Escuela Politécnica Superior

SAM Device

Dispositivo IoT para la asistencia a mayores

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Universitat d'Alacant
Universidad de Alicante

Autora:
Àlex Macià Fiteni

Tutor:
Francisco Maciá Pérez

septiembre 2021

SAM Device

Dispositivo IoT para la asistencia a mayores

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autora:
Àlex Macià Fiteni

Tutor:
Francisco Maciá Pérez

septiembre 2021

Imagen de cubierta: EPS
Diseño y maquetación: Álex Maciá Fiteni

Autora: Álex Maciá Fiteni
Tutor: Francisco Maciá Pérez

Primera edición: septiembre 2021

Versión del documento
2021.09.07.AMF

Licencia

Ningún viento será favorable para quien no sabe a qué puerto se encamina

Séneca

Resumen

En este trabajo se propone la **creación de un dispositivo físico** para facilitar un *Servicio de Acompañamiento a Mayores* (**SAM**) junto con **una plataforma software** para la síntesis, reconocimiento y generación de lenguaje natural que permite que dicho dispositivo pueda proporcionar una interfaz basada en voz y en lenguaje natural (VUI, por sus siglas en inglés: *Voice User Interface*).

Este dispositivo lo hemos denominado **SAM Device** y quedará integrado dentro de una *plataforma para el acompañamiento y cuidado de las personas mayores* (**SAM Project**) que el grupo de investigación *grupoM. Redes y Middleware* de la Universidad de Alicante está desarrollado para los servicios sociales de diferentes ayuntamientos en la cual también colaboré como becaria.

Se trata de un proyecto ambicioso que implica el diseño de hardware y de software, basado ampliamente en sistemas distribuidos y técnicas de inteligencia artificial (IA), principalmente en el procesamiento de lenguaje natural (NLP, por sus siglas en inglés: *Natural Language Processing*) ofrecido como *servicios de Internet* o *servicios en la Nube*.

El dispositivo físico o ***SAM Device*** consiste en un dispositivo inteligente, del tipo Internet de las Cosas (IoT por sus siglas en inglés: *Internet of Things*), capaz de conectarse a Internet a través de diferentes tipos de redes de comunicación, identificar y atender solicitudes de intervención, reconocer órdenes de voz en lenguaje natural, mantener sencillos diálogos con los mayores y responder igualmente mediante voz y lenguaje natural.

Tanto para el diseño del hardware como del software se ha procurado que las propuestas sean abiertas y que puedan estar a disposición de toda la comunidad. El dispositivo debe ser lo suficientemente potente como para prestar el servicio de acompañamiento que se requiere, pero lo suficientemente ajustado como para que pueda ser asequible para cualquier economía.

El proyecto también contempla el desarrollo de varios prototipos *SAM Device*, y el desarrollo y despliegue de la plataforma de IA para el reconocimiento de lenguaje natural en un entorno distribuido bajo un modelo de prestación de servicios.

Los ayuntamientos, a través de sus servicios sociales, han seleccionado un grupo de personas mayores para realizar las pruebas en sus propios hogares y realizar un seguimiento mediante su personal sociosanitario. Sin embargo, la actual situación provocada por la pandemia ha retrasado esta deseada fase, por lo que me he visto obligada a desechar este objetivo del ámbito del trabajo.

A pesar de esta y de algunas otras situaciones complicadas por la COVID, el proyecto ha podido llegar a su fin, diseñándose un conjunto de experimentos que avalan que tanto el *dispositivo SAM* como la plataforma NLP se comportan satisfactoriamente, y que se han podido integrar con facilidad en todo el ecosistema creado para el acompañamiento a mayores (**SAM Project**).

Resum

En aquest treball es proposa la **creació d'un dispositiu físic** per a facilitar un Servei d'Acompanyament a Majors (SAM) juntament amb **una plataforma programari** per a la síntesi, reconeixement i generació de llenguatge natural que permet que aquest dispositiu puga proporcionar una interfície basada en veu i en llenguatge natural (VUI, per les seues sigles en anglés: *Voice User Interface*).

Aquest dispositiu l'hem denominat **SAM Device** i quedarà integrat dins d'una plataforma per a l'acompanyament i cura de les persones majors (**SAM Project**) que el grup d'investigació grupoM. Xarxes i Middleware de la Universitat d'Alacant està desenvolupat per als serveis socials de diferents ajuntaments en la qual també vaig col·laborar com a becària.

Es tracta d'un projecte ambiciós que implica el disseny de maquinari i de programari, basat àmpliament en sistemes distribuïts i tècniques d'intel·ligència artificial (IA), principalment en el processament de llenguatge natural (NLP, per les seues sigles en anglés: *Natural Language Processing*) oferit com a serveis d'Internet o serveis en el Núvol.

El dispositiu físic o **SAM Device** consisteix en un dispositiu intel·ligent, del tipus Internet de les Coses (IoT per les seues sigles en anglés: *Internet of Things*), capaç de connectar-se a Internet a través de diferents tipus de xarxes de comunicació, identificar i atendre sol·licituds d'intervenció, reconéixer ordres de veu en llenguatge natural, mantindre senzills diàlegs amb els majors i respondre igualment mitjançant veu i llenguatge natural.

Tant per al disseny del maquinari com del programari s'ha procurat que les propostes siguen obertes i que puguen estar a la disposició de tota la comunitat. El dispositiu ha de ser prou potent com per a prestar el servei d'acompanyament que es requereix, però prou ajustat com perquè puga ser assequible per a qualsevol economia.

El projecte també contempla el desenvolupament de diversos prototips *SAM Device*, i el desenvolupament i desplegament de la plataforma de IA per al reconeixement de llenguatge natural en un entorn distribuït sota un model de prestació de serveis.

Els ajuntaments, a través dels seus serveis socials, han seleccionat un grup de persones majors per a realitzar les proves en les seues pròpies llars i realitzar un seguiment mitjançant el seu personal sociosanitari. No obstant això, l'actual situació provocada per la pandèmia ha retardat aquesta desitjada fase, per la qual cosa m'he vist obligada a rebutjar aquest objectiu de l'àmbit del treball.

Malgrat aquesta i d'algunes altres situacions complicades per la COVID, el projecte ha pogut arribar a la seua fi, dissenyant-se un conjunt d'experiments que avalen que tant *el dispositiu SAM* com la plataforma NLP es comporten satisfactòriament, i que s'han pogut integrar amb facilitat en tot l'ecosistema creat per a l'acompanyament a majors (**SAM Project**).

Abstract

In this work, **the creation of a physical device** is proposed to facilitate a Senior Support Service (SAM for its acronym in Spanish: *Servicio de Atención a Mayores*) together with a software platform for the synthesis, recognition and generation of natural language that allows said device to provide a *Voice User Interface* (VUI) based on natural language.

We have called this device **SAM Device** and it will be integrated into a platform for the accompaniment and care of the elderly (**SAM Project**) that the *GrupoM research group. Networks and Middleware* of the University of Alicante is developing for the social services of different municipalities in which I also collaborated as a scholarship holder.

It is an ambitious project that involves the design of hardware and software, based largely on distributed systems and *Artificial Intelligence* (AI) techniques, mainly *Natural Language Processing* (NLP) offered as Internet services or Cloud services.

The physical device or **SAM Device** consists of an *Internet of Things* (IoT) intelligent device, capable of connecting to the Internet through different types of communication networks, identifying and responding to requests of intervention, recognize voice commands in natural language, maintain simple dialogues with the elderly and respond equally through voice and natural language.

Both for the design of the hardware and the software, we have tried to ensure that the proposals are open and that they can be made available to the entire community. The device must be powerful enough to provide the required accompaniment service, but weak enough so that it can be affordable for any wallet.

The project also includes the development of several **SAM Device** prototypes, and the development and deployment of the AI platform for natural language recognition in a distributed environment under a service delivery model.

City Halls, through their social services, have selected a group of older people to carry out the tests in their own homes and to carry out a follow-up through their social health personnel. However, the

current situation caused by the pandemic has delayed this desired phase, which is why I have been forced to discard this objective from the field of work.

Despite this and some other situations complicated by COVID, the project has been able to come to an end, designing a set of experiments that guarantee that both the *SAM device* and the *NLP platform* behave satisfactorily, and that they have been able to integrate with ease in the entire ecosystem created to accompany the elderly (**SAM Project**).

Motivación, justificación y objetivo general

El cuidado de los mayores es un problema que he vivido de cerca por mis dos abuelas, una por parte de madre y otra por parte de padre. A su edad requieren atención constante, lo que implica que un hijo de cada una de ellas esté dedicados únicamente a su cuidado, y dada su falta de movilidad tampoco pueden ir y venir para visitar a sus otros familiares o amigos, lo que, en la práctica, las deja incomunicadas.

Esto supone un problema tanto para ellas y para gente de su edad, como para los familiares que deben cuidarlas. Este problema se agrava en aquellos casos en los que las personas mayores no cuentan con hijos o gente con el tiempo necesario para cuidar de ellas, ya que carecen de un servicio de cuidado estable que las mantenga activas y con ganas de realizar actividades. Estas energías para vivir son un elemento indispensable en un envejecimiento sano y libre de dolor (Arem et al., 2015), (Tak et al., 2013). Además, los familiares de dichos mayores que no pueden proveerles de estos cuidados, ya sea por falta de tiempo o de dinero, vivirán preocupados por no poder darles a sus padres, hermanos, tíos… el cuidado que se merecen.

Aunque existe la posibilidad de cuidar a nuestros mayores a través de los servicios sociales, estos suelen estar desbordados y sin recursos suficientes dada la gran cantidad de personas que tienen que atender y el presupuesto del que disponen (Lindberg et al., 2013). Esta situación se agrava día a día debido al cambio demográfico que se está produciendo en España, en Europa y en todo el mundo, hacia una población cada vez más envejecida en la que, para el 2025, la mayor parte tendrá más de 65 años (EC, 2017) y (WESS, 2007).

Las tecnologías pueden ayudar a mejorar el acompañamiento a los mayores, descargando de tanta responsabilidad a los familiares, proporcionando recursos sostenibles a las

administraciones y a los profesionales y facilitándoles un mejor servicio (Lindberg et al., 2013), (Wagner y Hunnerup. 2018) y (Demiris G., 2006).

En el grupo de investigación en el que he trabajado como becaria durante el curso 2018-2019, y desde entonces como colaboradora, venimos desarrollando una plataforma tecnológica para proveer un *Servicio de Asistencia a Mayores* (**SAM**) que en la actualidad se está probando en Alcoy (Alicante) y, en breve, en Torrent (Valencia), en colaboración con los servicios sociales de sus ayuntamientos.

Este sistema consiste en una plataforma de servicios en la nube que permite que los profesionales sociosanitarios incorporen información sobre las personas mayores que atienden, incluyendo aspectos como sus rutinas diarias de alimentación, ejercicio o medicación. Igualmente, los familiares responsables de los mayores pueden estar al tanto de dichas rutinas y de su cumplimiento por parte de los mayores. Pueden incluso modificar algunas de estas rutinas en función del tipo de rutina y de los permisos que se les conceda. El sistema guía a los mayores para que vayan realizando todas las tareas que tienen asignadas. A partir de la información que obtiene, el sistema puede informar a los profesionales y a los familiares sobre la evolución de los mayores y, en caso de que se requiera, alertarles sobre posibles situaciones que se consideren de riesgo. Se trata de alertas que pueden ir desde un sencillo “no se ha duchado esta mañana”, hasta cuestiones más peliagudas como “lleva todo el día sin levantarse” o “hace dos días que no se toma su medicina de las 15:00”.

Gracias a este proyecto hemos podido constatar que la interfaz del sistema con los mayores constituye un aspecto fundamental en este tipo de sistemas y servicios y que no basta con proporcionarles una interfaz gráfica convencional a través de un navegador Web ubicado en un PC o en un teléfono. Teniendo en cuenta esta idea, junto con mi motivación inicial, surgió finalmente la propuesta para este trabajo de fin de grado: *diseñar un dispositivo con interfaz de voz y lenguaje natural para los mayores, que sea lo más intuitivo posible para ellos, que tenga en cuenta sus limitaciones físicas y sus patologías, pero, al mismo tiempo, que pueda ser viable económicamente para todas las familias y sostenible por los servicios de salud públicos*.

A partir del estudio del estado actual de la técnica (véase **capítulo 2**) sabemos que hay muchas opciones comerciales para dar solución a este problema de interfaz. Sin embargo, todas ellas tienen importantes carencias y problemas que por el momento son insalvables para los intereses de las personas mayores: ni los dispositivos, ni su funcionamiento, ni los servicios que prestan,

están pensados y diseñados específicamente para las personas mayores; no están conectados ni enlazados con servicios sociosanitarios; y la información de los usuarios se envía a servidores externos, propiedad de los fabricantes, donde es almacenada un tiempo indeterminado con la justificación de entrenar los algoritmos que gestionan el sistema. Además de estos problemas, encontramos también riesgos importantes en basar la solución en desarrollos comerciales. Algunos de los que consideramos más sensibles son: condicionamiento total de las políticas de cada empresa para implementar soluciones y servicios sobre sus infraestructuras; políticas que además son cambiantes en función de los intereses del mercado; precios y tarifas poco definidas, altas o cambiantes; y políticas de seguridad y privacidad muy insuficientes, en algunos casos inexistente, para el tipo de usuario e información que se debe gestionar.

Teniendo en cuenta todo lo anterior, definimos **el objetivo general** de este trabajo como: *construir un dispositivo inteligente, capaz de asistir mediante interfaces basadas en voz y en lenguaje natural a las personas mayores y a dependientes, veinticuatro horas al día, y servir de interfaz para ayudarles en sus rutinas diarias y facilitarles la comunicación con sus familiares y con los profesionales que les atienden, mejorando así la situación de los tres grupos: mayores sin atención, familiares preocupados y profesionales saturados. Este dispositivo lo hemos denominado* ***SAM Device****.*

El dispositivo tendrá que integrarse en el ecosistema existente para la atención a mayores que ya es de por sí complejo y diverso. Además, para lograr la interfaz de lenguaje natural, dada su complejidad y necesidad de recursos de computación, deberá diseñarse como un servicio capaz de ofrecerse desde servidores suficientemente potentes y escalables, ubicados en la Nube.

En el **capítulo 4**, dedicado a *Objetivos y metodología*, se puede encontrar más detalles acerca de los objetivos concretos que se han definido para realizar el trabajo.

Agradecimientos

A mi padre y tutor, y al resto de mi familia,
por todo su apoyo incondicional.

A mi iaia y a mi abuela que han sido la
motivación principal de esta aventura.

A mis compañeros de ABP por dar todo de sí durante el curso y aguantar
mis broncas, y por esos inolvidables días de Dragones, Mazmorras y pizza.

A todo el profesorado que hemos tenido en la carrera. La cantidad de cosas que he aprendido en estos cuatro años, en tantos ámbitos de la informática, es desorbitada.

A la saga de videojuegos *The Legend of Zelda* por mantenerme
cuerda los últimos meses del trabajo con su música legendaria.

Todavía me queda mucho que aprender, mucha
música que escuchar, y mucha pizza que ingerir.
Nada de esto habría sido posible sin vuestra ayuda.

Tabla de contenido

1 Introducción 1

2 Estado de la técnica 5

2.1 El próximo cambio demográfico 6

2.2 Envejecimiento saludable 9

2.2.1 Ejercicio y buenos hábitos 10

2.2.2 Estrategia y plan de acción mundiales 11

2.2.3 Plan España 2050 11

2.2.4 Intelligent Cities Challenge (ICC) 12

2.2.5 Ambient Assisted Living (AAL) 13

2.2.6 Active and Assisted Living (AAL-2) 14

2.2.7 Uso de tecnología 14

2.3 Tecnologías: interfaces de usuario 15

2.3.1 Interfaz gráfica de usuario (GUI) 16

2.3.2 Interfaz de usuario basada en voz 17

2.3.3 Conclusiones 18

2.4 Tecnologías: asistentes inteligentes 18

2.5 Tecnologías: dispositivos inteligentes 20

2.5.1 Altavoces inteligentes 20

2.5.2 Dispositivos IoT 20

2.5.3 Robots de servicio 21

2.5.4 Conclusiones 22

2.6 Tecnologías: redes de interconexión 22

2.6.1 Conclusiones 24

2.7 Tecnologías: conclusiones 25

2.8 Inteligencia Artificial: campos y ramas 25
2.8.1 Machine Learning 26
2.8.2 Modelos de regresión 27
2.8.3 Árboles de decisión 27
2.8.4 Árboles de clasificación 27
2.8.5 Clustering 28
2.8.6 Aprendizaje profundo 28
2.8.7 Redes neuronales convolucionales 29
2.8.8 Natural Language Processing 29
2.8.9 Tecnologías de voz 29
2.9 Inteligencia artificial: procesamiento del lenguaje natural 30
2.9.1 Técnicas para NLP y algoritmos de voz 30
2.9.2 Comprensión del Lenguaje Natural (NLU) 31
2.9.3 Generación de Lenguaje Natural (NLG) 32
2.10 Inteligencia artificial: tecnologías de Voz 32
2.10.1 Reconocimiento automático de voz (ASR) 32
2.10.2 Generación automática de voz (TTS) 33
2.11 Asistentes de voz 34
2.11.1 Asistentes de voz comerciales 34
2.11.2 Asistentes de voz no comerciales 36
2.11.3 Conclusión 38
2.12 Dispositivos hardware disponibles 39
2.12.1 Arduino 39
2.12.2 Raspberry Pi 39
2.12.3 ReSpeaker 40
2.12.4 Conclusión 40
2.13 Sistemas de NLP disponibles 41
2.13.1 Wit.ai 42
2.13.2 Rasa 42
2.13.3 DeepPavlov.ai 43
2.13.4 Comparativa 44

2.14 Sistemas de ASR disponibles 44
2.14.1 Mozilla DeepSpeech 45
2.14.2 CMU Sphinx 45
2.14.3 Julius 45
2.14.4 Kaldi 45
2.14.5 Comparativa 46
2.15 Sistemas de TTS disponibles 46
2.15.1 Mozilla TTS 47
2.15.2 MARY 47
2.15.3 eSpeak 47
2.15.4 Mimic 47
2.15.5 CMU Flite 47
2.15.6 Comparativa 47
2.16 Motores de *hotword* disponibles 48
2.16.1 Porcupine 48
2.16.2 Snowboy 48
2.16.3 Comparativa 49
2.17 Conclusiones 49
3 Antecedentes y resultados previos 53
3.1 SAM Project 53
3.2 Infraestructura TI 55
3.3 Arquitectura conceptual de alto nivel 58
3.4 Front-End Level 59
3.5 API Level 62
3.6 Service Level 63
3.7 Data Level 63
3.8 Maintenance Level 64
3.9 Back-End 65
3.10 Conclusión 65
4 Objetivos y metodología 67
4.1 Objetivo 68

4.2 Metodología ... 69
5 Estudio de viabilidad ... 71
5.1 Análisis DAFO ... 71
6 Planificación ... 75
6.1 Planificación inicial ... 75
6.2 Contratiempos y plan de contingencia ... 75
6.3 Replanificación y Plan de ejecución ... 77
7 Análisis, especificación y diseño ... 81
7.1 Arquitectura conceptual de alto nivel ... 82
7.2 Front-End Level ... 83
7.2.1 Arquitectura hardware de SAM Device ... 83
7.2.2 Arquitectura software de SAM Device ... 85
7.3 API Level ... 86
7.4 Service Level ... 87
7.4.1 SAM Voice Service ... 87
7.4.2 Modelos de voz y de conversación ... 90
7.5 Data Level ... 91
7.6 Maintenance Level ... 91
7.7 Diseño de la carcasa ... 92
7.8 Integración con los servicios de SAM Core ... 93
8 Implementación ... 95
8.1 Prototipos hardware de dispositivo IoT ... 95
8.1.1 Prototipo ReSpeaker ... 96
8.1.2 Prototipos Raspberry + Seeed ... 99
8.1.3 Impresión de la carcasa ... 102
8.2 Software de dispositivo IoT ... 104
8.2.1 Configuración de la palabra de activación ... 104
8.2.2 Obtención de la orden de voz ... 106
8.2.3 Envío de la orden al servicio de voz ... 109
8.2.4 Recibimiento de la respuesta del servidor ... 110

8.2.5 Retroalimentación con leds ... 111
8.2.6 Personalización la palabra de activación ... 112
8.3 Servicios de voz ... 114
8.3.1 Arquitectura técnica ... 114
8.3.2 Arquitectura de despliegue ... 116
8.3.3 Instalaciones previas para el servicio de voz ... 116
8.3.4 Servicio de ASR ... 117
8.3.5 Servicio de TTS ... 120
8.3.6 Servicio de NLP ... 122
8.3.7 Servicio de voz: uniéndolo todo ... 123
8.3.8 Personalización del modelo de conversación ... 124
8.3.9 Despliegue en contenedores ... 130
8.3.10 Despliegue de alta disponibilidad ... 132
9 Pruebas y validación ... 137
9.1 Servicio de voz ... 137
9.1.1 Diseño de los escenarios de prueba ... 137
9.1.2 Diseño de los experimentos ... 140
9.1.3 Ejecución de las pruebas ... 141
9.1.4 Análisis de los resultados ... 142
9.2 Dispositivo IoT ... 142
9.2.1 Diseño de los experimentos ... 142
9.2.2 Ejecución de las pruebas ... 144
9.2.3 Análisis de los resultados ... 144
9.3 Comparativa de rendimiento ... 144
9.4 Pruebas en entornos reales ... 146
10 Conclusiones y trabajo futuro ... 149
10.1 Trabajo futuro ... 150
11 Referencias ... 153
12 Apéndices ... 167
12.1 Metodología de investigación para la revisión bibliográfica ... 167

12.1.1 Análisis y definición de la necesidad de información 167
12.1.2 Nivel y cobertura de la búsqueda 167
12.1.3 Selección de las fuentes de información 168
12.1.4 Elaboración de la estrategia de búsqueda 169
12.1.5 Valoración de los resultados y gestión de la información recuperada 169
12.2 Raspberry Pi con un arreglo de 4 micrófonos 172
12.2.1 Componentes principales de ReSpeaker 4-MIC Array 172
12.3 Raspberry Pi + Arreglo de 2 Micrófonos 175
12.4 ReSpeaker Core v2 177
12.4.1 Características 178
12.4.2 Especificaciones 178
12.4.3 Descripción general del hardware 179
12.4.4 Pines de salida (*pin out*) 180
12.4.5 Dimensiones 181
12.5 Glosario de acrónimos y términos 182
12.5.1 Glosario de acrónimos 182
12.5.2 Glosario de términos 184
12.6 Bibliotecas, herramientas y lenguajes utilizados 187
12.6.1 Bibliotecas 187
12.6.2 Herramientas 188
12.6.3 Lenguajes de programación 190
12.7 Código del proyecto 191
12.7.1 Software del dispositivo IoT 191
12.7.2 Software del servicio de voz 191

Índice de figuras

Figura 1. Proporción de personas de 60 años o más, por país, en 2015. FUENTE: Informe mundial sobre el envejecimiento y la salud (OMS, 2015). 7
Figura 2. Proporción de personas de 60 años o más, por país, proyecciones para 2050. FUENTE: Informe mundial sobre el envejecimiento y la salud (OMS, 2015). 7
Figura 3. Prevalencia de la inactividad física según la edad en personas de 60 años o más, por país. FUENTE: (Wave4, 2013) 10
Figura 4. Campos y ramas de la IA. FUENTE: elaboración propia. 26
Figura 5. Relación entre las tecnologías de voz, el NLP, y sus diferentes ramas y tecnologías. FUENTE: elaboración propia. 31
Figura 6. Arquitectura de *Rasa Open Source*. FUENTE: (Rasa, 2021). 43
Figura 7. Elementos que intervienen en el Servicio de Acompañamiento a Mayores. FUENTE: equipo de SAM. 54
Figura 8. Red Social (SAM Network) especializada para familiares y acompañados. FUENTE: equipo de SAM. 55
Figura 9. Infraestructura TI de SAM. FUENTE: equipo de SAM. 56
Figura 10. Arquitectura de alto nivel de SAM Service. FUENTE: equipo de SAM. 58
Figura 11. Dashboard de SAM en móviles. FUENTE: elaboración propia. 61
Figura 12. Dashboard de SAM en ordenador. FUENTE: elaboración propia. 61
Figura 13. Arquitectura API Gateway de SAM. FUENTE: equipo de SAM. 62
Figura 14. Diagrama Gantt con la planificación para septiembre. FUENTE: elaboración propia. 78
Figura 15. Tablero de Trello con las tareas realizadas hasta septiembre. FUENTE: elaboración propia. 79
Figura 16. Arquitectura de alto nivel del sistema que se propone en el TFG. FUENTE: elaboración propia. 82

Figura 17. Diagrama de bloques con los principales componentes hardware de SAM Device. FUENTE: elaboración propia. 84
Figura 18. Diagrama de flujo de la interacción con SAM Device en formato BPMN. FUENTE: elaboración propia. 86
Figura 19. SAM Voice Service High-level Architecture. FUENTE: elaboración propia. 88
Figura 20. Diagrama de actividad de SAM Voice Service. FUENTE: elaboración propia..... 88
Figura 21. Capturas de *Blender* con algunos de los diseños 3D realizados. FUENTE: elaboración propia. 92
Figura 22. Arquitectura de integración de alto nivel de **SAM Project** con el dispositivo VUI (**SAM Device**) y los servicios de voz (**SAM Voice Service**) incorporados. FUENTE: elaboración propia. 93
Figura 23. Arquitectura conceptual del dispositivo IoT (**SAM Device**). FUENTE: elaboración propia. 96
Figura 24. Dispositivo ReSpeaker Core v2. FUENTE: https://wiki.seeedstudio.com/ReSpeaker_Core_v2.0/. 97
Figura 25. Puertos del ReSpeaker FUENTE: elaboración propia. 98
Figura 26. (arriba) Raspberry Pi y placa ReSpeaker 4-Mic Array de Seeed. (abajo) Prototipo con dos micrófonos, 4 leds, altavoz, sensor de temperatura y humedad, y relé. FUENTE: (arriba) wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/. (abajo) wiki.seeedstudio.com/Snips_Voice_Interaction_Base_Kit/. 101
Figura 27. Impresora 3D *Monoprice Minidelta v2* utilizada para la creación de las carcasas de SAM Device. FUENTE: https://www.monoprice.com/..... 103
Figura 28. Captura de pantalla de la aplicación Ultimaker Cura empleada para gestionar la impresión 3D. FUENTE: elaboración propia. 103
Figura 29. Arquitectura técnica conceptual del servicio de voz (**SAM Voice Service**). FUENTE: elaboración propia. 114
Figura 30. Arquitectura de despliegue básica junto con las tecnologías involucradas. FUENTE: elaboración propia. 116
Figura 31. Interfaz para la creación de un intent en una Skill de Alexa. FUENTE: developer.amazon.com/en-US/docs/alexa/devconsole/about-the-developer-console.html. 125
Figura 32. Fragmentos del modelo personalizado para Rasa en formato YAML. FUENTE: elaboración propia. 129
Figura 33. Ejemplo de acción personalizada para *Rasa*. FUENTE: ejemplos de *Rasa*..... 130

Figura 34. Arquitectura de despliegue de alta disponibilidad junto con las tecnologías involucradas. FUENTE: elaboración propia. 133
Figura 35. Escenario de despliegue mínimo. FUENTE: elaboración propia. 138
Figura 36. Escenario de despliegue intermedio. FUENTE: elaboración propia. 139
Figura 37. Escenario de despliegue de alta disponibilidad. FUENTE: elaboración propia. 139
Figura 38. Esquema de toma de decisión sobre los elementos a determinar en las técnicas de búsqueda. FUENTE: (Biblioteca UA, 2021). 170
Figura 39. Raspberry Pi y placa ReSpeaker 4-Mic Array de Seeed. FUENTE: wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/. 172
Figura 40. Detalle de componentes de la placa de micrófonos ReSpeaker 4—Mic Array. FUENTE: wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/. 172
Figura 41. Diagrama con los esquemas de interconexión del chip AC108 y los micrófonos. FUENTE: www.cnx-software.com/2017/08/28/x-powers-ac108-is-a-quad-channel-adc-chip-for-microphone-arrays/. 173
Figura 42. Detalle de funcionamiento de los leds de la placa ReSpeaker 4-MIC. FUENTE: wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/. 173
Figura 43. Diagrama con el esquema de conexión del arreglo de leds. FUENTE: files.seeedstudio.com/wiki/ReSpeaker-4-Mic-Array-for-Raspberry-Pi/src/ReSpeaker%204-Mic%20Array%20for%20Raspberry%20Pi%20%20v1.0.pdf. 174
Figura 44. Detalle de los conectores de la ReSpeaker 4-Mic Array junto con una Raspberry Pi. FUENTE: wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/. 174
Figura 45. Diagrama con los esquemas de interconexión de los conectores y buses de la placa ReSpeaker 4-Mic Array. FUENTE: files.seeedstudio.com/wiki/ReSpeaker-4-Mic-Array-for-Raspberry-Pi/src/ReSpeaker%204-Mic%20Array%20for%20Raspberry%20Pi%20%20v1.0.pdf. 175
Figura 46. Prototipo con dos micrófonos, 4 leds, altavoz, sensor de temperatura y humedad, y relé. FUENTE: wiki.seeedstudio.com/Snips_Voice_Interaction_Base_Kit/. 175
Figura 47. ReSpeaker 2-Mics Pi HAT. FUENTE: wiki.seeedstudio.com/ReSpeaker_2_Mics_Pi_HAT/ 176
Figura 48. Placa ReSpeaker Core V2. FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/177
Figura 49. Detalle de los conectores de expansión que proporciona la placa ReSpeaker Core V2 y numeración de sus pines de salida. FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/. 180

Figura 50. Dimensiones de la placa ReSpeaker Core V2. FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/. 181

Índice de tablas

Tabla 1. Indicadores iniciales para evolucionar de un estado de bienestar a una sociedad más longeva. FUENTE: (ONPE, 2021). 12
Tabla 2. Social policy & innovation, Past projects. FUENTE: elaboración propia a partir de: www.age-platform.eu/project-topic/past-projects. 13
Tabla 3. Proyectos destacados del Programa Conjunto *Ambient Assisted Living* (AAL) 2008-2013. FUENTE: elaboración propia a partir de: www.aal-europe.eu/. 14
Tabla 4. Proyectos destacados del Programa Conjunto *Active and Assisted Living* (AAL-2) 2014-2020. FUENTE: elaboración propia a partir de: www.aal-europe.eu/. 15
Tabla 5. Comparación de bibliotecas de ASR. FUENTE: elaboración propia. 46
Tabla 6. Matriz DAFO del proyecto. FUENTE: elaboración propia. 72
Tabla 7. Planificación para junio (8 meses). FUENTE: elaboración propia. 75
Tabla 8. Replanificación para septiembre (8 meses). FUENTE: elaboración propia. 77
Tabla 9. Tabla comparativa con los tiempos medios de la ejecución de los distintos módulos del trabajo en hardware variado. FUENTE: elaboración propia. 145
Tabla 10. Tabla comparativa con los tiempos de la ejecución de los distintos módulos del trabajo en máquina virtual. FUENTE: elaboración propia. 145
Tabla 11. Concreción de los parámetros análisis y definición de necesidades de información. FUENTE: elaboración propia. 168
Tabla 12. Definición del nivel y de la cobertura de la búsqueda realizada. FUENTE: elaboración propia. 169
Tabla 13. Definición de las fuentes de información seleccionadas para la búsqueda. FUENTE: elaboración propia. 169
Tabla 14. Concreción de los parámetros estrategia de búsqueda. FUENTE: elaboración propia. 170
Tabla 15. Lista con las características principales de la placa ReSpeaker Core V2. FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/ 178

Tabla 16. ReSpeaker Core V2 **GPIO Pins**. FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/. 180
Tabla 17. ReSpeaker Core V2 **I2C Pins**. FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/. 180
Tabla 18. Tabla de acrónimos. FUENTE: elaboración propia. 182
Tabla 19. Tabla de términos. FUENTE: elaboración propia. 184
Tabla 20. Tabla de bibliotecas utilizadas. FUENTE: elaboración propia. 187
Tabla 21. Tabla de herramientas utilizadas. FUENTE: elaboración propia. 188

Memoria

1 Introducción

La atención a las personas mayores es un problema que ha existido siempre. Con la edad perdemos independencia, lo que significa que la gran mayoría de las personas mayores requieren de alguien o algo que cuide de ellas (*acompañamiento*). Esto implica unos costes de tiempo y dinero elevados que cada vez son más difíciles de cubrir. Además, como la esperanza y calidad de vida crecen a pasos agigantados (Bloom, 2011), (Christensen et al., 2009), cada vez hay más mayores y con más edad, lo que complica el mantenimiento todavía más.

Siendo este un problema que nos persigue desde hace tanto tiempo es evidente que ya existen soluciones, generalmente canalizadas por los servicios sociales, por servicios y asociaciones privadas o por el acompañamiento por parte de familiares.

Los servicios sociales son servicios del estado o de los ayuntamientos que se encargan de cuidar a nuestros mayores, proporcionarles los cuidados domésticos y médicos que necesitan, hasta mantenerlos entretenidos. El servicio prestado puede ser de atención en casa (a domicilio) o trasladar a la persona mayor a una residencia de mayores en la que estará con otras personas en su misma situación. En cualquiera de los casos dispondrán de personal dedicado a cuidarles. Esto supone el pago de los salarios, junto los recursos que se consuman, todo ello cubierto mediante el presupuesto público. El problema de este enfoque, tal y como se ofrece ahora, es que requiere mucho personal, muy dedicado y atento. Todo esto, pagado únicamente con dinero público, es difícil de mantener, o incluso insostenible.

Los servicios privados, por otro lado, los debe cubrir el usuario interesado o sus familias. La ventaja es que al ser de pago se puede disponer de más personal y de una mayor dedicación en función de lo que se esté dispuesto a pagar. El problema de esta solución es que no todas las economías familiares pueden permitírselo y, por lo tanto, la solución no es viable para toda la ciudadanía. En muchos casos se termina abusando de clases sociales desfavorecidas que pasan

veinticuatro horas trabajando y acompañando al mayor por muy poco dinero. Así, muchas familias se quedan sin una posibilidad sostenible de disfrutar de este servicio (Lindberg et al., 2013).

Por último, tenemos a los familiares que cuidan de sus propios mayores. En este caso, el único requisito es que haya una persona (o más) de confianza —normalmente familiar de la persona mayor—, que la cuide e idealmente viva con ella, y que se asegure de que esté lo más cómoda posible. La ventaja inicial de este enfoque radica en que no se debe realizar una inversión directa, al margen de los recursos consumidos en el cuidado del mayor o, en muchos casos, los ingresos que dejarán de percibir los cuidadores por reducir sus jornadas laborales, llegando en muchos a casos a tener que dejar totalmente sus trabajos. Aunque hace unas décadas era muy común que alguno de los hijos de la persona mayor se dedicara solo a cuidarla, hoy en día la mayoría de las personas trabajan o estudian, los hijos de la persona mayor están ocupados en tareas importantes para su propio mantenimiento y por ello no es posible dedicarle todo el tiempo que se requiere. Esta situación también deja a los familiares inquietos por no poder ocuparse adecuadamente de sus mayores.

Las tendencias indican que en los próximos años la mayor parte de la población mundial vivirá por encima de los 60 años, provocando una transición demográfica hacia poblaciones de mayor edad. Este envejecimiento de la población, provocado por el aumento de la esperanza de vida gracias a la medicina moderna y a los avances en las condiciones de vida, supone uno de los principales retos políticos, sociales y económicos a los que se tendrán que enfrentar las sociedades europeas del siglo XXI (Bloom, 2011), (Christensen et al., 2009). En 2025 más de un 20% de los europeos habrán cumplido los 65 años, con un aumento especialmente rápido de la población mayor de 80 años. Esto provocará un cambio de equilibrio entre generaciones que será radicalmente diferente al actual (EC, 2017-2019).

Los informes muestran que el perfil de los familiares que cuidan a los mayores también evolucionará debido al cambio demográfico, con un creciente peso en las parejas y en la participación de los hombres (Abellán et al., 2017) y (Zueras et al., 2018). El rol de cuidado y cuidador se irá intercambiando con frecuencia (Badenes y López, 2011) y (del Barrio et al., 2015), y se potenciarán las fórmulas de autocuidado basadas en innovaciones tecnológicas: telemedicina, *apps*, supervisión por Internet, adaptaciones domóticas.

Gracias a los avances tecnológicos en la salud —como el uso masivo de datos y sistemas de Inteligencia Artificial (IA), la robótica, los dispositivos de *Internet de las Cosas* (IoT), o los *wearables*— será viable a corto plazo el desarrollo de una medicina mucho más personalizada, predictiva y efectiva, especialmente en áreas terapéuticas (ONPE, 2021).

De todo lo anterior, se puede entender que es importante para la sociedad atender a sus mayores; que la esperanza de vida y la calidad de vida aumentan cada vez más y, por ello, los mayores buscan más autonomía; y que son cada vez más capaces de asimilar tecnología, y esta es cada vez más amigable. Sin embargo, también es cierto que los mayores siguen necesitando atención y cuidado; y que el actual ritmo de vida hace muy complicado, incluso imposible, que el peso del cuidado pueda recaer en sus familias. La conclusión es que las soluciones actuales son insuficientes, o si son suficientes entonces son demasiado caras y quedan al alcance de unos pocos, y que los cambios demográficos agravarán esta situación si no se adoptan medidas; que esta situación puede ser una oportunidad si se aborda con tiempo; y que las tecnologías pueden ayudar a proporcionar soluciones mucho más ajustadas a las necesidades cambiantes y, sobre todo, más asequibles y sostenibles en el tiempo.

En este trabajo se propone una solución basada en tecnología para facilitar el acompañamiento y cuidado de las personas mayores, de forma que sea mínimamente intrusiva, que sea amigable y adaptada a las necesidades de los mayores, y cuyos costes puedan ser asumidos por cualquier economía.

En esta memoria se recoge el trabajo realizado y se ha estructurado de la siguiente forma. En el siguiente capítulo (**capítulo 2.** Estado de la técnica) se realiza un exhaustivo estudio de la bibliografía existente tanto en el ámbito del envejecimiento saludable como en el de las diferentes tecnologías que pueden proporcionar soluciones y soporte para su mejora: dispositivos IoT, Inteligencia Artificial, sobre todo en el área del procesamiento de lenguaje natural, comunicaciones y plataformas tecnológicas.

En el **capítulo 3.** Antecedentes y resultados previos, revisaremos el trabajo realizado hasta ahora por el *grupoM* y por mí en el *proyecto de SAM*.

En el **capítulo 4.** Objetivos y metodología, veremos los objetivos tanto generales como concretos que se plantean para el trabajo y hablaremos de la metodología a emplear tanto en el desarrollo como en la investigación.

En el **capítulo 5.** Estudio de viabilidad, analizaremos la viabilidad del proyecto y estudiaremos los posibles riesgos de este, además de prepararnos para afrontar o minimizar dichos riesgos.

En el **capítulo 6.** Planificación, abordamos cómo se esperaba distribuir en el tiempo las tareas que componen el trabajo de fin grado.

En el **capítulo 7.** Análisis, especificación y diseño, trataremos en profundidad todas las fases relacionadas con la concepción del sistema propuesto.

En el **capítulo 8.** Implementación, veremos qué se ha implementado y cómo, tanto hardware como software, revisando partes relevantes del código y siguiendo paso a paso el proceso de desarrollo del trabajo de fin de grado.

En el **capítulo 9.** Pruebas y validación, se detalla qué pruebas se llevaron a cabo para validar y mejorar el trabajo, cómo se llevaron a cabo, sus resultados, y un pequeño análisis de estos para entender qué se puede mejorar y qué está cerrado.

En el **capítulo 10.** Conclusiones y trabajo futuro, ya habremos visto todo lo relacionado con la preparación y realización del trabajo. Hablaremos de las conclusiones que se desprenden del estudio y del trabajo realizado, reflexionando acerca de qué aspectos del trabajo pueden mejorarse en el futuro.

En el **capítulo 11.** Referencias, se presentan todas las referencias empleadas en el proyecto, relacionadas en orden alfabético.

En el **capítulo 12.** Apéndices, se proporciona información ampliada de algunas secciones de la memoria en las que, por no desviar la atención del objetivo principal, o por el tamaño que ocupaban, se decidió sintetizar.

2 Estado de la técnica

En este capítulo se realiza un amplio estudio de la técnica en el ámbito del envejecimiento saludable, centrado en la aplicación de las Tecnologías de la Información (TI) y cómo estas tecnologías están proporcionando soporte en este ámbito. Desde un punto de vista amplio, y gracias a las iniciativas de financiación europeas, las actuaciones han ido encontrando un espacio propio para desarrollarse conocido como *Active and Assisted Living* (AAL) o *vida activa y asistida.*

Dentro de este dominio, nuestro interés se focaliza en los sistemas y dispositivos que permiten asistir a nuestros mayores (seguimiento, monitorización, alertas, comunicaciones, …) y que les proporcionen una buena experiencia de acompañamiento.

Para asegurar la calidad, precisión y eficacia de la investigación se ha seguido la metodología que se explica en el **capítulo 4**. *Objetivos y metodología*, y que está ampliada en el **apartado 12.1** de los apéndices (*Metodología de investigación para la revisión bibliográfica*), donde se proporciona la concreción de los parámetros de análisis y definición de necesidades de información (**Tabla 11**), la definición del nivel y de la cobertura de la búsqueda realizada (**Tabla 12**), la definición de las fuentes de información seleccionadas para la búsqueda (**Tabla 13**), y la concreción de los parámetros de la estrategia de búsqueda (**Tabla 14**).

Concretamente, dividiremos el estado de la técnica en cuatro grandes áreas: la justificación del problema y las posibles soluciones (**apartados 2.1 y 2.2**), las tecnologías que pueden ser útiles para darle solución (**apartados 2.3** a **2.6**), las diferentes técnicas y algoritmos de inteligencia artificial relacionados con la solución que se propone (**apartados 2.8** a **2.10**) y, finalmente, las arquitecturas, tecnologías, bibliotecas de programación, plataformas y marcos de desarrollo

existentes junto con comparativas que ayudan a entender cuáles se pueden ajustar mejor para la implementación de una solución abierta (**apartados 2.11** a **2.16**).

2.1 EL PRÓXIMO CAMBIO DEMOGRÁFICO

Con una población cada vez más longeva y con una tasa de natalidad decreciente (Bloom, 2011), (Christensen et al., 2009), se prevé que para 2070 la mayor parte de la población en la Unión Europea tendrá más de 65 años (EC, 2017) y (WESS, 2007) (véanse **Figura 1** y **Figura 2**).

Este cambio en la estructura de la sociedad conllevará también cambios en cómo entendemos la vejez y su impacto en la vida de todos los seres humanos (Beard et al., 2012). A diferencia de otros eventos futuros, estos cambios son predecibles y debemos prepararnos y planificar en función de ellos.

Según la *Organización Mundial de la Sal*ud (OMS), los desafíos con los que nos encontraremos serán, a grandes rasgos, los estereotipos anticuados junto con nuevas expectativas, la diversidad en la vejez, la inequidad y los cambios en el mundo (OMS, 2015). También se apunta que es importante superar estos desafíos para fomentar el desarrollo sostenible, para mantener los derechos de las personas mayores y maximizar la productividad de estas mientras se reducen los costes del mantenimiento del envejecimiento (Report, 2012).

El acompañamiento a mayores complementa el cuidado de los mayores al acompañarlos, ayudarles a realizar compras, leerles libros o noticias, o seguir su medicación. Este acompañamiento requiere de total confianza entre acompañador y acompañado, y puede ser crucial en el desarrollo de la vejez (Le Galès y Bungener, 2019), (Casati y Donato, 2020) y (Faria et al., 2020). Lamentablemente, el esfuerzo y la confianza que requiere el acompañamiento complica su implantación para los servicios públicos y los bolsillos menos adinerados, dejando a muchas familias sin la posibilidad de este servicio (Lindberg et al., 2013).

El *Plan estratégico España 2050* (ONPE, 2021), elaborado por la Oficina de Prospectiva y Estrategia y basado en el informe España 2050, plantea el reto del envejecimiento activo.

Este informe dice que España deberá rediseñar parte del Sistema Nacional de Salud y apostar por la tecnología para conseguir que el sistema de pensiones se mantenga y lograr así que las personas mayores puedan seguir participando activamente en la economía y la vida social del país. Seguir estas indicaciones permitirá afrontar los cambios demográficos.

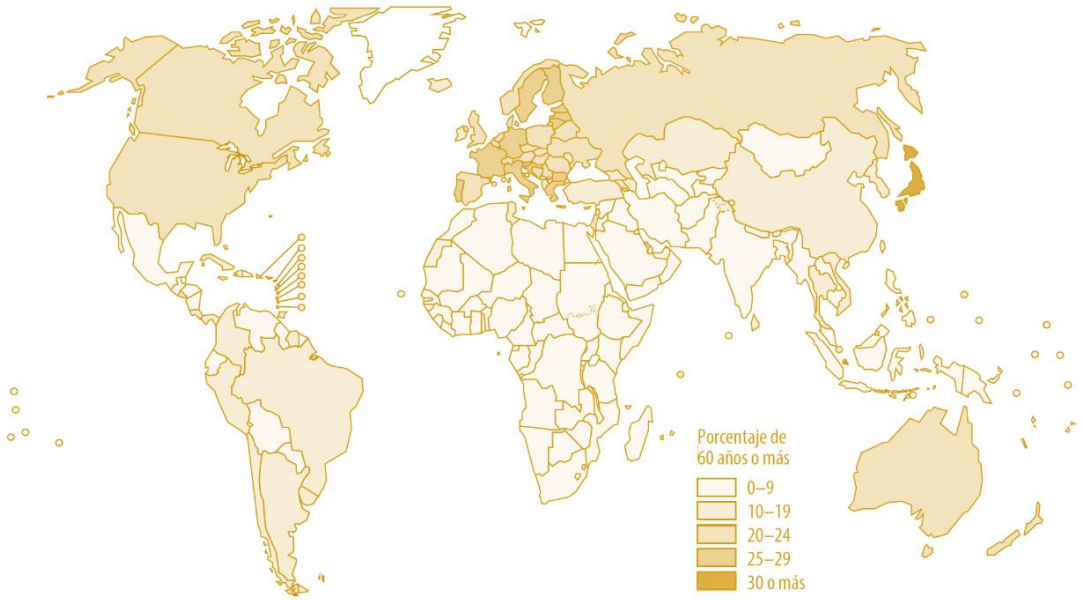


Figura 1. Proporción de personas de 60 años o más, por país, en 2015.
FUENTE: Informe mundial sobre el envejecimiento y la salud (OMS, 2015).

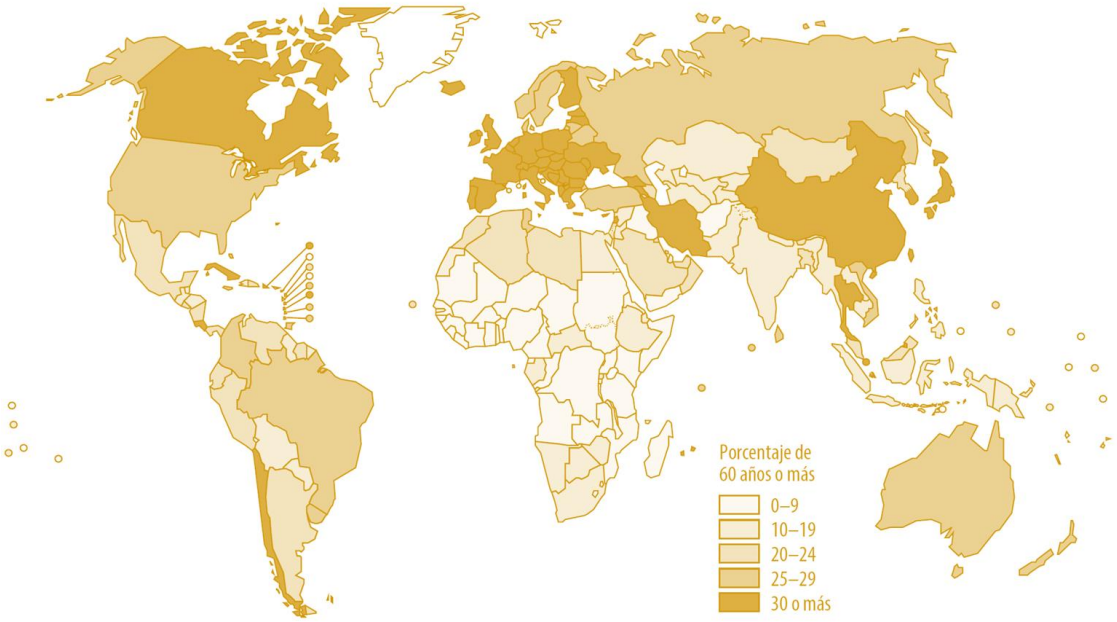


Figura 2. Proporción de personas de 60 años o más, por país, proyecciones para 2050.
FUENTE: Informe mundial sobre el envejecimiento y la salud (OMS, 2015).

Pero no solo está cambiando la cantidad de gente mayor que tendrá la población, si no cómo se vive y mantiene esa vejez. Se prevé una tendencia a preferir envejecer en casa en lugar de residencias (del Barrio y Sancho, 2016) y (Elizalde, 2018). También, cada vez más personas mayores de 65 años prefieren vivir solas (López y Pujades, 2018).

De todo esto se entiende que las residencias cambiarán su modelo, del actual institucional a un modelo centrado en cuidados en el hogar (Martínez et al., 2015). También se prevé un incremento de vida en casa con apoyo entre personas mayores de distintas familias, como los amigos (López y Estrada, 2016) o mediante *Bancos de Tiempo* y voluntariado de acompañamiento, para evitar la soledad no deseada y fomentar la participación de estas personas en la sociedad (INE, 2021), (Vega y Martínez, 2017) y (Velarde et al., 2016).

Además, ya hay informes que muestran que, cada vez más, las parejas cuidarán a sus mayores en lugar de solo los hijos. Los hombres participarán cada vez más en estos cuidados (Abellán et al., 2017) y (Zueras et al., 2018) e incluso se intercambiarán cuidador y cuidado (Badenes y López, 2011) y (del Barrio et al., 2015).

En poco tiempo será posible el autocuidado gracias a los avances en la telemedicina, la domótica, el uso masivo de datos, la Inteligencia Artificial (IA), las aplicaciones móviles y la supervisión por Internet, además de avances en la salud como la terapia génica o el uso de células madre (ONPE, 2021).

También, aunque mejoremos la tecnología y los buenos hábitos, será necesaria una mayor inversión en salud para incrementar su cobertura a la población. Esto es, evidentemente, un reto, pero también es una oportunidad que puede generar empresas y empleo.

Según el informe España 2050, nuestro país es uno de los mejores destinos del mundo para residir tras la jubilación por varias razones como nuestra posición geográfica y nuestro clima, la forma de vida que llevamos y la red de infraestructuras y transporte de la que disponemos. Esto nos coloca en una posición ventajosa para explotar el sector comercial de la atención a mayores y el turismo (ONPE, 2021).

Por todo lo mencionado, se concluye que es clave utilizar las nuevas tecnologías para asegurar un **envejecimiento saludable** y sostenible que permita abaratar los costes del cuidado y mantenimiento de los mayores y mejore dicho cuidado para lograr que este sector de la sociedad pueda seguir aportando al país y mejore su calidad de vida. Todo atendiendo a las

nuevas tendencias como ser cuidados en casa en lugar de en residencias, y permitiendo el cuidado por parte de otros mayores, el autocuidado y la asistencia a los cuidadores mediante el uso de tecnología.

En la próxima sección analizaremos los progresos hechos en materia de *Envejecimiento Saludable* (y su relación con las soluciones tecnológicas), y dedicaremos el resto del capítulo a analizar las tecnologías TI disponibles para identificar cuáles pueden ser las más adecuadas para el trabajo.

2.2 ENVEJECIMIENTO SALUDABLE

La OMS describe el envejecimiento saludable como el proceso de desarrollo y mantenimiento de la capacidad funcional que permite el bienestar en la vejez (OMS, 2015) y (OMS, 2019). También profundiza un poco más en cada uno de los términos de esta definición:

- La **capacidad funcional** comprende las capacidades que permiten a una persona ser y hacer lo que es importante para ella. Hay cinco dominios clave de la capacidad funcional, que los factores ambientales pueden ampliar (o restringir). Estas capacidades son: satisfacer las necesidades básicas; aprender, crecer y tomar decisiones; tener movilidad; establecer y mantener relaciones; y contribuir a la sociedad.
- Tener la posibilidad de vivir en entornos que apoyan y mantienen la *capacidad intrínseca* y la capacidad funcional es fundamental para el *envejecimiento saludable*. La capacidad funcional se compone de la capacidad intrínseca de la persona, las características del entorno que afectan esa capacidad y las interacciones entre la persona y esas características.
- La **capacidad intrínseca** es la combinación de todas las capacidades físicas y mentales de una persona e incluye su capacidad de caminar, pensar, ver, oír y recordar. Distintos factores influyen sobre la capacidad intrínseca como la presencia de enfermedades, los traumatismos y los cambios relacionados con la edad.
- El **entorno** comprende el hogar, la comunidad y la sociedad en general. En el entorno se encuentran una serie de factores que abarcan el entorno construido, las personas y sus relaciones, las actitudes y los valores, las políticas de salud y sociales, los sistemas que las sustentan y los servicios que prestan.

Los cambios en el envejecimiento pueden estar causados por una infinidad de razones y presentarse de muchas maneras, y pueden variar en gran medida de una persona a otra (OMS, 2016). No solo se trata de cambios físicos si no también mentales, en parte causados por los cambios biológicos en el cuerpo y otros por la manera en que el entorno entiende la vejez. Por ello, para alcanzar el envejecimiento saludable no solo es cuestión de cuidarse uno, también es cuestión de combatir los estereotipos actuales sobre lo que significa ser viejo (OMS, 2016).

2.2.1 Ejercicio y buenos hábitos

El ejercicio físico y la nutrición también son claves para envejecer saludablemente (OMS, 2016). Gran parte de los problemas en la vejez vienen siendo arrastrados por malos hábitos de cuando la persona es joven, y por ello es importante empezar a tener buenos hábitos desde el principio y mantenerlos toda la vida (Michel et al., 2008). El proyecto *Intellligent Cities Challenge* (ICC) que deriva de *Digital Cities Challenge* (DCC) hace mucho hincapié en este tema (ICC, 2020).

La actividad física tiene enormes beneficios a la hora de prevenir e incluso combatir gran parte de las enfermedades degenerativas (Paterson & Warburton, 2010) y (Tak et al., 2013). El ejercicio físico puede aumentar también la longevidad hasta en un 30%, y más en personas mayores de 60 años (Arem et al., 2015). También reduce el riesgo de perder capacidad funcional (Paterson & Warburton, 2010) y previene la perdida de función cognitiva (Jak, 2012). Sin embargo, en general, la gente hace menos ejercicio al hacerse mayor (Bauman et al., 2016) (véase **Figura 3**).

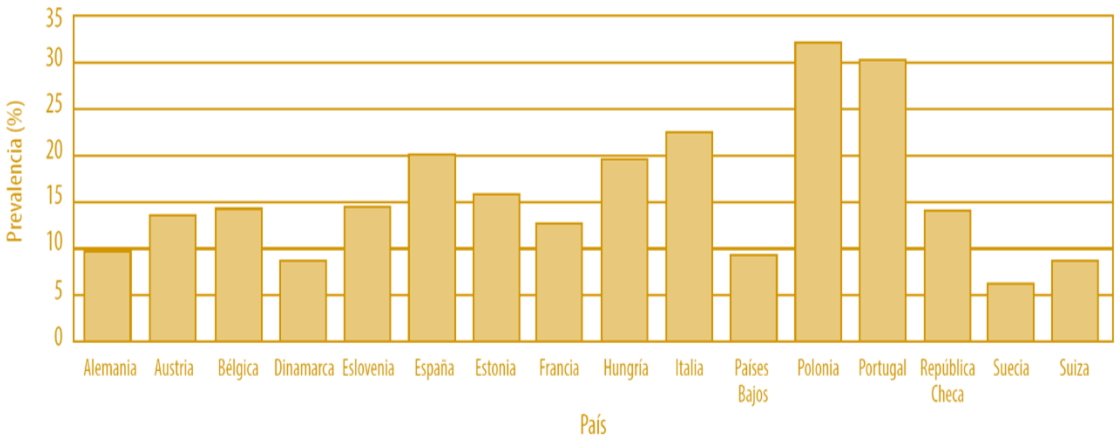


Figura 3. Prevalencia de la inactividad física según la edad en personas de 60 años o más, por país.
FUENTE: (Wave4, 2013)

Las encuestas apuntan a que los hábitos que aumentan el riesgo de enfermedades cardiovasculares se han extendido mucho en los últimos años (Lloyd-Sherlock, 2014). Todos estos malos hábitos varían enormemente de un país a otro, de lo que se deduce que el factor cultural afecta notablemente. Las intervenciones en la población aumentan la cantidad de actividad física realizada y promueven la fuerza muscular y la resistencia (Saelens & Papadopoulos, 2008) y (Liu & Latham, 2009).

2.2.2 Estrategia y plan de acción mundiales

La *Estrategia y plan de acción mundiales sobre el envejecimiento y la salud* es una estrategia acordada por ciento noventa y cuatro países que se reunieron en la *Asamblea Mundial de la Salud* en 2016. La visión de esta estrategia es un mundo en el que todas las personas puedan vivir una vida prolongada y sana, en el que se promueva la capacidad funcional durante todo el ciclo de vida y en el que las personas mayores tengan igualdad de derechos y oportunidades sin que sufran discriminación por motivos de edad.

La estrategia define cinco objetivos para catorce años (de 2016 a 2030) y un plan de acción con medidas a tomar entre 2016 y 2020 para preparar la *Década del Envejecimiento Saludable* (de 2020 a 2030).

2.2.3 Plan España 2050

Basándose en el informe *España 2050* de la Oficina Nacional de Prospectiva y Estrategia (ONPE), recientemente sale el Plan estratégico *España 2050* (en mayo de 2021) y se centra, entre otras cosas, en el *envejecimiento activo* (ONPE, 2021). Incluye cincuenta objetivos para 2050, nueve desafíos y doscientas propuestas agrupadas en doce ejes.

Uno de estos nueve desafíos consiste en *adaptar el estado de bienestar a una sociedad más longeva*, y propone, entre otras cosas, que *España debe apostar por la tecnología y rediseñar parte del Sistema Nacional de Salud, fortaleciendo los servicios públicos de salud y cuidados y cambiando el sistema de pensiones para que apoye a la ciudadanía según sus necesidades y no su historial laboral*.

Los cincuenta objetivos propuestos para 2050 se han diseñado con la palabra cuantificable en mente pues es su función medir cómo de efectivas son las medidas tomadas. Tratan de ser ambiciosos, pero sin dejar de ser realistas, y están preparados para ser modificados conforme

avance el desafío según se descubran distintos o mejores indicadores. Cuatro de los cincuenta objetivos ponen especialmente el foco en esta sociedad más longeva, son los objetivos del 28 al 31, listados a continuación en la **Tabla 1**:

Tabla 1. Indicadores iniciales para evolucionar de un estado de bienestar a una sociedad más longeva. FUENTE: (ONPE, 2021).

Indicadores		Promedio 2014-2019 o último dato disponible*	Objetivos		
			2030	2040	2050
28 Tasa de actividad.	Entre 55 y 64 años (%)	**62%***	63%	64%	67%
	Entre 65 y 74 años (%)	**5%***	7%	9%	11%
29 Gasto público en salud (% PIB) sin incluir el gasto sanitario en cuidados de larga duración.		**5,7%**	7,0%	7,0%	7,0%
30 Gasto público en cuidados de larga duración (% del PIB).		**0,8%**	1,5%	2,0%	2,5%
31 Porcentaje de personas que tienen reconocido el derecho a presentación del SAAD y no la perciben.		**17%**	0%	0%	0%

2.2.4 Intelligent Cities Challenge (ICC)

Intelligent Cities Challenge (ICC) (ICC, 2020) es un proyecto europeo que deriva del anterior *Digital Cities Challenge* (DCC) (DCC, 2018) con el objetivo de mejorar la calidad de vida de ciudadanos y empresas de las ciudades europeas participantes al fomentar el crecimiento sostenible, inteligente y ecológico.

El proyecto pretende crear una plataforma común para datos abiertos, un mercado con soluciones para ciudades inteligentes y apoyo para inversiones conjuntas. En la **Tabla 2** se pueden observar los proyectos financiados por la Unión Europea relacionados con el envejecimiento saludable.

Tabla 2. Social policy & innovation, Past projects.
FUENTE: elaboración propia a partir de: www.age-platform.eu/project-topic/past-projects.

Social policy & innovation, Past projects	
SEED	Silver Economy Awards
CARESS	An integrated framework for home care skills development
SUSTAIN	for sustainable tailored integrated care for older people in Europe
FrailSafe	Frailty prevention
ASCE	Active Senior Citizens for Europe
EMIN	works on adequacy of minimum old age income schemes
Goldenworkers	ICT for promoting active ageing at work (2)
WeDO	Wellbeing and Dignity for Older people
P.E.O.P.L.E.	Exchange and best practice network to support older workers in the EU
HSH	HOME SWEET HOME
Health, Digital technologies, Past projects	
OASIS	Project overview and conclusions AENEAS - Project overview and conclusions
AENEAS	Attaining Energy Efficient Mobility in an Ageing Society
PAMECUS	Positive Action Measures Promoting diversity management in the workplace.
ELDERS	Elders.
CIE	CONSULT-IN-EUROPE
Mediate	Methodology for describing the accessibility of transport in Europe
DAPHNE Eustacea	A European Strategy to Combat Elder Abuse
INCLUSage	Debating older people's needs
DREAMING	elDeRly-friEndly Alarm handling and MonitorING

2.2.5 Ambient Assisted Living (AAL)

En 2008 Europa inició *Ambien Assisted Living* (AAL) (AAL, 2008) dentro del programa marco *Horizonte2020* (Horizonte2020, 2014). Es un programa cuyo objetivo es que estemos sanos y conectados en la vejez al financiar programas europeos relacionados con el tema.

Se presentaron más de 125 proyectos con soluciones basadas en las tecnologías de la información y la comunicación (TIC) para los problemas de las personas mayores, con una gran

participación de PYMES. En la **Tabla 3** se muestra los proyectos que se han considerado más destacables del programa, siempre relacionados con el acompañamiento a mayores.

Tabla 3. Proyectos destacados del Programa Conjunto *Ambient Assisted Living* (AAL) 2008-2013.
FUENTE: elaboración propia a partir de: www.aal-europe.eu/.

Social policy & innovation, Past projects	
3rD-LIFE	Silver Economy Awards
A2E2	An integrated framework for home care skills development
AALUIS	For sustainable tailored integrated care for older people in Europe
ACCESS	Frailty prevention
AGNES	Active Senior Citizens for Europe
AHEAD	Works on adequacy of minimum old age income schemes
AiB	ICT for promoting active ageing at work (2)
ALIAS	Wellbeing and Dignity for Older people
CaMeLi	4Mvideo E-sport cycling at home for rehabilitation and daily exercise

2.2.6 Active and Assisted Living (AAL-2)

Por último, tenemos *Active and Assisted Living* (AAL-2) (AAL-2, 2014) que es una segunda parte del programa *Ambient Assisted Living* (AAL) visto en el apartado anterior. De nuevo parte del programa Horizonte2020, y con un objetivo similar a su predecesor. En la **Tabla 4** se muestra los proyectos más relevantes de esta iniciativa relacionados con el acompañamiento a mayores o con tecnologías que se podrían aplicar en este ámbito.

2.2.7 Uso de tecnología

Cada vez es más viable el desarrollo de una medicina personalizada, predictiva y efectiva gracias a muchos de los avances tecnológicos y en medicina (ONPE, 2021).

La utilidad de la robótica y la inteligencia artificial en este ámbito es reconocida por el Parlamento Europeo (EC, 2017). Por supuesto debemos poner los límites jurídicos y éticos que permitan mantener la dignidad, autonomía y autodeterminación de las personas, pero sin limitar el desarrollo de estas tecnologías (EUR-Lex, 2017).

Tabla 4. Proyectos destacados del Programa Conjunto *Active and Assisted Living* (AAL-2) 2014-2020.
FUENTE: elaboración propia a partir de: www.aal-europe.eu/.

Social policy & innovation, Past projects	
4ME	4Mvideo E-sport cycling at home for rehabilitation and daily exercise
ACCESO	Patient centric solution for smart and sustainable healthcare
ActiveHome	Active@Home Project
AgeWell	Virtual coaching to support a healthy and meaningful life of older adults and employees in their retirement process
APH-ALARM	Comprehensive safety solution for people with Aphasia
CAMI	Artificially intelligent ecosystem for self-management and sustainable quality of life in AAL
CARU	CARU Cares
CleverGuard	ICT solution based on energy data for protecting elderly at home, staying safe and independently
SI4SI	Smart Intervention for Senior Isolation

Ya existen numerosos trabajos que estudian la viabilidad de utilizar las nuevas tecnologías para facilitar la comunicación entre los profesionales de la salud, pacientes y sus familiares (Lindberg et al., 2013), (Wagner y Hunnerup. 2018) y (Demiris G., 2006). Así mismo, se ha estudiado la viabilidad de la teleasistencia como herramienta para disminuir costes en el acompañamiento a mayores (Francesca y Giovanni, 2009), (Martin-Matthews et al., 2013), y ya hay muchas investigaciones que se centran en este tipo de proyectos (Ângelo et al., 2012), (Marschollek M., 2012), (Su y Chiang, 2013) y (Mu-Hsing et al., 2016). Lamentablemente, en general estas propuestas no están enfocadas realmente a la persona mayor y no hacen que se sienta acompañada.

2.3 TECNOLOGÍAS: INTERFACES DE USUARIO

En esta sección buscamos investigar las tecnologías que resulten más agradables e intuitivas para ser utilizadas por las personas mayores. Las tecnologías más relevantes en este ámbito son las interfaces gráficas de usuario (GUI, por sus siglas en inglés: *Graphical User Interface*), las interfaces de usuario basadas en voz (VUI, por sus siglas en inglés: *Voice User Interface*), los asistentes personales, los *chatbots*, los dispositivos inteligentes, los *dispositivos IoT* y los robots.

Aunque cada vez es más normal que las personas mayores entiendan mejor la tecnología, es inevitable que tengan dificultades para usarla por razones muy variadas, desde la experiencia hasta razones físicas o cognitivas (Dodd et al., 2017). Por ello el estudio de estas tecnologías no solo tendrá en cuenta la propia tecnología, además, y seguramente más importante, atenderá a la capacidad de ser entendida y aceptada por las personas mayores.

Al ser el usuario final nuestra prioridad máxima —ya que un error en esta sección llevaría al fracaso completo del proyecto— empezaremos por tratar las interfaces de usuario.

2.3.1 Interfaz gráfica de usuario (GUI)

Las **GUI** (*Graphical User Interface*, o *Interfaz Gráfica de Usuario* en español) son un tipo de interfaz que se despliega en un entorno gráfico que el usuario puede *ver*, y que generalmente se controla con periféricos adicionales como teclado y ratón, mando o pantalla táctil.

Estas interfaces pueden suponer graves dificultades para los mayores por su limitada capacidad para leer y escuchar y por los movimientos poco precisos que impiden o dificultan el uso de los periféricos de entrada (Gregor et al., 2002). También suelen tener problemas por su reducida comprensión de la informática subyacente o controles que son poco intuitivos si falta experiencia (Dodd et al., 2017). Además, pueden tener problemas de atención y reducción de la memoria de trabajo y la memoria de largo plazo, causados por el deterioro de sus capacidades cognitivas (Glisky, 2007).

A las personas mayores les cuesta seguir las acciones necesarias para realizar una tarea, les es fácil perder la pista a mitad o cometer errores (Granata et al., 2013). También les cuesta recordar los pasos que han seguido, los que tendrán que seguir o encadenar más de tres acciones (Galitz, 2002). Los problemas de visión aumentan el tiempo necesario para entender la información que hay en pantalla o para centrarse en lo que realmente quieren dentro de la interfaz (Balakrishnan et al., 2012).

Hay cuatro puntos clave en los que enfocar el diseño a la hora de tratar con personas mayores para sortear los problemas vistos arriba (Dodd et al., 2017): diseño de interfaces y controles; elección del dispositivo adecuado; utilización de lenguaje natural; y realización de evaluaciones cognitivas que midan la usabilidad de la interfaz.

El **diseño de interfaces y controles** está muy estudiado. Hay normativas (ISO, 2017) y (ISO, 2018) y consorcios internacionales (W3C, 2020), y trabajos centrados en el diseño para personas mayores (Holt, 2000), (AgeLight, 2001), (Kurniawan & Zaphiris, 2005) y (Soto, 2013). Las claves son: utilizar textos y objetos normalizados, fácilmente reconocibles e intuitivos, realizar confirmaciones de operaciones, informar de errores y ayuda en contexto.

El **dispositivo elegido** es importante: en general una pantalla táctil es mejor que el ratón y el teclado (Bobeth et al., 2012), los comandos de voz solucionan los problemas ya vistos (Grindrod et al., 2014) y (Wang et al., 2016), e incluso a veces un mando de TV y una pantalla grande son útiles si hay problemas de visión (Picking et al., 2012) y (Bobeth et al., 2012).

El **uso de lenguaje natural** es muy importante ya que el lenguaje técnico complica la interacción, por lo que es mejor cambiarlo por un lenguaje más descriptivo (Grindrod et al., 2014) y (Sutter y Müsseler, 2007). Los mensajes largos confunden más (Zajicek, 2004) y (Balakrishnan et al., 2012), así que es preferible diseñar mensajes cortos y opciones acotadas en lugar de muy abiertas (Zajicek, 2004).

La **evaluación cognitiva** es un método para poner a prueba la usabilidad, prestando especial atención al aprendizaje que se requiere de la aplicación en gente afectada por el deterioro cognitivo. Los usuarios que potencialmente tendrían este deterioro son personas mayores y personas con ciertas discapacidades como en el caso del estudio de (Wilson, 2014). Estas evaluaciones no solo son para probar la aplicación con este tipo de usuarios, sino que además permiten que la aplicación sea entendida mucho más fácilmente por cualquier usuario.

2.3.2 Interfaz de usuario basada en voz

Las *interfaces de usuario basadas en voz* (**VUI**, por sus siglas en inglés *Voice User Interface*) son, como su propio nombre indica, interfaces que utilizan la voz para interactuar con el usuario. Los asistentes que la utilizan se conocen como *asistentes de voz*, y son una evolución de los asistentes inteligentes y *chatbots*.

Como profundizaremos en la **sección 2.4**, estas interfaces están compuestas por dos módulos encargados de convertir voz a texto (ASR, por sus siglas en inglés de *Automatic Speech Recognition*) y texto a voz (TTS, por sus siglas en inglés: *Text-to-Speech*), y por un asistente inteligente entre ambos módulos que, a su vez, se basa ampliamente en la comprensión del

lenguaje natural (NLU, por sus siglas en inglés: *Natural Language Understanding*) y la generación de lenguaje natural (NLG por sus siglas en inglés: *Natural Language Generation*).

No es una tecnología nueva, pero sí ha avanzado mucho en los últimos años, y a todos nos suenan al menos los nombres de los *asistentes de voz* de grandes empresas como *Alexa* de *Amazon*, *Google Assistant* de *Google*, *Siri* de *Apple* y *Cortana* de *Microsoft*.

Aunque las personas mayores suelen rechazar las innovaciones (Murthy & Mani, 2013) y (Gitlin, 1995), en realidad las interfaces de voz tienen muy buena acogida entre este sector de la población (Heerink, 2006) y (Esposito et al., 2019). En general se sienten más acompañados y conectados, y los tratan casi como personas (Basu, 2019).

Sin embargo, el envejecimiento reduce el rango auditivo cómodo con el tiempo de manera lineal (Galitz, 2002) y (Wang et al., 2016), lo que limita la funcionalidad de estos dispositivos.

Las interfaces de voz solucionan algunos problemas que se analizaron en las interfaces gráficas (**sección 2.3.1**). Sin embargo, la mayoría de estos problemas siguen necesitando soluciones especialmente pensadas para los mayores.

2.3.3 Conclusiones

Dada la aceptación de las personas mayores hacia las interfaces de voz con lenguaje natural, por su mayor cercanía al trato con un ser humano, una **VUI** sería la interfaz ideal para nuestro proyecto, dejando una interfaz gráfica como posible futura mejora no abarcable en este trabajo de fin de grado (TFG), pero igualmente importante como interfaz auxiliar y alternativa para la gente que no puede o le cuesta oír. Por otra parte, la voz por sí misma no soluciona todos los problemas asociados con los mayores, por lo que en el diseño de la interfaz se deberán tener en cuenta estos problemas y plantear soluciones concretas para cada uno de ellos.

2.4 TECNOLOGÍAS: ASISTENTES INTELIGENTES

En esta sección se abordan los asistentes inteligentes. En este ámbito escucharemos términos como *asistente virtual*, *asistente de voz* y *chatbot* mezclados a la ligera, pues están bastante relacionados, pero no son lo mismo. Veamos en primer lugar en qué consiste cada uno de ellos.

Los ***chatbots*** están programados para responder a determinados textos de entrada con determinados textos de salida, buscando determinadas palabras clave en las frases del usuario. En la actualidad también hay *chatbots* que usan audio e imágenes además del texto. Esta base no les permite adaptarse a situaciones inesperadas o cambios en el lenguaje, debido a su falta de habilidad en el *procesamiento del lenguaje natural* (*NLP*).

Los ***asistentes virtuales*** suponen un paso más allá de los *chatbots*. Sí poseen capacidad de *procesamiento del lenguaje natural* (*NLP*), lo que les permite extraer *intenciones*, *palabras clave* e incluso *sentimientos* del texto, y por ello pueden adaptarse mucho mejor a los cambios en el lenguaje y a situaciones inesperadas. Se podría decir que, mientras que para los *chatbots* la entrada es una cadena de texto sin sentido, y solo busca el más parecido en su base de datos, los asistentes virtuales sí *comprenden* hasta cierto punto lo que el usuario está diciendo y por ello pueden actuar en consecuencia. Se valen además de técnicas basadas en *Redes Neuronales Artificiales* (ANN, por sus siglas en inglés, *Artificial Neural Network*) para analizar la situación e incluso para aprender de ella de cara a futuras interacciones.

Los ***asistentes de voz*** constituyen, a su vez, un paso más allá de los *asistentes virtuales*. Los asistentes virtuales han logrado realizar un tratamiento adecuado del lenguaje natural a partir de texto. Los asistentes de voz hacen lo mismo a partir del audio y de la voz grabada. Igualmente deben ser capaces de responder mediante voz y lenguaje natural.

En la práctica, y sobre todo para aprovechar todo el conocimiento existente acerca del tratamiento y generación de lenguaje natural, los asistentes de voz se construyen como sistemas que incluyen en su núcleo la tecnología para el reconocimiento y generación de lenguaje natural de los asistentes virtuales encapsuladas entre un módulo de reconocimiento automático del habla (ASR, por sus siglas en inglés: *Automatic Speech Recognition*) que extrae una cadena texto a partir de un audio con voz, y un módulo de conversión de texto a voz (*TTS*, por sus siglas en inglés: *Text-to-Speech*) que, dado un texto, genera un audio con voz.

Con estos dos componentes, podemos utilizar la entrada de audio del usuario, convertirla a texto con ASR, introducirla en un asistente virtual, y la respuesta en forma texto de este convertirla con TTS en audio de nuevo. De esta manera el usuario está hablando con un asistente tan complejo como el virtual, pero con una interfaz de voz (VUI).

2.5 TECNOLOGÍAS: DISPOSITIVOS INTELIGENTES

A continuación, analizaremos el hardware que dará soporte a las interfaces ya vistas y a la lógica de negocio o parte de ella, si procede.

Un dispositivo inteligente es aquel capaz de realizar una o más tareas específicas de la manera más autónoma posible, conectándose para ello, si hace falta, con otros dispositivos, e interactuando con el usuario cuando sea necesario de la manera más natural posible.

Aunque hay cientos de tipos, los más destacados y los que vamos a estudiar por su relación con nuestro trabajo son: altavoces inteligentes, robots de servicio y dispositivos IoT.

2.5.1 Altavoces inteligentes

Consisten en un altavoz, un asistente virtual y una interfaz de voz. Por ejemplo, Echo de Amazon, Google Home o Apple HomePod. El asistente virtual tiene la lógica necesaria para manipular el altavoz e interactúa con el usuario a través de la interfaz de voz.

La idea principal es proporcionar un asistente de voz al usuario que le permita interactuar con contenidos y servicios: conocer el tiempo, las últimas noticias, realizar una compra online, acceder a películas (si tiene pantalla) y a música. Generalmente estos contenidos y servicios estarán *en línea*.

Un aspecto importante que se debe tener en cuenta en este tipo de dispositivos es que la mayor parte de la lógica de la aplicación reside fuera de la misma, generalmente en una plataforma de servicios TI. Uno de estos servicios característicos son los de conversión de voz y los de reconocimiento del lenguaje natural. Se trata de servicios muy complejos que requieren por el momento grandes infraestructuras y recursos que no se pueden proporcionar desde un hardware tan limitado como suele ser el de los altavoces inteligentes.

2.5.2 Dispositivos IoT

Los **dispositivos IoT** (del inglés, *Internet of Things* o *Internet de las Cosas* en español) son la explotación máxima del concepto de dispositivo inteligente. La interconexión entre distintas máquinas se lleva un paso más allá al hacer muchos dispositivos con potencia limitada (por tener microprocesadores muy básicos y muy poca memoria y capacidad de almacenamiento) que son capaces de interconectarse entre ellos para realizar cada uno su respectiva tarea.

Por ejemplo, podemos tener un dispositivo solo centrado en medir la temperatura y la humedad del aire y transmitirla a quien le pregunte, o a un dispositivo central que solo recopila datos de la casa. Luego podemos tener otro dispositivo cuya función sea, bajo demanda, cerrar o abrir las persianas automatizadas de la casa. De esta manera, el dispositivo que mide la temperatura podría avisar al dispositivo que controla las persianas de que las cierre cuando la temperatura en el exterior supere la del interior, si es que lo que se desea es mantener la vivienda lo más fresca posible.

Así, miles de dispositivos podrían interconectarse para realizar complejas tareas, con complejos análisis y con sencillas interfaces para el usuario. Cada uno de esos dispositivos es una "cosa" que está conectada generalmente a Internet (aunque podrían comunicarse de muchas maneras, como Bluetooth o una red local si están cerca), de ahí el nombre.

Los proyectos AAL y AAL-2 vistos en el **apartado 2.2** contienen muchas iniciativas para la asistencia a mayores basadas en dispositivos IoT.

El problema de esta tecnología es que no hay un estándar definido, por lo que a veces es complicado o incluso imposible conectar dispositivos IoT de distintos fabricantes, resulta muy engorroso gestionar tantas tecnologías diferentes y pueden suponer un importante problema de seguridad. Una solución actual basada en dispositivos IoT requiere una cuidadosa selección de dispositivos o que todos estén fabricados por el mismo proveedor o que sean proyectos muy personalizados.

2.5.3 Robots de servicio

Un robot de servicio es un robot que se encarga de servir, en general realizando tareas en el hogar como limpiar o barrer (Roomba de iRobot) o acompañar a niños o personas mayores. A diferencia de los *altavoces inteligentes*, estos robots consumen muchos más recursos hardware debido a que, en general, tienen que moverse, a veces desplazarse por entornos muy inciertos o complicados, o realizar tareas que requieren precisión, fuerza o ambas.

Es este componente hardware el que tradicionalmente ha complicado la comercialización de los robots, pues eran demasiado caros de producir (y por lo tanto de comprar) y difíciles de mantener. Los incesantes avances en robótica han permitido, con el paso del tiempo, abaratar costes de producción y hacer a los robots más resistentes, duraderos y precisos, lo que ayuda a que los humanos confiemos más en ellos.

Como hemos dicho, se clasifican en dispositivos inteligentes, por lo que pueden conectarse a otras máquinas para ampliar su funcionalidad o interactuar con el usuario cuando lo estimen necesario.

Ya hay proyectos que han mostrado robots capaces de vigilar y asistir a personas mayores como (Hendrich et al., 2015) en el proyecto *Ambient Assisted Living* (AAL) ya visto.

2.5.4 Conclusiones

Nuestra propuesta se acerca más a los dispositivos IoT y a los altavoces inteligentes debido a que buscamos soluciones económicas, al alcance de cualquier presupuesto, y a que no se requiere en principio características como la movilidad y la capacidad física de un robot. Más concretamente, nuestro enfoque es el diseño de un **dispositivo IoT** que se comporte como un asistente de voz.

2.6 TECNOLOGÍAS: REDES DE INTERCONEXIÓN

El tipo de dispositivo que queremos construir precisará de algún tipo de conexión a la red ya que nuestro asistente de voz tendrá que desplegarse en la nube debido a que los dispositivos inteligentes, basados en un hardware muy ajustado, carecerían de potencia suficiente para ejecutarlo. Analizaremos las principales propuestas que se puede encontrar en el entorno del hogar y que pueden ser válidas para nuestros propósitos: *Red Cableada*, *Red Wi-Fi*, *Red PLC*, *Red Mesh*, *Telefonía móvil* y *Red LoRa*. Una vez el dispositivo esté conectado a cualquiera de las redes analizadas, no entraremos en cómo será el mecanismo y las operadoras que proporcionen conexión a Internet. Cualquiera de las propuestas y operadores actuales es válido en principio.

La **red cableada** comunica los dispositivos de dicha red a través de cables Ethernet. Ofrecen una pérdida de calidad de red mínima, con un rendimiento estable y mucha seguridad, pues los datos se envían por el cable y no pueden ser interceptados sin conectarse físicamente a la red. Sin embargo, colocar todos los cables a través de una casa puede ser muy complicado, y por supuesto imposibilita o dificulta mucho el que un dispositivo se vaya moviendo por la casa.

La **red Wi-Fi** es inalámbrica, por lo que no requiere el pesado trabajo de los cables Ethernet. Sin embargo, no es tan estable y segura como la red cableada. Además, si la casa tiene una estructura muy intrincada habrá que poner repetidores, y un dispositivo que se mueva tendrá

que ir cambiando su punto de acceso de un repetidor a otro manualmente (por parte del usuario).

La **red PLC** (*Power-Line Communication*) es aquella en la que la información se codifica y envía a través de la red eléctrica de la casa, de manera que cualquier enchufe eléctrico convencional puede ser utilizado como salida para cable de Ethernet. En general se basan en el protocolo X10, cuya patente solo permite el uso no comercial del mismo, para consumo privado. Comparte prácticamente todas las ventajas y desventajas de la *red cableada*, excepto porque: tiene la ventaja de ser fácil de instalar y configurar a diferencia de la cableada que depende de la estructura física de la casa, pero tiene la desventaja de que el ancho de banda está más limitado y la conexión no tiene tanta calidad.

La **red Mesh** consiste en una malla de dispositivos que actúan parecido a repetidores *Wi-Fi,* pero cada repetidor es un *nodo* que cuenta como la misma red, con el mismo identificador y contraseña. Estos *nodos* (o *satélites*) gestionan a cuál de ellos se conectan los dispositivos que acceden a la red, haciendo transparente al usuario que haya varios puntos de acceso, dando la ilusión de un mismo punto de acceso en toda la casa. Por lo tanto, es como la red de repetidores *Wi-Fi*, pero transparente al usuario como un solo punto de acceso, con lo cual es mejor. En la práctica, sin embargo, una *red Mesh* suele ser más cara que una red de repetidores *Wi-Fi*.

La **red de telefonía móvil** de 5G tiene una gran capacidad de conexión, un gran ancho de banda y muy poca latencia: es perfecta para los dispositivos IoT. Además, con esta red cada dispositivo podría tener conexión directa a Internet, lo cual es genial, pero es ahí donde radica su problema. En el resto de las redes vistas partimos de que tenemos acceso a Internet. Este acceso se contrata con una operadora telefónica y tiene un coste mensual considerable, en función de la calidad de la red. Si tenemos decenas o cientos de dispositivos diminutos conectados directamente a Internet en lugar de a un punto de acceso común, entonces tenemos decenas o cientos de contratos con las operadoras telefónicas, uno por conexión. Esto es inviable económica y administrativamente, y de momento es insorteable. Por ello, una red IoT con una cantidad decente de dispositivos no puede usar este tipo de red todavía, más allá de tener un único contrato 5G y conectar el resto de los dispositivos con otro tipo red, y su uso en este ámbito dependerá de cómo cambie el modelo comercial de la telefonía móvil.

La **red LoRa** es muy robusta y es ideal para largas distancias, por ello se usa en el ámbito militar y espacial desde hace ya décadas. Sería ideal para una red en ciudades inteligentes (*Smart*

Cities). Además, esta tecnología requiere baja potencia de emisión y, por lo tanto, los dispositivos que la emplean consumen poco, lo que la hace ideal para *dispositivos IoT*. Las redes LoRa, aunque llevan usándose mucho tiempo en otros ámbitos, siguen en desarrollo y no poseen una tecnología tan madura como otras opciones.

En 2018 pusieron en Alcoy, de parte del ayuntamiento, una red LoRa que abarcaba toda la calle Entença, como parte de un proyecto *Street Lab* (Alcoy Street Lab, 2018), en el que los fabricantes y las organizaciones pueden probar y presentar sus proyectos para ciudades inteligentes, como por ejemplo un software que controlara las farolas a las cuales se le había asignado un dispositivo inteligente conectado a la red LoRa.

2.6.1 Conclusiones

Cualquiera de las propuestas analizadas es viable para el funcionamiento de nuestro proyecto. En cualquier caso, la decisión final tendrá mucho más que ver con los costes que tenga y la facilidad de instalación y mantenimiento, que con la capacidad de conexión que tenga cada una.

Nuestro dispositivo debería aceptar red cableada (o PLC, nos es transparente) pero también sería aconsejable algún tipo de red inalámbrica (red Wi-Fi, Mesh o incluso LoRa si hubiera infraestructura disponible).

Una red telefónica de 5G para conectar cada SAM Device sería una solución ideal si solo hay un dispositivo por casa, y permitiría que SAM estuviera disponible para las personas mayores, aunque su red personal de Internet caiga por cualquier razón o se sature por tener muchos dispositivos conectados. Sin embargo, ofrecer con dinero público una conexión 5G a cada persona mayor que se asista puede salirse del presupuesto fácilmente.

Una red LoRa sería ideal para el proyecto, si, por ejemplo, el ayuntamiento de cada localidad implantase una red LoRa local que abarcase la ciudad (una sola antena LoRa puede dar cobertura a un radio de 40Km y a miles de dispositivos), permitiendo tener de manera estable y más privada los servicios de SAM y evitando depender de que las personas mayores tengan una conexión a Internet en su casa o de estar ofreciéndosela. Sin embargo, eso es algo que no ocurrirá pronto, aunque será importante seguir de cerca el progreso de este tipo de red.

2.7 TECNOLOGÍAS: CONCLUSIONES

De la sección de **envejecimiento saludable** concluimos que el uso de la tecnología para asistir a las personas mayores es viable hoy en día, y que incluso ya hay muchos proyectos que han apostado por este ámbito.

De la sección de **interfaces de usuario** concluimos que lo que queremos es una interfaz de usuario basada en voz (VUI) que permita comunicarnos con las personas mayores (y cualquier usuario) mediante lenguaje natural, ya que se ha demostrado que las personas mayores las reciben adecuadamente. Por lo tanto, de los tipos de asistentes vistos en la sección de **asistentes inteligentes**, claramente queremos un asistente de voz.

De la sección de **dispositivos inteligentes** concluimos que queremos un dispositivo IoT, barato de producir y llevar las casas de las personas mayores, que se apoyen en un *servicio de voz* para el tratamiento y generación del lenguaje natural.

De la sección de **redes de interconexión** concluimos que nuestro dispositivo debe aceptar red tanto cableada como inalámbrica, ya que son las redes más comunes, y que necesitaremos dicha conexión pues el dispositivo IoT tendrá que conectarse a nuestro servicio de voz.

Así pues, vista la tecnología que se utiliza hoy en día, la conclusión es que reforzamos la idea del proyecto, y que el dispositivo IoT con asistente de voz más un servicio de voz propio constituye un proyecto basado en tecnología madura y apta para ser usada por las personas mayores, a quienes va dirigido este trabajo.

2.8 INTELIGENCIA ARTIFICIAL: CAMPOS Y RAMAS

La Inteligencia Artificial es un campo muy amplio, con multitud de usos en multitud de ámbitos, y en la mayoría de los niveles de cada ámbito: desde analizar una situación hasta controlar maquinaria pesada, pasando por conversar, aconsejar acciones, recomendar compras y mucho más. Además, se encuentra en constante investigación, desarrollo y progreso. Es por todo esto que es muy difícil proporcionar una clasificación de sus diferentes ramas y tecnologías, desde un solo punto de vista o resumir en qué se usa cada tipo. Por ello, la bibliografía actual presenta multitud de clasificaciones y taxonomías que varían entre sí y que dependen de la finalidad para la que se realizaron (Ertel, 2018) y (Jackson, 2019).

En nuestro caso, vamos a presentar de manera resumida las técnicas y algoritmos de IA más relacionados con los asistentes de voz, y por lo tanto con el procesamiento del lenguaje natural y de tratamiento de la voz. La **Figura 4** muestra los campos de la Inteligencia Artificial más importantes para este propósito, así como las relaciones entre ellos y con otras ramas principales.

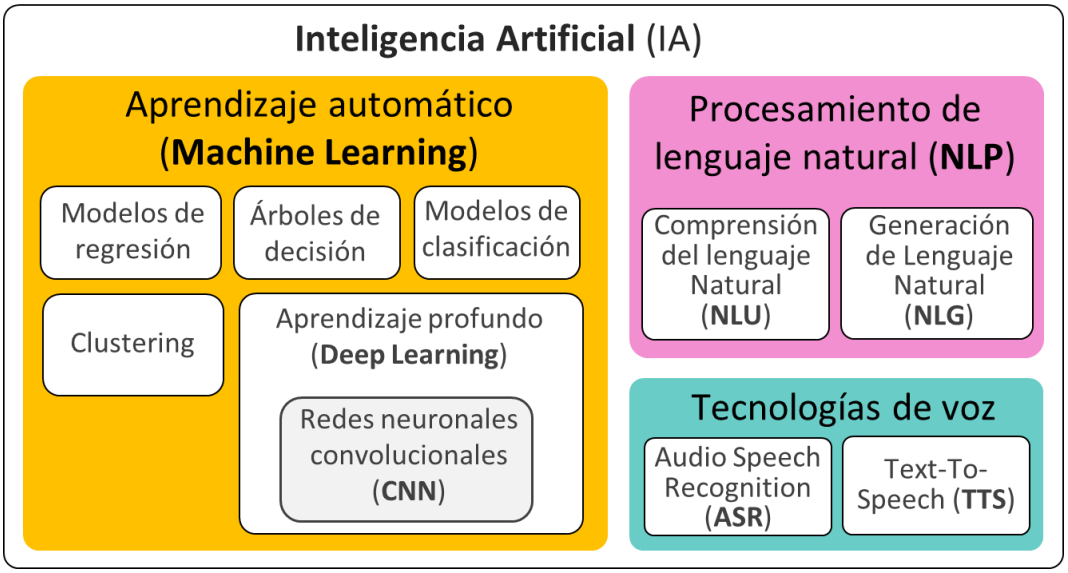


Figura 4. Campos y ramas de la IA.
FUENTE: elaboración propia.

Para cubrir el objetivo principal de nuestro trabajo, las ramas que más nos interesan son: **procesamiento del lenguaje natural** (NLP, siglas en inglés de *Natural Language Processing*) y las **tecnologías de voz**, dividida en dos grandes grupos: *ASR* (*Automatic Speech Recognition* o Reconocimiento Automático del Habla) y *TTS* (*Text-to-Speech*, o Texto a Voz). Adicionalmente, estas ramas se apoyan en técnicas de **Machine Learning** (*ML*) para su funcionamiento, y por ello también quedan recogidas y las estudiaremos a continuación.

2.8.1 Machine Learning

Machine Learning (ML, o en español *Aprendizaje Automático*) es la rama de la inteligencia artificial que investiga maneras de lograr que una máquina pueda resolver problemas muy variados de manera similar a como lo hacemos los humanos: identificando distintos patrones, clasificándolos o prediciéndolos, todo en base a patrones que la máquina ha *aprendido*, ya sea de manera asistida por humanos o aprendidos en aplicaciones reales. Así, las máquinas

aprenden a realizar estas tareas y a optimizarlas, a partir de los datos obtenidos con la experiencia o durante la fase de entrenamiento.

En la **Figura 4**, hemos incluido los algoritmos, modelos y técnicas de *Machine Learning* que más nos interesan para nuestro trabajo: modelos de regresión, árboles de decisión, árboles de clasificación, *clustering* (o agrupamiento), deep learning (o aprendizaje profundo) y, dentro de este último, redes neuronales convolucionales (CNN por sus siglas en inglés: *Convolutional Neural Network*).

2.8.2 Modelos de regresión

Los modelos de regresión utilizan técnicas estadísticas para descubrir la relación entre distintos factores y entender cómo se afectan entre ellos. Están especialmente pensados para casos complejos en los que la variable dependiente a estudiar no depende directamente de otra (independiente), si no que pueden influir varias a la vez y de distintas maneras (Peláez, 2016).

Se puede aplicar para crear modelos con fines explicativos, cuando se desea saber exactamente cómo las variables independientes afectan a la variable a estudiar (independiente) o, si no es posible tener esa precisión, crear modelos con fines predictivos que estiman o aproximan cómo influyen las variables independientes en la variable a estudiar.

Esta técnica resulta muy útil en el ámbito de la investigación, y permiten a las máquinas (y a los humanos) predecir cómo influirá en determinados resultados la actuación sobre ciertos factores.

2.8.3 Árboles de decisión

Los árboles de decisión son un método supervisado (es decir, que requiere intervención humana) que sirven como alternativa a los modelos de regresión, cuya ventaja es que el modelo resultante es sencillo de leer e interpretar (Roche, 2009). Además, el modelo resultante es muy resistente frente a valores atípicos.

2.8.4 Árboles de clasificación

Los árboles de clasificación son un tipo de árbol de decisión en los que la variable dependiente a estudiar es categórica, o sea que tiene un conjunto cerrado posible de valores (Roche, 2009).

2.8.5 Clustering

El *clustering* (o agrupación) consiste en agrupar instancias similares con el objetivo de buscar relaciones entre variables de manera descriptiva. Por lo tanto, sirve para encontrar similitudes entre datos y agruparlos por las características que comparten (Rokach y Maimon, 2005).

Las técnicas de *clustering* se emplean en los algoritmos de recomendación de Amazon o Netflix para encontrar qué tienen en común las series más vistas o los artículos más comprados y ofrecer nuevas propuestas personalizadas de contenidos a los usuarios.

2.8.6 Aprendizaje profundo

El *Deep Learning* (DL, o Aprendizaje Profundo) es una rama de la inteligencia artificial, más concretamente dentro de *Machine Learning*, que busca entender conceptos complejos en función de una jerarquía anidada de otros conceptos más sencillos que, a su vez, se basan en otros conceptos más sencillos (Bengio et al., 2017). De esta manera logra representar y comprender conceptos abstractos y complejos desmenuzándolos en conceptos más pequeños sucesivamente hasta que llega a un nivel de complejidad que la máquina es capaz de comprender de manera directa (operaciones predefinidas, o básicas, como sumar y restar) y realizar rápidamente.

Estos sistemas aprenden de la experiencia para montar sus propios modelos, de manera similar a como, conceptualmente, los humanos aprendemos a nivel biológico, creando conexiones entre neuronas capaces de reconocer patrones sencillos que llevan a conexiones entre neuronas capaces de reconocer patrones más complejos.

De manera resumida, las máquinas con un sistema de *Deep Learning* utilizan los datos iniciales para extraer cuáles son las características importantes y entrenarse a sí mismas, para luego seguir usando la experiencia para mejorar su propio modelo.

Estas técnicas, en lugar de modelos de regresión y clasificación como en el *Machine Learning* original, se basan en redes neuronales, y por ello *Deep Learning* también se conoce como DNN, de *Deep Neuronal Networks* (o *Redes Neuronales Profundas*).

2.8.7 Redes neuronales convolucionales

Las *Redes Neuronales Convolucionales* (o CNN, por sus siglas en inglés de *Convolutional Neural Network*) son un apartado muy relevante en *Deep Learning* en el que se utilizan convoluciones en lugar de multiplicaciones de matrices (Bengio et al., 2017) y son especialmente efectivas en tareas que operan con matrices como la visión por ordenador. Por ello, en lugar de aplicar operaciones de matrices directamente sobre, digamos, una imagen convencional de 1280 por 720 píxeles, se aplica la operación repetidas veces usando una matriz de convolución mucho más pequeña, de apenas decenas o centenas de píxeles en total.

Aunque esta diferencia puede parecer pequeña a simple vista, estas operaciones permiten reducir drásticamente el coste computacional de las redes neuronales. Estas mejoras ayudan, además, a reducir las dependencias entre nodos de la red.

Debido a su eficacia, las *Redes Neuronales Convolucionales* facilitaron la evolución del *Deep Learning* en los años noventa cuando el hardware no era tan potente como ahora y el desarrollo y pruebas de estos sistemas se alargaban más.

Este tipo de redes neuronales son una variación del *perceptrón multicapa* y en ellas se emula el funcionamiento de la corteza visual primaria de un cerebro humano.

2.8.8 Natural Language Processing

El Procesamiento de Lenguaje Natural (NLP por sus siglas en inglés de *Natural Language Processing*) se encarga de permitir que las máquinas entiendan y usen el lenguaje natural mediante texto para comunicarse con seres humanos. Debido a esto, esta rama de la IA resulta crucial para este trabajo y se verá en más en detalle en la **sección 2.9**.

2.8.9 Tecnologías de voz

Las tecnologías de voz se encargan de convertir voz a texto (ASR, *Automatic Speech Recognition,* o *reconocimiento automático de voz*) y de convertir texto a voz (TTS, *Texto-to-Speech*). Estas técnicas y algoritmos también son claves para este trabajo debido a que se encargarán de comunicar a nuestros usuarios con nuestro asistente inteligente (que en principio trabaja con textos) mediante una interfaz de usuario basada en voz (VUI). De nuevo, se tratará más adelante en la **sección 2.10**.

2.9 INTELIGENCIA ARTIFICIAL: PROCESAMIENTO DEL LENGUAJE NATURAL

El procesamiento del lenguaje natural (NLP, por sus siglas *Natural Language Processing*) es una rama de la inteligencia artificial encargada de lograr que las máquinas puedan trabajar con lenguaje natural, esto es, lenguajes como español, inglés, o cualquier idioma usado por los seres humanos. Para ello deben saber extraer las ideas más importantes de un texto, clasificarlas y entenderlas, y crear respuestas o frases también en lenguaje natural. El NLP se basa principalmente en algoritmos de *Machine Learning*.

Los dos bloques que componen a grandes rasgos el **NLP** son (véase **Figura 5**): **NLU** (*Natural Language Understanding*, o comprensión del lenguaje natural) y **NLG** (*Natural Language Generation* o generación de lenguaje natural).

El uso más directo del procesamiento del lenguaje natural es el de conseguir que las máquinas puedan conversar con seres humanos (o entre ellas, pero eso tendría un propósito más experimental que útil), consiguiendo que las máquinas entiendan lo que dice el humano y sean capaces de generar respuestas con sentido.

Cabe destacar que el procesamiento del lenguaje natural no incluye el proceso cognitivo que realizamos los humanos al pensar. Es decir, solo se encarga de extraer intenciones y datos del lenguaje, y luego de generar una respuesta dados otros datos, generados por otras técnicas de IA o algoritmos que han recibido como entrada la información entendida.

En la actualidad el procesamiento del lenguaje natural se utiliza para realizar traducciones automáticas, resumir textos o extraer ideas clave, e incluso extraer sentimientos del texto para saber cómo se siente la persona que escribió dicho texto. Esto último se aplica, por ejemplo, en márquetin para analizar mensajes en redes sociales respecto a un producto y saber qué opina la gente.

2.9.1 Técnicas para NLP y algoritmos de voz

La **Figura 5** muestra esquemáticamente los componentes de estas ramas de IA. Además, se ha añadido la rama de **Tecnologías de voz** para clarificar que van separadas del **Procesamiento del Lenguaje Natural** ya que la función de las tecnologías de voz es convertir del dominio del audio (el usado en interfaces de usuario basadas en voz) al dominio del texto, que es el que realmente

maneja la rama de NLP, y luego devolverlas al dominio del audio para continuar con la interfaz de voz.

En dicha figura vemos como parte de las técnicas del NLP están dirigidas a la comprensión del lenguaje natural (NLU), otras a la generación del lenguaje natural (NLG) y otras se solapan y se emplean en ambas ramas. Además, hay técnicas y algoritmos dentro de NLP que, aunque no forman parte estrictamente ni de NLU y de NLG, proporcionan funcionalidades muy interesantes para aplicaciones concretas.

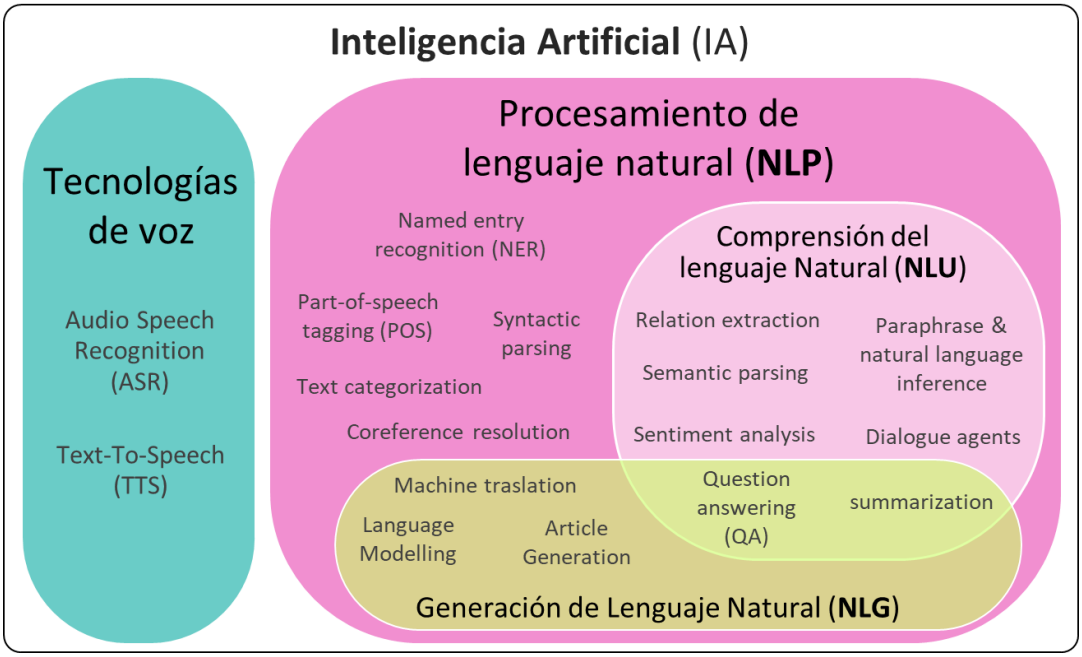


Figura 5. Relación entre las tecnologías de voz, el NLP, y sus diferentes ramas y tecnologías.
FUENTE: elaboración propia.

2.9.2 Comprensión del Lenguaje Natural (NLU)

La comprensión del lenguaje natural (NLU, *Natural Language Understanding*) se basa en identificar y extraer los elementos clave de un texto que nuestro lenguaje humano ordena de manera más desestructurada, con ambigüedades o que requieren contexto para ser entendidas, distintas maneras de expresar lo mismo y variadas reglas gramaticales que en un mismo idioma dependen incluso de la región en la que se hable. Este procesamiento se realiza sobre *cadenas de texto*.

Al final consiguen llevar esos elementos clave al ámbito ordenado y estructurado que las máquinas pueden entender y por lo tanto utilizar y procesar con otras técnicas de IA. Estos

elementos clave suelen incluir *intenciones* y parámetros (*entidades*) de la intención, pero también pueden incluir datos adicionales como el estado de ánimo del usuario o el contexto de la situación para identificar mejor la orden recibida.

Para este paso un sistema NLU requiere del *entrenamiento* de un modelo para entender un determinado idioma, ya que este modelo recoge esas normas gramaticales, sintaxis y manera de hablar que necesita conocer para separar los elementos clave.

2.9.3 Generación de Lenguaje Natural (NLG)

La generación del lenguaje natural (NLG) consiste en construir frases en lenguaje natural, entendibles para los seres humanos. Básicamente es el proceso inverso al anterior (NLU), ya que ahora la máquina obtiene datos estructurados para ella y los reorganiza en palabras de un idioma humano, con la sintaxis, normas gramaticales y conectores necesarios para que un ser humano que hable ese idioma pueda entender el mensaje como entendería a otro ser humano. Todo en formato de *cadenas de texto*.

Utiliza modelos que comprenden todas estas normas del lenguaje, con lo cual un sistema de NLG necesita también un modelo entrenado, distinto para cada idioma en el que quiera escribir.

2.10 Inteligencia artificial: tecnologías de Voz

Las tecnologías de voz tienen como objetivo en este proyecto servir de puente entre el usuario final que interactúa mediante su voz, hablando, y un asistente inteligente que, como ya se ha visto, se basa en texto. Para este fin se apoya en dos grandes bloques: *Automatic Speech Recognition* (ASR) y *Text-to-Speech* (TTS).

2.10.1 Reconocimiento automático de voz (ASR)

El reconocimiento automático de voz (ASR, por sus siglas en inglés de *Automatic Speech Recognition*) se utiliza para que la máquina (texto) entienda al usuario (voz) al extraer del audio grabado del usuario el texto que ha dicho. Esto es, de la cadena de audio “SAM, llama a mi hijo” obtenida grabando la voz del usuario, el sistema obtiene la cadena de texto correspondiente, de manera que pueda ser enviada al asistente inteligente.

El proceso de ASR también es comúnmente conocido como Speech-to-Text (STT, en español: conversión de voz a texto), como contraparte a la siguiente sección (Text-to-Speech).

El ASR no solo sirve para interfaces de voz con asistentes inteligentes, también se utiliza como sistema de dictado para sustituir al teclado y escribir más fácilmente, ya sea una instrucción para el equipo, un mensaje o email, redactar una nota, o escribir una cadena de búsqueda en un buscador web.

2.10.2 Generación automática de voz (TTS)

El proceso de conversión de texto a voz (TTS, por sus siglas en inglés *Text-to-Speech*) es el inverso del anterior, pues este se encarga de, a partir de un texto (generalmente escrito por nuestro asistente inteligente, generar un audio de voz con la frase en el idioma elegido (para el cual necesita un modelo entrenado de dicho idioma, y del cual puede depender enormemente la calidad y credibilidad del audio).

Este sería, en la interacción usuario-asistente, el siguiente paso, una vez que el asistente haya procesado la información obtenida de la última frase del usuario, haya tomado una decisión y haya devuelto la frase escrita que quería comunicar al usuario. Tras el TTS, la frase está en formato audio y puede ser reproducida al usuario para que este no tenga que leerla y mantenga la interacción por voz.

De momento es difícil que una frase leída por TTS suene totalmente humana, pero su finalidad principal es que sea inteligible, no engañar a los humanos, aunque es un objetivo para el futuro en el que ya se trabaja para lograr que estas interfaces nos resulten cada vez más cercanas y naturales.

El TTS tampoco limita su utilidad al ámbito de los asistentes inteligentes, teniendo capacidad para sustituir a narradores y actores o actrices de voz en proyectos cuyo presupuesto no puede permitirse a estos profesionales (especialmente si los personajes son robots, ya que la credibilidad de un sistema TTS de bajo presupuesto todavía puede mejorar mucho), o para leer el contenido de páginas web o aplicaciones informáticas en voz alta, lo cual es muy útil para gente que no puede o le cuesta ver.

2.11 ASISTENTES DE VOZ

En la **sección 2.3.3** concluimos que la solución que mejor se podía adaptar al problema planteado —diseñar un dispositivo capaz de acompañar a los mayores en su vida diaria, complementando el trabajo que hacen los cuidadores, ya sean familiares, servicios sociales, asociaciones, etc.— consistía en basar el diseño en un asistente de voz. Es por esto por lo que vamos a investigar qué asistentes de voz hay ya implementados para determinar si alguno de ellos se adapta al proyecto o si conviene crear nuestro propio asistente.

Aunque la prioridad es realizar el proyecto con código y dispositivos no comerciales de libre desarrollo, vamos a incluir también productos comerciales para calibrar qué funcionalidades se puede esperar en la actualidad de este tipo de dispositivos y qué ventajas e inconvenientes puede tener cada propuesta.

2.11.1 Asistentes de voz comerciales

Muchas empresas, incluidas las más grandes, han tratado de crear sus propios asistentes de voz para ayudar a sus usuarios a acceder con mayor facilidad a sus servicios y productos o para asistirles en general en multitud de tareas diarias. Sin embargo, solo algunos de estos asistentes han persistido al final, pues la mayoría eran poco utilizados y sus empresas optaron por descontinuarlos.

Las propuestas más maduras y que han demostrado ser utilizadas con el tiempo son *Siri* de *Apple*, *Cortana* de *Microsoft*, *Google Assistant* de *Google* y *Alexa* de *Amazon*.

2.11.1.1 Siri (Apple)

Siri es la asistenta virtual de Apple, disponible para todos sus dispositivos. Trabaja solo para dispositivos Apple en el contexto para el que fue creada, por lo tanto, no podemos personalizarla para que nos ayude en este trabajo.

2.11.1.2 Cortana (Microsoft)

Cortana es la asistenta virtual de Microsoft, cuyo nombre viene de un personaje ficticio que es una inteligencia artificial en la saga Halo, videojuego que también es de Microsoft. Aunque no funciona mal, se le da poco uso en general y Microsoft parece querer dejar de darle soporte,

hasta el punto de que las aplicaciones externas de Cortana para iOS y Android fueron eliminadas. De nuevo, no se puede personalizar lo suficiente para que nos sirva en este trabajo.

2.11.1.3 Google Assistant (Google)

Este es el asistente virtual de Google. Está disponible en una amplia gama de dispositivos de todos los fabricantes, además de disponer de un kit de desarrollo de software que permite extender sus funcionalidades a desarrolladores ajenos a Google.

2.11.1.4 Alexa (Amazon)

Alexa es la asistenta virtual de Amazon. En lugar de estar disponible como ayuda auxiliar para dispositivos que tienen otras funcionalidades, Alexa trabaja en dispositivos creados para ella, altavoces inteligentes, que a veces incluyen pantalla, cuya finalidad es permitirte usar los servicios de Alexa (aunque también está disponible la aplicación para dispositivos inteligentes, necesaria para configurar tu dispositivo Alexa).

Amazon permite el desarrollo externo de *skills* (habilidades) para Alexa, de manera que se puede personalizar considerablemente la funcionalidad de Alexa. En estas *skills* se puede definir un modelo de conversación propio, acceder a los servicios de Alexa y a servidores ajenos para disfrutar de otros servicios personalizados o propios. Todo esto es desarrollable desde una interfaz web que además permite probar el modelo.

Aunque es muy extensible y personalizable, una *skill* no deja de ser algo secundario para Alexa, y su uso, cada vez que se desee interactuar con una *skill* personalizada, pasa por activar a Alexa con la palabra de activación (que en la actualidad solo puede ser *Alexa*, *Echo* o *Amazon*) y luego pidiéndole que active nuestra *skill* para, entonces, poder interactuar con ella. Por ejemplo, en nuestro caso la persona mayor tendría que decir “Alexa, abre SAM” o “Alexa, activa a SAM” cada vez que quisiera interactuar con nuestro asistente.

2.11.1.5 Conclusión

De los productos comerciales, Alexa es la más personalizable y por lo tanto la que permitiría el desarrollo del trabajo de fin de grado. Sin embargo, las limitaciones de palabra de activación y de activar la *skill* harían el proceso de uso engorroso y romperían la interacción natural que buscamos. Además, recordemos que preferimos soluciones de código abierto para evitar que los datos de los usuarios pasen por servidores externos.

Las opciones comerciales tienen la ventaja de ser maduras y contar con entornos de desarrollo que permiten un prototipado y un despliegue muy rápido. También cuentan con infraestructuras TI para el despliegue que aseguran la escalabilidad, adaptabilidad y fácil mantenimiento de los servicios de voz. Sin embargo, también tienen importantes carencias: solo están disponibles las funcionalidades que a ellos les interesan atendiendo a sus políticas de desarrollo, comerciales o de seguridad; estas políticas son cambiantes y rara vez tienen en cuenta los intereses de los desarrolladores; la información privada de los mayores debe viajar a servidores externos, y lo que es peor, donde se dejará almacenada por tiempo indefinido para tareas como el entrenamiento de los modelos de voz y de diálogo; los modelos de diálogo y las funcionalidades que ofrecen no están específicamente pensadas para los mayores, por lo que muchos de los problemas que se analizaron con respecto al diseño de interfaces especializadas para ellos no se podrá cumplir; finalmente, aunque los productos son atractivos y robustos, no siempre están al alcance de todos los bolsillos.

Por lo tanto, aunque el desarrollo de una buena *skill* de *Alexa*, o una app de *Google assistant* podría bien valer para un trabajo de fin de grado, en general no se podrían cumplir los requisitos fijados para el proyecto y por ello investigamos las opciones de código abierto que hay disponibles para identificar con qué otras opciones podemos contar.

2.11.2 Asistentes de voz no comerciales

Los asistentes de voz comerciales, aunque suelen estar más pulidos y ser más robustos que otros asistentes, tienen la pega de que son un producto cerrado, que tienen que vender, y en cuya implementación hay un valor que no quieren que otros aprovechen.

Esta necesidad de ocultar cómo funciona tu asistente para evitar que te copien el trabajo habitualmente conlleva restricciones en la personalización al evitar que accedas a otros servicios o realices ciertas acciones, aunque otras veces se debe a temas comerciales, incompatibilidades impuestas adrede para que tengas que usar solo los productos con los que el asistente elegido funciona.

Además, no pueden permitir que ejecutes su software de procesamiento del lenguaje o de reconocimiento de voz en una máquina tuya propia, debes pasar por sus servicios en la nube que recopilarán la información que les envíes (en este caso, la de los usuarios de tu dispositivo, nuestros mayores en este caso) y la almacenarán. Es cierto que la almacenan con el objetivo de

mejorar y entrenar a sus máquinas, pero no quita que la privacidad de nuestros usuarios quede comprometida, especialmente porque estos datos permanecen en los servidores de las grandes empresas durante un tiempo indefinido, y porque la empresa no niega que pueda darles otro uso a los datos.

Por supuesto se añade que, si los dispositivos son propietarios, parte del presupuesto del proyecto se tiene que destinar a la empresa responsable, creando una dependencia insalvable hacia su hardware, sus decisiones de mercado, sus licencias y su disponibilidad, además de no poder modificar dicho hardware si la especificación de nuestro proyecto cambia o queremos añadir más funcionalidades.

Por todas estas razones, y especialmente para proyectos de investigación, nos interesa basar nuestro dispositivo en Open Hardware y Open Software, de manera que podamos, a partir de estos diseños, cambiar todo lo que necesitemos para que nuestro proyecto sea como queramos que sea, y sin depender de futuras decisiones de empresas. La desventaja de las tecnologías de código abierto es que no te dan tanto trabajo hecho, no has pagado por ellas y dependes de los avances que ha realizado la comunidad.

A continuación, veremos algunos de los asistentes de voz de código abierto disponibles.

2.11.2.1 Mycroft

Mycroft es un asistente inteligente de código abierto que puede instalarse en una máquina *Linux* o en *Raspberry P*i (Mycroft, 2021). Adicionalmente disponen de un dispositivo de hardware abierto llamado *Mark 1* con el asistente instalado listo para su uso. Este dispositivo Mark 1 es básicamente una *Raspberry Pi* con hardware adicional y una funda *mona*.

También tienen anunciado el dispositivo *Mark 2*, previsto para producción en este mismo año (2021), con muchas mejoras como arreglos de micrófonos, pantalla y altavoces de alta calidad, aunque este dispositivo no será open hardware (a pesar de que se anuncia como tal).

2.11.2.2 Stephanie

Stephanie es una plataforma de código abierto para la automatización de tareas mediante órdenes de voz, que funciona en *Linux*, *Raspberry Pi* y *Android* (Stephanie, 2021). Tiene un API muy sencillo que permite extender la funcionalidad de manera personalizada fácilmente, o bien

se puede reprogramar entera pues todo el código es libre. Sin embargo, no hay disponible un dispositivo físico concreto que actúe de asistente o altavoz inteligente.

2.11.2.3 Open Assistant

Open Assistant es también un asistente de voz para nuestro ordenador que puede realizar tareas varias mediante órdenes de voz (Open Assistant, 2021). Está escrito completamente en *Python*, y el código es totalmente abierto, por lo que se puede modificar a placer.

Nuevamente, no cuenta con un dispositivo concreto que actúe como asistente y su enfoque está más centrado en controlar la interfaz de nuestro PC que para construir un asistente inteligente.

2.11.2.4 Dragonfire

Dragonfire es un asistente de voz de código abierto, esta vez para Ubuntu y Android. También está escrito en Python (Dragonfire, 2021). Se basa en la biblioteca *Mozilla DeepSpeech* para el ASR. *Dragonfire* tiene una cuenta en *Twitter* a través de la cual se le puede consultar sobre diferentes temas.

2.11.2.5 LinTO

LinTO es un asistente de voz de código abierto enfocado en el uso profesional en el ámbito de los negocios, centrado en las tareas de una reunión de negocios (LinTO, 2021). El dispositivo está basado en código abierto y está pensado para que desarrolladores de asistentes de voz puedan usarlo y adaptarlo. Está disponible para Android, Raspberry Pi y páginas y aplicaciones web.

2.11.3 Conclusión

Aunque hay muchas ofertas en cuanto a asistentes de voz, en todos ellos la personalización conlleva un esfuerzo que no compensa el trabajo que viene ya hecho. Por ello, y por aumentar así el aprendizaje obtenido en este trabajo de fin de grado, optamos por realizar nuestro propio asistente de voz a partir de componentes individuales, que son los que ya hemos visto a lo largo del estado de la técnica:

- Un componente de ASR (Voz a texto) que traducirá a texto lo que diga el usuario
- Un componente de TTS (Texto a voz) que traducirá a voz lo que diga el asistente
- Un asistente inteligente que supone la sección de NLP, que trabaja con texto

- Un dispositivo hardware que actúa de cuerpo físico para nuestro asistente y accederá al servicio de voz que agrupa los tres primeros componentes para hablar con el usuario
- Adicionalmente, estos dispositivos suelen incluir una palabra de activación. El objetivo es que, en lugar de estar constantemente escuchando y transmitiendo a la web el audio del usuario, esperan a escuchar esta palabra o frase clave y entonces empiezan a grabar y a transmitir, como por ejemplo para hablar con Alexa se tiene que decir “Alexa”. Este reconocimiento debe hacerse en local en el dispositivo porque si no igualmente hay que estar transmitiendo la voz todo el rato.

A continuación, centraremos el estudio en las plataformas TI que hay disponibles para cada una de estas funcionalidades.

2.12 DISPOSITIVOS HARDWARE DISPONIBLES

El primer paso que debemos dar para montar nuestro propio asistente de voz es elegir la base física, el hardware más adecuado para nuestros propósitos. No se descarta la posibilidad de realizar varios prototipos si hay más de un candidato prometedor y contamos con el tiempo necesario.

Vamos a estudiar los dispositivos basados en una filosofía open hardware más conocidos: *Arduino*, *Rasberry Pi* y *ReSpeaker*.

2.12.1 Arduino

Arduino es una plataforma para la creación de prototipos electrónicos con microcontroladores. Incluye un entorno de desarrollo muy sencillo e intuitivo que permite a cualquiera iniciarse en la electrónica así que no es extraño ver *Arduinos* en institutos y centros formativos. Posee una comunidad muy amplia de desarrolladores software y hardware y una cantidad de componentes electrónicos y de bibliotecas software muy extendida que permite cubrir gran variedad de funciones. Tanto el software como el hardware son totalmente libres.

2.12.2 Raspberry Pi

Raspberry Pi se concibió como un pequeño ordenador para realizar proyectos de electrónica varios. Aun siendo de un tamaño muy reducido, posee gran cantidad de entradas y salidas

analógicas y digitales, y periféricos, e incluso puertos USB, HDMI o Ethernet. En la actualidad la *Raspberry Pi* se utiliza como una versión más potente de un *Arduino*. De nuevo está completamente basada en una filosofía de open hardware.

Las placas basadas en Raspberry Pi pueden mover sistemas operativos como Linux, lo que las convierte en una plataforma muy flexible. Sus usos abarcan desde la creación de dispositivo IoT hasta, como se quería inicialmente, pequeños ordenadores personales, pasando por emuladores de consolas o cualquier otro tipo de periférico o controlador que se precise en proyectos de electrónica.

2.12.3 ReSpeaker

Los dispositivos *ReSpeaker* son placas hardware de desarrollo creadas por el fabricante *Seeed*. Estas placas cumplen un propósito más específico que *Arduino* o *Raspberry Pi*; aunque sirven para muchos propósitos, están especialmente pensadas para actuar como asistentes de voz como el que queremos construir. Para ello, la placa incluye *arreglos de micrófonos*, arreglos de leds, y el propio fabricante proporciona bibliotecas software con la implementación de diferentes algoritmos para el reconocimiento y procesamiento de la voz, como la detección de la dirección de la que proviene el sonido (DOA —*Direction of Arrival*), la detección de actividad de voz (VAD —*Voice Activity Detection*), la posibilidad de discriminar si hay alguien hablando o es solo sonido; y la posibilidad de supresión de ruido (NS —*Noise Suppressor*).

Algunas de las placas solo incluyen las funcionalidades descritas en el párrafo anterior (arreglos de micrófonos y leds y las bibliotecas de código con los algoritmos descritos) diseñadas para trabajar con una placa de desarrollo *Raspberry Pi*.

Otra de las ventajas de utilizar estas placas (aunque el servicio está totalmente abierto para cualquier diseño hardware) es que Seeed proporciona servicio de fabricación de nuestras PCB (*Printed Circuit Board*) en caso de que queramos dar el paso para comercializar nuestro propio dispositivo.

2.12.4 Conclusión

Todos los dispositivos analizados son prometedores. Los principales candidatos son las placas de desarrollo *ReSpeaker* puesto que proporcionan muchas de las funcionalidades que se necesitarán para la construcción de un dispositivo asistente de voz y, lo más importante, son

diseño open hardware que permitirán que los podamos modificar según nuestras necesidades e integrar con relativa facilidad en nuestro propio hardware.

Las placas *Raspberry Pi* también son un candidato excelente ya que son de propósito más general, especialmente si la unimos con el pack de micrófonos de Seeed.

Las propuestas de *Arduino* no tienen tanta potencia como los otros dos, lo cual podría afectar al rendimiento global. Se debe tener en cuenta que, aunque la mayor parte del software para el reconocimiento de voz y del lenguaje natural se ejecutará en servidores externos de forma que nuestro dispositivo accederá a él en forma de servicios, debe haber un núcleo mínimo de funcionalidades que deben residir en el dispositivo y deben poder trabajar incluso aunque no exista una conexión a Internet.

2.13 SISTEMAS DE NLP DISPONIBLES

Para crear nuestro propio *asistente de voz*, como se vio en la **sección 2.4**, es crucial disponer de un buen *asistente inteligente*, pues será la lógica detrás del sistema. Esto implica encontrar (o implementar, pero eso se sale del alcance de este trabajo de fin de grado) una buena plataforma para el procesamiento del lenguaje natural (NLP) que nos proporcione, bien el servicio completo, bien un conjunto de librerías de IA que nos permitan montar nuestra propia plataforma a partir de ellas.

Aunque hemos analizado decenas de plataformas existentes, para simplificar las conclusiones solo incluiremos aquéllas que pueden adecuarse mejor a nuestros intereses; además de que la mayor parte de las plataformas existentes, tan solo proporcionan una interfaz más o menos cómoda y con más o menos funcionalidades más pensadas para ser utilizadas por usuarios finales y, en la práctica, trabajan sobre otras plataformas de NLP que son las que realizan el verdadero procesamiento. Son precisamente estas últimas plataformas las que nos han resultado más interesantes. Y de todas ellas, vamos a analizar las plataformas de procesamiento del lenguaje natural que son de código abierto, en concreto: *Wit.ai*, *Rasa* y *DeepPavlov.ai*. Como también se verá, estas tres plataformas, a su vez, se basan en librerías de IA de nivel más bajo como *Tensorflow*, *Keras*, *PyTorch* o *spaCy*.

2.13.1 Wit.ai

Wit.ai es una plataforma de código abierto para el procesamiento del lenguaje natural (Wit.ai, 2021). *Wit.ai* extrae la intención del usuario a partir del texto de entrada, así como las entidades correspondientes. Además, trabaja con contexto y es capaz de extraer sentimientos de la frase.

Funciona sobre *Javascript*, *Python*, *Go* y *Ruby*, y está pensada para dispositivos IoT, *wearables*, domótica e incluso aplicaciones móviles. Posee una amplia comunidad y permite iniciar el desarrollo de aplicaciones a partir de datos predefinidos estándar o de otras aplicaciones ya hechas por usuarios.

Una aplicación de *Wit.ai* puede aprender de la experiencia a partir de los usuarios que interactúan con ella.

2.13.2 Rasa

Rasa es una plataforma de código abierto para conversaciones mediante texto, y puede conectarse a aplicaciones de mensajería para automatizar respuestas (Rasa, 2021). Proporciona las herramientas para crear interacciones totalmente automatizadas a partir de la extracción de *intenciones* y *entidades* del texto del usuario, además de ser capaz de trabajar con contexto. En la **Figura 6** se puede observar la arquitectura de Rasa.

Rasa permite empezar proyectos de manera sencilla con el uso de datos predeterminados que más tarde pueden personalizarse. Está basada en las bibliotecas de algoritmos de IA *TensorFlow* de *Google* y *spaCy*.

Rasa se divide en tres componentes: *Rasa Open Source* que es la base de todo, y es código abierto; *Rasa X* que añade desarrollo guiado por la conversación (CDD —*Conversation Driven Development*) que permite que el asistente aprenda de las conversaciones reales que tiene con los usuarios, esta parte es totalmente gratis, pero de código propietario; por último, *Rasa Enterprise*, que añade funcionalidades para grandes empresas y es de pago. La visión de Rasa es que su tecnología debe ser abierta, y por ello solo y exclusivamente las funcionalidades para empresas que solo necesitan las empresas son de pago.

Según (Greyling, 2020), Rasa es excepcional en comparación con otras plataformas por varias razones, de entre las cuales destacan su buena documentación, una comunidad muy activa, el hecho de que aprenda de la conversación gracias a CDD, que permita definir conceptos en datos

gracias a las entidades y los roles, que acepte cualquier idioma y que ofrezca control total sobre los datos del usuario.

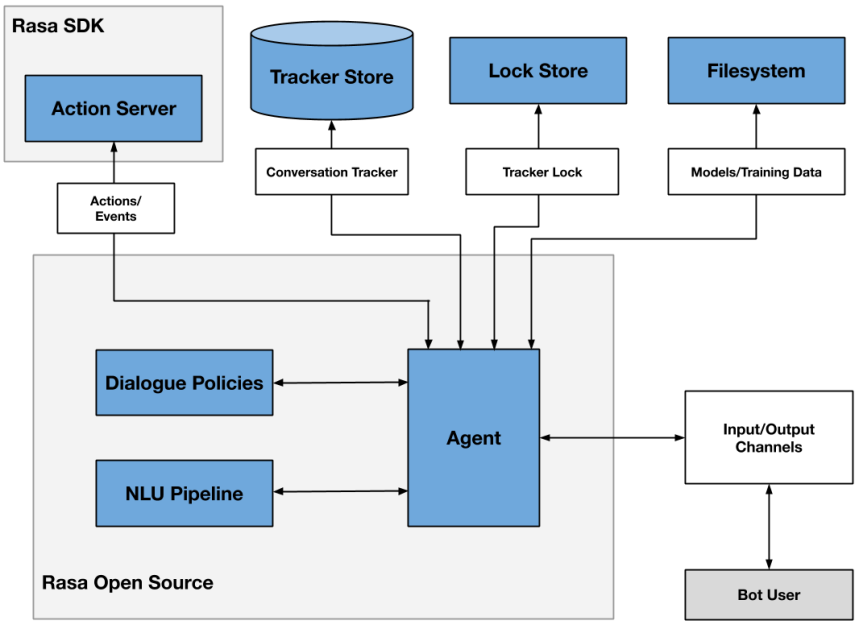


Figura 6. Arquitectura de *Rasa Open Source*.
FUENTE: (Rasa, 2021).

2.13.3 DeepPavlov.ai

DeepPavlov.ai es una plataforma de procesamiento del lenguaje natural pensada para desarrolladores e investigadores (DeepPavlov.ai, 2021).

La plataforma ofrece gran variedad de modelos NLP ya entrenados y componentes de diálogos predefinidos. Sean estos predefinidos o propios, pueden probarse desde Python, línea de comandos, su API o desde un servidor propio a partir de un Docker. Además, incluye herramientas para integrar sus aplicaciones en la infraestructura existente y un entorno para comparar modelos.

Al igual que el resto de las plataformas, *DeepPavlov* se apoya ampliamente en bibliotecas de IA existentes, concretamente en *TensorFlow*, *Keras* y *PyTorch*.

2.13.4 Comparativa

Desde el punto de vista únicamente del desarrollo o de los intereses de un desarrollador, la propuesta de **Wit.ai** es la más completa y atractiva y posee una comunidad muy potente que proporciona un amplio respaldo.

Sin embargo, desde un punto de vista más general y operativo (despliegue, mantenimiento, actualización, productividad, estrategia, etc.), la conclusión es que la propuesta de **RASA** es la más completa de las tres:

- La versión gratuita proporciona suficientes herramientas para emprender con garantía cualquier proyecto que precise de NLP.
- Posee una amplia comunidad que la respalda.
- Además de una potente plataforma de NLP de código abierto, también proporciona servicios de productividad más profesionales y específicos para aquellos que los necesiten. Nos referimos a servicios como alojamiento en la nube, empaquetamiento, seguridad, escalabilidad que, en muchas ocasiones, suponen unos costes inasumibles para muchos proyectos debido a que requieren altas inversiones en hardware y contar con personal altamente especializado.
- Permite comenzar con una versión propia, en servidores locales o gestionados por nosotros y, en caso de ser necesario, pasar a una estrategia de externalización y servicios gestionados en la nube; y viceversa.
- Proporciona un módulo de analíticas, cuadros de mando y evaluación de los modelos que son fundamentales a medida que estos van ganando en complejidad.

2.14 SISTEMAS DE ASR DISPONIBLES

El dispositivo que queremos crear (***SAM Device***) debe comunicarse mediante voz. Hasta el momento, hemos solucionado el problema de entender el lenguaje mediante NLP, pero esta tarea se realiza desde texto. Es por ello por lo que necesitaremos un sistema de reconocimiento automático de voz (ASR) para poder *hablarle* al asistente inteligente. En los siguientes apartados analizaremos algunas de las propuestas más interesantes de código abierto disponibles.

2.14.1 Mozilla DeepSpeech

Mozilla DeepSpeech es un proyecto de *Mozilla* para el reconocimiento automático de voz (DeepSpeech, 2021). Está basado en *TensorFlow* de *Google* y en la investigación sobre reconocimiento de voz de *Baidu* (Baidu, 2014).

Dispone de modelo de voz en español. Los modelos son entrenados por la comunidad utilizando *Mozilla Common Voice* (Mozilla Voice, 2021), que proporciona múltiples conjuntos de datos con muestras de voz donadas por la gente. En la página oficial de *Common Voice* se puede ver el progreso de los modelos en múltiples idiomas, o donar muestras de voz y validar muestras donadas para apoyar al proyecto.

En abril de 2021 *Mozilla* anunció que el desarrollo de *DeepSpeech* iba a relajarse y a centrarse en financiar la creación de aplicaciones que usen *DeepSpeech*.

2.14.2 CMU Sphinx

CMU Sphinx es una biblioteca para ASR que además incluye reconocimiento de palabra de activación (*hotword*) personalizable (CMU Sphinx, 2020). Dispone de modelos en español y tiene ejemplos oficiales en *Python* y *Java*.

2.14.3 Julius

Julius es una biblioteca para ASR (Julius, 2020), utilizada por la aplicación de escritorio *Simon* (Simon, 2020) que sirve para darle instrucciones a un ordenador personal y sustituir así al teclado y al ratón. No tiene modelos en español ya hechos, por lo que necesitaríamos entrenarlos.

2.14.4 Kaldi

Kaldi es una biblioteca para ASR que funciona sobre *Linux* (recomiendan *Debian* o *Red Hat*) (Kaldi, 2020). Idealmente, según los desarrolladores, el servidor que lo ejecute irá sobre un clúster de máquinas *Linux* con *SunGridEngine* y con alguna *GPU* de *NVidia*. No dispone de modelos en español.

2.14.5 Comparativa

En la **Tabla 5** se muestra un pequeño cuadro comparativo con las propuestas estudiadas atendiendo a las características que más nos pueden afectar.

Tabla 5. Comparación de bibliotecas de ASR.
FUENTE: elaboración propia.

Nombre	Hotword	Modelo en español	Sistema sobre el que trabaja
Mozilla DeepSpeech	No	Sí	Linux y Windows
CMU Sphinx	Personalizable	Sí	Linux y Windows
Julius	No	No	Linux y Windows
Kaldi	No	No	Linux

Por la simplicidad que supone el no tener que entrenar modelos en español, los principales candidatos son *Mozilla DeepSpeech* y *CMU Sphinx*. Se debe tener presente que para que estos modelos sean válidos, se requiere muchas muestras de voz, de mucha gente, con un enorme consumo de recursos TI)

Aunque *CMU Sphinx* incluye detección de palabra de activación, realmente nos interesa mantener separada la palabra de activación (que seguramente se detecte en local en nuestro dispositivo *SAM Device*) y el servicio de ASR (que se desplegará en los servidores), con lo cual esta funcionalidad no es tan relevante.

También, que la biblioteca esté disponible para *Linux* y para *Windows* es preferible por la flexibilidad, pero principalmente utilizaremos *Linux* por nuestra norma de código abierto.

Por tener una amplia comunidad detrás y un equipo de desarrollo conocido, nuestra elección para construir nuestro módulo ASR es *Mozilla DeepSpeech*.

2.15 SISTEMAS DE TTS DISPONIBLES

Ya disponemos de voz que se convierte a texto (ASR) y de texto que es interpretado por un asistente (NLP). El siguiente paso para la realización del asistente de voz será el componente que permitirá a dicho asistente responder al usuario de viva voz: se trata del componente de TTS

(*Text to Speech*). En los siguientes apartados analizaremos las propuestas más interesantes para nuestros intereses.

2.15.1 Mozilla TTS

Mozilla TTS es una biblioteca para la generación automática de voz hecha en *Python* con amplia documentación que dispone de un modelo en español ya entrenado y permite la creación de modelos nuevos, si se necesita (Mozilla TTS, 2021).

2.15.2 MARY

MARY es una biblioteca de TTS hecha en *Java* (por lo tanto, requiere la *Java Virtual Machine* para funcionar) que posee una interfaz sencilla de utilizar. Lamentablemente no dispone de modelos en español por defecto (Mary, 2020).

2.15.3 eSpeak

eSpeak es una biblioteca de TTS hecha en C (eSpeak, 2020). No dispone de modelo en español y los usuarios indican que la voz no es extremadamente realista.

2.15.4 Mimic

Mimic es la biblioteca de TTS utilizada por el asistente virtual *Mycroft* (ver **sección 2.11.2.1**). La biblioteca es muy ligera y los resultados son realistas, pero solo está disponible en inglés (Mimic, 2020).

2.15.5 CMU Flite

Es la biblioteca de TTS hecha por el mismo equipo de la biblioteca de ASR llamada *CMU Sphinx* (ver **sección 2.14.2**) (CMUFlite, 2020). Está hecha en C, pero no posee modelo en español.

2.15.6 Comparativa

Por razones similares a las de ASR, y porque es la única que tiene modelo en español ya creado, además de tener una amplia comunidad detrás, elegimos *Mozilla TTS* para nuestro componente TTS.

2.16 MOTORES DE *HOTWORD* DISPONIBLES

Por último, vamos a estudiar motores de reconocimiento de palabra de activación (*hotword*). Recordemos que, como principal requisito, estos componentes deberán funcionar en local en el propio dispositivo IoT (en nuestro caso, en *SAM Device*). Además, queremos que la palabra sea personalizable para poder cambiarla a "Sam", "Oye Sam", "Hola Sam", etc.

2.16.1 Porcupine

Porcupine es un motor de detección de palabra de activación que utiliza *Deep Learning* y con palabra de activación personalizable disponible para *Linux*, *Mac OS X* y *Raspberry Pi* (Porcupine, 2021). Parte de su código es abierto (disponible en su github), pero otra parte no.

Este motor es particularmente eficiente y preciso según una comparación con otros motores realizada por uno de los desarrolladores de *Porcupine* (Porcupine Benchmark, 2021). Además, es escalable, al no requerir más CPU para detectar más palabras de activación al mismo tiempo.

La creación de modelos propios para palabras de activación es sencilla, solo tienes que escribir la frase que quieres en su *PicoVoice Console* y podrás descargar el modelo, sin muestras de voz, y muy preciso, según sus desarrolladores.

Sin embargo, la creación de la palabra de activación tiene dos problemas. El primero es que para utilizar estas palabras de activación en ámbito no personal (como nuestro proyecto) hay que tener una cuenta de empresa, que cuesta 400 dólares al mes. Dejando de lado si es mucho dinero o poco, ya estamos hablando de depender de empresas externas, especialmente por el segundo problema: para evitar que crees un modelo y canceles la cuenta, los modelos "caducan" (el programa los rechaza) a los treinta días de generarlos, y es obligatorio que los usuarios vuelvan a descargar el modelo. Esto significa que, además, tendríamos que estar redistribuyendo el modelo cada mes.

2.16.2 Snowboy

Snowboy es un motor de detección de palabra de activación de código abierto que utiliza *Deep Learning*, funciona en local, consume apenas un 10% de CPU en la *Raspberry Pi* más débil, y la palabra de activación es personalizable (Snowboy, 2021).

Funciona en todas las *Raspberry Pi*, *Ubuntu*, *Mac OS X*, *iOS*, *Android* y *ARM64*. Está escrita en C++ y dispone de interfaz para ser utilizado en otros lenguajes populares como C, *Java*, *NodeJS*, *Perl* y *Python*.

La creación de modelos para una palabra de activación específica requiere muestras de voz, aunque con unas pocas se pueden crear modelos *personales* para hacer pruebas y más tarde pasar a modelos *universales* que funcionen con todo el mundo.

2.16.3 Comparativa

Porcupine es muy prometedora y permite crear los modelos de palabra de activación muy fácilmente (que es nuestro principal problema), lo cual es importante para que los usuarios puedan iniciar la interacción cómodamente, pero su modelo de negocio nos trae problemas mucho más grandes de los que nos soluciona. Por ello *Snowboy* es la mejor opción: tendremos que trabajar con un modelo personal por el momento, que solo funcionará bien conmigo misma, para luego hacer un modelo universal con más muestras, pero permitirá que seamos independientes de licencias comerciales y estrategias de empresa en esta pequeña (aunque importante) parte del proyecto.

2.17 CONCLUSIONES

Una vez finalizado todo el estudio de la técnica, y dada su amplitud, es conveniente resumir las principales conclusiones que se desprenden del trabajo.

La población de Europa ha experimentado un cambio fundamental en su estructura de edad, con personas que viven más tiempo que nunca. La mayor esperanza de vida y las tasas de natalidad constantemente más bajas han significado que nuestra población envejezca continuamente, y se prevé que más de la mitad de la población de la UE tendrá más de 65 años en 2070.

El envejecimiento de la población presenta muchos desafíos en torno a la calidad de vida de las personas mayores y de sus cuidadores, así como impactos en el mercado laboral. Estos desafíos deben abordarse ahora si queremos asegurarnos de que podemos continuar viviendo vidas saludables, activas e independientes hasta nuestra vejez.

El envejecimiento de la población también representa una gran oportunidad. Con la comunidad de expertos y con el apoyo adecuado, se puede aprovechar las grandes ventajas de las tecnologías conectadas para crear soluciones y productos que pueden cambiar la vida de las personas ahora y en el futuro.

Del estudio realizado, dentro del ámbito del envejecimiento saludable y de la atención a los mayores, se puede extraer el siguiente conjunto de conclusiones:

- El envejecimiento de la población es una realidad.
- Se están produciendo cambios en los modelos familiares y en los hábitos de los mayores hacia una mayor independencia.
- Los cambios son previsibles y se deben comenzar a abordar desde ahora para convertir los retos en oportunidades.
- España está en una situación idónea para obtener beneficio directo, convertirse en un foco de atracción y crear nuevas oportunidades de negocio.
- La salud a lo largo de la vida es fundamental, pero más si cabe en la etapa de madurez.
- El ejercicio y la alimentación deben mejorarse y debe crearse una cultura que facilite la adquisición de unos hábitos saludables.
- Las tecnologías pueden ayudar a proporcionar soluciones efectivas, eficaces y eficientes.
- Ya hay muchas soluciones basadas en la tecnología.
- Hay muchas tecnologías maduras.
- Es importante que el enfoque de las soluciones sea general y que contemple tanto a los mayores, como a sus familiares y a los sistemas de salud y al personal sociosanitario.
- El producto debe ser el servicio de acompañamiento, no la tecnología, pero también es importante entender que, si la tecnología no existe, no es económica, fiable y sostenible, difícilmente se podrá proporcionar un servicio asequible y sostenible a la creciente población…

De estas conclusiones, y desde una perspectiva abierta, se identifican los siguientes grandes retos:

- Se deben diseñar servicios de acompañamiento a los mayores que sean válidos frente al cambio demográfico que se está produciendo.
- Que dichos servicios sean universales y sostenibles.

- Que se cree un clima de confianza y colaboración entre todos los actores y sectores implicados.
- Que las tecnologías se pongan al servicio de los ciudadanos y de los profesionales.
- Que las tecnologías no definan el servicio, sino que el servicio determine qué tecnologías ayudarán a que sea factible
- Que las tecnologías sean fiables, económicas, etc.
- Que se alcance una normalización de los servicios de salud que posibilite una independencia de los fabricantes o de tecnologías concretas.
- Que los servicios se desarrollen bajo estándares de calidad.
- Que los servicios puedan ser fácilmente evaluados y que sus resultados se puedan medir.

También desde un punto de vista global, podemos decir que se parte de una serie de soluciones ya existentes que pueden facilitar mucho que se alcancen los restos identificados. Estas soluciones se pueden resumir en dos grandes aspectos:

- Se reconoce el problema del cambio demográfico, de la oportunidad que supone y de la necesidad de actuar cuanto antes.
- Existe multitud de tecnología e iniciativas, que muchas de ellas son maduras o muy maduras tanto en ámbitos como la robótica, los sensores inteligentes, los dispositivos IoT, la inteligencia artificial, como en los ámbitos de las aplicaciones médicas y los servicios de apoyo a los mayores y al fomento de un envejecimiento saludable.

3 Antecedentes y resultados previos

Tal y como se ha ido explicando en los apartados previos, este trabajo no parte de cero. Se trata de una propuesta para construir un dispositivo inteligente de acompañamiento a mayores (***SAM Device***) que, a pesar de que es autocontenido y puede operar de forma totalmente independiente, alcanza todo su valor cuando lo hace de forma integrada con el *Servicio de Acompañamiento a Mayores* (***SAM Project***) creado por el grupo de investigación *grupoM* de la Universidad de Alicante en el que tuve la oportunidad de participar como becaria durante el curso 2018-2019.

Este servicio se desarrolló en colaboración con la Universidad Politécnica de Valencia para el Ayuntamiento de Alcoy (Alicante), y en la actualidad ya se ha incorporado también el Ayuntamiento de Torrent (Valencia).

El servicio de acompañamiento a mayores se basa en una plataforma tecnológica que permite mejorar la experiencia de los mayores, proporcionar un espacio de encuentro e interacción con los familiares que los cuidan y con sus sanitarios, abaratando los costes del servicio de acompañamiento hasta hacerlo asequible para las instituciones y, sobre todo, para todas las familias.

3.1 SAM PROJECT

El concepto de *Servicio* es muy ambicioso, y abarca mucho más que la concepción y el desarrollo de una determinada plataforma o tecnología, vertebrando todos los elementos que se requieren para lograrlo: organismos, instituciones, normativas, recursos materiales y humanos e infraestructuras TI.

SAM Project se concreta en la creación de una plataforma tecnológica de servicios especializados en la nube, junto con la utilización de *dispositivos IoT* para proporcionar un servicio de seguimiento y acompañamiento a mayores y a personas en riesgo que resulte lo menos invasivo y lo más natural posible para ellas (véase **Figura 7**).

La mera utilización de dispositivos, aunque es cierto que puede proporcionar soporte a un sinfín de problemáticas relacionadas, no aporta en sí misma un valor añadido a las propuestas existentes, tanto en el ámbito de la innovación, como en el de los mercados de servicios y aplicaciones informáticas. El verdadero potencial de **SAM Project** radica en el novedoso concepto en el que se basa: más allá de estar pensado para proporcionar un servicio únicamente a personas mayores, está concebido para hacerlo extensivo a su entorno, concretamente a sus familiares y a todo el personal socio sanitario que los atiende. De esta forma se logra crear toda una red social muy especializada que asegura la sostenibilidad del servicio frente a innumerables imponderables de índole muy diversa: limitaciones económicas, temporales, sociales o falta de recursos materiales o humanos.

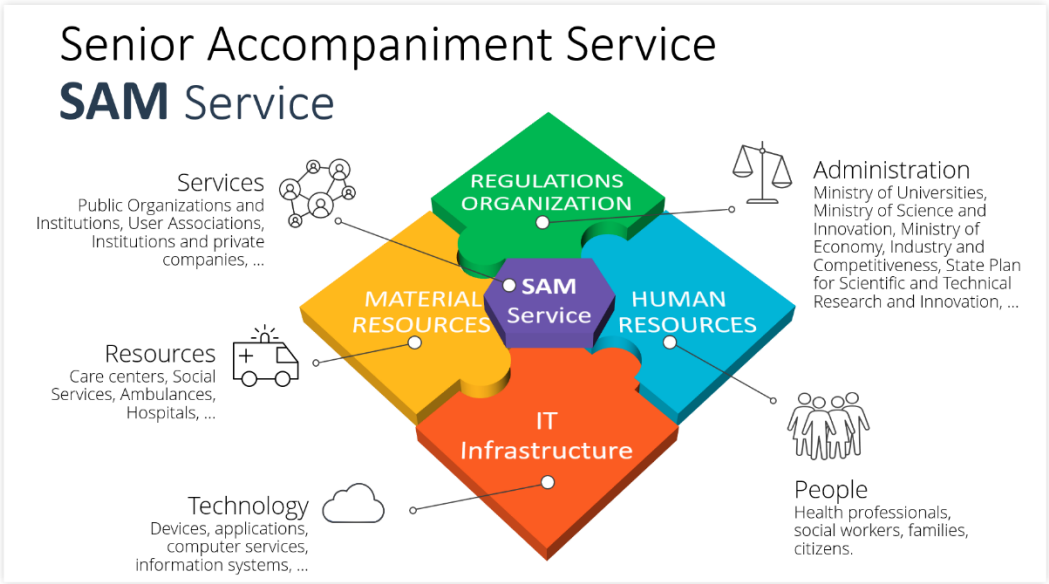


Figura 7. Elementos que intervienen en el Servicio de Acompañamiento a Mayores.
FUENTE: equipo de SAM.

Esta red la hemos denominado *SAM Smart Social Network* o, sencillamente, **SAM Network** (véase **Figura 8**), y aglutina tanto a los propios usuarios, es decir, las personas pertenecientes a los colectivos vulnerables, como a sus familiares y a todos los profesionales sociosanitarios que intervienen de alguna forma en el proceso de acompañamiento, proporcionando un espacio

común que facilita establecer y fortalecer relaciones entre colectivos con las mismos intereses: usuarios-usuarios, familiares-familiares, facultativos-facultativos que pueden compartir experiencias similares; como entre los propios colectivos: usuarios-familiares-facultativos, proporcionando un nuevo nivel de relación.

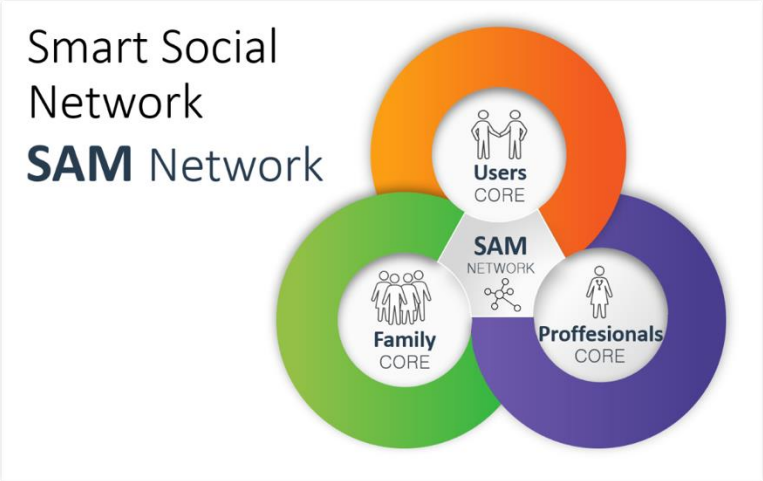


Figura 8. Red Social (SAM Network) especializada para familiares y acompañados.
FUENTE: equipo de SAM.

En este apartado expondremos brevemente la arquitectura conceptual y la infraestructura básica de este proyecto (**SAM Project**) para entender posteriormente cómo se puede integrar nuestra propuesta (**SAM Device**) en él.

3.2 INFRAESTRUCTURA TI

Desde un punto de vista aplicado, el soporte tecnológico de **SAM Project** lo proporciona la ***infraestructura TI*** sobre la que se ofrece el *Servicio de Acompañamiento a Mayores*. Esta infraestructura hace referencia a un conglomerado de modelos, servicios, dispositivos físicos y aplicaciones de software que proporcionan soporte tecnológico y metodológico a la estrategia de asistencia social, facilitando la creación ágil y eficaz de servicios de valor agregado que favorezcan la comunicación entre los usuarios, sus familiares y los profesionales que los atienden. De igual forma, estas infraestructuras tienen un potente impacto sobre las estrategias de actuación, y definen cómo se proporcionarán los servicios, determinando, incluso, la estructura de trabajo más adecuada, por lo que antes de entrar de lleno en el plan de trabajo, realizaremos un breve análisis de estas.

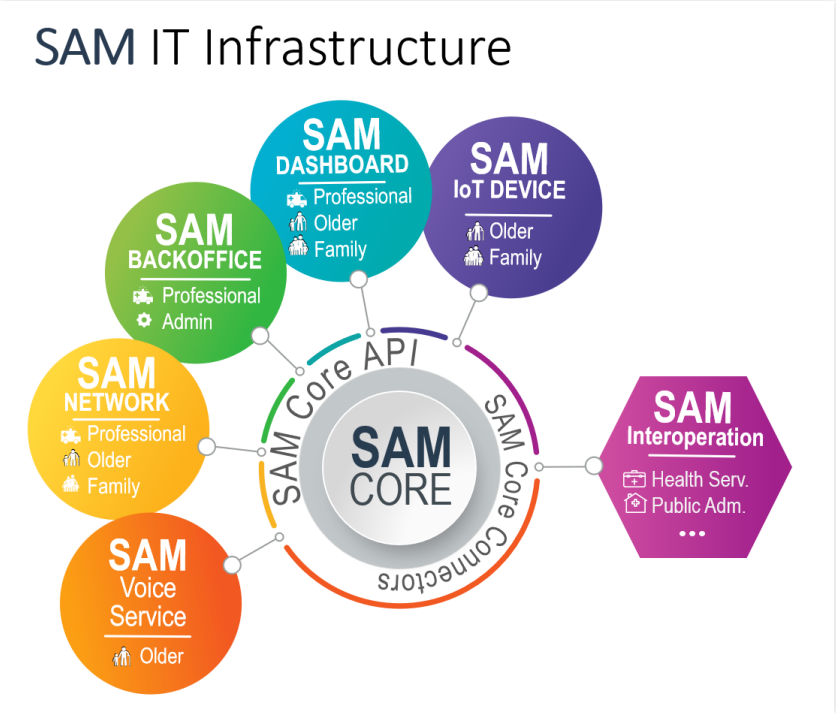


Figura 9. Infraestructura TI de SAM.
FUENTE: equipo de SAM.

En la **Figura 9** se pueden observar los componentes principales de la **Infraestructura TI** junto con la relación existente entre los mismos. Estos cinco grandes componentes se pueden resumir en:

- ***SAM Core***, que representa el núcleo fundamental del sistema, siendo el responsable de integrar el resto de los componentes, incluido los sistemas de información del *Servicio de Acompañamiento a Mayores*, y los dispositivos personales de acompañamiento o **SAM Device**. **SAM Core** proporciona la base de la **Plataforma TI** en forma de *servicios en la nube* que podrán ser consumidos a través de *API abiertas*.
- ***SAM Backoffice*** es un *Software como Servicio* (*SaaS— Software as a Service*) cuya funcionalidad consiste en proporcionar las herramientas necesarias para gestionar la plataforma en la nube y el sistema de información que precisa el Servicio de Acompañamiento a Mayores. Aunque quizá sea la pieza menos glamurosa del servicio, es fundamental para su futura viabilidad y verdadera integración con otras herramientas y sistemas del Sistema Nacional de Salud o de posibles organizaciones que estuvieran interesadas en incorporar el servicio.

- ***SAM Dashboard*** es un *SaaS* que proporciona soporte para la toma de decisiones y proporciona la herramienta de interacción de la red social (**SAM Network**). La funcionalidad y la información de la aplicación se ajustan al perfil del usuario en función de si es un usuario, un familiar o un profesional. Todos ellos podrán acceder al área de red social que permitirá la interacción entre los diferentes colectivos. Los familiares podrán acceder al estado en tiempo real de sus mayores, programar y recibir alertas, realizar análisis histórico, etc. Los profesionales tendrán un acceso similar al de los familiares, pero referidos a todos o parte de los mayores, pudiendo hacer un tratamiento del seguimiento y alertas por grupos.
- ***SAM Voice Service*** es la implementación, también en forma de servicio, del modelo de conversación que se creará para que la interacción entre el usuario y el servicio sea mediante lenguaje natural, incluso, interpretando sus sentimientos. Para desarrollar este modelo se tendrán en cuenta los aspectos de género y los estudios para su concepción se realizarán diferenciados por sexos, aplicando técnicas de IA para el reconocimiento.
- ***SAM Device*** es el nombre dado a los dispositivos personales que acompañarán a cada usuario apoyándose en el **SAM Voice Service**. Estos dispositivos son los responsables de que la experiencia de acompañamiento les resulte a sus usuarios lo más natural y lo menos invasiva posible.

De todos estos componentes, **SAM Device** y **SAM Voice Service** constituyen el objeto principal de este trabajo de fin de grado.

Además de los elementos indicados, también están involucrados en esta **Infraestructura TI** otros aspectos fundamentales como la seguridad informática, archivos y redes a todos los niveles, la virtualización de servidores y puestos de trabajo, el hosting de colocación, los servidores físicos, las redes de computadoras y el almacenamiento de los datos. Sin embargo, y puesto que en la actualidad el proyecto se deber orientar principalmente a la prueba de concepto y la consecución de los aspectos funcionales, nos vamos a centrar en los elementos tecnológicos que distinguen al *Servicio*, dejando de lado, en la medida de los posible, los detalles concretos sobre los aspectos antes descritos debido a que, por un lado, pueden ser resueltos con tecnologías y patrones arquitectónicos genéricos y que, por otra parte, pueden desviar la atención de lo verdaderamente relevante de **SAM Project** en esta fase de definición.

3.3 ARQUITECTURA CONCEPTUAL DE ALTO NIVEL

La arquitectura de **SAM Project** está basada en servicios y microservicios (véase **Figura 10**) buscando la máxima flexibilidad. De esta manera podemos incorporar tecnologías de diversos desarrolladores, modificar servicios sueltos sin causar cambios en el resto del sistema y reasignar los recursos en cualquier momento, cualidades perfectas para un sistema como este en el que cada módulo es un mundo.

En esta arquitectura ha sido clave tener suficientes microservicios para que cada tarea sea un módulo independiente que no genere acoplamiento en el resto de los servicios, pero sin ser demasiados como para complicar de manera innecesaria el desarrollo y mantenimiento del sistema.

La **Figura 10** muestra el diseño de la arquitectura de alto nivel de la actual implementación de **SAM Project** donde puede apreciarse que los servicios de voz (**SAM Voice Services**) y los dispositivos interfaz de usuario basados voz (**SAM Device**) todavía no han sido incorporados.

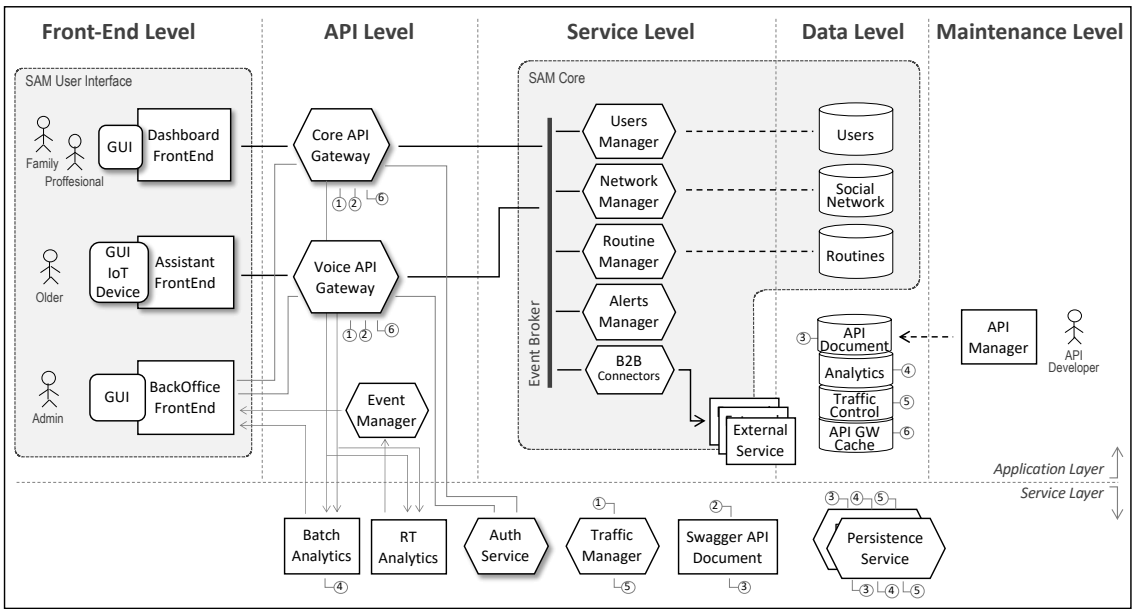


Figura 10. Arquitectura de alto nivel de SAM Service.
FUENTE: equipo de SAM.

Se ha empleado un estilo arquitectónico basado en *arquitecturas de n-niveles* que estructura los elementos en niveles y capas. Los elementos se definen fundamentalmente como servicios, siguiendo un patrón arquitectónico basado en Arquitecturas Orientadas a Servicios (SOA) y Microservicios. Así pues, se detallan las más relevantes, junto con los principales servicios, que

se precisan para el desarrollo adecuado del servicio de acompañamiento. Según esto, la capa de aplicación se ha dividido en los niveles de **Front-End**, **API, Service**, **Data** y **Maintenance**.

Cada nivel tiene un propósito específico:

- El ***Front-End Level*** define los elementos necesarios para interactuar con los usuarios finales aglutinados bajo el concepto de *SAM User Interface* y organizados mediante el patrón arquitectónico Modelo, Vista Controlador (MVC).

- El ***API Level*** contiene los servicios que actúan como *Application Programming Interface* (API) del resto de los servicios y microservicios que proporciona la plataforma, centralizando la exposición de todos los *endpoints* definidos en el *backend*, incluido el control del acceso, que será proporcionado finalmente por el Auth Service ubicado en la Service Layer, basado en una estrategia *OAuth*.

- En el ***Service Level*** encontramos el *core* de la plataforma, agrupado en dos grandes módulos: *SAM Core* y *SAM Voice Service*. Este nivel actúa como *Software as a Service* (SaaS) proporcionando toda la funcionalidad que precisa el *SAM Service*.

- Cada uno de los servicios ubicados en el *Service Level* accede al ***Data Level*** a través de los *Persistence Services* ubicados en la *Service Layer* (en el diagrama de la **Figura 10** se muestra la relación entre los servicios y sus principales fuentes de datos mediante líneas discontinuas para ilustrar que dicha relación es indirecta).

- En el **Maintenance Level** encontramos tareas de mantenimiento del sistema.

En los siguientes apartados describiremos con más detalle cada uno de estos niveles para entenderlos mejor.

3.4 FRONT-END LEVEL

El ***Front-End Level*** define los elementos necesarios para interactuar con los usuarios finales aglutinados bajo el concepto de *SAM User Interface* y organizados mediante el patrón arquitectónico Modelo, Vista Controlador (MVC).

Se han identificado tres tipos de interfaces:

- La interfaz ***Assistant Front-End***. Es la interfaz con los mayores. La interfaz desarrollada hasta el momento es muy sencilla y está basada en un GUI, pero el objetivo siempre ha sido desarrollar una interfaz especialmente diseñada para los mayores y proporcionada por un dispositivo específicamente diseñado para ellos. Este objetivo es justamente el que se aborda en este TFG. La idea es diseñar una interfaz de usuario basada en voz (VUI) proporcionada por un *IoT Device* (**SAM Device**). Este dispositivo estará especialmente concebido para acompañar a los mayores, aunque también puede ser muy útil para los familiares que quieran acceder a los servicios de SAM a través de la interfaz de voz.

- *El* ***Dashboard Front-End*** es la interfaz para los profesionales de la salud y para los familiares. Está basada en una interfaz gráfica de usuario (GUI) diseñada como una aplicación de página única (SPA —*Single Page Application*) accesible desde cualquier dispositivo que posea un navegador Web. Es la interfaz actual del sistema. En la **Figura 11** se pueden ver algunas capturas de pantalla a modo de ejemplo de esta completa interfaz cuando funciona sobre un teléfono móvil. Igualmente, en la **Figura 12** se muestra un conjunto de capturas de esta misma interfaz accedida desde un equipo de sobremesa.

- El ***BackOffice Front-End*** pensado para que los administradores técnicos del servicio puedan acceder a las funcionalidades de gestión, configuración, alertas y análisis del sistema, y gestión del tráfico (apoyándose en los servicios *RT Analytics*, *Batch Analytics* y *Traffic Manager* ubicados en la *Service Layer*) y al servicio de generación de alertas en tiempo real.

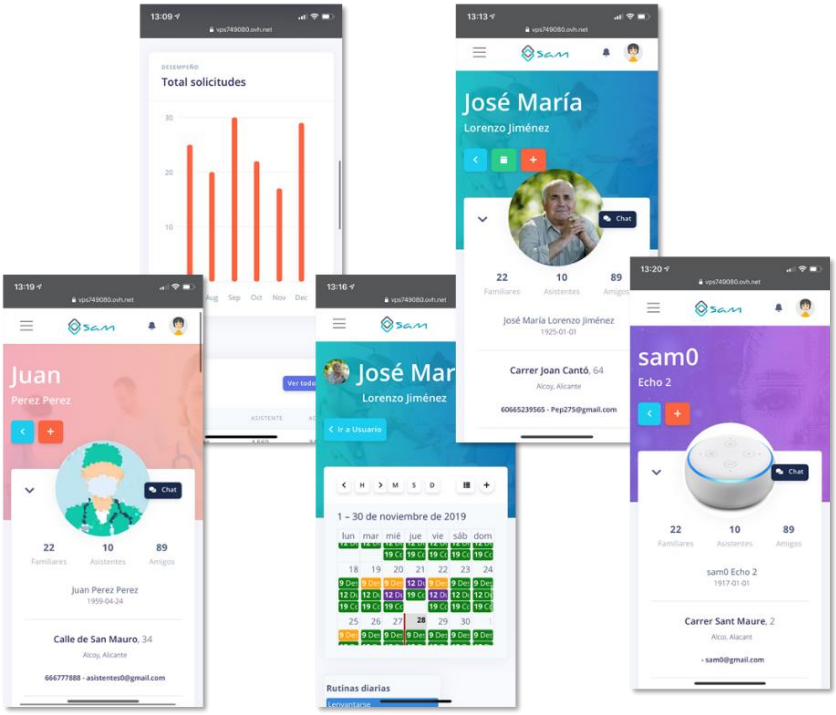


Figura 11. Dashboard de SAM en móviles.
FUENTE: elaboración propia.

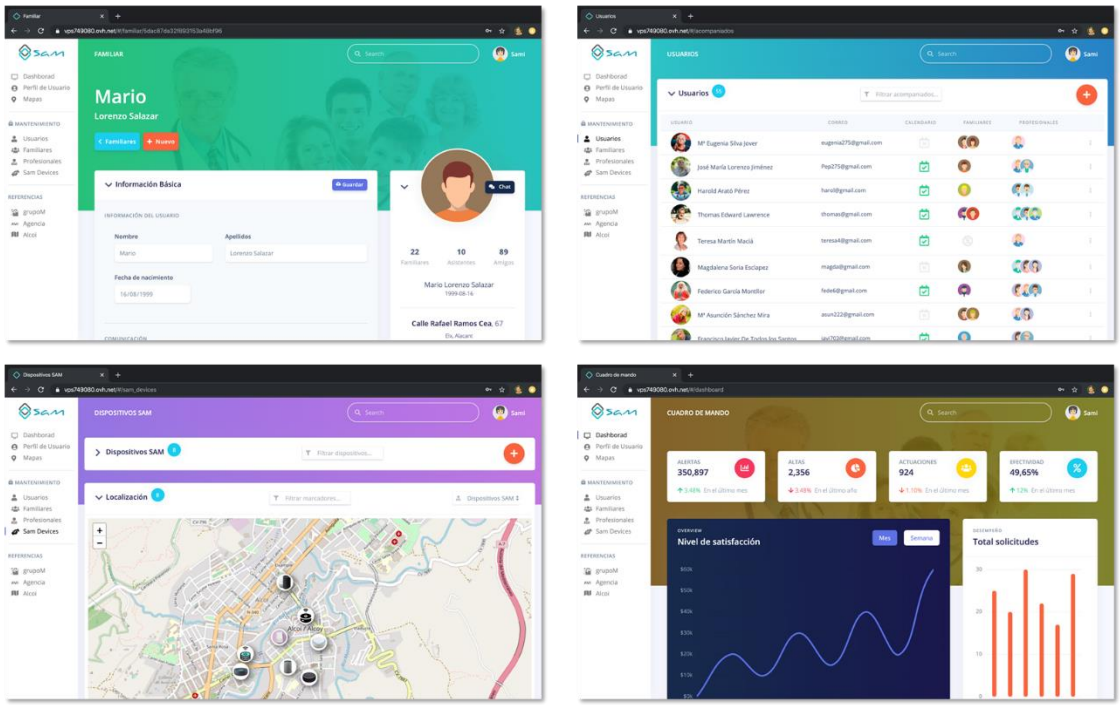


Figura 12. Dashboard de SAM en ordenador.
FUENTE: elaboración propia.

El acceso de los usuarios, con independencia de su rol (asistido, familiar, profesional, administrador), se realiza siempre a través del *Auth Service* ubicado en la *Service Layer*, pasando por el *API Gateway Service* que centralizará el acceso a todos los *end-points* del sistema. Todo el proceso de autenticación y autorización se basará en la utilización de tokens y será gestionado por el *Auth Service*.

3.5 API LEVEL

El ***API Level*** contiene los servicios que actúan como *Application Programming Interface* (API) del resto de los servicios y microservicios que proporciona la plataforma, centralizando la exposición de todos los *endpoints* definidos en el *backend*, incluido el control del acceso, que será proporcionado finalmente por el Auth Service ubicado en la Service Layer, basado en una estrategia *OAuth*.

Tal y como se puede observar en el esquema arquitectónico que se presenta en la **Figura 13**, se han identificado dos grandes servicios (*Core API Gateway* y *Voice API Gateway*) que actúan como punto de entrada a *SAM Core* y a *SAM Voice Service* respectivamente, ambos ubicados en el *Service Level*.

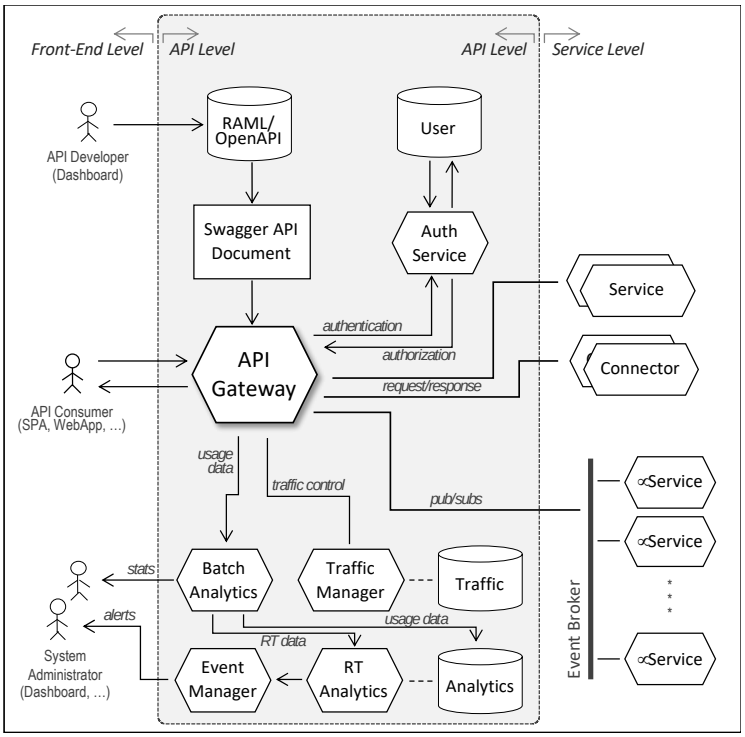


Figura 13. Arquitectura API Gateway de SAM.
FUENTE: equipo de SAM.

Estos *API Gateways* alimentan a los servicios *RT Analytics* y *Batch Analytics* con los datos de uso para su monitorización en tiempo real o para un posterior análisis forense. También se trata y se almacena la información para el control del tráfico a través del *Manager Traffic Service*.

Igualmente, toda la información de tráfico es gestionada por el Traffic Manager que proporciona un sistema de escalado automático para gestionar el volumen de tráfico recibido. Este sistema está conectado con la gestión de la caché.

Para facilitar y agilizar la gestión de los end-points se ha definido un *Swagger API Document Service* ubicado en la *Service Layer* que gestiona la documentación necesaria para el descubrimiento de los servicios ofrecidos por la plataforma y está enlazado con el Maintenance Level a través del *API Manager*.

3.6 SERVICE LEVEL

En el ***Service Level*** encontramos el *core* de la plataforma: ***SAM Core***. Este nivel actúa como *Software as a Service* (SaaS) proporcionando toda la funcionalidad que precisa el ***SAM Service***.

SAM Core está compuesto por un conjunto de servicios que proporcionan la funcionalidad básica de *SAM Service: Users Manager, Network Manager, Routine Manager* y *Alerts Manager*. Todos estos servicios están interconectados con el Gateway y entre sí a través de un *Event Broker* que gestiona los mensajes. Es en este nivel en el que se pueden definir diferentes conectores (a través de los *B2B Connectors*) que permiten la integración del *SAM Service* con otros sistemas como, por ejemplo: el sistema de salud pública, los servicios sociales de las administraciones locales, o el sistema de gestión de una compañía privada que preste servicios de acompañamiento. Estos conectores también permiten la interacción del *SAM Service* con otras plataformas de servicios y aplicaciones de terceros.

3.7 DATA LEVEL

Cada uno de los servicios ubicados en el *Service Level* accede al ***Data Level*** a través de los *Persistence Services* ubicados en la *Service Layer*.

Las principales bases de datos directamente relacionadas con la funcionalidad del *SAM Service* son: *Users* (principalmente mantiene la información de los acompañados, familiares y

profesionales), *Social Network* (gestiona toda la información de SAM Network) y *Routines* (almacena las diferentes rutinas de salud, estilo de vida, toma de medicamentos, y cualquier tipo de evento que se desee programar para los asistidos).

Las bases de datos relacionadas con *SAM Voice Service* son: dos *Voice Model* (Uno para ASR que contiene el modelo de voz utilizado para convertir el streaming de audio en texto y otro para TTS que convierte el texto en streaming de audio), *Dialogue Model* (responsable de identificar las *intents* y *entities* de una conversación, así como de ayudar a la toma de decisiones) y *Tracker Store* que permite guardar todas las conversaciones con el objetivo, para poder tanto realizar un análisis forense, en caso de requerirse, como realimentar el *Dialogue Move Engine* (DME) para mejorar su comportamiento.

Finalmente, en este nivel se ubican también las bases de datos que facilitan la gestión y administración de todo el sistema: *API Documents*, *Analytics*, *Traffic Control* y *API GW Cache*.

3.8 MAINTENANCE LEVEL

Aunque hay muchos tipos de tareas de mantenimiento, hemos resaltado en la arquitectura dos de ellas por la especial importancia que tienen para el correcto funcionamiento de todo el sistema.

La primera de ellas es el *API Manager*. Al basar la propuesta en una arquitectura de microservicios, desacoplando totalmente el *backend* del *frontend*, es fundamental contar con un mecanismo ágil para mantener las API de los servicios, sobre todo de los gateways. El *API Manager* permite crear y mantener *endpoints*, generar y actualizar documentación y código, y gestionar versiones.

Es igualmente importante para el sistema la gestión de los modelos de diálogo que deben pasar por una constante actualización. Puesto que están basados en herramientas de IA, concretamente en *Neural Networks* (NN) y otras heurísticas asociadas, precisan de un entrenamiento previo que debe ser contemplado dentro de todo el proceso. Estas tareas se desarrollan por los *Data Expert* a través del módulo *NN Training*.

3.9 BACK-END

De nuevo volvemos al Back-End Level. Es en este nivel en el que trabajé cuando estuve como becaria. Nuestro trabajo consistió en realizar pruebas sobre el Back-End implementado por el *grupoM* de manera directa, usando Postman, y cuando estuvo la versión final nos encargamos de insertar datos de prueba, usuarios ficticios con todos sus datos para probarlos en el Dashboard.

3.10 CONCLUSIÓN

Contamos con una plataforma tecnológica desarrollada y puesta en producción que facilita el acompañamiento a mayores y a personas dependientes por parte de los servicios sociales de los ayuntamientos y de sus familias.

Aunque todo el ecosistema cumple con sus objetivos de conectar a todos los implicados en el proceso, no tiene en la actualidad una interfaz diseñada desde el inicio desde el punto de vista puesto en sus usuarios: nuestros mayores.

La propuesta de este trabajo consiste, pues, en la creación de un asistente virtual inteligente basado en voz que hemos denominado **SAM Device**. Este dispositivo es el encargado de acompañar a los mayores al entretenerlos, monitorizarlos, y asistirlos. El dispositivo se ubicaría en las casas de los mayores y podría estar con ellos las veinticuatro horas del día para acompañarlos.

La idea es que en lugar de actuar como un dispositivo IoT convencional que está pensado para pasar desapercibido, SAM Device estará ahí para acompañar al mayor como uno más, integrado en la vida cotidiana de la persona mayor y dedicado a asistirla. Esa interfaz es la que abordaremos en este proyecto.

Teniendo en cuenta los componentes mostrados en la **Figura 9** que forman parte del proyecto SAM, este TFG desarrollará dos de ellos para lograr sus objetivos: **SAM IoT Device** y **SAM Voice Service**.

4 Objetivos y metodología

Analizando todo lo descrito en los apartados anteriores, se puede entender *que es importante para la sociedad atender a sus mayores, que la esperanza de vida y la calidad de vida aumentan cada vez más, que los mayores desean mayor autonomía, que son cada vez más capaces de asimilar tecnología, y que la tecnología es cada vez más madura y amigable*.

Sin embargo, también es cierto *que los mayores siguen necesitando atención y cuidado, y que el actual ritmo de vida hace muy complicado, incluso imposible, que el peso del cuidado pueda recaer en sus familias*.

Por lo tanto, las principales conclusiones que se extrae es que *las soluciones actuales son insuficientes, o si son suficientes entonces son demasiado caras, difícilmente sostenibles en el tiempo y quedan al alcance de unos pocos, que los cambios demográficos agravarán esta situación si no se adoptan medidas, que esta situación puede ser una oportunidad si se aborda con tiempo, y que las tecnologías pueden ayudar a proporcionar soluciones mucho más ajustadas a las necesidades cambiantes de los mayores y, sobre todo, más asequibles y sostenibles*.

Con todo ello, **la solución** que se propone en este trabajo consiste en **crear** un dispositivo digital (**SAM Device**) para el acompañamiento a los mayores que les proporcionará una interfaz basada en voz mediante lenguaje natural, apoyado en una plataforma software en *la Nube* que proporciona un servicio de voz (**SAM Voice Service**) que aporta la Inteligencia Artificial necesaria para el funcionamiento de dicho dispositivo.

4.1 OBJETIVO

Este proyecto busca mejorar la calidad de vida de nuestros mayores mediante la creación de un Servicio TI sostenible de Acompañamiento a Mayores y a personas dependientes a través de un dispositivo IoT (**SAM Device**) que los conecta con su red de familiares, cuidadores y personal de la salud, realizando las labores de acompañamiento a través de un asistente personal inteligente que les atenderá las veinticuatro horas del día, los siete días de la semana, proporcionándoles una interfaz basada voz y en lenguaje natural (**SAM Voice Service**).

Teniendo en cuenta todo lo anterior, **el objetivo general** de este trabajo se puede definir, de forma resumida como:

- Construir un dispositivo inteligente, capaz de asistir mediante interfaces basadas en lenguaje natural a las personas mayores y a dependientes, veinticuatro horas al día, y servir de interfaz para facilitarles la comunicación con sus familiares y los profesionales que les atienden, mejorando así la situación de los tres grupos (mayores sin atención, familiares preocupados y profesionales saturados). Este dispositivo lo hemos denominado **SAM Device**.

El dispositivo tendrá que integrarse en un ecosistema existente que es complejo y diverso (**SAM Project**). Además, para lograr la interfaz de lenguaje natural, dada también su complejidad, deberá estar diseñada para trabajar como un servicio ofrecido desde servidores suficientemente potentes en la Nube (**SAM Voice Service**). Para lograr este objetivo general, se ha definido un conjunto de **objetivos concretos** que se resumen en:

- Diseñar una arquitectura hardware para un dispositivo IoT (***SAM Device***) que actúe de interfaz.
- Diseñar una arquitectura software para el análisis y la síntesis de voz natural, además de diseñar una arquitectura distribuida para proveer el Servicio de Acompañamiento a Mayores a través de *SAM Device* (**SAM Voice Service**).
- Crear un prototipo funcional tanto del dispositivo hardware de acompañamiento, como de todos los servicios basado en IA que se desplegarán en la Nube.

Se ha definido también una serie de **objetivos complementarios** que serán importantes para realizar todo el trabajo:

- Analizar el estado de la técnica, con especial atención a la situación de nuestros mayores en la actualidad y a su evolución durante los próximos 25 años, y a las tecnologías que podrían proporcionar soluciones: hardware IoT, IA para el análisis y síntesis de voz, protocolos de comunicación y arquitecturas distribuidas.
- Establecer una hoja de ruta y un plan de trabajo acordes con el tiempo del que se dispone.
- Diseñar un conjunto de experimentos que permitan validar diferentes aspectos del sistema propuesto, junto con un plan de pruebas del sistema y de integración con otros sistemas y servicios existentes.

4.2 METODOLOGÍA

Durante el desarrollo de este TFG se han empleado diferentes métodos y técnicas para adaptarse lo mejor posible al desarrollo de cada tipo de tarea.

Según esto, para el estado de la técnica se ha seguido una metodología estandarizada de búsqueda de información que se puede encontrar en (Biblioteca UA, 2013). Esta metodología consiste en cinco fases bien diferenciadas: análisis y definición de la necesidad de información; nivel y cobertura de la búsqueda; selección de las fuentes de información; elaboración de la estrategia de búsqueda; y, finalmente, valoración de los resultados y gestión de la información recuperada. En el **apéndice 12.1** se muestran las tablas resumen que recogen: la concreción de los parámetros de análisis y la definición de las necesidades de información (**Tabla 11**); la definición del nivel y la cobertura de la búsqueda realizada (**Tabla 12**); la definición de las fuentes de información seleccionadas (**Tabla 13**); y por último, la concreción de los parámetros de búsqueda (**Tabla 14**).

Para el desarrollo y la implementación del sistema propuesto se ha utilizado la metodología ágil *Scrum*, principalmente porque está pensada para proyectos que tienen que ofrecer resultados en poco tiempo e iterar sobre estos resultados, gracias a la división del tiempo de trabajo en iteraciones cortas, de generalmente una semana, llamadas *sprints* (Trigás, 2012). Esto es importante para nuestro proyecto porque hay que crear muchos componentes y unirlos desde el principio, y de ahí se iterará sobre los que más afecten al funcionamiento global o los que más rápido puedan salir. Scrum está pensada para trabajo en equipo, pero es aplicable igualmente a equipos de una sola persona, como es este caso, sin perder los beneficios que aporta.

Para la creación de la memoria y la gestión del proyecto en general se ha utilizado *Kanban*, ya que esta metodología se centra en el estado del proyecto y no requiere de la creación de *sprints* que deben cerrarse en un tiempo fijo, permitiendo así reunirse con el tutor o pararse a reflexionar solo cuando el proyecto avanza (Castellano Lendínez, 2019). Esta flexibilidad ha sido clave, pues parte de este trabajo se ha realizado en paralelo al curso académico en el que se realizaba un trabajo de *Aprendizaje Basado en Proyectos* (ABP) y la cantidad de tiempo que se le podía dedicar a cada trabajo paralelo era muy arbitraria.

De esta manera, mezclando *Scrum* y *Kanban* (lo que en la literatura se conoce como *Scrumban* (Rojas, 2019)), se ha logrado tener la flexibilidad de *Kanban*, pero con la precisión de las iteraciones de *Scrum*, aplicadas al desarrollo en segmentos del curso en los que podía dedicarme más tiempo o tenía más certeza en cuanto al tiempo disponible.

Adicionalmente, y a pesar de que en este proyecto tan solo hemos trabajado mi tutor y yo, dada la gran complejidad y volumen de tecnologías, lenguajes de programación y entornos y *frameworks* de desarrollo que están involucrados, ha sido muy importante incorporar algunas técnicas para reducir los tiempos de desarrollo, despliegue y prueba, al tiempo que se mejoraba la eficiencia y se flexibilizaba la incorporación de cambios en la planificación. Es por ello por lo que toda la fase de desarrollo, despliegue, integración y validación se ha realizado incorporando técnicas de: *integración continua* (*Continuous Integration*), *entrega continua* (*Continuous Delivery*) y *despliegue continuo* (*Continuous Deployment*) (Bobrovskis y Jurenoks, 2018).

5 Estudio de viabilidad

Aunque hayamos estudiado a fondo las tecnologías que queremos utilizar, un proyecto de una envergadura aceptable siempre correrá el riesgo de que algo salga mal y de no triunfar. Por ello siempre es importante realizar un estudio de viabilidad para identificar los peligros que puedan afectar al proyecto y anticiparse a sus posibles consecuencias.

No existe una manera única de realizar un estudio de viabilidad. Sí hay métodos estándar que ahorran reinventar la rueda a la hora de estudiar la viabilidad del proyecto. Nosotros realizaremos un análisis DAFO para identificar nuestros puntos fuertes (los que hay que potenciar) y nuestros puntos débiles (los que hay que cuidar).

5.1 Análisis DAFO

Un análisis **DAFO** consiste en identificar los aspectos tanto positivos como negativos de las características internas de nuestro proyecto, es decir las que imponemos nosotros con esta idea (llamadas **Fortalezas** y **Debilidades**, respectivamente) y también los aspectos tanto positivos como negativos de las características externas al proyecto, es decir, las que nos afectan por como es el entorno o el mercado, según el tipo de proyecto (conocidas como **Oportunidades** y **Amenazas**, respectivamente).

Debilidades, Amenazas, Fortalezas y Oportunidades, en ese orden, dan lugar a las siglas DAFO.

Tabla 6. Matriz DAFO del proyecto.
FUENTE: elaboración propia.

	Características positivas	Características negativas
Características internas	**Fortalezas** - Apoyo de ayuntamientos - Basado en tecnologías maduras - Proyecto ya empezado en el que he participado	**Debilidades** - Falta de experiencia en las tecnologías utilizadas - Compaginado con el proyecto grupal ABP
Características externas	**Oportunidades** - Resuelve un problema de carácter social y por lo tanto afecta a todos	**Amenazas** - Rechazo a las tecnologías por parte de los usuarios

Comenzaremos hablando de nuestras características internas. En primer lugar, nuestras **fortalezas** consisten en que el proyecto está basado en tecnologías maduras, tal y como hemos visto en el estado de la técnica, con lo cual evitamos riesgos basados en la incertidumbre sobre tecnologías experimentales. Además, este trabajo es parte de un proyecto mucho más grande en el que estuve trabajando, con lo cual no es algo que empiece de cero si no que ya estaba familiarizada con muchos de sus aspectos. Por último, es un proyecto real que tiene el apoyo de instituciones reales (en este caso los ayuntamientos), con lo cual tiene una utilidad más allá del ámbito académico.

Siguiendo con las características internas, tenemos los puntos negativos, nuestras **debilidades**. Principalmente, nuestra mayor debilidad es la falta de experiencia con las tecnologías a utilizar (tecnologías hardware, tecnologías de voz y para procesamiento del lenguaje natural, basadas ampliamente en técnicas y algoritmos de Inteligencia Artificial). Esta falta de experiencia puede causar retrasos o avanzar por el camino equivocado. Además, el desarrollo de este proyecto se compagina con el desarrollo de un proyecto grupal de fin de carrera. Es un proyecto muy ambicioso que también consume mucho tiempo y según se desarrollen las diferentes tareas, puede afectar a este trabajo.

A continuación, pasamos a las características externas. Por la parte positiva tenemos nuestras **oportunidades**. Nuestra principal oportunidad radica en que el proyecto trata de dar solución a un problema de carácter social que nos afecta a todos y que, como se puede ver claramente tras el estudio de la técnica, es de amplio interés para los gobiernos: la vejez y el envejecimiento saludable, y por ello tiene un amplio público de destino y resulta de mucho interés para la sociedad general. Desde un punto de vista personal, el proyecto incorpora también dos grandes oportunidades: sentir que puedo ayudar con los conocimientos adquiridos a mis seres queridos (a mi *iaia* y a mi *abuela*) y que podré aprender y aplicar muchas tecnologías, muy diferentes, de campos también muy variados (hardware, software, sistemas distribuidos, inteligencia artificial, …).

Por último, tenemos la parte negativa de las características externas: las **amenazas**. La principal amenaza a la que nos enfrentamos es que los mayores suelen rechazar las nuevas tecnologías. Aunque en el estado de la técnica hemos visto que las interfaces de voz suelen tener muy buenos resultados, el rechazo de personas mayores que no hayan probado todavía estas tecnologías y no estén dispuestas a probarlas puede afectar al avance y a la implantación del proyecto. Es por esta razón que hemos dedicado mucho tiempo del diseño y especificación en trabajar de forma conjunta con los servicios sociales de los ayuntamientos para entender lo mejor posible las necesidades y los requisitos para lograr una buena experiencia de acompañamiento a nuestros mayores.

6 Planificación

6.1 Planificación inicial

Al principio del curso académico, conforme la idea terminó de asentarse, se realizó una planificación que abarcaba todo el periodo lectivo para organizar las tareas del TFG y presentarlo en la convocatoria de junio. La **Tabla 7** muestra dicha planificación de manera resumida.

Tabla 7. Planificación para junio (8 meses).
FUENTE: elaboración propia.

%	Actividad	Tiempo	Fecha límite
05%	Planteamiento / Identificación del problema	2 semanas	18 de octubre
30%	Estado de la técnica	2mes 2sem	3 de enero
10%	Diseño de la solución	3 semanas	24 de enero
30%	Implementación	2mes 2sem	4 de abril
20%	Pruebas y validación	1mes 1sem	9 de mayo
05%	Presentación	2 semanas	23 de mayo

Sin embargo, a lo largo de este curso han ido apareciendo diferentes contratiempos que han obligado a aplicar los planes de contingencia y a replanificar el trabajo.

6.2 Contratiempos y plan de contingencia

Como ya se ha comentado, durante la ejecución del proyecto fueron apareciendo diferentes contratiempos que han ido condicionando la ejecución de la planificación. En este apartado

analizaremos dichos inconvenientes y expondremos las soluciones que se adoptaron para cada caso.

El primer inconveniente se produjo al estar también trabajando en el proyecto ABP (*Aprendizaje Basado en Proyectos*) del cuarto curso de Ingeniería Multimedia. Este proyecto, tal y como se temía en las debilidades del análisis DAFO (véase **capítulo 5.1**), consumió mucho más tiempo del que creíamos en un inicio y, al ser grupal, dejarlo de lado para centrarme en este TFG suponía dejar a mis compañeros con un componente menos.

El resto de los inconvenientes fueron provocados directa o indirectamente por la situación de pandemia que estamos viviendo:

- No hubo posibilidad de adquirir la mayor parte de los componentes electrónicos y placas de desarrollo para crear el dispositivo físico.
- El Ayuntamiento detuvo, en una primera instancia, las pruebas reales de campo con los mayores por razones evidentes de seguridad.
- En una segunda fase, los servicios sociales del Ayuntamiento, debido al enorme volumen de trabajo que tenían, decidieron dejar aparcado el proyecto hasta que la situación permitiera volver a retomarlo. Esta decisión nos afectó notablemente debido a que ellos eran la principal fuente que quedaba para elaborar y refinar los modelos de conversación que son fundamentales para este trabajo.

En el caso de la sobrecarga provocada por el ABP, se decidió aumentar el tiempo definido para el proyecto alargándolo hasta la convocatoria C4 (septiembre). Para ello se realizó una replanificación de las tareas.

El segundo problema, no disponer del material electrónico planeado, se resolvió rehaciendo los diseños hardware de los prototipos aplicando los componentes que se pudieron localizar de diferentes fabricantes. Gracias a que todos los diseños estaban basados en Open Hardware, este rediseño fue posible y se pudo realizar en pocos días. Igualmente, puesto que se produjo un importante retraso en la recepción del material, se replanificaron las tareas involucradas para ajustarlas adecuadamente.

En cuanto a la suspensión de las pruebas de campo, sencillamente no se han podido realizar. Como sustitución, he realizado alguna prueba con mis propias abuelas. Aunque esto no sustituyen ni de cerca el estudio de campo, al menos me ha dado una interesante realimentación

del funcionamiento del dispositivo (**SAM Device**) con los usuarios reales. Nuevamente, se recogieron estos cambios en la planificación inicial.

Finalmente, las entrevistas para la realización de los modelos de conversación, a pesar de haberse detenido, ya teníamos una importante base de datos de proyectos anteriores. En este caso, acudimos a dicha información como plan de contingencia para elaborar un modelo que fuera válido y realista. Queda pendiente mejorar dicho modelo, pero, en cualquier caso, esta mejora del modelo es continua y se deberá hacer siempre. Es por esto por lo que se ha diseñado el sistema incorporando y facilitando las tareas de entrenamiento de las redes neuronales de forma que se pueda realizar esta tarea, aunque no se tengan especiales conocimientos en inteligencia artificial.

6.3 Replanificación y Plan de ejecución

La **Tabla 8** muestra la nueva planificación establecida para la convocatoria de septiembre. Esta nueva planificación se realizó a finales de enero (29 de enero). Para entonces ya se había resuelto el planteamiento del problema, pero el estado de la técnica se había retrasado un mes. La planificación abarca también 8 meses, de finales de enero a principios de septiembre.

Tabla 8. Replanificación para septiembre (8 meses).
FUENTE: elaboración propia.

%	Actividad	Tiempo	Fecha límite
	Nuevo inicio	-	29 de enero
00%	Planteamiento / Identificación del problema	-	Hecho
30%	Estado de la técnica	2mes 2sem	9 de abril
15%	Diseño de la solución	1mes 1sem	14 de mayo
45%	Implementación	3mes 1sem	13 de agosto
05%	Pruebas y validación	2 semanas	27 de agosto
05%	Vídeo presentación y conclusiones	2 semanas	10 de septiembre

Nótese que en ambas planificaciones la fecha final es unos días antes de la entrega. Estos días de más se dejaron para amortiguar el impacto de imprevistos menores.

El diagrama Gantt con la planificación y ejecución final del proyecto queda recogido en la **Figura 14**. Básicamente, hasta marzo se realizó la investigación del estado de la técnica y el estudio de viabilidad, al tiempo que se iban fijando los objetivos concretos del trabajo y la metodología a utilizar. De abril a mayo se realizó el estudio de viabilidad y se diseñó la solución al problema. De mediados de mayo a mediados de agosto se realizó la implementación tanto del prototipo hardware como de los servicios software. Durante la segunda quincena de agosto se comenzó con las pruebas de integración. Primero cada módulo por separado, y después agregando funcionalidad. En la última semana se cerró la memoria y se comenzó a preparar el guion del vídeo de presentación.

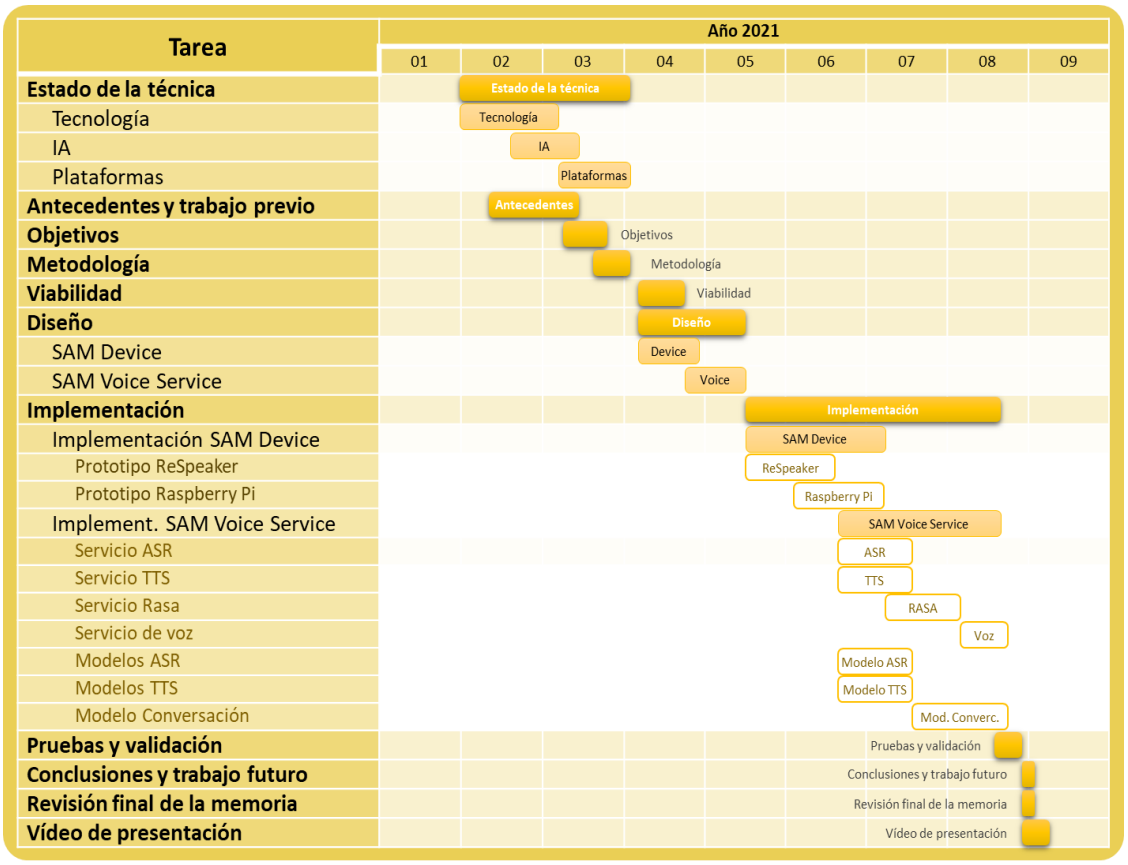


Figura 14. Diagrama Gantt con la planificación para septiembre.
FUENTE: elaboración propia.

En la **Figura 15** se presenta una captura de pantalla de la aplicación Trello, la aplicación empleada para gestionar el desarrollo del trabajo. En las capturas se pueden ver las listas que componen el tablero ordenadas por meses. En ellas se recogen las tareas planificadas, subdivididas a su vez en tareas menores en listas internas a cada tarjeta.

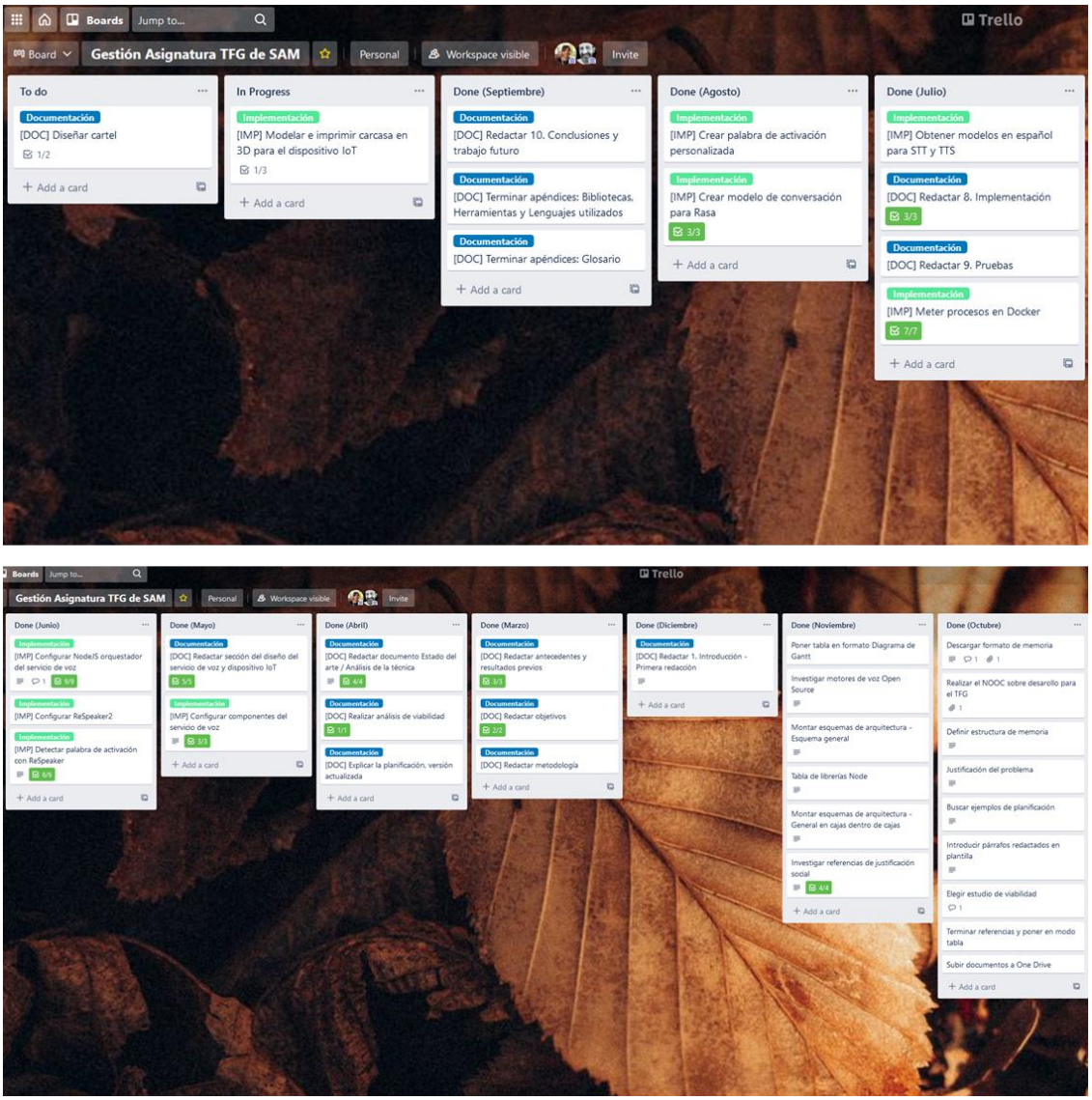


Figura 15. Tablero de Trello con las tareas realizadas hasta septiembre.
FUENTE: elaboración propia.

7 Análisis, especificación y diseño

Una vez planteado el proyecto global, los objetivos del trabajo, y habiendo estudiado su viabilidad, cuál es la mejor metodología para su desarrollo, y habiendo establecido la planificación, llega el momento de diseñar con detalle nuestra propuesta de solución, es decir, qué vamos a crear en este trabajo de fin de grado.

Como se ha explicado a lo largo de la memoria, este *Trabajo de Fin de Grado* se encarga de desarrollar dos grandes módulos que hemos denominado ***SAM Device*** y ***SAM Voice Service***. Se crearán atendiendo a todo lo visto en el estado de la técnica siguiendo los objetivos establecidos.

En la primera sección de este capítulo diseñaremos la arquitectura global de todo el sistema. Después, en las siguientes secciones, diseñaremos cada componente en detalle.

Es importante recordar que se utilizará únicamente *software libre* (*OpenSource*), ya que las versiones comerciales de estas tecnologías comprometen muchas de las decisiones de diseño, incluyendo los aspectos de privacidad. Tal y como se analizó en el capítulo 2, al manejar estas empresas parte de los datos tratados en el proceso (conversaciones, datos personales…), impiden que podamos asegurar la confidencialidad de nuestros usuarios, además de estar limitadas en personalización. De la misma manera se utilizará *hardware libre* para tener la libertad de analizar y replicar los esquemas electrónicos y poder fabricar así los dispositivos IoT para su posterior uso sin depender de diseñadores externos. Igualmente, compartir tanto el diseño software como hardware puede ayudar a crear una comunidad que mejore y ayude a mantener y crecer el proyecto.

7.1 ARQUITECTURA CONCEPTUAL DE ALTO NIVEL

Para diseñar adecuadamente este trabajo debe tenerse en cuenta su función y ubicación en el marco global del proyecto de SAM. En la **Figura 10** (en el **capítulo 3.** *Antecedentes y resultados previos*) pudimos observar la arquitectura de alto nivel del proyecto, que se dividía en *Front-End Level*, *API Level*, *Service Level*, *Data Level* y *Maintenance Level*.

Cada uno de los niveles tiene un propósito específico. La arquitectura de alto nivel de este trabajo, que recordemos que abarca el dispositivo IoT denominado **SAM Device** y el servicio de voz (**SAM Voice Service**) que le da soporte, encaja de manera paralela en el esquema. La **Figura 16** muestra los elementos de este TFG que añadiremos a la arquitectura ya propuesta y que se han estructurado en los mismos niveles.

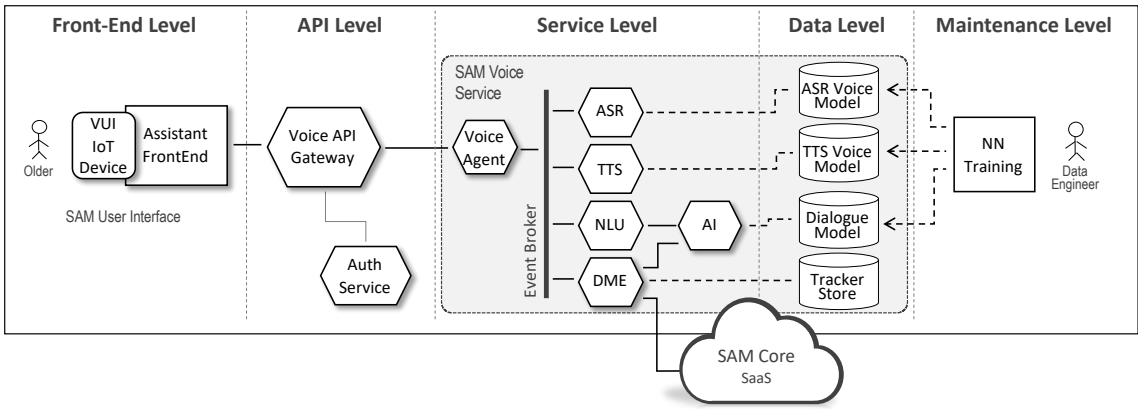


Figura 16. Arquitectura de alto nivel del sistema que se propone en el TFG.
FUENTE: elaboración propia.

Nótese (**Figura 16**) la nube de *SAM Core* representando precisamente la contraparte del *Service Level* vista en el esquema arquitectural de **SAM Project** (véase **Figura 10**). Los dos esquemas son paralelos, pero no independientes. **SAM Voice Service** utilizará **SAM Core** como *Software as a Service* (SaaS) para acceder a la información de sus usuarios y poder asistirles correctamente.

Los siguientes apartados, de **7.2** a **7.6**, explicarán en detalle cada nivel. La última sección, **7.8**, trata sobre la integración de esta propuesta de TFG dentro del proyecto global, proporcionando un esquema arquitectónico de alto nivel completo tras su incorporación.

7.2 FRONT-END LEVEL

El nivel de Front-End es el encargado de agrupar todos los elementos necesarios para la interacción con los usuarios, o sea las interfaces. Lo que nuestro trabajo hace es añadir otra interfaz a este nivel: la interfaz de usuario basada en voz (**VUI**) ofrecida por el dispositivo IoT (**SAM Device**), pensada para las personas mayores.

Este nivel abarca, por lo tanto, todo lo relacionado con el dispositivo IoT, tanto hardware como software, de su capacidad para detectar la palabra de activación, y de su interacción con el servicio de voz (el cual está ubicado en el *Service Level*, véase **sección 7.3**).

7.2.1 Arquitectura hardware de SAM Device

Antes de comenzar esta sección, recordemos que los componentes hardware elegidos (*ReSpeaker* y *Raspberry Pi*) actuarán como base para los prototipos que vamos a realizar en este TFG. En un futuro, si es necesario, desarrollaremos nuestra propia placa. Por eso es tan importante que los diseños hardware utilizados sigan una filosofía Open Hardware. Cuando queramos crear nuestros propios prototipos, contactaremos con una empresa especializada, a la cual se le envían los diseños (Schematics) de nuestras PCB (*Printed Circuit Board*), y se encargan de su fabricación, incluso de la soldadura de los componentes, si así se desea; por ejemplo, la propia *Seeed*, fabricante de nuestras placas de prueba *ReSpeaker* (ver **apéndices 12.2** a **12.4**).

Nuestra propuesta es que sea un dispositivo IoT el que actúe como asistente de voz (**SAM Device**), haciendo de interfaz entre el usuario y la compleja lógica que lleva detrás el servicio de SAM. Aunque mucha de la lógica del asistente se llevará al servicio de voz, las funcionalidades que se deben seguir desarrollando en local en el **SAM Device** no es banal. Este dispositivo se encarga de estar escuchando al usuario constantemente, esperando la llamada palabra de activación (que en este caso será “Oye Sam”). Cuando escuche esta palabra, empezará a grabar lo que escuche a continuación, y cuando detecte que el usuario ha dejado de hablar enviará el audio grabado al servicio de voz (**SAM Voice Service**) externo.

En el servicio de voz se desarrollará todo el análisis de la voz y la ejecución de las tareas encomendadas por el usuario (véase **sección 7.4.1**) y devolverá la respuesta del asistente de voz. Entonces, el dispositivo reproducirá la respuesta por los altavoces para que la escuche el usuario, quedándose de nuevo a la espera de la palabra de activación para recibir nuevas instrucciones.

La **Figura 17** muestra la arquitectura de alto nivel de hardware del dispositivo IoT con todos los elementos que necesita para desempeñar su función.

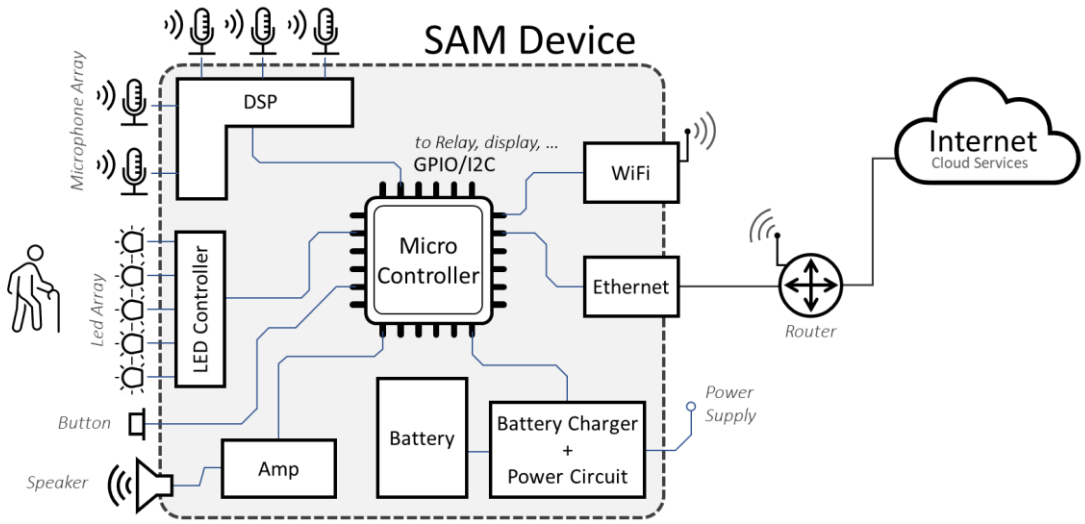


Figura 17. Diagrama de bloques con los principales componentes hardware de SAM Device.
FUENTE: elaboración propia.

El arreglo de micrófonos sirve para poder escuchar al usuario. Usamos un arreglo de micrófonos en lugar de un solo micrófono ya que esto permite capturar mejor el sonido, desde cualquier dirección y más resistente al ruido. Esto es especialmente útil porque parte del ruido podríamos generarlo nosotros (el dispositivo IoT) al estar hablando, reproduciendo música o haciendo sonar una alarma.

El altavoz sirve para reproducir la voz del asistente y otros sonidos de retroalimentación, como un sonido que indique que empieza a escuchar una orden, o bien música y alarmas puestas por el usuario o sus familiares.

Los leds están por motivos de retroalimentación, ya que se encargarán de mostrar distintos patrones de luz según lo que esté ocurriendo en la conversación. Por ejemplo, si está "hablando" el dispositivo (está reproduciendo la respuesta por los altavoces) hará brillar los leds de manera distinta a cuando está escuchando al usuario.

La entrada de alimentación sirve para alimentar al dispositivo. Adicionalmente convendría tener una batería pues, aunque está pensado para estar conectado a la electricidad todo el rato, puede haber momentos en que se desconecte, como apagones o mientras mueven el dispositivo de una sala a otra. Con una batería nuestro dispositivo podrá sobrevivir un tiempo sin estar

conectado y asegurar una cobertura en el tiempo ampliada. Para el dispositivo será transparente si su energía proviene de la batería o del suministro de la red eléctrica.

Las conexiones Wifi y ethernet son necesarias para que el dispositivo se conecte al servicio de voz en la nube, y para que pueda acceder a otros servicios, como los propios servicios de SAM Core (véase **sección 3.6.** *Service Level*) o servicios de terceros como vídeo bajo demanda, comunicaciones o entretenimiento.

Para el microcontrolador nos hemos decantado por una *arquitectura ARM*, es decir, con un juego de instrucciones reducido, debido a que son ideales para dispositivos IoT por ocupar mucho menos espacio y consumir y calentarse mucho menos que uno basado en una *arquitectura x86*. Aunque esto acarrea menor potencia, maximiza la vida del dispositivo, minimiza lo que estorba, y prescinde de una potencia que no necesitamos (pues la IA pesada irá en el servicio de voz externo).

El botón o pulsador realmente puede tener varios propósitos a nivel de hardware, ya que su funcionalidad se programa por separado, pero en nuestro caso, queremos un pulsador para poder hablarle al dispositivo sin decir la palabra de activación, es decir, saltándonos la parte de “Oye Sam”. Esto sirve en muchos dispositivos en los que no quieres que se active sin querer por distintas razones —como un entorno muy ruidoso que provoque falsos positivos o que obligue a tener que acercarse demasiado al dispositivo; o porque se quiere poder decir "*Oye Sam*" sin que se active el dispositivo.

Los pines de GPIO son entradas y salidas analógicas y digitales que sirven de expansión, permitiendo la conexión de dispositivos externos, como relés que permitan a SAM, por ejemplo, controlar las luces o activar una cafetera.

7.2.2 Arquitectura software de SAM Device

Veamos el flujo de una interacción entre un usuario y SAM Device:

- El dispositivo está escuchando en local, a la espera de la palabra de activación y con los leds apagados. Cuando el dispositivo escucha la palabra de activación considera que va a recibir una orden e inicia una petición HTTP al servicio de voz. Además, pone el patrón de leds de escuchar.

- Durante la petición HTTP el dispositivo está escuchando y envía solo el audio que considera que contiene habla. Cuando escuche un tiempo considerable de silencio tras haber oído algo de voz terminará la petición HTTP con lo que ha grabado.
- Mientras espera la respuesta del servicio de voz pondrá el patrón de leds de espera (o pensando). Cuando reciba la respuesta del servidor (en formato audio) reproducirá la respuesta por los altavoces. Mientras hable pondrá el patrón de leds de hablar.
- Cuando termine de reproducir apagará los leds y volverá a la escucha pasiva esperando la palabra de activación.

La **Figura 18** muestra esquemáticamente este flujo de trabajo.

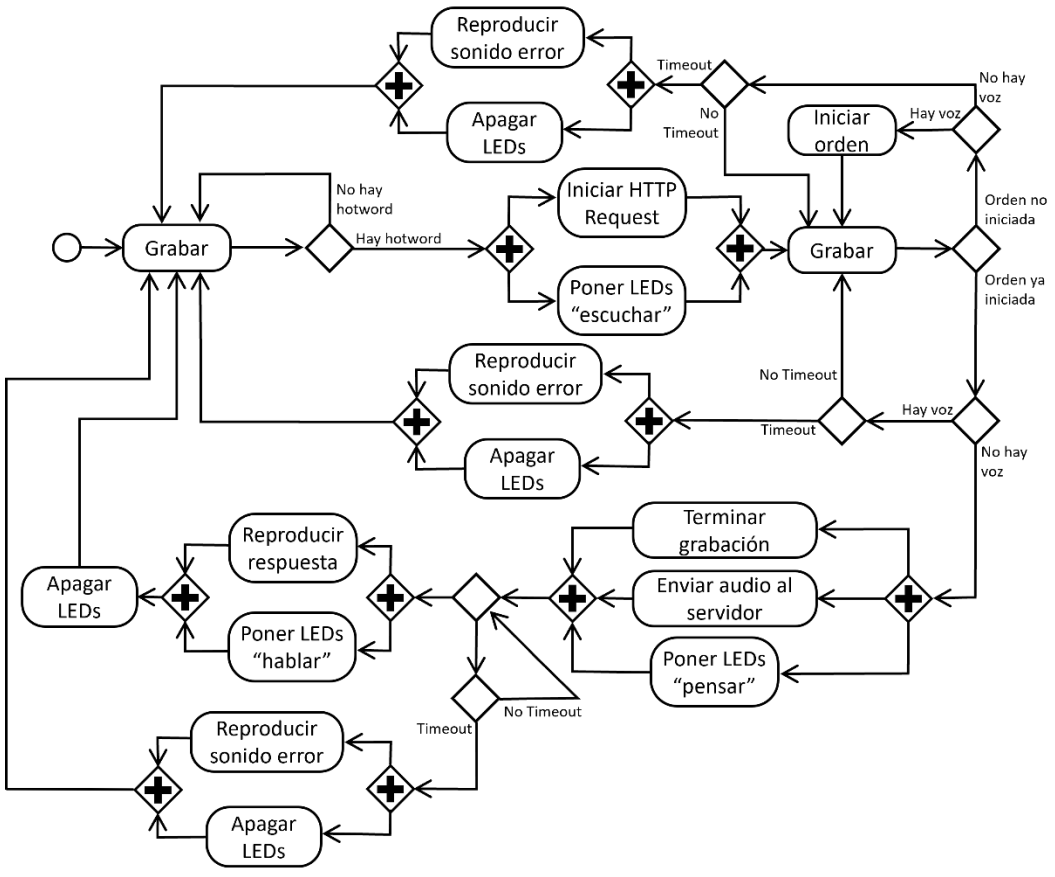


Figura 18. Diagrama de flujo de la interacción con SAM Device en formato BPMN.
FUENTE: elaboración propia.

7.3 API LEVEL

El *API Level*, como ya dijimos, contiene los servicios que actúan como *Application Programming Interface* (API) para el resto de los componentes de SAM, al centralizar todos los endpoints

definidos en el *backend* y pasando siempre por el Servicio de Autenticación (*Auth Service*) (ubicado, como su nombre indica, en el *Service Level* del proyecto global).

A efectos de nuestra parte, este nivel incluye la pasarela o *Gateway* del Servicio de voz por el que un **SAM Device** tendrá que pasar para llegar a dicho servicio. En el prototipo que vamos a desarrollar en este TFG, debido a que no vamos a interactuar con otros servicios, no se implementará, pero se menciona aquí pues sigue siendo una pieza importante para facilitar la integración con el proyecto global.

7.4 SERVICE LEVEL

En el *Service Level* viene el núcleo de la inteligencia artificial asociada al **SAM Device**. Aquí se ubica el servicio de voz que proponemos crear (**SAM Voice Service**), con los componentes identificados en el estado de la técnica (**capítulo 2**): ASR (conversión de voz a texto), TTS (conversión de texto a voz) y NLP (Procesamiento del Lenguaje Natural).

Este nivel proporciona la funcionalidad que un **SAM Device** necesita para comunicarse con las personas por vía de la voz y mediante lenguaje natural y así ofrecer la experiencia que buscamos. El servicio de voz, a su vez, podrá acceder al servicio de SAM Core ubicado también en este nivel, y desarrollado en el proyecto global (véase **Figura 22** para la arquitectura global del proyecto completo junto a este TFG).

7.4.1 SAM Voice Service

El servicio de voz (*Voice Service*), que se desarrollará en este TFG, es el encargado de hacer posible la interacción por voz con los usuarios. Su función es recibir peticiones del SAM Device en forma de audio, entenderlas, procesarlas y devolver una respuesta, de nuevo, en forma de audio.

Como se aprecia en la **Figura 19**, toda la comunicación entre el *Voice Agent* y los Servicios internos del *SAM Voice Service* se realizan mediante paso de mensajes a través de un *Event Broker* privado.

El flujo en el servicio de voz sigue un modelo orquestado, en el cual el agente de voz actúa como orquestador y llama a los subservicios que necesita en el orden que necesita. Preferimos este modelo frente al coreografiado, en el que los componentes se organizan entre ellos, porque

tener un componente orquestador central ayuda mucho a la independencia de los componentes, permitiendo quitar, añadir o reordenar los elementos fácilmente.

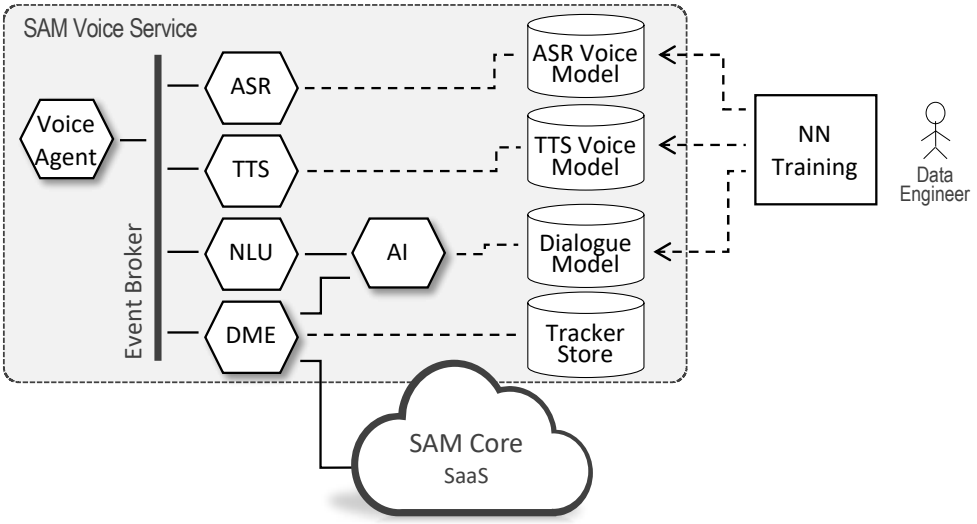


Figura 19. SAM Voice Service High-level Architecture.
FUENTE: elaboración propia.

En la **Figura 20** se muestra el diagrama de actividad de la orquestación que realiza el *Voice Service*.

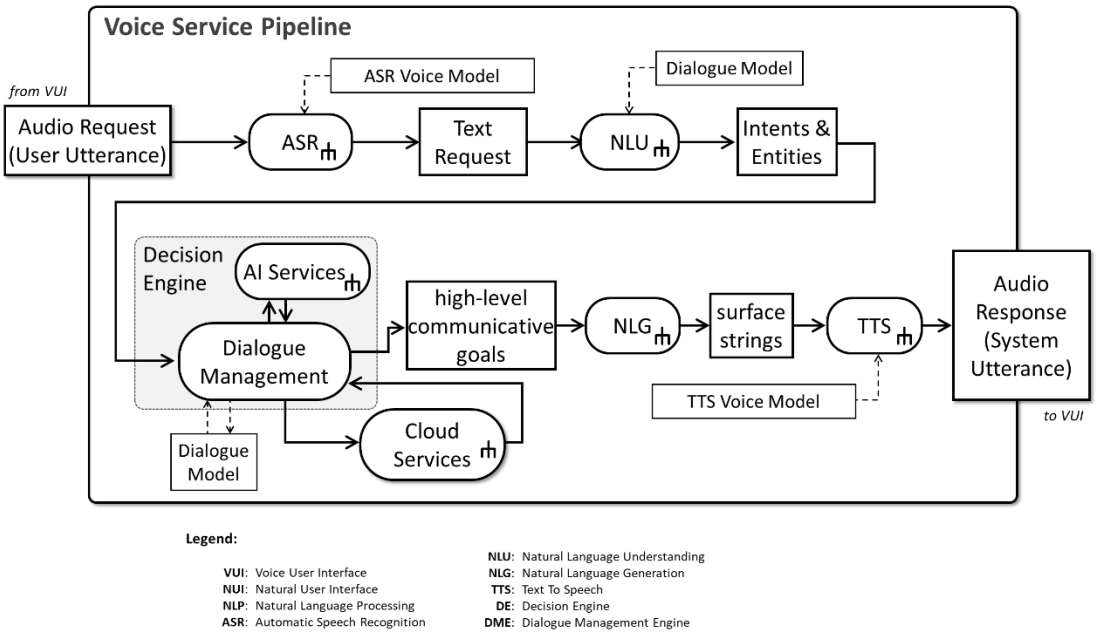


Figura 20. Diagrama de actividad de SAM Voice Service.
FUENTE: elaboración propia.

La actividad del servicio de voz se inicia cuando recibe una petición de un SAM Device en forma de audio (***Audio Request***). El primero paso es invocar al servicio de **ASR** (Automatic Speech Recognition) que, como ya sabemos, se encargará de escribir un texto en base a la voz que escuche en el audio recibido, para lo cual utilizará su ***ASR Voice Model*** almacenado en el Data Level que veremos en la siguiente sección. Con esto pasamos de tener la "Audio Request" a una "**Text Request**".

El siguiente paso es el servicio de **NLU** (*Natural Language Understanding*). Su función es entender el lenguaje natural, lo cual se traduce a extraer las intenciones (*intents*) del usuario y las entidades (*entities*) de la frase, que actuarán como parámetros de la intención. Con este paso obtenemos "**Intents & Entities**", ya listos para que la máquina pueda trabajar con ellos.

A continuación, entramos en el ***Decision Engine***, que contiene al ***Dialogue Management*** (gestión del diálogo), encargado de tomar las decisiones en el diálogo y preparar la respuesta a nivel conceptual. Como ya dijimos en el estado de la técnica, este bloque no forma parte del **NLP** (procesamiento del lenguaje natural), sino que es el punto intermedio entre NLU y NLG que usa cualquier otra técnica de IA para tomar decisiones varias. Por ello puede invocar a otros servicios de IA (**AI Services**) si lo necesita, o incluso a servicios más genéricos (***Cloud Services***) externos a él. De hecho, en este caso se usarán los servicios de ***SAM Core*** de manera externa cuando el sistema necesite acceder a información de los usuarios. Para todo esto, utiliza un ***Dialogue Model*** que permite clasificar la decisión a tomar en base a la intención recibida. Este bloque da como resultado una respuesta, pero todavía en el formato ordenado de las máquinas, en base a intenciones y conceptos sueltos sin lenguaje natural, pero de alto nivel: ***High-Level Communicative Goals***.

Esta respuesta conceptual entra al módulo de **NLG** (Natural Language Generation) el cual se encargará, ahora sí, de montar una respuesta en lenguaje natural a partir de los conceptos que la máquina quiere expresar, uniéndolos a las normas sintácticas y gramaticales del idioma para el que está entrenado. Esto nos da cadenas de texto legibles por un humano: "**Surface Strings**".

Por último, pero igualmente importante, llamaremos al servicio de **TTS** (Text- to-Speech) con las cadenas de texto obtenidas. El módulo de TTS generará un audio que *dicta* el texto escrito por la máquina, para lo cual se apoya en un ***TTS Voice Model*** previamente entrenado. Por fin, después de todos estos pasos, tenemos una respuesta en formato de audio lista para ser oída

por el usuario: "**Audio Response**". Esta respuesta se devolverá a la aplicación que inició la solicitud.

Cabe destacar, sin embargo, que al utilizar la plataforma **Rasa** hay una pequeña diferencia en el servicio de voz final con el aquí explicado. El rango de acción de **Rasa** abarca desde NLU, que recibe el texto del usuario, hasta NLG, justo después de obtener la respuesta en texto de la máquina. Sin embargo, **Rasa** no tiene NLG debido a que se entrena para responder frases predefinidas. Si bien estas frases pueden tener parámetros, partes variables, o incluso aleatorias, por dentro no ha habido NLG, si no que el ***Dialogue Management*** genera el texto como respuesta directamente. Por ello en nuestro esquema, realmente del ***Dialogue Management*** saldría una flecha directa a ***Surface Strings***. Es gracias a esto que **Rasa** acepta cualquier idioma, porque el modelo de conversación se escribe y entrena con frases en un idioma específico.

7.4.2 Modelos de voz y de conversación

Para el funcionamiento de los módulos ya descritos en la anterior sección, tal y como se ha explicado, se requieren modelos entrenados para cada uno de los componentes de inteligencia artificial: ***modelo de voz* para ASR**, ***modelo de voz* para TTS** y ***modelo de conversación*** para el asistente inteligente.

Los ***modelos de voz*** son para las tecnologías de voz, y van por separado para **ASR** y para **TTS**. Aunque tienen un fin muy relacionado, realmente son opuestos. Un ***modelo para ASR*** contiene toda la información necesaria para convertir audio (con voz) a texto en un idioma determinado, de manera que indica cómo reconocer ciertos fonemas y las letras que le pertenecen. Un ***modelo de TTS***, por otro lado, tiene toda la información necesaria para convertir texto a audio (con voz) en un idioma determinado, indicando cómo reconocer letras y conjuntos de letras, y qué fonemas (pronunciación) les pertenecen. Finalidades muy relacionadas, pero información opuesta.

El ***modelo de conversación*** define cómo se desarrolla una conversación con el asistente digital. Más concretamente, describe las posibles *intenciones* del usuario y las posibles respuestas del asistente (ya sea directamente en texto o a nivel conceptual), los posibles caminos que puede tomar una conversación y cómo seguirlos, y las acciones internas que debe seguir en según qué

situaciones, como acceder a bases de datos o servicios externos para recuperar información solicitada por el usuario.

7.5 DATA LEVEL

En el Data Level encontramos toda la persistencia de datos necesaria para el funcionamiento de SAM. En la arquitectura global (**Figura 10**) podíamos observar las distintas bases de datos de usuarios que servían para SAM Core. En cuanto al servicio de voz, sin embargo, no necesitamos datos extra de usuarios pues para todo lo que necesite saber **SAM Voice Service**, accederá a **SAM Core** para consultarlo.

Los datos que utiliza el servicio de voz son los modelos entrenados para los distintos algoritmos de inteligencia artificial que componen este trabajo: **Modelo de voz para ASR** (utilizado por para convertir el audio a texto en un idioma específico), **Modelo de voz para TTS** (utilizado para convertir texto a audio en un idioma específico), **Modelo de conversación** (utilizado para identificar intenciones y entidades en el texto, avanzar en la conversación y realizar las acciones necesarias en cada momento). De nuevo, véase la **Figura 16** para repasar la arquitectura de este trabajo.

Adicionalmente tenemos un **Tracker Store** que sirve para almacenar las conversaciones con los usuarios, para mejorar el comportamiento del asistente y realizar análisis forenses si se requieren.

El propósito de este nivel es solo almacenar estos modelos, no entrenarlos o tratarlos. Por ello dicho entrenamiento u obtención no se tratarán ahora, si no en la **sección 8.3.8** (*Modelo de conversación para Rasa*). Los modelos de voz se descargan de Internet (Modelo ASR, 2021) y (Modelo TTS, 2021), pues hacer el nuestro propio requeriría muestras de voz de miles de personas y no es abarcable en este TFG.

7.6 MAINTENANCE LEVEL

El *Maintenance Level* trata las tareas asociadas con el mantenimiento del sistema. A efectos del TFG cabe destacar el mantenimiento de los modelos de IA mencionados en la anterior sección (véase **7.5.** *Data Level*).

La razón por la que estos modelos necesitan mantenimiento es porque están en constante mejora. Principalmente, el *modelo de conversación* o de diálogo (**Dialogue Model**) cambiará para añadir nuevas posibles interacciones al asistente, y para mejorar o corregir interacciones antiguas. Es imposible predecir todo lo que pueden decir los usuarios y de todas las maneras en las que lo pueden expresar, y por ello siempre hará falta estar al tanto de maneras de solicitar las intenciones para añadirlas al modelo.

Los *modelos de voz*, tanto para entender la voz y convertirla a texto (ASR) como para leer un texto y leerlo con voz sintética (TTS) pueden mejorar infinitamente: en nuevas palabras, casos gramaticales extraños, acentos no reconocidos, etc.

7.7 DISEÑO DE LA CARCASA

Para cada prototipo del dispositivo IoT basado en la placa ReSpeaker Core V2 se ha diseñado un conjunto de carcasas o fundas que actúan de protector y de embellecedor. Para el modelado 3D de la carcasa se ha utilizado la aplicación *Blender*, junto con modelos 3D de base de la comunidad, con medidas para la *ReSpeaker Core V2* (Thingiverse ReSpeaker, 2021). La **Figura 21** muestra capturas de pantalla de la aplicación *Blender* durante la edición de estas piezas. La impresión 3D no se podrá llevar a cabo antes de la entrega de la memoria, pero el proceso se detalla en la **sección 8.1.3**.

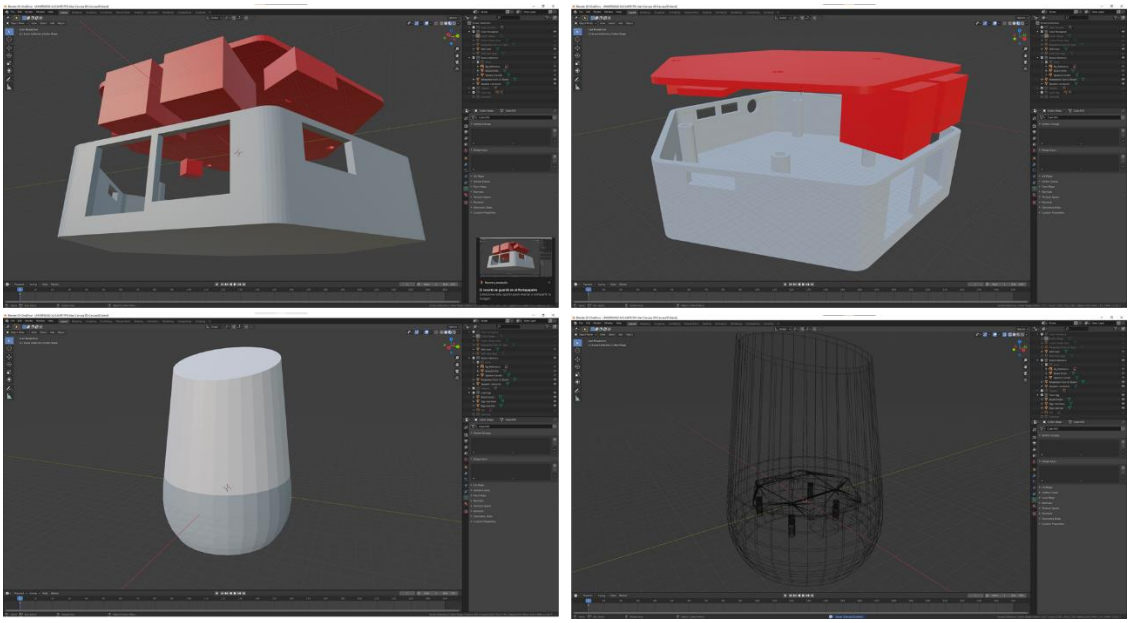


Figura 21. Capturas de *Blender* con algunos de los diseños 3D realizados.
FUENTE: elaboración propia.

7.8 INTEGRACIÓN CON LOS SERVICIOS DE SAM CORE

La integración con **SAM Core** no es compleja gracias a su arquitectura orientada a servicios. La manera en la que se conectan es que el servicio de voz, más concretamente las acciones personalizadas que hemos puesto en Rasa, utilizan los servicios de SAM Core cuando necesitan acceder a la información de su usuario: rutinas, medicación, etc.

Después de diseñar todos estos elementos, la **Figura 22** muestra la arquitectura de alto nivel de todo el proyecto, juntando la arquitectura mostrada inicialmente en **3.** *Antecedentes y resultados previos* (véase **Figura 10**) y el trabajo añadido en este Trabajo de Fin de Grado (véase **Figura 16**).

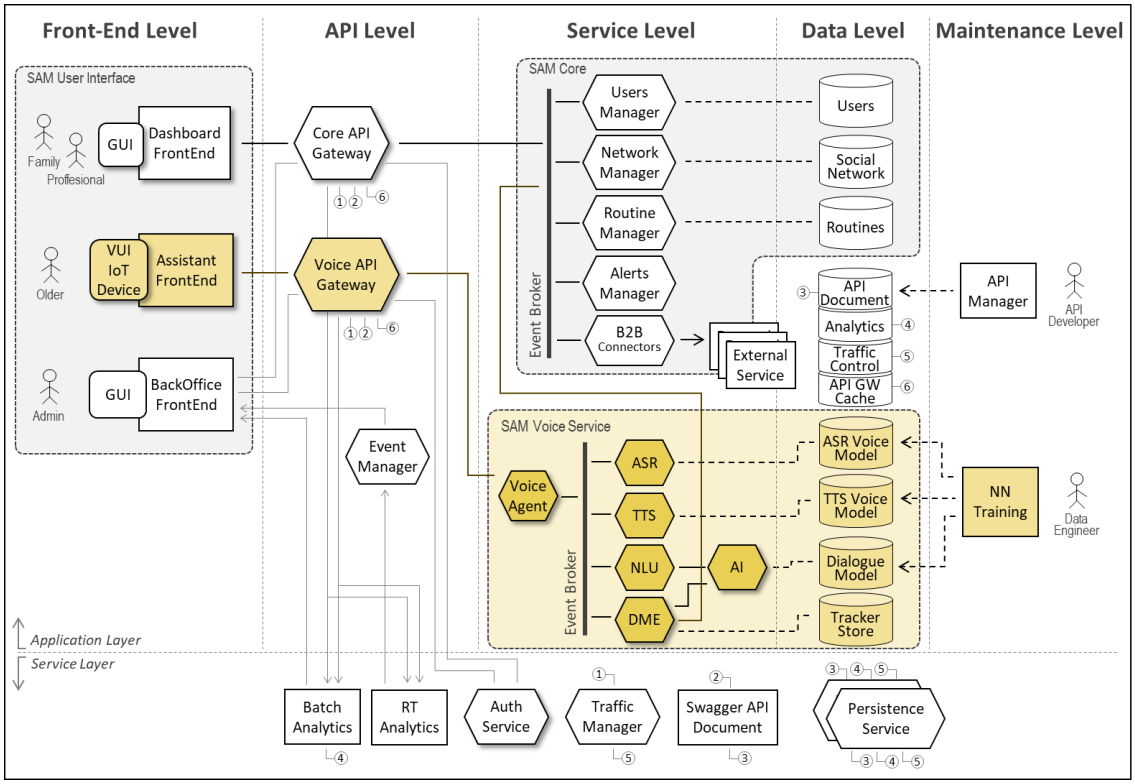


Figura 22. Arquitectura de integración de alto nivel de **SAM Project** con el dispositivo VUI (**SAM Device**) y los servicios de voz (**SAM Voice Service**) incorporados.
FUENTE: elaboración propia.

8 Implementación

En este capítulo nos centraremos en la fase de implementación de todo el sistema diseñado en el capítulo anterior, es decir, del dispositivo de acompañamiento (**SAM Device**) —hardware y software— y de la plataforma en la nube (**SAM Voice Service**) que proporciona los servicios de *backend* de voz y de procesamiento del lenguaje natural.

8.1 PROTOTIPOS HARDWARE DE DISPOSITIVO IOT

En primer lugar, vamos a construir los dispositivos físicos que darán soporte a nuestro asistente de voz. Por razones de investigación, construiremos tres dispositivos hardware distintos: uno basado en la placa ReSpeaker Core v2 (**apartado 8.1.1**); otro basado en una Raspberry Pi junto con una placa ReSpeaker 4-Mic Array; y un último prototipo basado en Raspberry y una placa ReSpeaker 2-Mic Array. Estos dos últimos, al estar basados en una Raspberry Pi y tener muchas similitudes en cuanto a su montaje, configuración y programación, se presentarán de forma conjunta (**apartado 8.1.2**).

Todos los prototipos siguen las mismas especificaciones y proporcionan los mismos componentes hardware que fueron diseñados en el apartado anterior y que están recogidos en la **Figura 17**. En los **apéndices 12.2**, **12.3** y **12.4**, se pueden ver todos los detalles técnicos y especificaciones de estos tres dispositivos hardware.

En este apartado nos centraremos en los aspectos más relevantes para que se entiendan los pasos seguidos y el código desarrollado. En la **Figura 23** se muestra un esquema con la arquitectura técnica conceptual (hardware y software) así como su mecanismo de interconexión con el servicio de voz (**SAM Voice Service**).

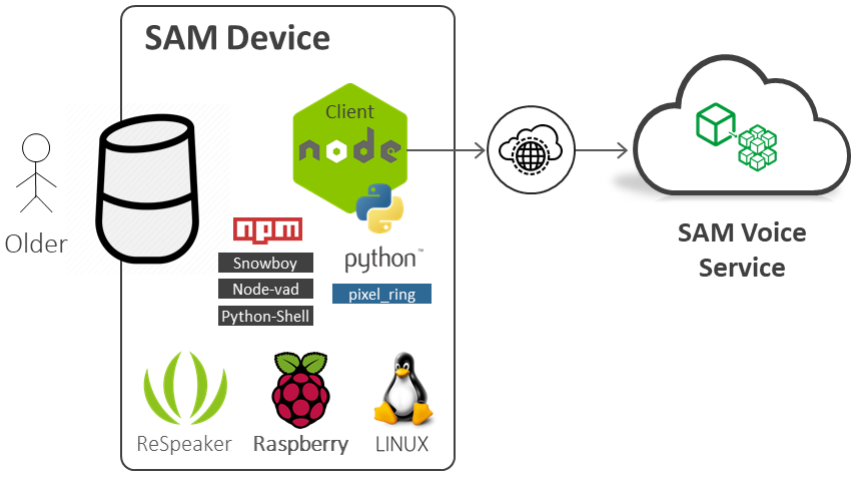


Figura 23. Arquitectura conceptual del dispositivo IoT (**SAM Device**).
FUENTE: elaboración propia.

8.1.1 Prototipo ReSpeaker

8.1.1.1 Material necesario

Para la configuración de ReSpeaker Core v2 tendremos que disponer del siguiente material:

- ReSpeaker Core v2.
- MicroSD de 4GB o más.
- Cable USB-MicroUSB de datos (no solo carga).
- Un ordenador con lector de SD y conexión Wi-Fi.

El primer paso es instalar la imagen del sistema operativo en el dispositivo.

8.1.1.2 Instalación de la imagen

Empezamos con la placa *ReSpeaker* apagada. Tendremos que instalar en nuestro ordenador una utilidad de creación de unidades *flash* favorita. En este caso se ha utilizado *BalenaEtcher* (Balena Etcher, 2021).

En este mismo equipo descargaremos la imagen del sistema operativo que vamos a instalar en la placa *ReSpeaker*. Para ello vamos a la página de descargas oficial de *ReSpeaker* (ReSpeaker OS, 2021). En esta página veremos varias opciones que se diferencian por el segmento de en medio. Unas tienen en el primer segmento **iot** o **lxqt**. En el segundo segmento pueden estar etiquetadas con **flasher** o **sd**.

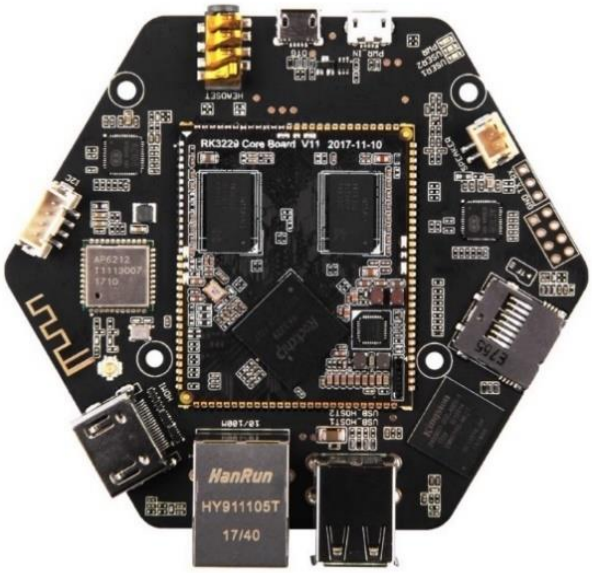


Figura 24. Dispositivo ReSpeaker Core v2.
FUENTE: https://wiki.seeedstudio.com/ReSpeaker_Core_v2.0/.

La diferencia entre la versión **iot** y la versión **lxqt** es que la segunda tiene una interfaz de escritorio mientras que la primera, pensada para IoT, no tiene. La versión **sd** instalará la imagen en la propia *MicroSD* (por lo que necesitará que esté insertada en todo momento). **Flasher**, en cambio, instala la imagen en la memoria no volátil de la propia placa.

Para desarrollo seleccionamos la versión **lxqt-sd** ya que la interfaz de escritorio nos facilitará trabajar con el dispositivo si lo necesitamos. Que la imagen esté en una *SD* nos permitirá cambiar de dispositivo si el original se corrompe en las pruebas y acceder a sus datos desde un ordenador potente, entre otras cosas.

Para la versión final de producción utilizaremos la versión **iot-flasher**, ya que el usuario final no necesita ni tiene que ver esa interfaz de escritorio, y no queremos que pueda extraer la tarjeta SD y acceder al código y a la configuración del dispositivo.

A continuación, conectamos la *microSD* al ordenador, abrimos *BalenaEtcher*, seleccionamos la unidad de la *microSD* y la imagen que acabamos de descargar y seleccionamos la opción de montar. En poco tiempo ya tendremos la *microSD* con la imagen grabada y podremos sacarla del ordenador.

Con la placa todavía apagada, introducimos la *microSD* a su lector, encendemos la *ReSpeaker* con el cable USB —normalmente usaríamos el puerto PWR_IN (ver **Figura 25**) para cargar y más

tarde el puerto OTG para comunicar la placa con el ordenador de desarrollo, pero el puerto OTG también sirve para alimentar a la placa, así que para la versión de desarrollo podemos conectar el cable en OTG. En la versión de producción solo haría falta un cable de carga y se conectaría al puerto PWR_IN. El cable *microUSB* que viene con el set de *ReSpeaker* es de datos.

Cuando se encienda la placa deberían iluminarse los leds de USER1 y USER2. Si todo va bien, USER1 parpadea durante el arranque y USER2 se enciende al leer de la SD. Estos leds están dispuestos siguiendo el borde de la placa en sentido horario desde el puerto PWR_IN y OTG en los que hemos enchufado el cable de carga.

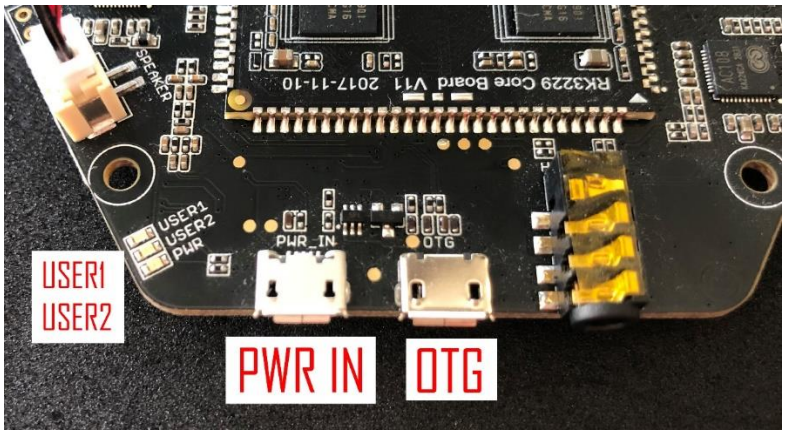


Figura 25. Puertos del ReSpeaker
FUENTE: elaboración propia.

La placa debería estar leyendo de la SD correctamente.

8.1.1.3 Acceso a consola desde el ordenador

El siguiente paso es acceder a la consola de la placa desde nuestro ordenador para poder trabajar con ella. Ahora sí es obligatorio que tengamos el cable *USB-MicroUSB* de datos conectado por un lado a nuestro ordenador y por otro lado al puerto OTG.

Comprobamos desde nuestro ordenador si el puerto serie está activo. Para ello, desde *Windows*, iremos al *Administrador de dispositivos* y buscaremos COMx (siendo x un número que cambiará según el puerto físico utilizado). Desde *Linux* teclearemos en consola:

```
$ ls /dev/ttyACM*
```

Esto encontrará todos los dispositivos que empiecen por *ttyACM*. Buscamos uno que sea *ttyACMx* donde, de nuevo, x es un número que cambia según el puerto, no nos importa cuál.

Si encontramos el puerto serie, todo está correcto. Procedemos a conectarnos a él con nuestra herramienta de depuración favorita. En este caso, y aconsejados por la guía oficial, se ha usado *Putty* (Putty, 2021) desde *Windows*. La configuración es:

- 115200 baud rate.
- 8Bits.
- Parity: None.
- Stop Bits: 1.
- Flow Control: None.

El nombre y contraseña por defecto son *respeaker* y *respeaker*, respectivamente. Ya hemos accedido a la consola del dispositivo, ahora ya podemos trabajar.

Para más detalles sobre la preparación de un dispositivo *ReSpeaker Core v2* véase la guía oficial (ReSpeaker Guia, 2021).

8.1.1.4 Conexión a Wi-Fi

Para *ReSpeaker* usamos la utilidad *nmtui* que viene ya instalada por defecto ejecutando el siguiente comando en la terminal de la placa a la accedimos en el paso anterior:

```
$ sudo nmtui
```

Aparecerá una interfaz en la que elegiremos **Activate a connection**. En el siguiente menú vemos una lista de las conexiones disponibles. Elegimos la que queramos e introducimos la contraseña. Cuando termine (y aparezca un asterisco al lado del nombre de la red) pulsamos escape dos veces para salir. Ya estamos conectados.

Si queremos conectarnos por cable lo único que necesitamos hacer en esta sección es enchufar el cable al puerto ethernet de la placa.

El servidor SSH empieza automáticamente, por lo que también podemos utilizar cualquier cliente SSH para conectarnos al dispositivo una vez tenga conexión.

8.1.2 Prototipos Raspberry + Seeed

Configuraremos ahora los prototipos basados en *RaspberryPi* junto con placas del fabricante *Seeed* (modelos *4-Mic Array* y *2-Mic Array*). Las instrucciones son prácticamente las mismas

tanto con el 4-Mic Array como con el 2-Mic Array, y se indicará explícitamente si hay que hacer algo distinto para cada modelo.

8.1.2.1 Material necesario

Para la configuración de Raspberry Pi con 4-Mic Array o con 2-Mic Array tendremos que disponer del siguiente material:

- Raspberry Pi (cualquier versión, nosotros tenemos la 4).
- ReSpeaker 4-Mic Array o ReSpeaker 2-Mic Array (solo una por Raspberry).
- MicroSD (No hay almacenamiento mínimo, excepto por lo que ocupe el SO y el software).
- Monitor con puerto HDMI y cable HDMI.
- Teclado USB.
- Un ordenador con lector de SD y conexión Wi-Fi.

8.1.2.2 Montaje de Seeed y Raspberry

El montaje de una *Raspberry Pi* con una placa *ReSpeaker 4-Mic* Array o una placa *ReSpeaker 2-Mic Array* no requiere un gran esfuerzo, pues las dos tarjetas *ReSpeaker* están diseñadas para encajar en los zócalos de conexión de Raspberry Pi. Lo único que hay que hacer es conectar los pins de la Raspberry Pi en los puertos para *pins* de las *ReSpeaker* tal y como se indica en la **Figura 26**.

8.1.2.3 Instalación de la imagen

Para ambos prototipos partimos de una Raspberry Pi limpia, únicamente con su sistema operativo instalado. Por lo tanto, desde un ordenador externo, accederemos a la página oficial de Raspberry Pi en (RaspberryPi OS, 2021) y descargaremos un sistema operativo. Lo ideal sería elegir la *versión Lite*, ya que no necesitamos entorno de escritorio para un dispositivo IoT, aunque podemos bajar la versión con escritorio (*with desktop*) si queremos comodidad para desarrollar.

Tal y como hicimos con la *ReSpeaker*, bajamos nuestra herramienta favorita para crear unidades flash, como *BalenaEtcher*. Después conectamos la *microSD* al ordenador, abrimos *BalenaEtcher*, seleccionamos la unidad de la *microSD* y la imagen que acabamos de descargar y seleccionamos la opción de montar. No tardará mucho en grabar la imagen en la *microSD*, y cuando acabe

podemos extraerla del ordenador y enchufarla a la *Raspberry* Pi. Después encendemos la *Raspberry Pi*.

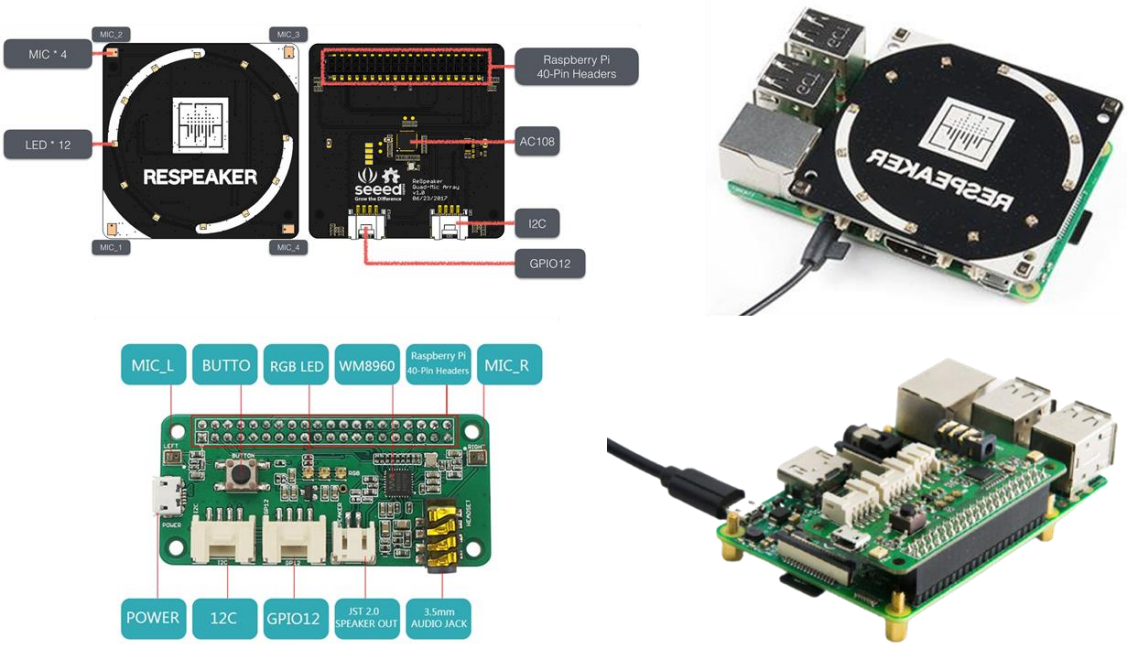


Figura 26. (arriba) Raspberry Pi y placa ReSpeaker 4-Mic Array de Seeed. (abajo) Prototipo con dos micrófonos, 4 leds, altavoz, sensor de temperatura y humedad, y relé.
FUENTE: (arriba) wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/.
(abajo) wiki.seeedstudio.com/Snips_Voice_Interaction_Base_Kit/.

8.1.2.4 Acceso a consola

Para la Raspberry Pi no utilizaremos un acceso externo por puerto serie como con la ReSpeaker. Conecta el cable HDMI a la Raspberry por un lado y a tu monitor por otro, y un teclado USB a la Raspberry. Cuando la enciendas verás la pantalla de la Raspberry Pi en tu monitor.

8.1.2.5 Conexión a Wi-Fi

Para conectar la Raspberry a internet vía *Wi-Fi* basta con ejecutar la herramienta de configuración con:

```
$ sudo raspi-config # Seleccionamos 2 (Network Options)
```

Y eligiendo *2. Network Options*. Ahí podremos seleccionar nuestra red y poner la contraseña. De nuevo, si queremos red cableada, basta con conectar el cable de Ethernet al puerto correspondiente en la Raspberry Pi.

8.1.2.6 Instalación de los drivers

Ahora que tenemos nuestra Raspberry Pi, vamos a instalar los drivers de *Seeed Voice*. Para ello lo haremos desde la consola de la Raspberry Pi.

```
$ sudo apt-get update
$ git clone https://github.com/respeaker/seeed-voicecard.git
$ cd seeed-voicecard
$ sudo ./install.sh
$ sudo reboot now
```

Tras este último comando, la placa se reiniciará. Ya debería estar todo listo, así que vamos a comprobarlo en la configuración de audio de la Raspberry ejecutando:

```
$ sudo raspi-config # Seleccionamos 1 (System Options) y luego S2 (Audio)
```

Y debería aparecernos un informe extenso, en el que aparece múltiples veces "*seeed-4mic-voicecard*". Si aparece, está perfecto.

8.1.3 Impresión de la carcasa

En la sección de diseño se mostraron varios diseños realizados como carcasa para cubrir el dispositivo IoT (SAM Device) y darle un aspecto más agradable junto con un acabado más profesional.

Debido a los tiempos, no podemos mostrar en esta memoria los resultados, puesto que los prototipos se lanzarán a imprimir durante los días próximos a la entrega y pueden tardar entre tres y cuatro horas cada pieza.

En la **Figura 27** se muestra una imagen de la impresora 3D empleada para la creación de las carcasas (*Monoprice Minidelta v2*). Debido a su reducido tamaño es posible que necesitemos separar las piezas en segmentos más pequeños o buscar una impresora 3D más grande.

En la **Figura 28** se muestra una captura de pantalla de la pieza básica siendo preparada para la impresión 3D utilizando el programa *Ultimaker Cura*. La finalidad de este programa es leer los modelos 3D creados en Blender (véase **sección 7.7**) y convertirlos a un formato que la impresora pueda leer (GCode). Este formato incluye el modelo 3D en láminas horizontales (que es lo que la impresora 3D entiende porque imprime capa por capa), además de los soportes extra, cómo realizar el relleno, el tamaño real exacto, etc.

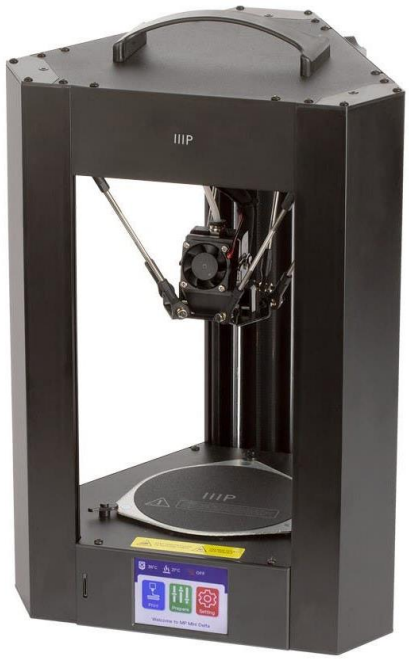


Figura 27. Impresora 3D *Monoprice Minidelta v2* utilizada para la creación de las carcasas de SAM Device.
FUENTE: https://www.monoprice.com/.

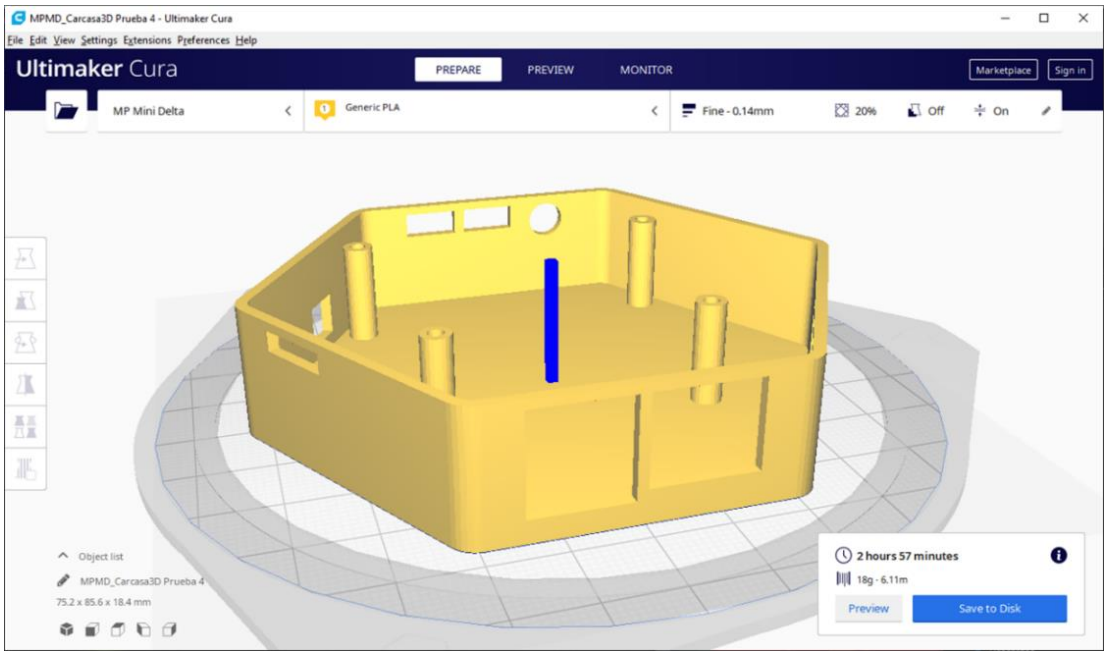


Figura 28. Captura de pantalla de la aplicación Ultimaker Cura empleada para gestionar la impresión 3D.
FUENTE: elaboración propia.

8.2 SOFTWARE DE DISPOSITIVO IOT

A continuación, instalaremos y configuraremos el software necesario y escribiremos el código conveniente para que estos dispositivos desempeñen la función de Asistentes de voz. Todo el código de esta sección fue subido a *GitHub* y puede verse en abierto y en detalle la versión final en el repositorio que se indica en el **apéndice 12.7.1**.

8.2.1 Configuración de la palabra de activación

El primer paso es conseguir que el dispositivo esté escuchando constantemente, esperando la palabra de activación. Es un paso clave ya que cuando se escuche dicha palabra tendremos que atender al usuario y ofrecerle una respuesta, pasando su orden por el servicio de voz. Este paso se hace en local ya que, si enviáramos lo que está grabando al servidor todo el tiempo para buscar la palabra de activación, la vida entera del usuario estaría pasando por Internet. Por ello la palabra de activación la detectaremos aquí, y cuando la detectemos enviaremos la orden y solo la orden al servicio de voz.

Este problema de privacidad es uno de los principales inconvenientes que tiene utilizar dispositivos comerciales o fabricados por terceros. Incluso *Alexa* de *Amazon* funcionaba de esta forma hasta no hace mucho tiempo.

Vamos a instalar **Snowboy** en el dispositivo. Como vimos en el estado de la técnica, *Snowboy* es una biblioteca de *código libre* para reconocimiento de palabras de activación en local. Abrimos la terminal de la placa con *Putty* (o la herramienta elegida). Importante, si no lo hemos hecho todavía, haber conectado la placa a Internet por cualquiera de los métodos ya explicados. En este caso usaremos la versión de *NodeJS* de *Snowboy*.

```
# Instalamos NodeJS y npm (Node Package Manager)
$ sudo apt update && sudo apt upgrade
$ curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
$ sudo apt-get install -y nodejs
```

Creamos un directorio e iniciamos un proyecto en *NodeJS*. Luego instalamos *Snowboy* para *NodeJS* usando su gestor de paquetes *npm*.

```
# Creamos una carpeta vacía y entramos
$ mkdir SnowboyNode
$ cd SnowboyNode
```

```
# Creamos un proyecto de NodeJS (Rellenando los datos del formulario)
$ npm init

# Creamos el punto de entrada (index.js)
$ touch index.js

# Instalamos Snowboy para NodeJS y otras bibliotecas necesarias
$ npm install snowboy@sonus
$ sudo apt-get installs ox libsox-fmt-all libasound2-dev
$ npm install node-record-lpcm16
```

Ahora tenemos que abrir el archivo *index.js* recién creado para escribir nuestro código. La idea es crear un detector con la biblioteca de *Snowboy* y conectarlo mediante un *pipe* (tubería) al micrófono, para que lo que grabe el micrófono se pase directamente por *stream* al detector y esté en constante funcionamiento.

```
const models = new Models();

models.add({
  file: 'models/snowboy.umdl',
  sensitivity: '0.5',
  hotwords : 'snowboy'
});

const detector = new Detector({
  resource: "resources/common.res",
  models: models,
  audioGain: 2.0,
  applyFrontend: true
});

const mic = record.record({
  threshold: micThreshold,
  verbose: true
}).stream().pipe(detector);
```

El detector funciona por eventos. Cada cuarta parte de segundo recibiremos una evaluación de si se ha encontrado la palabra de activación. En concreto, cada cuarta parte de segundo se activará uno de los tres eventos posibles: **Hotword** (palabra detectada), **Sound** (ruido detectado, sin la palabra), **Silence** (silencio máximo, no hay ni ruido). Los eventos los recibimos de la siguiente manera.

```
detector.on('hotword', function (index, hotword, buffer)
  { console.log('Hotword'); });

detector.on('sound', function (buffer)
  { console.log('Sound'); });
```

```
detector.on('silence', function ()
  { console.log('Silence'); });

detector.on('error', function ()
  { console.log('Error in Snowboy'); });
```

Si añadimos logs básicos en cada evento y ejecutamos con

```
$ node .
```

Veremos que en la terminal van saliendo mensajes de *Sound* o *Silence* cada poco tiempo. Si decimos "*Snowboy*" en voz alta saldrá una línea de *Hotword* en la terminal y seguirá con *Silence* o *Sound*.

Nótese que los eventos de *Hotword* y *Sound* proporcionan, además, un buffer. Este buffer contiene el audio grabado y evaluado en ese cuarto de segundo. Nos servirá más adelante para recibir la orden de voz del usuario.

8.2.2 Obtención de la orden de voz

Ahora que sabemos cuándo tenemos que escuchar al usuario, debemos empezar a grabar en el momento oportuno, parar cuando el usuario deje de hablar y enviar el audio al servicio de voz y que este sea procesado.

Primero implementamos un pequeño sistema de estados para saber cuándo grabar. Ahora mismo siempre vamos a estar recibiendo eventos *Sound* si el usuario está hablando, y en ellos reside el contenido que queremos enviar al servicio de voz, pero no siempre estamos recibiendo órdenes. El usuario podría estar hablándole a cualquier otra persona o cosa, nosotros solo queremos grabar si le habla directamente a nuestro dispositivo. Por ello, los eventos *Sound* deberían descartarse hasta que se identifique la palabra de activación, entonces, marcamos el evento *Hotword* para que *inicie* el proceso de grabación.

Igualmente queremos parar cuando el usuario se calle, por lo que el evento de *Silence* detendrá la grabación si ya estábamos grabando. Además, es normal que el usuario haga una pausa entre la palabra de activación y la orden, por lo que el evento *Silence* no detendrá la grabación iniciada si no ha habido un evento *Sound* entre la palabra de activación y el silencio. Por lo tanto, tenemos los siguientes estados:

- 0: Esperando palabra de activación
- 1: Palabra detectada, esperando inicio de orden (primer *Sound*)
- 2: Al menos un *Sound* detectado, grabando orden y esperando silencio

Así que el evento de *Hotword* siempre reiniciará la grabación al estado inicial (0), el evento de *Sound* solo valdrá si estamos en 1 o 2, y el evento de Silence solo detendrá la grabación si estamos en 2.

El código resumido queda como sigue:

```
function OnHotword(buffer) {
  if(currentState != 0) return;
  currentState = 1;

  console.log('Hotword detected!');
}

function OnSound(buffer) {
  if(currentState < 1) return; // Only 1 or 2
  currentState = 2;

  console.log('Sound');
}

function OnSilence() {
  if(currentState == 1) console.log('Silence but waiting order...');
  if(currentState != 2) return;

  console.log('Silence');
}
```

Y añadiendo la llamada a los eventos antes vistos

```
detector.on('hotword', function (index, hotword, buffer)
  { OnHotword(buffer); });

detector.on('sound', function (buffer)
  { OnSound(buffer); });

detector.on('silence', function ()
  { OnSilence(); });
```

Ejecutando de nuevo veremos que los eventos *Sound* y *Silence* no salen hasta que veamos *Hotword*, y el primer *Silence* que haya después de un *Sound* detendrá la salida por terminal a la espera de otra *Hotword*.

Pero hay un problema: aunque el usuario se calle, el dispositivo sigue detectando sonido y activando el estado *Sound*, de forma que nunca termina de grabar la *orden*. Esto ocurre porque

el evento *Silence* de *Snowboy* es estrictamente *Silence*, o con un umbral muy bajo. Cualquier sonido, incluso un ordenador en funcionamiento puede activar el evento *Sound* y evitar que se llegue nunca al silencio real. Así que nuestro problema es que los eventos *Silence* siempre son verdaderos *Silence*, pero los eventos *Sound* a veces son *Sound* y otras, en realidad, *Silence* (si contamos como *Silence* que el usuario no esté hablando al dispositivo).

Por suerte, *Sound* nos proporciona el buffer analizado como *Sound*, por lo que podemos filtrar los silencios nosotros mismos. Para solucionar el problema, utilizaremos una biblioteca de *VAD* (*Voice Activity Detection*). La instalamos con

```
$ npm install node-vad
```

Para su uso, en lugar de conectar el evento *Sound* de Snowboy con nuestra función de *OnSound*, escribiremos una función intermedia llamada *ClassifyBuffer* que analizará el buffer y llevará a *OnSound* o *OnSilence* según el contenido del buffer. Esta función queda como sigue:

```
function ClassifyBuffer(buffer) {
  vad.processAudio(buffer, sampleRate).then(res => {
    switch (res) {
        case VAD.Event.ERROR:
        case VAD.Event.SILENCE:
            OnSilence();
            break;
        case VAD.Event.NOISE:
        case VAD.Event.VOICE:
            OnSound(buffer);
            break;
    }
  }).catch();
}
```

Ahora, si ejecutamos de nuevo, la detección de silencio o voz debería ser mucho más precisa y la orden debería detenerse en cuanto nos callemos. La orden se detendrá si nos paramos un segundo a respirar, lo cual es un poco agobiante. Se ha añadido un pequeño control en el que se necesitan varios silencios seguidos para considerar la orden terminada. El código requerido es muy trivial así que véase el archivo *index.js* en el apéndice del software para SAM Device (**apéndice 12.7.1**) para más detalles.

Ya tenemos acotado un punto en el que iremos obteniendo buffers con el audio de la orden solo cuando queramos grabar y que se detendrá cuando toque. Ahora hay que enviar la orden al servidor.

8.2.3 Envío de la orden al servicio de voz

A continuación, montaremos la orden y la enviaremos al servidor para que sea procesada. Necesitaremos la biblioteca de NodeJS de HTTP:

```
$ npm install http
```

Vamos a crear funciones auxiliares para organizar el código, dedicadas a iniciar una petición HTTP a un servidor, a enviar datos a través de dicha petición, y a terminar la petición.

```
function StartHTTPRequest() {
  const options = {
    hostname: voiceServiceHost,
    port: voiceServicePort,
    path: '',
    method: 'POST',
    headers: {
      'Content-Type': 'application/octet-stream' // Generic binary data
    }
  };

  httpRequest = http.request(options, res => {
    console.log('La petición ha sido respondida');
  });

  httpRequest.on('error', error => { console.error(error); });
}

function WriteChunkToServer(buffer) {
  if(httpRequest) httpRequest.write(buffer);
}

function FinishHTTPRequest() {
  httpRequest.end();
  httpRequest = null;
}
```

Ahora las utilizamos en los lugares correspondientes: iniciar la petición HTTP en el evento *Hotword*, enviar datos (el buffer) en el evento *Sound* y terminar la petición cuando *Silence* decida que la orden ha terminado.

Nótese que la petición HTTP se ha iniciado en el momento en el que oímos la palabra de activación, y enviamos datos cada vez que se procesan como *Sound* ya que, aunque en todo momento hemos descrito esta parte como “Primero montamos la orden, luego la enviamos” en realidad la orden de voz se envía conforme se graba y así cuando el usuario termina de hablar la orden ya está en el servidor y se ahorra esperas innecesarias.

Aunque en este punto de la memoria el servicio de voz no existe todavía, en realidad ya había creado un prototipo cuando se programó esta parte y se pudo conectar sin más problemas.

8.2.4 Recibimiento de la respuesta del servidor

Ahora tenemos que recibir la respuesta desde el servicio de voz (que será un audio) y reproducirla. Primero instalaremos bibliotecas para este propósito.

```
$ npm install speaker
$ npm install wav
```

La respuesta del servidor se recibe a modo de *callback* definido al hacer la petición HTTP. Añadimos este código a la función *StartHTTPRequest* en lugar de la llamada anterior.

```
httpRequest = http.request(options, res => {
    let body = Buffer.alloc(0);
    res.on('data', chunk => {
       body = Buffer.concat([body, chunk], body.length + chunk.length);
    });
    res.on('end', () => {
       PlayOnSpeakers(body);
    });
  });
```

En esta función de *callback* estamos indicando que cuando lleguen datos los guarde al final del *body*, y que cuando la respuesta haya terminado de escribirse llame a una función auxiliar *PlayOnSpeakers* con los datos recopilados (que son la respuesta en formato audio). La función *PlayOnSpeakers* queda como sigue:

```
function PlayOnSpeakers(data) {
  wavReader.on('format', function(format) {
    wavReader.pipe(new Speaker(format));
  });
  let speakerStream = new Readable({ read() {} });
  speakerStream.push(data);
  speakerStream.push(null);

  speakerStream.pipe(wavReader);
}
```

Y con esto, la respuesta se reproducirá por los altavoces conectados a la placa en cuanto llegue del servidor.

8.2.5 Retroalimentación con leds

Con todo lo anterior el dispositivo IoT ya está funcionando, pero queda un detalle que mejorará enormemente la interacción con el usuario. En este momento, aunque el usuario diga la palabra de activación, no sabrá realmente si el dispositivo le ha oído correctamente y puede estar dando la orden en vano, dejando ese incómodo momento en el que no sabe si esperar para intentarlo de nuevo o dejar a la máquina funcionar y no cortarle.

Para evitar esta situación tenemos que añadir retroalimentación. Lo normal es añadir algún tipo de iluminación que indica que el dispositivo está escuchando, esperando, pensando, etc., según el patrón de iluminación.

Por suerte para nosotros la placa *ReSpeaker* incluye varios *leds* dispuestos en forma de anillo. Se pueden encender y apagar manualmente utilizando las salidas digitales conectadas a los pines de la placa. También podemos utilizar una biblioteca oficial de *ReSpeaker* llamada *pixel_ring* que proporciona un controlador para los *leds* y varios patrones ya predefinidos, lo cual facilitará enormemente la labor. Primero instalamos dependencias:

```
$ sudo apt install python-mraa libmraa1
$ pip install setuptools_scm
$ npm install python-shell # Para usar Python desde NodeJS
```

Y luego instalamos la biblioteca de manera normal:

```
$ pip install pixel_ring
```

O bien clonamos el repositorio (Pixel Ring, 2021) y la instalamos desde ahí. Con esta instalación podemos modificar la biblioteca si queremos cambiar cómo funcionan las cosas o personalizar algún patrón:

```
$ git clone –depth 1 https://github.com/respeaker/pixel_ring.git
$ cd pixel ring
$ pip install -U -e .
```

Esta biblioteca se utiliza desde *Python*. Una de las dependencias que hemos instalado se llama *Python-shell*, y nos permitirá llamar a un *script* de *Python* desde *NodeJS*. Creamos un script llamado ***led_controller.py*** que simplemente recibe cadenas por la entrada estándar (*stdin*) que es la manera de comunicar *NodeJS* con *Python-shell*, y cambiará el patrón que están mostrando los leds según la cadena que llegue. El archivo *led_controller.py* se puede ver en el **apéndice 12.7.1**.

Ahora lanzamos el script desde nuestro archivo *index.js* usando *Python-shell* con:

```
var ledCnt_ScriptPath = 'led_controller.py';
const {PythonShell} = require('python-shell');
var ledCnt_pyOptions = {
  mode: 'text',
  pythonPath: '/usr/bin/python'
};
var ledController = new PythonShell(ledCnt_ScriptPath, ledCnt_pyOptions);
```

Y a continuación usamos la línea:

```
ledController.send('listen');
```

En los lugares adecuados para indicar el patrón que debe usarse en cada momento. Ahora si ejecutamos nuevamente nuestra aplicación, observaremos que los *leds* se encienden cuando escuchan la palabra de activación, cambian la manera en la que están encendidos cuando el usuario comienza a hablar, cuando está *pensando* (o esperando respuesta del servidor a la orden de voz) y mientras el dispositivo está *contestando* (reproduciendo el audio obtenido del servidor).

8.2.6 Personalización la palabra de activación

Ahora el dispositivo IoT está terminado y hace todo lo que queremos que haga a nivel funcional, pero queda un elemento de personalización que es crucial en el factor humano para que empaticen con nuestro dispositivo y lo vean más amigable: no tiene un nombre.

Hasta ahora hemos utilizado "Snowboy" como palabra de activación ya que esta viene por defecto, pero nuestro dispositivo se llama "SAM", y lo ideal es que los usuarios puedan hablarle diciendo "Sam" o "Oye Sam" o frases por el estilo.

La plataforma de *Snowboy* cerró el 31 de diciembre de 2020, y con ella cerró la API que permitía entrenar y generar nuevas palabras de activación. Sin embargo, dejaron un script que ayuda a generar nuevos modelos personales. La diferencia principal entre los modelos personales y los universales (como el que hemos usado hasta ahora) es que los personales solo sirven para la persona que da las muestras de voz (u otras personas que se le parezcan en voz) pero a cambio requieren solo tres muestras de voz. Los modelos universales, aunque sería ideal tener uno, requieren un mínimo de 1500 muestras de voz (al menos tres muestras de quinientas personas distintas) y ya no se pueden generar con la plataforma cerrada. La solución para futuras versiones pasa por esperar a que la comunidad de Snowboy, que sigue activa, recupere la

manera de crear estos modelos universales, o de tener que cambiar de motor de detección de palabra de activación.

Como este proceso requiere de instalaciones que no necesitaremos en el dispositivo, vamos a otra máquina Linux distinta y así no consumimos innecesariamente los recursos del dispositivo IoT. Primero instalamos las bibliotecas básicas necesarias:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install git python python-pip
$ sudo apt-get install portaudio19-dev
```

Después descargamos el repositorio que contiene los scripts de generación de modelos personales:

```
$ git clone https://github.com/seasalt-ai/snowboy.git
```

Y procedemos a instalar las dependencias del script en un entorno virtual de *Python* (para evitar posibles conflictos de versiones de bibliotecas con otros proyectos):

```
$ virtualenv -p python2 venv
$ source venv/bin/activate
$ cd snowboy/examples/Python
$ pip install -r requirements.txt
```

Luego grabamos tres muestras de audio diciendo, en este caso, "Oye Sam". Los audios tienen que estar en ***wav***, con una frecuencia de muestreo de **16000 Hz**, **mono-canal** y **16 bits** de profundidad. Cuando las tengamos las guardamos en *examples/Python* y ejecutamos

```
$ python generate_pmdl.py -r1=s1.wav -r2=s2.wav -r3=s3.wav -lang=es -n=oyesam.pmdl
```

Donde *s1.wav*, *s2.wav* y *s3.wav* son las tres muestras que hemos grabado. Ahora deberíamos tener nuestro archivo **oye*sam.pmdl*** en el directorio en el que nos encontramos. Este es el archivo que tendremos que cambiar por ***snowboy.umdl*** para que el dispositivo espere la frase "Oye Sam" en lugar de "Snowboy".

El dispositivo IoT ya está listo para funcionar y asistir a nuestros mayores. A partir de ahora nos centraremos en la creación del servicio de voz que nos dará soporte *backend* en esta tarea de reconocimiento.

8.3 SERVICIOS DE VOZ

El servicio de voz (**SAM Voice Service**) constituye la parte *backend* de este proyecto y se ubica en los niveles de servicio y de datos (*Service Level* y *Data Level*. Véase **Figura 16**). Su finalidad es proporcionar soporte al asistente de voz realizando tareas de inteligencia artificial (IA) que no encajan en el dispositivo IoT ya sea por su elevado coste computacional o por la lógica de negocio que implican.

8.3.1 Arquitectura técnica

En la **Figura 29** se presenta la arquitectura técnica conceptual del servicio de voz, donde se puede ver de forma esquemática los servicios que intervendrán, en qué tecnología se basará su implementación y cómo se relacionan entre sí y con el resto de los elementos externos.

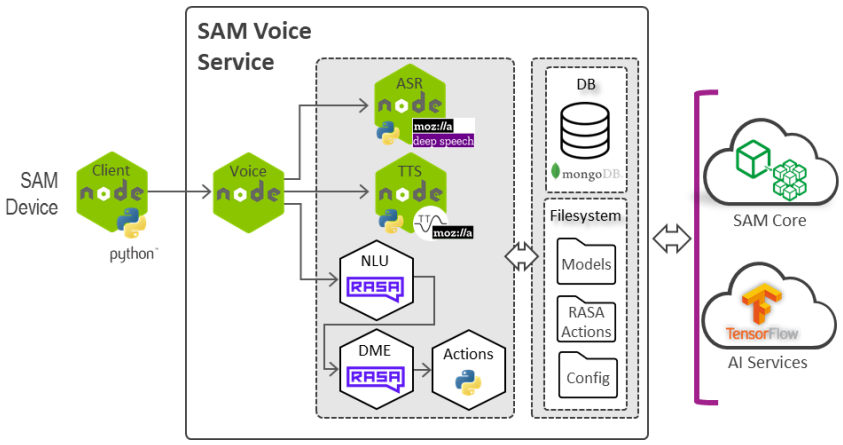


Figura 29. Arquitectura técnica conceptual del servicio de voz (**SAM Voice Service**).
FUENTE: elaboración propia.

En concreto, el servicio de voz se encarga de tres tareas fundamentales:

- Convertir el audio del usuario en texto para poder pasarlos por un asistente inteligente de texto. Este componente se llama Automatic Speech Recognition (ASR) o Speech to Text (STT).
- Después tenemos un asistente inteligente de texto encargado de recibir lo que ha dicho el usuario y avanzar en la conversación. Este componente abarca *Natural Language Understanding* (NLU) y *Natural Language Generation* (NLG). Este paso dará como salida otro texto, que es la respuesta del asistente inteligente después de leer al usuario,

realizar las acciones correspondientes y pensar una respuesta en lenguaje natural o humano.

- Por último, actúa un componente llamado *Text to Speech* (TTS) que convierte este texto respuesta a formato audio para que el dispositivo pueda reproducirlo por sus altavoces.

Además de estas tareas, el servicio de voz necesitará un orquestador que las gestione todas como si de una sola se tratase, sirviendo de intermediario con los clientes. Llamaremos a este servicio *Voice Agent*.

Recordemos que uno de nuestros objetivos es realizar el máximo trabajo posible con software de *código abierto*. Esto influirá en la decisión de qué bibliotecas utilizar en cada segmento.

Para el componente de *ASR* utilizaremos *Mozilla DeepSpeech*. *Mozilla DeepSpeech* es una biblioteca de reconocimiento de voz de *código abierto*. La idea es crear un servidor que actúe de subservicio para el servicio de voz, al cual se le pasará un audio y devolverá un texto, siendo el audio la orden del usuario y el texto devuelto la orden convertida a texto.

Para el componente de NLP utilizaremos *Rasa*. *Rasa* es una plataforma de *código abierto* para conversaciones mediante texto, que proporciona las herramientas para crear interacciones totalmente automatizadas a partir de la extracción de *intenciones* y *entidades* del texto del usuario, además de ser capaz de trabajar con contexto. *Rasa* se despliega como un servidor, con lo cual también actuará como subservicio para nuestro *servicio de voz*. Para que el asistente inteligente actúe como nosotros queremos será necesario trabajar en un *modelo de conversación* personalizado que responda a las intenciones de nuestros usuarios y tome las acciones adecuadas. En el **apartado 8.3.8** se explica cómo hemos realizado dicho modelo.

Para el componente de TTS utilizaremos *Mozilla TTS*. *Mozilla TTS* es una biblioteca de texto a voz de *código abierto*. En este caso recibirá como entrada el texto que nos ha respondido el asistente inteligente y devolverá un audio que es la respuesta en formato voz. Este será el último paso del servicio de voz y cuando termine se le enviará la respuesta al usuario en su dispositivo IoT para que sea reproducida.

De nuevo, todo el código que se verá en las **secciones 8.3.3** a **8.3.7** se ha subido a un repositorio de *GitHub*, que se indica en el **apéndice 12.7.2**.

8.3.2 Arquitectura de despliegue

Antes de ponernos a programar, veremos cómo van a estar distribuidos los componentes del servicio de voz en el despliegue final. La **Figura 30** muestra la arquitectura de despliegue básica, construida a partir de todo lo explicado hasta ahora, que ofrece la funcionalidad mínima que necesita el servicio de voz para estar completo.

En este esquema de despliegue los elementos se han encapsulado en un contendor *Docker* que permite desplegarlos fácilmente y además de manera automática (véase la **sección 8.3.9** para más detalles), proporcionando el entorno que necesitan y el propio programa en un paquete autocontenido.

8.3.3 Instalaciones previas para el servicio de voz

Para poner en marcha el servicio de voz necesitaremos realizar primero unas instalaciones generales. Para empezar, necesitamos Python en el sistema:

```
# Instalamos Python
$ sudo apt update
$ sudo apt install python3-dev python3-pip

# Actualizamos pip (el gestor de paquetes de Python)
$ pip3 install -U pip
```

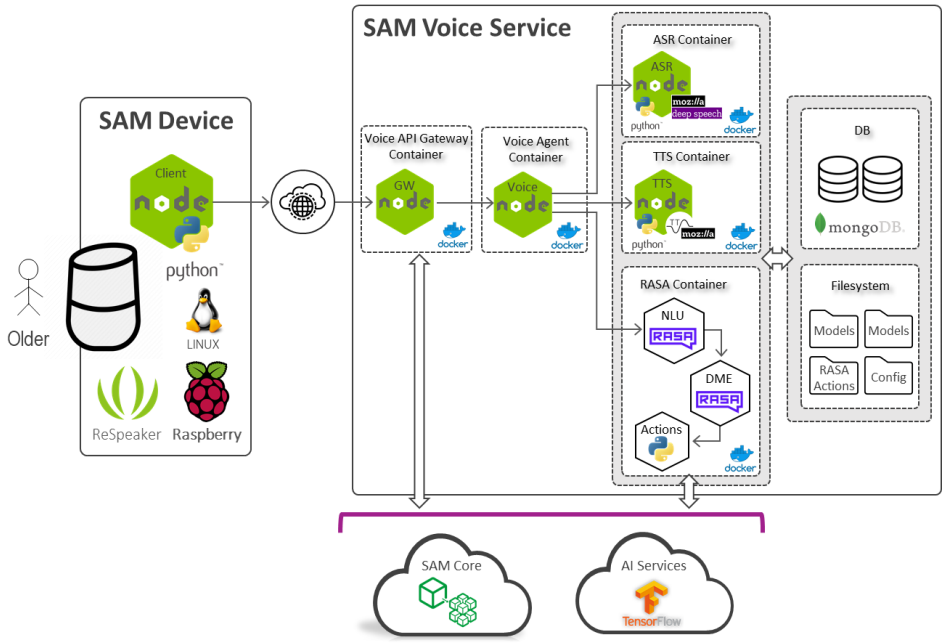


Figura 30. Arquitectura de despliegue básica junto con las tecnologías involucradas.
FUENTE: elaboración propia.

Para evitar instalar los paquetes para todo el sistema y ahorrarnos posibles conflictos de dependencias usaremos un entorno virtual aislado.

```
# Creamos el entorno virtual
$ python3 -m venv ./venv
# Entramos en el entorno virtual
$ source ./venv/bin/activate
```

Y por último vamos a instalar una serie de bibliotecas de audio requeridas para utilizar Mozilla *DeepSpeech* y *Mozilla TTS*:

```
# Instalamos las bibliotecas de audio necesarias
$ sudo apt-get install libasound-dev portaudio19-dev
$ sudo apt-get install libportaudio2 libportaudiocpp0
$ sudo apt-get install ffmpeg libav-tools
$ pip3 install pyaudio
```

Por último, necesitamos *NodeJS*, además creamos un proyecto de *NodeJS* para trabajar en él. También instalamos bibliotecas útiles que usaremos.

```
# Instalamos NodeJS y npm (Node Package Manager)
$ sudo apt update && sudo apt upgrade
$ curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
$ sudo apt-get install -y nodejs

# Creamos un proyecto de NodeJS (Rellenando los datos del formulario)
$ npm init
$ touch index.js

# Instalamos Python-shell para usar Python desde NodeJS
$ npm install python-shell
```

Ahora que tenemos la base preparada podemos entrar en servicios específicos.

8.3.4 Servicio de ASR

Como hemos explicado ya, el servicio de ASR (*Automatic Speech Recognition*) se encargará de, dado un audio de entrada del usuario, obtener lo que ha dicho en formato texto para así poder pasarlo por el servicio de NLP.

Para utilizar *Mozilla Deepspeech* primero tendremos que instalar el paquete de *Deepspeech* para *Python*:

```
# Instalamos la biblioteca de Deepspeech
$ pip3 install deepspeech
```

Para que funcione, necesitamos un modelo entrenado. Podemos descargar un modelo en español de la siguiente carpeta pública de *Google Drive* (Modelo ASR, 2021):

```
# Descargamos el modelo de Mozilla DeepSpeech en español
$ mkdir stt_models
$ cd stt_models
$ wget --load-cookies /tmp/cookies.txt
"https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies
/tmp/cookies.txt --keep-session-cookies --no-check-certificate
'https://docs.google.com/uc?export=download&id=1q9HhT8IYx5sFEcADjtLLLZ_PsrrsyYQE' -O-
| sed -rn 's/.*confirm=([0-9A-Za-
z_]+).*/\1\n/p')&id=1q9HhT8IYx5sFEcADjtLLLZ_PsrrsyYQE" -O modelo_stt.pbmm && rm -rf
/tmp/cookies.txt
$ cd ..
```

Para utilizar la biblioteca de *Deepspeech* en *Python* hay dos pasos básicos. El código que viene a continuación está recopilado en el apéndice, pero aquí mostraremos lo más importante. El archivo que vamos a editar ahora es *launch_stt.py* (En el repositorio se encuentra en *MozillaSTTLib/launch_stt.py*). Primero tenemos que cargar el modelo pasándole una serie de parámetros básicos vía variables globales:

```
def load_deepspeech_model():
    N_FEATURES = 25
    N_CONTEXT = 9
    BEAM_WIDTH = 500
    LM_ALPHA = 0.75
    LM_BETA = 1.85

    return Model('stt_models/modelo_stt.pbmm')
```

Esta función nos devuelve una instancia de **Model** de *Deepspeech* con el modelo que hemos descargado antes. Para traducir el audio a texto leemos el archivo de audio y usamos la función **stt** del **Model**.

Para nuestro propósito queremos un *script* que reciba audio y devuelva el texto. Este *script* será llamado desde *NodeJS*, y partimos de que nos enviará audio por la entrada binaria estándar y le devolveremos un texto (sacándolo por pantalla, *NodeJS* lo capturará). Por lo tanto, teniendo la función definida antes, escribimos el código principal *main* como el siguiente:

```
if __name__ == '__main__':
    audio_input = sys.stdin.buffer.read()
    model = load_deepspeech_model()
    predicted_text = model.stt(numpy.frombuffer(buffer=audio_input,dtype=numpy.int16))
    print(predicted_text) # Text output captured by NodeJS
```

En este *main* recibimos el contenido binario de audio a leer por parámetro y utilizamos la función ya definida antes que carga el modelo, para después obtener el texto correspondiente en *predicted_text*. Después lo escribimos en la salida estándar para que *NodeJS* lo reciba.

Desde *NodeJS* vamos a preparar un servicio muy básico con la biblioteca de *HTTP* de *NodeJS* y el *PythonShell* para ejecutar scripts de *Python*. Para ello, crearemos un nuevo archivo de texto (llamado *indexSTT.js*) cuya versión final está disponible en el **apéndice 12.7.2**, y escribiremos lo siguiente:

```
const http = require('http');

var myPythonScriptPath = 'MozillaSTTLib/launch_stt.py';
const {PythonShell} = require('python-shell');
```

Ahora tenemos incluidas las bibliotecas de *HTTP* y *PythonShell*. Creamos pues un servidor y definimos su respuesta a peticiones:

```
http.createServer(function(req, res) {
```

Y dentro de esta respuesta recuperamos el mensaje binario (el audio). Pero estos datos pueden estar fraccionados y llegar en momentos separados ya que la petición es asíncrona, con lo cual el código de llamar al *script* debe recogerlo cada vez que llegue un fragmento con el evento *on data*:

```
    let body = Buffer.alloc(0);
    req.on('data', chunk => {
        body = Buffer.concat([body, chunk], body.length + chunk.length)
    });
```

Después, cuando acabe, ejecutamos el *script* de *Python*.

```
    req.on('end', () => {
        var pysh = new PythonShell(myPythonScriptPath, { mode:'binary' });
        pysh.send(body);
```

Esto ejecutará de manera asíncrona el *script* antes descrito. Le indicamos que para cada mensaje recibido (línea impresa en consola desde *Python*), lo guardamos en una cadena resultado.

```
        let result = '';
        pysh.stdout.on('data', function(message) {
            result += message.toString();
        });
```

Por último, le decimos que cuando termine de ejecutar el script, el servidor HTTP escriba la respuesta al cliente y termine la llamada. La respuesta al cliente (que en este caso es el servicio de voz) será el texto obtenido de procesar el audio capturado.

```
        pysh.end(function(err) {
            if(err) { throw err; };

            res.write(result);
            res.end();
            console.log('Finished!');
        });
    }); // Cerramos req.on('end')
```

Y por último iniciamos el servicio de ASR indicándole el puerto en el que debe escuchar. En este caso el puerto se ha asignado al **8081**.

```
}).listen(port);
```

Ahora lanzamos el servicio desde consola con la siguiente instrucción:

```
$ node indexSTT.js
```

El servicio de ASR ya está listo para funcionar. Para invocarlo más tarde le enviaremos el audio del usuario y nos responderá con la transcripción a texto.

8.3.5 Servicio de TTS

Para utilizar *Mozilla TTS* tenemos que descargarnos o clonar su repositorio desde *GitHub* e ir a la rama adecuada:

```
# Clonamos el repositorio y vamos a la rama deseada
$ git clone https://github.com/mozilla/TTS.git
$ cd TTS
$ git checkout db7f3d3
```

Ahora instalamos las bibliotecas con el script que viene con el repositorio:

```
# Instalamos paquetes y dependencias varias
$ python3 setup.py develop
$ pip install -e .
$ sudo apt-get install espeak
```

Para que *Mozilla TTS* pueda funcionar debemos indicarle la ubicación del repositorio clonado a través de una variable de entorno. Para ello usamos

```
# Ejemplo para "/home/tfg/project-folder/TTS":
$ export PYTHONPATH="/home/tfg/project-folder"
```

Donde la ruta exportada debe apuntar a la carpeta padre de la carpeta *TTS* obtenida al clonar el repositorio. Para no tener que repetir este paso siempre que encendamos el ordenador podemos añadir la línea al archivo de perfil para que se ejecute automáticamente al iniciar. Esto es sencillo editando el archivo con:

```
# Abrimos el archivo de perfil con el editor de texto
$ nano ~/.profile
# Y copiamos la línea de export PYTHONPATH=”/home…” al final del todo
```

Igual que en *Mozilla Deepspeech*, necesitamos un modelo ya entrenado en español, como el siguiente (Modelo TTS, 2021):

```
# Descargamos el modelo en español para Mozilla TTS
$ mkdir tts_models
$ cd tts_models
$ wget --load-cookies /tmp/cookies.txt
"https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies
/tmp/cookies.txt --keep-session-cookies --no-check-certificate
'https://docs.google.com/uc?export=download&id=1WXSmsl_eO4BGtJfah-nY8ARKDcsEwpW9' -O-
| sed -rn 's/.*confirm=([0-9A-Za-z_]+).*/\1\n/p')&id=1WXSmsl_eO4BGtJfah-
nY8ARKDcsEwpW9" -O best_model.pth.tar && rm -rf /tmp/cookies.txt
$ wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=1s7g4n-
B73ChCB48AQ88_DV_8oyLth8r0' -O config.json
$ cd ..
```

El *script* de *Python* aquí está dividido en varios pasos. Creamos un nuevo archivo de texto llamado *launch_tts.py* (En el repositorio se encuentra en *MozillaTTSLib/TTS/launch_tts.py*). De nuevo tenemos que cargar el modelo antes descargado. Para ello:

```
    num_chars = len(phonemes) if CONFIG.use_phonemes else len(symbols)
    model = Tacotron(num_chars, CONFIG.embedding_size, CONFIG.audio['num_freq'], CONF
IG.audio['num_mels'], CONFIG.r, attn_windowing=False)
```

Donde *CONFIG* es una variable en la que hemos guardado la configuración leía del archivo de configuración por defecto de la biblioteca. Este archivo contiene parámetros sobre cómo procesar la onda generada.

Ahora hay que generar el audio. La función ***synthesis*** de *Mozilla TTS* nos da, entre otras cosas, la onda de audio sintetizado dado un texto y el modelo:

```
waveform, alignment, spectrogram, mel_spectrogram, stop_tokens
       = synthesis(model, text, CONFIG, use_cuda, ap)
```

Ahora escribimos la onda en un archivo de audio:

```
ap.save_wav(waveform, OUT_FILE)
```

El archivo completo se llama *launch_tts.js* y está en el **apéndice 12.7.2**. Tal y como hicimos con el servicio de *ASR*, vamos a crear un servicio con *NodeJS* que reciba peticiones de textos a convertir y sintetice audio usando el script de *Python* ya mostrado. La estructura será muy parecida al código de *NodeJS* usado para el servicio de ASR. De nuevo, para ver el código completo, véase el **apéndice 12.7.2**.

El archivo que vamos a crear y editar ahora se llama *indexTTS.js*. El cambio principal es que el *PythonShell* lo lanzamos en modo texto y con el texto a traducir como parámetro:

```
        var pyOptions = {
            mode: 'text',
            args: [reqMessage]
        };
```

Además, este servicio se lanza en el puerto **8083**.

Por último, lanzamos el servicio con

```
$ node indexTTS.js
```

El servicio de *TTS* ya está listo.

8.3.6 Servicio de NLP

Instalamos *Rasa* con el gestor de paquetes de *Python*

```
# Instalamos Rasa
$ pip3 install rasa
```

Configuramos un proyecto con la configuración por defecto y el modelo de conversación de ejemplo que viene con la descarga.

```
# Creamos un proyecto con datos de ejemplo
$ rasa init

# Entrenamos a Rasa con el modelo predefinido, por ahora
$ rasa train # --domain path_a_los_datos_de_domain.yml
```

Lanzamos el servicio de rasa

```
# Lanzamos el servidor de Rasa
$ rasa run --enable-api -p 8082
```

Rasa ya está lista para funcionar (a falta de crear nuestro propio *modelo de conversación*, lo cual se abordará en el **apartado 8.3.8**).

8.3.7 Servicio de voz: uniéndolo todo

Para nuestro *servicio de voz* (**Voice Agent**) necesitamos un último servicio de *NodeJS* que actúe de interfaz con el usuario y de *orquestador* para llamar a los distintos servicios que hemos implementado. Para ello escribimos el siguiente código en un nuevo archivo llamado *index.js*:

```
const callSTT = require('./helpers/callSTT.js').callSTT;
const callTTS = require('./helpers/callTTS.js').callTTS;
const callNLU = require('./helpers/callNLU.js').callNLU;

http.createServer(function(req, res)
{
  let body = Buffer.alloc(0);
  req.on('data', buffer => {
    body = Buffer.concat([body, buffer], body.length + buffer.length)
  });
  req.on('end', () => {
    console.log('Request fully received: ' + body.length);

    callSTT(body).then((msgstt) =>
    {
      callNLU(msgstt).then((msgnlu) =>
      {
        let parsed = JSON.parse(msgnlu);
        msgnlu = parsed[0].text;
        callTTS(msgnlu).then((msgtts) =>
        {
          const readStream = fs.createReadStream(outputFilepath);
          readStream.pipe(res);
        }).catch((msgerr) => {console.log(msgerr)});
      }).catch((msgerr) => {console.log(msgerr)});
    }).catch((msgerr) => {console.log(msgerr)});
  });
}).listen(port);
```

Que se apoya en los archivos incluidos en las tres primeras líneas para modular el código de la llamada a los tres servicios. Estos tres archivos solo tienen código que hace llamadas *HTTP* sencillas a los servidores que hemos montado en esta sección (*ASR*, *NLP* y *TTS*) en forma de promesa, y cuando se resuelve una llama a la siguiente. El código de cada uno de ellos, así como el de este archivo (*index.js*), se puede revisar también en el **apéndice 12.7.2**.

Para terminar, lanzamos todos los servicios (incluyendo los ya explicados, por si no lo están todavía):

```
# Lanzamos los sub-servicios
$ node indexSTT.js
$ node indexTTS.js
$ rasa run --enable-api -p 8082

# Lanzamos el servicio principal
$ node index.js
```

El *servicio de voz* (**Voice Agent**) ya está listo para atender las peticiones de **SAM Device** (del *dispositivo IoT*).

8.3.8 Personalización del modelo de conversación

El *modelo de conversación* es un elemento clave de este TFG y resulta primordial para el éxito de la interacción entre los usuarios y el asistente de voz. Para diseñar correctamente estos modelos se requieren uno o más expertos en la materia (en este caso, expertos en atención a mayores) que puedan plantear conversaciones cercanas a la realidad de los usuarios. También se requiere mucha experiencia de diálogos con los usuarios. Esta experiencia se obtiene poniendo a usuarios reales a conversar con el asistente, de forma que pueda *aprender* de estas conversaciones que ni siquiera los expertos han podido predecir.

En el caso de este trabajo de fin de grado no se ha podido lograr toda la atención de expertos que se desearía para desarrollar este modelo de conversación, sobre todo debido a la situación actual de pandemia, pero el proyecto global sí ha estado supervisado por los profesionales de los servicios sociales del ayuntamiento de Alcoy. De las reuniones con dichos expertos nos hemos inspirado para encaminar el proyecto, y han permitido definir una base sólida para las situaciones en las que nuestro asistente de voz puede ser útil e identificar qué caminos puede tomar una conversación habitual según la intención del usuario.

Por supuesto, por muchas veces que nos reunamos con los expertos, el modelo de conversación tiene que salir al campo y ser probado en situaciones reales para poder ser perfeccionado. Ahora que la situación epidemiológica está mejorando, se están retomando nuevamente las tareas pendientes de pruebas con los usuarios reales.

Un modelo de conversación en *Rasa* es relativamente sencillo de definir. Aunque no posee una interfaz sencilla para hacerlo como la de, por ejemplo, Amazon Alexa (ver **Figura 31**), tampoco hace falta tener un conocimiento amplio de inteligencia artificial para crearlos.

Los modelos de *Rasa* se crean en archivos de texto en formato *YAML* (YAML, 2021). En *Rasa Playground* (Rasa Playground, 2020) se explica paso a paso la sintaxis, qué significa cada apartado, y además permite entrenar el modelo *online* y probarlo en un chat de texto sin tener que instalar *Rasa*.

También permite descargar el modelo entrenado para la prueba, con lo que ni siquiera es obligatorio entrenar el modelo en un equipo propio.

El modelo de conversación de *Rasa*, de manera similar a otros asistentes de texto, incluye:

- Definición de dos tipos de elementos básicos:
    - Frases que puede decir el usuario humano, agrupadas en intenciones (*intents*).
    - Frases que puede responder el asistente (*utterances*), sin especificar a qué responden.

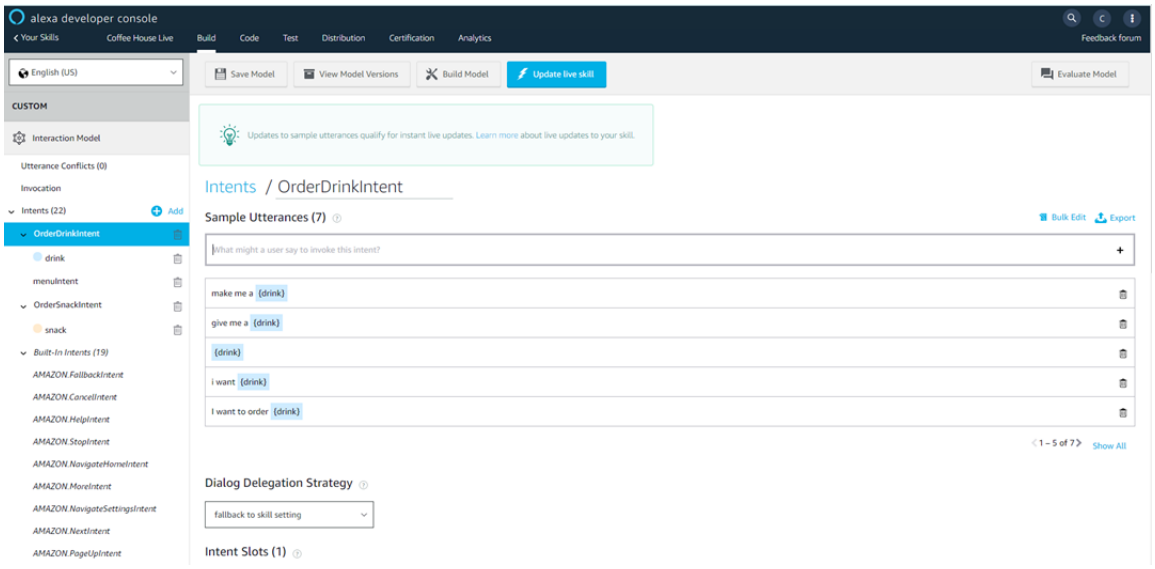


Figura 31. Interfaz para la creación de un intent en una Skill de Alexa.
FUENTE: developer.amazon.com/en-US/docs/alexa/devconsole/about-the-developer-console.html.

- Definición de historias (*stories*) que cuentan un camino predeterminado. Consisten en una secuencia de *intents*, *utterances*, acciones y otros elementos que ahora veremos. Cuando el usuario interactúe con *Rasa*, esta entenderá sus *intents*. Cuando sigan una

secuencia lógica definida en estas historias, *Rasa* sabrá con qué *utterance* responder pues estará siguiendo el guion establecido por la historia. Los algoritmos de IA de Rasa entran en acción para elegir la historia que más se acerque a lo que está conversando con el usuario.

- Las reglas o normas (*rules*). Son parecidas a las historias, pero estas están pensadas para situaciones cortas y específicas. En lugar de situaciones varias en las que la IA elegirá la más parecida, son reacciones a situaciones específicas: por ejemplo, se puede definir una norma para que siempre que el usuario responda con el *intent* de saludar (que pueden ser varias frases, como "Hola", "¿Qué tal?", etc.) Rasa responda con una *utterance* de responder al saludo. De esta manera, si el usuario hace esto en mitad de la conversación, saltará la norma, Rasa responderá, y retomará el contexto de la conversación que estaban teniendo.
- Los formularios (*forms*). Son un conjunto de campos a rellenar con entidades de las frases del usuario. Se pueden activar como acción en mitad de una historia o norma, de manera que la siguiente frase del usuario se utilice para tratar de rellenar estos campos y utilizarlos en la siguiente acción.
- El dominio, que es un archivo *YAML* que recopila todos los elementos arriba descritos a modo de cabecera, sin entrar en detalle con ninguno, pero resumiendo el modelo.

Vamos a realizar una primera versión del modelo de conversación teniendo en cuenta lo que hemos visto. Las intenciones del usuario que se han identificado, así como la conversación que podría seguir después, son las siguientes:

- Saber detalles de una cita médica (fecha, ubicación, doctor, propósito de la cita, …).

- *SAM ¿Tengo médico hoy?*
  o *Sí, tienes cita con el doctor [inserte médico] en el centro de salud a las cinco y cuarto de la tarde ¿Quieres saber algo más de esta cita?*
- *¿Dónde has dicho?*
  o *En el centro de salud, consulta 35 ¿Algo más?*
- *Nada más.*

- *SAM ¿Cuándo es mi siguiente visita al endocrino?*   o *Tu siguiente visita al endocrino es el jueves 24 de abril a las nueve y media de la mañana ¿Quieres saber algo más de esta cita?* - *No gracias.*

- Si tiene que tomarse alguna pastilla (color, dónde están, nombre, …).

  - *SAM ¿Qué pastilla me tengo que tomar a las 5?*
    - *La pastilla de las cinco es Frenadol. Si no la encuentras puedes pedirme que llame a [inserte familiar] y preguntarle dónde están.*
  - *Vale, llama a [inserte familiar].*
    - **Llamada**
  - *SAM, cuelga la llamada.*

- Entretenimiento (chiste, historia, noticias, música).

  - *SAM, pon algo de música.*
    - **Pone música* (Finaliza la interacción).*

- Contactar con personas (familiares, médicos, amigos…) u obtener sus números / correos.

  - *SAM, llama a mi médico.*
    - *Llamando al doctor [médico] *Llamada* (no lo coge).*
    - *Parece que el doctor [médico] no está disponible, puedo llamar a [familiar].*
  - *Sí, por favor.*

- Anotar recordatorios (para sí mismos o para familiares, …).

  - *SAM, recuérdame que me tome la pastilla a las cinco.*
    - *Recordatorio anotado a las cinco para "me tome la pastilla"*

- Informar de algún problema (le duele la cabeza, se siente mal, …).

    - *Recomendar solución (evidentemente SAM no debería recetarles nada).*

Atendiendo a estos ejemplos podemos escribir las frases, tanto del usuario como del asistente, y luego redactar historias que sigan cada interacción. Después creamos formularios y normas para poder personalizar cada intención y atender a la petición. Por ejemplo, en la siguiente figura (**Figura 32**) se pueden apreciar los datos asociados a cuando un usuario quiere consultar su siguiente cita médica.

Nota: para ver los archivos *YAML* completos véase el **apéndice 12.7.2** (en la ruta "*Rasa/data*"). Para entrenar a *Rasa* con estos datos tenemos que colocarlos en el proyecto de *Rasa* creado y ejecutar:

```
$ rasa train --domain data/domain.yml --data data/
```

Por defecto, el archivo *domain.yml* se busca en el directorio en el que se ejecuta el comando, pero aquí estaba en la carpeta *data* así que se ha tenido que especificar. El resto de los archivos se buscan por defecto en una carpeta llamada *data*, pero se ha añadido al comando para dejarlo explícito.

Si se observa con atención la **Figura 32**, se puede apreciar que hemos escrito frases predeterminadas para ciertas consultas, como el médico que es siempre el martes. Esto es porque todavía no le hemos incorporado lógica de negocio. Queremos que cuando el usuario pregunte por su médico, *Rasa* active acciones especiales para, en este caso, acceder a los servicios de **SAM Core** y consultar dicha cita.

Siguiendo con la conversación de la anterior captura (**Figura 32**), para que SAM pueda acceder a los datos de su siguiente cita médica e indicar al usuario cuándo es, igualmente tendrá que acceder a **SAM Core**. Para ello tenemos que definir funciones en *Python* que hagan lo que queremos e indicarle a Rasa que cuando en la conversación llegue a este punto, ejecute esa función.

Estas funciones se sirven desde otro servidor denominado *Servidor de acciones de Rasa*. Escribimos la función necesaria en *Python*, la cual define internamente su nombre, el que usa Rasa para identificarla, y le indicamos a *Rasa* que llame a dicha acción en el modelo de conversación, volviendo a los archivos *YAML*, en el *dominio*. Este *servidor de acciones* se lanza con la siguiente instrucción:

```
$ rasa run actions
```

```yaml
nlu:
- intent: greet
  examples: |
    - Hola
    - Saludos

- intent: ask_next_appointment
  examples: |
    - ¿Cuál es mi siguiente cita con el médico?
```

```yaml
responses:
  utter_greet:
      - text: |
          Hola! ¿Puedo ayudarte en algo?
      - text: |
          Saludos humano
  utter_next_appointment:
      - text: |
          Tu siguiente cita médica es con el doctor Blabla en el centro de salud a
  utter_ask_specialist_appointment:
      - text: |
          ¿Con qué médico quieres consultar tu siguiente cita?
  utter_specialist_appointment:
      - text: |
          Tu siguiente cita con {specialist} es el martes a las nueve de la mañana
```

```yaml
rules:
 - rule: ask appointment specialist
   steps:
   - intent: ask_next_appointment
   - action: utter_ask_specialist_appointment
   - action: appointment_specialist_form
   - active_loop: appointment_specialist_form

 - rule: submit appointment specialist form
   condition:
   - active_loop: appointment_specialist_form
   steps:
   - action: appointment_specialist_form
   - active_loop: null
   - action: utter_specialist_appointment
```

```yaml
forms:
  appointment_specialist_form:
    required_slots:
      specialist:
      - type: from_text
```

Figura 32. Fragmentos del modelo personalizado para Rasa en formato YAML.
FUENTE: elaboración propia.

Y *Rasa* utilizará la función cuando se invoque por el progreso de la conversación, llamándola con todo un contexto de parámetros. La **Figura 33** muestra un ejemplo de acción personalizada que escribe "Hello World".

```
from typing import Any, Text, Dict, List

from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher


class ActionHelloWorld(Action):

    def name(self) -> Text:
        return "action_hello_world"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        dispatcher.utter_message(text="Hello World!")

        return []
```

Figura 33. Ejemplo de acción personalizada para *Rasa*.
FUENTE: ejemplos de *Rasa*.

8.3.9 Despliegue en contenedores

El servicio de voz está listo para funcionar, pero desplegarlo es costoso. Por suerte esta tarea se puede automatizar. Una tecnología que permite la automatización del despliegue son los llamados *contenedores*, como por ejemplo *Docker* (Docker Install, 2021).

La finalidad de los *contenedores* consiste en albergar de forma empaquetada una aplicación o proceso, junto a sus datos, configuración y todo el entorno que necesita para funcionar (sistema operativo, bibliotecas, etc.). De esta manera, ejecutando un contenedor se puede desplegar una aplicación sobre cualquier sistema operativo que lo soporte sin preocuparnos por la instalación ni por las dependencias. Los *contenedores* siguen una filosofía similar al de las *máquinas virtuales*, solo que, en lugar de emular un equipo completo, con su sistema operativo y su entorno de usuario, se emula únicamente el núcleo del sistema operativo y los elementos y servicios mínimos y necesarios para la ejecución de la aplicación o servicio.

Las características de los contenedores facilitan el despliegue automático por parte de un equipo que actúe de controlador, como es el caso de los clústeres, permitiéndoles lanzar o detener instancias de la aplicación si se requiere.

Instalar *Docker* en *Ubuntu* requiere que configuremos su repositorio (Docker Install, 2021) primero con:

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -
o /usr/share/keyrings/docker-archive-keyring.gpg
$ echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-
keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Desde aquí es tan sencillo como ejecutar:

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

A continuación, configuraremos Docker para nuestros proyectos de *NodeJS* (Docker NodeJS, 2021). Creamos un archivo llamado *Dockerfile* con el siguiente contenido, cambiando el puerto (*EXPOSE*) y el punto de entrada (*index.js*) donde sea necesario, ya que por ejemplo el servicio de *ASR* usa *indexSTT.js.*

```
FROM node:10
EXPOSE 8080
CMD ["node","index.js"]
```

Nótese que en `node:10` estamos especificando la versión de *NodeJS* a utilizar. Después ejecutamos el siguiente comando para construir el contenedor:

```
$ docker build -f path/to/Dockerfile -t voice-service .
```

Cambiando en "`path/to/Dockerfile`" la ruta hasta el *Dockerfile* creado, a no ser que se llame *Dockerfile* y esté en el mismo directorio, en cuyo caso será el que utilice por defecto. Para ejecutar el contenedor ejecutamos la instrucción:

```
$ docker run -it --rm --name voice-service-running voice-service
```

Repetimos el proceso para los otros dos servicios de *NodeJS*, el de *ASR* y el de *TTS*, cambiando el puerto, el punto de entrada y el nombre del *Dockerfile* donde corresponda:

```
# DockerfileASR
FROM node:10
EXPOSE 8081
CMD ["node","indexSTT.js"]

# DockerfileTTS
FROM node:10
EXPOSE 8083
```

```
CMD ["node","indexTTS.js"]

$ docker build -f DockerfileASR -t voice-service-asr .
$ docker build -f DockerfileTTS -t voice-service-tts .
```

Solo nos queda añadir Rasa a un contenedor de Docker (Docker Rasa, 2021). En este caso se nos han adelantado y Rasa ya permite ser lanzada desde un Docker. Solo hay que ejecutar:

```
$ docker run -v $(pwd):/rasaApp rasa/rasa:2.8.4-full train --domain data/domain.yml
--data data --out models
```

En este último comando, "*rasa:2.8.4-full*" indica la versión de Rasa a utilizar. *Domain* y *Data* indican las rutas a los archivos de entrenamiento ya vistos en la anterior sección. Adicionalmente, recordemos que el servidor de acciones de Rasa va por separado. Para comunicar el servidor de Rasa recién lanzado en Docker y el servidor de acciones que estamos a punto de lanzar, creamos una conexión a través de Docker con:

```
$ docker network create rasa-actions-network
```

Y ahora sí lanzamos el servidor de acciones con:

```
$ docker run -d -v $(pwd)/actions:/rasaApp/actions --net rasa-actions-network --
name rasa-action-server rasa/rasa-sdk:2.8.4
```

Gracias a esto, podremos ejecutar los servicios fácilmente la próxima vez que iniciemos la máquina, y además hemos automatizado el proceso permitiendo una futura integración con clústeres (ver **sección 8.3.10**).

8.3.10 Despliegue de alta disponibilidad

En la **sección 8.3.2** vimos la arquitectura de despliegue básica. Esta arquitectura incluía los elementos necesarios para el funcionamiento esperado del servicio de voz, y excepto por el Gateway está todo implementado.

Sin embargo, aunque no se ha podido desplegar debido a la falta de medios, se ha diseñado una arquitectura de despliegue de alta disponibilidad que utiliza clústeres y balanceadores de carga para permitir un rendimiento y consumo de recursos escalables. La **Figura 34** recoge esta arquitectura.

El servicio debe ser viable y sostenible tanto si hay pocos usuarios como si tiene éxito y debe proporcionar soporte a miles de ellos. Igualmente debe ser suficientemente elástico como para

adaptarse a situaciones variables en las que en determinados momentos haya muchos más o menos usuarios. De esta forma, y bajo esta arquitectura, el servicio se puede desplegar desde unos pocos servidores (del orden de dos, como mucho tres equipos) en el caso de tener que dar soporte a unos pocos cientos de usuarios (como es el caso de la fase de pruebas), hasta despliegue en diferentes nubes públicas y privadas, basadas en *clústeres* escalables de servidores para cada microservicio y redes de almacenamiento para los sistemas de archivo y bases de datos. Además, la escalabilidad se puede aplicar tanto a todo el sistema como a cualquiera de sus componentes por separado, actuando de esta forma únicamente en aquellos puntos en los que se detecten deficiencias en el servicio o que, por el contrario, se encuentren sobredimensionados.

La finalidad de estos *clústeres* consiste en hacer expandible los recursos dedicados al servicio de voz, al permitir a la máquina lanzar más instancias de los servicios si se necesitan, lo cual es sencillo gracias también a los *contenedores*. El *balanceador de carga* envía las peticiones a las instancias menos concurridas para no sobrecargar una sola.

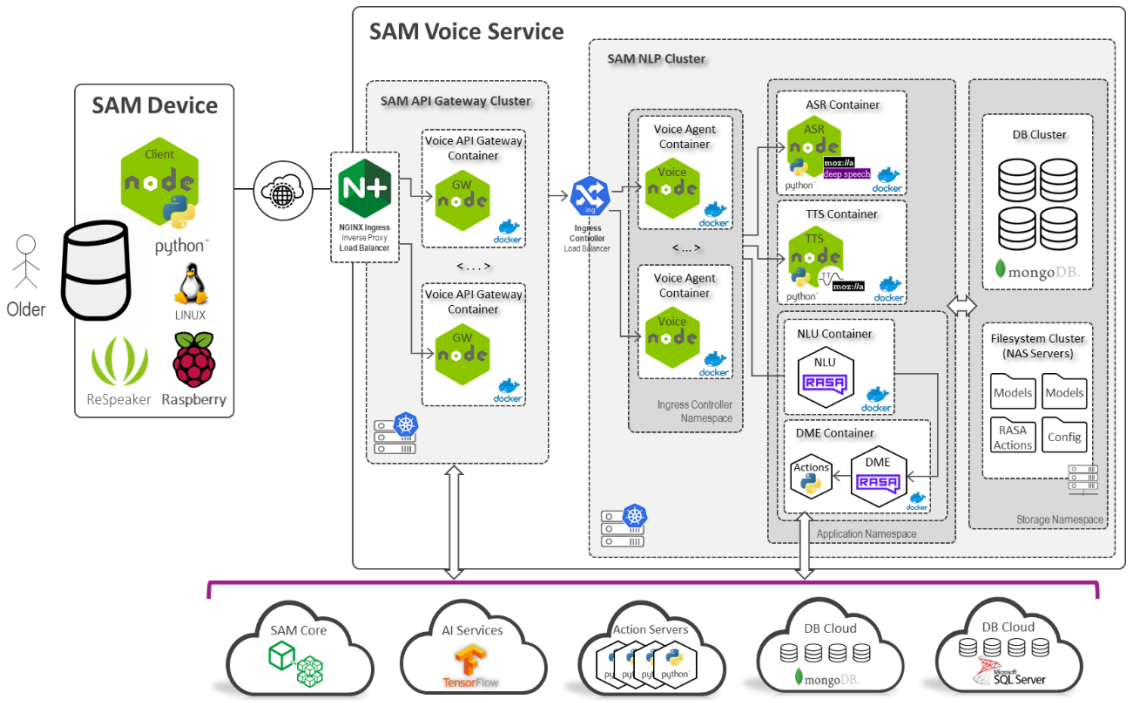


Figura 34. Arquitectura de despliegue de alta disponibilidad junto con las tecnologías involucradas.
FUENTE: elaboración propia.

Para la implementación de los *clústeres* utilizaremos *Kubernetes* (Kubernetes, 2021) con *Ngnix* (Nginx, 2021). ***Kubernetes*** es una plataforma de código abierto para la gestión de servicios y

procesos en contenedores (como *Docker*) que automatiza el lanzamiento y parada de instancias de estos procesos según la carga de trabajo necesaria. ***Nginx*** es un software de código abierto que actúa de *proxy inverso*, dirigiendo las peticiones entrantes al servidor a la ruta que necesitan, centralizando el acceso y ocultando la lógica de distribución de los servicios al exterior para un uso más transparente por parte del usuario.

Para configurar *Nginx* teniendo un clúster de *Kubernetes* basta con seguir los siguientes pasos:

```
# Aplicamos una de las configuraciones base de Kubernetes, guardada en su repositorio oficial en formato YAML
$ kubectl apply -f https://github.com/kubernetes/ingress-nginx/blob/master/deploy/static/provider/cloud/deploy.yaml
$ kubectl get pods -n ingress-nginx # Comprobamos que se ha configurado correctamente
```

Configuramos el *balanceador de carga*. Primero creamos un directorio para configuración de *Kubernetes* y un archivo de configuración vacío con:

```
# Creamos un directorio y entramos
$ mkdir ingress-deployment
$ cd ingress-deployment
# Creamos el archivo nginx-ingress.yaml
$ touch nginx-ingress.yaml
```

Y dentro de este archivo creado pegamos, cambiando los puertos y datos que correspondan:

```
kind: Service
apiVersion: v1
metadata:
  name: ingress-nginx
  namespace: ingress-nginx
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
spec:
  externalTrafficPolicy: Local
  type: LoadBalancer
  selector:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
  ports:
    - name: http
      port: 80
      targetPort: http
    - name: https
      port: 443
      targetPort: https
```

Después creamos el servicio de *Ingress* (que actúa como balanceador de carga) con:

```
$ kubectl apply -f nginx-ingress.yaml # Creamos el servicio de Ingress
$ kubectl get svc -n ingress-nginx # Comprobamos que se ha unido al balanceador de carga
```

A continuación, vinculamos un nombre de dominio al balanceador de carga, ya sea único o con comodín (este último permite enrutar distintos subdominios). Por último, vamos a añadir una aplicación a *Kubernetes* para permitirle gestionarla. Creamos un espacio de nombres y un archivo de configuración con:

```
$ kubectl create namespace sam # Creamos un espacio de nombres para SAM
$ touch sam-voice-service-app.yaml # Creamos el archivo de configuración vacío
```

Y dentro del archivo de configuración escribimos la siguiente configuración para el espacio de nombres:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sam-voice-service-app
  namespace: sam
spec:
  selector:
    matchLabels:
      app: sam-voice-service
  replicas: 3
  template:
    metadata:
      labels:
        app: sam-voice-service
    spec:
      containers:
      - name: sam-voice-service
        image: "www.samservice.es/icon.png"
```

Para luego ordenar el despliegue mediante las instrucciones:

```
$ kubectl create -f sam-voice-service-app.yaml # Creamos el despliegue
$ kubectl get deployments -n sam # Comprobamos que se ha desplegado bien
```

A continuación, vamos a crear la configuración del servicio que se ejecutará dentro del espacio de nombres antes configurado:

```
$ touch sam-voice-service-app-service.yaml # Creamos otro archivo de configuración
```

En este nuevo archivo copiamos la siguiente configuración:

```
apiVersion: v1
kind: Service
metadata:
  name: sam-voice-service-service
  namespace: sam
  labels:
    app: sam-voice-service
spec:
  type: ClusterIP
  selector:
    app: sam-voice-service
  ports:
  - port: 80
    targetPort: 8080
    protocol: TCP
```

Desplegamos el servicio y comprobamos su estado:

```
$ kubectl create -f sam-voice-service-app-service.yaml # Creamos
$ kubectl get svc -n sam # Comprobamos
```

Y por último creamos la configuración del balanceador de carga creando un objeto de *Ingress*:

```
$ touch ingress.yaml # Creamos otro archivo de configuración
```

Pegamos, cambiando la dirección del *host* por la que corresponda:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  namespace: sam
spec:
  rules:
  - host: <test.apps.example.info>
    http:
      paths:
      - backend:
          serviceName: sam-voice-service-service
          servicePort: 80
```

Y le describimos el objeto a *Ingress*:

```
$ kubectl describe ingress -n dev
```

Ya estaría configurado.

9 Pruebas y validación

En este capítulo se abordarán las pruebas que el trabajo desarrollado tuvo que pasar para ser considerado válido además de pruebas que validaron su robustez y su reacción a casos próximos a la realidad.

9.1 SERVICIO DE VOZ

9.1.1 Diseño de los escenarios de prueba

Para la realización de las pruebas se han construido tres prototipos hardware de **SAM Device** y, al mismo tiempo, se ha diseñado tres escenarios de despliegue diferente para **SAM Voice Service**: un escenario mínimo, un escenario intermedio y un escenario de alta disponibilidad.

El escenario mínimo (véase **Figura 35**) es similar al escenario de desarrollo. Puede ser válido para la realización de pruebas e incluso para el lanzamiento inicial del servicio con muy pocos usuarios —en el caso de las pruebas programadas inicialmente por el ayuntamiento, se seleccionaron 10 mayores.

El escenario intermedio (véase **Figura 36**) se ha diseñado para que cada servicio esté ubicado en un servidor propio. En este escenario se incorporan elementos adicionales a los meramente funcionales de forma que se facilite la escalabilidad y mejorar la seguridad del sistema;

El escenario de alta disponibilidad (véase **Figura 37**) está basado ampliamente en clústeres de servidores y redes de almacenamiento, y en balanceadores de carga, obteniendo un entorno válido para producción con gran capacidad de seguridad, escalabilidad, elasticidad y resiliencia.

Debido a la falta de recursos (servidores, balanceadores, clústeres, proxys, etc.), no se ha podido implementar totalmente el escenario de alta disponibilidad. Sin embargo, para la realización de las pruebas, hemos podido contar con un potente equipo que ha actuado de servidor apoyado por una potente *GPU GeForce RTX™ 3080* de 8704 núcleos y 10GB de memoria, de forma que las pruebas de rendimiento, sobre todo cuando están involucrados algoritmos de IA —como la aplicación de redes neuronales o su propio entrenamiento—, sí se han podido realizar de forma individualizada para cada servicio: *Servicio ASR*, *Servicio TTS*, *Servicio NLU*, *Servicio NLG*.

Hemos evitado incorporar en estos diseños otros servicios fundamentales para el desarrollo del sistema como son los servicios de identificación y autorización, los servicios de gestión y de *backOffice*, etc. debido a que seguirán esquemas de despliegue ampliamente utilizados e incorporarían una complejidad adicional a los esquemas que podría opacar los elementos importantes para el proyecto.

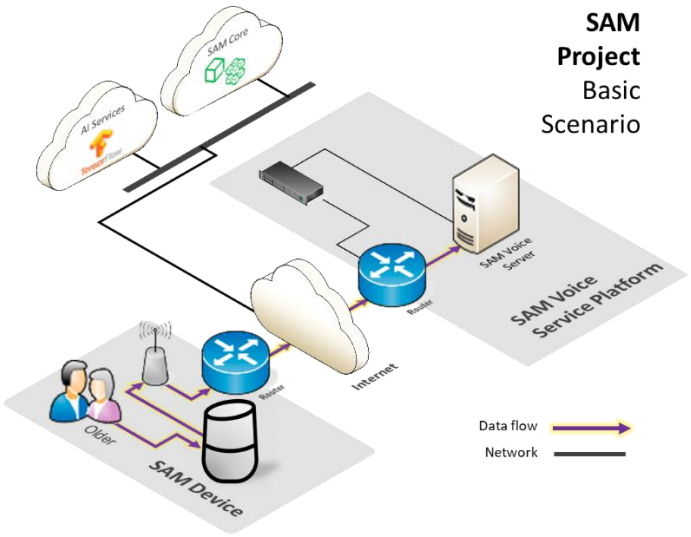


Figura 35. Escenario de despliegue mínimo.
FUENTE: elaboración propia.

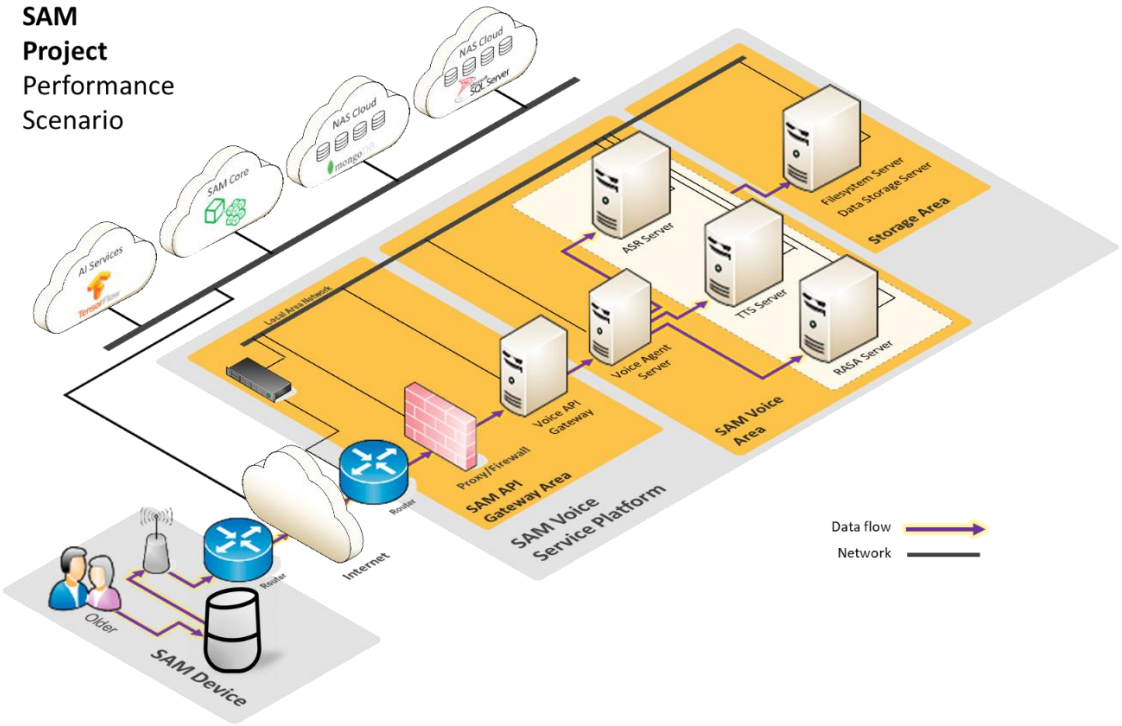


Figura 36. Escenario de despliegue intermedio.
FUENTE: elaboración propia.

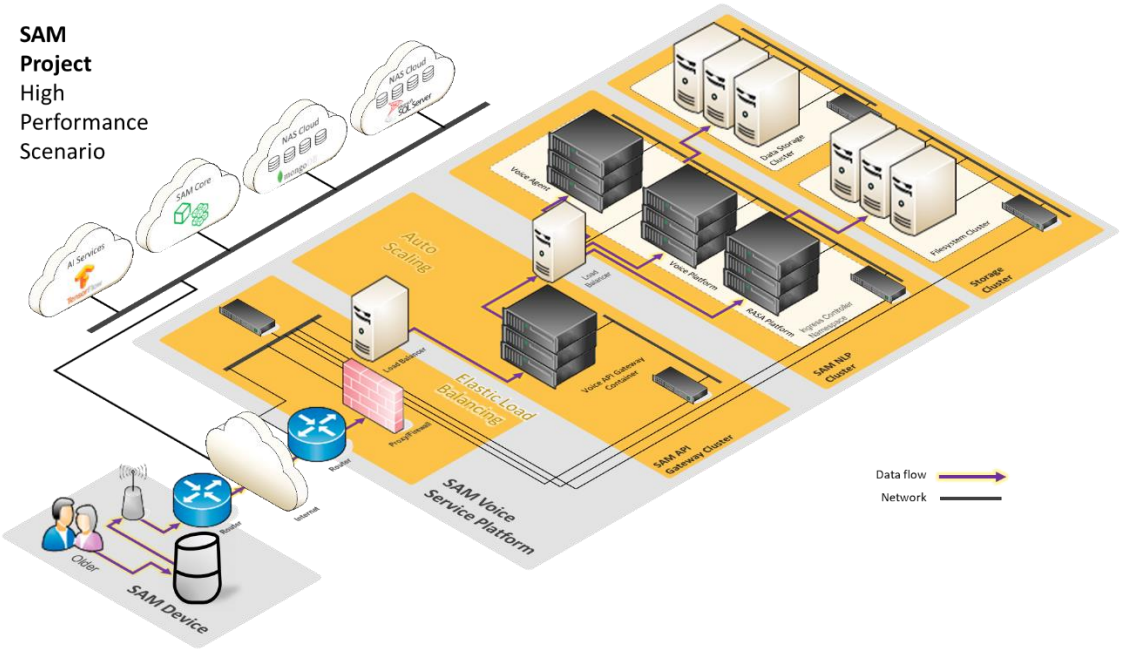


Figura 37. Escenario de despliegue de alta disponibilidad.
FUENTE: elaboración propia.

9.1.2 Diseño de los experimentos

Durante la construcción del servicio de voz tenemos que asegurarnos de que cada parte funciona correctamente. Lo primero es probar que los tres módulos que componen el servicio de voz (*ASR* basado en las bibliotecas *Mozilla Deepspeech*, *NLP* sobre una instancia local de la plataforma *Rasa*, y *TTS* basado en las bibliotecas *Mozilla TTS*) funcionan adecuadamente por separado. Para ello los probaremos con datos preparados de antemano cuyo resultado conocemos.

Prueba tipo 1: En el caso de *ASR*, probamos *Mozilla DeepSpeech* leyendo archivos de audio con frases pregrabadas del tipo "Esto es una prueba para reconocimiento de voz". En el caso de *TTS* probamos a usar como entrada textos como "Esto es una frase de prueba" y escribimos el audio resultante en un archivo que se pueda revisar varias veces. Para Rasa probamos a enviar mensajes sencillos que sabemos que están en su modelo. El objetivo de este último es ver que recibe los mensajes y que efectivamente nos responde.

Sabiendo que estos módulos funcionan de manera independiente en casos controlados tendremos que ponerlos en situaciones más cercanas a las que tendrán que afrontar cuando se encuentren todos trabajando de forma interconectada.

Prueba tipo 2: Para *ASR* y *TTS*, en lugar de un programa que lee ciertos datos al ejecutarse, lanzamos un servidor HTTP que hace lo mismo para cada petición que recibe. Además, para *ASR*, recibe en la petición un audio en formato binario que será el que tenga que usar para traducir, así como el de *TTS* que tendrá que devolver el audio en binario en la respuesta al cliente en lugar de escribir el resultado en un archivo de audio.

Con esto, los tres módulos estarían listos para actuar como subservicios para el servicio de voz principal.

Prueba tipo 3: Vamos a probar un servicio de voz que recibe audio como entrada y devuelve audio como salida. Para la prueba, este servicio responderá con el mismo audio que ha recibido.

Por último, cuando pase todas estas pruebas, querremos probarlo todo junto. Si el servicio de voz es capaz de recibir y enviar audio, y los tres módulos son capaces de hacer su trabajo bien, un servicio de voz que pase el audio de entrada por todos estos módulos debería ser sencillo.

Prueba tipo 4: El servicio de voz central orquesta la llamada tal y como diseñamos. Primero envía el audio que ha recibido al servicio de ASR, envía el texto que devuelve *ASR* al servidor de *NLP*, recupera la respuesta en texto y la envía al servicio de *TTS* para que este le devuelva un audio, y por último devuelve este audio respuesta al cliente.

Con estas cuatro pruebas el servicio de voz estará listo para ofrecer su funcionalidad en un entorno previsto, pero ahora tenemos que probar qué ocurre si hay algún evento inesperado. Para ello diseñamos dos tipos de pruebas más:

Prueba tipo 5: Audio casi silencioso como entrada al módulo de *ASR*, que pueda resultar en un texto errático o sin sentido.

Prueba tipo 6: Enviar un texto vacío al módulo de *TTS*. Si se da esta situación es que ha fallado algo totalmente inesperado antes, así que debemos asegurarnos de que esto no detenga la ejecución del programa.

9.1.3 Ejecución de las pruebas

La **prueba tipo 1** (módulos independientes con datos de entrada preestablecidos) funcionó sin mayor problema, aunque la calidad de los modelos en español de *ASR* y *TTS* era mejorable, pues algunas palabras no las reconocía o no la pronunciaba bien. Habrá que seguir atentos al desarrollo de estos modelos por parte de la comunidad de Mozilla conforme se donen más muestras de voz.

La **prueba tipo 2** (*ASR* y *TTS* en servidores), tras averiguar cómo enviar y recibir el audio a través de peticiones http en *NodeJS*, fue sencillo de hacer funcionar y de probar.

La **prueba tipo 3** (servicio HTTP que recibe y responde con audio) fue trivial habiendo completado la prueba 2, que tenía elementos muy similares.

La **prueba tipo 4** (todos los módulos juntos, orquestados por el servicio de voz) consistió en unir las piezas ya probadas arriba y funcionó correctamente. Tal y como hemos dicho antes los modelos en español de *ASR* y *TTS* son mejorables, lo que hace que a veces el asistente y el usuario no se entiendan a la perfección.

La **prueba tipo 5** (audio demasiado silencioso en *ASR*) hace que el módulo de *ASR* dé como resultado un texto sin sentido. Esto terminará en el *NLP* como una frase no reconocida y le dirá

al usuario que no le ha entendido y que vuelva a intentarlo. Es lo mejor que se puede hacer en esta situación, así que está bien.

La prueba **tipo 6** (pasar un texto vacío al módulo de TTS) da como resultado un audio vacío, sin mayor error. Esto es lo que esperábamos y es lo mejor que podemos hacer en esta situación. Al fin y al cabo, es traducir lo que ha dicho el asistente inteligente a voz, y no entorpece al resto del servicio.

9.1.4 Análisis de los resultados

El servicio de voz se comporta adecuadamente, pero los modelos de *ASR* y *TTS* responsables de la calidad del reconocimiento de la voz en cada idioma y de la síntesis de voz, respectivamente, deberán mejorar para ofrecer una interacción perfecta: mejor entendimiento y una voz más cercana a la de una persona real.

Entrenar y mejorar estos modelos requiere mucho tiempo e infinidad de muestras para entrenar las redes neuronales, por lo que queda totalmente fuera del alcance de este trabajo.

9.2 DISPOSITIVO IOT

9.2.1 Diseño de los experimentos

Desde un punto de vista físico, gracias a que casi toda la implementación se ha basado en placas de desarrollo hardware, tan solo se ha tenido que verificar el funcionamiento de algún componente electrónico adicional como botones, leds, pulsadores, fuente de alimentación y altavoz. Queda pendiente la incorporación de otros elementos como sensores (temperatura, luz, humedad, etc.) y de algún actuador (relés) para controlar luces y electrodomésticos. La validación de los prototipos básicamente se puede realizar mediante su encendido y comprobación de los leds de inicio y carga del sistema operativo. Es por esta razón que no hemos diseñado por el momento ningún tipo de prueba especial para el hardware además de la que ya se incluye implícitamente en su propio diseño.

Por otra parte, el dispositivo IoT, desde el punto de vista de su software, se encarga de una sola tarea que tiene unos pasos muy marcados. Tendremos que asegurarnos de que cada paso se

realiza correctamente y luego comprobar que se conectan bien entre ellos. Los pasos que da el dispositivo IoT son los siguientes:

Prueba tipo 1: Grabar audio y detectar *Hotword* con *Snowboy*. El dispositivo debe estar grabando en todo momento, y cuando escuche la palabra de activación entonces indicarlo.

Prueba tipo 2: Detectar el ruido o el silencio en el audio grabado. El dispositivo debe diferenciar si en un audio hay habla o solo silencio.

Prueba tipo 3: Ejecutar las pruebas tipo 1 y 2 de manera conjunta. Tendrá que grabar para *Snowboy* y con los segmentos de audio que *Snowboy* clasifica, filtrar los que tengan habla o los que no.

Prueba tipo 4: Identificar cuando empieza y termina una orden. El dispositivo debe ser consciente de cuándo empieza la orden (cuando oye la palabra de activación) y cuándo termina (cuando el usuario se calla).

Prueba tipo 5: Enviar audio al servicio de voz en *stream* y realizar petición HTTP. El dispositivo debe iniciar una petición HTTP al servicio de voz, ir enviando los segmentos de audio que grabe y que contengan habla, y terminar la petición cuando la orden acabe.

Prueba tipo 6: Reproducir *stream* de audio por los altavoces. El dispositivo debe ser capaz de reproducir audio por los altavoces, y deberá reproducir el audio con el que le responda el servidor cuando termine la orden.

Prueba tipo 7: Todo junto, en orden. Realizar todos los pasos arriba descritos en orden, formando así el flujo de trabajo mínimo del dispositivo IoT.

Además, tendremos que probar las funcionalidades secundarias, las cuales son importantes, pero no afectan al flujo de trabajo principal.

Prueba tipo 8: Usar los patrones de leds. El dispositivo debe tener los leds apagados de manera normal. Cuando escuche la palabra de activación debe poner el patrón de *escuchar* hasta que oiga el final de la orden. En ese momento pondrá el patrón de *pensar* mientras espera la respuesta del servidor. Cuando reciba dicha respuesta y empiece a reproducirla por los altavoces, pondrá el patrón de *hablar*. Cuando termine de reproducir la respuesta apagará los leds.

El dispositivo IoT tendrá que pasar todas estas pruebas para funcionar. Ahora definimos las pruebas de casos no tan perfectos, muy probables en entornos reales.

Prueba tipo 9: Empezar una orden teniendo otra empezada. Si el usuario dice la palabra de activación en mitad de otra orden, el dispositivo debe descartar la primera orden e iniciar una nueva, cerrando la primera petición HTTP y reiniciando los patrones de leds.

9.2.2 Ejecución de las pruebas

Las **pruebas de tipos 1 a 7** se llevaron a cabo durante el desarrollo, pues prueban la funcionalidad mínima del dispositivo paso por paso, y terminaron sin mayor problema. La **prueba tipo 8** (usar los leds) dio buenos resultados también.

La **prueba tipo 9** (empezar una orden teniendo otra empezada) dio problemas al principio, ya que la aplicación no tenía en cuenta esta situación, pero con unas pocas líneas de código al principio para resetear ciertos parámetros pasó sin problemas.

9.2.3 Análisis de los resultados

El dispositivo IoT se comporta adecuadamente, aunque la detección de la *Hotword* no es tan precisa como se desearía. Cambiar el motor de *Hotword* a uno activo queda pendiente para más adelante, por lo que no es una preocupación grande ahora.

De nuevo, similar al apartado anterior, lo que más mejora requiere es un modelo de IA. Estos modelos, como ya hemos dicho antes, cambiarán enormemente cuando tengamos muestras de muchas personas, por lo que no es de extrañar que ahora no sean perfectos.

9.3 COMPARATIVA DE RENDIMIENTO

Probamos los distintos módulos que componen el trabajo (Detección de *Hotword*, ASR, NLP y TTS) en varios dispositivos para comparar los tiempos de ejecución en función de sus características y de los recursos hardware que posean. Así podremos estudiar qué dispositivo es innecesariamente potente para el trabajo y cuál no puede soportarlo, estableciendo unos márgenes para el equipamiento que necesitaremos para el despliegue.

Los dispositivos para comparar son: nuestros tres prototipos para dispositivo IoT (*ReSpeaker Core V2*, *Raspberry Pi + 4-Mic Array* y *2-Mic Array*), una máquina virtual y un ordenador personal. Adicionalmente, se prueban distintas frases de distintas longitudes en la máquina de prueba principal para comprar los tiempos de ejecución según la carga.

En la **Tabla 9** se muestran los tiempos medios obtenidos tras las pruebas en distintos hardware. En la **Tabla 10** se muestran los tiempos medios, mínimos y máximos de las pruebas para cada frase, así como el número de pruebas, todas realizadas en la máquina virtual.

Tabla 9. Tabla comparativa con los tiempos medios de la ejecución de los distintos módulos del trabajo en hardware variado.
FUENTE: elaboración propia.

	Hotword	ASR	NLP	TTS
ReSpeaker Core V2	0.052s	-	-	-
Raspberry Pi	0.051s	-	-	-
Máquina virtual sobre PC	-	2.611s	0.251s	8.610s
PC	-	0.562s	0.190s	1.784s

Tabla 10. Tabla comparativa con los tiempos de la ejecución de los distintos módulos del trabajo en máquina virtual.
FUENTE: elaboración propia.

Módulo	Frase	Media	Mínimo	Máximo	Pruebas
ASR Largo	Esto es una prueba para reconocimiento de voz	2.6108s	2.509s	3.682s	50
TTS Largo	Hi there! How are you my dear human being?	8.6102s	8.202s	9.705s	50
ASR medio	Esto es una prueba	1.6454s	1.355s	1.979s	50
TTS medio	Hey! How are you?	5.8645s	5.261s	6.661s	50
ASR corto	Hola	0.7968s	0.562s	1.078s	50
TTS corto	Hi	4.3248s	4.069s	4.755s	50

Se trata de experimentos muy rudimentarios y primitivos que solo pretenden en estos momentos proporcionar algún marco de referencia para tomar decisiones acerca del despliegue del sistema y poder establecer un cociente entre el rendimiento obtenido y los costes de cada plataforma.

Ya estamos trabajando un conjunto de experimentos mucho más exhaustivos que permitan obtener una información más interesante. Sin embargo, este estudio queda totalmente fuera de los objetivos de este trabajo debido a diferentes condicionantes (como disponer de todos los

componentes y escenarios para su ejecución o tener un lugar adecuado para realizarlas) pero, sobre todo, por la necesidad de contar con el tiempo que se requiere.

9.4 PRUEBAS EN ENTORNOS REALES

Lamentablemente, la situación sanitaria actual ha impedido que las pruebas programadas con mayores se lleven a cabo en el plazo del TFG. Por suerte ya se ha vuelto a retomar el proyecto y todo apunta a que durante los próximos meses se podrían retomar nuevamente.

En cuanto sea posible, se llevarán a cabo las pruebas y los resultados se utilizarán para mejorar el servicio, los modelos de conversación, las interfaces y los protocolos y rutinas establecidos inicialmente.

Conclusiones

10 Conclusiones y trabajo futuro

En este trabajo hemos analizado el problema del envejecimiento activo, llegando a la conclusión de que hacen falta soluciones que aprovechen las nuevas tecnologías para paliar los efectos de la vejez en los mayores y en su entorno.

Nos planteamos el objetivo de crear un dispositivo físico capaz de asistir y de acompañar a los mayores comunicándose con ellos mediante voz y lenguaje natural. Se trata de una propuesta que se podrá integrar dentro de un proyecto más ambicioso desarrollado de forma conjunta con otras universidades y empresas para algunos ayuntamientos de la Comunidad Valenciana.

He creado una solución basada en las TI para lo cual he concebido una arquitectura distribuida, he diseñado los sistemas software y hardware necesarios y he desarrollado tres prototipos hardware de dispositivos IoT que interactúan por voz junto con una plataforma capaz de ofrecer servicios en la nube para el procesamiento de voz y del lenguaje natural. Finalmente, he podido integrar todo el nuevo sistema dentro del ecosistema de servicios para el envejecimiento saludable creados para los ayuntamientos.

Al final, este proyecto ha abarcado casi todos los ámbitos que he aprendido en la carrera, ayudándome a afianzar y extender todos los conocimientos que he adquirido desde que empecé el grado, hace cuatro años. Algunas de las más relevantes, aunque por supuesto todas han contribuido en mayor o menor media, son: **programación** y **desarrollo de aplicaciones y servicios web**, para la creación del dispositivo IoT y del servicio de voz; **sistemas distribuidos**, para la adecuada interconexión de los distintos elementos del trabajo repartidos por la red; **usabilidad y accesibilidad**, para el cuidadoso diseño que requiere una interfaz para personas

mayores o dependientes; **modelaje 3D**, para el diseño de la carcasa del dispositivo; y el **ABP** en general para la organización y estructuración.

Aunque las tecnologías nunca pueden sustituir a los familiares y a los profesionales, nuestro proyecto puede ayudar enormemente a mejorar la independencia y la calidad de vida de todos los implicados, tanto la de los mayores por estar y sentirse acompañados, como la de los asistentes y familiares, aliviando la carga de trabajo y las preocupaciones que acarrea la responsabilidad de que sus seres queridos se encuentren bien en todo momento.

Este trabajo ha visto su planificación alterada gravemente por la situación de pandemia, ya que ha limitado las interacciones del grupo de trabajo con los servicios sociales con los que colaboraban, reduciendo así la cantidad de información obtenida para el desarrollo de un modelo de conversación satisfactorio; ha aplazado las pruebas ya planificadas con personas mayores reales que habrían permitido mejorar la usabilidad de nuestro servicio enormemente; y ha causado estragos en los recursos de los proveedores, evitando que llegaran a tiempo los componentes electrónicos que se precisaban.

Aun así, ha salido adelante con todos los objetivos que se proponían en un inicio. Por supuesto todavía puede crecer más. Todas las tareas retrasadas por la pandemia siguen en el calendario, y en cuanto la situación lo permita se llevarán a cabo. También hay muchas mejoras técnicas que pueden hacer más amigable la interacción con el dispositivo, fruto de ideas que se planteaban para un futuro o que surgieron durante las pruebas del trabajo.

10.1 TRABAJO FUTURO

A corto plazo, y para poder integrar la plataforma de voz en los servicios de SAM con los requerimientos mínimos, es necesario crear un Gateway de acceso al **SAM Voice Service** que gestione el acceso a todo el sistema.

También hay que añadir al dispositivo IoT (**SAM Device**) la funcionalidad de cortar una orden si se alarga demasiado, o si el usuario activa el dispositivo y no dice nada. Esto sirve principalmente para evitar que demasiado tiempo de grabación sature la memoria del dispositivo o del servicio de voz, y que se caliente trabajando en vano, especialmente si se ha activado sin querer y se ha quedado escuchando.

Después, cuando la situación sanitaria actual lo permita, se llevarán a cabo las reuniones y pruebas planificadas con las personas mayores seleccionadas en Alcoy, con el objetivo de mejorar el modelo de conversación del asistente para el gran público antes de implementarlo en los hogares.

Adicionalmente, se propone un despliegue de alta disponibilidad (ver **Figura 34**) que utiliza *clústeres* de *Kubernetes* y balanceadores de carga para extender automáticamente la capacidad del servicio de voz según el tráfico.

A más largo plazo, se proponen ideas que sigan mejorando la experiencia de interacción con SAM Device.

La primera de estas propuestas consiste en añadir una pantalla al SAM Device para ofrecer una interfaz gráfica desde el asistente, que permita interactuar con la información de manera complementaria y sirva de alternativa para las personas que no pueden oír bien, ya sea por razones auditivas o por el entorno en el que viven.

También permitir al usuario configurar parámetros del dispositivo, así como la conexión *Wi-Fi*, desde una app externa o desde el *Dashboard* ya existente si ya están conectados.

Otra línea muy interesante es la de incorporar sensores y actuadores a SAM Device para que pueda interaccionar directamente con su entorno y tener una percepción más ajustada a la realidad de lo que está ocurriendo en el entorno del mayor.

Dentro del campo de la inteligencia artificial se pueden plantear infinidad de proyectos diferentes, como la detección de sentimientos a partir de la voz, la toma de decisiones inmediatas o de emergencia, incluso ante la desconexión del dispositivo a la red de comunicaciones, ayuda a la toma de decisiones por parte de todos los implicados: familias, facultativos, personal sociosanitario.

Y por supuesto, hasta que termine la vida del proyecto, siempre habrá que estar mejorando y manteniendo los modelos para inteligencia artificial que utilizan nuestros servicios: ASR, TTS, detección de *Hotword*, y el más importante: el modelo de conversación.

También en un futuro hipotético se cambiará el motor de detección de *Hotword* a uno nuevo, que esté activo, o incluso crear el nuestro propio, ya que *Snowboy* se descontinuó a principios de este año.

En resumen, ha sido un proyecto tan apasionante como trepidante, en el que además de haber aprendido mucho y de sentir que soy capaz de crear algo útil para la sociedad, incluso para el bienestar de los míos, siento que he crecido enormemente y que he aprendido a valorar detalles que ni había imaginado antes de comenzar con el trabajo: como son la investigación, la documentación y la organización.

Todo el trabajo se ha desarrollado bajo una filosofía de código abierto y de hardware abierto, por lo que espero que pueda llegar a ser una pequeña aportación para todos aquellos que quieran trabajar o estén trabajando dentro del ámbito de la aplicación de las TI para ayudar a un envejecimiento saludable.

11 Referencias

A continuación, se listan todas las referencias utilizadas en este Trabajo de Fin de Grado, organizadas por orden alfabético. A la izquierda de cada elemento se puede ver cómo se referencia en el texto, paréntesis incluidos.

(AAL, 2008) AAL Program (2008). Ageing Well in the Digital World. EU. Since 2008. http://www.aal-europe.eu/.

(AAL-2, 2014) AAL-2 Program (2014). Active and Assisted Living Research and Development Program. AAL Program. AAL-2 Program. http://www.aal-europe.eu/.

(Abellán et al., 2017) Abellán, Antonio, et al. Partner care, gender equality, and ageing in Spain and Sweden. International Journal of Ageing and Later Life 11, no. 1, 2017. https://doi.org/10.3384/ijal.1652-8670.16-305.

(AgeLight, 2001) AgeLight (2001) Interface design guidelines for users of all ages. Agelight. Technology & Generational Marketing Strategies. 2001

(Alcoy Street Lab, 2018) Alcoy Street Lab (2018). Noticia en la página del ayuntamiento de Alcoy sobre el Proyecto Street Lab realizado allí. https://www.alcoi.org/es/portada/noticias2/noticia_0219.html.

(Ângelo et al., 2012) Ângelo Costa, José Carlos Castillo, Paulo Novais, Antonio Fernández-Caballero, Ricardo Simoes, Sensor-driven agenda for intelligent home care of the elderly, Expert Systems with Applications, 2012, 39, 12192-12204

(Arem et al., 2015) Arem H, Moore SC, Patel A, Hartge P, Berrington de Gonzalez A, Visvanathan K, et al. Leisure time physical activity and mortality: a detailed pooled analysis of the dose-response relationship. JAMA Intern Med. 2015 Jun;175(6):959–67. PMID: 25844730

(Badenes y López, 2011) Badenes Plá, Nuria, and López López, M. T. Doble dependencia: abuelos que cuidan nietos en España. Zerbitzuan: Gizarte

zerbitzuetarako aldizkaria= Revista de servicios sociales, n.º 49, 2011. https://doi.org/10.5569/1134-7147.49.09.

(Baidu, 2014) Baidu, 2014. Computation and Language (cs.CL); Machine Learning (cs.LG); Neural and Evolutionary Computing (cs.NE) arXiv:1412.5567v2. https://arxiv.org/abs/1412.5567v2.

(Balakrishnan et al., 2012) Balakrishnan, S., Salim, S. S. B., and Hong, J. L. (2012). "User Centered Design Approach for Elderly People in Using Website," Proceedings of the 2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT 2012), pp. 382-387.

(Balena Etcher, 2021) Balena Etcher. Página oficial de la herramienta BalenaEtcher. https://etcher.io/. (Último acceso: 2021).

(Basu, 2019) Basu, T. (2019). Alexa will be your best friend when you're older. MIT Technology Review. Aug, 2019. URL: https://www.technologyreview.com/2019/08/30/102685/alexa-will-be-your-best-friend-when-youre-older/.

(Bauman et al., 2016) Bauman A, Singh M, Buchner D, Merom D, Bull F. (2016). Physical activity in older adults. Gerontologist. 2016 (In press.).

(Bengio et al., 2017) Bengio, Y., Goodfellow, I., & Courville, A. (2017). Deep learning (Vol. 1). Massachusetts, USA:: MIT press.

(Biblioteca UA, 2013) Biblioteca UA. Universidad de Alicante. La búsqueda de información científica. Universidad de Alicante. Biblioteca Universitaria. Disponible en: http://hdl.handle.net/10045/51538

(Bloom, 2011) Bloom DE. 7 billion and counting. Science. 2011 Jul 29;333(6042):562–9.doi: http://dx.doi.org/10.1126/science.1209290 PMID: 21798935.

(Bobeth et al., 2012) Bobeth, J., Deutsch, S., Schmehl, S., and Tscheligi, M. (2012). "Facing the User Heterogeneity when Designing Touch Interfaces for Older Adults: A Representative Personas Approach," NordiCHI 2012 Proceedings, pp 1-4.

(Bobrovskis y Jurenoks, 2018) Bobrovskis, S. y Jurenoks, A. (2018). A Survey of Continuous Integration, Continuous Delivery and Continuous Deployment. Joint Proceedings of the BIR 2018 Short Papers, Workshops and Doctoral Consortium, Stockholm, Sweden, September 24-26, 2018. Edited by: Jelena Zdravkovic, Jānis Grabis, Selmin Nurcan, Janis Stirna, Riga Technical University, Latvia. Disponible en: http://ceur-ws.org/Vol-2218/paper31.pdf. (Último acceso: 27/08/2021).

(Casati y Donato, 2020) Casati, E., Donato, S. The relationship between family members and the care team of health care residence (RSA): The operators' perspective, Psicologia della Salute, 2020, 2, 25-51.

(Castellano Lendínez, 2019) Castellano Lendínez, L. (2019). Kanban. Metodología para aumentar la eficiencia de los procesos. 3C Tecnología. Glosas De Innovación Aplicadas a La Pyme, 30-41. Recuperado a partir de http://ojs.3ciencias.com/index.php/3c-tecnologia/article/view/766. (Último acceso: 2020).

(Christensen et al., 2009) Christensen K, Doblhammer G, Rau R, Vaupel JW. Ageing populations: the challenges ahead. Lancet. 2009 Oct 3;374(9696):1196–208.doi: http://dx.doi.org/10.1016/S0140-6736(09)61460-4. PMID: 19801098. Descripción: referenciado en (OMS, 2015).

(CMU Sphinx, 2020) Página oficial del proyecto CMU Sphinx. https://cmusphinx.github.io/. (Último acceso: 2020).

(CMUFlite, 2020) Página oficial de CMUFlite. http://www.festvox.org/flite/. (Último acceso: 2020).

(DCC, 2018) DCC (2018) Digital Cities Challenge: A step to Achieve Digital Development in Europe. https://europa.eu/regions-and-cities/news/digital-cities-challenge-step-achieve-digital-development-europe_en. (Último acceso: 28/05/2021).

(DeepPavlov.ai, 2021) DeepPavlov.ai, página oficial. http://deeppavlov.ai/. (Último acceso: 2021)

(DeepSpeech, 2021) DeepSpeech Mozilla, Github official de Mozilla DeepSpeech. https://github.com/mozilla/DeepSpeech. (Último acceso: 2021)

(del Barrio et al., 2015) Del Barrio, Elena, Olga Mayoral, y Mayte Sancho (Matia Instituto Gerontológico). Estudio sobre las condiciones de vida de las personas de 55 y más años en Euskadi. Vitoria-Gasteiz: Servicio Central de Publicaciones del Gobierno Vasco, Documentos de Bienestar Social, n.º 77, 2015. https://www.matiainstituto.net/es/publicaciones/estudio-sobre-las-condicionesde-vida-de-las-personas-de-55-y-mas-anos-en-euskadi.

(Demiris G., 2006) Demiris G. The diffusion of virtual communities in health care: concepts and challenges, Patient Education and Counseling, 2006, 62(2), 178–188.

(Docker Install, 2021) Docker, página de instalación en Ubuntu. https://docs.docker.com/engine/install/ubuntu/. (Último acceso: 2021).

(Docker NodeJS, 2021) Docker, uso con NodeJS. https://github.com/nodejs/docker-node/blob/main/README.md#how-to-use-this-image. (Último acceso: 2021).

(Docker Rasa, 2021) Docker, uso con Rasa. https://rasa.com/docs/rasa/docker/building-in-docker/. (Último acceso: 2021).

(Dodd et al., 2017) Dodd, C., R. Athauda and M.T.P. Adam. (2017) Designing User Interfaces for the Elderly: A Systematic Literature Review. Conference: Australasian Conference on Information Systems. Hobart, Australia. 2017. Disponible en: https://www.researchgate.net/publication/321757825_Designing_User_Interfaces_for_the_Elderly_A_Systematic_Literature_Review.

(Dragonfire, 2021) Dragonfire, Open Source Voice Assistant. https://github.com/DragonComputer/Dragonfire. (Último acceso: 2021)

(EC, 2017-2019) EC. European Commission. MDCG, Guidance on Qualification and Classification of Software in Regulation (EU) 2017/745 – MDR and Regulation (EU) 2017/746 – IVDR. https://ec.europa.eu/docsroom/documents/37581; MDCG 2019-11, Guidance on Qualification and Classification of Software in Regulation (EU) 2017/745 – MDR and Regulation (EU) 2017/746 – IVDR. Brussels: European Commission. https://ec.europa.eu/health/sites/health/files/md_topicsinterest/docs/md_mdcg_2019_11_guidance_en.pdf.

(Ertel, 2018) Ertel, Wolfgang (2018). Introduction to artificial intelligence 2nd ed. Springer. Undergraduate Topics in Computer Science. 2018. url: https://books.google.es/books?hl=es&lr=&id=geFHDwAAQBAJ&oi=fnd&pg=PR5&dq=introduction+artificial+intelligence&ots=3F4u8ajA4x&sig=3Xg2NbllzVYMDhM-4bl4nGgTszk#v=onepage&q=introduction%20artificial%20intelligence&f=false.

(eSpeak, 2020) Página oficial de eSpeak. http://espeak.sourceforge.net/. (Último acceso: 2020).

(Esposito et al., 2019) Esposito, A., Terry Amorese, Marialucia Cuciniello, Maria Teresa Riviello, Antonietta M. Esposito, Alda Troncone, and Gennaro Cordasco. The Dependability of Voice on Elders' Acceptance of Humanoid Agents. INTERSPEECH 2019 September 15–19, 2019, Graz, Austria. https://www.isca-speech.org/archive/Interspeech_2019/pdfs/1734.pdf.

(EUR-Lex, 2017) EUR-Lex. (2017). Regulation (EU) 2017/745 of the European Parliament and of the Council of 5 April 2017 on medical devices, amending Directive 2001/83/EC, Regulation (EC) No 178/2002 and Regulation (EC) No 1223/2009 and repealing Council Directives 90/385/EEC and 93/42/EEC (Text with EEA relevance.). Brussels: European Commission. https://eur-lex.europa.eu/legalcontent/EN/TXT/?uri=CELEX%3A32017R0745; not yet fully applicable.

(Faria et al., 2020) Faria, A.D.C.A., Martins, M.M.F.P.D.S., Ribeiro, O.M.P.L., Gomes, B.P., Fernandes, C.S.N.D.N. Elderly residents in the community: gaining knowledge to support a rehabilitation nursing program Revista brasileira de enfermagem, 2020, 73 (3), e20200194.

(Francesca y Giovanni, 2009) Francesca Bettio, Giovanni Solinas, Which European Model for Elderly Care? Equity and Cost-Effectiveness in Home Based Care in Three European Countries, Center for the Analysis of Public Policies (CAPP), Universita di Modena e Reggio Emilia, Dipartimento di Economia "Marco Biagi", 2009, 64.

(Galitz, 2002) Galitz, W. O. (2002). “The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques,” Second Edition, John Wiley and Sons.

(Gitlin, 1995) Gitlin, L.N. (1995). WHY OLDER PEOPLE ACCEPT OR REJECT ASSISTIVE TECHNOLOGY. Generations: Journal of the American Society on Aging. Vol. 19, No. 1, Technology and Aging: Developing and Marketing New Products for Older People (SPRING 1995), pp. 41-46 (6 pages). Published by: American Society on Aging. URL: https://www.jstor.org/stable/44877289.

(Glisky, 2007) Glisky, E. L. (2007). “Changes in Cognitive Function in Human Aging,” In Brain Aging: Models, Methods, and Mechanisms, pp 4-20, CRC Press.

(Granata et al., 2013) Granata, C., Pino, M., Legouverneur, G., Vidal, J. S., Bidaud, P., and Rigaud, A. S. (2013). “Robot Services for Elderly with Cognitive Impairment: Testing Usability of Graphical User Interfaces,” Technology & Health Care (21:3), pp 217-231.

(Gregor et al., 2002) Gregor, P., Newell, A. F., and Zajicek, M. (2002). “Designing for Dynamic Diversity: Interfaces for Older People,” Proceedings of the 5th International ACM Conference on Assistive Technologies, pp 151-156, Edinburgh, Scotland.

(Greyling, 2020) Greyling, C. (2020) Ocho cosas que diferencian a Rasa de otras plataformas de chatbot. 2020. https://planetachatbot.com/diferencias-rasa-otras-plataformas-chatbot/. (Último acceso: 2021)

(Grindrod et al., 2014) Grindrod, K. A., Li, M., and Gates, A. (2014). “Evaluating User Perceptions of Mobile Medication Management Applications with older Adults: A Usability Study,” JMIR Mhealth Uhealth, (2:1), e11.

(Heerink, 2006) Heerink, M., B. Krose, V. Evers and B. Wielinga. (2006). "The Influence of a Robot's Social Abilities on Acceptance by Elderly Users," ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human

Interactive Communication, 2006, pp. 521-526, doi: 10.1109/ROMAN.2006.314442.

(Hendrich et al., 2015) Hendrich, N., Hannes Bistry, Jianwei Zhang (2015). Architecture and Software Design for a Service Robot in an Elderly-Care Scenario. Engineering 2015, 1(1): 27–35 DOI 10.15302/J-ENG-2015007.

(Holt, 2000) Holt, B. (2000) Creating senior-friendly web sites. PMDI. Issue Brief Cent Medicare Educ. 2000;1(4):1-8.

(Horizonte2020, 2014) Horizonte2020 (2014) Programa Marco Horizonte 2020. https://eshorizonte2020.es/.

(ICC, 2020) ICC (2020) Proyecto Europeo Intelligent Cities Challenge' (ICC) de la Comisión Europea. https://www.intelligentcitieschallenge.eu/. (Último acceso: 28/05/2021).

(ISO, 2017) ISO (2017). UNE-EN ISO 9241-125:2017 Ergonomía de la interacción hombre-sistema. Parte 125: Guía relativa a la presentación visual de la información (ISO 9241-125:2017)

(ISO, 2018) ISO (2018). UNE-EN ISO 9241-306:2018 Ergonomía de la interacción hombre-sistema. Parte 11: Usabilidad. Definiciones y conceptos (ISO 9241-11:2018)

(Jackson, 2019) Jackson, Philip C. (2019). Introduction to artificial intelligence (3rd Ed.). Dover Publications, Inc. New York. 2019. url: https://books.google.es/books?hl=es&lr=&id=vC-oDwAAQBAJ&oi=fnd&pg=PA33&dq=introduction+artificial+intelligence&ots=XLW34HKyDq&sig=ltUlkskWbPGi997-bHZFrAqCGZw#v=onepage&q=introduction%20artificial%20intelligence&f=false.

(Jak, 2012) Jak AJ. (2012). The impact of physical and mental activity on cognitive aging. Curr Top Behav Neurosci. 2012; 10:273–91. PMID: 21818703

(Julius, 2020) Página oficial del proyecto Julius. https://julius.osdn.jp/en_index.php. (Último acceso: 2020).

(Kaldi, 2020) Página oficial de Kaldi. https://kaldi-asr.org/. (Último acceso: 2020).

(Klimlova et al., 2018) Klimova B., Poulova P. Older People and Technology Acceptance. In: Zhou J., Salvendy G. (eds) Human Aspects of IT for the Aged Population. Acceptance, Communication and Participation. ITAP 2018. Lecture Notes in Computer Science, 2018, 10926, 85-94.

(Kubernetes, 2021) Kubernetes, página oficial sobre qué es Kubernetes. https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/. (Último acceso: 01-09-2021).

(Kurniawan & Zaphiris, 2005) Kurniawan, S.H. & Zaphiris, P. (2005) Research-Derived Web Design Guidelines for Older People. Conference: Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2005, Baltimore, MD, USA, October 9-12, 2005.

(Le Galès y Bungener, 2019) Le Galès, C., Bungener, M. The family accompaniment of persons with dementia seen through the lens of the capability approach, Dementia, 2019, 18 (1), 55-79.

(Lindberg et al., 2013) Lindberg B, Nilsson C, Zotterman D, Söderberg S, Skär L. Using Information and Communication Technology in Home Care for Communication between Patients, Family Members, and Healthcare Professionals: A Systematic Review, Int J Telemed Appl, 2013, no.461829.

(LinTO, 2021) LinTO, Open Source Business Voice Assistant. https://linto.ai/. (Último acceso: 2021)

(Liu & Latham, 2009) Liu CJ, Latham NK. (2009). Progressive resistance strength training for improving physical function in older adults. Cochrane Database Syst Rev. 2009; (3):CD002759. PMID: 19588334

(Lloyd-Sherlock, 2014) Lloyd-Sherlock P, Beard J, Minicuci N, Ebrahim S, Chatterji S. (2014). Hypertension among older adults in low- and middle-income countries: prevalence, awareness and control. Int J Epidemiol. 2014 Feb;43(1):116–28. doi: http://dx.doi.org/10.1093/ije/dyt215 PMID: 24505082

(Marschollek M., 2012) Marschollek, M. Decision support at home (DS@HOME) – system architectures and requirements. BMC Med Inform Decis Mak, 2012, 12:43

(Martin-Matthews et al., 2013) Martin-Matthews, A., Sims-Gould, J., & Tong, C. E. Canada's complex and fractionalized home care context: Perspectives of workers, elderly clients, family carers, and home care Managers1. Canadian Review of Social Policy, 2013, 68, 55-74.

(Mary, 2020) Página oficial de Mary. http://mary.dfki.de/. (Último acceso: 2020).

(Michel et al., 2008) Michel JP, Newton JL, Kirkwood TB. (2008). Medical challenges of improving the quality of a longer life. JAMA. 2008 Feb 13;299(6):688–90. doi: http://dx.doi.org/10.1001/jama.299.6.688 PMID: 18270358

(Mimic, 2020) Página oficial de Mimic. https://mimic.mycroft.ai/. (Último acceso: 2020).

(Modelo ASR, 2021) Carpeta de Google Drive con modelos en español para Mozilla DeepSpeech. https://drive.google.com/drive/folders/1-3UgQBtzEf8QcH2qc8TJHkUqCBp5BBmO. (Último acceso: 30-08-2021).

(Modelo TTS, 2021) Carpeta de Google Drive con modelos en español para Mozilla DeepSpeech. https://drive.google.com/drive/folders/1HxFUHQl6REh8CifOXL8IMlIyR9SDVcdu. (Último acceso: 30-08-2021).

(Mozilla TTS, 2021) Mozilla TTS, Github Oficial de Mozilla TTS. https://github.com/mozilla/TTS. (Último acceso: 2021)

(Mozilla Voice, 2021) Página oficial del proyecto Mozilla Voice. https://voice.mozilla.org/. (Último acceso: 2021).

(Mu-Hsing et al., 2016) Mu-Hsing Kuo, Shu-Lin Wang, Wei-Tu Chen. Using information and mobile technology improved elderly home care services, Health Policy and Technology, 2016, 5 (2), 131-142

(Murthy & Mani, 2013) Sudhir Rama Murthy & Monto Mani (2013). Discerning Rejection of Technology. SAGE Open 3(2) Follow journal. April 2013. DOI: 10.1177/2158244013485248

(Mycroft, 2021) Página oficial de Mycroft. https://mycroft.ai/. (Último acceso: 2021).

(Nginx, 2021) Nginx, página oficial. https://www.nginx.com/. (Último acceso: 01-09-2021).

(OMS, 2015) OMS. WHO global strategy on people-centred and integrated health services. Ginebra, 2015. https://www.who.int/servicedeliverysafety/areas/people-centred-care/global-strategy/en/.

(OMS, 2016) OMS. Informe de la Secretaría. Acción multisectorial para un envejecimiento saludable basado en el ciclo de vida: proyecto de estrategia y plan de acción mundiales sobre el envejecimiento y la salud. 69ª Asamblea Mundial de la Salud, Organización Mundial de la Salud, 2016. Link: https://apps.who.int/gb/ebwha/pdf_files/WHA69/A69_17-sp.pdf. (Último acceso: 17/05/2021).

(OMS, 2019) OMS. Alana Officer y Mary Manandhar (Coord.). Década del Envejecimiento Saludable 2020-2030. Envejecimiento y curso de vida, Organización Mundial de la Salud, 2019. Link: https://www.who.int/docs/default-source/documents/decade-of-health-ageing/decade-healthy-ageing-update1-es.pdf?sfvrsn=d9c40733_0. (Último acceso: 14/05/2021). Descripción: Primer informe de progreso, marzo 2019 del proyecto de Década del envejecimiento saludable de las Naciones Unidas. Define el concepto de envejecimiento saludable, su alineación con los ODS y la justificación de su conveniencia.

(ONPE, 2021) ONPE. Oficina Nacional de Prospectiva y Estrategia de la Presidencia del Gobierno. España 2050. Fundamentos y propuestas para una

Estrategia Nacional de Largo Plazo. https://www.lamoncloa.gob.es/presidente/actividades/Documents/2021/200521-Estrategia_Espana_2050.pdf. (Último acceso: 28/05/2021).

(Open Assistant, 2021) Open Assistant, Open Source Voice Assistant. https://github.com/openassistant/oa-core. (Último acceso: 2021)

(Paterson & Warburton, 2010) Paterson DH, Warburton DE. (2010). Physical activity and functional limitations in older adults: a systematic review related to Canada's Physical Activity Guidelines. Int J Behav Nutr Phys Act. 2010;7(1):38. doi: http://dx.doi.org/10.1186/1479-5868-7-38 PMID: 20459782

(Peláez, 2016) Peláez, I. M. (2016). Modelos de regresión: lineal simple y regresión logística. Revista Seden, 14, 195-214. https://www.revistaseden.org/files/14-cap%2014.pdf.

(Picking et al., 2012) Picking, R., Robinet, A., McGinn, J., Grout, V., Casas, R., and Blasco, R. (2012). "The Easyline+ Project: Evaluation of a User Interface Developed to Enhance Independent Living of Elderly and Disabled People," Universal Access in the Information Society (11:2), pp. 99-112.

(Pixel Ring, 2021) Pixel Ring. Repositorio oficial de git de la biblioteca pixel_ring. https://github.com/respeaker/pixel_ring. (Último acceso: 2021).

(Porcupine Benchmark, 2021) Porcupine Wake Word Engine benchmarking. "Yet Another Wake-Word Detection Engine". https://medium.com/@alirezakenarsarianhari/yet-another-wake-word-detection-engine-a2486d36d8d4. (Último acceso: 2021)

(Porcupine, 2021) Porcupine Wake Word Engine, by PicoVoice. https://picovoice.ai/platform/porcupine/. (Último acceso: 2021)

(Putty, 2021) Putty. Página oficial de Putty, un cliente de SSH para Windows. https://www.putty.org/. (Último acceso: 2021).

(Rasa Playground, 2020) Rasa Playground. Página oficial para probar modelos de conversación de Rasa. https://rasa.com/docs/rasa/playground. (Último acceso: 2021).

(Rasa, 2021) Rasa, página oficial. https://rasa.com/. (Último acceso: 2021)

(RaspberryPi OS, 2021) Página de descarga de sistemas operativos oficiales para Raspberry Pi. https://www.raspberrypi.org/software/operating-systems/. (Último acceso: 2021).

(Report, 2012) Report. Ageing in the twenty-first century: a celebration and a challenge. New York, London: United Nations Population Fund; HelpAge International; 2012.

http://www.unfpa.org/sites/default/files/pub-pdf/Ageing%20report.pdf. (Último acceso: 20/05/2021).

(ReSpeaker Guia, 2021) ReSpeaker Guia. Guía official para configurar un dispositivo ReSpeaker Core v2: https://wiki.seeedstudio.com/ReSpeaker_Core_v2.0/. (Último acceso: 2021).

(ReSpeaker OS, 2021) ReSpeaker OS. Página oficial de descargas de sistemas operativos Debian para el dispositivo ReSpeaker v2. http://respeaker.seeed.io/images/respeakerv2/debian/. (Último acceso: 2021).

(Roche, 2009) Roche, A. (2009). Arboles de decisión y series de tiempo. https://hdl.handle.net/20.500.12008/24343.

(Rojas, 2019) Rojas Claros, L. (2019). METODOLOGÍAS KANBAN Y SCRUM PARA EL DESARROLLO DE PROCESOS ÁGILES (Doctoral dissertation). http://hdl.handle.net/123456789/14414. (Último acceso: 2020).

(Rokach y Maimon, 2005) Rokach L., Maimon O. (2005) Clustering Methods. In: Maimon O., Rokach L. (eds) Data Mining and Knowledge Discovery Handbook. Springer, Boston, MA. https://doi.org/10.1007/0-387-25465-X_15.

(Saelens & Papadopoulos, 2008) Saelens BE, Papadopoulos C. (2008). The importance of the built environment in older adults' physical activity: a review of the literature. Wash State J Public Health Pract. 2008;1(1):13–21.

(Simon, 2020) Página oficial de Simon. https://simon.kde.org/. (Último acceso: 2020).

(Snowboy, 2021) Snowboy, Hotword detection engine. https://github.com/Kitt-AI/snowboy. (Último acceso: 2021)

(Soto, 2013) Soto, G. (2013). Análisis relacional entre usuarios de la tercera edad y servicios transaccionales digitales para la generación de guías de diseño. Tesis doctoral. Universidad del Desarrollo. Chile. 2013.

(Stephanie, 2021) Stephanie, Open Source Voice Assistant. https://slapbot.github.io/. (Último acceso: 2021)

(Su y Chiang, 2013) Su, C.-J.; Chiang, C.-Y. IAServ: An Intelligent Home Care Web Services Platform in a Cloud for Aging-in-Place. Int. J. Environ. Res. Public Health, 2013, 10, 6106-6130.

(Sutter y Müsseler, 2007) Sutter, C., and Müsseler, J. (2007). "User Specific Design of Interfaces and Interaction Techniques: What Do Older Computer Users Need?" UAHCI 2007 Proceedings, pp 1020-1029, Beijing, China.

(Tak et al., 2013) Tak E, Kuiper R, Chorus A, Hopman-Rock M. Prevention of onset and progression of basic ADL disability by physical activity in community dwelling older adults: a meta-analysis. Ageing Res Rev. 2013

Jan;12(1):329–38. doi: http://dx.doi.org/10.1016/j.arr.2012.10.001 PMID: 23063488

(Thingiverse ReSpeaker, 2021) Modelo 3D de la placa ReSpeaker Core V2 con sus proporciones ajustadas, y una propuesta de carcasa sencilla. https://www.thingiverse.com/thing:3065699. Último acceso: 06/09/2021.

(Trigás, 2012) Trigás Gallego, M. (2012). Metodologia scrum.

(W3C, 2020) W3C (2020). Older Users and Web Accessibility: Meeting the Needs of Ageing Web Users. Web Site. Available from: https://www.w3.org/WAI/older-users/.

(Wagner y Hunnerup. 2018) Wagner, S., Hunnerup, E. Ambient assisted living ecosystem for supporting senior citizens' human system interaction, Proceedings - 2018 11th International Conference on Human System Interaction, HIS, 2018, no. 8431357, 221-225.

(Wang et al., 2016) Wang, N., Broz, F., Di Nuovo, A., Belpaeme, T., and Cangelosi, A. (2016). “A User-Centric Design of Service Robots Speech Interface for the Elderly,” In Recent Advances in Nonlinear Speech Processing, pp. 275-283, Cham, Switzerland: Springer.

(Wave4, 2013) Wave4 (2013) Wave 4, release 1.1.1 (28 March 2013). In: Survey of Health, Ageing and Retirement in Europe (SHARE) [website]. Munich: Munich Center for the Economics of Aging; 2013. http://www.share-project.org/data-documentation/waves-overview/wave-4.html. (Último acceso: 27/05/2021).

(WESS, 2007) WESS (2007) World Economic and Social Survey 2007: development in an ageing world. New York: United Nations Department of Social and Economic Affairs; 2007 (Report No. E/2007/50/Rev.1 ST/ESA/314; http://www.un.org/en/development/desa/policy/wess/wess_archive/2007wess.pdf. (Último acceso: 20/5/2021).

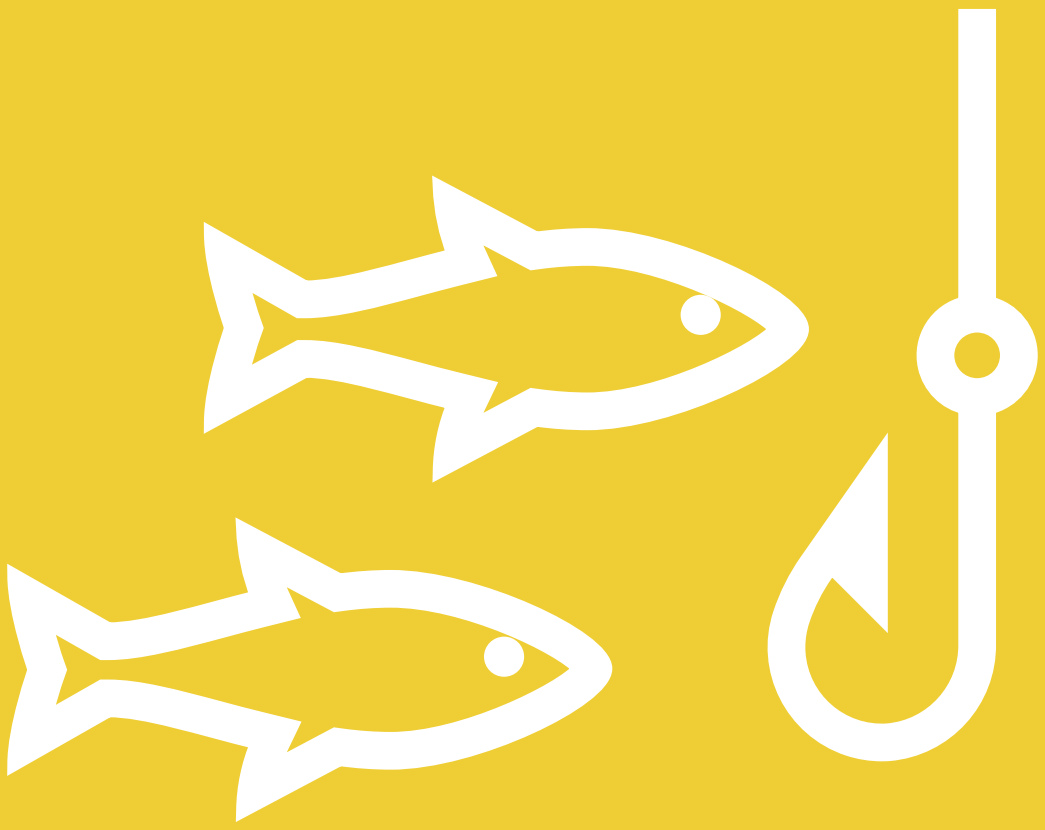
(Wilson, 2014) Wilson, C. (2014). “Cognitive Walkthrough,” In User Interface Inspection Methods: A User-Centered Design Method, Chapter 4, Morgan Kaufmann Publishers

(Wit.ai, 2021) Wit.ai, página oficial. https://wit.ai/. (Último acceso: 2021)

(YAML, 2021) YAML. Página oficial de YAML. https://yaml.org/. (Último acceso: 06/08/2021).

(Zajicek, 2004) Zajicek, M. (2004). “Successful and Available: Interface Design Exemplars for Older Users,” Interacting with Computers (16:3), pp. 411-430.

(Zueras et al., 2018) Zueras, Pilar, Jeroen Spijker, y Amand Blanes. Evolución del perfil de los cuidadores de personas de 65 y más años con discapacidad en la persistencia de un modelo de cuidado familiar. Revista Española de Geriatría y Gerontología 53, n.º 2, 2018. https://doi.org/10.1016/j.regg.2017.07.004.



Apéndices

12 Apéndices

12.1 METODOLOGÍA DE INVESTIGACIÓN PARA LA REVISIÓN BIBLIOGRÁFICA

Para el estado de la técnica se ha seguido una metodología estandarizada de búsqueda de información que se puede encontrar en (Biblioteca UA, 2013). Esta metodología consiste en cinco fases bien diferenciadas: análisis y definición de la necesidad de información; nivel y cobertura de la búsqueda; selección de las fuentes de información; elaboración de la estrategia de búsqueda; y, finalmente, valoración de los resultados y gestión de la información recuperada.

12.1.1 Análisis y definición de la necesidad de información

En la primera fase, **Análisis y definición de la necesidad de la información**, se deben detallar los objetivos de la búsqueda con precisión, con tal de tener claro qué es lo que queremos investigar. En este caso, todo el estado de la técnica gira entorno al envejecimiento saludable, las actuales soluciones tecnológicas, las tecnologías que podrían ayudar, y el estado y madurez de estas tecnologías. En la **Tabla 11** se muestra de forma resumida la concreción de estos parámetros.

12.1.2 Nivel y cobertura de la búsqueda

En la segunda fase, **Nivel y cobertura de la búsqueda**, hay que definir una serie de características que la información encontrada tendrá que cumplir para ser válida. Empezando por la **cobertura temporal**, que será totalmente abierta en temas de visión histórica o teóricos sobre el envejecimiento, pero sí con un límite de diez años en proyectos y tecnologías, ya que cualquier información anterior estaría potencialmente desactualizada u obsoleta. También se define la **cobertura geográfica**, que en este caso será principalmente centrada en España, aunque también se acepta información relativa a Europa y en última instancia de nivel global.

Tabla 11. Concreción de los parámetros análisis y definición de necesidades de información.
FUENTE: elaboración propia.

Sección Justificación del problema	1.	Identificar y establecer el ámbito de la actuación y que permita determinar los principales retos y oportunidades existentes. Este apartado se nutre ampliamente del propio estudio, por lo que las referencias bibliográficas emanarán en su gran mayoría del resto de los apartados.
Sección Envejecimiento Saludable	2.	Conocer la situación actual y las proyecciones más probables acerca del cambio demográfico ocasionado por el envejecimiento de la población, identificando los restos más importantes y los nichos de actuación prioritarios.
Sección Tecnologías	3.	Conocer en qué consisten las interfaces de usuario, qué tipos y dispositivos hay, qué implicaciones tienen en el diseño de sistemas y servicios para el cuidado y la atención a los mayores, y qué tecnologías están asociadas, especialmente las tecnologías de reconocimiento del lenguaje natural, ejemplos de uso, implicación de organismos y empresas en su creación, normativas y estándares para la prestación de servicios.
Sección Inteligencia Artificial	4.	Presentar a grandes rasgos el concepto de inteligencia artificial, e identificar las principales ramas que la forman junto con la relación existente entre ellas. Dentro del campo de la inteligencia artificial, se analizará con especial atención la rama dedicada al reconocimiento del lenguaje natural, nuevamente identificando sus principales subramas y la relación entre las mismas y con otras ramas afines. Conocer qué tecnologías y marcos de desarrollo existen, qué características tienen y cómo pueden ayudar en el desarrollo de proyectos, aplicaciones, dispositivos y servicios para la atención a los mayores.

Además, hay que hablar de la **tipología documental**, en la que definimos características de los documentos a elegir. En este caso nos vale cualquier tipo de documento siempre y cuando sean de origen fiable. El formato del documento también nos es irrelevante, pero se prefiere PDF por ser el más habitual en publicación y difusión de documentos. En la **Tabla 12** se resumen los niveles y la cobertura de la búsqueda realizada.

12.1.3 Selección de las fuentes de información

En la tercera fase, **Selección de las fuentes de información**, elegimos qué fuentes se consideran válidas para la investigación. Nosotros hemos seleccionado fuentes donde se puedan encontrar publicaciones relacionadas con el contexto del trabajo (Google, Google Scholar, Web of Science, Scopus), además de boletines oficiales de gobiernos tanto nacionales, como europeos, como internacionales. Como requisito, estas fuentes deben ser de dominio público o de acceso a investigadores como nosotros (con acuerdos con la UA que permiten acceder a estos documentos). En la **Tabla 13** se muestra de forma resumida la concreción de estos parámetros.

Tabla 12. Definición del nivel y de la cobertura de la búsqueda realizada.
FUENTE: elaboración propia.

Cobertura Temporal	Un periodo temporal no superior a los 10 años en lo que respecta a proyectos, sistemas, tecnologías y plataformas, y no marcaremos límites sobre aspectos teóricos o de base o cuando se requiera proporcionar una visión histórica.
Cobertura Geográfica	Nos centraremos en información que tenga un origen nacional o en el marco europeo en primera instancia, pero valoraremos también información que tenga una proyección internacional de calado.
Tipología documental	Documentos técnicos, artículos científicos, libros de editoriales reconocidas, manuales tecnológicos, normativas y regulaciones oficiales de cualquier nivel, informes emitidos por órganos gubernamentales y artículos de difusión en medios de reconocido prestigio del entorno TIC y científico.

Tabla 13. Definición de las fuentes de información seleccionadas para la búsqueda.
FUENTE: elaboración propia.

Fuentes	Google, Google Scholar, Web of Science y Scopus

12.1.4 Elaboración de la estrategia de búsqueda

En la cuarta fase, **Elaboración de la estrategia de búsqueda**, elegimos palabras clave y relaciones entre ellas que permitirán seleccionar documentos en las fuentes encontradas (**Figura 38**).

En general, realizaremos búsquedas simples, en campos de tipo texto general. En la **Tabla 14** se recogen todos los términos de búsqueda empleados, organizados por capítulos.

12.1.5 Valoración de los resultados y gestión de la información recuperada

En la quinta fase, **Valoración de los resultados y gestión de la información recuperada**, seleccionamos los documentos encontrados en la anterior búsqueda que realmente hablen de lo que queremos investigar. Además, guardamos las referencias para que el que lea este documento pueda acceder o revisar las fuentes por sí mismo. En este documento las referencias se pueden encontrar en la **sección 11. Referencias**.

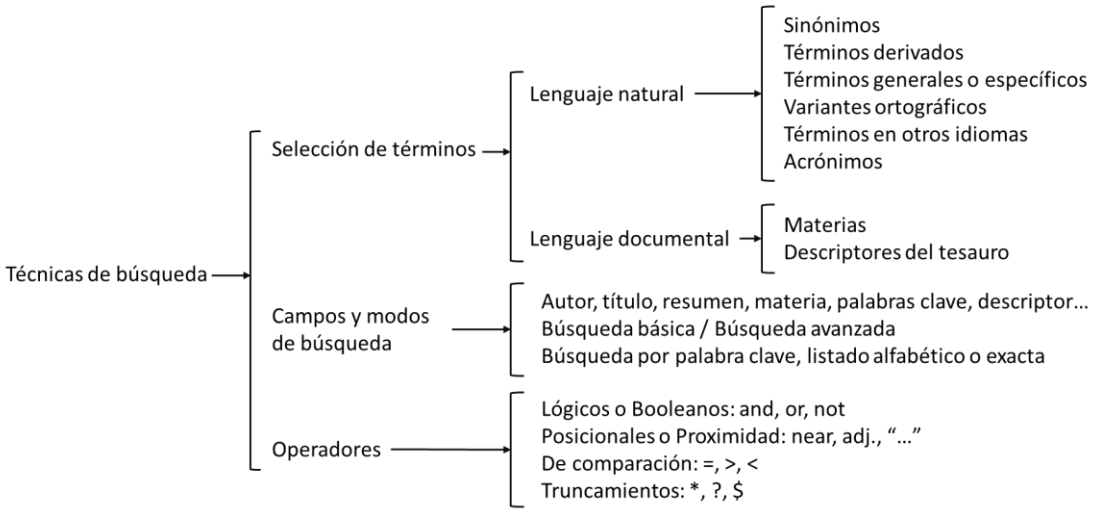


Figura 38. Esquema de toma de decisión sobre los elementos a determinar en las técnicas de búsqueda.
FUENTE: (Biblioteca UA, 2021).

Tabla 14. Concreción de los parámetros estrategia de búsqueda.
FUENTE: elaboración propia.

Sección 1. Introducción	("adulto" + "mayor" or "senior" + "citizen" or "elderly" or "aging" or "ageing" or "envejecimiento") and ((("cambios" or "evolución" or "retos") and ("demográficos" or "población" or "sociedad" or "cuidados" or "sostenible" or "saludable") and ("OMS" or "ONPE" or "CE" or "proyectos europeos" or "servicios")) or (("changes" or "evolution" or "challenges") and ("demographic" or "population" or "society" or "care" or "sustainable" or "healthy") and ("WHO" or "ONPE" or "EC" or "European projects" or "services"))) and ("servicio asitencial" or "care service" or "vida asistida" or "assisted living" or "vida activa" or "active living" or "AAL").
Sección 2. Envejecimiento Saludable	("adulto" and "mayor" or "senior" and "citizen" or "elderly" or "aging" or "ageing" or "envejecimiento") and ((("cambios" or "evolución" or "retos") and ("demográficos" or "población" or "sociedad" or "cuidados" or "sostenible" or "saludable") and ("OMS" or "ONPE" or "CE" or "proyectos europeos" or "servicios")) or (("changes" or "evolution" or "challenges") and ("demographic" or "population" or "society" or "care" or "sustainable" or "healthy") and ("WHO" or "ONPE" or "EC" or "European projects" or "services"))) and ("servicio asitencial" or "care service" or "vida asistida" or "assisted living" or "vida activa" or "active living" or "AAL").
Sección 3. Tecnologías	("adulto" + "mayor" or "senior" + "citizen" or "elderly" or "aging" or "ageing" or "envejecimiento") and

	((("interfaz de usuario" or " interfaz gráfica de usuario" or "asistente virtual, asistente de voz, altavoz inteligente, dispositivo inteligente" or "robots de servicio" or " IA" or "PLN" or "redes de interconexión") and ("definición" or "sistemas comerciales" or "plataformas abiertas" or "código abierto" or "hardware abierto")) AND (("user interface" or "GUI" or "graphical user interface" or "VUI" or "virtual assistant, voice assistant, smart speaker, smart device" or "service robots" or "IoT device" or "IA" or "PLN" or "interconnection networks") and ("definition" or "commercial systems" or "open platforms" or "open source" or "open hardware"))) and (("servicio" and "tecnología" and "plataforma") or ("service" and "technology" and "framework")) and ("servicio asitencial" or "care service" or "vida asistida" or "assisted living" or "vida activa" or "active living" or "AAL").
Sección 4. Inteligencia Artificial	("adulto" and "mayor" or "senior" and "citizen" or "elderly" or "aging" or "ageing" or "envejecimiento") and ((("inteligencia artificial" or "IA" or "procesamiento de lenguaje natural" or "PLN" or "aprendizaje profundo" or "tecnologías de voz") and ("aplicaciones" or "plataformas") and ("comparativas" or "definiciones" or "fabricantes")) AND ("artificial intelligence" or "AI" or "natural language processing" or "NLP" or "deep learning" or "speech technologies") and ("applications" or "platforms") and ("benchmarks" or "definitions" or "manufacturers"))) and (("servicio" and "tecnología" and "plataforma") or ("service" and "technology" and "framework")) and ("servicio asitencial" or "care service" or "vida asistida" or "assisted living" or "vida activa" or "active living" or "AAL").

12.2 RASPBERRY PI CON UN ARREGLO DE 4 MICRÓFONOS

Prototipo basado en la placa *ReSpeaker 4-Mic Array* de la compañía *Seeed*. Esta placa proporciona un arreglo de 4 micrófonos gestionado por el chip conversor analógico digital AC108. También proporciona un arreglo de 12 leds y está especialmente diseñado para conectarse a una placa de desarrollo Raspberry Pi 3, por lo que resulta ideal para lograr un prototipo de SAM Device, rápido y fiable, con posibilidades de ampliación y, lo más importante, basado en hardware abierto.

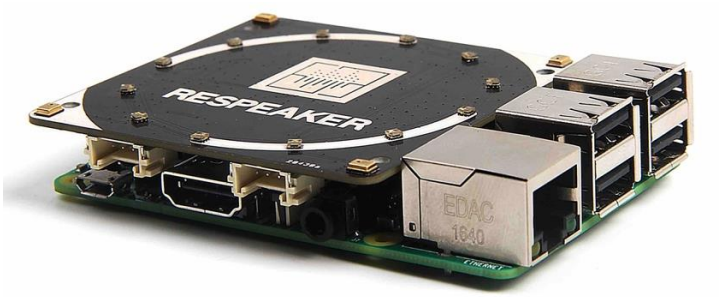


Figura 39. Raspberry Pi y placa ReSpeaker 4-Mic Array de Seeed.
FUENTE: wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/.

12.2.1 Componentes principales de ReSpeaker 4-MIC Array

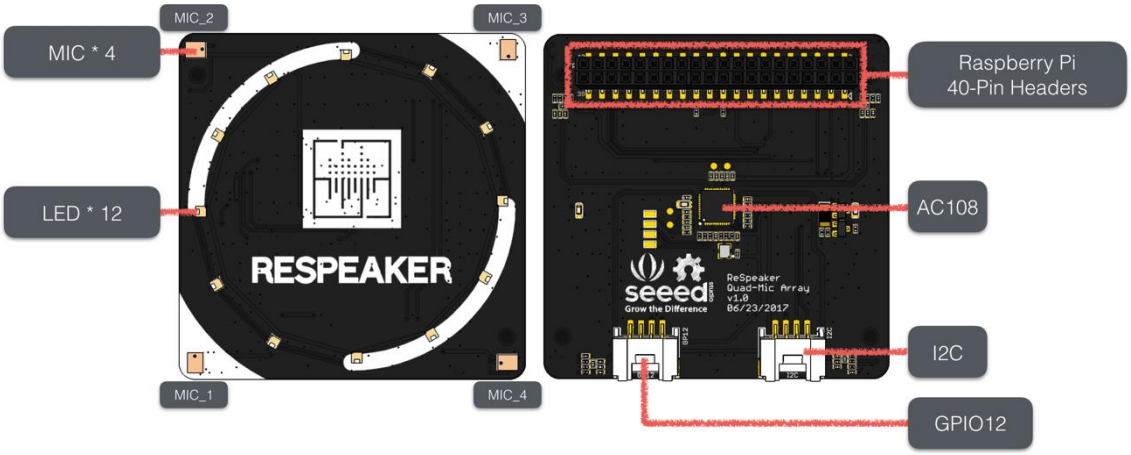


Figura 40. Detalle de componentes de la placa de micrófonos ReSpeaker 4—Mic Array.
FUENTE: wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/.

12.2.1.1 Chip ADC para arreglos de micrófonos (AC108)

El chip de audio AC108 es un chip conversor analógico a digital diseñado específicamente para arreglos de micrófonos con soporte para 4 micrófonos y una interfaz de salida I2C + I2S para el

procesador que actúa de equipo anfitrión (host). Las matrices de micrófonos son particularmente útiles para altavoces inteligentes, y especialmente para la detección de palabras de activación (Hotwords) para la detección de actividad de voz, ya que las configuraciones de un solo micrófono pueden tener problemas para detectar *Hotwords* cortas como "Oye Sam" o “OK Google” en entornos ruidosos: al reproducir música, sonar una alarma, etc.

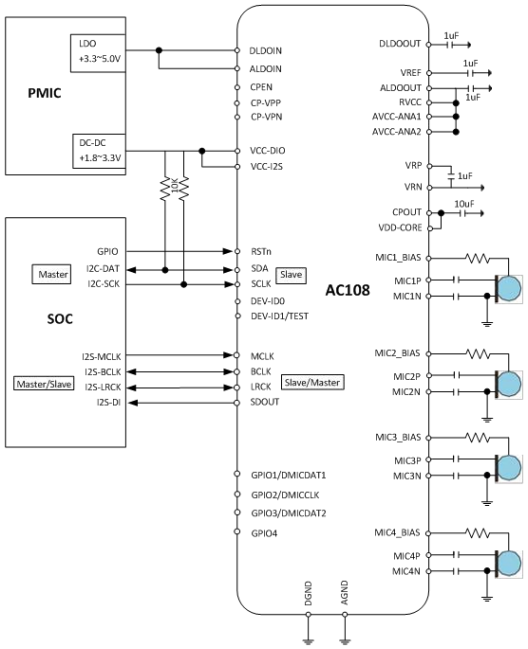


Figura 41. Diagrama con los esquemas de interconexión del chip AC108 y los micrófonos.
FUENTE: www.cnx-software.com/2017/08/28/x-powers-ac108-is-a-quad-channel-adc-chip-for-microphone-arrays/.

12.2.1.2 Arreglo de Leds

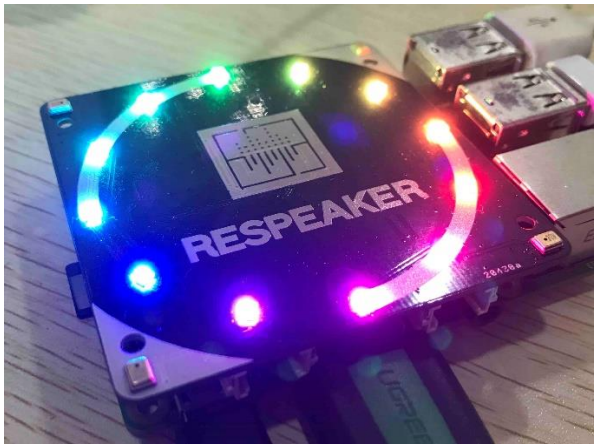


Figura 42. Detalle de funcionamiento de los leds de la placa ReSpeaker 4-MIC.
FUENTE: wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/.

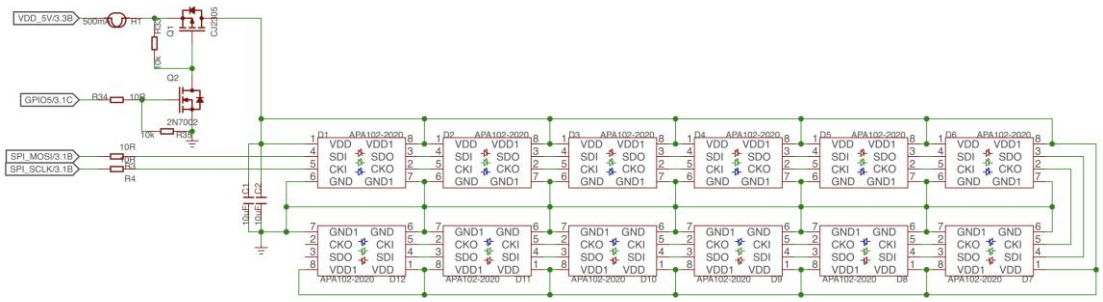


Figura 43. Diagrama con el esquema de conexión del arreglo de leds.
FUENTE: files.seeedstudio.com/wiki/ReSpeaker-4-Mic-Array-for-Raspberry-Pi/src/ReSpeaker%204-Mic%20Array%20for%20Raspberry%20Pi%20%20v1.0.pdf.

12.2.1.3 Conectores de ReSpeaker 4-Mic Array: Raspberry Pi, Grove

La placa está preparada para conectar fácilmente con una placa Raspberry Pi de desarrollo a través de un conector de 40 pines. También se han planteado dos conectores adicionales para conectar dispositivos compatibles I2C o que puedan ser controlados mediante salidas digitales (GPIO). En la **Figura 44** se puede ver un detalle de los conectores y en la **Figura 45** se muestra un extracto de los esquemas eléctricos de conexión.

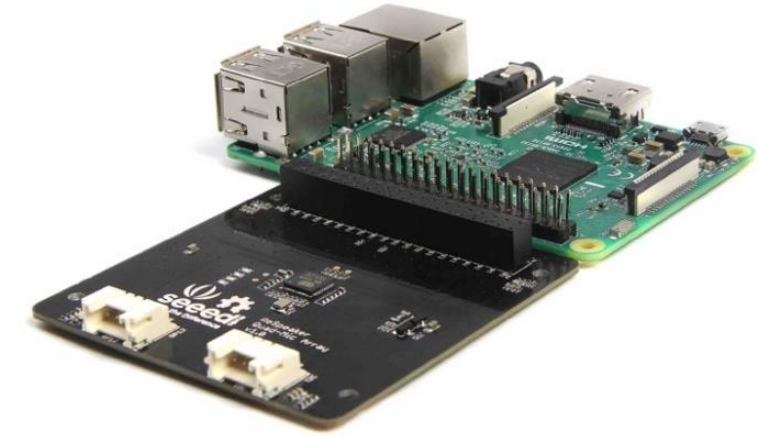


Figura 44. Detalle de los conectores de la ReSpeaker 4-Mic Array junto con una Raspberry Pi.
FUENTE: wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/.

12.3 Raspberry Pi + Arreglo de 2 Micrófonos

Este prototipo es uno de los más sencillos desde el punto de vista de los componentes más específicos que requiere el diseño de nuestro dispositivo. Sin embargo, tiene la ventaja de que se pueden adquirir todos los componentes de golpe en un Kit que se denomina *Snips Voice Interaction Base Kit* de la casa *Seeed*.

Dados los problemas de abastecimiento causados por el estado de pandemia, ha resultado ser un gran aliado. Además, esta versión incorpora una serie de componentes adicionales que la hacen atractiva para explorar funcionalidades añadidas y llevar el prototipo de asistente de voz más hacia un *dispositivo IoT* más genérico, incorporando sensores y actuadores.

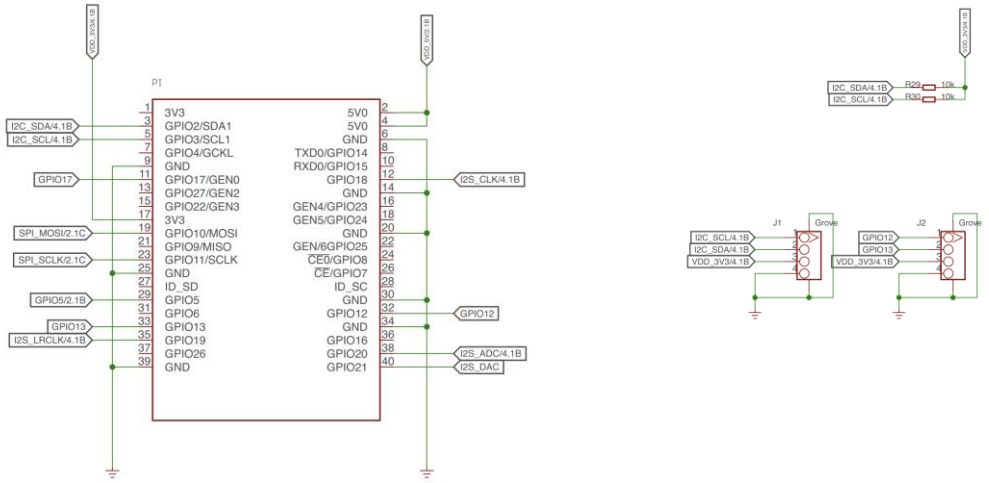


Figura 45. Diagrama con los esquemas de interconexión de los conectores y buses de la placa ReSpeaker 4-Mic Array. FUENTE: files.seeedstudio.com/wiki/ReSpeaker-4-Mic-Array-for-Raspberry-Pi/src/ReSpeaker%204-Mic%20Array%20for%20Raspberry%20Pi%20%20v1.0.pdf.

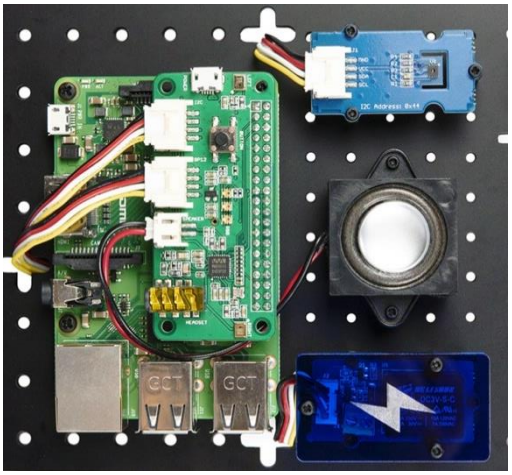


Figura 46. Prototipo con dos micrófonos, 4 leds, altavoz, sensor de temperatura y humedad, y relé. FUENTE: wiki.seeedstudio.com/Snips_Voice_Interaction_Base_Kit/.

Este prototipo constituye una versión más simple del prototipo presentado en el aparado anterior y se basa en un pequeño arreglo de dos micrófonos gestionado por la placa *ReSpeaker 2-Mics Pi HAT* de la compañía *Seeed*.

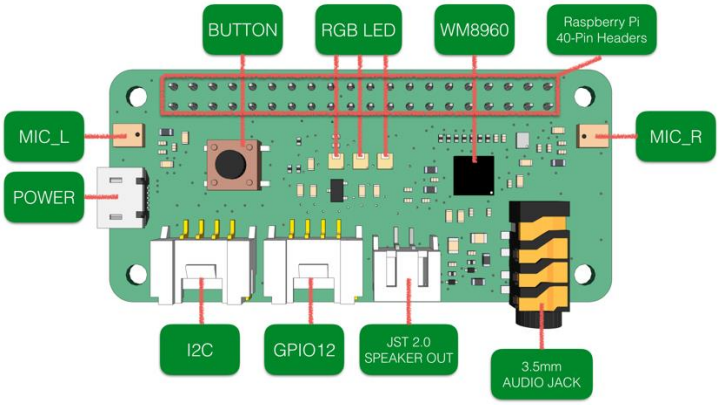


Figura 47. ReSpeaker 2-Mics Pi HAT.
FUENTE: wiki.seeedstudio.com/ReSpeaker_2_Mics_Pi_HAT/

ReSpeaker 2-Mics Pi HAT es una placa de expansión de doble micrófono para *Raspberry Pi* diseñada para aplicaciones de voz e inteligencia artificial.

La placa está desarrollada en base al circuito integrado *WM8960*, un códec estéreo de baja potencia. Hay 2 micrófonos en ambos lados de la placa para recopilar sonidos y también proporciona 3 LED APA102 RGB, 1 botón de usuario y 2 interfaces Grove integradas para expandir su funcionalidad. Adicionalmente cuenta con un conector de audio de 3,5 mm y una salida de altavoz JST 2.0 disponibles para la salida de audio.

En este prototipo se han incorporado, a modo de ejemplo, un par de sensores (de temperatura y humedad) y un actuador (un relé) que puede servir para activar algún tipo de electrodoméstico, lámpara, etc. del hogar.

12.4 RESPEAKER CORE V2

Se trata de una placa en la que básicamente se han integrado todos los componentes que necesita SAM Device y que ya han sido analizados en los prototipos que se ha mostrado anteriormente. La integración se realiza mediante un sistema basado en chip (SoC –System on Chip) que permite un diseño muy compacto.

Figura 48. Placa ReSpeaker Core V2.
FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/

Seeed ReSpeaker Core v2 está diseñado para aplicaciones interactivas de voz. Se basa en ARM Cortex-A7 de cuatro núcleos, hasta 1,5 Ghz y 1 GB de RAM en la placa. Además, cuenta con seis conjuntos de micrófonos y con una biblioteca de algoritmos de voz, como DoA (*Direction of Arrival* o dirección de llegada), BF (*Beam-Forming* o formación de haz), AEC (*Acoustic Echo Cancellation* o cancelación de eco acústico), etc.

ReSpeaker Core v2 ejecuta el sistema operativo GNU/Linux. Gracias a una comunidad fuerte y activa, se puede usar una gran cantidad de software y herramientas existentes para desarrollo, prueba e implementación, facilitando la creación de productos. No solo está diseñado para realizar prototipos, también puede ser una solución llave en mano para empresas comerciales. El hardware consta de dos partes, una es el módulo SoC minimizado que es pequeño y fácil de fabricar y está listo para el producto final, la otra es una placa inferior que puede ser completamente personalizable.

12.4.1 Características

- High performance SoC
- 1GB RAM & 4GB eMMC
- 6 Microphone Array
- USB OTG, USB device
- WiFi b/g/n and BLE 4.0
- Detect range: ~5 meters
- Grove socket for other sensor
- 3.5mm audio jack & JST2.0 connector
- 8 channel ADCs for 6 microphone array and 2 loopback (hardware loopback
- Debian-based Linux system
- SDK for speech algorithm with Full documents
- C++ SDK and Python w rapper
- Speech algorithms and features
- Keyword wake-up
- BF (Beam-Forming)
- DoA (Direction of arrival)
- NS (Noise suppression)
- AEC (Acoustic echo cancellation) and AGC (Automatic gain control)

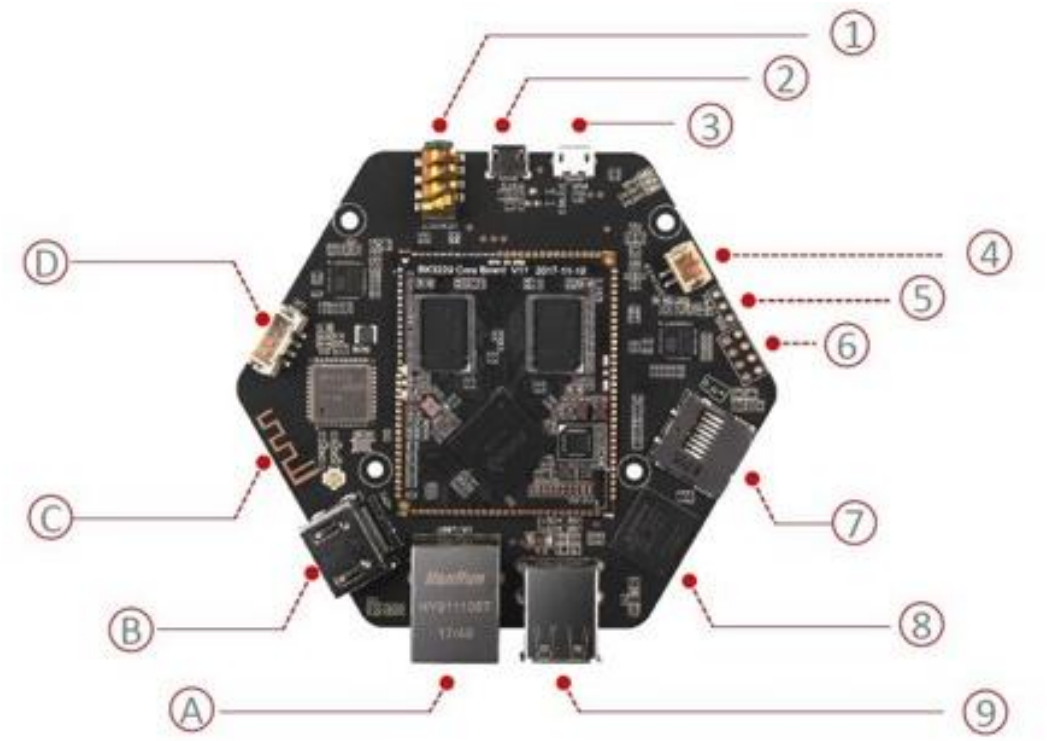
12.4.2 Especificaciones

Tabla 15. Lista con las características principales de la placa ReSpeaker Core V2.
FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/

Features		
Soc(Rockchip RK3229)	CPU	Quad-Core Cortex-A7, up to 1.5GHz
	GPU	Mali400MP, Support OpenGL ES1.1/2.0
	Memory	1GB RAM (Core Module includes RAM and PMU)
	System	Operating Voltage:3.6-5V
		80 pins on-module
		PMU on-module
Peripheral	Networks	WiFi b/g/n; BLE 4.0; Ethernet
	USB	2 x USB Host; 1 x USB OTG; 1 x USB power
	Grove	1 x Grove socket (I2C and Digital)
	Vedio	HDMI 2.0 with HDCP 1.4/2.2, up to 4K/60Hz
	Audio	6 Microphone Array; 3.5mm Audio Jack; JST2.0 Audio output connector
	Storage	4GB eMMC on-board; SD slot
	Others	12 x RGB leds; 8 GPIO pins
Power Consumption	Standby Mode	360mA

12.4.3 Descripción general del hardware

Interfaz y almacenamiento



① **3.5mm Headphone jack**: Output audio. You can plug active speakers or Headphones into this port.

② **USB OTG**: This USB Port is used to connect to your computer via serial mode of putty (or other serial tools).

③ **USB Power Input**: This port is used to provide pow er for ReSpeaker Core v2.

④ **Speaker Jack**: Output audio for passive speakers. Jst 2.0 Socket.

⑤ **UART**: You also can connect the ReSpeaker Core v2 with your computer via this UART port.

⑥ **8 Pins GPIO**: General Purpose Input Output interface for extended applications.

⑦ **SD Card Slot**: To plug in micro-SD card.

⑧ **eMMC**: Embedded Multimedia Card. You can burn the image into eMMC, so that the ReSpeaker Core v2 can boot from the eMMC.

⑨ **USB Host**: You can plug USB device, such as USB mouse, USB keyboard and USB flash disk into ReSpeaker Core v2 via those two USB hosts.

Ⓐ **Ethernet**: Access to the Internet.

Ⓑ **HDMI**: Output video.

Ⓒ **Bluetooth and WIFI Antenna**: The onboard Antenna is for WIFI and Bluetooth. Also, we provide an interface for 2.4G Antenna or PCB Antenna.

(D) **Grove Socket**: Grove Socket for digital or I2C.

12.4.4 Pines de salida (*pin out*)

Definición del índice de los pines de salida de los conectores de la placa ReSpeaker Core V2.

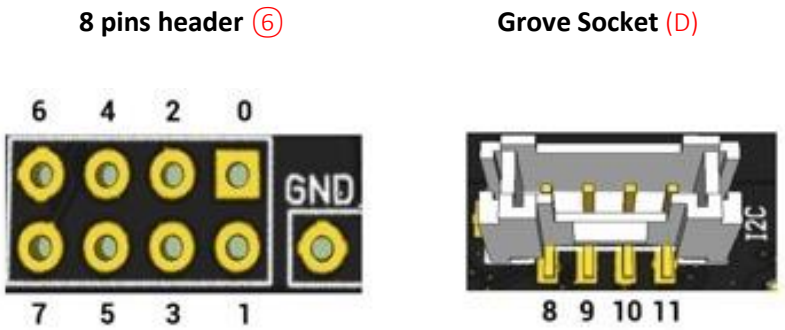


Figura 49. Detalle de los conectores de expansión que proporciona la placa ReSpeaker Core V2 y numeración de sus pines de salida.
FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/.

Tabla 16. ReSpeaker Core V2 **GPIO Pins**.
FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/.

MRAA	HEADER PIN INDEX	SYSFS PIN	RK3229 PIN
0	0	1091	GPIO2_D3
1	1	--	VCC
2	2	1043	GPIO1_B3
3	3	1127	GPIO3_D7
4	4	1017	GPIO0_C1
5	5	1067	GPIO2_A3
6	6	--	GND
7	7	1013	GPIO0_B5
8	8	1085	GPIO2_C5
9	9	1084	GPIO2_C4
10	10	--	VCC
11	11	--	GND

Tabla 17. ReSpeaker Core V2 **I2C Pins**.
FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/.

MRAA	HEADER PIN INDEX	SYSFS PIN	RK3229 PIN
0	8	--	I2C2_SCL
0	9	--	I2C2_SDA

12.4.5 Dimensiones

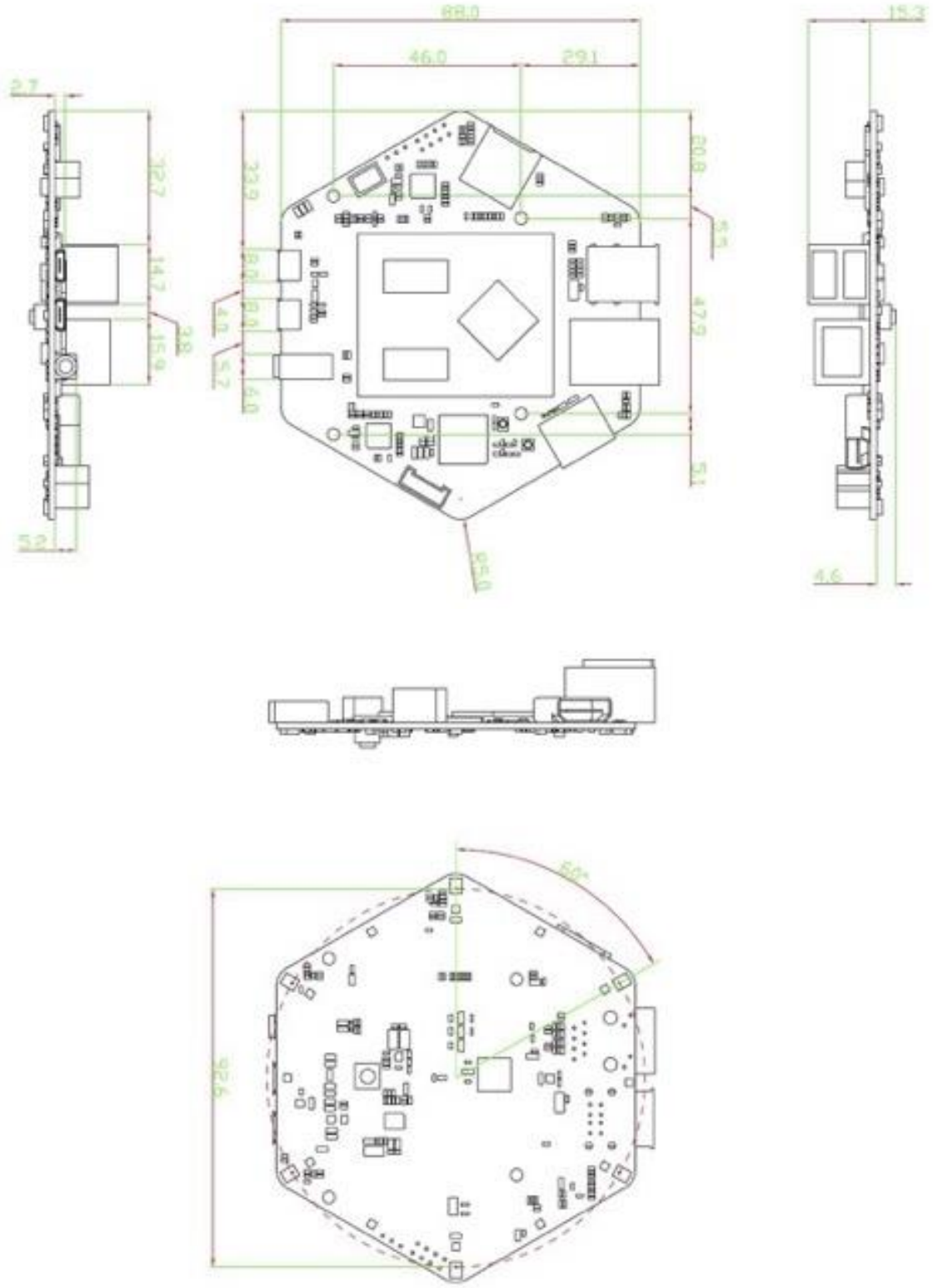


Figura 50. Dimensiones de la placa ReSpeaker Core V2.
FUENTE: wiki.seeedstudio.com/ReSpeaker_Core_v2.0/.

12.5 GLOSARIO DE ACRÓNIMOS Y TÉRMINOS

El glosario se divide en dos grandes secciones. Primero el glosario de acrónimos y después el glosario de términos.

12.5.1 Glosario de acrónimos

Este glosario contiene los acrónimos que aparecen en el trabajo junto a su expansión, y su traducción al español o al inglés, además de su acrónimo en el otro idioma si se utiliza en el ámbito general. En la siguiente tabla se muestran en negrita los acrónimos utilizados en el texto.

Tabla 18. Tabla de acrónimos.
FUENTE: elaboración propia.

Acr (EN)	Descripción (EN)	Acr (ES)	Descripción (ES)
-	-	**ABP**	Aprendizaje Basado en Proyectos
-	-	**ONPE**	Oficina Nacional de Prospectiva y Estrategia
-	-	**SAM**	Servicio de Acompañamiento a Mayores
-	-	**TFG**	Trabajo de Fin de Grado
-	-	**UA**	Universidad de Alicante
AAL	Ambient Assisted Living	-	Vida asistida por el entorno
AAL-2	Active and Assisted Living	-	Vida asistida y activa
AEC	Acoustic Echo Cancellation	-	Cancelación del eco acústico
AGC	Automatic Gain Control	-	Control de ganancia automática
AI	Artificial Intelligence	**IA**	Inteligencia Artificial
API	Application Programming Interface	-	Interfaz de programación de aplicaciones
ARM	Advanced RISC Machine	-	Máquina RISC avanzada
ASR	Automatic Speech Recognition	-	Reconocimiento automático del habla
BF	Beam Forming	-	Matriz de haces
BPMN	Business Model Process and Notation	-	-
CDD	Conversation Driven Development		
CNN	Convolutional Neural Network	-	Red neuronal convolucional
CPU	Central Processing Unit	-	Unidad de procesamiento central
DCC	Digital Cities Challenge	-	Reto de ciudades digitales
DL	Deep Learning	AP	Aprendizaje Profundo
DME	Dialogue Move Engine	-	Motor de gestión del diálogo
DNN	Deep Neural Network	-	Red neuronal profunda

DoA	Direction of Arrival	-	Reconocimiento de la dirección de llegada
eMMC	embedded Multi-Media Card	-	Tarjeta multimedia embebida
EU	European Union	**UE**	Unión Europea
GPIO	General Purpose Input / Output	-	Entrada / Salida de propósito general
GPU	Graphics Processing Unit	-	Unidad de procesamiento gráfico
GUI	Graphical User Interface	-	Interfaz gráfica de usuario
HDMI	High-Definition Multimedia Interface	-	Interfaz multimedia de alta definición
HTTP	Hypertext Transfer Protocol	-	Protocolo de transferencia de hipertexto
I2C	Inter-Integrated Circuit	-	-
ICC	Intelligent Cities Challenge	-	Reto de ciudades inteligentes
IoT	Internet of Things	-	Internet de las cosas
IT	Information Technology	**TI**	Tecnologías de la Información
LoRa	Long Range	-	-
ML	Machine Learning	AA	Aprendizaje Automático
NLG	Natural Language Generation	-	Generación de lenguaje natural
NLP	Natural Language Processing	-	Procesamiento del lenguaje natural
NLU	Natural Language Understanding	-	Compresión del lenguaje natural
NN	Neural Network	RNA	Red Neuronal Artificial
NS	Noise Suppression	-	Supresión de ruido
PCB	Printed Circuit Board	-	Tarjeta de circuito impreso
PLC	Power-Line Communication	-	Comunicación mediante línea de potencia
RISC	Reduced Instruction Set Computer	-	Computador con conjunto de instrucciones reducido
SaaS	Software as a Service	-	Software como servicio
SDK	Software Development Kit	-	Kit de desarrollo de software
SoC	System on a Chip	-	Sistema en chip
STT	Speech to Text	-	Conversión de voz a texto
SWOT	Strengths, Weaknesses, Opportunities y Threats	**DAFO**	Debilidades, Amenazas, Fortalezas, Oportunidades
TTS	Text to Speech	-	Conversión de texto a voz
USB	Universal Serial Bus	-	Bus serie universal
VAD	Voice Activity Detection	-	Detección de actividad de voz
VUI	Voice User Interface	-	Interfaz de usuario basado en voz
WHO	World Health Organization	**OMS**	Organización Mundial de la Salud

12.5.2 Glosario de términos

El glosario de términos recoge las palabras más técnicas o menos comunes del documento y les da una breve explicación. Esta explicación no trata de ser una definición formal y académica de dicho término, si no que actúa de descripción divulgativa y siempre enfocada en el contexto en el que se utiliza en este documento, omitiendo detalles que, aunque sean importantes, no afectan ni influyen en el uso que les hemos dado ni en su entendimiento en el trabajo.

Se recogen las palabras del documento que no tienen explicación en el texto porque su significado se asume de carácter general en este ámbito o porque explicarlas en mitad del texto puede desviar la atención de lo importante, haciéndolo innecesariamente enrevesado. Otras simplemente se explican de nuevo para recapitular.

Tabla 19. Tabla de términos.
FUENTE: elaboración propia.

Término	Descripción
Acompañamiento	Como su propio nombre indica, es el acto de acompañar. En este contexto se refiere a la acción de acompañamiento que se realiza con los mayores, al estar con ellos, cuidarlos y atender sus necesidades.
BackOffice	Conjunto de actividades que se llevan a cabo de manera ajena al usuario. En este caso abarca los procesos de SAM Service que se ejecutan de cara a administradores y en general, los no usuarios.
Balanceador de carga	Sistema que controla el acceso a un servicio software distribuido en distintas instancias del mismo programa que envía las peticiones a una instancia u otra en función de la carga de cada una, tratando de no dejar a ninguna sin trabajo ni demasiado cargada.
Clúster	En este contexto, conjunto de máquinas trabajando juntas como una sola para formar una sola máquina muy potente, generalmente permitiendo añadir o quitar hardware variando así la potencia del conjunto, haciéndolo muy escalable.
Contenedor	En este contexto, es una versión simplificada de una máquina virtual para un programa o aplicación específicos, que contiene solo el núcleo del sistema operativo que dicho programa necesita para funcionar y las dependencias,

	permitiendo ejecutar el programa en cualquier sitio de manera más ligera que una máquina virtual.
Dashboard	Traducido literalmente como cuadro de mando, se trata de una aplicación generalmente web que permite visualizar fácilmente datos y estadísticas sobre un proceso determinado (en este caso, el acompañamiento a mayores), así como realizar acciones y consultas sobre dichos datos.
Docker	Biblioteca para la creación y gestión de contenedores.
Front-End	Se trata de la parte de una aplicación software que da la cara al usuario y se ejecuta en su máquina, generalmente incluyendo la interfaz y lógica de negocio básica o nula. Es la contraparte del *backend* (la parte que se ejecuta en el servidor).
Hardware	Pieza física de maquinaria, como nuestro dispositivo IoT.
Hotword	Palabra de activación. Es la frase que tienes que decir en voz alta para activar a un asistente de voz. Por ejemplo, “Oye SAM” en este Proyecto o “Alexa” con Amazon Alexa.
HTTP	Protocolo de comunicación a nivel de aplicación. Ampliamente utilizado en internet, en este trabajo se menciona para peticiones que nuestro dispositivo IoT realiza a través de internet.
HTTPS	Versión segura de HTTP. Cuando se menciona HTTP en este trabajo, se utiliza de manera genérica y no descarta el uso de esta versión.
Ingress	Es una biblioteca que actúa de balanceador de carga.
Intent	Traducido literalmente como intención, es la idea clave que se extrae de la frase del usuario para identificar cuál es su intención (como consultar una cita médica o poner música).
Kubernetes	Plataforma para la gestión de clústeres con aplicaciones en contenedores.
Microservicios	Como su propio nombre indica, es un servicio pequeño, generalmente pensado para actuar en conjunto con otros servicios o servir de apoyo a un servicio mayor.
Nginx	Proxy inverso de *Ingress*.

Open Source	Hardware o software cuya información sobre cómo está hecho es de carácter público y cualquiera puede verlo e incluso replicarlo y modificarlo. En software suele ser código y en hardware los esquemas. En español suele traducirse como libre o abierto.
Proxy	Programa que dirige tus peticiones a la web con una ip pública apta para internet. Actúa a nivel de aplicación.
Proxy inverso	Programa que recibe peticiones web, y las redirige dentro de su red local al lugar que correspondan. Su utilidad es hacer la lógica de despliegue transparente al usuario ofreciendo un punto de acceso centralizado.
Servicio Web/ Web Service	Servicio que se ofrece a través de la web. Puede tener una infinidad de utilidades, desde páginas web para humanos hasta puntos de acceso de datos para otras máquinas.
Software	Pieza no física de maquinaria, compuesta solo por datos. Esta descripción incluye todo tipo de programas informáticos, aplicaciones, código y datos. Si no lo puedes tocar, es software.
Streaming	Acción de mover datos en un flujo (*stream*), es decir, sin que se paren, permitiendo al receptor de los datos empezar a utilizar los datos conforme llegan los primeros bits, sin tener que esperar a que llegue el mensaje completo. Muy utilizado para ver series en plataformas como Netflix, o para reproducir música en línea.
Unidad flash	Unidad de almacenamiento pequeña y portable.
WAV	Proveniente de *Waveform*, es un formato de audio de Microsoft con compresión sin pérdida de calidad (esto es, pesa más pero el archivo queda intacto). El códec es abierto y por ello nos viene bien para el proyecto.
Wearable	Dispositivo hardware que se lleva puesto, como un reloj inteligente.
Web Socket	Tecnología de comunicación que permite la comunicación bidireccional entre los dos comunicados.

12.6 BIBLIOTECAS, HERRAMIENTAS Y LENGUAJES UTILIZADOS

Este apéndice recoge todas las herramientas, bibliotecas y lenguajes utilizados en el proyecto, además de un breve resumen de para qué se han utilizado, y enlaces de interés.

Aunque se ha tratado de dejar su uso explícito a lo largo de la memoria, es difícil llevar la cuenta de todas estas tecnologías cuando están repartidas por el texto. Por ello esta sección actúa de resumen y referencia directa.

12.6.1 Bibliotecas

Tabla 20. Tabla de bibliotecas utilizadas.
FUENTE: elaboración propia.

Nombre	Uso / Descripción	Lenguaje	Sitio web
Mozilla DeepSpeech	Biblioteca de ASR utilizada en el servicio de voz para el módulo de Voz-a-Texto	Python	https://github.com/mozilla/DeepSpeech
Mozilla TTS	Biblioteca de TTS utilizada en el servicio de voz para el módulo de Texto-a-Voz	Python	https://github.com/mozilla/TTS
Rasa	Plataforma de procesamiento del lenguaje natural (NLP).	Python	https://rasa.com/
Snowboy	Motor de detección de palabras de activación (*Hotword*).	Wrapper para Javascript (NodeJS)	https://github.com/Kitt-AI/snowboy
Node-vad	Biblioteca de VAD (Voice Activity Detection) utilizada para discernir entre habla y silencio en el dispositivo IoT.	Javascript (NodeJS)	https://www.npmjs.com/package/node-vad
Python-Shell	Biblioteca para la ejecución de scripts de Python desde NodeJS.	Javascript (NodeJS)	https://www.npmjs.com/package/python-shell
Pixel_ring	Biblioteca que incluye un controlador para los leds de las	Python	https://github.com/respeaker/pixel_ring

	placas ReSpeaker y varios patrones ya definidos.		
Node-record-lpcm16	Biblioteca para el manejo de entrada de audio por micrófono.	Javascript (NodeJS)	https://www.npmjs.com/package/node-record-lpcm16
speaker	Biblioteca para el manejo de salida de audio.	Javascript (NodeJS)	https://www.npmjs.com/package/speaker
wav	Biblioteca para el manejo de audio en formato WAV.	Javascript (NodeJS)	https://www.npmjs.com/package/wav
Built-in de NodeJS	Bibliotecas ya incluidas por defecto como **http**, **fs** (file system) o **stream**.	Javascript (NodeJS)	-

Adicionalmente, se han utilizado todas las dependencias de las bibliotecas aquí recogidas.

12.6.2 Herramientas

Tabla 21. Tabla de herramientas utilizadas.
FUENTE: elaboración propia.

Logo	Nombre	Descripción	Sitio web
Visual Studio Code	Visual Studio Code	Editor de código principal.	https://code.visualstudio.com/
GNU nano	Nano	Editor de código (desde consola, usado en dispositivo IoT).	https://www.nano-editor.org/
node	NodeJS	Para programar los servicios y el control central del dispositivo IoT.	https://nodejs.org/es/
PuTTY	Putty	Herramienta de depuración para acceder al puerto serie del prototipo en la placa ReSpeaker Core V2.	https://www.putty.org/

	GitHub	Control de versiones, usado durante el desarrollo del servicio de voz y del dispositivo IoT (ver **apéndice 0** para el enlace a los repositorios).	https://github.com/
	Trello	Para la planificación del trabajo en tablero.	https://trello.com
	ClickUp	Para el diagrama de Gantt (usando la licencia gratuita de prueba, que ya ha caducado).	https://clickup.com/
	Microsoft Word	Para la redacción y edición de esta memoria final.	https://www.microsoft.com/es-es/microsoft-365/word
	Microsoft Power Point	Para la creación y edición de las figuras e ilustraciones personalizadas del trabajo.	https://www.microsoft.com/es-es/microsoft-365/powerpoint
	Blender	Para importar y editar el modelo 3D de la carcasa del dispositivo IoT.	https://www.blender.org/about/logo/
	Ultimaker Cura	Para preparar el modelo 3D de la carcasa para su impresión.	https://ultimaker.com/es/software/ultimaker-cura
	Google Meet	Para reuniones con el tutor y con el Ayuntamiento.	https://meet.google.com/
	Firefox	Navegador utilizado durante el desarrollo.	https://www.mozilla.org/es-ES/firefox/new/

En el tema de licencias de pago, Microsoft *Word* y *Microsoft Power Point* se han utilizado con la cuenta de *MSCloud* ofrecida por la Universidad Alicante, y *Google Meet* se ha utilizado con la cuenta de *GCloud* ofrecida también por la Universidad de Alicante.

12.6.3 Lenguajes de programación

Se ha utilizado **Javascript** en *NodeJS*, que a su vez compone el núcleo del software del dispositivo IoT y del servicio de voz, así como los puntos de entrada a los subservicios de ASR y TTS.

Se ha utilizado **Python** en los scripts que más tarde se lanzan desde *NodeJS* usando la biblioteca *Python-Shell*: para usar *Mozilla DeepSpeech* y *Mozilla TTS* en el servicio de voz, y el controlador de leds en el dispositivo IoT.

También se ha usado Python en las acciones personalizadas de Rasa.

12.7 CÓDIGO DEL PROYECTO

En este apéndice se recogen los repositorios de código pertenecientes al proyecto, programados por mí, junto a una breve descripción.

12.7.1 Software del dispositivo IoT

Este repositorio contiene el código que se ejecuta en el dispositivo IoT **SAM Device**. Su principal funcionalidad es detectar en local una palabra de activación personalizada, y una vez detectada grabar una orden hasta que detecte silencio, para enviar dicha orden a un servicio de voz externo y reproducir su respuesta por los altavoces. Además, cambia los patrones de leds del dispositivo para indicar qué está haciendo en cada momento (hablando, esperando, etc…).

Se ha utilizado *Snowboy* en su versión para *NodeJS*, así como *node-vad* para la detección de silencio y *pixel_ring* para controlar los patrones de leds.

https://github.com/AlexMaciaFiteni/OpenSourceHotwordDetection.

12.7.2 Software del servicio de voz

Este repositorio contiene el código que se ejecuta en el **SAM Voice Service**. Se encarga de recibir órdenes en forma de audio, de convertirlas a texto con su módulo ASR (*Mozilla DeepSpeech*), de usar ese texto para mantener la conversación con un asistente inteligente (hecho con *Rasa*), y de convertir la respuesta de dicho asistente a audio de nuevo con su módulo TTS (*Mozilla TTS*).

En este repositorio no se incluyen los *modelos de voz para ASR* y *TTS* ya que son demasiado voluminosos como para subirlos, pero sí hay un script que descarga los modelos en español desde la propia plataforma (`download_models.sh`).

https://github.com/AlexMaciaFiteni/OpenSourceVoiceService.

to be continued

Escuela
Politécnica
Superior