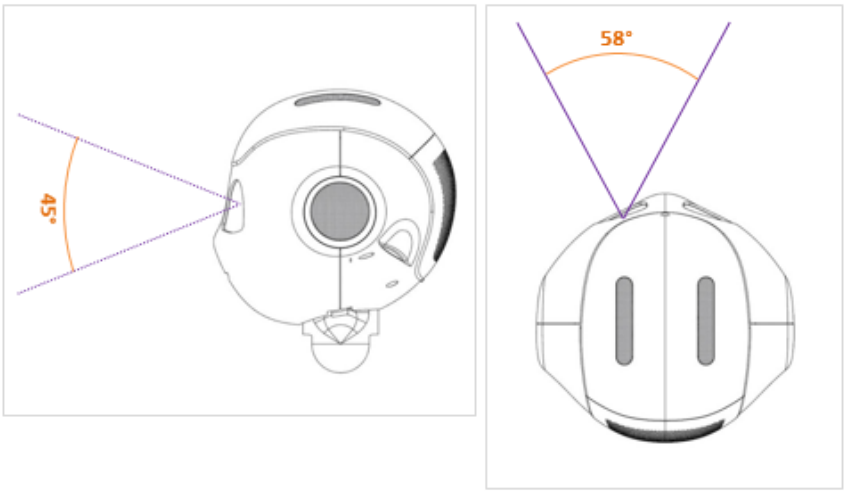


Figura 2.5: Localización y ángulos del sensor 3D.[1]

2.1.2 Sensores

2.1.2.1 Unidad Inercial

Pepper cuenta con una IMU (Unidad de Medición Inercial) constituida por giróscopos en los tres ejes con una velocidad angular de aproximadamente 500°/s y por acelerómetros, también en los tres ejes, con una aceleración de unos 2g. Con estos sensores es capaz de orientarse y navegar por un determinado espacio.

2.1.2.2 Sensores de distancia por láser

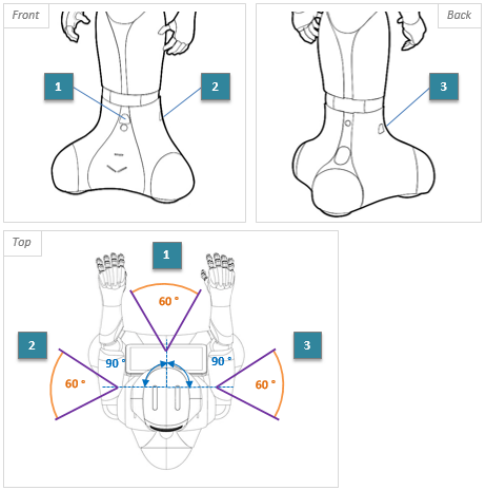
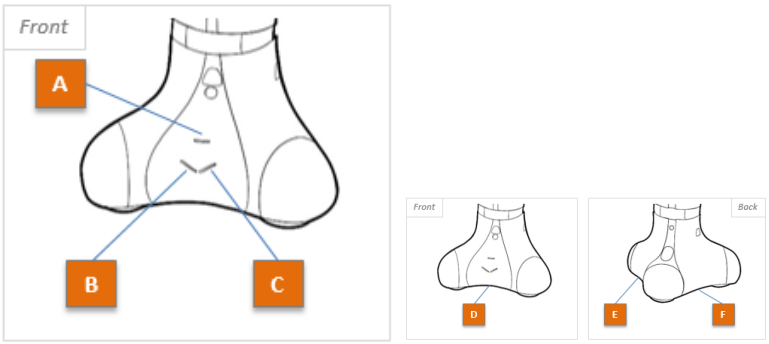


Figura 2.6: Sensores de distancia por láser.[1]

Este robot dispone de una serie de medidores de distancia por láser, lo que le permite detectar objetos a menos de 7m de distancia. Una ventaja de este tipo de sensores es que permite saber exactamente donde se encuentra el objeto, a diferencia de los sensores de distancio por ultrasonidos. Aunque como contra partida, estos sensores no son capaces de detectar superficies translúcidas.

Estos elementos emiten una serie de rayos láser de una longitud de onda de 808 nm, con una frecuencia de muestreo de 6.25 Hz por laser.



(a) Sensores de análisis de terreno vertical

(b) Sensores de análisis de terreno horizontal

Figura 2.7: Sensores de análisis de terreno.[1]

Tal como se muestra en la Figura 2.6, se puede observar los ángulos de apertura de los sensores y donde se encuentran.

Pese a la sofisticación del robot, a este le es imposible desplazarse por superficies que no sean extremadamente lisas. Por ello dispone de una serie de sensores láser que permiten analizar el terreno, como por ejemplo detectar desniveles. En las Figuras 2.7a y 2.7b se muestran la localización de dichos elementos.

2.1.2.3 Sensores de distancia por ultrasonidos

Como complemento de los sensores por láser, el robot dispone de dos sensores de distancia por ultrasonidos o sonars. El motivo por en cual dispone de estos sensores es el de detectar objetos translúcidos los cuales no so detectados por los láser.

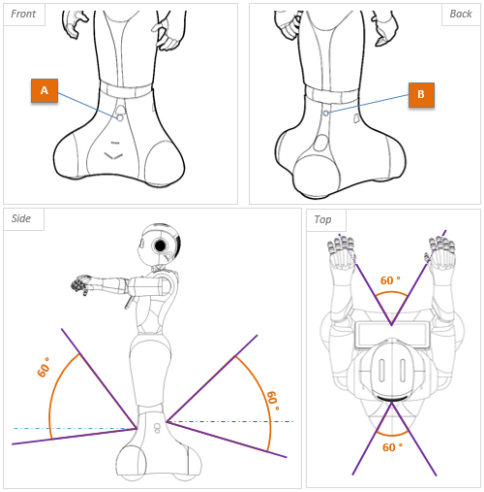


Figura 2.8: Sensores de distancia por ultrasonidos.[1]

Estos sensores poseen dos desventajas frente a los sensores láser, siendo la primera la imposibilidad de detectar elementos que absorban las ondas sonoras. La segunda desventaja consiste en que no es posible detectar la posición del elemento detectado. Esto significa que si se detecta un objeto dentro del ángulo de detección, solo será posible conocer la distancia a la que se encuentra.

La señal ultrasónica es de 42 kHz con un alcance máximo de 5m, y una potencia de sensibilidad de -86dB. Se pueden observar los ángulos y posición de estos sensores en la Figura 2.8.

2.2 Raspberry pi Zero W

La Raspberry Pi Zero W, es el nuevo ordenador de placa reducida desarrolado por la fundación Raspberry Pi[2], el cual salió a la venta en marzo de 2017 a un precio de 11 €. El objetivo principal de la fundación Raspberry Pi ha sido el desarrollar ordenadores de tamaño y coste reducido para acercar la robótica y la informática al mundo educativo, desarrollado una gama de productos perfecta para para la creación, tanto a nivel particular como profesional, de sistemas electrónicos y aplicaciones pudiendo sustituir a plataformas de microcontroladores, como Arduino, en muchos de los proyectos de electrónica que han sido desarrollados por la comunidad.

Figura 2.9: Raspberry Pi modelo Zero W.[2]

Al contrario de la Raspberry Pi Zero, este modelo W incluye las comunicaciones inalámbricas de WiFi y BlueTooth, por lo que resulta una mejora muy considerable teniendo en cuenta el gran potencial que resulta de trabajar con un ordenador, con un bajo coste y un reducido tamaño, conectado a internet. Todas estas características y mejoras respecto a su antecesor, convierte a este ordenado en una herramienta perfecta para desarrollar aplicaciones para el mundo del IoT.

2.2.1 Características Técnicas.

Las características técnicas de este modelo se muestran a continuación[2]:

- 1GHz, single-core CPU
- 512MB RAM
- Puerto Mini-HDMI
- Puerto Micro-USB On-The-Go
- Micro-USB power
- HAT-compatible 40-pin header

- Composite video y reset headers
- Conector de cámara CSI
- 802.11n wireless LAN
- Bluetooth 4.0

Una de las principales características de los ordenadores de placa reducida de Raspberry, es la posibilidad de manejar una serie de GPIO, entre los que se encuentran pines para establecer comunicaciones serie como I2C, serie asíncrona e ISP. También se incluyen pines con salida PWM y otras funciones como se muestra en la figura 2.10, como JTAG, PCM y SDIO.

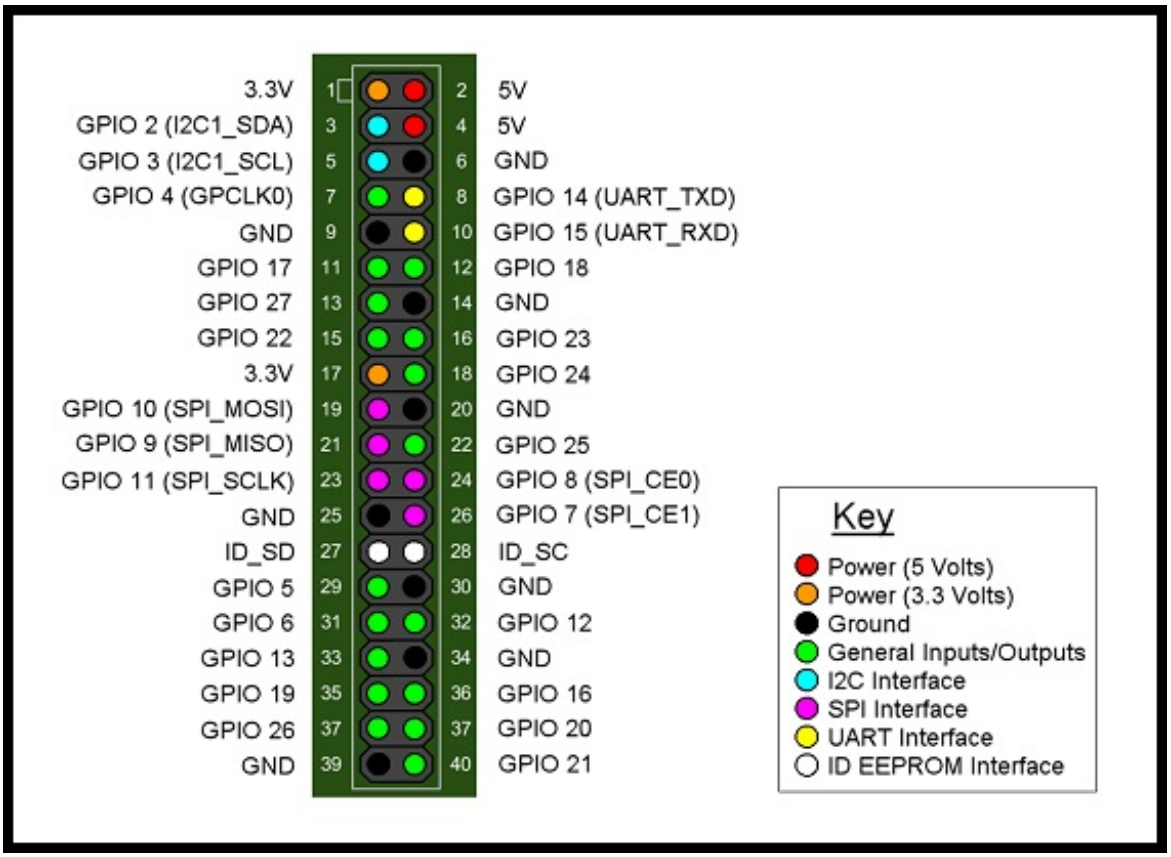


Figura 2.10: Configuraciones de los pines incluidos en el modelo Zero W.[3]

Este modelo de Raspberry posee, a parte de los pines GPIO, dos conectores micro USB, uno de alimentación y otro de datos, una salida mini HDMI, lector de tarjetas micro SD, y por último el conector de la cámara CSI.

2.2.2 Hardware extra

Debido a que faltan elementos, es necesario añadir unos módulos para suplir las necesidades del usuario. Dado que este modelo no posee pines para realizar tareas de gestión de energía, ni posee un sistema de arranque, es necesario añadir una serie de módulos que permitan realizar todas estas tareas.

2.2.2.1 Bases del funcionamiento de la comunicación por infrarrojos

La comunicación por control remoto esta basada en la transmisión de datos en formato binario, por medio de emisión y recepción de pulsos de luz infrarroja. La luz infrarroja[10] es el espectro de las ondas electromagnéticas que corresponde a unas longitudes de onda entre 10^{-3}m y $7,8 \times 10^{-7}$m, en un rango de frecuencias de 3×10^{11}Hz a $3,84 \times 10^{14}$Hz. Al ser una comunicación basada en transmisión de luz, no interfiere con señales de radio electromagnéticas de los equipos, por ello es utilizada para el control remoto de dispositivos como la televisión, radio, etc.

Al igual que todos los sistemas de comunicación, existe una parte emisora y otra que se encarga de recibir la señal. La mayoría de los sistemas que utilizan esta comunicación son unidireccionales. Uno de los protocolos mas utilizados para la comunicación de los controles remotos es el protocolo NEC, desarrollado por la compañía japonesa Nippon Electronic Company. [4]

Al ser una comunicación digital, se compone de dos señales moduladas por distancia del pulso, PDM, que corresponden a los valores binarios '1' y '0', tal como se muestra en la Figura 2.11.

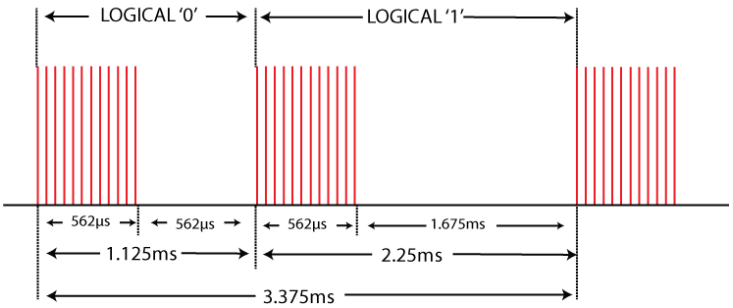


Figura 2.11: Caracteristicas de la señal modulada PDM en la comunicación por infrarrojos. [4]

Tal como se puede apreciar en la Figura 2.11, el ciclo de trabajo de las PWMs del 0 lógico y del 1, están a su vez modulados, La frecuencia de la onda portadora depende de protocolo empleado pero, en general, varía entre 36-50 kHz, siendo el más habitual en torno a los 38 kHz. [4]

En cuanto al receptor, simplemente recibe la señales PDM directamente, por lo que simplemente debe leer los datos que vaya recibiendo. Dependiendo del protocolo los datos se transmiten de una manea u otra, por ejemplo en el protocolo NEC [4], se transmite el primer byte, y a contunuación se transmite ese mismo byte negado, tal como se muestra en la Figura 2.12.

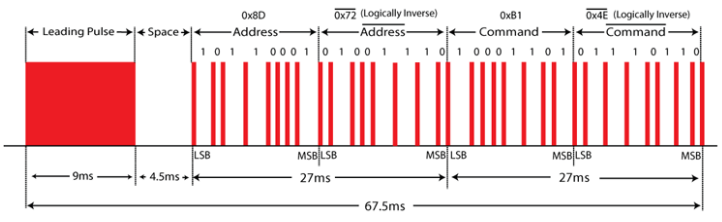


Figura 2.12: Protocolo de comunicación NEC.[4]

2.2.2.2 Módulo de control de carga y descarga de baterías DD05CVSA

Este módulo[5] permite gestionar la carga y descarga de una batería de ion litio de 3.7 V. Tal como se puede apreciar en la figura 2.13, este módulo posee tres partes diferenciadas.

La primera consiste en las patillas de alimentación externas, que deberían ir a un jack que permita la conexión con un cargador externo de 4.5V a 8V.

Las dos siguientes patillas van directamente conectadas a la batería. Este módulo es capaz de dirigir la energía a la batería, siempre que se esté alimentando externamente, y también dirigirla hacia la salida.

Por último, las dos ultimas patillas, son la salida fija de un convertidor DC/DC cuyo voltaje es de 5V, independientemente del tipo de entrada que exista en ese momento.

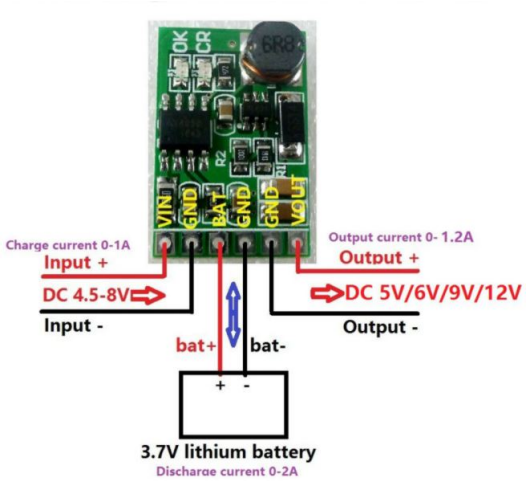


Figura 2.13: Cargador y descargador de baterías de litio.[5]

2.2.2.3 ADC/DAC PCF8591

Dado que los modelos de las placas de Raspberry no suelen incorporar ADCs, es necesario contar con módulos externos. En este caso se ha optado por un módulo[11][6] que consta de 4 entradas ADC y una salida DAC, tal como se muestra en la figura 2.14.

La comunicación empleada es I2C, que consiste en una comunicación serie basada en un protocolo de direcciones, en el que cada elemento conectado a los dos hilos de la red I2C, posee una dirección única, por lo que el Maestro solo debe saber la dirección del Esclavo con el que desea intercambiar información. Este tipo de comunicación es relativamente lenta, ya que hasta que no se termina la comunicación con uno de los los Esclavos, no es posible contactar con otro.

Este módulo será el encargado de medir el voltaje de la batería, el cual indicará el nivel de batería que posee el sistema en todo momento, y permitirá al sistema detectar cuando el usuario debe conectar de nuevo el sistema a un alimentador externo.

Debido a esta función, la relativa lentitud de la comunicación I2C no es un problema, ya que afortunadamente no es necesario un periodo de muestreo muy bajo, por lo que este sistema será capaz de realizar el seguimiento de la información del nivel de batería.

Para garantizar una mediad más fiable, se utilizarán las cuatro entradas para medir la información del nivel de voltaje de la batería. Esto implica que no es necesario realizar ningún filtro analógico, ya que posteriormente se realizará un filtro digital. Esto es posible ya que no se introduce un ruido alto, y la variación en la medida oscila entorno al 8 % [11].

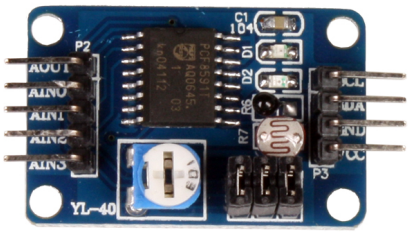


Figura 2.14: Convertidor Analógico/Digital y Digital/Analógico por comunicación I2C.[6]

2.2.2.4 Pulsador de encendido y apagado

Uno de los defectos de las placas ordenador que diseña Raspberry, es que no disponen de un botón de encendido/apagado, por lo que al utilizar un sistema operativo Linux, resulta peligroso a nivel software tener que cortar de forma brusca la alimentación.

Figura 2.15: Botón pulsador de apagado y encendido de la placa Raspberry.[7]

A pesar de que el sistema operativo permite hacerlo de forma segura siempre que se tenga acceso al sistema, pero en muchas aplicaciones es inviable o simplemente no es posible. Así que para facilitar y mejorar la experiencia del usuario, se ha decidido incluir este módulo que con un solo pulsador[7], que permite arrancar y apagar el sistema de manera segura.

La manera que utiliza para arrancar el sistema, es simplemente dejando pasar el flujo de corriente directamente a la placa. Por otro lado, el apagado es más complicado, ya que hay que modificar algunos

archivos internos del sistema operativo Raspbian, para que al detectar que en uno de los pines GPIO se está pulsando el botón de encendido/apagado, este debe comenzar el apagado de la máquina.

Capítulo 3

Descripción de las herramientas software utilizadas

3.1 Sistema operativo Gentoo

El sistema operativo de Pepper[1] es una distribución GNU/Linux basada en Gentoo, esta desarrollada específicamente para las necesidades de los robots de la compañía Aldebaran-Softbank[1]. En este sistema operativo ya se encuentran instalados numerosos programas y bibliotecas requeridas por NAOqi, el software que le da vida al robot.

Como dato de interés, en el robot Pepper no se tienen privilegios de superusuario Unix, a diferencia de su hermano NAO. Esto genera una serie de inconvenientes como la imposibilidad de descargar software directamente en el robot o configurar la herramienta PIP de Python para la descarga automática de librerías, entre otros.

3.2 Framework NAOqi

Naoqi es el nombre que reciben los programas principales instalados en los robots desarrollados por Aldebaran-Softbank (Nao, Romeo y Pepper)[1] y que se encargan de controlarlos. Además, es un framework necesario para la programación de estos humanoides y responde a las necesidades comunes de la robótica, incluyendo: paralelismo, recursos, sincronización, eventos, ect. Gracias a este framework hay comunicación homogénea de información y eventos entre los diferentes módulos o servicios (como pueden ser el de movimiento, el de audio, o el de vídeo entre otros servicios), con independencia de los lenguajes empleados para su programación en cada uno de ellos.

3.2.1 Servicios de NAOqi

NAOqi contiene por defecto una serie de módulos o servicios básicos del robot. Los cuales permiten una comunicación y sincronización con todos los elementos del sistema. Cada módulo anuncia los métodos que desea poner a disposición de los clientes que participan en la red, a un broker dentro del mismo proceso. Esto permite que el cliente maneje estos métodos sin importar donde se esté ejecutando el

servicio.

Los servicios se pueden agrupar en distintos grupos, dependiendo de las funciones para las que se utilicen. Por lo que podemos encontrarlos en los siguientes grupos:

3.2.1.1 Módulos del núcleo:

Se pueden definir como aquellos que permiten acceder a los métodos para el control interno del sistema, como es por ejemplo la memoria o el manejo de aplicaciones instaladas, usuarios detectados, etc. Ver Cuadro 3.1 con todos los servicios que engloban esta categoría.

ALBehaviorManager	Arranca y para los behaviors.
ALConnectionManager	Maneja la conexión a la red y su configuración.
ALDiagnosis	Obtiene el último diagnóstico activo o pasivo realizado.
ALExpressionWatcher	Combina los eventos de memoria en una expresión para generar eventos más complejos.
ALKnowledge	Permite administrar datos persistentes en forma de triplete (sujeto - predicado - objeto).
ALMemory	Ver sección de **Módulo específico de ALMemory** 3.2.1.2.
ALModule	Permite crear módulos propios.
ALMood	Administra el estado emocional de los usuarios activos.
ALNotificationManager	Administra las notificaciones.
ALResourceManager	Administra los recursos.
ALSystem	Administra el sistema del robot.
ALUserInfo	Administra los datos persistentes acerca de los usuarios.
ALUserSession	Administra el estado de los usuarios activos y los enlaces a sus datos.
ALTabletService	Carga la aplicación web, reproduce vídeos y administra su propia tablet.
ALWorldRepresentation	Almacena datos a largo plazo sobre objetos detectados en una base de datos espacialmente estructurada.
PackageManager	Permite la instalación y desinstalación de las aplicaciones empaquetadas.
ServiceManager	Administra el comienzo, el final y el estado de los servicios.

Tabla 3.1: Servicios del núcleo.[1]

3.2.1.2 Modulo específico de AlMemory:

ALMemory es un servicio del núcleo que permite acceder directamente a memoria. Lo que significa un control total tanto de las variables como de todos los eventos del sistema. Gracias a este servicio es posible acceder, leer y escribir variables, que son compartidas por todas las aplicaciones instaladas en el

robot. Lo cual convierte a este módulo en una poderosa herramienta de comunicación entre aplicaciones. También es donde se almacena toda la información, tanto de sensores como de actuadores.

Por todo ello, es posible tener un completo control del robot, ya que gracias a los métodos de este servicio, es posible conocer el estado de las variables, saber cuando una variable cambia, o cambiar el valor de una variable, o simplemente manejar los eventos.

Un evento es una señal interna del software que permite saber cuando el valor de una variable en memoria es actualizado. Estos eventos no solo se activan cuando cambie el valor de la variable, sino que es posible saber cuando una aplicación simplemente accede a ese cluster de memoria.

En el Cuadro 3.2 se explican los métodos más utilizados para manejar la memoria.

declareEvent(string key)	Declarar un evento que permita futuras suscripciones al evento.
getData(string key)	Obtienes la información guardada en una variable (key).
getSubscribers(string key)	Devuelve una lista con todos los subscriptores e un evento.
insertData(string key, ALValue value)	Inserta un valor en una variable determinada pero no se lanza el evento.
raiseEvent(string key, ALValue value)	Actualiza el valor de una variable lanzando un evento.
removeData(string key)	Elimina la variable de memoria.
subscribeToEvent(string NombreEvento, string NombreModulo, string NombreMetodo)	Método para suscribirse a un evento, en el cual se deben especificar el nombre del evento, nombre del módulo para llamar con notificaciones y el nombre del método del módulo para llamar cuando se cambia un dato
unsubscribeToEvent(string key, string callbackModule)	Cancela la suscripción al evento.

Tabla 3.2: Métodos más utilizados para controlar la memoria.[1]

3.2.1.3 Módulos de interacción:

Son aquellos servicios que permiten al robot interactuar con las personas, y mostrarse con vivacidad, dando la sensación a los usuarios de estar realmente vivo. Con estos módulos se pueden controlar los estímulos a los que se puede ver sometido el robot. Además de gestionar todos los métodos referentes a los diálogos. Ver Cuadro 3.3 con todos los servicios que engloban esta categoría.

ALAutonomousLife	Mantiene el ciclo de vida del robot y gestiona el lanzamiento de las actividades.
ALAutonomousBlinking	Permite al robot hacer que sus ojos parpadeen cuando detecta a alguien y cuando está interactuando.
ALBackgroundMovement	Define los ligeros movimientos que hace el robot cuando no mueve ningún miembro.
ALBasicAwareness	Permite al robot reaccionar en función del medio para establecer y mantener el contacto visual con las personas.
ALListeningMovement	Permite al robot hacer ligeros movimientos para dar la sensación de que está escuchando.
ALSpeakingMovement	Habilita los movimientos autónomos que realiza el robot mientras reproduce un discurso.
ALDialog	Crea una base de conocimientos básicos para las habilidades de conversación, creando diálogos para interactuar con los usuarios.

Tabla 3.3: Servicios de los sistemas de interacción.[1]

3.2.1.4 Módulos de movimiento:

Estos servicios permiten al robot realizar movimientos y desplazarse por el suelo. Ver Cuadro 3.4 con todos los servicios que engloban esta categoría.

ALAnimationPlayer	Inicia y detiene las animaciones.
ALRobotPosture	Fuerza al robot a realizar una postura predefinida.
ALNavigation	Hace al robot moverse de forma segura, parando y evitando obstáculos.
ALRecharge	Detecta la estación de carga del robot y le hace desplazarse hasta ella.
ALTracker	Seguimiento de diferentes objetivos utilizando diferentes medios (cabeza, todo el cuerpo, mover).

Tabla 3.4: Servicios de los sistemas de movimiento.[1]

3.2.1.5 Módulos de audio:

Son aquellos con los que el robot es capaz de detectar sonidos así como de reproducir sonidos, ya sea un archivo de sonido o bien una cadena de caracteres que es capaz de convertir en sonido. Ver Cuadro 3.5 con todos los servicios que engloban esta categoría. Ver Cuadro 3.5 con todos los servicios que engloban esta categoría.

ALAudioDevice	Maneja las entradas y salidas de audio. Este módulos es usado por el resto de módulos de audio, excepto **ALAudioPlayer**.
ALAudioPlayer	Reproduce archivos de audio en el robot.
ALAudioRecorder	Graba archivos de audio desde los micrófonos del robot.
ALSoundDetection	Detecta los eventos de sonido.
ALSoundLocalization	Localiza los sonidos detectados por el modulo **ALSoundDetection**.
ALSpeechRecognition	Permite al robot reconocer lo que los humanos dicen.
ALTextToSpeech	Permite que el robot diga una frase escrita.
ALAnimatedSpeech	Hace al robot moverse mientras habla.
ALVoiceEmotionAnalysis	Identifica las emociones expresadas por la voz de la persona con la que esté interactuando.

Tabla 3.5: Servicios de audio.[1]

3.2.1.6 Módulos de percepción:

Mientras que los módulos de visión permiten detectar personas, estos módulos de percepción son los que se encargan de analizar los resultados obtenidos por los módulos de visión y así sacar las características, posición, estados, etc; de las personas detectadas. Toda esta información es procesada y utilizada para permitir una correcta interacción del robot con los seres humanos. Ver Cuadro 3.6 con todos los servicios que engloban esta categoría.

ALEngagementZones	Analiza la posición de una persona respecto al robot. Siempre englobando tres zonas de distancia distintas.
ALFaceCharacteristics	Proporciona las características estimadas basadas en el analisis facial.
ALFaceDetection	Hace que el robot detecte y reconozca caras humanas.
ALGazeAnalysis	Analiza la dirección de la mirada de una persona.
ALPeoplePerception	Hace al robot guarde referencias de la gente que le rodea.
ALSittingPeopleDetection	Hace que el robot detecte si una persona se encuentra sentada o no.
ALWavingDetection	Permite detectar si las personas tratan de llamar la atención del robot.

Tabla 3.6: Servicios de los sistemas de percepción.[1]

3.2.1.7 Módulos de sensores y LEDs:

En muchos casos, al desarrollar una aplicación para el robot, es necesario tener acceso a los datos devueltos por los sensores, ya que permite un control más primario del robot. Por eso mismo, es posible acceder a dicha información para así poder desarrollar correctamente una función especifica. El fabricante también

proporciona unos servicios para controlar las luces LEDs. Ver Cuadro 3.7 con todos los servicios que engloban esta categoría.

ALLeds	Controla los LEDs del robot.
ALTactileGesture	Administra los gestos táctiles realizados en los sensores táctiles de la cabeza.
ALBattery	Maneja el hardware referente a la batería del robot.
ALBodyTemperature	Detecta cuando alguna parte del hardware alcanza una temperatura elevada.
ALChestButton	Controla el botón del pecho.
ALFsr	Controla los cambios detectados en los pies.
ALSensors	Controla los sensores del robot.
ALSonar	Controla los sensores de ultrasonidos del robot.
ALTouch	Controla si el robot es tocado.
ALLaser	Controla los láseres de la cabeza.
ALInfrared	Permite una comunicación por señales de infrarojos. Esta API solo está disponible en NAO.

Tabla 3.7: Servicios de los sensores y los LEDs.[1]

3.2.1.8 Módulos de visión:

Los módulos de visión son aquellos que permiten al robot captar y analizar toda la información de sus cámaras. Con estos módulos, el robot es capaz de detectar objetos, personas y obstáculos. Ver Cuadro 3.8 con todos los servicios que engloban esta categoría.

ALPhotoCapture	Toma fotografías y las guarda en la memoria.
ALVideoDevice	Controla todos los dispositivos de vídeo.
ALVideoRecorder	Graba archivos de vídeo.
ALBacklightingDetection	Comprueba si la imagen de la cámara está retro iluminada.
ALBarcodeReader	Detecta y lee códigos de barra en una imagen.
ALColorBlobDetection	Detecta burbujas de un color determinado, no necesariamente circulares.
ALDarknessDetection	Detecta si el entorno está oscuro.
ALLandMarkDetection	Detecta un monumento visual especifico.
ALMovementDetection	Detecta algunos movimientos e informa de donde provienen.
ALRedBallDetection	Detecta objetos rojos y circulares.
ALVisionRecognition	Permite al robot reconocer y aprender patrones visuales.
ALVisualSpaceHistory	Construye un mapa con fecha y hora de las posiciones de la cabeza.
ALLocalization	Da al robot la habilidad de regresar a un lugar ya aprendido.
ALVisualCompass	Utiliza las imágenes para orientarse en el entorno.
ALCloseObjectDetection	Detecta objetos que se encuentran demasiado cerca como para ser detectados por la cámara 3D.
ALSegmentation3D	Segmenta las imágenes devueltas por el sensor 3D.

Tabla 3.8: Servicios de visión.[1]

3.2.2 SDK de NAOqi

El framework del robot permite la programación mediante un SDK[1], permite la comunicación con NAOqi. Este SDK se encuentra disponible en los siguientes lenguajes de programación mostrados en la figura 3.1.

Programming Languages	Bindings running on		Choregraphe support	
	Computer	Robot	Build Apps	Edit code
Python	✔	✔	✔	✔
C++	✔	✔	⊘	⊘
Java	✔	⊘	⊘	⊘
JavaScript	✔	✔	✔	⊘
ROS	✔	⊘	⊘	⊘

✔	OK
⊘	Not available

Figura 3.1: Soporte de lenguajes para el SDK de Naoqi.[1]

C++ y Python son los lenguajes de programación que tienen soporte más completo de Naoqi. Sin embargo el lenguaje más pensado para el desarrollo de aplicaciones por parte del fabricante es Python, esto se puede apreciar en sus recomendaciones y lo utilizan como lenguaje base para su suite de programación choregraphe (sección 3.2.3). C++ esta recomendado para el fabricante como lenguaje para el desarrollo de funciones o módulos que requieran un mayor rendimiento. Como puede ser acceder al buffer de audio o vídeo, procesado de imágen... Javascript es un lenguaje con soporte del SDK pensado sobre todo para el desarrollo de su tablet, ya que esta funciona mediante HTML5 como si de un navegador se tratase.

Finalmente llama la atención de como en la figura 3.1, se muestra un soporte para ROS, el cual se comentará más adelante en la sección 3.2.5.

3.2.3 Choregraphe

Choregraphe es un una herramienta de programación realizada por Aldebaran-Softbank[1] para el desarrollo de aplicaciones de sus robot. Se trata de un software gratuito pero no libre y se encuentra disponibles para GNU-Linux, MAC y Windows.

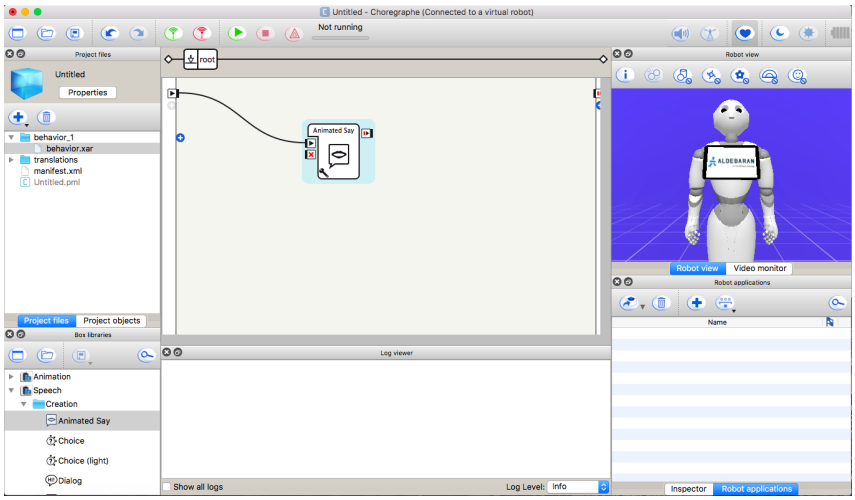


Figura 3.2: Software Choregraphe.

Es una aplicación que ha sido desarrollada para un público cuyos conocimientos de programación no son muy amplios, debido a ello cuenta con una programación visual (parecida a los esquemas de Simulink) mediante esto se tiene la intención de facilitar la programación y el seguimiento del ciclo de la aplicación.

En la figura 3.2 se puede apreciar que Choregraphe cuenta con distintos paneles o vistas. A continuación se va a comentar brevemente la función que aportan cada uno de ellos.

3.2.3.1 Panel Project Files.

En este panel se puede interactuar con la estructura y los ficheros que componen el proyecto.

El fichero que contiene las cajas de programación se denomina Behaviour y tienen extensión *.xar. Un mismo proyecto puede tener diferentes Behaviour, pudiendo estos funcionar a la vez, dependiendo de

su configuración.

La tablet que presenta Pepper se programa en HTML5, siendo la libreria de NAOqi de Javascript la que le otorga la iteracción con Pepper). El proyecto HTML de la tablet se debe ubicar en una carpeta denominada 'html' en la raiz del proyecto y por defecto se cargará el fichero index.html que se encuentre en esta carpeta.

3.2.3.2 Panel Box Libraries.

Por defecto hay muchos bloques o cajas de aplicación desarrollados por Aldebaran-Softbank, bloques para hablar, para el desplazamiento, para mostrar u ocultar la tablet, ect. Esta tarea facilita mucho la programación para usuarios poco avanzados ya que se conseguiran estos resultados en el robot con tan solo arrastrar estas cajas de este panel al panel del programación.

3.2.3.3 Panel de programación.

Esta puede ser la vista más importante para el desarrollo de la aplicación, ya que en esta se implementa la lógica. Se pueden arrastrar cajas del bloque del panel de las librerías o desarrollar cajas nuevas programadas en Python. En esta vista se enlazan las cajas y se define el flujo del behaviour, desde que comienza el hasta el fin de este.

3.2.3.4 Panel Robot View

En este panel se puede ver en tiempo real los movimientos del robot virtual o el robot real al que estamos conectados.

3.2.3.5 Panel Robot Application.

En esta vista se pueden ver todas la aplicaciones que se encuentran instaladas en el robot, instalar una nueva aplicación a partir de un paquete 'PKG' generado por Choregraphe, desinstalar una aplicación y por último ejecutar o parar las aplicaciones.

3.2.3.6 Inconvenientes en el desarrollo de grandes aplicaciones con Choregraphe.

Aunque el uso de Choregraphe para el desarrollo de aplicaciones facilita mucho la entrada a los usuarios que quieran programar estos robots sin tener amplios conocimientos en programación. Tiene una serie de inconvenientes para usuarios más experimentados o a la hora de desarrollar aplicaciones más complejas, estos inconvenientes se enumerarán a continuación:

- Dificulta la utilización de programa de gestión de versiones como puede ser GIT. Ya que cualquier cambio en el esquematico (arrastrar un bloque), supone un cambio total en el fichero del behaviour, por lo que GIT no puede seguir un solo cambio.
- Cuando la aplicación es más compleja, la cantidad de bloques o cajas que se emplea es mucho mayor por lo que se satura el espacio visual y es dificil de seguir la aplicación.

3.2.4 Desarrollo Software por medio de Servicios

Debido a los numerosos puntos negativos encontrados para el desarrollo de aplicaciones complejas mediante la suite Choregraphe ofrecida por el fabricante (sección 3.2.3). Es casi un requisito el abandono de esta herramienta y la programación del robot mediante su SDK. Renunciando a la idea de una programación visual en favor de una programación tradicional.

El desarrollo de aplicaciones mediante el SDK se encuentra disponible para varios lenguajes de programación, que como se pudo ver en la seccion 3.2.2, para el desarrollo de aplicaciones que se pudieran instalar en el robot, los lenguajes disponibles son Python y C++. Al comienzo de la aplicación es necesario definir si esta se va a tratar de un cliente o un servicio de qimessagin.

- Un cliente permite realizar una aplicación para Pepper, que puede ejecutarse dentro o fuera del robot, y que puede conectarse a los servicios de este. Pero no se pueden acceder a métodos desarrollados en esta clase desde otro servicio.
- Sin embargo si desarrollamos un servicio tendremos todas las posibilidades que ofrece un cliente, pero además los métodos del objeto o clase desarrollado pueden ser utilizado por otros servicios y clientes de Naoqi, local o remotamente, con independencia del lenguaje de programación.

No obstante al abandonar Choregraphe, no se tienen disponibles las herramientas para la gestión de aplicaciones desarrolladas por el fabricante. Notando como principal carencia, la instalación o desinstalación de aplicaciones y el control de cual se esta ejecutando. Por lo que para solventar este problema, se puede crear una aplicación en Choregraphe que únicamente contenga un bloque de programación que se encargaría de controlar un servicio desarrollado en Python. Mediante este procedimiento se podría prescindir Choregraphe para el desarrollo de la aplicación mientras que se podría utilizar para la instalación, desinstalación y el control de las aplicaciones que se encuentren instaladas.

3.2.5 ROS en Pepper

ROS (*Robot Operating System*) es un framework cuya finalidad es el desarrollo de software para robots que provee la funcionalidad de un sistema operativo, ya que proporciona los servicios propios de este: abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. ROS es software libre bajo términos de licencia BSD, la cual permite la libertad para uso comercial e investigador.

3.2.5.1 Instalar ROS en Pepper.

Aunque el principal middleware y solución que propone el fabricante es NAOqi. Aldebaran-Softbank ha desarrollado ROS interface, se tarta de un puente de comunicación entre NAOqi y ROS. El cual permite la programación del robot en ROS y convierte las ordenes al entorno de NAOqi.

ROS interface, ha sido desarrollado para instalarse en un ordenador externo, el cual tendría el control de uno o varios robots. Esto puede suponer un inconveniente, ya que Pepper vería reducida su independencia.

No obstante, hay constancia de grupos de trabajo en universidades que han conseguido la instalación de ROS en el robot, aunque aún no se ha publicado ningún trabajo al respecto. Para ello se ha tenido que compilar el software previamente en la maquina virtual de NAOqi 2.1.4, sin embargo se trata de un proceso muy costoso. Este proceso se puede complicar más, ya que una vez instalado ROS, si se desean paquetes adicionales de este, es necesario la compilación en la maquina virtual y su instalación en el robot para cada uno de los paquetes.

3.2.6 Dialogos con QiChat

Pepper[1] es un robot capaz de mantener conversaciones con los seres humanos, siendo esta una de sus características más llamativas. Existen dos maneras de conseguirlo, "speech to text.[en] el que se transcribe el audio escuchado a texto, y por otro lado predefinir las posibles respuestas del usuario en un script llamado topic.

Al no encontrar el ASR (speech to text) de forma nativa en español, aunque si se encuentra en inglés, francés y japonés, obliga a recurrir a APIs externas como las de Google, Microsoft, Watson, etc. La mayor desventaja es que si el robot no posee una conexión a internet no podría utilizar estos servicios, además del coste económico por consulta.

Por otro lado el sistema de comparación llamado Topic el cual se encuentra nativo en Pepper utilizando el motor de Nuance. El robot analiza el audio, otorgándole un valor correspondiente a la probabilidad de acierto llamado *confidence*, si este supera un límite, el robot considera que ha escuchado esa regla y continua en el árbol de decisiones. Tiene como ventajas la disponibilidad offline aunque ofrece menos libertad en las respuestas. Este sistema permite crear un árbol de decisiones basado en reglas.

Un Topic es un script, el cual permite al robot realizar una acción cada vez que se cumple una *rule* (regla).

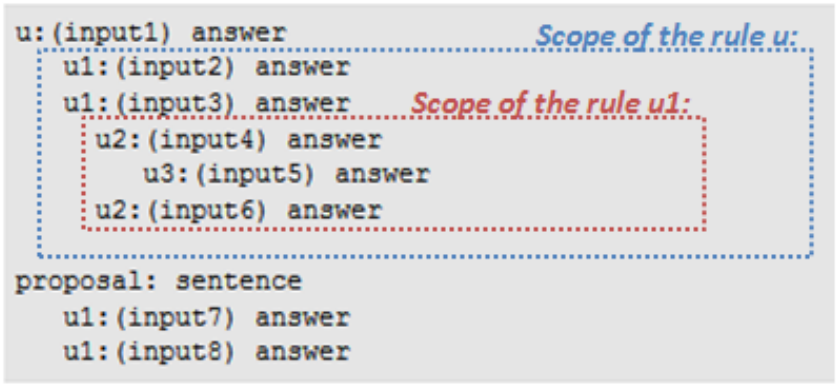


Figura 3.3: Esquema de organización de los dialogos en NAOqi.[1]

Un *rule* o regla, es la relación entre la entrada de información por parte del humano, y la acción que debe realizar el robot al cumplirse esa regla. Como se muestra en la Figura 3.1, se puede apreciar la estructura que deben seguir la reglas y subreglas en los dialogos. Las subreglas son llamadas también *scopes* (extensiones) de una regla.

Por último, nombrar la posibilidad de poder realizar diferentes funciones desde los diálogos, como iniciar un behavior, llamar al método de un servicio, cambiar o leer una variable en memoria. También es posible establecer un evento como regla para comenzar un diálogo. Al igual que las reglas pueden cumplirse al activarse un evento o iniciar el dialogo.

3.3 Sistema operativo Raspbian

Raspbian[2] es una distribución del sistema operativo GNU/Linux basado en Debian Jessie, especialmente diseñado para las placas computadoras de Raspberry Pi[2]. Se trata de un software libre y de código abierto, por lo que a nivel profesional resulta muy sencillo realizar tareas personalizadas. Además existe una gran comunidad de desarrolladores que han creado una serie de librerías que permiten gestionar todas las funcionalidades de las GPIO.

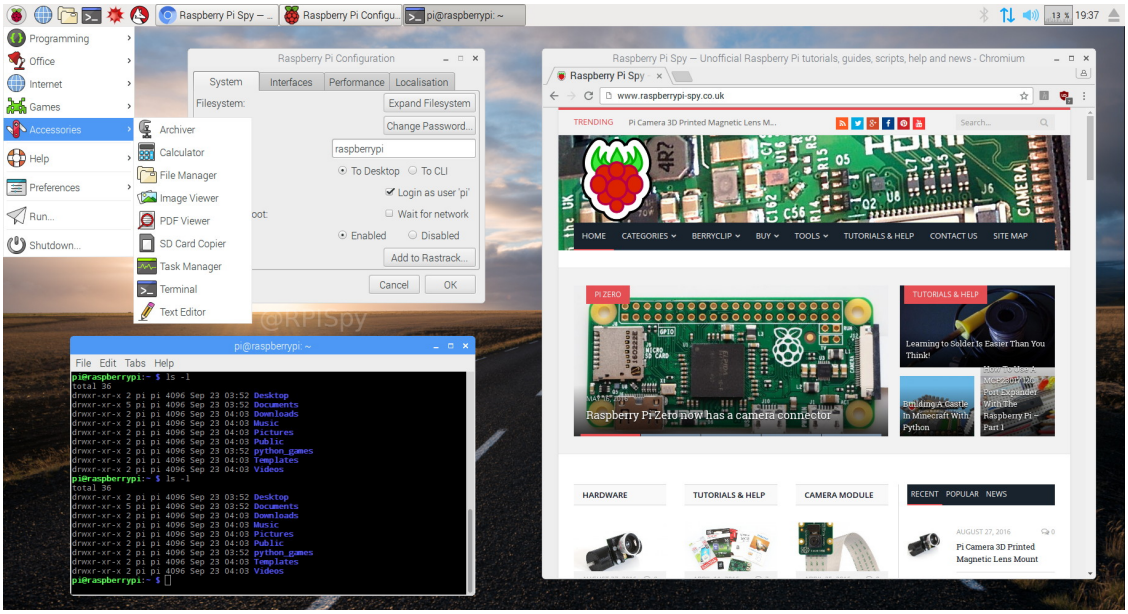


Figura 3.4: Captura de pantalla de interfaz gráfica de Raspbian.

Gracias a su interfaz gráfica, es posible realizar configuraciones de manera sencilla, aunque al ser una distribución Debian posee su bash, que permite realizar toda la configuración mediante terminal.

Esta distribución, permite controlar las GPIO por scripts de Python, lo que resulta muy cómodo a la hora de incluir el manejo de los pines en cualquier aplicación o funcionalidad.

3.3.1 Paquete LIRC de control de emisor IR

LIRC (Linux Infrared Remote Control)[12] es un paquete de software que permite codificar, decodificar y enviar señales por infrarrojos, siempre que el hardware lo permita. En este proyecto se utiliza como hardware una Raspberry Pi Zero W,que permite la posiblilidad de conectar el emisor y el receptor de infrarrojos a unos pines específicos GPIO.

Gracias a que Raspbian y LIRC comparten módulos del kernel, es muy sencillo realizar las configuraciones iniciales. También resulta muy ventajoso el echo de que LIRC permite decodificar los datos

recibidos por infrarrojos, permitiendo crear archivos de comandos propios de prácticamente cualquier dispositivo.

3.3.1.1 Instalación y configuración

Como cualquier librería de Linux, se deben realizar una serie de pasos para poder disfrutar de las funcionalidades de LIRC[13][12]. Ya que se deben modificar una serie de ficheros para configurar los pines GPIO emisor y receptor.

En primer lugar se debe instalar la librería escribiendo en una terminal la siguiente sentencia:

```
$ sudo apt-get install lirc
```

Una vez finalizada la instalación, se realiza la configuración y establecimiento de pines GPIO. Se comienza modificando el archivo **/etc/modules**:

```
$ sudo nano /etc/modules
```

En este archivo se deben añadir las líneas para establecer los pines de entrada y salida correspondientes, en este proyecto se establece el pin GPIO18 como entrada y el pin GPIO17 como salida:

```
lirc_dev
lirc_rpi gpio_in_pin=18 gpio_out_pin=17
```

A continuación, se realizan las modificaciones en el archivo **/etc/lirc/hardware.conf**

```
$ sudo nano /etc/lirc/hardware.conf
```

```
##################################################
# /etc/lirc/hardware.conf
#
# Arguments which will be used when launching lircd
LIRCD_ARGS="--uinput"

# Don't start lircmd even if there seems to be a good config file
# START_LIRCMD=false

# Don't start irexec, even if a good config file seems to exist.
# START_IREXEC=false

# Try to load appropriate kernel modules
LOAD_MODULES=true

# Run "lircd --driver=help" for a list of supported drivers.
DRIVER="default"
# usually /dev/lirc0 is the correct setting for systems using udev
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"

# Default configuration files for your hardware if any
LIRCD_CONF=""
LIRCMD_CONF=""
##################################################
```

Posteriormente se realiza la modificación oportuna en el archivo **/boot/config.txt** añadiendo la siguiente línea:

```
$ sudo nano /boot/config.txt
```

```
dtoverlay=lirc-rpi,gpio_in_pin=18,gpio_out_pin=17
```

Por último se debe reiniciar **lircd**:

```
$ sudo /etc/init.d/lirc stop
$ sudo /etc/init.d/lirc start
```

Llegados a este punto, el dispositivo ya estaría perfectamente configurado para utilizar la librería LIRC.

3.3.1.2 Archivos de comandos.

Los diferentes comandos y datos que se pueden transmitir, están guardados en un tipo de ficheros *.conf* específicos en una ruta concreta. En estos ficheros se establecen las siguientes configuraciones:

- Nombre del mando a distancia.
- Duración de los pulsos PDW, tanto para el '1' lógico como para el '0'.
- Código en hexadecimal de las diferentes teclas.
- Nombre de las diferentes teclas.

```
begin remote

  name NOMBRE_MANDO
  bits 16 # Tamaño del dato
  flags SPACE_ENC|CONST_LENGTH
  eps 30
  aeps 100

  header 9024 4462 # Tiempos correspondientes a la cabecera.
  one 585 1662 # Tiempo en alto y bajo de un uno lógico.
  zero 585 538 # Tiempo en alto y bajo de un cero lógico.
  ptrail 588
  repeat 9018 2225 # Tiempo del comando de repetición
  pre_data_bits 16
  pre_data 0x4F9
  gap 107941
  toggle_bit_mask 0x0

      begin codes
          KEY_POWER 0x00FF # Nombre de la tecla(KEY_POWER) y código(0x00FF)
          KEY_BACK 0x807F # Nombre de la tecla(KEY_BACK) y código(0x80F7)
          .
          .
          .
          .
```

```
    end codes

end remote
```

Teniendo en cuenta el anterior archivo de configuración, para hacer que el dispositivo envie el comando correspondiente a pulsar una vez la tecla de *ENCENDER*, desde la terminal se debe enviar el siguiente comando:

```
$ irsend send_once NOMBRE_MANDO KEY_POWER
```

Lo que enviará por el canal el dato *0x00FF* hasta el receptor.

3.4 Tornado websocket

Tanto NAOqi como Raspbian poseen un sistema de comunicación por websocket llamado Tornado, en el caso de Raspbian se podría utilizar otro, pero esta opción es imposible de realizar en NAOqi debido a la falta de permisos de administrador.

Tornado[14] es un web framework de Python y una librería de redes asíncronas, desarrollada originalmente en FriendFeed. Usando redes sin bloqueo de entrada y salida, Tornado puede manejar decenas de miles de conexiones abiertas, lo que lo hace ideal para long polling, WebSockets y otras aplicaciones que requieren una conexión de larga duración para cada usuario. En este proyecto únicamente se utilizará tornado como Websocket[14].

Los Websocket permiten una comunicación bidireccional entre cliente y servidor. Tras recibirse la petición de conexión del cliente, el servidor permite una conexión directa y en ambas direcciones, lo que permite mantener a dos sistemas conectados todo el tiempo que sea necesario.

Existen diferentes métodos que gestionan tanto los distintos eventos como las salidas que suceden tanto en el lado del servidor como del cliente. Todos estos métodos deben pertenecer a una clase especifica:

```
class tornado.websocket.WebSocketHandler(application, request, **kwargs)
```

3.4.1 Eventos (Events):

Eventos característicos de Tornado Websocket[14]:

- `WebSocketHandler.open(*args, **kwargs)`

  Este evento se ejecuta cuando se abre una nueva conexión websocket.

- `WebSocketHandler.on_message(message)`

  Este evento es activado al recibirse un mensaje.

- `WebSocketHandler.on_close()`

  Al activarse, este método permite realizar las funciones que se deseen al cerrar la conexión, como por ejemplo borrar la información de un usuario.

3.4.2 Salidas (Outputs):

Salidas correspondientes de Tornado Websocket[14]:

- `WebSocketHandler.write_message(message, binary=False)`

  Este método permite enviar información. En el caso del servidor se envía a todos los clientes, mientras que los mensajes que salen desde el cliente solo son recibidos por el servidor.

- `WebSocketHandler.close(code=None, reason=None)`

  Con este método es posible cerrar una conexión.

Capítulo 4

Funcionamiento completo del sistema

Una computadora puede ser llamada "inteligente" si logra engañar a una persona haciéndole creer que es un humano.

Alan Turing

En este proyecto se busca realizar una aplicación sobre el robot Pepper, la cual permita al robot interactuar con los usuarios en un entorno domestico. Como ya se ha mencionado anteriormente, este robot carece de ciertas características hardware, por lo que es necesario diseñar un sistema de comunicación con los módulos hardware adicionales sin ser necesario modificar su hardware interno, despojando al usuario de la consiguiente garantía.

Como herramienta en un entrono doméstico, la comunicación mediante radiación infrarroja (IR), es muy util debido a la gran cantidad de electrodomésticos o aparatos, que son controlados mediante este sistema. Es por eso que se ha elegido que la ampliación de hardware que este robot necesita es un dispositivo que pueda emitir y recibir comandos por IR.

Por todo lo anteriormente mencionado, se van a desarrollar una aplicación que permita comunicar ordenes desde NAOqi a un dispositivo externo al robot conectado a su misma red local o LAN. El dispositivo actuará como servidor, mientras que el robot actuara como cliente, permitiendo al robot enviar la información oportuna y ser él el que abra la conexión.

La comunicación se realiza por medio de un websocket creado con la herramienta Tornado (3.4). Por lo tanto, es necesario crear un protocolo de comunicación, intercambiando cadenas de caracteres entre cliente y servidor, para así poder enviar de un extremo a otro información.

Teniendo en cuenta los objetivos de este proyecto, se ha diseñado un método que permite no solo comunicar a Pepper con el dispositivo, sino que se podrá controlar el emisor de infrarrojos desde cualquier aparato que disponga de un navegador web como puede ser un PC, una tablet, un teléfono móvil, etc.

4.1 Descripción de funcionamiento de la aplicación en Pepper

Basándonos en los apartados anteriores 2.1 y 3.2, en los que se ha estudiado el robot tanto a nivel software como a nivel hardware, es posible desarrollar una aplicación que cubra las necesidades de este proyecto.

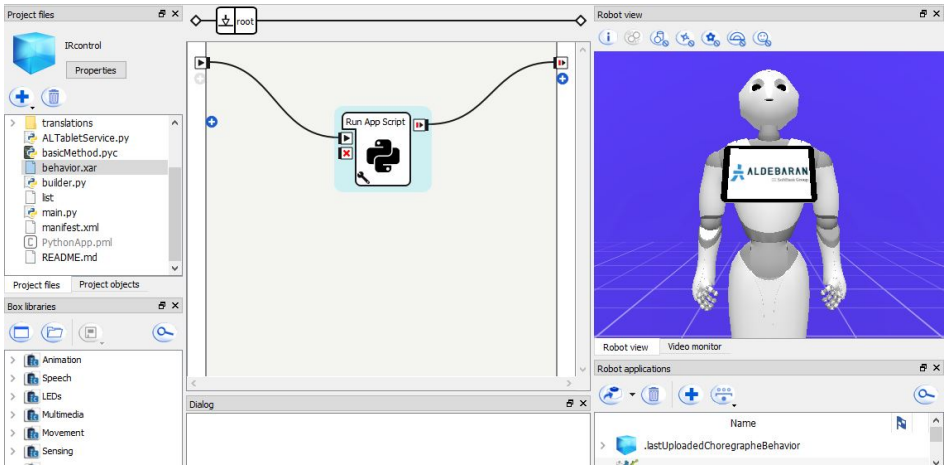


Figura 4.1: Programa desarrollado sobre Choregraphe. Solo arranca el script en el que se ejecuta el servicio.

Tal como se comenta en el aparatado 3.2.3, las aplicaciones que son ejecutadas en el robot se diseñan con un programa distribuido por el fabricante, el cual simplemente será utilizado para diseñar un sistema de arranque y destrucción de un servicio. Dicho servicio, posee una serie de métodos, que serán utilizados tanto por la aplicación web que se ejecuta en la tablet, como en el script del dialogo.

Debido al hecho de que esta parte del proyecto se realiza como un servicio más de NAOqi, resulta más cómodo y sencillo comunicar los tres elementos de la aplicación, el script del árbol de diálogos, la aplicación web de la tablet y el websocket.

4.1.1 Variables en NAOqi

En la sección 3.2.1.2, se explica que NAOqi permite gestionar las variables de sus sistema, esto implica que es posible tener, compartir y suscribirse a eventos de estas variables desde cualquier aplicación y programa del robot. Así como desde cualquier aplicación conectada a NAOqi, también es posible acceder a esta información.

NOMBRE	TIPO	USO
IR_control/Dispos/nuevo	INT	Utilizada para guardar el número de dispositivos servidores encontrados en una red.
IR_control/Dispos/dispositivoN	ARRAY	Donde N=[1,2,3,...] es el número de dispositivo. Guarda la información relativa a un dispositivo en concreto. Los aparatos de los que dispone, el nombre del dispositivo y la dirección IP.
IR_control/actualizando	BOOL	Se encuentra a '1' cuando se está realizando el proceso de barrido de búsqueda de dispositivos.
IR_control/act_stop	BOOL	Para el proceso de búsqueda de dispositivos por barrido de IP cuando se activa el evento y la variable vale 1.
IR_control/dispositivo	STRING	Guarda el nombre dispositivoN que corresponde al dispositivo que está seleccionado y que por lo tanto es con el que se comunicará.
IR_control/IP_server	STRING	Guarda la dirección IP que corresponde al dispositivo que está seleccionado.

Tabla 4.1: Lista de variable globales almacenadas en NAOqi.

En la tabla 4.1, se muestra una lista de todas las variable utilizadas en el sistema de NAOqi, haciendo posible que desde varios nodos sea posible el control y manejo de dichas variables.

4.1.2 Descripción del servicio y los métodos desarrollados

Tal como se define en la sección 3.2.4, se ha desarrollado una aplicación la cual define un servicio en el robot. Esto permite, una vez se ejecute el programa, que cualquier aplicación podrá acceder a casi todos los métodos públicos del servicio, independientemente del lenguaje de programación utilizado. En este proyecto se ha desarrollado el servicio en lenguaje Python, por las razones explicadas en la sección 3.2.2.

En primer lugar definir todos los archivos y herencias implicadas en la elaboración del servicio, tal como se muestra en la figura 4.2, destacar el archivo *main.py*, programa principal que se encuentra almacenado en el directorio raíz del proyecto. El archivo *basicMethod.py*, guardado en el directorio */lib* el cual contiene una serie de métodos desarrollados por la empresa Juguetrónica SL utilizados para realizar las gestiones oportunas en NAOqi. Por último destacar el acceso a todos los servicios activos en el robot pertenecientes al framework NAOqi.

4.1.2.1 Explicación de los métodos del archivo *basicMethod.py*

Para comprender mejor el programa principal, es necesario explicar todos los métodos que son heredados desde el archivo *basicMethod.py*(figura 4.2). Todods ellos son heredados en la clase principal del programa desarrollada en el archivo *main.py* llamada **ROBIRcontrol**. Dichos métodos, son los encargados de realizar las configuraciones, suscripciones y gestión de los diferentes elementos utilizados en la aplicación, como los diálogos o el uso de la tablet.

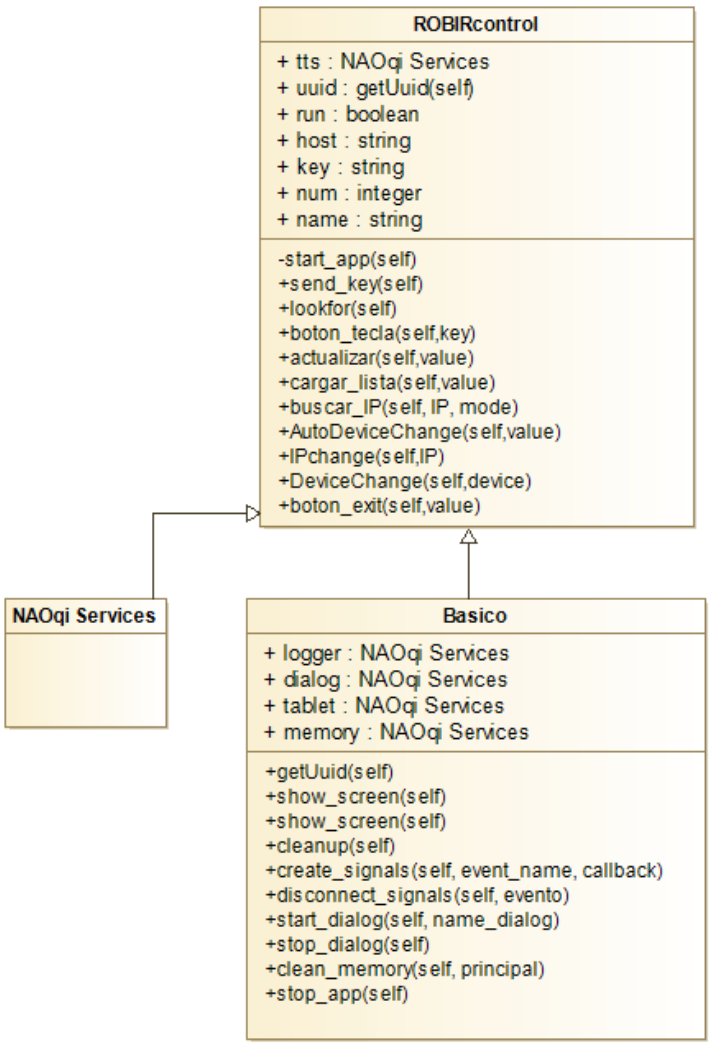


Figura 4.2: Diagrama de clases del servicio ROBIRcontrol.

- **getUuid(self)** Retorna el valor del UUID de la aplicación. Dato utilizado en otros métodos para dar de alta diferentes servicios y realizar gestiones de hardware.
- **show_screen(self)** Método que realiza la puesta en marcha de la aplicación web sobre la tablet del robot.
- **hideTablet(self)** Gestiona el cierre de la aplicación web dejando la tablet en su estado de reposo.
- **cleanup(self)** Llama a los métodos que se encargan de cerrar la aplicación web de la tablet y el dialogo.
- **create_signals(self, event_name, callback)** Crea la suscripción a un evento, y establece el método que será ejecutado al detectarse el evento. Retorna dos datos utilizados para la desuscripción del evento.
- **disconnect_signals(self, evento)** Realiza la desuscripción de un evento.
- **start_dialog(self, name_dialog)** Carga un script de dialogo o *topic* guardado en la carpeta */Dialogos*.
- **stop_dialog(self)** Desactiva y cierra el último dialogo cargado en el robot.

- **clean_memory(self, principal)** Limpia todas las variables en memoria cuyo nombre comience por la cadena de caracteres que es pasado como argumento.
- **stop_app(self)** Para la aplicación y permite el cierre y final del servicio.

4.1.2.2 Explicación de los métodos desarrollados en el servicio ROBIRcontrol

Esta clase define el servicio que se desea crear para que sea posible la comunicación por websocket y el intercambio de información entre los diferentes elementos del robot. Por ello se han creado una serie de métodos públicos (figura 4.2), aunque existe un método privado que es en el que se encuentra el bucle principal del programa.

Pese a que NAOqi al abrir un servicio permite mantenerlo arrancado, al estar desarrollándose una aplicación, es necesario realizar dejar cerrada la gestión de los diálogos y de la tablet, por lo que no sería adecuado cerrar la aplicación sin dejar estas gestiones finalizadas. Por ello es necesario dejar un método bloqueado con un bucle del que debe ser posible salir para realizar el correcto cierre de los procesos.

Todos los métodos que forman parte de este servicio son explicados a continuación:

- **start_app(self)** Es el método principal, ya que contiene las tres partes fundamentales del programa, configuraciones iniciales, bucle principal y cierre. Permite mantener el servicio activo hasta salir del bucle. Es el único método privado del servicio, por motivos de seguridad.
- **boton_tecla(self,key)** Este método contiene el bucle característico de la comunicación por Websocket. El parámetro que recibe es un string con el código de la tecla pulsada. Comprueba que sigue habiendo conexión con el dispositivo servidor y le envía el comando.
- **send_key(self)** Método que contiene el protocolo de comunicación para realizar el envío del código de la tecla seleccionada. Es llamado por el bucle de Tornado, el cual finaliza al terminar de ejecutar todas las sentencias de este método.
- **actualizar(self,value)** Es el encargado de realizar una búsqueda automática de dispositivos con direcciones IP en un rango de que está comprendido entre la dirección 192.168.1.15 a 192.168.1.25, comprobando que tengan abierto el puerto 1888, y por lo tanto permitiendo una conexión con ellos. Una vez comprobada la conexión, se realiza un protocolo de comunicación para obtener la información pertinente del dispositivo.
- **lookfor(self)** Método que contiene el protocolo de comunicación para realizar una toma de contacto con el servidor, y obtener toda la información pertinente. Es llamado por el bucle de Tornado, el cual finaliza al terminar de ejecutar todas las sentencias de este método.
- **cargar_lista(self,value)** Realiza una búsqueda de las direcciones almacenadas en el archivo *list*, alojado en el directorio raíz.
- **buscar_IP(self, IP, mode)** Realiza una búsqueda de una dirección IP determinada, y agrega el dispositivo en caso de realizar una conexión.
- **AutoDeviceChange(self,value)** Realiza el cambio de dispositivo determinado al siguiente almacenado en la memoria de NAOqi. Este método solo es llamado desde los diálogos, ya que facilita el cambio de dispositivo por medio de comandos por voz.

- **DeviceChange(self,device)** Realiza el cambio de dispositivo determinado al *dispositivoN*, que es pasado como parámetro.
- **boton_exit(self,value)** Método que fuerza la salida del bucle principal, permitiendo al método **start_app(self)** continuar con el cierre de la aplicación.
- **IPchange(self,IP)** Cambia la dirección IP predeterminada de envío por la dirección establecida por el parámetro del método.

4.1.3 Descripción del control por medio de la tablet

Tal como se explica en la sección 2.1.1.5, Pepper posee una tablet con la que se comunica con el usuario. Esta tablet, en el momento de ejecutar los métodos apropiados del servicio *ALTabletService*, carga el archivo **index.html**, el cual su diseño y programación se han realizado con HTML5, CSS y JavaScript.

Dado que la aplicación de la tablet es básicamente una página web, se ha realizado con la ayuda de una librerías llamadas Materialize [15]. Estas librerías, contienen los archivos ejecutables y de estilos adecuados, para simplemente con código HTML conseguir un estilo y animaciones propias de las aplicaciones Android.

Dado que gracias que a Materialize se encarga de la estética y apariencia de la aplicación web, solo queda explicar dos archivos fundamentales para comprender el funcionamiento de la tablet de Pepper en este proyecto. Un archivo fundamental es el *index.html*, el cuan contiene toda la estructura de la aplicación web, y otro es el archivo *functions.js*, el cual se encarga de conectar la aplicación web con NOAqi, e interactuar con el servicio **IRcontrol**.

El archivo *index.html*, es el archivo principal que contiene la estructura básica de la aplicación web en HTML. Al igual que todos los archivos HTML, se divide en dos partes, la llamada cabecera o *header*, en donde se realizan y cargan los archivos de estilo CSS, y el cuerpo o *body*, el cual contiene todos los elementos que aparecen en la web, como el título, los botones, etc.

4.1.3.1 Explicación del contenido del archivo *fuctions.js*

JavaScript es un lenguaje de programación orientado a objetos,basado en prototipos, débilmente tipado y dinámico. En web, es el encargado de realizar las tareas, mostrar u ocultar objetos, suscribirse a los eventos del panel táctil y por lo tanto gestionas las entradas y salidas de la aplicación web.

En este proyecto, el archivo *functions.js*, es el encargado de sincronizar la aplicación web con el robot Pepper, por médio de NAOqi, por lo que en primer lugar, se debe realizar la conexión con Pepper, y posteriormente se realizan todos las escuchas o *EventListener* de los diferentes botones.

También se encarga de añadir y eliminar elementos como la lista de aparatos, la lista de dispositivos, la pantalla del proceso de búsqueda de dispositivos, etc. Tal como se muestra en el diagrama de la figura 4.4.

A continuación se procede a explicar las diferentes funciones que componen este archivo:

- `$( document ).ready()`

(a) Pantalla principal vacía. (b) Pantalla del proceso de actualización. (c) Pantalla principal.

(d) Lista de dispositivos. (e) Botones correspondientes al aparato.

Figura 4.3: Apariencia y colocación de la aplicación web

Figura 4.4: Diagrama de procesos del archivo *fuctions.js*

Esta función se ejecuta al cargar el archivo, por lo que da comienzo a la aplicación web. En esta función en primer lugar se cargan las plantillas de los diferentes aparatos, en donde se encuentra el código HTML correspondiente de cada aparato. A continuación se realiza la conexión a NAOqi.

- `listenerCreate()`

  Esta función es llamada desde la anterior, por lo que ya es posible añadir los eventos de escucha de todos los botones. Además se ejecutan también las funciones:

  - `subsNuevo()`

    Esta función se suscribe a la variable *IR_control/Dispos/nuevo* que lanza un evento cada vez que se agrega un dispositivo nuevo, por lo que esta función añade un botón nuevo que corresponde al nuevo dispositivo, así como su evento de escucha correspondiente.

    * `eventDispTablet(dispositivo)`

Realiza la suscripción al evento de la variable *IR_control/Dispos/dispositivo***N**, siendo N el número correspondiente a ese nuevo dispositivo agregado.

- `subsActualizando()`

  Esta función se suscribe al evento que genera la variable *IR_control/actualizando*, por lo que se encarga de vaciar la pantalla principal y mostrar los elemento que aparecen en la figura 4.3b.

- `subsdipositivo()`

  Esta función se suscribe a la variable *IR_control/dispositivo*, que contiene el nombre del dispositivo mostrado en la pantalla, por lo que si se cambia de dispositivo, esta función se encarga de cambiar toda la información y contenido al que se cambia.

- `eventKeysTablet(argument)`

  Esta función se encarga de añadir los eventos de escucha de los botones que pertenezcan a la clase especificada por la variable *argument*.

4.1.4 Descripción del control por voz

El otro modo de controlar el comportamiento del robot en esta aplicación, es mediante sentencias y ordenes por voz. Tal como se ha explicado en la sección 3.2.6, este robot tiene la capacidad de ser controlado por medio de diálogos, basados en árboles de decisiones.

Como se ha explicado anteriormente, NAOqi permite programar dichos árboles de decisiones por medio de scripts. Dichos scripts poseen una sintaxis que permite estructurar el árbol, establecer las condiciones de activación de un hilo, controlar variables, eventos y servicios de NAOqi.

En este proyecto, el árbol de decisiones programado corresponde al diagrama mostrado en la figura 4.5. En él, se puede apreciar como va progresando el dialogo dependiendo de la entrada de voz que reciba el robot.

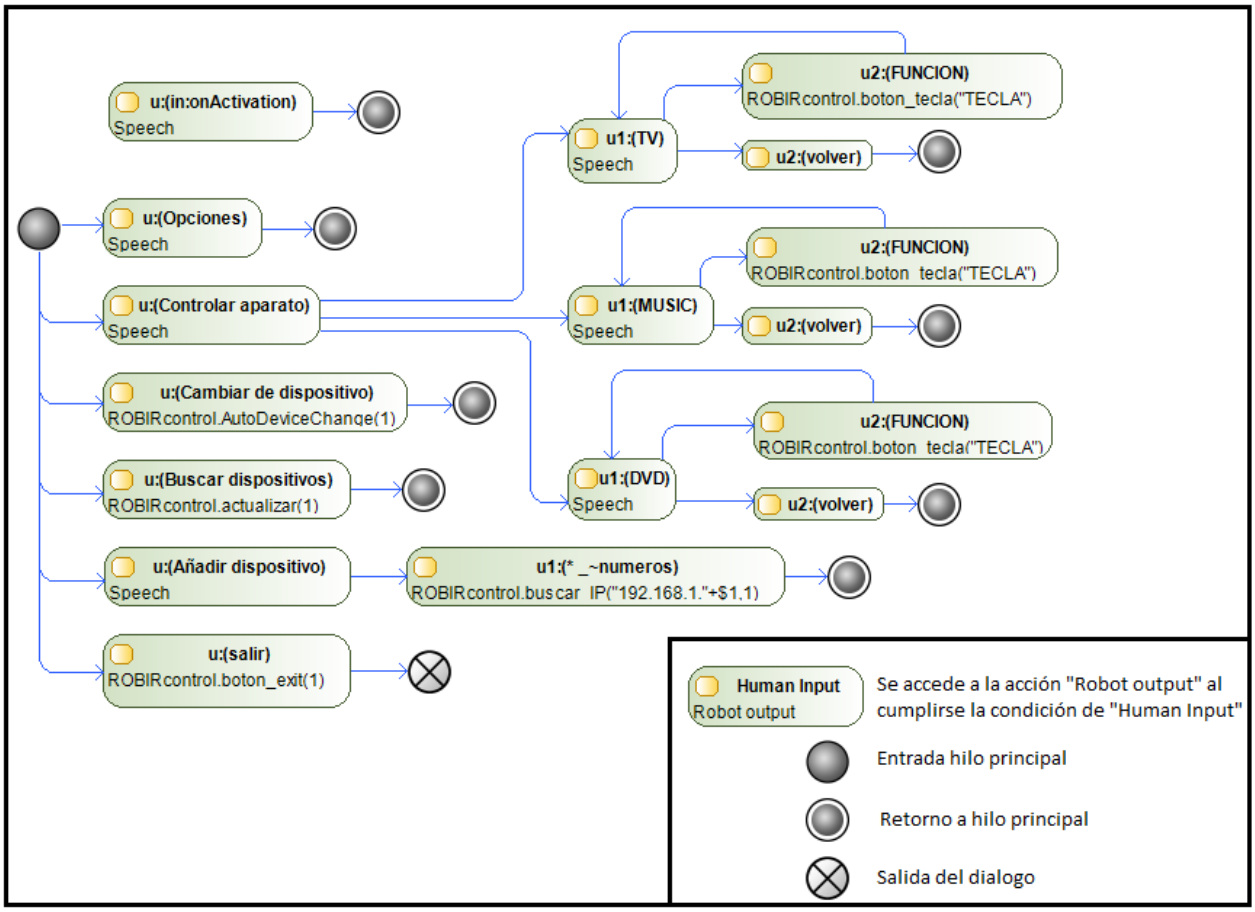


Figura 4.5: Árbol de decisiones correspondiente a la aplicación desarrollada.

A continuación se procede a explicar todos los elementos de la sintaxis utilizados para el desarrollo de este script:

- **u:()** Elemento de identificación y definición de un hilo del árbol de decisiones. La entrada de voz emitida por el usuario o bien la suscripción a un evento, se establece entre los paréntesis. Mientras que la salida correspondiente a la acción del robot corresponde a las sentencias o frases escritas posteriormente a los paréntesis.

- **uX:()** Mientras que **u:** representa las opciones del hilo principal, un número al lado como **u1:**, representa el subnivel del hilo correspondiente, tal como se muestro en la figura 3.3.

- **u:([X Y "a b c"...])** Cuando existen varias opciones validas para ir progresando por el árbol de decisiones, se deben poner entre corchetes dentro del paréntesis y separar unas de otras mediante un espacio. Si la opción corresponde a una frase que ya contiene espacios, se debe introducir entre comillas dobles.

- **u:(e:NombreEvento)** De esta manera, es posible acceder a un hilo que se desee comenzar al activarse un evento.

- **u:(in:onActivation)** El hilo que posea este método de activación comenzará cuando el dialogo sea cargado por el programa principal.

- **u:(* _~concept) $variable/del/sistema = $1** Una forma de de agrupar varias opciones, sobre todo cuando son numerosas como por ejemplo una lista de números o nombres, es mediante los

denominados conceptos o *concept*. La respuesta se almacena en una variable que se denomina como **$1**, por lo que es posible guardar la información en una variable del sistema.

- **%PuntoRetorno y ^gotoReactive(PuntoRetorno)** Cuando un hilo termina automáticamente se regresa al hilo principal, por ellos si se desea ir a cualquier otro hilo o nivel se debe poner % antes del nombre del punto de retorno, y se debe llamar desde otro hilo con la sentencia **^gotoReactive()**, colocando entre los paréntesis el nombre del punto al que se desea retornar.

- **^call(Servicio.método(parametros))** Una de las acciones que puede realizar el robot en un script de dialogo, es el poder ejecutar un método de cualquier servicio activo en el robot. Esta sentencia resulta muy útil, ya que permite la comunicación con los diferentes módulos sin necesidad de acceder a memoria.

- **^run(), ^start(), ^stop() y ^wait()** Estas sentencias permite introducir animaciones durante los discurso que realiza el robot. ^run ejecuta la animación y continua cuando esta acaba, por lo que no permite realizar discursos mientras realiza la animación. Por otra parte, ^start permite al robot reproducir una frase mientas realiza la animación, siendo esta frase cerrada por ^stop si se desea realizar la animación solo hasta que termine de hablar, o ^wait, en cuyo caso continua la animación hasta que esta termine, independientemente si ha terminado la frase.

4.2 Descripción de funcionamiento del dispositivo de control remoto

Para el desarrollo del dispositivo de control remoto, se ha decidido utilizar como elemento de procesamiento principal un ordenador Raspberry Pi Zero W. Como se ha explicado en la sección 2.2, este mini ordenador, es una potente placa de procesamiento que funciona bajo una distribución Linux. Esto ha facilitado el desarrollo hardware y software del dispositivo, ya que debido a su bajo coste, su bajo consumo y una gran comunidad de desarrolladores, es una mejor opción que cualquier micro controlador del mercado.

4.2.1 Desarrollo hardware

El desarrollo hardaware consiste en una serie de módulos y circuitos conectados a la Raspberry Pi Zero W, los cuales realizan las tareas de emisión y recepción de comandos por infrarrojos, tareas de gestión de energía y tareas de encendido y apagado. También se cuenta con una pantalla LCD, que mostrará la dirección IP del dispositivo así como su nivel de batería.

Los diferentes módulos que se han utilizado están explicados en las secciones 2.2.2.2, 2.2.2.3 y 2.2.2.4; por lo tanto, una vez se conozca su funcionamiento, simplemente hay que integrarlos en el proyecto, tal como se muestra en la figura 4.6, y realizar las configuraciones y programas software necesarios para su correcto funcionamiento.

En cuanto al diseño del bus I2C, es necesario mencionar que los dos módulos que se comunican con la Raspberry Pi Zero W, ya poseen resistencias de pull-up, por lo que no es necesario su incorporación en el bus I2C.

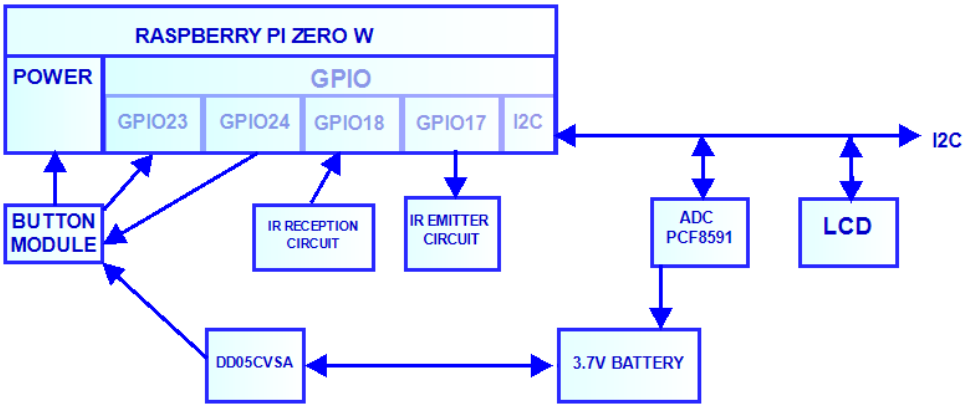


Figura 4.6: Diagrama de bloques del dispositivo desarrollado.

La principal razón del uso en este proyecto de la Raspberry Pi Zero W, a parte de su precio, es su bajo consumo. Debido a que no realiza trabajos con mucha carga, es posible realizar un *underclock*[8], el cual consiste en bajar la velocidad del micro procesador, y por lo tanto su consumo baja. En la figura 4.7 se puede apreciar el consumo que posee una Raspberry Pi Zero W. Pese a que existen otros modelos con menor consumo, el Zero W, es el que posee comunicaciones inalámbricas.

	Zero	Zero W	A+	A	B+	B	Pi2B	Pi3B
	/mA	/mA	/mA	/mA	/mA	/mA	/mA	/mA
Idling	100	120	100	140	200	360	230	230
Loading LXDE	140	160	130	190	230	400	310	310
Watch 1080p Video	140	170	140	200	240	420	290	290
Shoot 1080p Video	240	230	230	320	330	480	350	350

Figura 4.7: Comparativa del consumo de los diferentes modelos de Raspberry[8].

4.2.1.1 Cálculo de las resistencias del circuito emisor

El circuito para diseñar el emisor de IR y poder controlarlo desde un puerto configurado previamente (sección 3.3.1), es necesario un circuito que garantice que el LED emita a la potencia adecuada para un correcto funcionamiento sin dañar ningún elemento.

Dado que el LED utilizado necesita una corriente y voltaje más elevado que el que permite proporcionar una salida GPIO perteneciente a la Raspberry Pi Zero W, es necesario diseñar un circuito que sea alimentado a una diferencia de potencial de 5V y con una corriente superior a los 10mA. Basándose en el circuito ya diseñado[9], se procede a demostrar la elección de los componentes y su correcto funcionamiento.

Los componentes seleccionados para la realización del circuito de amplificación son dos resistencias de 1/2W de potencia y un transistor NP2222[16][9], los cuales se demuestra a continuación que cumplen las especificaciones. El LED de infrarojos utilizado es el IR333[17].

Tal como se muestra en la figura A.1[9], el transistor está configurado para funcionar como conmutador, ya que la señal que deberá emitir el LED es digital. Según la documentación del fabricante del LED

IR333[17], las especificaciones de funcionamiento requieren una corriente mínima de 10mA y una máxima de 100mA, pudiendo soportar picos de hasta 1A. Por ello, tal como se muestra en las ecuaciones 4.3, 4.1 y 4.10, esto limita la resistencia R1 a un margen de valores, por lo que se adopta el valor de 220Ω, el cual se aducua a los rangos. Para ello es necesario realizar los cálculos polarizando el transistor en la zona de saturación, el cual según la documentación[16] posee una ganancia de 200.

- Ecuaciones de condición del transistor en saturación, condición de voltaje de saturación del transistor V_{CEsat}[16] y voltaje V_{OHmin} de los GPIO de la Raspberry Pi Zero W[18]:

$$V_{CEsat} = 0{,}4V \tag{4.1}$$

$$V_{OHGPIO17min} = 2{,}4V \tag{4.2}$$

$$V_{CEsat} \leqslant V_{CC} - V_{IR333C} - I_C * R1 \tag{4.3}$$

$$I_C < \beta * I_{Bmin} \tag{4.4}$$

- Rango de valores de R1 según las ecuaciones anteriores y los datos proporcionados por los fabricantes[16][17]:

$$1{,}2V \leqslant V_{IR333C} \geqslant 1{,}5V \tag{4.5}$$

$$10mA \leqslant I_{IR333C} \geqslant 100mA \tag{4.6}$$

$$I_{IR333C} = I_C \tag{4.7}$$

$$R1 \geqslant \frac{V_{CC} - V_{IR333Cmax} - V_{CEsat}}{I_C max} \tag{4.8}$$

$$R1 \leqslant \frac{V_{CC} - V_{IR333Cmin} - V_{CEsat}}{I_C min} \tag{4.9}$$

$$31\Omega \leqslant R1 \leqslant 340\Omega \tag{4.10}$$

- Rango de valores de R2, suponiendo $R1 = 220\Omega$ y por lo tanto $I_C = 18mA$, para forzar una corriente I_{Bmin} que garantice trabajar en la zona de saturación:

$$I_{Bmin} = \frac{V_{OHGPIO17min}}{R2} \tag{4.11}$$

$$R2 \leqslant 26{,}67k\Omega \tag{4.12}$$

Como es demostrado en las ecuaciones 4.11 y 4.12, el valor de R2 no debe exceder los $27k\Omega$, por lo que se ha decidido utilizar una resistencia R2 de $10k\Omega$, tal como se muestra en la figura 4.6.

4.2.2 Desarrollo software

El software desarrollado para el dispositivo consiste en dos scripts en lenguaje Python que al ser ejecutados, uno controla y gestiona el servidor Websocket, mientras que el otro script se encarga de realizar el control de batería y representación de datos en el LCD.

4.2.2.1 Script de gestión del servidor websocket

Este script es el encargado de gestionar el servidor Websocket gracias a las librerías de Tornado, explicadas en la sección 3.4. En este script, se ha desarrollado un programa capaz de controlar el

dispositivo a control remoto desde un robot Pepper que tenga instalada la aplicación desarrollada en este proyecto [4.1], o bien, también permite su control desde cualquier explorador web que soporte la tecnología de Tornado Websocket.

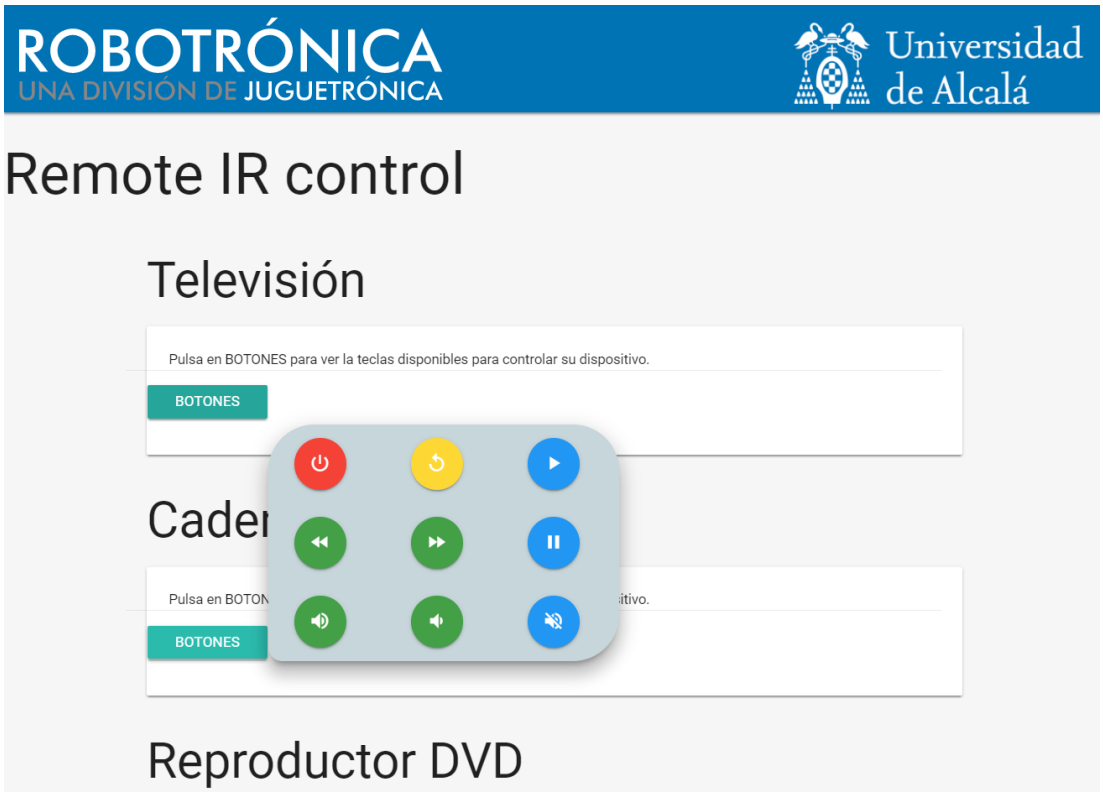


Figura 4.8: Entorno web del control del dispositivo de control remoto.

Una de las partes en las que se puede dividir este software corresponde a la parte back-end del servidor, el cual realiza la gestión de las conexiones de usuarios, la interpretación de comandos recibidos por el cliente, y por lo tanto la gestión de la información a intercambiar entre cliente y servidor, y la ejecución de las sentencias oportunas para la comunicación por IR.

Esta parte pertenece a la clase *WebSocketHandler*, y posee los siguientes métodos:

- **open(self, *args)** Se ejecuta cuando un cliente abre una nueva conexión con el servidor. Almacena los datos oportunos del cliente en una base de datos.
- **on_message(self, message)** Este método se ejecuta cuando se recibe un nuevo mensaje del cliente. Por lo que permite analizar el mensaje y realizar las funciones que pida el cliente.
- **send_mss(self,user,message)** Envia un mensaje a un solo usuario.
- **on_close(self)** Método al que se accede al cerrar la conexión entre servidor y cliente. Borra toda la información del cliente almacenado.

Por otro lado, en cuanto a la gestión front-end, este script referencia todos los archivos necesarios para que el cliente pueda realizar el control sobre el emisor de infrarrojos. Por ello, permite que el cliente se descargue todos los archivos de del servidor.

4.2.2.2 Script de gestión del bus I2C

Este es el script desarrollado en lenguaje Python, el cual realiza la lectura oportuna del voltaje de la batería, y tras realizar los cálculos oportunos muestra el nivel de batería en %. Además se encarga de mostrar en una pantalla, controlada también por el bus I2C, la dirección IP del dispositivo. Gracias a la información mostrada al usuario, este es capaz de saber si es necesario realizar un proceso de carga de las baterías, o bien poder obtener información del estado de conexión de la Raspberry Pi Zero W.

La estructura que posee el script consiste en una serie de funciones entre las que destaca una función principal. A continuación se explica el funcionamiento de todas las funciones que conforma el programa:

- **main()** Función principal del programa. Es llama al iniciar la ejecución del script. En ella se encuentra el bucle principal del programa.
- Funciones correspondientes al LCD[19]:
  - **lcd_init()** Realiza las sentencias oportunas para inicializar la pantalla LCD.
  - **lcd_byte(bits, mode)** Se encarga de enviar y recibir los comandos y datos por el bus I2C al LCD.
  - **lcd_toggle_enable(bits)** Realiza el protocolo pertinente para la comunicación I2C. Esto permite enviar y recibir la información necesaria del dispositivo LCD.
  - **lcd_string(message,line)** Escribe una cadena de caracteres en una de las filas de la pantalla LCD.
- Funciones correspondientes al ADC:
  - **leeINPUT(X)**[6] Lee el dato analógico del ADC correspondiente, estando este comprendido entre el 1 y el 4.
  - **calculo()** Esta función toma los datos de los 4 ADC, y realiza una media que se guarda en un array, pudiendo así sacar la mediana del array (el valor central), y poder representar una medida correcta del voltaje de la batería en %.
  - **insertBufferValue(x)**[6] Inserta un nuevo valor al buffer correspondiente y lo ordena de menor a mayor, devolviendo el array ordenado.
- **leerIP()** Lee la dirección IP de la Raspberry Pi Zero W, la cual viene fijada por el archivo */etc/-network/interfaces*, previamente configurado para forzar una dirección IP fija.

4.3 Funcionamiento global del proyecto

Una vez que ya se ha desarrollado todo el hardware y software necesarios, se procede a explicar el funcionamiento global del proyecto, que no es otro que el proceso de comunicación entre el robot Pepper[1] y el dispositivo diseñado en base a la placa Raspberry Pi Zero W[2].

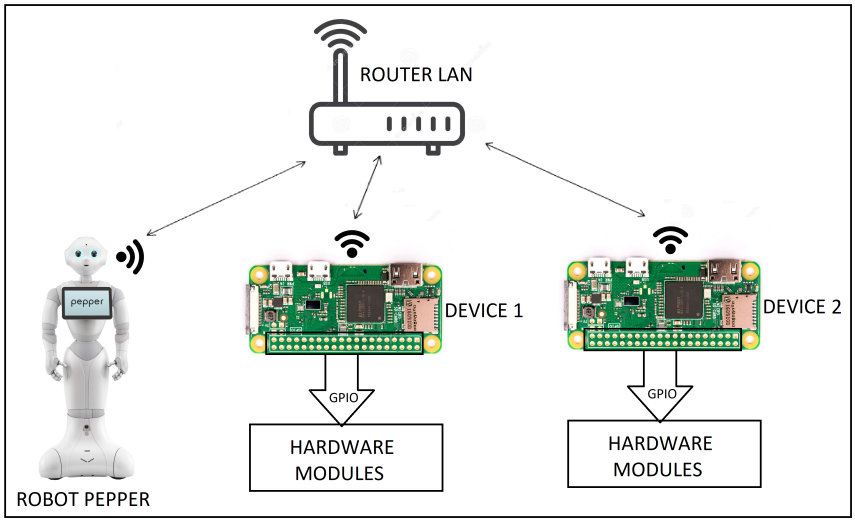


Figura 4.9: Diagrama general del proyecto.

En primer lugar para que las aplicaciones desarrolladas funcionen correctamente, es preciso que ambos elementos (el robot y el dispositivo) estén conectados a una misma red local. En este proyecto se ha elegido un rango de direcciones IP para el dispositivo de 192.168.1.15 a 192.168.1.25, siendo necesario cumplir este requisito para que el robot encuentre los dispositivos automáticamente.

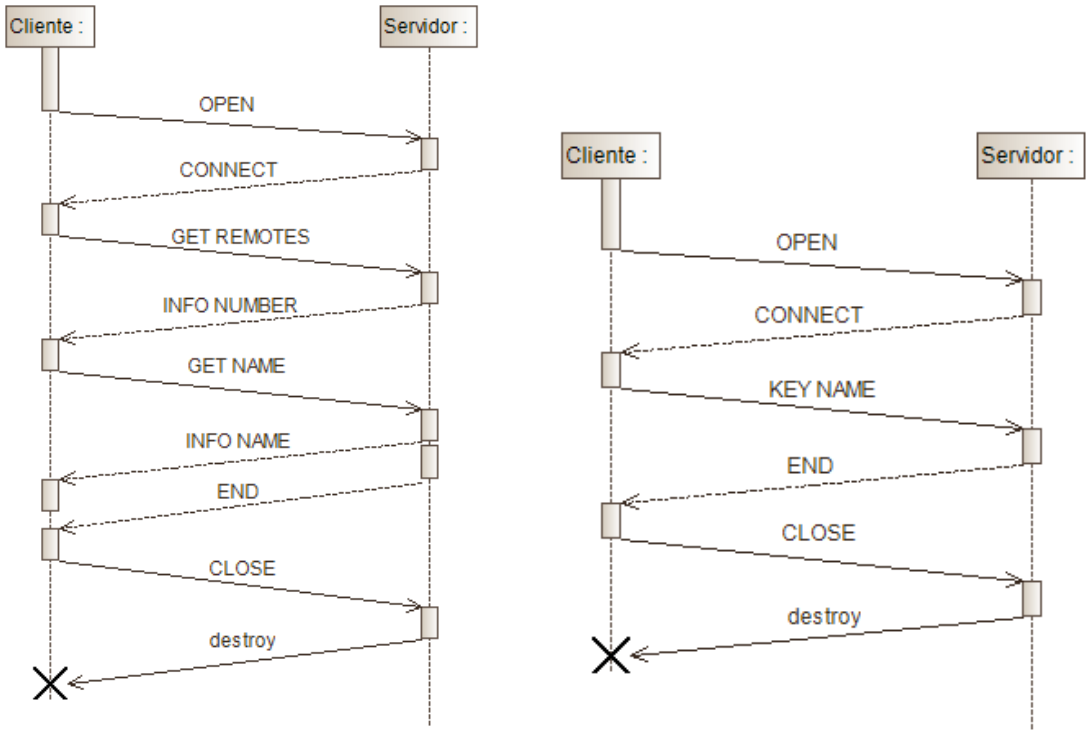
Tal como se muestra en la figura 4.9, es posible tener varios dispositivos en una misma red, siempre y cuando no compartan la misma dirección IP.

También el sistema permite varios robots Pepper[1] en la red de área local(LAN), ya que al comportarse estos como clientes y al ser sus direcciones IP asignadas mediante DHCP, no hay posibilidad de que los servidores puedan generar errores en ese sentido.

Para una correcta comunicación entre servidor(dispositivos basados en la Raspberry Pi Zero W) y cliente(robot Pepper y clientes web), se ha desarrollado una serie de protocolos de comunicación para el intercambio de información entre clientes y servidores, tal como se muestra en la figura 4.10.

Existen dos protocolos diferentes, el primero (figura 4.10a) es utilizado únicamente por la aplicación del robot, ya que este necesita conocer una serie de características del dispositivo al que se conecta, para así almacenar su información y poder utilizarla para enviar los comandos que controlan los diferentes aparatos. Esa información consiste en la cantidad de aparatos que puede controlar el dispositivo (televisión, cadena de música o reproductor DVD) y el nombre que posee el dispositivo. Toda esta información es utilizada en la *tablet* del robot para representar los diferentes elementos.

El otro protocolo (figura 4.10b), es utilizado por todos los clientes que se conectan por websocket al servidor. Consiste en el envío del comando de la tecla pulsada, dependiendo del aparato que se desee controlar. Este protocolo es utilizado tanto por la aplicación del robot como por los clientes conectados al servidor mediante web.



(a) Protocolo de comunicación para la obtención de la información del dispositivo encontrado.

(b) Protocolo de comunicación realizado para el envío de una tecla pulsada desde el cliente.

Figura 4.10: Protocolo de comunicación desarrollados.

Capítulo 5

Conclusiones y líneas futuras

5.1 Conclusiones

Se ha desarrollado una serie de aplicaciones software que han permitido una interconexión segura entre los diferentes elementos de un entorno IoT, entre los que se han incluido al robot humanoide Pepper, desarrollado por Softbank Robotics.

Esta interconexión ha sido posible gracias a Tornado Websocket, unas librerías para python, que ya vienen instaladas en el robot. Tal como se ha demostrado durante este proyecto, la conexión websocket, permite realizar de forma segura una comunicación en tiempo real y de larga duración entre un cliente y un servidor. También permite, que mediante un explorador web conectado a un servidor, el cliente pueda ver una aplicación web que se comunica con el servidor, permitiendo el la comunicación entre el servidor y cualquier dispositivo movil o PC.

Para realizar las gestiones de cliente websocket, de control de la tablet y del árbol de diálogos, ha sido necesario realizar una aplicación sobre el robot Pepper. Finalmente se ha optado por realizar la aplicación abriendo un nuevo servicio sobre NAOqi, y permitiendo así una reducida utilización del programa Choregraphe, el cual resulta un inconveniente a la hora de ejecutar aplicaciones sobre el robot. Por todo ello, la utilización de la aplicación como un servicio, ha permitido eliminar los fallos en la aplicación por culpa de las malas gestiones realizadas por los bloques propios de Choregraphe.

En cuanto al sistema hardware desarrollado, su principal componente es una Raspberry Pi Zero W, ya que es un potente ordenador en miniatura, de bajo consumo y bajo coste, el cual es capaz de realizar las gestiones oportunas de un servidor websocket, y a su vez, poder controlar los puertos GPIO de manera eficiente. Por todo ello se ha optado por utilizar este mini ordenador en oposición a un microcontrolador, como es por ejemplo el módulo ESP8266, que pese a poseer un tamaño más reducido, su consumo no es mucho más bajo al de la Raspberry Pi Zero W y su potencia excesivamente baja para realizar las tareas oportunas en las diferentes capas de la red. La principal razón por la que se ha optado por esta placa en oposición a otras menos potentes, es para garantizar una conexión segura y duradera a la red, y permitir así crear un precedente para realizar en un futuro proyectos con hardware más complejo que un emisor y receptor de IR.

Por último, debido al reducido tamaño de la Raspberry Pi Zero W y su bajo consumo, se ha optado por desarrollar a nivel hardware, un sistema que permita convertir a este dispositivo en un módulo portátil, por lo tanto ha sido necesario incluir y conectar diferentes módulos para poder gestionar el uso y control de una baterías de ion-litio.

5.2 Líneas futuras

El principal objetivo de este proyecto es demostrar y facilitar los conocimientos necesarios para poder ampliar el hardware del robot Pepper sin perder la garantía por realizar modificaciones en su hardware. Por ello, aunque en este proyecto la única modificación ha sido un sistema de control remoto por infrarrojos, es posible aplicar la misma solución para cualquier elemento con el que se desee ampliar las funciones de Pepper.

También es necesario mencionar la posibilidad de realizar de otra manera la comunicación entre el robot Pepper y el ordenador Raspberry Pi Zero W, la cual consiste en compilar el framework NAOqi en el sistema operativo Raspbian.

NAOqi solo está disponible para microprocesadores de 64 bits, no se encuentra disponible para microprocesadores ARM11, por lo tanto es necesario realizar la compilación de todas las librerías de NAOqi para su utilización en este tipo de microprocesadores. Aunque una vez conseguido, permitiría una conexión, control y acceso a todos los servicios y métodos del robot, permitiendo así un control total sobre el robot.

Es un requisito fundamental mencionar que, debido a la corta duración de la conexiones entre robot y dispositivo, y a la simpleza del protocolo de comunicación y de los mensajes intercambiados, no ha sido necesario recurrir a la compilación de las librerías de NAOqi sobre microprocesadores ARM. Ya que resulta más simple que el robot se conecte con el servidor cada vez que se desee realizar alguna acción.

Parte V

PLIEGO DE CONDICIONES

Pliego de condiciones

5.3 Introducción

En este apartado se evaluaran las condiciones para poner en marcha el software y hardware que se ha especificado en los apartados anteriores. Los cuales permiten a un usuario tomar el control de los aparatos controlados por IR mediante el robot desarrollado por Aldebaran y SOftbank Robotics.

5.4 Requisitos de hardware

5.4.1 Requisitos de hardware recomendados

Estos requisitos son necesarios para implementar un sistema comunicación entre Pepper y un hardware externo.

- PC de 64 bits, con lector de tarjetas micro SD.
- Utilización de al menos 1 Gb de memoria RAM.
- Robot Pepper con hardware V1.8.
- Raspberry Pi Zero W V1.1.
- LED IR333.
- Transistor NP2222/NP2222A
- LCD de 16x2 con comunicación por I2C.
- Batería de ion-litio de 3.7V y 1800 mAh.
- Módulo PCF8591.
- Módulo DD05CVSA.
- Resistencias de 220Ω y 10$K\Omega$
- Módulo con pulsador apagado/encendido para Raspberry Pi.

5.5 Condiciones hardware

El módulo de expansión de hardware para Pepper que se propone es un emisor y receptor de IR, el cual permite controlar varios aparatos de un entorno doméstico.

Es recomendable utilizar un robot Pepper con una versión hardware V1.8 ya que este modelo posee una *tablet* más potente que la anterior versión, y permite un correcto funcionamiento de la aplicación.

Es necesario también utilizar como base del módulo de expansión hardware una placa Raspberry Pi Zero W, ya que permite una fiabilidad en la conexión y la gestión de recursos. Para completar las funcionalidades de la placa, se recomienda utilizar un LCD de 16x2 comunicado por I2C, un módulo ADC que se comunique por I2C y un pulsador que permita un apagado y encendido seguro de la placa.

Para el montaje del emisor de IR, es necesario un transistor y un LED que emita a 940nm de longitud de onda.

5.6 Requisitos de software

5.6.1 Requisitos de software recomendados

- Utilización de un sistema operativo Ubuntu 15.04 o superior.
- Robot Pepper con software V2.55.
- Raspbian 4.9.
- Librería `naoqi`.
- Librerías `Materialize`.
- Librería `Tornado`.
- Librería `smbus`.
- Librería `numpy`.
- Librería `LIRC`.
- Para el desarrollo de la aplicación sobre el robot, es necesario utilizar las librerías desarrolladas por la empresa JUGUETRÓNICA SL.

5.7 Condiciones software

Es recomendable utilizar un sistema operativo LINUX Debian, dado que permite conexión con el robot y la Raspberry Pi Zero W por comunicación SSH, lo que posibilita un mayor control sobre el proyecto.

También se requieren varias librerías para programar en Python el websocket, la comunicación I2C. Además que al tener instalado en la placa Raspberry Pi Zero W el sistema operativo Raspbian, posibilita

la instalación del sistema LIRC.

5.8 Condiciones generales

La utilización de este sistema de control domótico con el robot Pepper, requiere una serie de condiciones para su correcto funcionamiento:

- Red WiFi a 2.5GHz y a 5GHz con conexión a Internet.
- El robot Pepper y el dispositivo desarrollado deben estar en la misma subred con rango de direcciones entre 192.168.1.2 hasta 192.168.1.154.
- El dispositivo desarrollado o módulo de expansión hardware, es autónomo por medio de una batería. Por ello es necesario cargarla con un cargador de entre 4.8V a 8V a 500mA.
- El el dispositivo tiene una apertura de emisión de IR de 20° y un alcance máximo en vacío de hasta 10m sin errores en la transmisión.

Parte VI

PRESUPUESTO

Presupuesto

5.9 Equipo de trabajo

Para la realización del proyecto se va a necesitar un Ingeniero de Telecomunicaciones.

5.10 Timing

Las fases necesarias para la realización del desarrollo son las siguientes.

- Configuración del hardware (0,5 meses)
  - Instalación de los sistemas operativos, librerías, IDE. (0,25 meses)
  - Configuración y puesta en marcha. (0,25 meses)
- Diseño e implementación de los módulos software y hardware necesarios (5 meses):
  - Diseño e implementación del protocolo de comunicación (1 mes).
  - Diseño e implementación de la aplicación de sobre el robot Pepper (3 meses).
    - Diseño e implementación del servicio (1.5 meses).
    - Diseño e implementación de la aplicación web de la tablet de Pepper (1 mes).
    - Implementación del árbol de decisiones correspondiente a la interactuación por voz (0.5 meses).
  - Diseño e implementación de las aplicaciones sobre la placa Raspberry Pi Zero W (1 mes).
    - Diseño e implementación de la aplicación de servidor websocket (0.75 meses).
    - Diseño e implementación de la aplicación de control del bus I2C (0.25 meses).
- Documentación

Las fases de diseño, desarrollo y documentación son cíclicas y abarcan todo el periodo de la vida del proyecto de 5,5 meses.

5.11 Recursos

Los recursos utilizados para este proyecto son los siguientes.

- Amortización del Robot Pepper V1.8 para entorno académico durante $\Rightarrow \frac{14{,}400{,}00€}{6 \text{años de amortización} * 12 meses} * 5,5 meses de proyecto = 1100,00€$

- Amortización del ordenador ASUS A53S $\Rightarrow \frac{599,90€}{5 \text{años de amortización} * 12 meses} * 5,5 meses de proyecto = 45,83€$
- Raspberry Pi Zero W $\Rightarrow 9,00€$
- LCD I2C de 16x2 $\Rightarrow 6,19€$
- Módulo pulsador apagado/encendido $\Rightarrow 15,91€$
- Módulo ADC por I2C $\Rightarrow 6,90€$
- Transistor NP2222 $\Rightarrow 0,55€$
- LED IR333 $\Rightarrow 1,9€$
- Placa de prototipado y pines y materiales electrónicos $\Rightarrow 6,17€$

5.12 Presupuesto total

Por todo lo anterior, supone una duración de 5,5 meses naturales con un coste de 8.250 €(sin IVA), y un coste de materiales y recursos de 1.183,82 €(sin IVA). Lo que hace un total de coste del proyecto de 9.433,82 €(sin IVA).

Parte VII

MANUAL DE USUARIO

Manual de usuario

5.13 Introducción

Este manual ha sido especialmente diseñado para el uso y estudio del Trabajo Fin de Grado titulado **DESARROLLO DE DISPOSITIVOS DE CONTROL REMOTO CONTROLADOS POR ROBOT CON CAPACIDAD DE INTERACCIÓN HUMANA** realizado por **MIGUEL ESCRIBANO GUMIEL**.

En el manual, se explica como se deben encender y configurar los diferentes elementos para el correcto funcionamiento del proyecto. Es importante seguir los puntos paso a paso, para que la aplicación no falle.

5.14 Manual

5.14.1 Conexión y puesta en marcha del robot Pepper

En esta sección, se explica como se debe configurar e instalar la aplicación sobre el robot, para así poder disfrutar de ella.

5.14.1.1 Encendido del robot

Para arrancar el robot en primer lugar se debe desactivar el botón de seguridad del robot, tal como se muestra en la figura 5.1.

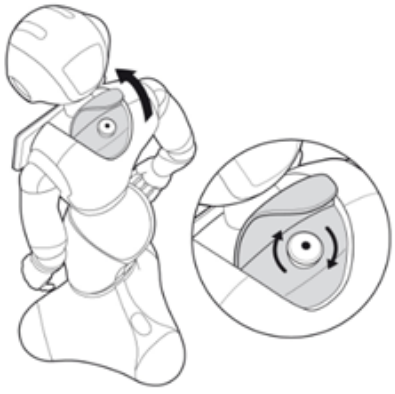


Figura 5.1: Botón de seguridad del robot.[1]

A continuación se pulsa dos veces seguidas para el arranque normal, tal como se muestra en la figura 5.2. Si se deja pulsado hasta que las luces de los hombros empiecen a parpadear, se realizará una limpieza completa de la memoria del robot o *hard reset*, sin perdida de información.

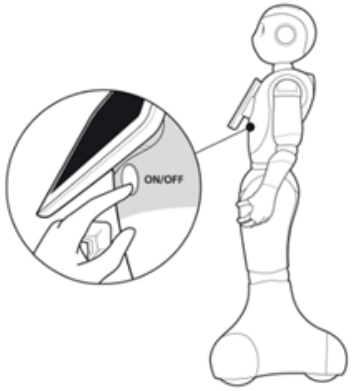


Figura 5.2: Botón de encendido del robot.[1]

5.14.1.2 Conexión del robot Pepper a la red LAN

Para conectar a Pepper a la red por primera vez, es necesario conectarle al router por cable ethernet, y desde un ordenador conectado a la misma red local, escribir en el navegador web el host `pepper.local` o la dirección de IP, la cual es dicha por el robot al pulsar una vez en el botón del pecho (figura 5.2).

Una vez accedida a la pagina web web de Pepper, se accede al apartado de conexión, figura 5.4, y se selecciona la red WiFi deseada, la cual pedirá sus correspondientes credenciales en caso de estar protegida (figura 5.3).

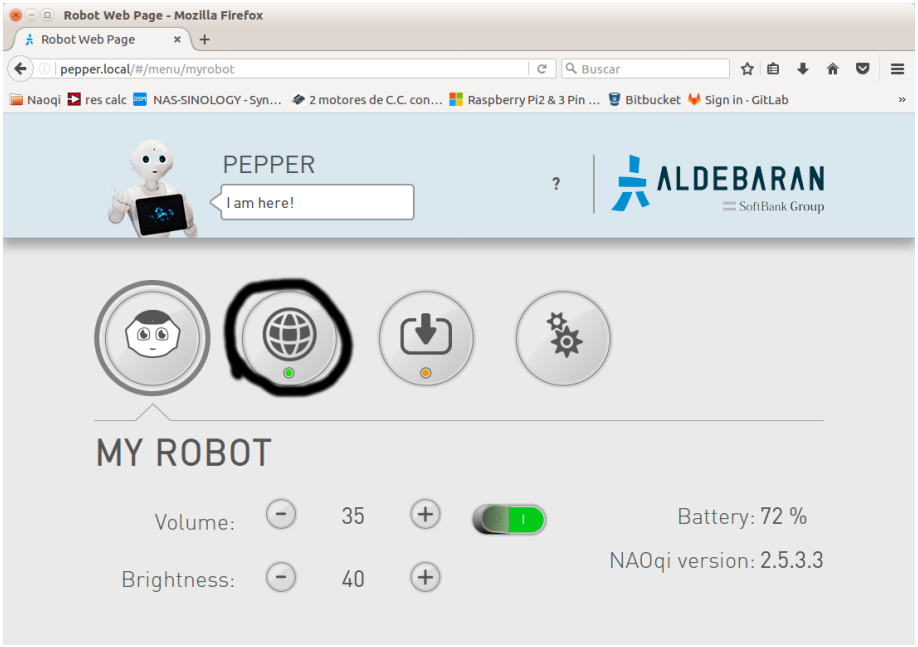


Figura 5.3: Página web principal de Pepper.

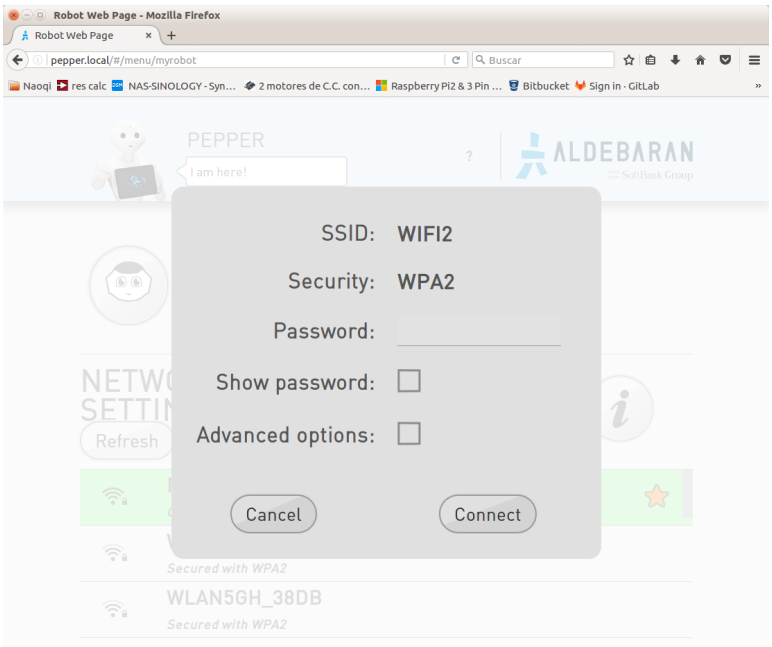


Figura 5.4: Configuración de la conexión a la red desde la página web.

5.14.1.3 Instalación de la aplicación en el robot Pepper

En primer lugar se debe conectar Choregraphe al robot Pepper tal como se muestra en las figura 5.5.

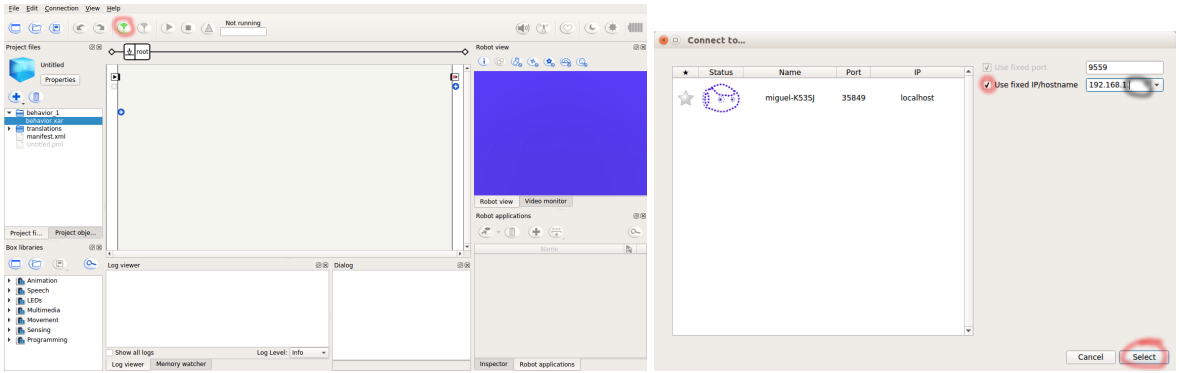


Figura 5.5: Sincronización de Pepper con Choregraphe.

Posteriormente se carga el programa en Choregraphe (figura 5.6).

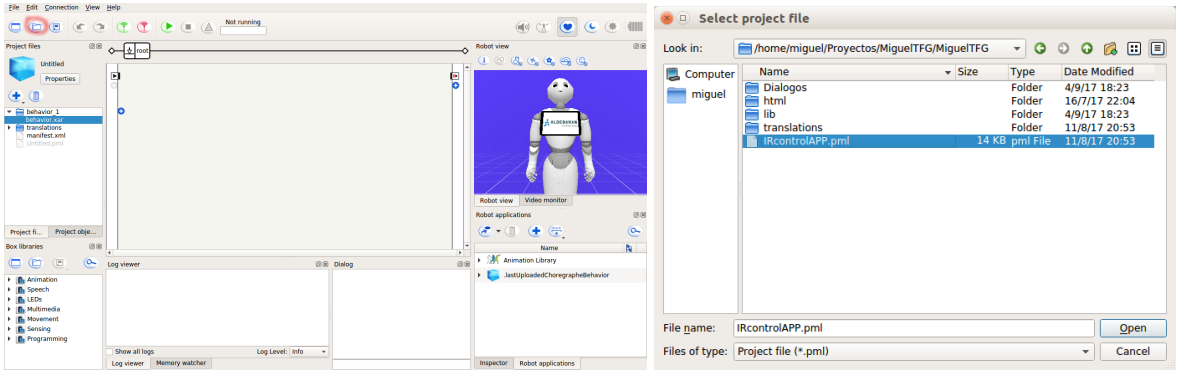


Figura 5.6: Carga del proyecto en Choregraphe.

A continuación se configura un *triger* para arrancar la aplicación al saltar un evento, en el caso de la figura 5.7 sería al tocar la mano izquierda.

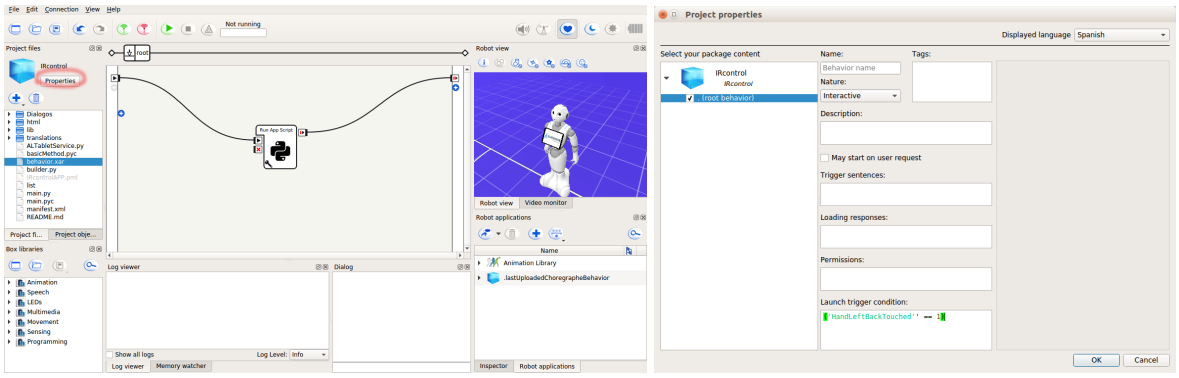


Figura 5.7: Configuración del *triger*.

Para finalizar con la instalación, se carga la aplicación tal como se muestra en la figura 5.8

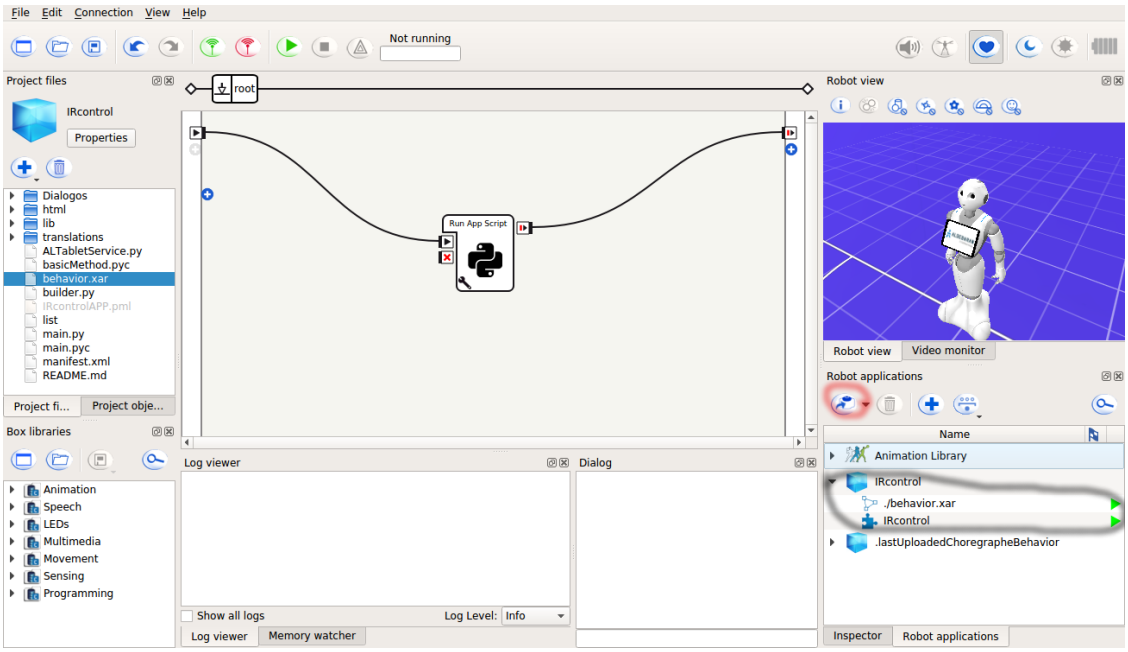


Figura 5.8: Instalación de la aplicación en el robot.

5.14.2 Conexión y puesta en marcha del módulo de expansión de hardware

En esta sección se explica como configurar y conectar a la red local (LAN), el dispositivo desarrollado.

5.14.2.1 Conexión del dispositivo a la LAN

Debido a que la Raspberry Pi Zero W no incluye ningún conector de ethernet RJ45, es necesario forzar su conexión a una red WiFi modificando directamente dos archivos.

En primer lugar, es necesario extraer de la placa la tarjeta SD y leerla desde un PC, como se muestran en la figura 5.9.

Figura 5.9: Lectura de la targeta micro SD donde se encuentra el sistema operativo Raspbian instalado.

A continuación se modifican, desde una terminal y con derechos de superusuario, se modifican los archivos `/media/username/root0/etc/network/interfaces` y `/media/username/root0/etc/wpa_supplicant/wpa_supplicant.conf`, de la manera mostrada en las figuras 5.10 y 5.11.

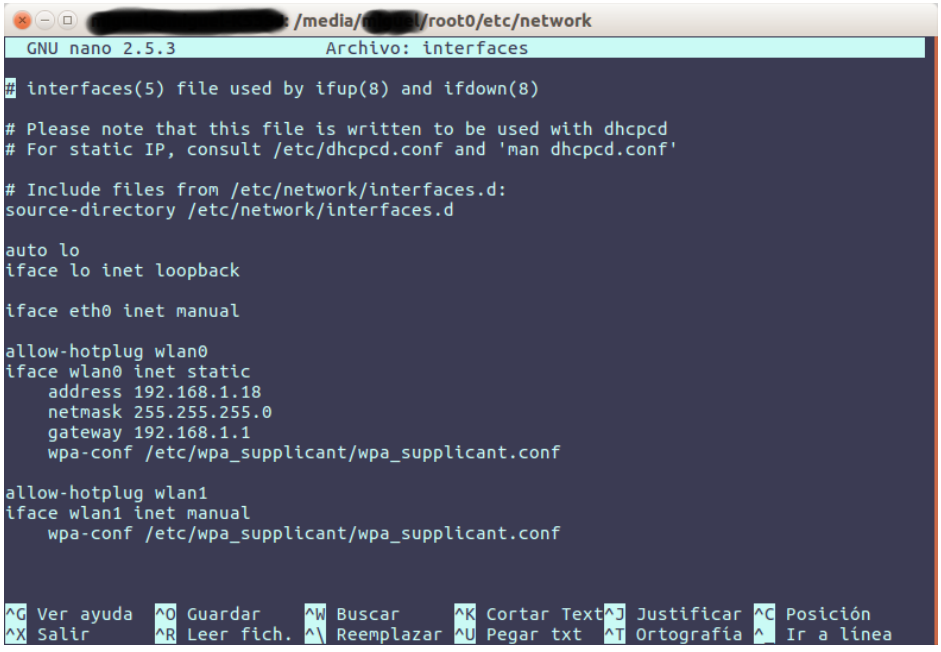


Figura 5.10: Configuración de la configuración de red de la Raspberry Pi Zero W.

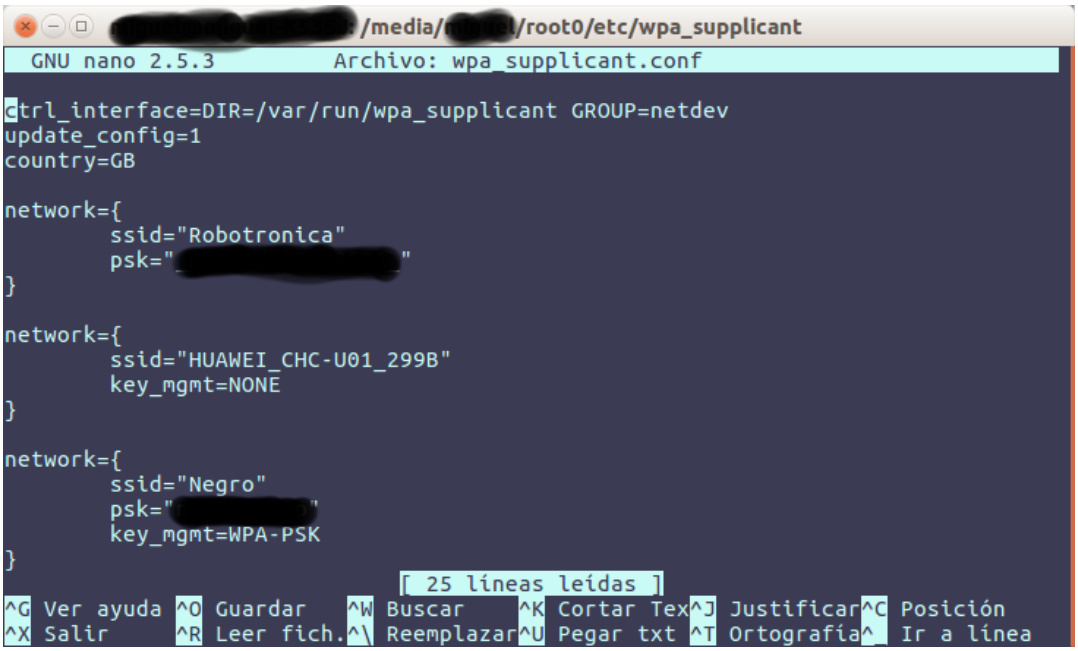


Figura 5.11: Conexión a una red WiFi de la Raspberry Pi Zero W.

Una vez modificados los archivos, se debe volver a introducir la tarjeta SD en la Raspberry Pi Zero W, tal como se muestra en la figura 5.12.

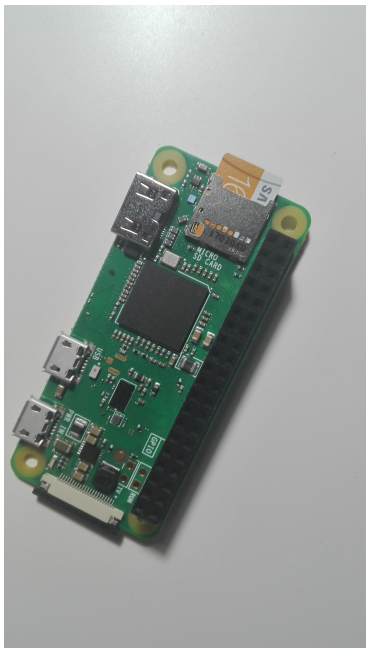


Figura 5.12: Instalación de la tarjeta micro SD en la placa del dispositivo.

5.14.2.2 Encendido del dispositivo

Tanto para encender como para apagar el dispositivo, se debe pulsar una única vez el botón de encendido/apagado instalado en el módulo (figura 5.13).

Figura 5.13: Botón de encendido/apagado del dispositivo.

5.14.3 Manual de uso de la aplicación y el módulo de expansión hardware

Una vez arrancada la aplicación en el robot, y puestos en marcha todos los dispositivos de control remoto deseados en el entorno, se pasa a explicar el funcionamiento general mediante las tres formas diferentes de control del entorno. Una consiste en commandos por voz dirigidos al robot, otra manera de controlar los diferentes dispositivos es mediante la *tablet* instalada en el pecho del robot, y por último desde un explorador web conectando directamente al dispositivo deseado.

5.14.3.1 Control del dispositivo por ordenes de voz desde el robot

Para controlar la aplicación mediante comandos de voz al robot Pepper, es necesario utilizar los siguientes comandos y en el orden adecuado.

- **¿QUE OPCIONES TENGO?** Recita todas las opciones que se pueden realizar mediante comandos de voz.
- **CONTROLAR APARATO** Permite controlar los diferentes aparatos.
  - **TELEVISIÓN** Controla la televisión.
    - **CAMBIAR DE CANAL**
      - ***NÚMERO*** Cambia de canal al número seleccionado.
    - **SUBIR VOLUMEN**
    - **BAJAR VOLUMEN**
    - **SIGUIENTE CANAL**
    - **CANAL ANTERIOR**
    - **APAGAR TELEVISIÓN**
    - **ENCENDER TELEVISIÓN**
    - **ACTIVAR MUTE**

    - **DESACTIVAR MUTE**
    - **VOLVER** Retorna al menú principal.
  - **CADENA DE MÚSICA** Controla la cadena de música.
    - **REPRODUCIR**
    - **PAUSAR**
    - **SUBIR VOLUMEN**
    - **BAJAR VOLUMEN**
    - **SIGUIENTE CANCIÓN**
    - **CANCIÓN ANTERIOR**
    - **APAGAR**
    - **ENCENDER**
    - **ACTIVAR MUTE**
    - **DESACTIVAR MUTE**
    - **REPETIR**
    - **VOLVER** Retorna al menú principal.
  - **REPRODUCTOR DVD** Controla el reproductor DVD.
    - **REPRODUCIR**
    - **PAUSAR**
    - **AVANZAR**
    - **REBOBINAR**
    - **SIGUIENTE CAPITULO**
    - **CAPITULO ANTERIOR**
    - **APAGAR**
    - **ENCENDER**
    - **ACTIVAR MUTE**
    - **DESACTIVAR MUTE**
    - **ABRIR MENÚ**
    - **EXPULSAR DISCO**
    - **CERRAR DISQUETERA**
    - **VOLVER** Retorna al menú principal.
- **CAMBIAR DE DISPOSITIVO** Selecciona otro dispositivo que controlar.
- **AÑADIR DISPOSITIVO** Añade un nuevo dispositivo.
- **BUSCAR DISPOSITIVOS** Actualiza la lista de dispositivos.
- **SALIR** Sale de la aplicación del robot.

5.14.3.2 Control del dispositivo mediante la tablet del robot

Los diferentes botones y pantallas que se muestran en la *tablet* del robot están representadas y explicadas en las figuras 5.14, 5.15, 5.16, 5.17, 5.18 y 5.19.

Figura 5.14: Pantalla vacía, se muestra cuando no se ha encontrado ningún dispositivo.

Figura 5.15: Botones de añadir dispositivos, actualizar lista de dispositivos y salir de la aplicación.

Figura 5.16: Lista de aparatos que el dispositivo puede controlar.

Figura 5.17: Pantalla mostrada al actualizar la lista de dispositivos.

Figura 5.18: Lista de dispositivos encontrados en una red local.

Figura 5.19: Botones de control de los diferentes aparatos.

5.14.3.3 Control del dispositivo desde cualquier explorador web

Para acceder directamente a la aplicación web proporcionada por la conexión websocket, es necesario introducir la dirección IP del dispositivo mostrada en su pantalla y conectarse a ella a través del puerto **1888** de la siguiente manera: `192.168.1.X:1888`, y aparece una pantalla similar a la pantalla de la *tablet* de Pepper, pero sin posibilidad de controlar otros dispositivos ().

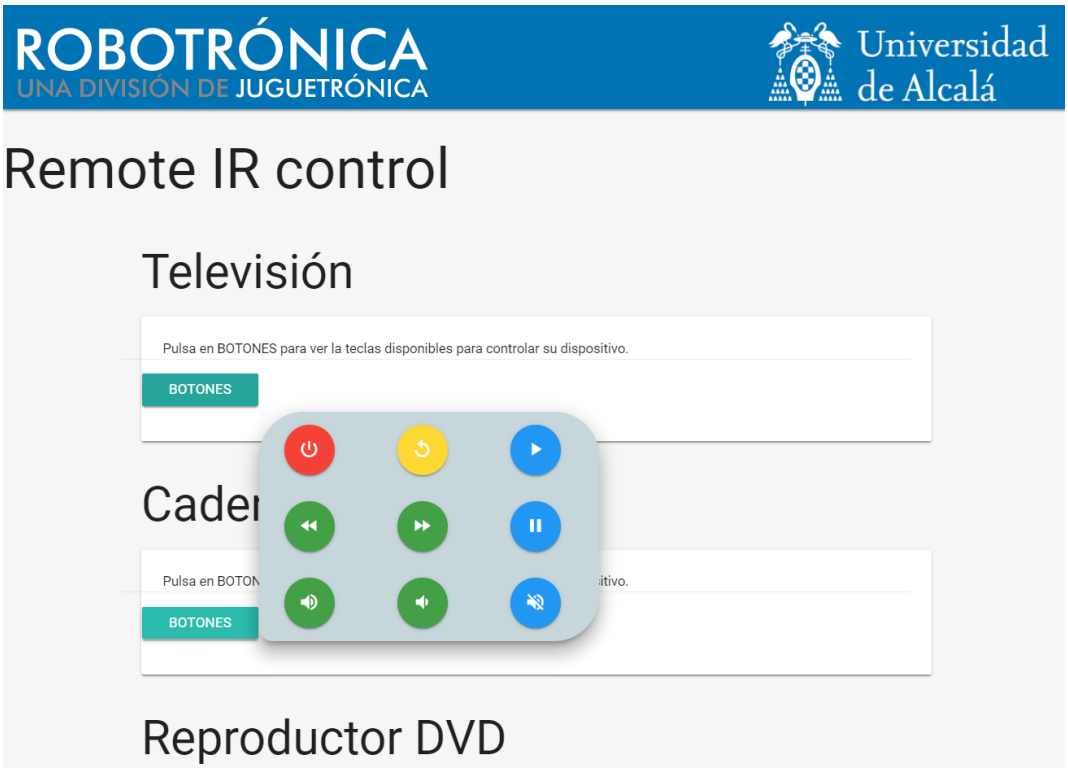


Figura 5.20: Pantalla principal de la página web del módulo de expansión hardware.

Parte VIII

BIBLIOGRAFÍA

Bibliografía

[1] "Naoqi - developer guide," SOFTBANK ROBOTICS DOCUMENTATION, http://doc.aldebaran.com/2-5/index_dev_guide.html [Último acceso 19/agosto/2017].

[2] "Raspberry pi," RASPBERRY PI FOUNDATION, https://www.raspberrypi.org/ [Último acceso 2/julio/2017].

[3] "Java i/o library for the raspberry pi," July 2016, http://pi4j.com/ [Último acceso 29/agosto/2017].

[4] L. Llamas, "Controlar arduino con un mando a distancia infrarrojo," https://www.luisllamas.es/arduino-mando-a-distancia-infrarrojo/ [Último acceso 10/agosto/2017].

[5] "Cargador y descargador de 5 v dc," Canton-powe Store, https://es.aliexpress.com/store/product/3-7V-4-2V-Charger-5V-6V-9V-12V-Discharger-Board-DC-DC-Converter-Boost-Module/2348129_32793055741.html [Último acceso 30/agosto/2017].

[6] "Raspi y entrada analógica (pcf8591)," DIVERTEKA, August 2013, http://www.diverteka.com/?p=1814 [Último acceso 1/septiembre/2017].

[7] "Illuminated led shutdown switch," Mausberry Circuits, 2017, https://www.mausberrycircuits.com/collections/frontpage/products/illuminated-led-shutdown-switch [Último acceso 15/agosto/2017].

[8] "How much power does pi zero w use," RASPBERRY PI TUTORIALS, March 2017, http://raspi.tv/2017/how-much-power-does-pi-zero-w-use [Último acceso 1/septiembre/2017].

[9] "Raspberry pi ir remote," Raspberry Pi Geek, October 2015, http://www.raspberry-pi-geek.com/Archive/2015/10/Raspberry-Pi-IR-remote [Último acceso 10/septiembre/2017].

[10] D. L. S. Chin, "Electromagnetic radiation," https://crisp.nus.edu.sg/~research/tutorial/em.htm [Último acceso 20/julio/2017].

[11] "Pcf8591 8-bit a/d and d/a converter," NXP B.V, June 2013, https://www.nxp.com/docs/en/data-sheet/PCF8591.pdf [Último acceso 3/septiembre/2017].

[12] K. S. . C. Bartelmus, "Linux infrared remote control," 1999, http://www.lirc.org/ [Último acceso 28/julio/2017].

[13] A. Bain, "Setting up lirc on the raspberrypi," http://alexba.in/blog/2013/01/06/setting-up-lirc-on-the-raspberrypi/ [Último acceso 14/julio/2017].

[14] "Tornado web server," The Tornado Authors, http://www.tornadoweb.org/en/stable/ [Último acceso 1/agosto/2017].

[15] "Documentación materialize," http://materializecss.com/ [Último acceso 13/julio/2017].

[16] "Pn2222, pn2222a general purpose transistors," PDF Datasheet, Semiconductor Components Industries, February 2010, https://www.onsemi.com/pub/Collateral/PN2222-D.PDF [Último acceso 29/agosto/2017].

[17] "Technical data sheet 5mm infrared led, t-1 3/4 ir333-a," PDF Datasheet, EVERLIGTH, Decembre 2016, http://www.everlight.com/file/ProductFile/IR333-A.pdf [Último acceso 29/agosto/2017].

[18] "Gpio electrical specifications," Mosaic Industries, http://www.mosaic-industries.com/embedded-systems/microcontroller-projects/raspberry-pi/gpio-pin-electrical-specifications [Último acceso 2/septiembre/2017].

[19] M. Hawkins, "Using an i2c enabled lcd screen with the raspberry pi," RASPBERRY PI SPY, May 2015, https://www.raspberrypi-spy.co.uk/2015/05/using-an-i2c-enabled-lcd-screen-with-the-raspberry-pi/ [Último acceso 1/septiembre/2017].

Parte IX

APÉNDICES

Apéndice A

Esquemas eléctricos

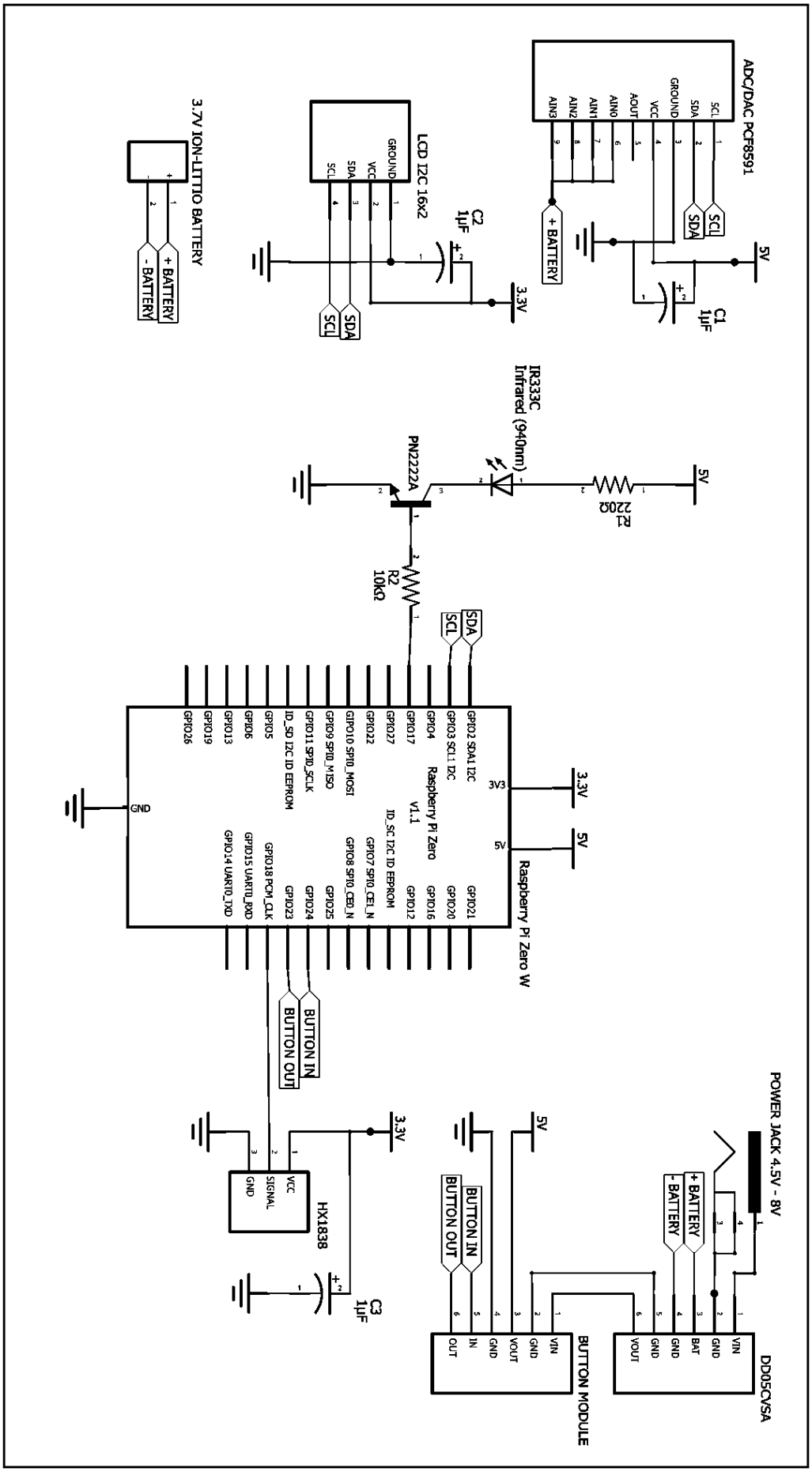


Figura A.1: Esquema eléctrico del dispositivo desarrollado.[9]

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá