Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

Puesta en servicio de un robot social y desarrollo de un conjunto de herramientas de ayuda asistencial.

Autor:
Izquierdo Gómez, Miguel

Tutor:
Gómez García-Bermejo, Jaime
Ingeniería de Sistemas y Automática

Valladolid, Junio 2020

Resumen

El presente Trabajo de Fin de Grado tiene como objetivo principal el desarrollo de herramientas que permitan la comunicación con las personas que residan en una planta de hospital o en un centro de personas de avanzada edad sin la necesidad de la presencia física del personal sanitario o los familiares a través del robot social.

Para ello se ha realizado un análisis comparativo de las diferentes posibilidades existentes en el mercado y se han estudiado tanto sus capacidades como si era viable aplicarlas a este tío de proyecto. Una vez analizado se eligió el robot social James para realizar esta aplicación que cumple con los requisitos necesarios y tiene la autorización de la Comunidad Europea para su implantación. Se han examinado las capacidades y funciones que puede realizar este robot así como todas las herramientas que podemos añadir para mejorar sus habilidades para su uso asistencial.

Se ha utilizado como principal modo de comunicación del robot la tecnología MQTT. Por tanto, en este documento se ha analizado esta tecnología y sus fundamentos teóricos, así como las funciones que puede desarrollar el robot social James.

Para desarrollar una aplicación que sea intuitiva y útil al usuario que maneje el robot se han estudiado dos posibilidades: la primera es una aplicación para móviles Android, y la segunda, una aplicación HTML. La aplicación para el sistema operativo Android tiene la ventaja de ser más visual pero limita la posibilidad de un campo de dispositivos que, aunque es amplio, no es universal. Sin embargo, la aplicación HTML al ser un lenguaje estándar permite que cualquier usuario desde cualquier dispositivo tenga acceso a la aplicación. Por eso este trabajo se ha centrado en desarrollar en lenguaje HTML el interfaz con el usuario.

También se ha desarrollado un soporte físico donde poder instalar una tableta, ya que el robot James no permite realizar videollamadas con voz. Para solucionar este problema se ha desarrollado un esqueleto que se puede obtener mediante impresión 3D por deposición de material fundido. Esto ha permitido añadir una función vital para el objetivo del proyecto, que es facilitar la comunicación de personas en un entorno aislado.

Palabras clave: Robótica asistencial y de servicios, Robots autónomos, Android, HTML, MQTT.

Índice:

1. Introducción 7
1.1. Marco del proyecto 7
1.2. Objetivos: 9
1.3. Descripción de la memoria: 10
2. Robótica social 12
2.1 Introducción a la robótica social 12
2.2. Ejemplos robot sociales anteriores. 14
3. El Robot asistencial James 20
3.1. Estructura física 20
3.2. Funciones principales 23
3.3. Aplicación control ZBOS 32
4. Protocolo de comunicación (MQTT) 35
4.1. Introducción a la tecnología MQTT. 35
4.2. Conceptos de la terminología MQTT 36
4.3. Protocolo de conexión 38
4.4. Funciones MQTT James 39
5. Desarrollo de las funciones MQTT en dispositivos móviles 45
5.1. Introducción 45
5.2. Build.gradle(app): 45
5.3. AndroidManifest.xml 47
5.4. MainActivity.java 48
5.5 Colors.xml 53
5.6 Strings.xml 54
5.7 Activiy_main.xml 55
6. Desarrollo de las funciones MQTT en lenguaje HTML 57
6.1. Introducción 57
6.2. Cabecera del código 59
6.3. Función para conectarse: 59
6.4. Función de perdida de conexión 60
6.5. Función cuando sucede un fallo al conectarse 60
6.6. Función de llegada de mensaje 60
6.7. Función de después de la conexión 61

6.8. Función para subscribirse a un topic dado por el usuario..... 61
6.9. Función para mandar un topic elegido por el usuario 62
6.10. Función para publicar topics de entre una lista..... 62
6.11. Función para subscribirse a topics dentro de una lista..... 63
6.12. Función actualizar el mapa de localización..... 63
6.13. Función para actualizar la posición del robot James..... 65
6.14. Función para hacer un mapa interactivo..... 67
6.15. Función para comprobar la posición de destino a un POI..... 68
7. Implementación de un soporte para la tableta 70
7.1. Samsung Tab A 70
7.2. Soporte tableta..... 72
8. Resultados..... 75
9. Estudio Económico 80
9.1. Introducción..... 80
9.2. Recursos empleados 80
9.3. Costes directos..... 80
9.4. Costes indirectos..... 84
9.5. Costes totales..... 84
10. Conclusiones y Líneas futuras..... 85
Bibliografía..... 87
Anexos..... 89
Código aplicación HTML..... 89
Código App Android 104

Índice de figuras

Ilustración 1 Robot Teresa....14
Ilustración 2 Robot Paro....15
Ilustración 3 Robot Pearl....16
Ilustración 4 Robot Pepper....18
Ilustración 5 Partes robot Pepper....19
Ilustración 6 Sensores robot Pepper....19
Ilustración 7 Robot James lateral....22
Ilustración 8 Robot James vista frontal....22
Ilustración 9 esquema de aplicaciones....25
Ilustración 10 Esquema y pantalla principal James....25
Ilustración 11 Todas las aplicaciones instaladas....26
Ilustración 12 Multimedia....26
Ilustración 13 Applications....27
Ilustración 14 Zorabots....27
Ilustración 15 Encuesta....28
Ilustración 16 Reconocimiento facial....28
Ilustración 17 Lista de configuraciones de kiosk....29
Ilustración 18 Seleccionamos editar Kiosk....29
Ilustración 19 Settings Kiosk....30
Ilustración 20 Habilitar el botón superior....30
Ilustración 21 Editar apariencia item....31
Ilustración 22 Características del item....31
Ilustración 23 Opciones Zorabots aplicación....32
Ilustración 24 Dashboard....33
Ilustración 25 Mapa slam James....33
Ilustración 26 Scheduler....34
Ilustración 27 MQTT diferentes usos....35
Ilustración 28 Distribución de mensajes....36
Ilustración 29 MQTT en el modelo OSI....37
Ilustración 30 Ejemplo de zbos/dialog/set/message....39
Ilustración 31 Ejemplo obtención del mapa....42
Ilustración 32 Suscripción a todos los topics....43
Ilustración 33 Publicación de las id de las composiciones....44
Ilustración 34 librerías de Paho (Kock, 2015)....46
Ilustración 35 Service (Kock, 2015)....47
Ilustración 36 Permisos de Android.xml (Kock, 2015)....48
Ilustración 37 Creación del cliente MQTT (Kock, 2015)....49
Ilustración 38 Funciones de MQTT (Kock, 2015)....50
Ilustración 39 Aplicación Android de MQTT....56
Ilustración 40 Código de javascript con websocket (Martin, 2020)....58
Ilustración 41 Diseño 3D robot social James....70
Ilustración 42 Samsung Tab A....72
Ilustración 43 Primer diseño soporte frontal....73

Ilustración 44 Primer diseño soporte trasero 73
Ilustración 45 Soporte James definitivo 74
Ilustración 46 Diseño página web....75
Ilustración 47 Mensajes página web James 76
Ilustración 48 Simulación mapa hospital....77
Ilustración 49 Cuestionario paciente....78
Ilustración 50 Composición....79

Índice de tablas

Tabla 1 Comandos de voz en ingles 24
Tabla 2 Comandos de voz en español....24
Tabla 3 Código de mensajes CONNACK 38
Tabla 4 Coste anual del personal 81
Tabla 5 Días efectivos por año....82
Tabla 6 Distribución temporal del trabajo....82
Tabla 7 Costes de amortización....83
Tabla 8 Costes Indirectos 84
Tabla 9 Costes totales 84

1. Introducción

1.1. Marco del proyecto

El presente Trabajo Fin de Grado se enmarca en el ámbito de la robótica social y la tecnología IoT, también conocida como el Internet de las cosas (Internet of things). Estos campos están en auge y cada vez más presentes en nuestra vida cotidiana. Ya no son un mero un experimento que se muestra en convenciones sino una aplicación realista en la actividad cotidiana.

En robótica se pueden diferenciar dos grandes campos: la robótica industrial y la robótica de servicios. Los robots de tipo industrial son aquellos que se encuentran en factorías o empresas y que, por lo general, realizan o ayudan en la realización de tareas industriales. Por su parte, los robots de servicios se encargan de la asistencia y la ayuda a los humanos en diversas actividades, generalmente no enfocadas al ámbito industrial.

Este proyecto se centrará más concretamente en la robótica de servicios. Para ello es importante la interacción hombre-máquina, especialmente utilizando el reconocimiento de voz y ayudas visuales para generar una comunicación fluida entre el robot y la persona.

Hay muchos tipos de robots de servicios: esto dependerá de su tarea y del entorno en el que se desarrolle. Un robot para la mente colectiva tiene forma humanoide, aunque estos sean los más escasos, tanto por cantidad como por capacidad de uso. Sin embargo, dentro de la robótica de servicios, debido a la gran cantidad de aplicaciones que hay disponibles, se pueden encontrar todo tipo de robots: biomiméticos, humanoides, plataformas móviles… Cada uno de ellos es más adecuado para unas tareas u otras.

Para el desarrollo del presente proyecto se ha tomado como punto de partida la plataforma robótica James. Se trata de un robot comercializado por la compañía Zorabots. Nos centraremos en el robot James cuyas características principales son ser un robot autónomo y móvil, que es capaz de escuchar comandos de voz y responder o actuar de forma acorde. Dentro de sus funciones encontramos, por ejemplo, que puede moverse hacia un punto o dar el parte meteorológico.

La aplicación informática de James ha sido diseñada para que responda a la voz de los usuarios y se pueda desplazar entre puntos predeterminados o mostrar información en su pantalla. James también puede controlar las diferentes partes de una casa domótica (apagar o encender luces, abrir puertas…).

James escanea continuamente su entorno usando un sistema láser (LIDAR). Este sistema ya está instalado de base en el robot y genera un mapa de su entorno que puede ser ajustado por el usuario para añadir barreras, de modo que el robot no acceda a zonas sensibles. Este mapa se puede mostrar

en la aplicación que nos proporciona la compañía del robot y también nos muestra los puntos ya marcados previamente, además de la estación de carga, posición inicial del robot James.

El robot James que se ha descrito es el que vamos a utilizar para desarrollar este proyecto usando la tecnología IoT y más concretamente, MQTT que es un protocolo de comunicación por mensajería (MQ Telemetry Transport or Message Queuing Telemetry Transport). Se trata de un estándar abierto de OASIS e ISO (ISO/IEC 20922). Es un protocolo ligero que da lugar a un sistema de publicación y suscripción que permite la comunicación de mensajes entre diferentes aparatos. Este protocolo normalmente se realiza en TCP/IP, si bien es compatible con otros que tengan características de conexión. El internet de las cosas es un sistema que nos permite intercomunicar dispositivos móviles, ordenadores, elementos domóticos y máquinas digitales sin necesidad de que haya interacción humana.

1.2.Objetivos:

Objetivo principal:

El objetivo central del presente Trabajo de fin de Grado consiste en crear herramientas para implantar el robot social James en un entorno interior como una planta hospital o en una residencia de ancianos. Estas herramientas son para hacer más fácil e intuitiva la comunicación tanto con el robot como con los usuarios de modo que puedan usarlo para establecer comunicaciones con otras personas.

Para llevar a cabo este objetivo se van a realizar una serie de pasos:

1. Debemos conocer las funciones que puede realizar un robot, realizando un estudio sobre las diferentes posibilidades que nos ofrece el mercado, y elegir el robot que mejor se adapte a las necesidades del proyecto.
2. Crear una aplicación que haga que la comunicación con el robot sea fácil e intuitiva y se pueda aplicar al entorno sanitario o al de una residencia de personas de avanzada edad.
3. Crear herramientas para facilitar la comunicación del a través del robot James. Haciendo que este sea una vía de comunicación de telemática sin necesidad de contacto físico.

Basándonos en lo anterior definimos unos subobjetivos que dirigirán el desarrollo del proyecto:

- Analizar los diferentes tipos de robot sociales diseñados hasta la actualidad, de modo que podamos elegir un robot que tenga una capacidades y funciones útiles dentro del proyecto.
- Llevar a cabo una investigación sobre el funcionamiento del robot elegido, su software, sus características, sus compatibilidades y las aplicaciones de las que dispone.
- Llevar a cabo el proyecto de un modo costo-efectivo.

1.3.Descripción de la memoria:

Una vez definidos los objetivos del proyecto, a la hora de abordar la memoria y la documentación de este, hay unos apartados bien diferenciados, que se corresponderán con los diferentes capítulos del presente texto.

En el primer capítulo se realiza una introducción a la robótica social y su relación con los sistemas de IoT y, más concretamente, sus aplicaciones en el robot James. Así mismo, se define el objetivo principal y los objetivos secundarios de todo el proyecto.

En el segundo capítulo se hace un recorrido por el estado actual, tanto de la robótica social, principal ámbito de aplicación de este proyecto, como de la interacción hombre-máquina, totalmente necesaria para el desempeño de la primera, haciendo un especial hincapié en sistemas de comando por voz que será la principal vía de comunicación con James. La explicación del marco en el que se encuentra la tecnología actual nos permite comprender la importancia de este proyecto y de este tipo de robótica

En el tercer capítulo se realiza el análisis en detalle de las características del robot James, tanto de las funciones a realizar como de sus características físicas. Es necesario conocer cuáles son sus capacidades y limitaciones para tenerlas en cuenta a la hora de desarrollar el proyecto.

El cuarto capítulo describe la programación basada en el protocolo MQTT. Esta tecnología es la que nos permite comunicarnos con el robot. Por ello se va a explicar los fundamentos teóricos de MQTT y las funciones que puede realizar el robot con este tipo de comunicación.

En el capítulo quinto se describen los pasos que se han seguido para realizar un programa en Android capaz de comunicarse con el robot bidireccionalmente. Es importante analizar las funciones que nos permiten desarrollar una aplicación Android que combine la comunicación MQTT, así como las funcione específicas del robot social James.

En el capítulo sexto se describen los pasos que se han seguido para realizar una aplicación HTML capaz de comunicarse con el robot bidireccionalmente. El lenguaje HTML es mucho más compatible que el código Android, lo que nos permite que sea ejecutable desde casi cualquier dispositivo. Se analizará las funciones que nos permiten desarrollar la comunicación MQTT en el lenguaje HTML. Este lenguaje se ha elegido para desarrollar la aplicación final porque es más universal que el lenguaje Android y el lenguaje Android no nos da ninguna ventaja funcional sobre una aplicación web.

El capítulo séptimo desarrolla los pasos seguidos para realizar el diseño de una parte de soporte al robot James que nos permita colocar una tableta. Una de las funciones principales que queremos introducir es la realización de

videollamadas. Con este diseño nos permitirá tener una pequeña tableta conectada para realizar este tipo de conexiones.

En el octavo capítulo se hace una relación clara de los resultados obtenidos en los apartados anteriores, las pruebas realizadas con las diferentes aplicaciones y una comprobación del grado de cumplimiento de los objetivos del proyecto.

El noveno capítulo consiste en un estudio del coste económico de todo el proyecto, explorando en detalle los costes del sistema adyacente incluido y el software desarrollado en este proyecto, los costes de personal y recursos utilizados en la elaboración de este, con el objetivo de ver la viabilidad económica.

Para terminar, en el capítulo décimo se hace una recapitulación general del proyecto en su totalidad para enumerar las conclusiones obtenidas, determinar lo aprendido y los problemas encontrados, así como proponer posibles líneas de mejora o ampliaciones de este.

Por último y para concluir la memoria, se enumera la bibliografía consultada y referenciada a lo largo de la memoria y los anexos.

2. Robótica social

Es necesario, antes de comenzar a desarrollar este trabajo, realizar una introducción sobre la robótica social que es la parte de la robótica que se centra en la interacción y servicio a las personas. Se va a realizar un recorrido por los trabajos previos realizados sobre esta temática, y se va a analizar diferentes tipos de robots sociales que tienen, de una manera u otra, relación con nuestro robot.

2.1 Introducción a la robótica social

2.1.1 Definición

Un robot está definido por el International Standard of Organization (ISO) como "una máquina multifuncional reprogramable que está diseñada para mover material, piezas, herramientas o dispositivos especializados a través de movimientos programados variables para realizar una variedad de tareas" (ISO, 2020). Como un subconjunto de los robots, los robots sociales realizan uno o todos estos procesos en el contexto de una interacción social. Hay muchos tipos de interacciones sociales y sus tareas pueden variar desde tareas de apoyo simple, como facilitar una herramienta a un trabajador, hasta la comunicación y conversación compleja, como en el entorno sanitario. A los robots sociales se les pide que trabajen al lado de los humanos en espacios de entorno colaborativos. Además, los robots sociales están diseñados para un entorno más íntimo como puede ser la atención médica o el hogar.

2.1.2 Historia del robot social

Si bien los robots a menudo se han descrito con cualidades sociales, la robótica social es una rama bastante reciente de la robótica. Desde principios de la década de 1990, los investigadores de inteligencia artificial y robótica han desarrollado robots que se involucran explícitamente a nivel social. Este progreso ha traído grandes beneficios, pero también muchas dudas sobre el avance de la tecnología sobre las relaciones humanas. Un ejemplo son los problemas que pueden surgir cuando las personas desarrollen afecciones con un robot (Sherry Turkle, Will Taggart, Cory D. Kidd, & Olivia Dasté, 2006).

El diseño de un robot social autónomo que sea capaz de interactuar con las personas sin ser manejado por otra persona es particularmente desafiante, ya que el robot necesita interpretar correctamente la acción de las personas y responder adecuadamente, lo que actualmente aún no es posible debido a la multitud de variables que se presentan. Uno de los problemas a los que se enfrenan los robots sociales son que las personas que interactúan con un robot social pueden tener expectativas muy altas de sus capacidades, basadas en representaciones de ciencia ficción de robots sociales avanzados. Como tal,

muchos robots sociales son controlados parcial o totalmente a distancia para simular capacidades avanzadas. Este método de control (a menudo de forma encubierta) de un robot social se conoce como Mechanical Turk (Rice, 2004). Los estudios de este tipo son útiles en la investigación de la robótica social para evaluar cómo las personas responden a los robots sociales

2.1.3 Interacción social

Diferentes estudios han investigado la interacción del usuario con un compañero robot. Algunos experimentos han explorado un escenario naturalista como un trabajo que el que los niños juegan al ajedrez con el iCat, como su compañero robot . Se presenta un enfoque bayesiano independiente de la persona para detectar el compromiso del usuario con el robot iCat. En su marco modelan tanto las causas como los efectos del compromiso: las características relacionadas con el comportamiento no verbal del usuario, la tarea y las reacciones afectivas del compañero se identifican para predecir el nivel de compromiso de los niños. Se realizó un experimento para entrenar y validar nuestro modelo. Los resultados muestran que nuestro enfoque basado en la integración multimodal de características basadas en la tarea y la interacción social supera a las basadas únicamente en el comportamiento no verbal o la información contextual (94.79% vs. 93.75% y 78.13%).Este experimento muestra que las interacciones con un robot social que sea capaz de mostrar reacciones faciales crean más vínculos con su interlocutor. Esto es una muestra más de la importancia de que un robot social siga los patrones humanos para que comunicación con los seres humanos sea la mejor posible. (Castellano, Leite, Pereira, Paiva, & McOwan, 2009)

2.1.4 Impacto social

El uso cada vez más extendido de robots sociales más avanzados es uno de varios fenómenos que se espera que contribuyan a la posthumanización tecnológica de las sociedades humanas, a través del cual el proceso "una sociedad llega a incluir miembros distintos de los seres humanos biológica 'naturales' que, de una forma u otra , contribuyen a las estructuras, dinámicas o significado de la sociedad " (Gladden, 2018)

El impacto de crear un mundo donde nuestra experiencia sea solo de forma virtual crea riesgo en nuestra sociedad y en nuestra manera de comportarnos. “Sin una comprensión profunda de lo que muchas personas expresamos en lo virtual, no podemos usar nuestras experiencias allí para enriquecer lo real. Sin embargo, si cultivamos nuestra conciencia para entender lo que está detrás de nuestras pantallas, es más que probable que tengamos éxito en el uso de la experiencia virtual para la transformación personal.” (Turkle, 1996)

2.2.Ejemplos robot sociales anteriores.

2.2.1. Teresa

Teresa (TElepresence REinforcement-learning Social Agent) es un robot social semiautomático de telepresencia está diseñado para personas con algún problema de movilidad o de edad avanzada (Shiarlis, y otros, 2015).Su objetivo es que la persona pueda asistir a eventos sociales sin la necesidad de encontrarse presente en ellos.

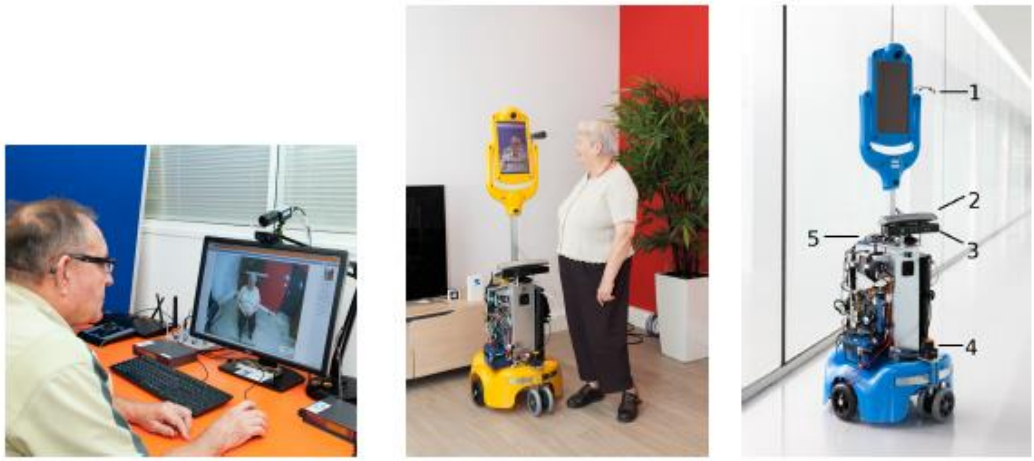

Ilustración 1 Robot Teresa

TERESA se controlará desde un ordenador, donde estará la persona que maneja el robot (*ilustración 1*). A través de una cámara se podrá ver el rostro de la persona en la pantalla central del robot (*ilustración 1*). El usuario podrá interaccionar con el robot ya que podrá ver gracias a una cámara instalada en la parte superior de la pantalla, esta imagen se reproducirá a tiempo real en el ordenador. TERESA es un medio para permitir la interacción humano-humano.

El robot también cuenta con sistema de reconocimiento de voz y reconocimiento facial de las expresiones de la persona que habla con el robot, interpretando en función de las expresiones de su rostro sus sentimientos.

El robot también es capaz de interaccionar con los entornos, por ejemplo, si todas las personas se desplazan a un punto determinado de la sala, él también lo hará. Para poder realizar este tipo de movimientos tiene instalado un sistema de navegación que permite trazar rutas, eligiendo la mejor dependiendo de los obstáculos del camino. Una característica de TERESA es que se puede adaptar a la altura de su interlocutor para estar siempre cara a cara con él. Esto permite que mantener una conversación más cómodamente (*ilustración 1*). A través del audio el robot se colocará de forma que sea mejor la captación de sonido, esto hace que se oriente de hacia el usuario de forma automática. Además, puede mantener conversaciones en movimiento.

Las conclusiones una vez estudiado este robot es que es un robot que permite la interacción de humano a humano de una manera muy cómoda pero no hay una interacción humano-robot. Es decir, siempre tiene que haber una persona que maneje el robot no puede realizar actividades de manera autónoma.

2.2.3 Paro

El paro es un robot social con forma de bebe foca como se puede ver en la *ilustración 2*. Este robot se ha diseñado para relacionarse con humanos como medio de terapia para la persona que lo utilice. Tiene un peso de unos 2.8kg y una autonomía de una hora. (Kidd, Taggart, & Turkle, 2006).

Ilustración 2 Robot Paro

Dentro de sus funciones básicas podemos nombrar las siguientes: Tacto, reconocimiento de expresiones faciales y puede expresar algún movimiento de forma limitada con la cabeza y las aletas. Estas capacidades las puede realizar gracias a que dispone de un sensor de luz, un sensor de audio que le permite orientarse a la fuente de sonido además de otros sensores que le permiten ser capaz de mantener el equilibrio (Kanamori, y otros, 2003).

Paro está destinado para su uso por personas de avanzada edad, tanto por su faceta de robot asistencial como de forma terapéutica siendo su función parecida a la que pudiera tener una mascota. Está orientado a centros residenciales y a hogares con personas de mayores donde una mascota real no es viable y Paro puede ser un buen sustito.

Las conclusiones una vez estudiado este robot es que es un robot que está destinado al mismo campo al que se va a dedicar este proyecto, pero con un enfoque muy diferente, ya que no hay comunicación verbal robot-humano y tampoco el robot se puede desplazar de forma autónoma.

2.2.4 Pearl

Pearl es un robot asistencial que está destinado para personas de avanzada edad (*ilustración 3*). Este robot tiene como funciones primarias

recordar a las personas sus actividades diarias como puede ser comer, beber, ingerir las medicinas que les corresponder, guiarles a dentro del hogar o ayudarles en sus tareas cotidianas. (E. Pollack, y otros, 2002).

Ilustración 3 Robot Pearl

Pearl está orientado a personas que viven en su domicilio, pero necesitan de algún tipo asistencia y tienen un deterioro cognitivo leve. Tiene integrados dos ordenadores, una conexión vía wifi, un láser para la búsqueda de objetos, sensores sonar, micrófonos, altavoces, una pantalla táctil, un sistema de cámara estéreo.

Cuenta además con un sistema de navegación autónoma que le permite desplazarse por el entorno. Este sistema cuenta con la capacidad de captura de imágenes rápida, de emisión de video en línea, reconocimiento facial y un programa de seguimiento. Este conjunto de capacidades es lo que le facilita la movilidad autónoma.

Una de sus funciones principales es la de recordar a la persona a la que da servicio de algunas actividades preprogramadas como puede ser tomarse la mediación. Para ello tiene un sistema flexible que le permite dar respuesta a las acciones del usuario, siendo capaz de introducir pequeñas variaciones según la rutina del cliente para no ser molesto.

Otra de función que tiene Pearl es la de ayudar al movimiento de sus usuarios. Para ello es capaz de adaptarse a la velocidad de la persona. Con los escáneres puede evitar los obstáculos y guiar la persona de manera correcta.

Las conclusiones de este robot es que tiene una interacción parecida a lo que puede ofrecer nuestro robot James ya que cuenta con unas

características parecidas como los escáneres, además de tener unos programas preinstalados que pueden dar un servicio similar.

2.2.5 Pepper

Pepper (*Ilustración 4*) es un robot humanoide, esto significa que es un robot con una forma similar a la humana, producido industrialmente lanzado en junio de 2014 que se creó por primera vez para las necesidades B2B, es decir para comercio entre empresas (business to business), y luego adaptado para propósitos B2C, comercio para el consumidor (business to consumer). La máquina es capaz de exhibir lenguaje corporal, percibir e interactuar con sus alrededores. También puede analizar las expresiones y los tonos de voz, utilizando los últimos avances y algoritmos patentados en reconocimiento de voz y emoción para provocar interacciones. El robot está equipado con funciones e interfaces de alto nivel para la comunicación multimodal con los humanos a su alrededor. (Amit Kumar Pandey & Rodolphe Gelin, 2018)

Pepper es un robot humanoide con ruedas de 1,2 m de altura, con 17 articulaciones para un lenguaje corporal elegante y expresivo, tres omnidireccionales ruedas para moverse suavemente, aproximadamente 12 horas de duración de la batería para realizar actividades sin parar y la capacidad de volver a la estación de recarga, si fuera necesario. Es un robot con forma agradable, sin bordes afilados, para un aspecto más atractivo y una presencia más segura en el entorno humano. Tiene partes blandas en algunas articulaciones (por ejemplo, el codo, el hombro y la cadera) que impiden el riesgo de que se produzca un atrapamiento. El tamaño y el aspecto de la máquina apuntan a hacerla apropiado y aceptable en la vida diaria para interactuar con seres humanos. Está diseñado para una amplia gama de funciones, gestos y comportamientos y está equipado con una tableta para facilitar la comunicación.

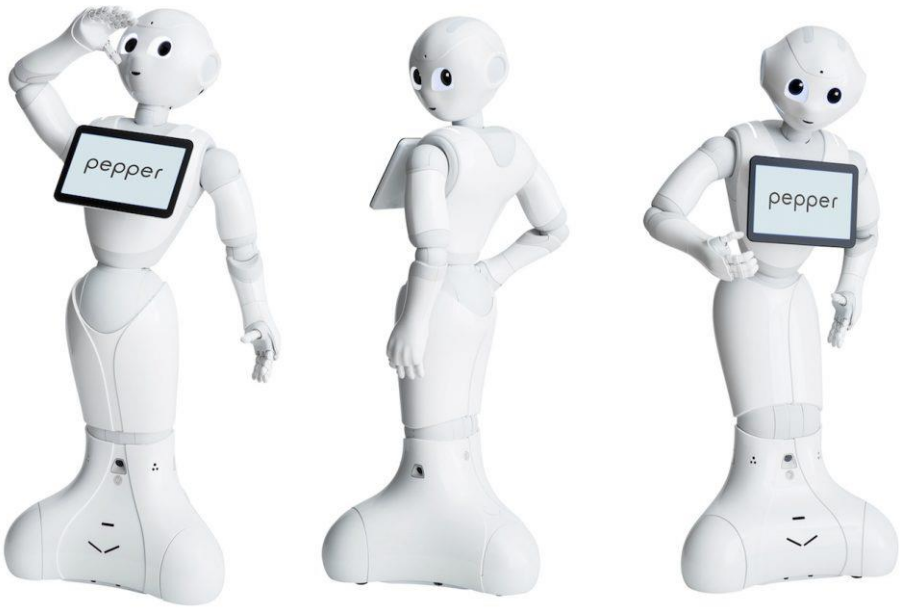

Ilustración 4 Robot Pepper

La interacción es una de las características clave del robot Pepper. La necesidad de interacción natural e intuitiva en un entorno social por eso el diseño de la máquina también considera situaciones de la vida real en las que un medio de comunicación puede que no siempre sea particularmente confiable o útil. Por lo tanto, Pepper tiene una interfaz multimodal de interacción. Esta incluye una pantalla táctil, voz, cabeza y manos táctiles, y diodos emisores de luz (LED). Además, tiene la capacidad de reconocer y responder a las emociones humanas, una biblioteca de gestos expresivos y comportamientos para mostrar interacción real. Para lograr mayor expresividad a través del lenguaje corporal la estructura del robot fue diseñada con 17 articulaciones (*ilustración 5*). Las tres ruedas omnidireccionales ayudan a lograr una que se desplace suavemente.

El robot Pepper tiene una gama de sensores para permitirle percibir objetos y humanos en su entorno y ayudan al software para que pueda realizar las tareas de la mejor manera posible. Estos son mostrados en la *ilustración 5*.

Podemos ver que Pepper es el robot más avanzado que se ha analizado en este capítulo. Además, Pepper se vende de forma comercial desde hace unos años teniendo bastante buena aceptación. Para este trabajo es muy importante analizar un robot como Pepper porque muestra muchas similitudes con el James que es el robot en el que se centra este trabajo de fin de grado. Se ha decidido trabajar con James porque tiene unas ventajas claras sobre

todos los robots estudiados en este capítulo. El robot James nos permite aplicarlo dentro del entorno europeo ya que cuenta con el marcado CE y muchos robots como el Pepper solo se pueden usar para aplicaciones académicas. Además, el robot James es capaz de moverse en espacios abiertos de forma autónoma, mostrar su posición en un mapa que genera y otras funcionalidades que se analizaran en el capítulo siguiente.

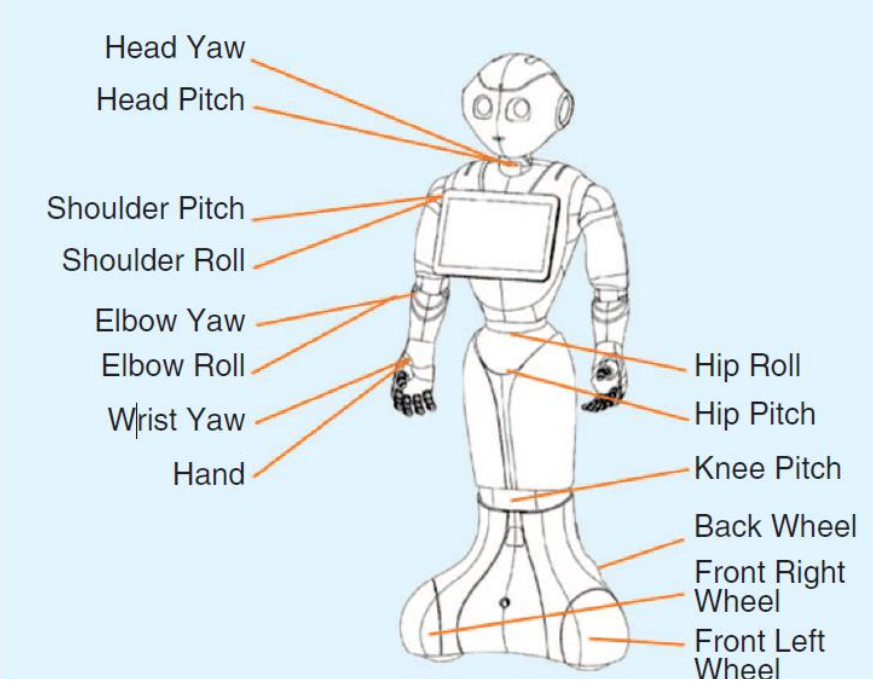

Ilustración 5 Partes robot Pepper

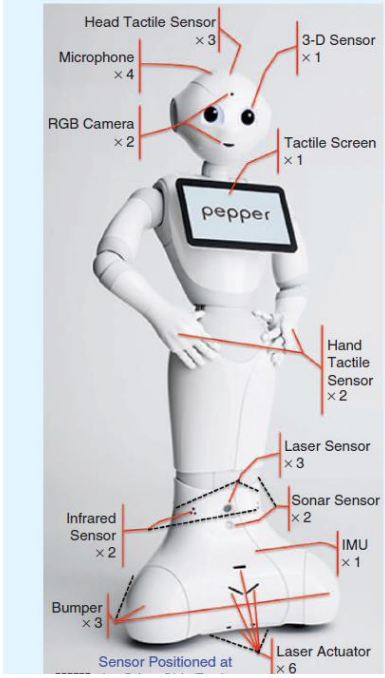

Ilustración 6 Sensores robot Pepper

3. El Robot asistencial James

James es un robot humanoide construido por SiaSun y programado por el equipo Zorabots. El principal objetivo que tuvieron al diseñar a James es crear un robot que pueda guiar a los usuarios a sitios específicos, pero tiene muchas más características complementarias.

James puede interactuar con los usuarios con una tableta Android que tiene instalada en la cabeza, también se controla con una aplicación desde el móvil o desde el ordenador . Además, cuenta con un altavoz y un micrófono en el cuerpo del robot. James analiza continuamente sus alrededores con un sistema SLAM (Simultáneos Localization And Mapping) que está colocado en la base del robot. Este sistema nos permitirá conocer los obstáculos que están cerca del robot generando un mapa del entorno y es capaz de diseñar trayectorias para ir de un punto a otro del mapa.

3.1.Estructura física

Las dimensiones máximas del robot son 420 mm x 420 mm x 800 mm y su estación base mide 240 mm x 85 mm x 95 mm . Es un robot humanoide, aunque su altura es la mitad de un ser humano medio. Este tamaño más bajo es usual en este tipo de robots para que sea visto de forma amable por las personas que interactúan con él.

Las condiciones de funcionamiento para el correcto desarrollo del robot son:

- Una temperatura entre 0-40ºC
- La humedad entre 20-90%
- Máximo tiempo de operación 9-12h

Características de la placa base:

- Tiene 8 núcleos
- La velocidad del reloj 2.0 GHz
- El modelo e CPU MSM8953

Características de almacenamiento:

- 3 GB de RAM
- 32 GB de almacenamiento interno

Periféricos:

- Pantalla táctil de 10 pulgadas capacitiva de alta sensibilidad
- Cámara de 13 MP
- 6 micrófonos del tipo ring-array. Permite una recepción de 360º.
- 2 altavoces de 5W de alta fidelidad.

Conectividad:

- Una tarjeta WIFI de 2.4/5 G de doble banda (802.11 b/g/n).
- 4G Network acepta tarjetas SIM.
- Bluetooth versión 4.1.

Características de la Tableta que está instalada en la cabeza:

- Qualcomm MSM8953 CPU (octa-core)
- LAN 9500A microchip (USB a Ethernet-conversor)
- USB2517-J2X (controlador- HUB)
- RF7460 (Amplificador de potencia)
- Tiene un botón en la parte alta del robot que permite programar sus interacciones

Características de la base del robot:

- En esta parte del robot se coloca la batería
- Hay 2 parachoques a ambos lados de la base
- 2 ruedas motorizadas y rueda sin tracción (rueda loca).
- En la parte delantera se encuentra un escáner SLAM.
- Microswitches + LIDAR en la parte trasera de la base para detectar la estación de carga

El robot en resume tiene una forma compacta y una estatura baja (ilustración 7 y 8).

Ilustración 8 Robot James vista frontal

Ilustración 7 Robot James lateral

3.2.Funciones principales

3.2.1.Comandos de voz

El robot James responde a comandos preprogramados de voz tras escuchar las palabras de activación "OK James". La *tabla 1* contiene los comandos de voz que James admite:

Comandos de Voz	Acciones de James
Call help	Lanza una señal de alarma. Esta opción debe estar instalada previamente.
I need/want help	
Start application <app> E.g. Start Application Slam	El robot inicia una aplicación que se indica. (SLAM en su ejemplo)
What is the date of today?	El robot dice la fecha.
What year is it?	El robot dice el año actual.
What's the day? Which day are we today?	El robot dice que día de la semana es por ejemplo Lunes.
What time is it? How late is it?	El robot dice el la hora y la fecha.
Hello	El robot hace una introducción.
Goodbye	El robot se despide.
Introduce yourself/Who are you?	El robot hace una introducción.
What will the wheater be like this morning/afternoon/evening?	El robot dice la predicción del tiempo de ese momento del día.
What is the wheater for today? What is the wheather forecast for today?	El robot dice el tiempo para hoy.

What will the wheater be like tomorrow?	El robot dice el tiempo para mañana.
Go to <point-Of-Interest>	El robot se moverá al POI indicado.
Enable/disable offer more help feature	¿El robot activara o desactivara el mensaje "Can I help you with something else?".

Tabla 1 Comandos de voz en ingles

En esta *tabla 2* están representados los comandos que están disponibles en español en nuestro robot James. Si bien hay algún comando menos que en la versión inglesa, los comandos principales están en ambos lenguajes.

Comandos de Voz	Acciones de James.
Abrir aplicación <Aplicación> Ej. abrir aplicación CIT	El robot inicia una aplicación que se indica. (CIT en su ejemplo)
¿Qué año es?	El robot dice el año actual.
¿Qué día es hoy?	El robot dice que día de la semana es por ejemplo Lunes.
¿Qué hora es?	El robot dice el la hora y la fecha.
Preséntate	El robot hace una introducción.
¿Qué tiempo va a hacer por la mañana/tarde/noche?	El robot dice la predicción del tiempo de ese momento del día.
¿Qué tiempo va a hacer hoy?	El robot dice el tiempo para hoy.
¿Qué tiempo va a hacer mañana?	El robot dice el tiempo para mañana.
Ve hacia el <punto de interés> Ve hacia la estación de carga	El robot se moverá al POI indicado.

Tabla 2 Comandos de voz en español

3.2.2. Interacción con la tableta

La tableta es una de las principales herramientas con las que cuenta el usuario para interactuar con James. Es una tableta Android que tiene una serie de aplicaciones preinstaladas, cuyas características veremos más adelante. Además, podemos descargarnos cualquier aplicación de la *app store.*

La pantalla principal de la tableta es la que se muestra en la *ilustración 9*.

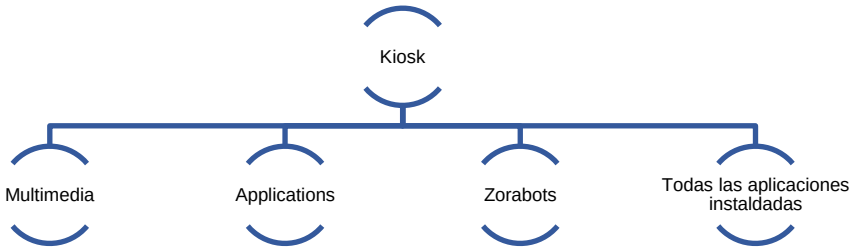

Ilustración 9 esquema de aplicaciones

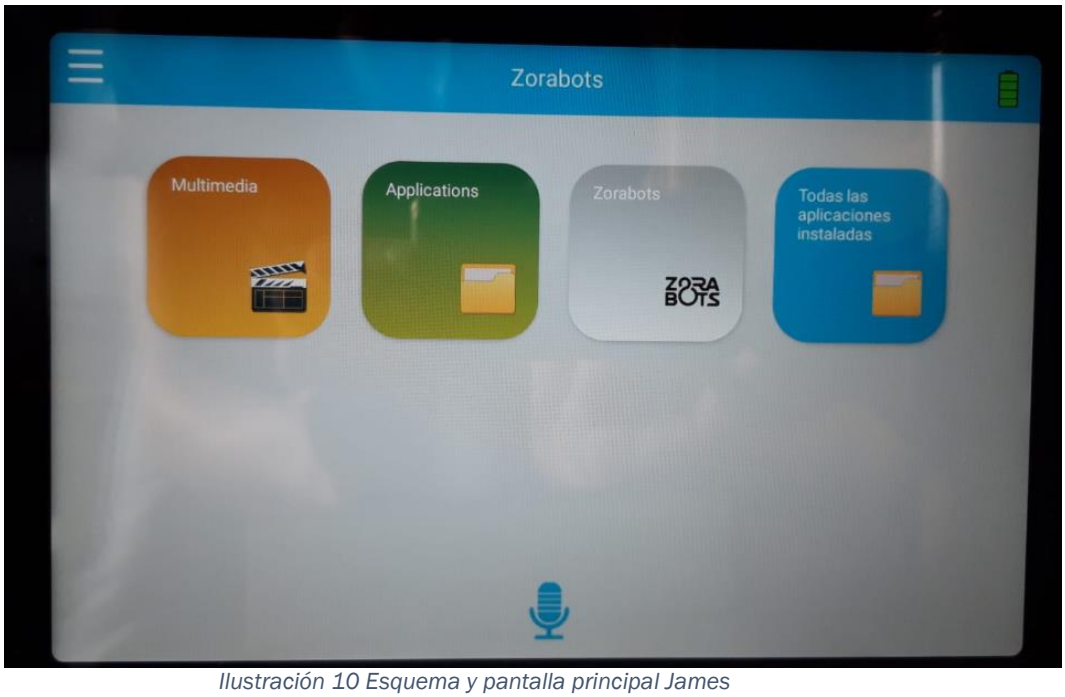

Ilustración 10 Esquema y pantalla principal James

. Kiosk es el principal programa del robot desde el cual accedemos al resto de funciones. A partir de estos cuatro iconos podemos desplegar todas las aplicaciones disponibles en el robot James. Por ejemplo, el último icono despliega las opciones que se muestran en la *ilustración 9 y 10.* En la *ilustración 11* nos muestran las pantallas que se ven en James cuando pulsamos el icono todas las aplicaciones. En la *ilustración 12* vemos lo que nos muestra el icono multimedia. En la *ilustración 13 y 14* se muestra lo que hay cuando pulsamos el icono applications y Zorabots respectivamente

Ilustración 11 Todas las aplicaciones instaladas

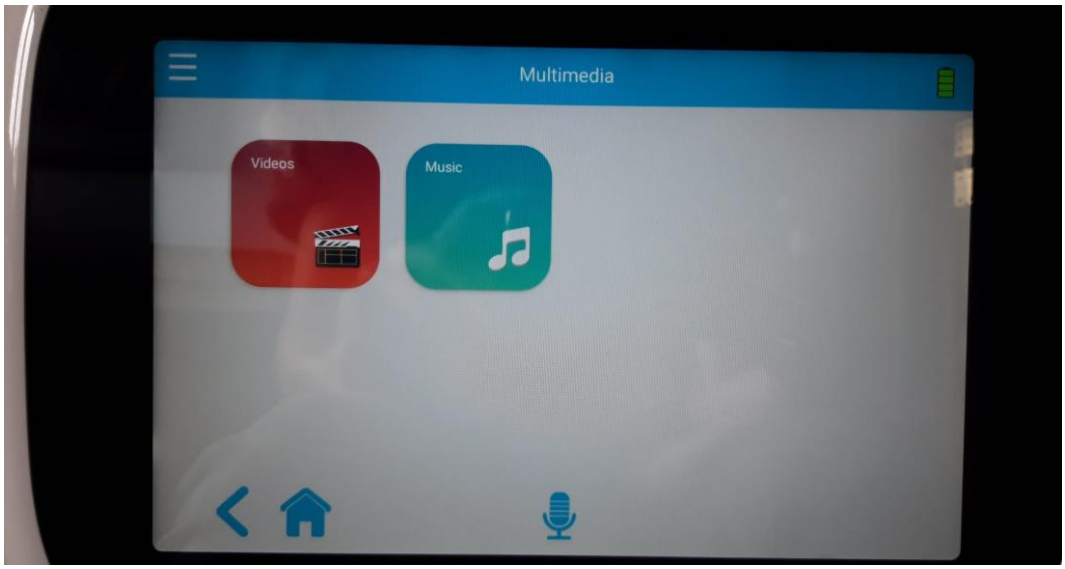

Ilustración 12 Multimedia

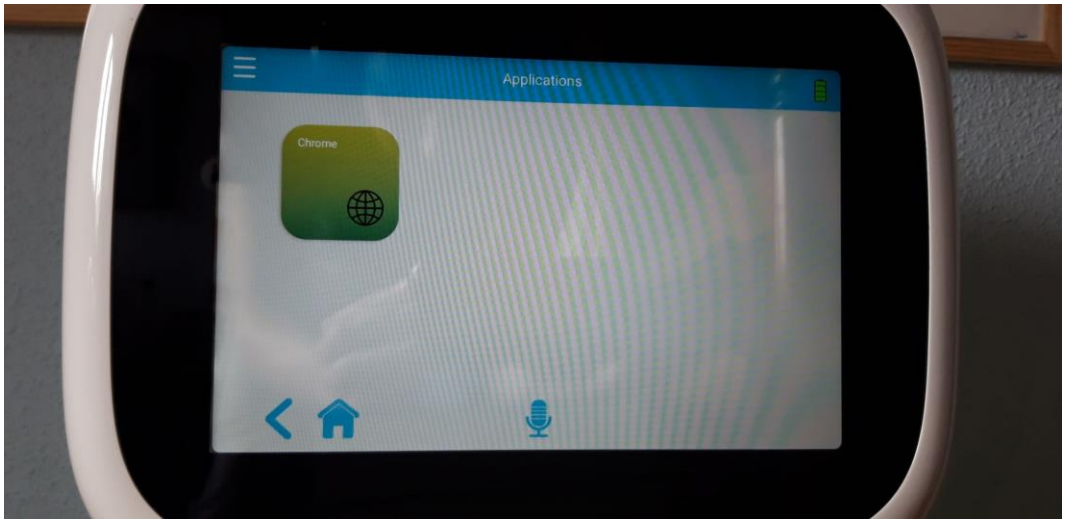

Ilustración 13 Applications

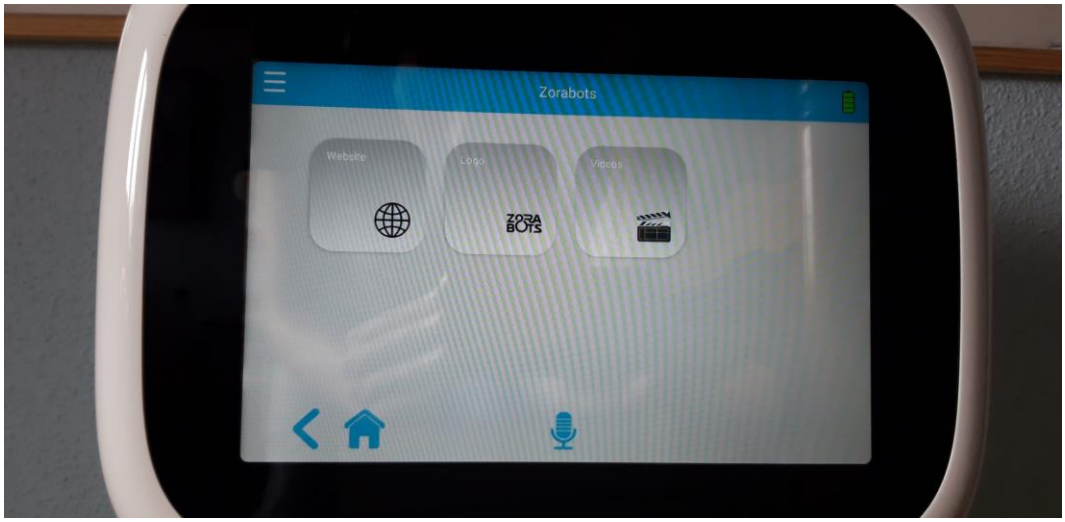

Ilustración 14 Zorabots

Las aplicaciones más importantes son dos que ya vienen preinstaladas en el robot: la aplicación de encuestas y otra de reconocimiento facial.

La aplicación de encuestas nos permite interactuar con el usuario y obtener información acerca de él. Ya que el robot James tiene un uso asistencial es importante que el usuario pueda enviarnos información, y un modo sencillo y cómodo puede ser este formato. Como vemos en *la ilustración 15* nos permite hacer diferentes tipos de preguntas y recibir diferentes tipos de respuestas para una mayor flexibilidad.

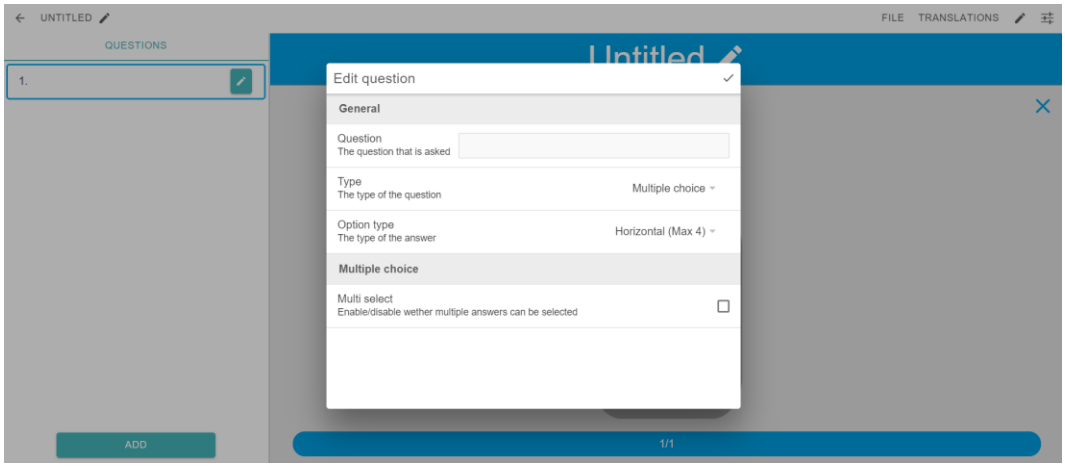

Ilustración 15 Encuesta

La otra aplicación destacable es la de reconocimiento facial que se muestra en la *ilustración 16,* que nos permite, con tres fotos tomadas en diferentes posiciones, reconocer a un usuario. Esto puede ser muy útil para interaccionar en los casos en los que James va a estar siempre con unas mismas personas, como puede ser en una residencia sociosanitaria o en un domicilio.

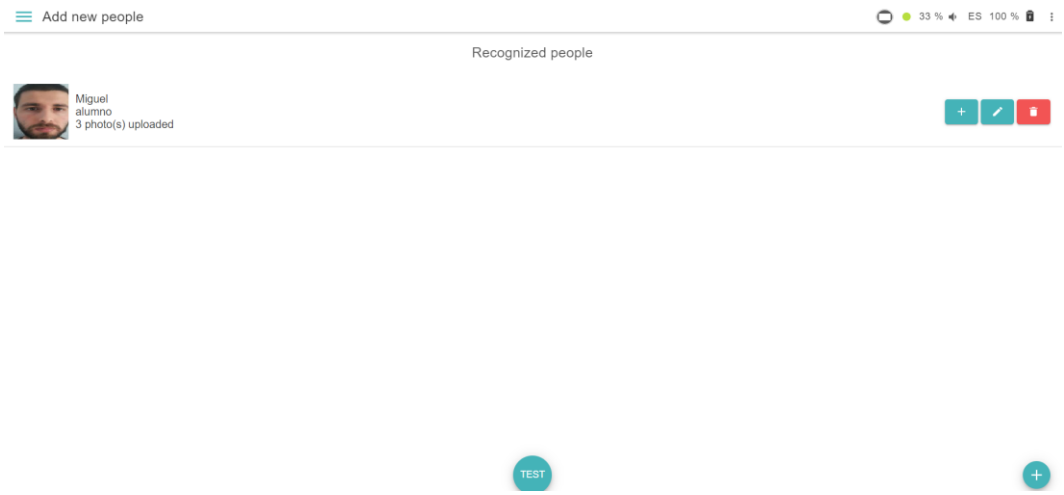

Ilustración 16 Reconocimiento facial

3.2.3.Botón

Como se ha indicado previamente, el robot tiene un botón en la parte superior de la cabeza, el cual nos permite interactuar de una manera cómoda con James.

La configuración de este robot se lleva a cabo en el “Kiosk”, que es como el sistema madre de la tableta que controla la apariencia de pantalla que nos muestra James. Se pueden tener varias configuraciones guardadas de “Kiosk”, tal y como se ve en la *ilustración 17*.

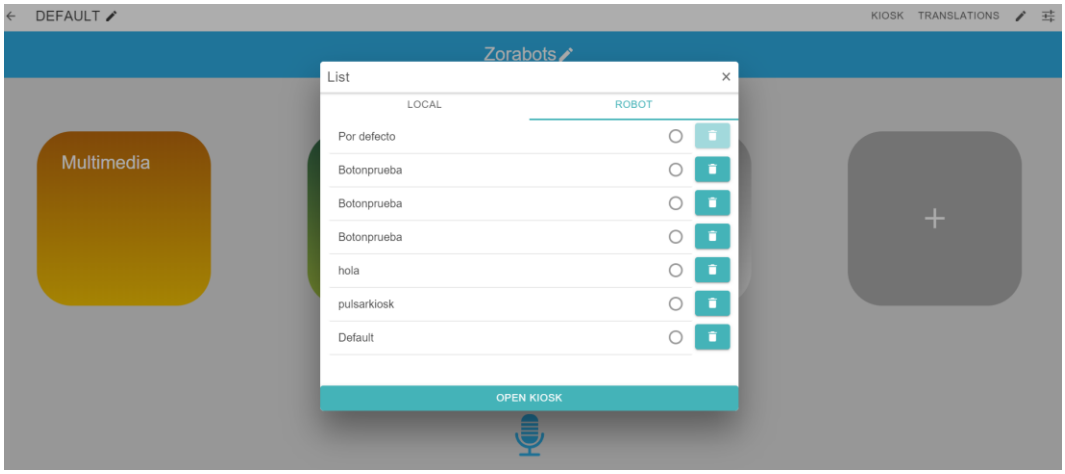

Ilustración 17 Lista de configuraciones de kiosk

Para configurar el robot hay que seguir una serie de pasos, explicados a continuación:

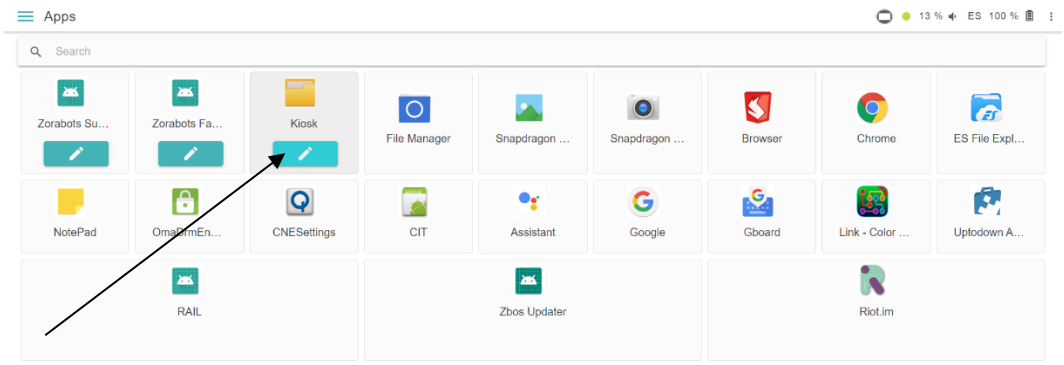

Ilustración 18 Seleccionamos editar Kiosk

17

Entramos en la aplicación de Zorabots, luego pulsamos en la sección Apps y, dentro de esta, en el icono del lápiz, ya que vamos a editar la aplicación, como podemos ver en la *ilustración 18*.

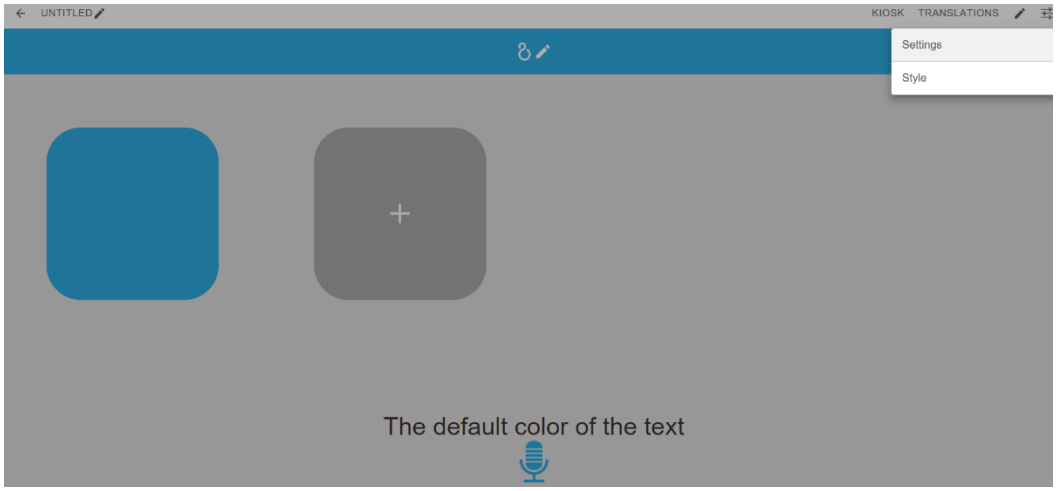

Ilustración 19 Settings Kiosk

Entramos en la zona de *settings*, como se ve en la *ilustración 19,* para habilitar el botón.

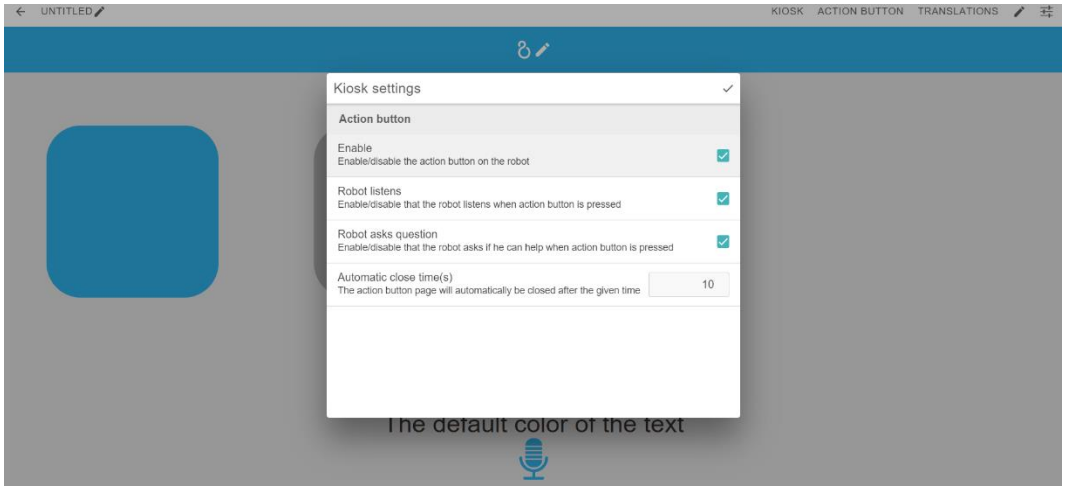

Ilustración 20 Habilitar el botón superior

Activamos las características del botón y podemos ver todas las opciones del robot en la *ilustración 20.*

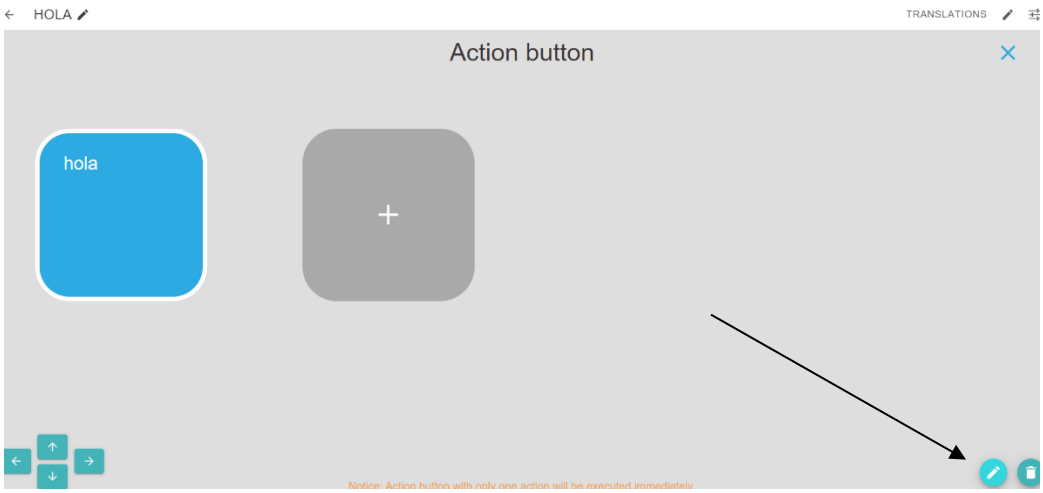

Ilustración 21 Editar apariencia item

Para editar la apariencia del nuevo *Kiosk* se pulsa en el lápiz, como se muestra en la *ilustración 21.*

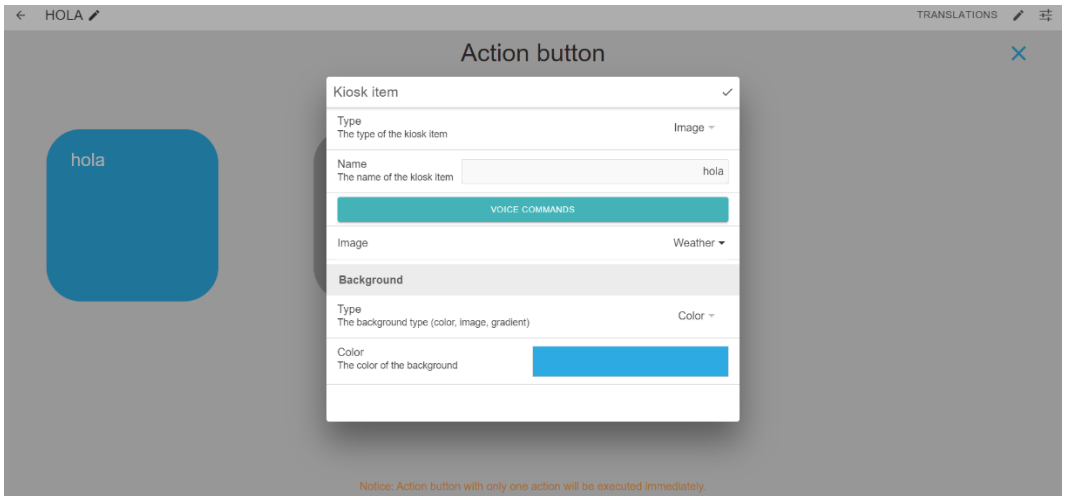

Ilustración 22 Características del item

Un ítem es un elemento del sistema *Kiosk* que podemos modificar. En la imagen (*ilustración 22*) podemos ver todas las características que podemos personalizar. Este se activará cuando se pulse el botón, en este caso al pulsar se mostrará la imagen *Wheather*.

3.3.Aplicación control ZBOS

La aplicación de ZBOS control se puede instalar en cualquier dispositivo Android y también se puede acceder a ella desde la web https://control.zoracloud.com/. Esto permite que desde casi cualquier dispositivo podamos acceder al manejo del robot.

La aplicación tiene las siguientes funcionalidades principales, mostradas en la *ilustración 23.*

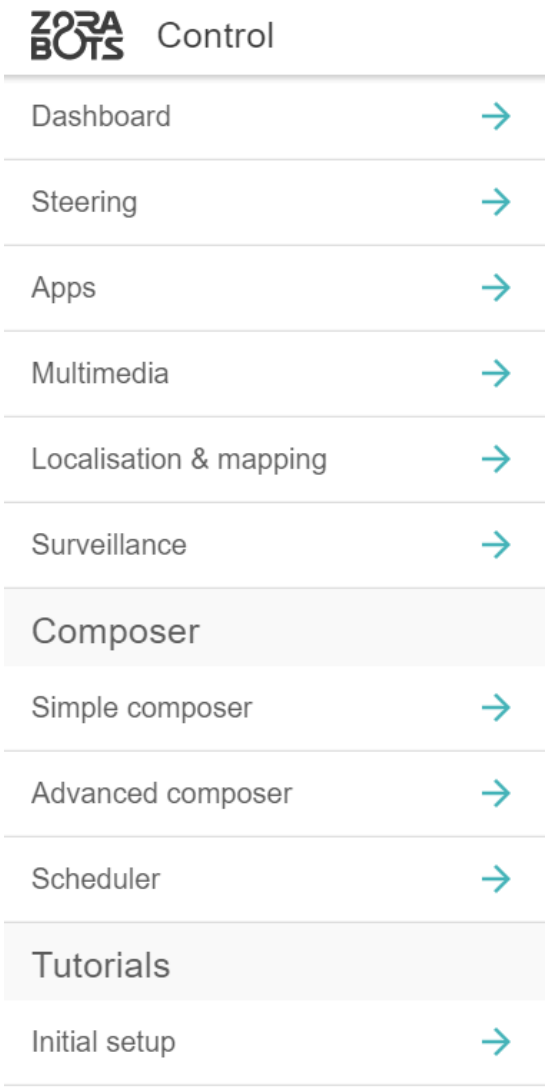

Ilustración 23 Opciones Zorabots aplicación

La aplicación de *Dashboard* funciona como pantalla de control general que nos permite visualizar varias características del robot, así como moverle con un control táctil, como se ve la *ilustración 24*.

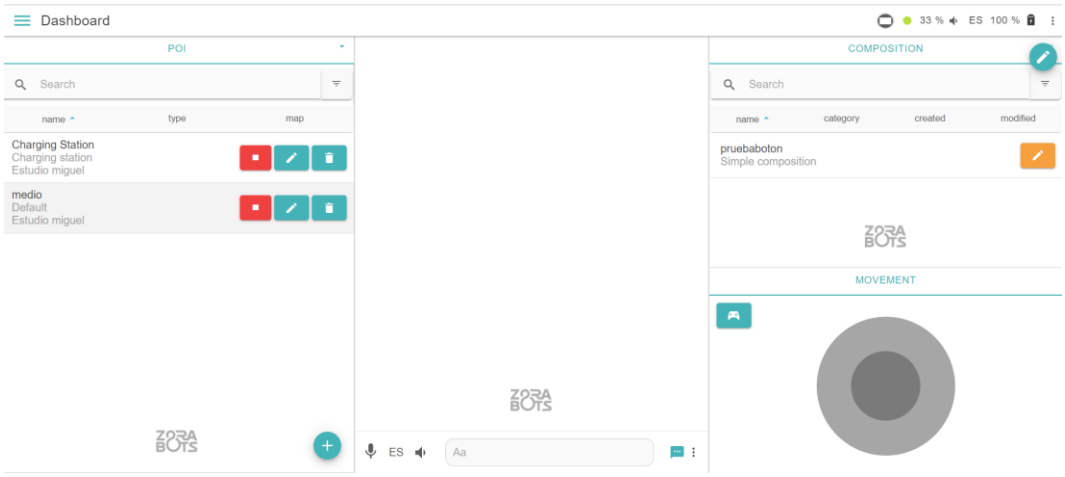

Ilustración 24 Dashboard

La otra aplicación relevante del primer sector de opciones es la de Location and mapping. El resto de las funciones que podemos acceder desde la aplicación son:

- Steering nos permite mover el robot.
- Apps nos muestra las aplicaciones disponibles del robot ya se ha mostrado su apariencia en la ilustración 17.
- Multimedia accede a los archivos de foto y video del robot.
- Surveillance accede a las imágenes que se han podido tomar cuando hemos puesto al robot en modo vigilancia.

La aplicación de mapa no solo permite visualizar el mapa, como se ve la *ilustración 25*, sino también añadir POIs (puntos de interés) y muros para que el robot no acceda a sitios determinados.

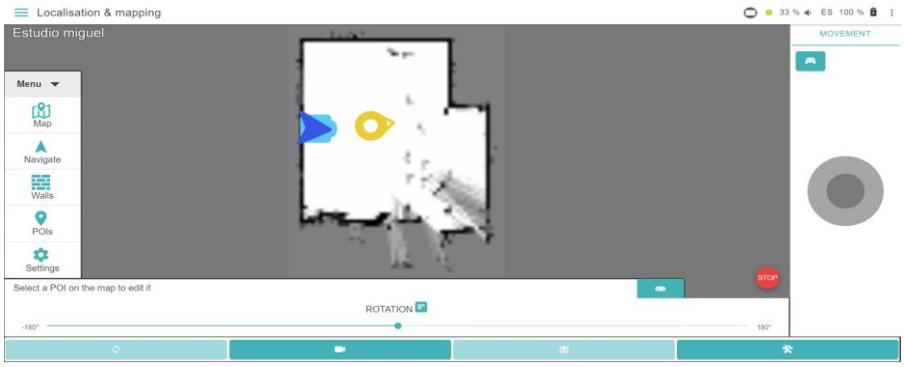

Ilustración 25 Mapa slam James

Por último, la aplicación de Zorabots tiene una función llamada un *Scheduler* que es un planificador que nos permite elegir en que momento del día queremos de realizar una acción. Hay varias opciones que se puede programar la parte mas importante es lanzar composiciones ya que nos permite crear rutinas a horas específicas. Como se puede ver en la *ilustración 26* esta función nos permite elegir a fecha de la acción si esta acción se repite diaria o semanalmente y que tipo de actuación va a realizar el robot

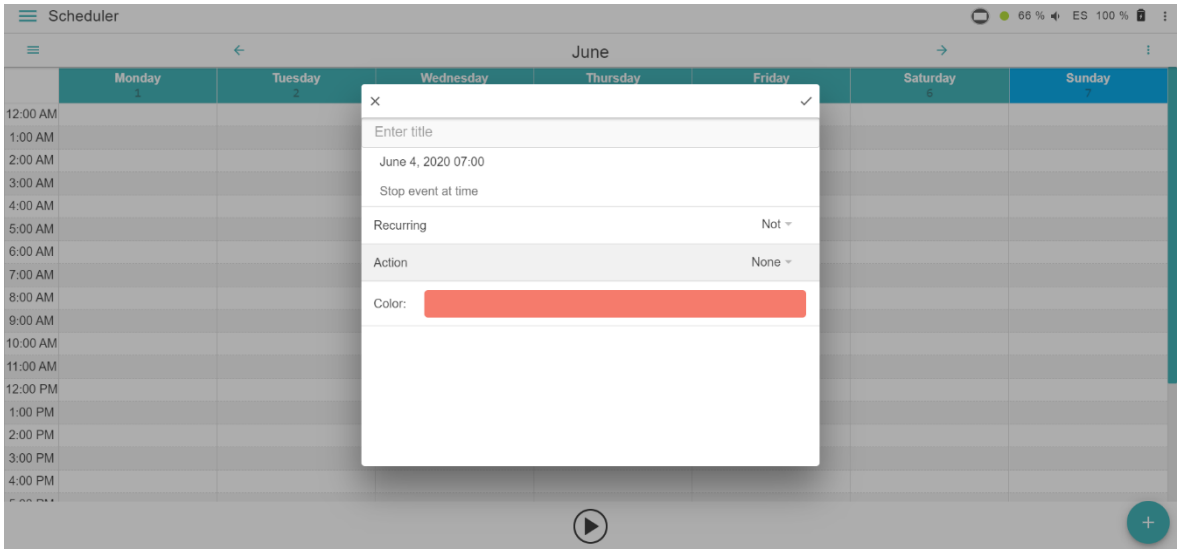

Ilustración 26 Scheduler

4. Protocolo de comunicación (MQTT)

Analizar el formato de comunicación MQTT es necesario en el presente Trabajo de Fin de Grado ya que es el medio de comunicación básico con el robot social James. En este se analizará el cómo funciona la técnica de mensajería MQTT y sus funcionalidades específicas para el robot social James.

4.1. Introducción a la tecnología MQTT.

MQTT significa MQ Telemetry Transport. Es un protocolo ligero de publicación y suscripción de mensajes. Está diseñado principalmente para minimizar el consumo de ancho de banda y que se pueda ejecutar en dispositivos con recursos limitados. Este protocolo es ideal para las tecnologías "machine-to-machine" (M2M) o "Internet of Things", que conectan elementos inteligentes con aplicaciones domóticas con dispositivos móviles. (MQTT.org, 2020)

Estas tecnologías están sufriendo un gran crecimiento en los últimos años porque permiten la conexión de elementos tecnológicos diversos con aplicaciones móviles, con un consumo bajo de potencia y ancho de banda.

Este sistema fue diseñado por Dr Andy Stanford-Clark de IBM y Arlen Nipper de Eurotech en 1999. Surgió para solucionar el problema que existía a la hora de conectar los sensores de los oleoductos a través de satélites, que necesitaban un protocolo que consumiera poca batería y con un ancho de banda pequeño. Sin embargo, en su versión actualizada su uso se ha extendido a diversas tecnologías y campos ver en la *ilustración 27*. OASIS es el consorcio internacional que se encarga de desarrollar los estándares.

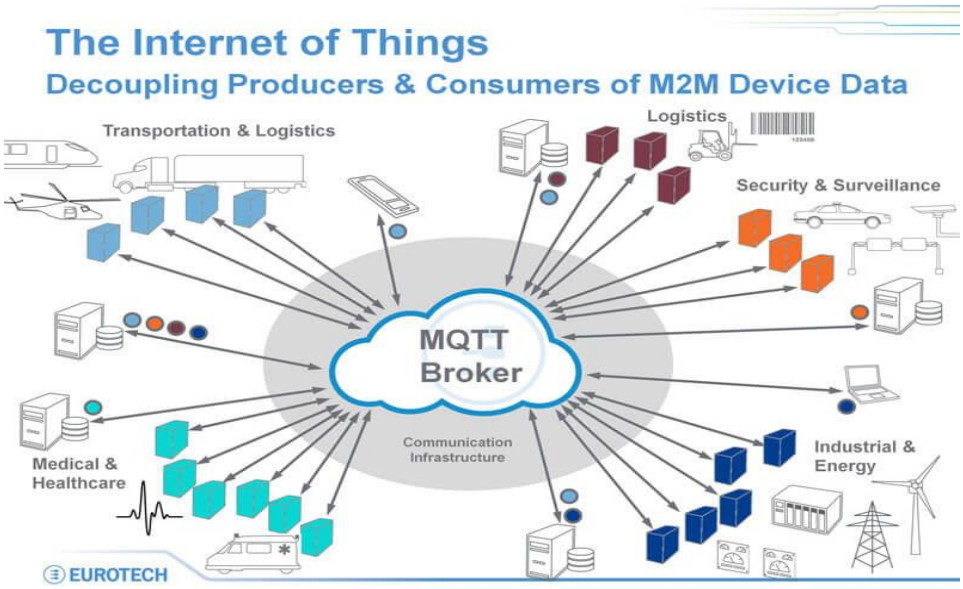

Ilustración 27 MQTT diferentes usos

4.2.Conceptos de la terminología MQTT

Hay dos tecnologías principales para la comunicación por mensajes MQTT y MOM (Message-Oriented Middleware) esta última es una infraestructura cliente/servidor. La principal diferencia en la implementación de MQTT y MOM es que en la segunda los mensajes se almacenan y se entregan. A diferencia de MOM, MQTT no está diseñado para tratar mensajes duraderos y persistentes. MOM está diseñado para la entrega confiable de mensajes entre aplicaciones empresariales. MQTT es un protocolo más simple con solo cinco API diseñadas para conectar dispositivos.

MQTT utiliza el patrón publicación/suscripción para conectar los diferentes dispositivos (*ilustración 28)*. Lo hace desacoplando el dispositivo que envía (publicador) con el receptor (suscriptor). Un dispositivo envía un mensaje con un tópico en el mensaje que se reenvía a todos los dispositivos que estén suscritos a ese tópico. Las publicaciones y las suscripciones son autónomos, lo que significa que no tienen por qué conocer el emisor y el receptor.

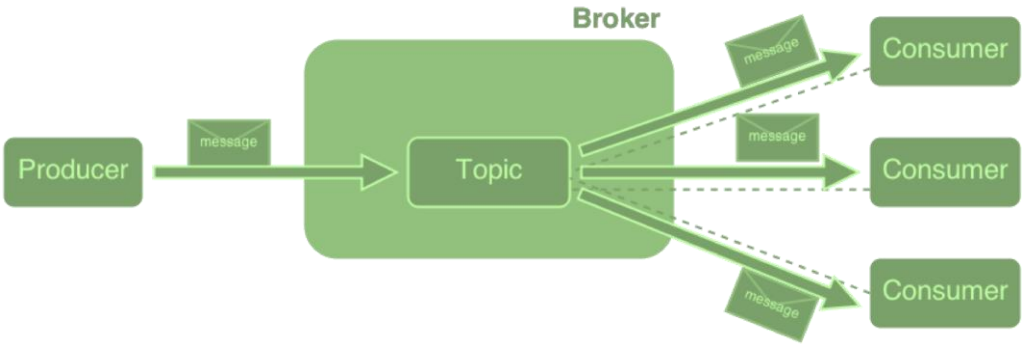

Ilustración 28 Distribución de mensajes

Términos importantes de la tecnología MQTT:

- Cliente: Cualquier dispositivo que publique o se suscriba a un bróker dentro de una red MQTT es considerado cliente. Es importante destacar que quienes publican y se suscriban en una red son considerados clientes y no hay distinción entre ellos, ya que se conectan en forma de estrella al bróker. Los clientes persistentes son los que mantienen una conexión continua con el bróker mientras que los clientes puntuales no están conectados constantemente. Los clientes se suelen conectar mediante librerías SDKs. Hay muchas librerías adaptables a diversos lenguajes, por ejemplo, C, C++, Java, PHP. Python, Node.js y Arduino.

- **Broker**: El Broker es el elemento que recibe todos los mensajes de los clientes cuando estos los publican. Una vez recibidos estos mensajes, los reenvía a los clientes suscritos, y además mantiene la conexión con los clientes persistentes. El Broker es un cuello de botella y por eso un fallo en este haría que todo el sistema fallara. Dependiendo de los implementadores se crean diferentes soluciones, siendo algunas de las más utilizados HiveMQ, Xively, AWS IoT y Loop.

- **Topic**: Un topic de MQTT es un punto en el que se conectan los clientes, que solo tiene que ser conocido por el cliente y el suscriptor. Por tanto, se pueden crear nuevos topics con el tiempo sin necesidad de configurar de nuevo ningún dispositivo. Los topics son cadenas jerárquicas simples, codificadas en UTF-8, delimitadas por una barra diagonal. Por ejemplo, edificio1/habitacion1/temperatura y edificio1/habitacion1/humedad son nombres topics posibles. Una suscripción de edificio1/+/temperatura se suscribirá a todos los topics de temperatura del edificio1. Otra posibilidad es edificio1/#, que conectará con todos los topics del edificio1.

- **Conexión**: MQTT puede ser utilizado por clientes basado en TCP/IP ver la *ilustración 29*. El puerto estándar utilizado para los Brokers es el 1883, que no es un puerto seguro. Los Brokers que soportan TLS/SSL utilizan el puerto 8883. Para la seguridad en la comunicación los clientes y los Brokers deben tener un certificado digital. AWS IoT es una de las implementaciones seguras de MQTTT y requiere a los clientes que utilicen un certificado X.509.

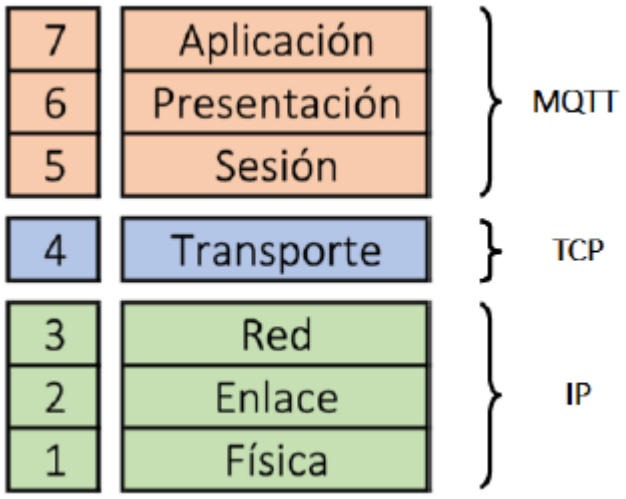

Ilustración 29 MQTT en el modelo OSI

4.3.Protocolo de conexión

El protocolo MQTT proporciona un desacople entre clientes que no saben de la existencia del otro. Esto sucede porque la conexión se establece siempre de cliente a Broker. Para ello, el cliente comienza la comunicación enviándole un mensaje de conexión (CONNECT) al Broker, y este le responde con otro en el que le informa si se ha producido la conexión (CONNACK).

El mensaje CONNECT contiene un ID del cliente a modo de identificador para el Broker, e informa si va a ser una sesión persistente o no. Si es persistente, el Broker guarda las suscripciones y los mensajes perdidos de las suscripciones con QoS. Si se necesita autenticación por parte del Broker, el mensaje contiene el usuario y la contraseña. Sin embargo, estos datos son texto plano y no contienen cifrado, lo que lo hace inseguro. Para solucionar esto se recomienda el uso de TLS o certificados SSL. El mensaje de CONNACK responde con un código (*Tabla 3*) que indica el estado de la conexión del cliente, y si había iniciado una sesión permanente.

Código devuelto	Significado
0	Conexión aceptada
1	Conexión incorrecta. Protocolo erróneo
2	Conexión incorrecta. Identificador rechazado
3	Conexión incorrecta. Servidor no disponible
4	Conexión incorrecta. Usuario o contraseña errónea
5	Conexión incorrecta. No autorizado

Tabla 3 Código de mensajes CONNACK

4.4.Funciones MQTT James

La conexión del robot James con el Broker se puede realizar de varias maneras: mediante la IP 127.0.0.1, que es el del local Broker, que permite el acceso al puerto 8883 y 9001, o mediante el Cloud Broker zbos-mqtt.zoracloud.com. Ninguno de los dos necesita usuario o contraseña (Zorabots, 2020).

A continuación de describen las funciones principales de MQTT que tiene el James, y que son las que utilizaremos en las aplicaciones en capítulos posteriores:

4.4.1.Funciones de diálogo

Empezaremos por las relacionadas con el modo conversación con el robot:

Comando para que James pronuncie un texto. Esto nos permite que el robot diga el texto que nosotros le proporcionemos cuando publiquemos un mensaje (*ilustración 30*)del siguiente tipo:

- Topic: zbos/dialog/set/message
- Payload: Cadena de caracteres

Ejemplo:

Ilustración 30 Ejemplo de zbos/dialog/set/message

Comando para iniciar y parar la escucha del robot. Esto nos permite activar la escucha del robot para que el usuario realice comandos de voz, que se producirán con los siguientes mensajes. El de inicio será el siguiente:

- Topic: zbos/dialog/listen/start
- Payload: No es necesario rellenar este campo

Con la publicación de este mensaje detenemos la escucha:

- Topic: zbos/dialog/listen/stop
- Payload: No es necesario rellenar este campo

Comando para iniciar y parar el dialogo con el robot. Esto nos permite activar el modo diálogo, que crea una pseudo conversación con James. Con la publicación de este mensaje se activará este modo:

- Topic: zbos/dialog/service/start
- Payload: No es necesario rellenar este campo

Con la publicación de este mensaje terminamos el diálogo:

- Topic: zbos/dialog/service/stop
- Payload: No es necesario rellenar este campo

4.4.2.Funciones de audio

Comando para que el robot emita un pitido. Esta función puede parecer simple, pero nos puede permitir localizar al robot de una manera cómoda en ciertas situaciones.

- Topic: zbos/audio/beep
- Payload: No es necesario rellenar este campo

Comando para suscribimos a un topic, que en este caso será un evento, como el cambio de volumen. Esta función nos avisará cuando cambie el volumen del robot James:

- Topic: zbos/audio/volume/evento

4.4.3.Funciones de movimiento

Comando para que el robot se mueva de una determinada manera. Esta función nos permite desplazar el robot con unos parámetros predefinidos (ángulo, fuerza, distancia):

- Topic: zbos/motion/control/movement
- Payload: { “angle”: null, “degree”:90, “force”:100, “distance”:1}

Comando para que el robot se mueva a un determinado POI (punto de interés). Esta función nos permite desplazar el robot hasta un lugar prefijado del mapa, con una cierta velocidad:

- Topic: zbos/slam/poi/moveto/uuid
- Payload: { “mapname”: “string”, “uuid”: “string”, “speed”:1}

Comando para suscribirnos y que el robot nos avise cuando haya tenido un choque. Esta función reacciona a los sensores que tiene en la base el robot.

- Topic: zbos/slam/collision/start/event

- Payload: [{"id":string,"translationkey":string,"type":string}]

Comando para que el robot se mueva a unas determinadas coordenadas. Esta función nos permite desplazar el robot a unas coordenadas del mapa, hay que especificar el mapa porque el robot puede almacenar varios mapas:

- zbos/slam/interaction/moveto
- Payload: {"coordinate": { "x": null, "y": null}, "mapName": "string"}

Comando para que el robot envíe la información sobre su posición. Esta función nos permite localizar el robot en el mapa:

- Topic: zbos/slam/location/current/get
- Payload: {“key”: “ABCxyz}

Comando para suscribirnos al canal que nos proporciona la información de donde se encuentra el robot cuando cambia de posición, es decir, cada vez que cambian las coordenadas del robot se envía un mensaje. Esta función permite localizar el robot en el mapa:

- Topic: zbos/slam/location/current
- Mensaje recibido: {"x": null, "y": null, "rotation": null}

4.4.4.Funciones de mapa

Comando para que el robot nos envíe la información sobre que mapa está utilizando. Esta función nos permite recibir el mapa de la localización de James:

- Topic: zbos/slam/mapview/current/get
- Payload: {“key”: “ABCxyz”}

Comando para suscribirnos al canal que nos proporciona la información obtenida del sistema SLAM que se representa en forma de mapa:

- Topic: zbos/slam/ mapview/current/ ABCxyz
- La información que nos proporciona esta canal la podemos ver en la *ilustración 31*. Los datos que nos proporciona por MQTT James es una imagen en formato base64.

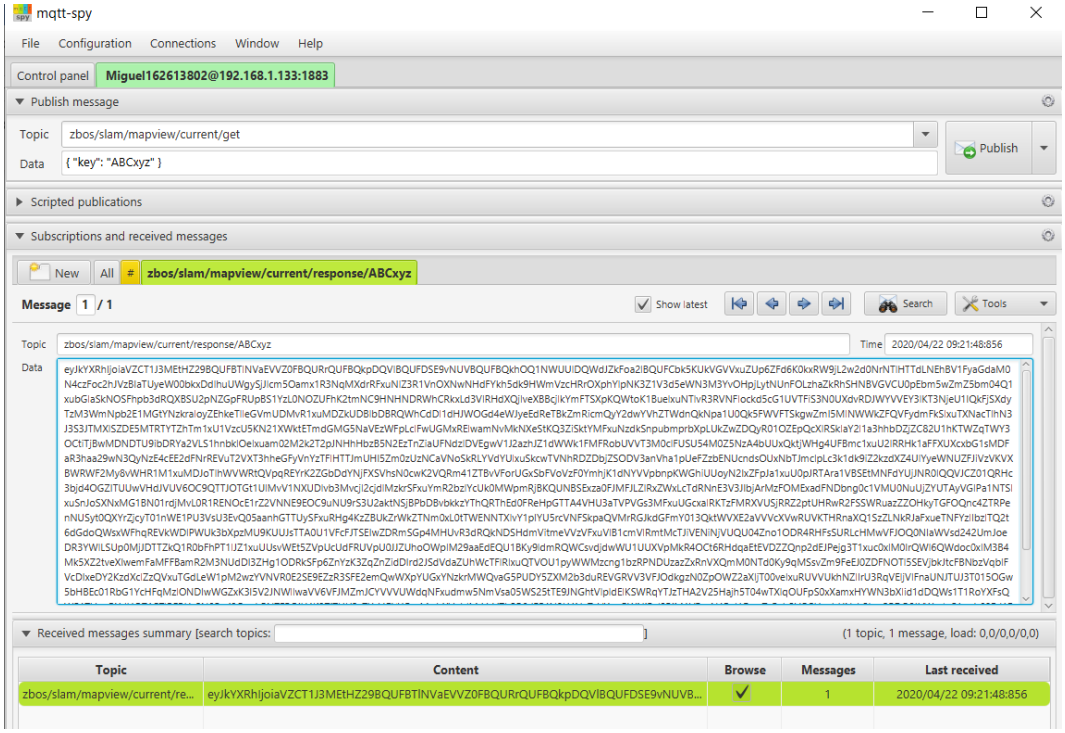

Ilustración 31 Ejemplo obtención del mapa

4.4.5.Funciones de composiciones

Una composición es una cadena de funciones que permite que el robot realice funciones complejas, por ejemplo, que el robot se mueva a una determinada posición, una vez en este lugar que el robot pronuncie un texto o ponga un video.

Comando para suscribirnos al canal que nos indica cuando una composición se ha detenido:

- Topic: zbos/coposition/stop/event/

Comando para suscribirnos al canal que nos dice cuando una composición ha finalizado:

- Topic zbos/composition/end/event

Comando para suscribirnos al canal que nos dice cuando una composición ha tenido un error:

- Topic: zbos/composition/error
- Lista de posibles errores

  - 'INVALID_ID': La identificación del bloque no es correcta o está corrupta.
  - 'INVALID_TYPE': El tipo del bloque no es correcto o está corrupto.
  - 'INVALID_BLOCK': Una de las propiedades del bloque que no son ni el ID ni el tipo no es correcta.
- Forma del mensaje: {"id": "string", "type": "string", "reasons": ["string"]}

Para publicar un mensaje MQTT que inicie una composición primero tenemos que saber qué ID tiene esta. Para ello seguiremos los siguientes pasos:

1º Nos suscribimos a todos los topics con el comando #, como se ve en la *ilustración 32.*

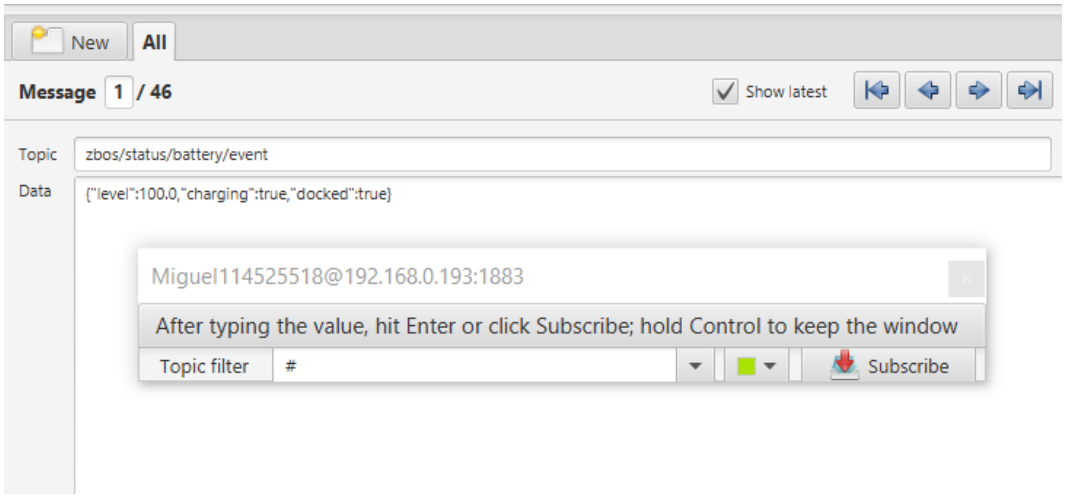

Ilustración 32 Suscripción a todos los topics

2º Publicamos un mensaje MQTT con el topic zbos/composition/list para saber el ID de nuestras composiciones. En la *ilustración 33* podemos ver el ID "_sqwya9yjsw\".

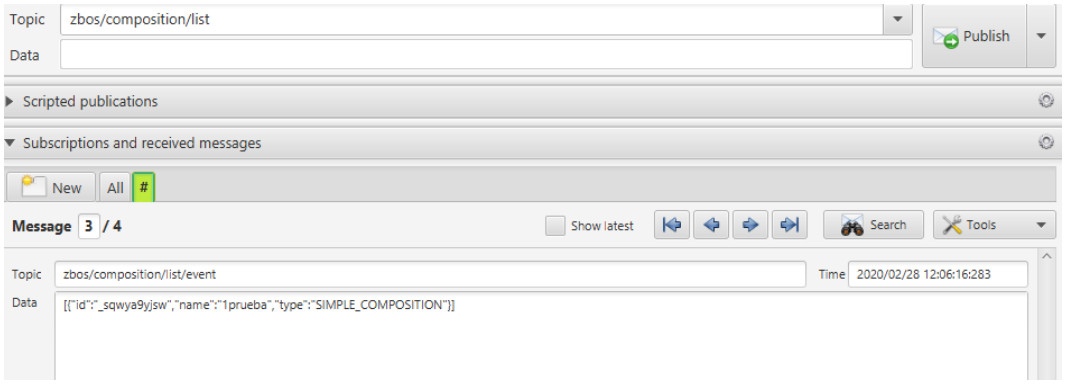

Ilustración 33 Publicación de las id de las composiciones

3º Una vez que sabemos el ID de nuestras composiciones podemos enviar el comando correspondiente por MQTT, que es el siguiente.

- Topic: zbos/composition/start/id
- Payload: {"id":"_sqwya9yjsw"}

De esta manera podemos mandar ejecutar al robot James cualquier composición que tengamos guardada en el robot.

5. Desarrollo de las funciones MQTT en dispositivos móviles

5.1. Introducción

Se ha desarrollado una aplicación para móviles Android con el lenguaje Java ya que puede ser interesante para algunos usuarios, aunque solo se puede ejecutar en un determinado sistema operativo . Este tipo de aplicaciones tiene como positivo que son más visuales. Además, dado que el móvil es algo que la mayoría de las personas tienen a mano constantemente, es un recurso que puede ser muy útil.

Para desarrollar esta aplicación se ha utilizado en el servicio Android de Paho que es una interfaz para la biblioteca del cliente Paho Java MQTT para la plataforma Android. La conexión MQTT está encapsulada dentro de un servicio de Android que se ejecuta en segundo plano de la esta aplicación, manteniéndola activa cuando está cambiando entre diferentes actividades. Esta capa de abstracción es necesaria para poder recibir mensajes MQTT de manera confiable. Como el servicio Paho Android se basa en la biblioteca cliente Paho Java, puede considerarse estable y usarse en producción. El proyecto es mantenido activamente por el proyecto Eclipse Paho. Esta información está en la página https://www.hivemq.com/blog/mqtt-client-library-enyclopedia-paho-android-service/ .

5.2. Build.gradle(app):

Este es el código que se ha introducido en la build.gradle(app) de nuestra aplicación.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"
    defaultConfig {
        applicationId "com.pdhn.mqttdemotfg2"
        minSdkVersion 15
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android-optimize.txt'), 'proguard-rules.pro'
```

```
        }
    }
}
repositories {
    maven {
        url "https://repo.eclipse.org/content/repositories/paho-
releases/"
    }
}
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation
'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.2.0'
    implementation
'androidx.localbroadcastmanager:localbroadcastmanager:1.0.0'

implementation('org.eclipse.paho:org.eclipse.paho.android.service:
1.0.2') {
        exclude module: 'support-v4'
    }
    implementation 'com.android.support:cardview-v7:28.0.0'
}
```

Incluimos en este parte del código lo siguiente:

```
repositories {
    maven {
        url "https://repo.eclipse.org/content/repositories/paho-releases/"
    }
}


dependencies {
    compile('org.eclipse.paho:org.eclipse.paho.android.service:1.0.2') {
        exclude module: 'support-v4'
    }
}
```

Ilustración 34 librerías de Paho (Kock, 2015)

Esto nos permitirá acceder a las funciones de MQTT ya implementadas por Paho (*ilustración 34)* esto hace más fácil el uso de MQTT en Android.

Siempre que incluimos librerías en Android Studio hay que sincronizar para que estas sean hábiles desde los demás apartados del código.

5.3.AndroidManifest.xml

El servicio Android de Paho encapsula la conexión MQTT y ofrece una API para eso. Para poder crear un enlace al servicio Paho Android, el servicio debe declararse en AndroidManifest.xml. Agregue lo siguiente dentro de la etiqueta <application>:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.pdhn.mqttdemotfg2">

    <uses-permission android:name="android.permission.WAKE_LOCK"
/>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.READ_PHONE_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.NoActionBar">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service
android:name="org.eclipse.paho.android.service.MqttService" >
        </service>
    </application>

</manifest>
```

Desde la página de Hivemq nos indican que debemos incluir estas líneas de código que se muestran en la *Ilustración 35* que permite el servicio MQTT

```
<service android:name="org.eclipse.paho.android.service.MqttService" >
</service>
```

Ilustración 35 Service (Kock, 2015)

También es impórtate incluir los permisos (*ilustración 36*)

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

Ilustración 36 Permisos de Android.xml (Kock, 2015)

Con esto indicamos al sistema operativo Android que tenemos permiso para realizar las operaciones y eso nos permitirá acceder a MQTT.

5.4.MainActivity.java

Esta es la parte principal del programa de nuestra aplicación así que trataremos sus diferentes partes de manera individual

5.4.1Cabecera:

```
package com.pdhn.mqttdemotfg2;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import android.view.View;
import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.IMqttActionListener;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.IMqttToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import java.io.UnsupportedEncodingException;
```

En esta cabecera están todas las librerías que son necesarias. Lo bueno que tiene AndroidStudio es que no tienes que colocarlas a priori. Puedes colocar las funciones y si no tienes esa librería te lo indicará poniendo la función en rojo y simplemente con la combinación Alt+Enter se incluye la librería en este encabezado.

5.4.2 Variables globales

```
public class MainActivity extends AppCompatActivity {
    //variables globales
    static String MQTTHOST ="tcp://192.168.1.133:1883";
```

```
    static String topicStr="zbos/dialog/set/message";
    static String topicStr2="zbos/audio/beep";
    static String topicStr3="zbos/composition/start/id";
    static String USERNAME ="";
    static String PASSWORD ="";
```

En este apartado indicamos las variables globales que vamos a utilizar en el programa en nuestro caso la IP y puerto que vamos a utilizar, así como indicamos los topics que luego vamos a implementar.

5.4.3 Cliente de MQTT

```
MqttAndroidClient client;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    String clientId = MqttClient.generateClientId();
    client = new MqttAndroidClient(this.getApplicationContext(),
MQTTHOST, clientId);
    MqttConnectOptions options = new MqttConnectOptions();
```

Para generar un cliente lo que nos dice la página de Paho hay que introducir el código de la *ilustración 37*:

```
String clientId = MqttClient.generateClientId();
MqttAndroidClient client =
        new MqttAndroidClient(this.getApplicationContext(), "tcp://broker.hivemq.com:1883",
                              clientId);

```

Ilustración 37 Creación del cliente MQTT (Kock, 2015)

5.4.4 Conexión:

```
try {
    IMqttToken token = client.connect(options);
    token.setActionCallback(new IMqttActionListener() {
        @Override
        public void onSuccess(IMqttToken asyncActionToken) {
            // Si mqtt conecto
            Toast.makeText(MainActivity.this, "Conectado a Mqtt",
Toast.LENGTH_LONG).show();
            suscripcionTopics();
        }

        @Override
        public void onFailure(IMqttToken asyncActionToken,
Throwable exception) {
            // Si hay error de conexion
            Toast.makeText(MainActivity.this, "Error de conexion",
Toast.LENGTH_LONG).show();
```

```
        }
    });
} catch (MqttException e) {
    e.printStackTrace();}
```

Para ejecutar las funciones Paho nos recomienda incluir las líneas de código que aparecen en la *ilustración 38* para realizar las funciones MQTT:

```
try {
    IMqttToken token = client.connect();
    token.setActionCallback(new IMqttActionListener() {
        @Override
        public void onSuccess(IMqttToken asyncActionToken) {
            // We are connected
            Log.d(TAG, "onSuccess");
        }

        @Override
        public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
            // Something went wrong e.g. connection timeout or firewall problems
            Log.d(TAG, "onFailure");

        }
    });
} catch (MqttException e) {
    e.printStackTrace();
}
```

Ilustración 38 Funciones de MQTT (Kock, 2015)

5.4.5 Callback:

```
client.setCallback(new MqttCallback() {
    @Override
    public void connectionLost(Throwable cause) {

    }

    @Override
    public void messageArrived(String topic, MqttMessage message)
throws Exception {
        //Aqui cuando los mensajes lleguen
        //Card 2

        if(topic.matches("zbos/audio/volume/event")){
            textVol.setText(new String(message.getPayload()));
        }
    }

    @Override
    public void deliveryComplete(IMqttDeliveryToken token) {

    }
});
```

Esta parte del código nos permite reaccionar cuando algo sucede en el canal de mensajes MQTT ya sea por perdida de conexión, llegada de mensaje o la señal de mensaje recibido por el bróker. En la parte de mensajes de llegada

es donde discriminamos los topics a los que nos queremos suscribir. En este caso, por ejemplo, es el de cambio de volumen del robot. Cada vez que llegue un mensaje de este tipo se visualizará por pantalla.

5.4.6 Funciones al pulsar

```
public void handleClick(View v) {
    if (v.getId() == R.id.msg1) {
        String topic = topicStr;
        String message = "Hola mundo, soy james";
        try {
            client.publish(topic, message.getBytes(),0, false);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    } else if(v.getId() == R.id.act1) {
        String topic = topicStr3;
        String message = "{\"id\":\"_sqwya9yjsw\"}";
        try {
            client.publish(topic, message.getBytes(),0, false);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }else if(v.getId() == R.id.vac1) {
        String topic = topicStr;
        String message = "";
        try {
            client.publish(topic, message.getBytes(),0, false);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }else if(v.getId() == R.id.vac2) {
        String topic = topicStr;
        String message = "";
        try {
            client.publish(topic, message.getBytes(),0, false);
        }
        catch (Exception e)
        {
            e.printStackTrace();
       }
    }else if(v.getId() == R.id.beep) {
        String topic = topicStr2;
        String payload = "";
        byte[] encodedPayload = new byte[0];
        try {
            encodedPayload = payload.getBytes("UTF-8");
            MqttMessage message = new MqttMessage(encodedPayload);
            client.publish(topic, message);
        } catch (UnsupportedEncodingException | MqttException e) {
```

```
            e.printStackTrace();
        }}}
```

Estas son las funciones que se pueden ejecutar cuando el usuario se pulsa en la pantalla uno de los cuadros disponibles. Están pensadas para ejecutar topics ya predeterminados así el usuario no tiene que introducir el nombre del topic, solo pulsar en la pantalla.

En este ejemplo se puede ver que hemos implementado varias funciones, recordad que :

```
static String MQTTHOST ="tcp://192.168.1.133:1883";
static String topicStr="zbos/dialog/set/message";
static String topicStr2="zbos/audio/beep";
static String topicStr3="zbos/composition/start/id";
```

La primera manda un mensaje al James para que lo diga, la segunda emite un beep y la última ejecuta una composición previamente carga en el James.

5.4.7 Suscripciones a Topics

```
private void suscripcionTopics(){
    try {
     client.subscribe("zbos/audio/volume/event", 0);
     client.subscribe("zbos/system/robot/identification/event",0);
     client.subscribe("zbos/status/battery/event", 0);
     client.subscribe("zbos/slam/charging/goto/started", 0);
    }catch (MqttException e){
        e.printStackTrace();
    }
}
```

Estas son las funciones de suscripción. De esta forma el cliente pide al bróker que le mande mensajes sobre estos topics. Como hemos visto en la parte de callbacks podemos luego distinguir de donde proviene cada mensaje y actuar en consecuencia.

5.5 Colors.xml

En este archivo se ha introducido una librería que nos proporciona diversos colores. Se ha extraído de la siguiente página web https://github.com/JavierSegoviaCordoba/Resources/blob/master/XML/material_design_colors.xml . Esta es la manera más cómoda de introducir todos los colores sin tener que añadir su código hexadecimal.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#fafafa</color>
    <color name="colorPrimaryDark">#fff</color>
    <color name="colorAccent">#D81B60</color>


    <array name="reds">
        <item>@color/md_red_500</item>
        <item>@color/md_red_50</item>
        <item>@color/md_red_100</item>
        <item>@color/md_red_200</item>
        <item>@color/md_red_300</item>
        <item>@color/md_red_400</item>
        <item>@color/md_red_600</item>
        <item>@color/md_red_700</item>
        <item>@color/md_red_800</item>
        <item>@color/md_red_900</item>
        <item>@color/md_red_A100</item>
        <item>@color/md_red_A200</item>
        <item>@color/md_red_A400</item>
        <item>@color/md_red_A700</item>
    </array>
    <color name="md_red_50">#ffebee</color>
    <color name="md_red_100">#ffcdd2</color>
```

Este parte del código se encarga de proporcionar el código de colores que podemos poner en nuestra aplicación. Se ha optado por insertar una librería ya hecha con muchos tipos de colores, aunque en la red también hay paletas de colores predeterminadas o puedes poner directamente el código hexadecimal del color. Con la librería es más fácil porque podemos elegir el color con un comando sencillo y no tenemos que buscar cada color.

5.6 Strings.xml

```
<resources>
    <string name="app_name">MQTTdemoTFG2</string>
    <string name="nombre">App prueba TFG Miguel</string>
    <string name="boton1">Msg Inical</string>
    <string name="boton2">Act Inicial</string>
    <string name="boton3">Vacio</string>
    <string name="boton4">Vacio</string>
    <string name="boton5">Beep</string>
</resources>
```

Una práctica dentro de la programación de aplicaciones Android es que todo el texto que aparezca en la interfaz este guardado en este archivo. De esta manera se desliga la parte de diseño de la parte de texto y cambiando una vez el texto al nombrar la variable cambiará todas las veces que aparece en la aplicación.

5.7 Activiy_main.xml

Este es parte del código en esta sección que se utiliza para dar la forma visual a la aplicación Android.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/md_green_50"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="180dp"
        android:layout_marginLeft="180dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="180dp"
        android:layout_marginRight="180dp"
        android:layout_marginBottom="600dp"
        android:text="@string/nombre"
        android:textColor="@color/md_black_1000"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0" />

    <TableLayout
        android:id="@+id/tableLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="0dp"
        android:layout_marginLeft="0dp"
        android:layout_marginEnd="180dp"
        android:layout_marginRight="180dp"
        android:layout_marginBottom="262dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">
</androidx.cardview.widget.CardView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Ilustración 39 Aplicación Android de MQTT

Con este código se consigue implementar el aspecto visual de nuestra aplicación que se puede apreciar en la *Ilustración 39* . Determinamos el tamaño de los objetos, así como su colocación y su posición. También creamos una jerarquía donde la pantalla se divide en diferentes bloques y dentro de esos bloques colocamos cuadros de texto o botones. Esto nos permite ser organizados y distribuir mejor el espacio de la pantalla.

6. Desarrollo de las funciones MQTT en lenguaje HTML

6.1. Introducción

En este capítulo se va a abordar el problema de como introducir funciones MQTT en una página HTML. Hacemos esto porque el código HTML es un código estándar y se puede ejecutar desde cualquier dispositivo. Cualquier navegador desde Chrome, Mozilla o Safari puede abrir este tipo de código y ejecutarlo de forma correcta. Esto permite que sea accesible a cualquier usuario independientemente del dispositivo que tenga.

Para poder implantar este protocolo hemos seguido las indicaciones mostradas en la página web http://www.steves-internet-guide.com/using-javascript-mqtt-client-websockets/

Las características del código están explicadas en la *ilustración 40* donde se puede ver las partes básicas del código JavaScript. El código JavaScript se encaja con el código HTML perfectamente para generar una página de web funcional.

Además, este código utiliza websockets para aplicar las conexiones MQTT. MQTT sobre websockets le permite a la aplicación HTML recibir datos MQTT directamente en un navegador web. Esto es importante ya que el navegador web puede convertirse en la interfaz de facto para mostrar datos MQTT. El cliente JavaScript proporciona el soporte de MQTT websocket para navegadores web.

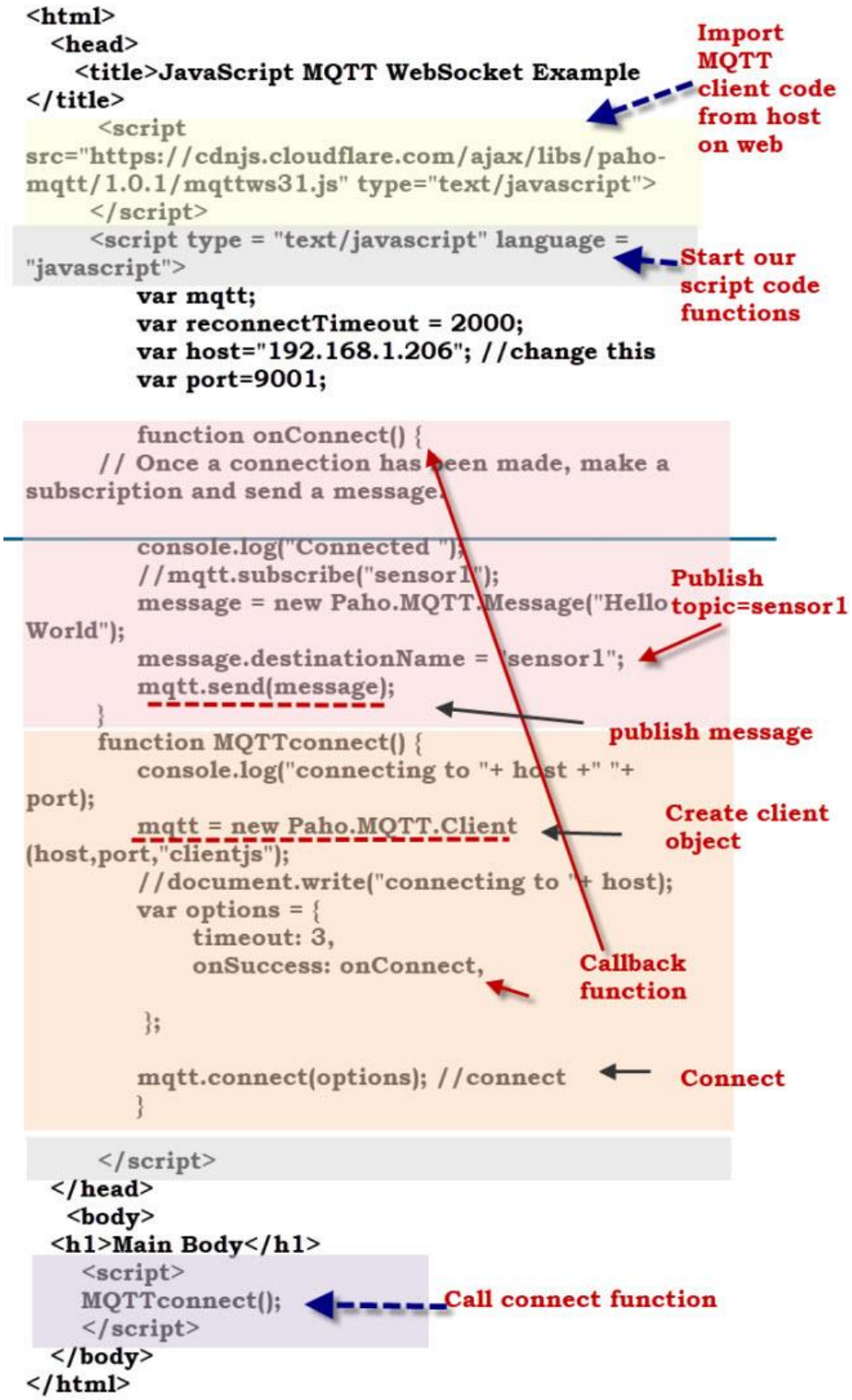

Ilustración 40 Código de javascript con websocket (Martin, 2020)

6.2.Cabecera del código

```
    <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-
mqtt/1.0.1/mqttws31.js" type="text/javascript"></script>
    <script type = "text/javascript"
         src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/j
query.min.js"></script>
    <script type = "text/javascript" src="myScript.js"></script>
```

Estas librerías nos permiten acceder a las funciones MQTT de una manera más cómoda y eficiente.

6.3.Función para conectarse:

```
function MQTTconnect() {
    document.getElementById("messages").innerHTML ="";
    var s = document.forms["connform"]["Serv"].value;
    var p = document.forms["connform"]["Port"].value;
    var SSL= document.getElementById("SSL_act").checked;
    if (p!="")
    {
    console.log("ports");
        port=parseInt(p);
        console.log("port" +port);
        }
    if (s!="")
    {
        host=s;
        console.log("host");
        }
    console.log("connecting to "+ host +" "+ port);
    mqtt = new Paho.MQTT.Client(host,port,"clientjsaaa");
    var options = {
        timeout: 3,
        onSuccess: onConnect,
        onFailure: onFailure,
        useSSL:SSL,

     };
    mqtt.onConnectionLost = onConnectionLost;
    mqtt.onMessageArrived = onMessageArrived;
    mqtt.onConnected = onConnected;
    mqtt.connect(options);
    return false;}
```

En primer lugar, se limpia la zona de mensajes, lo que nos permite iniciar una conexión limpia con tan solo los mensajes de esta. En este caso hemos introducido la IP y el puerto desde el código, pero se podría incluir en la página web para que lo introduzca el usuario.

Luego se crea el cliente de MQTT, que es la función que va a realizar la página web. El nombre del cliente no es importante porque solo se verá reflejado en los datos internos del Broker.

Después indicamos el tipo de conexión que queremos realizar, y unimos las funciones que luego explicitaremos con las funciones de MQTT, por último, iniciamos la conexión.

6.4.Función de perdida de conexión

```
function onConnectionLost(){
   console.log("connection lost");
   document.getElementById("status").innerHTML = "Connection Lost";
   document.getElementById("messages").innerHTML ="Connection Lost";
    connected_flag=0;
}
```

Esta función nos permite identificar cuándo se ha producido una desconexión del Broker, mostrándolo por pantalla, además de desactivar la connected_flag, que es la que permite realizar las funciones cuando existe conexión.

6.5.Función cuando sucede un fallo al conectarse

```
function onFailure(message) {
        console.log("Failed");
        document.getElementById("messages").innerHTML = "Connection
Failed- Retrying";
        setTimeout(MQTTconnect, reconnectTimeout);
        }
```

Si al intentar conectarnos con el broker no conseguimos dicha conexión, saltamos a esta función, que además resetea el timer para volver a intentar conectarse.

6.6.Función de llegada de mensaje

```
function onMessageArrived(r_message){
        out_msg="Message received "+r_message.payloadString+"<br>";
        out_msg=out_msg+"Message received Topic "+r_message.destinat
ionName;
        console.log(out_msg);
```

```
        document.getElementById("messages").innerHTML =document.getE
lementById("messages").innerHTML+"<br>"+out_msg;
        if(r_message.destinationName=="zbos/slam/mapview/current/res
ponse/all")
        {obtencionmapa(r_message.payloadString)}
        if(r_message.destinationName=="zbos/slam/location/current")
        {obtencioncoor(r_message.payloadString)}
    }
```

Esto nos permite organizar las llegadas de mensajes del Broker, que iremos mostrando al cliente según lleguen.

6.7.Función de después de la conexión

```
function onConnect() {
    document.getElementById("messages").innerHTML ="Connected to "+h
ost +"on port "+port;
    connected_flag=1
    document.getElementById("status").innerHTML = "Conectado";
    console.log("on Connect "+connected_flag);
      }
```

En esta función debemos marcar en la sección de mensajes que estamos conectados, y en qué puerto. Además, también lo indicamos en la sección de estatus y por consola. Lo más importante es activar la bandera de conectado, que nos dará la posibilidad de realizar las funciones de publicación y suscripción.

6.8.Función para subscribirse a un topic dado por el usuario

```
    function sub_generico(){
        document.getElementById("messages").innerHTML ="";
        if (connected_flag==0){
            out_msg="<b>Not Connected so can't subscribe</b>"
            console.log(out_msg);
            document.getElementById("messages").innerHTML = out_msg;
            return false;
        }
        var stopic= document.forms["subs"]["Stopic"].value;
        console.log("Subscribing to topic1 ="+stopic);
        mqtt.subscribe(stopic);
        return false;
    }
```

En primer lugar, debemos comprobar que estamos conectados al broker de manera correcta porque en caso contrario esta función carecería de sentido.

Después extraemos la información que el usuario ha escrito en la página web, que nos indicará el topic al que suscribirse.

6.9.Función para mandar un topic elegido por el usuario

```
function Topic_generico(){
    document.getElementById("messages").innerHTML ="";
    if (connected_flag==0){
    out_msg="<b>Not Connected so can't send</b>"
    console.log(out_msg);
    document.getElementById("messages").innerHTML = out_msg;
    return false;
    }
    var msg = document.forms["smessage"]["message"].value;
    console.log(msg);

    var topic = document.forms["smessage"]["Ptopic"].value;

    message = new Paho.MQTT.Message(msg);
    if (topic=="")
        message.destinationName = "test-topic"
    else
        message.destinationName = topic;
    mqtt.send(message);
    return false;
}
```

Al igual que en la función anterior, primero comprobamos que estamos bien conectados al Broker, y extraemos tanto la información del topic como el mensaje que se quiere mandar.

Primero traducimos el mensaje dado por el usuario con la función Paho.MQTT.Message(msg), para que sea entendible por el Broker, y añadimos un mensaje por si el usuario no rellena el topic, tras lo que publicamos el mensaje al Broker de MQTT.

6.10.Función para publicar topics de entre una lista

```
function pub_topic(topics,mensajetopic){
        document.getElementById("messages").innerHTML ="";
        if (connected_flag==0){
        out_msg="<b>Not Connected so can't send</b>"
        console.log(out_msg);
        document.getElementById("messages").innerHTML = out_msg;
        return false;
```

```
        }
        var msg = mensajetopic;
        console.log(msg);
        var topic =topics;
        message = new Paho.MQTT.Message(msg);
        if (topic=="")
            message.destinationName = "test-topic"
        else
            message.destinationName = topic;
        mqtt.send(message);
        return false;
    }
```

Es una función muy similar a la anterior de publicación, pero en este caso la información del mensaje y del topic se extrae de una variable que elige el usuario en vez de extraer un texto como en el anterior caso.

6.11.Función para subscribirse a topics dentro de una lista.

```
function sub_topics(topico){
    document.getElementById("messages").innerHTML ="";
        if (connected_flag==0){
        out_msg="<b>Not Connected so can't subscribe</b>"

        console.log(out_msg);
        document.getElementById("messages").innerHTML =out_msg;
        return false;
        }
    var stopic= topico;
    console.log("Subscribing to topic ="+stopic);
    mqtt.subscribe(stopic);
    return false;
    }
```

Es una función muy similar a la anterior de suscripción, pero en este caso la información del mensaje y del topic se extrae de una variable que elige el usuario en vez de extraer un texto como en el anterior caso.

6.12.Función actualizar el mapa de localización

```
<div id="mapa">
       <img id="mimapa" src="data:mapa/png;base64,iVBORw0KGgoAAAANSU
hEUgAAADkAAABJCAYAAACHOo5EAAAABHNCSVQICAgIfAhkiAAABnNJREFUeJzdWz+I1E

…
```

```
oc/l6wCk59GM7K4HAWQrhgZ3Cxs1hshIW0Hi52ym4nLCvXnQh2J2ernN9Varho2UK1rS
rJyhlPU1glr1a4MmcGqmxGXRmSVfAfZO5+BPUFXk4AAAAASUVORK5CYII003d"
            alt="Mapa Estudio Miguel" width="400" height="400" >
 </div>
```

Este es el código en base64 que se obtiene al pedir por comando MQTT el mapa al robot James. Esta forma de mandarnos la imagen en forma de código alfanumérico nos permite mandar por MQTT imágenes ya que este protocolo de mensajería solo permite mandar mensajes en forma de texto.

El mapa se puede modificar porque el robot James analiza constantemente su entorno. Por lo tanto, podemos pedir que se actualice el mapa cuando pulsamos un botón:

```
<button onclick="refrescarmapa()">refrescar mapa</button>
```

Esto nos lleva a las siguientes funciones:

```
function refrescarmapa(){
    suscripcionmapa();
    document.getElementById("messages").innerHTML ="";
    if (connected_flag==0){
        out_msg="<b>Not Connected so can't send</b>"
        console.log(out_msg);
        document.getElementById("messages").innerHTML = out_msg;
        return false;
    }
    var topic ="zbos/slam/mapview/current/get";
    var msg = "{“key”: “ABCxyz”}";
    console.log(msg);
    message = new Paho.MQTT.Message(msg);
    if (topic=="")
        message.destinationName = "test-topic"
    else
        message.destinationName = topic;
    mqtt.send(message);
    return false;
}
function obtencionmapa(datosmapa){

    var res = datosmapa.slice(9,-29);
    res= res.replace(/\\n/g, '');
    res= res.replace(/\\u/g, '');

    imagen=""+res;
    document.getElementById("mimapa").src = imagen;
```

```
        document.getElementById("messages").innerHTML ="Mapa actuali
zado";
    }
    function suscripcionmapa(){

            if (connected_flag==0){
            out_msg="<b>Not Connected so can't subscribe</b>"

            console.log(out_msg);
            document.getElementById("messages").innerHTML =out_msg;
            return false;
            }

        var stopic= "zbos/slam/mapview/current/response/all";
        console.log("Subscribing to topic ="+stopic);
        mqtt.subscribe(stopic);
        return false;
        }
```

Lo primero que hacemos es suscribirnos al topic que nos permite recibir los mensajes que nos manda el James con su mapa actual. La función suscripcionmapa es idéntica a las vistas anteriormente con la única diferencia del topic al que se suscribe.

Una vez suscritos a ese canal de información publicamos el comando que hace que el robot nos mande su mapa actual. Esta parte de la función refrescarmapa es igual a las vistas anteriormente que servían para publicar en MQTT.

Por último, cuando llegue un mensaje del canal "zbos/slam/mapview/current/response/all" es cuando entramos en la función obtención mapa, que lo que hace es transformar la cadena de caracteres del mensaje del robot para extraer la imagen del mapa. Una vez extraída se cambia la imagen que se muestra en la página web.

6.13.Función para actualizar la posición del robot James

Para mostrar la posición del James se va a utilizar un punto rojo que se moverá por la imagen del mapa según las coordenadas que recibamos por la comunicación MQTT.

```
<div id="punto">
    <img alt="puntorojo" style="position:absolute; left: 200px; top:
 80px;" width="15" height="15" src="
oAAAANSUhEUgAAAAEAAAABCAIAAACQd1PeAAAADElEQVR42mP4z8AAAAMBAQD3A0FDAA
AAAElFTkSuQmCC" >
```

```
</div>
```

Este es el código que nos permite visualizar el punto rojo en la página web y cuando pulsemos el botón “mostrar robot” nos empezará a localizar al robot en el mapa.

```
<button onclick="robotxy()">mostrar robot</button>
    function robotxy(){
        if (connected_flag==0){
                out_msg="<b>Not Connected so can't subscribe</b>"
            console.log(out_msg);
            document.getElementById("messages").innerHTML =out_msg;
                return false;
        }

        var stopic= "zbos/slam/location/current";
        console.log("Subscribing to topic ="+stopic);
        mqtt.subscribe(stopic);
        return false;
    }
```

Esta función simplemente nos suscribe al canal de información del James que manda datos cada vez que cambia sus coordenadas, lo que nos permitirá actualizar de manera automática la posición del robot cada vez que cambian las coordenadas.

```
    function obtencioncoor(datoscoor){
        datoscoor=datoscoor.slice(26);
        var xi = datoscoor.search(/x/i);
        xi=xi+3;
        var xf=xi+2;
        var yi = datoscoor.search(/y/i);
        yi=yi+3;
        var yf=yi+2;
        x = datoscoor.slice(xi,xf);
        y = datoscoor.slice(yi,yf)
        var coorx = parseInt(x)*5;
        var coory= parseInt(y)*10.3-100;
        document.getElementById("punto").style.position= "absolute";
        document.getElementById("punto").style.top= coory+"px";
        document.getElementById("punto").style.left= coorx+"px";
        document.getElementById("messages").innerHTML =document.getE
lementById("messages").innerHTML+"<br>"+"X: "+x+" Y:  "+y;}
```

Esta es la función que se activa cuando se recibe un mensaje del topic "zbos/slam/location/current". Como este mensaje tiene más información que

las coordenadas del robot lo que hacemos es manipular la cadena de caracteres para obtener las coordenadas X e Y. Una vez obtenidas lo que cambiamos es la posición del robot con las funciones:

- document.getElementById("punto").style.top=coory+355+"px"
- document.getElementById("punto").style.left= coorx+20 +"px"

Así conseguimos transformar los datos que nos envía James en un punto visible en el mapa de la aplicación web.

6.14.Función para hacer un mapa interactivo

```
document.addEventListener("click", function(){
        if(event.clientX>220 && event.clientX<470){
            if(event.clientY>320 && event.clientY<590){
                //document.getElementById("messages").innerHTML =  "
clientX: " + event.clientX + " - clientY: " + event.clientY;
                var x= (event.clientX -90)/11.5;
                var y=(event.clientY+100)/10.3;
                var coorx = parseInt(x);
                var coory= parseInt(y);
                var topic ="zbos/slam/interaction/moveto";
                var msg = "{\"coordinate\": {\"x\":"+ coorx + ",\"y\
":"+coory+" }, \"mapName\": \"Estudio miguel\"}";
                document.getElementById("messages").innerHTML =  doc
ument.getElementById("messages").innerHTML + msg+"\n";
                message = new Paho.MQTT.Message(msg);
                message.destinationName = topic;
                mqtt.send(message);


                return false;
            }
        }
    });
```

Esta función tiene como objetivo que cuando se clique en el mapa se manda las coordenadas donde el robot debe localizarse. Lo primero que se hace es poner los límites para que solo se ejecute cuando se pulsa encima de la imagen del mapa. Luego se hace una conversión para que la posición del ratón se transforme en las coordenadas para el robot. Una vez que tenemos las coordenadas solo es necesario publicar un mensaje del tipo "{"coordinate": {"x":"+ coorx + ","y":"+coory+" }, "mapName":"Estudio miguel"}"; en el topic "zbos/slam/interaction/moveto".

6.15.Función para comprobar la posición de destino a un POI

```
    function composicion(){
        document.getElementById("messages").innerHTML ="hola";
        var msg = "{\"mapName\": \"Mapacompo\",\"uuid\": \"03a07a6c-
f672-4c07-966e-8264135d758d\",\"speed\": 75 }";
        console.log(msg);

        var topic ="zbos/slam/poi/moveto/uuid";
        message = new Paho.MQTT.Message(msg);
        if (topic=="")
            message.destinationName = "test-topic"
        else
            message.destinationName = topic;
        mqtt.send(message);
        return false;

    }
```

Esta función es la que manda el mensaje para que inicie el recorrido. En nuestro caso se hace al pulsar un botón y su funcionamiento es muy simple se manda un mensaje preprogramado con el código de identidad del POI (punto de interés) al que queremos hacer llegar al robot.

```
    function fincompo(datoscoor){
        datoscoor=datoscoor.slice(26);
        var xi = datoscoor.search(/x/i);
        xi=xi+3;
        var xf=xi+3;
        var yi = datoscoor.search(/y/i);
        yi=yi+3;
        var yf=yi+3;
        x = datoscoor.slice(xi,xf);
        y = datoscoor.slice(yi,yf)

        document.getElementById("messages").innerHTML =document.getE
lementById("messages").innerHTML+"<br>"+"X: "+x+" Y:  "+y;
        var coorx = parseInt(x);
        var coory= parseInt(y);
        if(coorx>140 && coorx<160 && coory>90 && coory<100){
            document.getElementById("messages").innerHTML ="Exito: X
: "+x+" Y:  "+y;
        }
        else{
```

```
            document.getElementById("messages").innerHTML ="Fracaso
X: "+x+" Y:  "+y;
            var msg = "{\"mapName\": \"Mapacompo\",\"uuid\": \"fc3ec
255-ce31-4d89-90be-e1e3859cfd3c\",\"speed\": 75 }";
            console.log(msg);

            var topic ="zbos/slam/poi/moveto/uuid";
            message = new Paho.MQTT.Message(msg);
            if (topic=="")
                message.destinationName = "test-topic"
            else
                message.destinationName = topic;
            mqtt.send(message);

        }
        return false;
    }
```

Esta función se activa cuando el robot manda por MQTT el mensaje de finalización del recorrido. Cuando esto pasa accedemos a la última posición del robot que en la función se representa por la variable “datoscoor”. Extraemos de esta información en bruto la parte importante que son el numero de coordenadas en x e y, después con la instrucción “if” valoramos si es la posición correcta o no y actuamos en consecuencia. Si es correcto publicamos un mensaje de éxito y si no lo es además de publicar un mensaje de fallo reubicamos al robot en otra posición.

7. Implementación de un soporte para la tableta

El robot James no puede realizar videollamadas con voz debido a que el micrófono se dedica en exclusiva a la escucha de los comandos de voz. Esto debe ser implementado ya que una de las funciones principales que debe tener este robot es facilitar el contacto de personas de avanzada edad con gente de su entorno o con personal sanitario. Se ha propuesto una posible solución.

Para solucionar este problema se va a instalar una tableta adjuntada en la parte superior del robot, que nos permite realizar videollamadas con otras personas de una manera cómoda, dejando la tableta principal del robot para otras funciones, como puede ser aplicaciones elegidas por el usuario o funciones ya instaladas, como la realización de encuesta. Para ello lo primero fue diseñar un modelo 3D para poder realizar los diseños adyacentes (ver *ilustración 41)*.

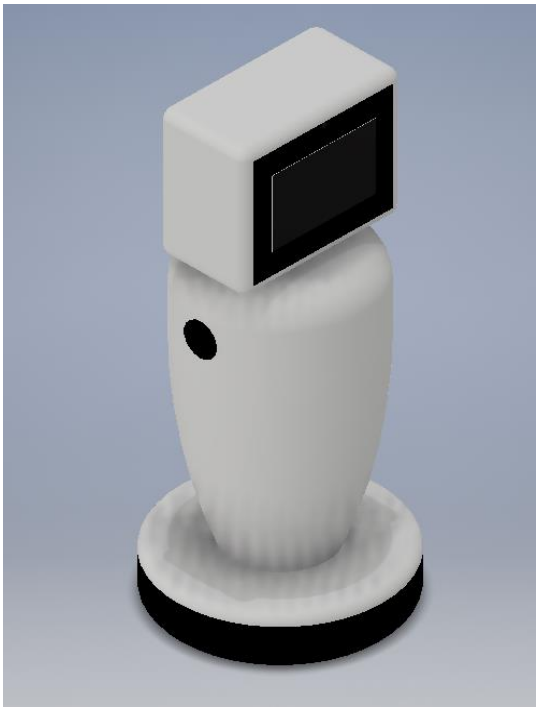

Ilustración 41 Diseño 3D robot social James

7.1.Samsung Tab A

Se ha elegido la tableta Samsung Galaxy Tab A de 8 pulgadas ya que tiene un formato compacto y cumple con la función principal: la posibilidad de realizar videollamadas. Las características de la tableta son:

- Procesador:
  - Velocidad CPU: 2GHz

  - Tipo CPU: Quad-Core
  - Pantalla: Tamaño 8.0" (203.1mm)
  - Resolución: 1280 x 800 (WXGA)
  - Tecnología: TFT
  - Número de colores: 16M
- Cámara
  - Cámara principal – Resolución 8.0 MP
  - Cámara principal - Autofocus
  - Cámara frontal – Resolución 2.0 MP
  - Resolución de grabación de vídeo FHD (1920 x 1080)@30fps
- Memoria
  - RAM 2 (GB) 2
  - Memoria Interna 32 (GB)
  - Memoria Disponible 21.3 (GB)
  - Externa MicroSD (hasta 512GB)
- Sistema Operativo: Android
- Sensores Acelerómetro, Luminosidad
- Especificaciones físicas
  - Dimensiones (AlxAnxProf, mm) 210.0 x 124.4 x 8.0
  - Peso 345 (g)
- Capacidad de batería 5100 (mAh, típico)

Se ha creado una réplica 3D (*ilustración 42*) con el Autodesk Inventor para diseñar posibles diseños del soporte que se enganchara al James.

Ilustración 42 Samsung Tab A

7.2.Soporte tableta

Se ha diseñado un soporte con el objetivo de integrar la tableta de una manera sencilla al James. Se ha colocado en la parte superior porque es la zona más cómoda para establecer un contacto visual si se realiza una videollamada. Además, en la parte inferior de la cabeza del robot hay un conector USB que nos permite dar alimentar a la tableta y así no es necesario sacarla del soporte para realizar la carga.

El objetivo es fabricar el soporte con una impresora 3D para ello se han hecho un diseño inicial, que se puede ver en las *ilustraciones 43 y 44.*

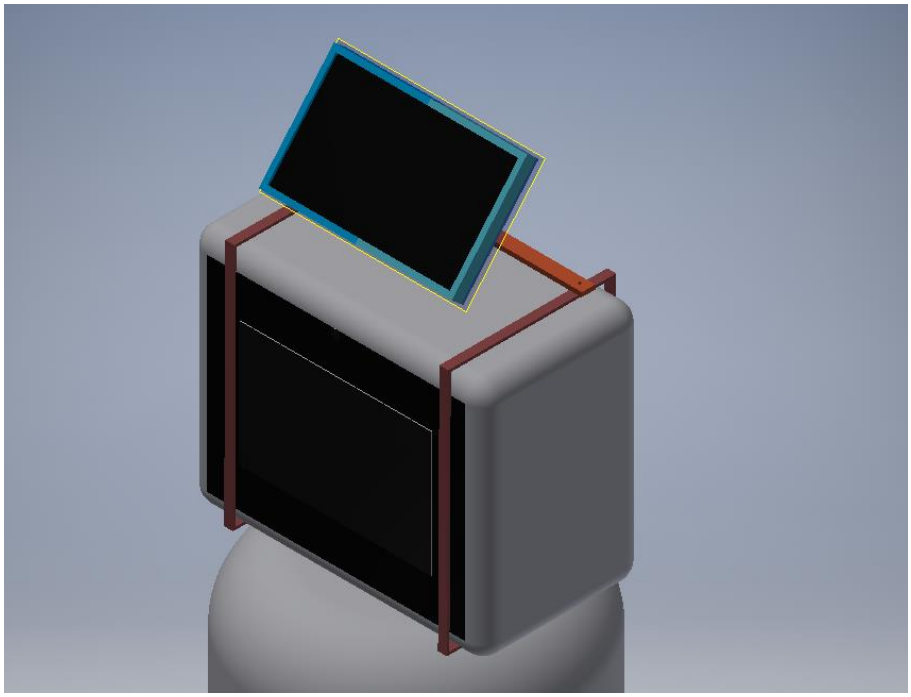

Ilustración 43 Primer diseño soporte frontal

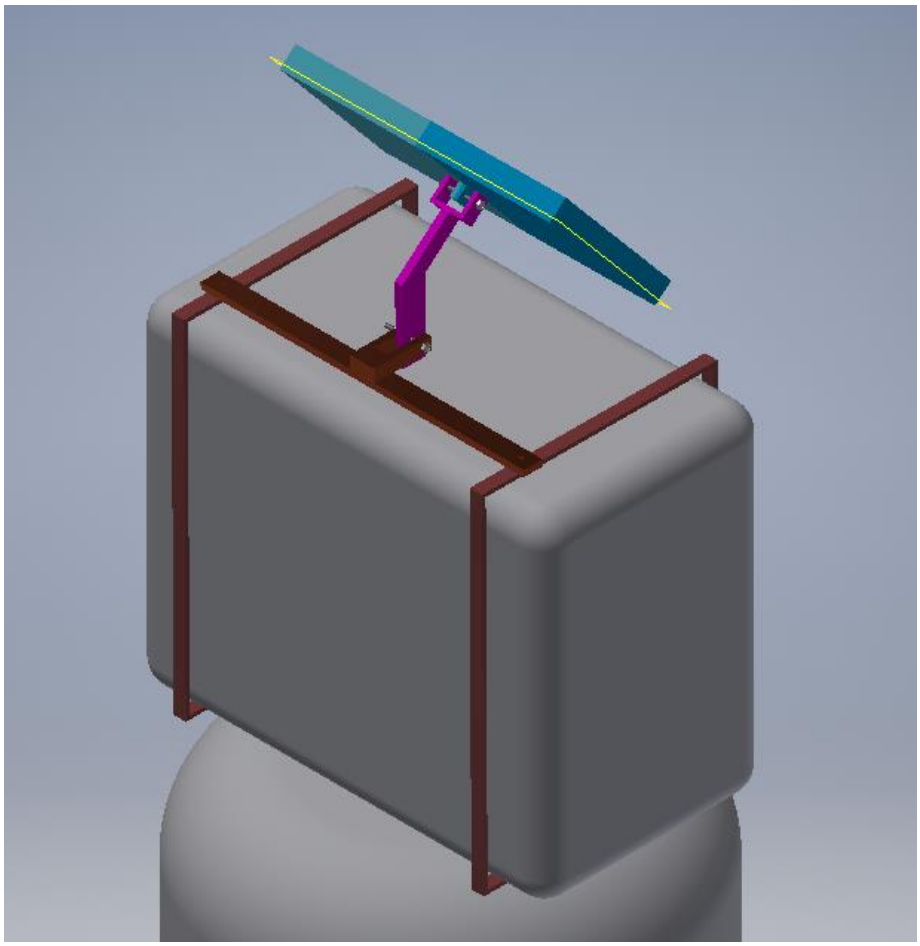

Ilustración 44 Primer diseño soporte trasero

Se ha diseñado un soporte mejorado que es el que se muestra en la ilustración 45. En este diseño la tableta está posición vertical, lo que permite que la cámara se encuentre a mayor altura, facilitando la comunicación con el usuario. Además, este soporte cubre completamente la cabeza del James para que la fijación sea máxima.

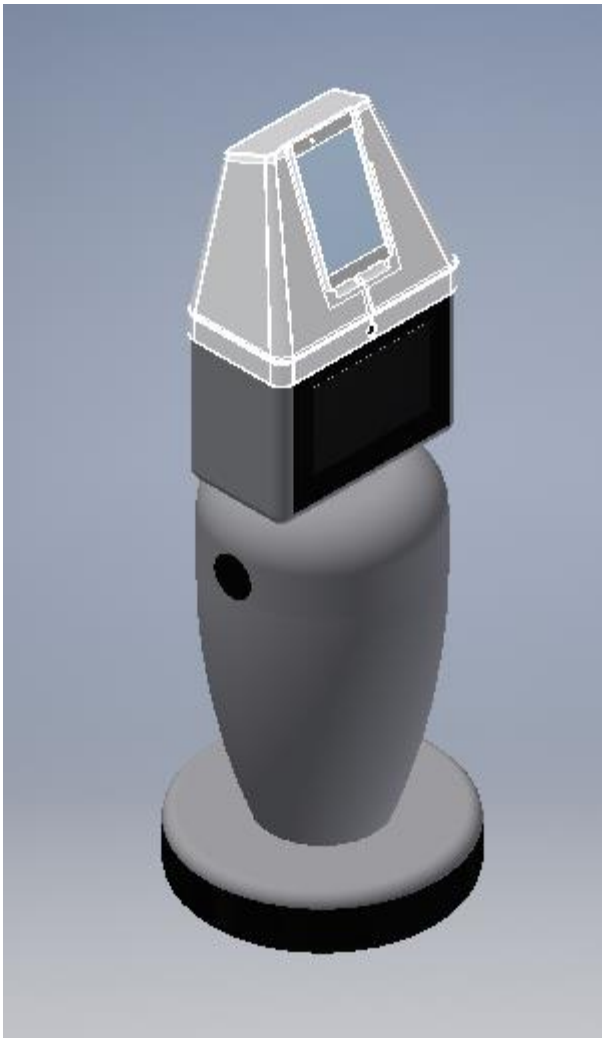

Ilustración 45 Soporte James definitivo

8. Resultados

Después de analizar los dos métodos posibles para crear una aplicación que controle el robot social James se ha optado por crear una página web HTML. Esto nos permitirá abrir esta aplicación desde cualquier dispositivo porque cualquier navegador puede abrir este tipo de código.

En la aplicación de Android se ha conseguido implementar las funciones MQTT del robot James aunque para nuestro objetivo no es la mejor solución, pero puede ser una solución viable para otro tipo de aplicaciones. Por ejemplo, para situaciones donde solo es posible controlar a James desde un móvil o se necesita una aplicación con visualización más sencilla entonces la posibilidad de este tipo de programas será una buena opción.

Se ha decido crear una aplicación con un diseño compacto de página web donde todas las funciones se puedan ver en pantalla a la vez como se puede ver en la *ilustración 46*.

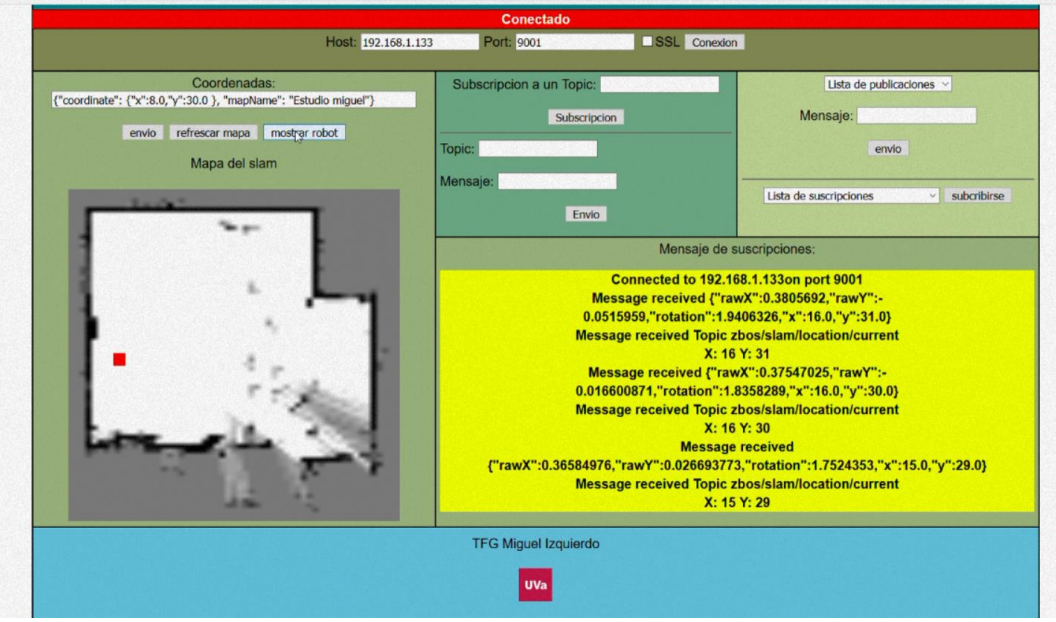

Ilustración 46 Diseño página web

Se ha dividido la aplicación en varias zonas:

- La primera a la izquierda que tiene el mapa donde se va a mostrar la localización del robot en la imagen. Además, se puede mover el robot por este poniendo las coordenadas.
- En el parte central se escribe cualquier topic que quiera el usuario tanto para suscribirse como para publicar.
- A la derecha tenemos una serie de funciones predeterminadas para que el usuario no tenga que saberse de memoria el nombre

completo del topic que quiere publicar o al que quiere suscribirse.

- En la parte derecha abajo (en amarillo) están los mensajes que nos llegan de James dependiendo de los topics a los que estemos suscritos. Estos mensajes se irán acumulando y con una barra podemos ver el historial como se ve en la *ilustración 47.*

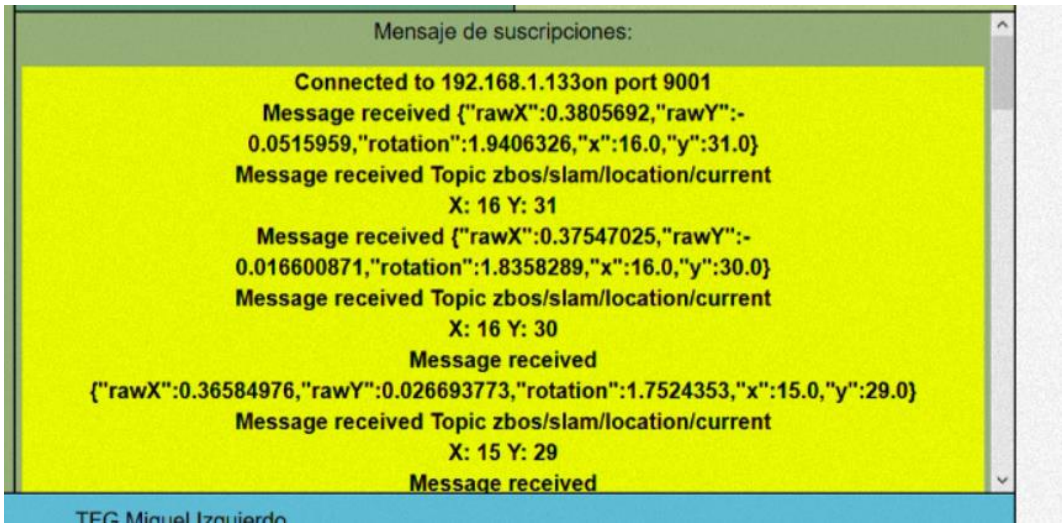

Ilustración 47 Mensajes página web James

Una vez tenemos la aplicación web final se puede adaptar de forma sencilla a un entorno donde el robot puede ser más útil como puede ser una residencia de personas de avanzada edad o una planta de hospital donde haya riesgo de contagio con los pacientes. El mapa del James que se ve en la *ilustración 48* es una simulación de cómo sería la situación de una planta de hospital. En este mapa se marcan los puntos importantes: el punto azul es el punto de carga del robot que estaría enfrente de la recepción de la planta y los puntos rojos son los posibles puntos de parada del robot.

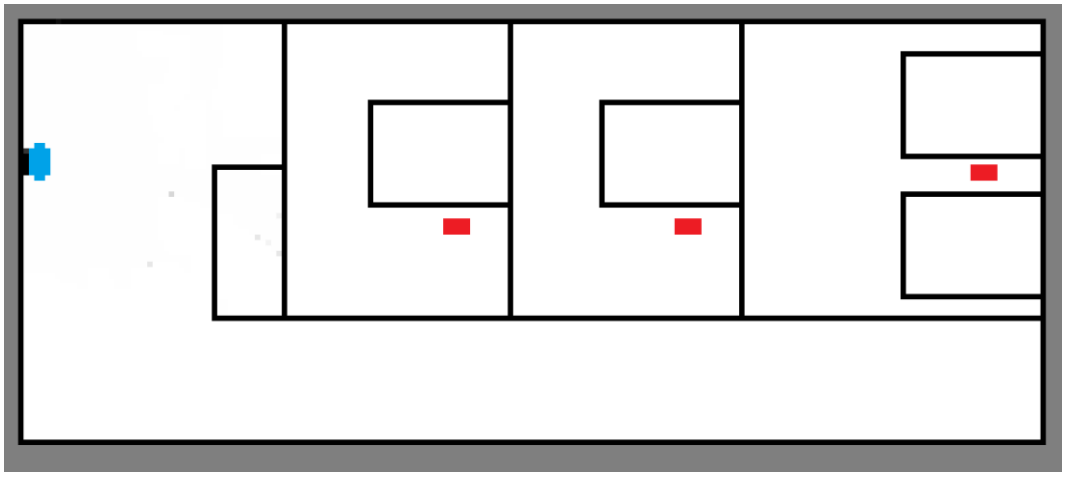

Ilustración 48 Simulación mapa hospital

Cuando el robot llegue a uno de estos puntos rojos el robot James puede activar la videollamada para que el sanitario que necesite contactar con el paciente puede realizar las preguntas oportunas o simplemente ver el estado de los monitores y del enfermo. Otra posibilidad es hacer que el paciente rellene un pequeño cuestionario para ver y registrar la evolución de su estado o su historial sin necesidad de contacto físico.

La primera opción se realiza con la tableta instalada en la parte superior de robot. En la tableta se creará una app que reaccione a comandos MQTT con la misma página web que se ha desarrollado en este Trabajo de fin de grado de tal manera que al mandar un comando que sea “Videollamada” se active una aplicación que permita la comunicación como puede ser Skype u otra aplicación del mismo tipo.

La segunda opción es hacer un formulario que el robot James ya tiene por defecto instalada, esta aplicación nos permite que el usuario que interactúa con James responda a algunas preguntas de manera cómoda. Algunas preguntas pueden ser:

1. Nombre y apellidos
2. Edad
3. Antecedentes personales: enfermedades e intervenciones quirúrgicas
4. Alergias medicamentosas
5. Tratamiento habitual: dosis y hora
6. Grupo sanguíneo
7. Persona y teléfono de contacto en caso de urgencia
8. Tipo de dieta (fácil masticación, liquida, rica en fibra, sin gluten…)

9. Episodios de ingreso en el último periodo: si/no

10. Cambios en el tratamiento: si/no

Con estas preguntas podríamos hacer una historia clínica básica del paciente sin necesidad de establecer un contacto físico con el paciente (*ver ilustración 49*). Se podrían cambiar las preguntas dependiendo de la planta del hospital o si es una residencia de personas de avanzada edad.

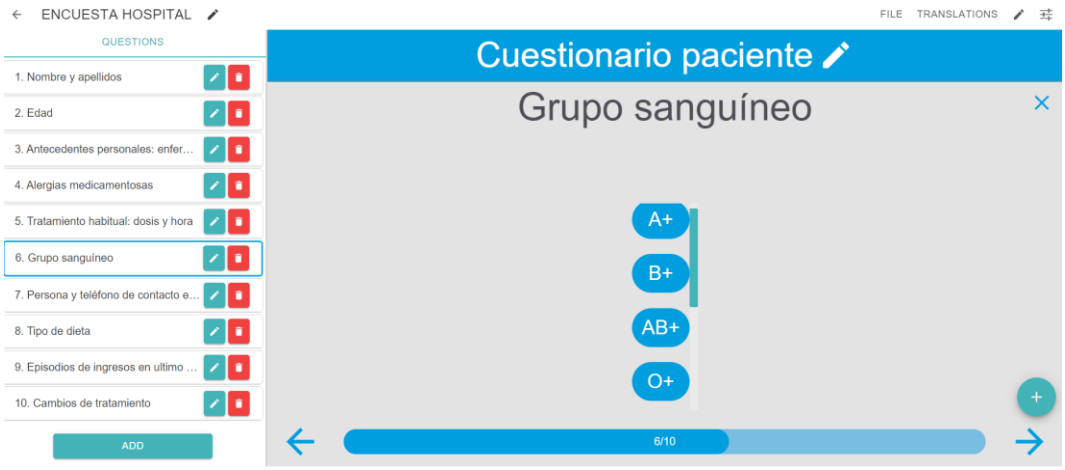

Ilustración 49 Cuestionario paciente

Una vez teniendo todas las aplicaciones se puede desarrollar una composición que no es más que una cadena de acciones programadas para que el robot desarrolle sus funciones. Por ejemplo, una composición puede ser:

- Anunciar que va a comenzar el recorrido:
- Ir al punto 1 que puede ser antes de entrar en la habitación del paciente.
- Ir al punto 2 que estará cerca de la cama a una distancia que permita la interacción con el robot.
- Iniciar la aplicación de la encuesta.
- Iniciar el comando MQTT que comience la videollamada en la tableta superior.
- Anunciar el fin de la reunión cuando reciba el comando MQTT de en topic Fin_videollamada.
- Una vez terminada la videollamada se dirige a la posición de carga.

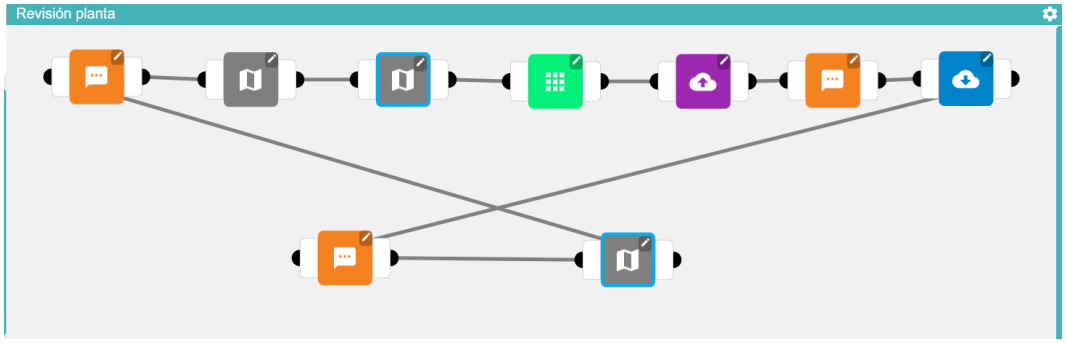

Ilustración 50 Composición

Este tipo de composiciones como la que se puede ver en la *ilustración 50* nos permite crear funciones personalizadas. Este ejemplo sirve para tomar los datos de un paciente, pero se puede aplicar a hacer reuniones online con familiares u otro tipo de actividades.

Estas son las aplicaciones desarrolladas en el presente trabajo que nos permitirá implantar el robot James en un entorno asistencial como un hospital o una residencia. Con las funciones y herramientas que se han planteado en este Trabajo de Fin de Grado se podría crear una implementación de manera rápida y adaptarla a las necesidades específicas.

9. Estudio Económico

9.1.Introducción

En este apartado se realizará un estudio del coste económico que supone la realización del proyecto. Es importante analizar la viabilidad económica del mismo ya que en todo proyecto hay que asegurarse que no es solo viable técnicamente sino también que se puede sostener económicamente.

En los siguientes apartados se estudiarán por separado costes directos e indirectos y en el último apartado se mostrarán los costes totales del proyecto.

9.2.Recursos empleados

A continuación, se muestra un resumen de los recursos empleados en el desarrollo del proyecto. Es importante tener en cuenta únicamente la amortización del material durante el período de tiempo que ha sido utilizado en el proyecto, para calcular el coste real.

- Software:
    - Sistemas operativos: Windows 10 HOME y Android.
- Hardware:
    - Ordenador portátil: Lenovo Z50-70.
    - Tablet – Samsung TAB A 8”
    - Robot James
- Material ofimático:
    - Libros de Consulta.
    - Material de papelería.
    - Otros consumibles.

9.3.Costes directos

En cuanto a los costes directos, se evaluarán:

- Coste del personal.

- Costes amortizables de programas y equipos.
- Costes de materiales directos empleados.

9.3.1. Coste de personal

La realización del presente proyecto ha sido llevada a cabo por un ingeniero encargado del diseño y puesta a punto de la interfaz y el estudio del robot.

Se calcula el coste anual para un ingeniero para posteriormente adecuarlo al número de horas trabajadas por en este proyecto. Este coste anual incluye:

- Sueldo bruto anual, así como los posibles incentivos por su trabajo.

- Cotización a la Seguridad Social, que es un 35% del sueldo bruto.

Teniendo en cuenta esto, el coste anual del ingeniero será:

COSTE ANUAL	
Sueldo bruto más incentivos	27.000 €
Seguridad social (35% sueldo bruto)	9.450 €
Coste total	36.450 €

Tabla 4 Coste anual del personal

A continuación, se hecho una estimación de los días efectivos trabajados al año:

DIAS EFECTIVOS POR AÑO	
Año medio	365.25 días
Fin de semana	-104.36 días
Días de vacaciones efectivos	-20 días
Días festivos reconocidos	-15 días

Días perdidos estimados	-5 días
Total días efectivos estimados	220,89 días

Tabla 5 Días efectivos por año

Una vez conocido el número total de días efectivos de trabajo, y sabiendo que la jornada laboral es de 8 horas, obtenemos el total de horas efectivas de trabajo:

220,89 días/año x 8 horas/día = 1.767,12 horas/año

El coste por hora de un ingeniero se calcula como la división del sueldo anual entre las horas efectivas trabajadas al año:

Coste/hora= 36450 €/1767.12 horas=20.63 €/hora

En la siguiente tabla (*Tabla 6*) se muestra una distribución temporal aproximada del trabajo realizado en el proyecto .

DISTRIBUCION TEMPORAL DE TRABJO	
Formación y documentación	50 horas
Estudio del problema	100 horas
Desarrollo de la aplicación	150 horas
Elaboración de la documentación	200 horas
Total de horas empleadas	500 horas

Tabla 6 Distribución temporal del trabajo

El coste personal directo es la multiplicación de las horas y el coste efectivo de una hora de trabajo del ingeniero:

500horas x 20,63 €/hora = 10.315,00 €

9.3.2.Coste de amortización de programas y equipo

Para el cálculo de los costes se debe realizar primero una inversión total y calcular la amortización lineal correspondiente según los criterios que aconseja la ley. En este apartado se estudiarán tanto los costes de la amortización del material de oficina como los costes de amortización de los equipos.

Se ha estimado como tiempo de amortización 5 años, ya que es el que se puede considerar como vida útil del material informático. De forma que al calcular el coste hay que multiplicar por un factor de (por ejemplo,0.2 para 1 año) los precios mostrados.

MATERIAL	IMPORTE	AMORTIZACION 20%
Sistema operativo Windows 10	251,10€	50,22€
Tablet – Samsung TAB A 8”	175,00€	35,00€
Robot social James	4900,00€	490,00€
Total material de informática	5326,10€	575,22€

Tabla 7 Costes de amortización

El coste por hora de la utilización del material informático se calculará mediante la división de la amortización anual entre el número de horas de uso en dichos equipos.

Coste/hora= 575.22 €/1767.12 horas=0.33€/hora

Se considera el tiempo de uso como el tiempo total necesario en todas las etapas. Por tanto, el coste de amortización de material será:

500horas x 0.33 €/hora = 165€

9.3.3 Costes directos totales

También se debe tener en cuenta el coste del alquiler del robot James. Ya que desde el comienzo del proyecto se inició con la adquisición del robot. Si a esto le añadimos los costes calculados en los anteriores apartados el coste directo total será:

Total: 10.315€ + 165 € = 10.480 €

9.4.Costes indirectos

Los costes indirectos son los gastos producidos por el desarrollo de proyecto, pero no se atribuyen directamente al uso del proyecto.

COSTES INDIRECTOS	
Dirección y servicios administrativos	150€
Consumo de electricidad	180€
Consumo de telefonía e internet	50€
Total gastos indirectos	380€

Tabla 8 Costes Indirectos

9.5.Costes totales

Los costes totales son la suma de los apartados anteriores, es decir, de los gastos directos e indirectos.

COSTES TOTAL	
Costes directos	10.480 €
Costes indirectos	380€
Costes totales	10.860 €

Tabla 9 Costes totales

En conclusión, el coste total del proyecto asciende a la siguiente cantidad:

COSTE TOTAL DEL PROYECTO: 10.860€

10. Conclusiones y Líneas futuras

En este proyecto se ha conseguido desarrollar en profundidad todas las capacidades del robot James. Este robot es una novedad en el mercado por lo que ha sido necesario conocer las capacidades y limitaciones para luego poder desarrollar aplicaciones. Es un robot que tiene aspectos positivos como su aplicación ya que es muy fácil interactuar y está disponible para el móvil u ordenador con la que podemos controlar al robot social James y que proporciona la propia compañía Zorabots. Esto hace que cualquier usuario con un poco de formación pueda manejar este robot de manera rápida y sencilla. Sin embargo, sus funcionalidades tienen algunos problemas como no poder realizar videollamadas porque el micrófono solo puede recibir comandos de voz y no transmitir la voz del usuario. Además, tiene un tamaño muy pequeño lo que hace que interactuar sea bastante complicado desde una posición erguida y su aplicación no es versátil por lo que hemos creado nuestra propia aplicación.

Con respecto a las aplicaciones desarrolladas en este Trabajo Fin de Grado es un primer paso en el desarrollo del robot James. Con estos fundamentos que se han elaborado será más fácil construir otros proyectos en el futuro. Se ha cumplido el objetivo principal que es desarrollar una herramienta que haga que el robot social James sea útil en un entorno de apoyo a personas de avanzada edad o con algún problema médico. Esta aplicación tiene multitud de posibilidades ya que podemos utilizar todas las funciones del robot aplicadas en la actividad que nos interese. Cambiando solo las funciones predeterminadas de la página web se puede hacer que active una aplicación u otra.

Se han resuelto los problemas que presenta el robot como es la posibilidad de hacer videollamadas añadiendo la tableta con el soporte superior. También se ha conseguido hacer más personalizable las funciones del robot creando las aplicaciones que hemos visto en los capítulos anteriores.

Una de las aplicaciones a la que se puede dedicar el robot social James es como una forma de comunicación de James con personas mayores u personas con algún problema médico sin necesidad de contacto humano. Así el robot se puede desplazar por una planta de hospital parando en cada cama y con la tableta superior con la función de videollamada el interlocutor puede establecer la comunicación. Esto permitiría a un doctor pasar revisión a una planta sin estar directamente en contacto con los pacientes y esto puede ser muy interesante en algunas situaciones. También se puede aplicar al contacto de los familiares con las personas de avanzada edad u hospitalizadas sin tener que estar presentes en la habitación.

Algunas de las posibles futuras mejoras que se pueden hacer al robot social James es incluir algún tipo de funcionalidad física. Ahora mismo el robot

tiene como única capacidad física la movilidad, sería interesante añadir un compartimento para repartir medicina u otros objetos. El cuerpo del robot solo tiene una la función de soporte y almacenaje del hardware, así que se podría añadir un pequeño brazo mecánico o un compartimento para que sea más útil y ampliar sus funciones.

Otra posible línea de trabajo es mejorar la comunicación entre la tableta que se ha incluido en la parte superior del robot con el funcionamiento del robot para que funcionen como un solo ente. Así poder manejar las dos pantallas del robot a voluntad del usuario. Esto se podría hacer con la comunicación MQTT o incluso en una conexión física, aunque esto supondría intervenir en el funcionamiento del robot James.

Bibliografía

Amit Kumar Pandey, & Rodolphe Gelin. (2018). A Mass-Produced Sociable Humanoid Robot: Pepper: The First Machine of. *IEEE Robotics & Automation Magazine* ·.

Castellano, G., Leite, I., Pereira, A., Paiva, A., & McOwan, P. (2009). Detecting user engagement with a robot companion using task and social interaction-based features. *international conference on Multimodal interfaces*, (págs. 119-126). Cambridge Massachusetts USA.

E. Pollack, M., Brown, L., Colbry, D., Orosz, C., Peitner, B., RamKrishnan, . . . Roy, N. (2002). Pearl: A mobile robotic assistant for the elderly. En K. Haigh, *Proceedings of Workshop on Automation as Caregiver: the Role of Intelligent Technology in Elder Care.* AAAI.

Gladden, M. (2018). Sapient Circuits and Digitalized Flesh: The Organization as Locus of Technological Posthumanization. En M. Gladden, *Sapient Circuits and Digitalized Flesh: The Organization as Locus of Technological Posthumanization* (pág. 19). Indianapolis: Defragmenter Media.

ISO. (06 de Mayo de 2020). *ISO 8373:2012(en) Robots and robotic devices — Vocabulary*. Obtenido de https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en

Kanamori, M., Suzuki, M., Oshiro, H., Tanaka, M., Inoguchi, T., Takasugi, H., . . . Yokoyama, T. (2003). Pilot study on improvement of quality of life among elderly using a pet-type robot. *Computational Intelligence in Robotics and Automation* (pág. Volumen 1). Kobe, Japan, Japan: IEEE.

Kidd, C., Taggart, W., & Turkle, S. (2006). A sociable robot to encourage social interaction among the elderly. *Proceedings - IEEE International Conference on Robotics and Automation* ·. Orlando, Florida, USA.

Kock, S. (2 de diciembre de 2015). *Paho Android Service - MQTT Client Library Encyclopedia*. Obtenido de https://www.hivemq.com/blog/mqtt-client-library-enyclopedia-paho-android-service/

Martin, S. (29 de abril de 2020). *Using javascript mqtt client websockets*. Obtenido de http://www.steves-internet-guide.com/using-javascript-mqtt-client-websockets/

MQTT.org. (29 de abril de 2020). *MQTT*. Obtenido de http://mqtt.org/faq

Rice, S. P. (2004). Minding the Machine: Languages of Class in Early Industrial America . En S. P. Rice, *Minding the Machine: Languages of Class in Early Industrial America* (pág. 12). Berkeley: University of California Press.

Sherry Turkle, Will Taggart, Cory D. Kidd, & Olivia Dasté. (2006). Relational Artifacts with Children and Elders: The Complexities of Cybercompanionship. *Connection Science* (págs. 347-361). Cambridge: Taylor & Francis.

Shiarlis, K., M. J., v. S., Whiteson,Shimon, K. J., V. J., . . . m. H. (2015). TERESA: A Socially Intelligent Semi-autonomous Telepresence System. *Conference: International Conference on Robotics and Automation.* Seattle.

The new stack. (29 de abril de 2020). *Get to Know MQTT: The Messaging Protocol for the Internet of Things*. Obtenido de https://thenewstack.io/mqtt-protocol-iot/

Turkle, S. (1996). Who Am We? : We are moving from modernist calculation toward postmodernist simulation, where the self is a multiple, distributed system,. *Wire Magazine*.

Zorabots. (29 de abril de 2020). *ZBOS MQTT API 1.2.8*. Obtenido de https://docs.zoracloud.com/mqtt-api/

Anexos

Código aplicación HTML

James_web.html

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE html PUBLIC "-
//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="es">
<head>
    <title>HTML James Tfg Miguel</title>
    <meta charset="UTF-8">
    <meta name="description" content="Esta es mi primera pagina mqtt
 del robot james"/>
    <meta name="author" content="Miguel Izquierdo"/>
    <meta name="Keywords" content="Robot James,mqtt"/>
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" type="text/css" href="mystyle.css">



    <script src="https://cdnjs.cloudflare.com/ajax/libs/paho-
mqtt/1.0.1/mqttws31.js" type="text/javascript"></script>
    <script type = "text/javascript"
         src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/j
query.min.js"></script>
    <script type = "text/javascript" src="myScript.js"></script>

</head>
<body>
    <h1 class="titulo">Pagina html con conexion MQtt para James</h1>

        <script type = "text/javascript">
        //ll</script>
    <script>
        var connected_flag=0
        var mqtt;
        var reconnectTimeout = 2000;
        //var host="192.168.1.133";
        //var port=9001;
        //var host=document.forms["connform"]["Serv"].value;
        //var port=document.forms["connform"]["Port"].value;
        var host="000.000.0.000";
        var port="0000";
    </script>
```

```
<div id="status">Estado de la conexion: No Conectado</div>

<div id="inicio">
    <form name="connform" action="" onsubmit="return MQTTconnect()">
        Host:   <input type="text" name="Serv">
        Port:   <input type="text" name="Port">
        <input type="checkbox" id="SSL_act">SSL
        <input type="submit" value="Conexion">
    </form>


    <br>
</div>
<div id="esquema">
    <div id="topicogenerico">
        <form name="subs" action="" onsubmit="return sub_generico()"
>

            Subscripcion a un Topic:   <input type="text" name="Stop
ic"><br><br>
            <input type="submit" value="Subscripcion">
        </form>
        <hr>


        <form name="smessage" action="" onsubmit="return Topic_gener
ico()">

            <div id="publicar">
            Topic:   <input type="text" name="Ptopic"><br><br>
            Mensaje: <input type="text" name="message"><br><br>

            </div>
            <input type="submit" value="Envio">
        </form>

    </div>
    <div id="topicopre">
            <select id="mySelect2">
                <option value=-1>Lista de publicaciones</option>
                <option value="zbos/dialog/set/message">Hablar con l
lames</option>
                <option value="zbos/audio/beep">Emitir un pitido</op
tion>
                <option value="zbos/motion/control/movement">Mover e
l robot</option>
```

```
                <option value="zbos/audio/player/start">Poner un aud
io</option>
            </select>
            <br><br>
        Mensaje: <input id="msg1" type="text" name="message"><br><br
>
        <button onclick="envios()">envio</button>
        <script>
            function envios() {
                var x = document.getElementById("mySelect2").value;
                var y = document.getElementById("msg1").value;
                pub_topic(x,y);
            }
        </script>
        <br><br>
        <hr>
        <select id="mySelect" >
            <option value=-1>Lista de suscripciones</option>
            <option value="zbos/slam/location/current">Aviso cambio
de posicion</option>
            <option value="zbos/slam/location/reset/chargingstation/
started">Aviso de vuelta a poscion inicial</option>
            <option value="zbos/slam/charging/required/started">Avis
o de bateria baja</option>
            <option value="zbos/audio/volume/event">Aviso cambio de
volumen</option>
        </select>
        <button onclick="subcripciones()">subcribirse</button>
            <script>
                function subcripciones() {
                    var x = document.getElementById("mySelect").valu
e;
                    document.getElementById("NombreTopic").innerHTML
 =x;
                    sub_topics(x);

                }
            </script>
        </script>
        <br>



    </div>
    <div id="zonamapa">

        Coordenadas:<br>
```

```
        <input id="coor" type="text" name="message"  size="70">
        <script>
            var x= document.getElementById("coor");
            x.setAttribute("value", "{\"coordinate\": {\"x\":8.0,\"y
\":30.0 }, \"mapName\": \"Estudio miguel\"}");
        </script>
        <br><br>
        <script> var c = document.getElementById("coor").value;</scr
ipt>
        <button onclick="coordenadas(c)">envio</button>
        <button onclick="refrescarmapa()">refrescar mapa</button>
        <button onclick="robotxy()">mostrar robot</button>
        <p id="Titulomapa">
            Mapa del slam
        </p>

        <div id="mapa">
            <img id="mimapa" src="data:mapa/png;base64,iVBORw0KGgoAA
AANSUhEUgAAADkAAABJCAYAAACHOo5EAAAABHNCSVQICAgIfAhkiAAABnNJREFUeJzdW
z+I1Eoc/l6wCk59GM7K4HAWQrhgZ3Cxs1hshIW0Hi52ym4nLCvXnQh2J2ernNjluGsj1
wkD6VwGUg9sp4wEbHyvOGZesptk9zabZM+vuWwyycw3v/8zc/+MRqN/8ZfDaHsATeBSJ
Dnnpffynm8CliZJCNHXiowQApRSjMdjEEJAKV3/CNeAa+kf4/G4sCEhBFLKwueDwWB9o
ypAr9dbaSIzJAkh+Pnz59oGtU4YhrGypmTU1bKstQwoD2VaUDcyJOscyMHBAcIwrO37Z
cioa50kX79+jZ2dHdy9exefP3/G06dP0el0QAhBt9utrV8gxybrxGQy0dfdbrd2cgpaX
YMggBCikU4BNEYQSJH0fb92SVZFEAQrvadJMsZq9a7rwKrS1yRd1914Sa6aNmqSUsqNJ
7mWZKDNgF0nMiQ3XZKrQpPc1DIpjco2CWy+JCvbJKX077dJznnjkmzKRFpd42GMNdJPq
yQJIZciukxal6cd/6SXJMfjMf78+bN0p03CMOb14XkeKKXgnINzjm63i6OjI4xGo0y7a
3NvbiAMw9CTH0WRvk8IgWVZME0ThmEUOs4rQTINx3Fy708mE1BKcXx8PPfsSiQDy5hQW
Qw1lml01ZGxZkop7t2718pA6tSkDEnOeWOxaxarVv3LYGM2fPr9fm3S3BiShJDKaWXRJ
G0MSaD6Cn6R85wjuakZTxVslCTrwl9LMm2frZKsy5vObgi3uvxRV2ycTdRbW5JUpVEav
3//Xtu302hNXVUdmAZjDF++fFl7X61WIXl7G0+ePFn5e0oTZ+PlRlUhW1tbhc+WMaUin
zJXhbQJ27b1dZIkSJIE0+kUAHKL4WXR2MGIPARBgMPDw7n7x8fHME0TwIUEz8/PM2NLk
uRS/WSWP9oos/r9vr7e39+Hbdvo9XqI41jfF0JACAHLskAI0RKfTqelKq7Q6q7WrHn0+
31N0LZtSCnx4sULSCnR6XQ02TiOkSSJJrhIsq0mA7Mkv3//DuDCNpMkgeM4OD8/n5OkK
u6jKMJ0Ol0onI3a8PE8DwAQxzFM08Tu7i5OTk70Pg1jTJO1LEuTJ4QgjmPtpGaRIdnk6
Y8y2LaNBw8e4OzsTK+tAv+f72OMgXMOSulSq/CNZjxFM52HDx8+6ARFkZFS6m1/KSXCM
MyocZHaNkqSEJJZAS+DbdtafcMwBKVUq6iUUqupEEJLtgit5K6K6DIhy3Ec9Ho9SCkzt
ghAk1XXjDEdX9OYI7mzs1OJQBlM04SUEpRSHB0d4fHjx0u/GwQBCCHwfV+fyUsTVPure
eFkLq2bTCb65TAM8fz584xDqlIDKps0TRNCCHielwn6Rbhx4waiKIJt22CM6dCTtlXOO
YIgQK/XKyeZTqGEEEiSBIeHh9je3oZhGDAMA+/evdPX+/v7YMTQui2Dx8+zDwjhKDT6
YBzvtBOz87OsLe3B+AizAghsLe3px2NWgkoyr0zaZ0QAjdv3sTt27dx584dHZzfv3+vj
fvbt2+wbRuWZeHVq1eFA5NSZrYc3rx5o4O453k6qFuWBc457t+/j1+/foExBtd1M99yH
EcnBmpPMo5jnQUp21x66+7WrVsALgKymi3l3aSUGA6HODg4wHQ6zd0YzXPlX79+1d/hn
OP69es6ownDEFEUwTRNvH37Fi9fviycOM/zEEURHMfR+StjTIcVQgiCIMBwOMy8l9lpD
sMQlmXpqj39N420Xfq+r9WbMZYpiUzTRLfbBaUUQgj4vl96elkNVKnmLOI4hmVZetIJI
dja2sLp6WnGj8yON0Myj9AiKLWjlGacwmybwWCAk5OThaqlAnueAwGA09NTEELw7Nkz/
PjxA51OB58+fcLu7q62TeXBc0mugnS+W5YopztNx7e876lUznEcbG9vF35T7ToPh0O4r
```

```
ouPHz/mCqoxkkpCCkUZiiKZjoODwQCj0ahU0yilcF0Xjx49mjvXXjnjUYa/roUwlZ8CF
xMjpdROpsyUlOeO43hugawySdWxCgVlg09fL1pqUe055/B9f+kzPHkHJ9aau5bN9GUPD
SuJqolb9mh2XrtGE/R0qVRGOJ3JrOO/DhqvQi4jUVUrVkUrpZaS6KKViHWdi29FksD89
lpeO875Wjx3o8fO0nWfSiHL1FFlL67rVloubVSSKm9Varho2UK1rSrJyhlPU1glr1a4M
mcGqmxGXRmSVfAfZO5+BPUFXk4AAAAASUVORK5CYII003d" alt="Mapa Estudio Mi
guel" width="400" height="400" >


        </div>

    </div>
    <div id="zonasuscripciones">
        Mensaje de suscripciones:
        <p id="NombreTopic"></p>
        <p id="messages"></p>
    </div>

    <br><br>
</div>
    <div id="punto">
        <img alt="puntorojo" style="position:absolute; left: 200px;
top: 80px;" width="15" height="15" src="
0KGgoAAAANSUhEUgAAAAEAAAABCAIAAACQd1PeAAAADElEQVR42mP4z8AAAAMBAQD3A0
FDAAAAAElFTkSuQmCC" >
    </div>
</body>
  <footer>
      TFG Miguel Izquierdo
      <div id=logo>
      <img id=imglogo src="https://upload.wikimedia.org/wikipedia/co
mmons/thumb/1/1d/Logotipo_de_la_Universidad_de_Valladolid.svg/584px-
Logotipo_de_la_Universidad_de_Valladolid.svg.png"
      width="40" height="40" />
      </div>
   </footer>
</html>
```

myScript.js

```
function onConnectionLost(){
    console.log("connection lost");
    document.getElementById("status").innerHTML = "Connection Lost";
    document.getElementById("messages").innerHTML ="Connection Lost"
;
    connected_flag=0;
}
function onFailure(message) {
        console.log("Failed");
        document.getElementById("messages").innerHTML = "Connection
Failed- Retrying";
        setTimeout(MQTTconnect, reconnectTimeout);
        }
function onMessageArrived(r_message){
        out_msg="Message received "+r_message.payloadString+"<br>";
        out_msg=out_msg+"Message received Topic "+r_message.destinat
ionName;
        console.log(out_msg);
        document.getElementById("messages").innerHTML =document.getE
lementById("messages").innerHTML+"<br>"+out_msg;
        if(r_message.destinationName=="zbos/slam/mapview/current/res
ponse/all")
        {
            obtencionmapa(r_message.payloadString)
        }
        if(r_message.destinationName=="zbos/slam/location/current")
        {
            obtencioncoor(r_message.payloadString)
        }
    }
    function onConnected(recon,url){
    console.log(" in onConnected " +reconn);
    }
    function onConnect() {
    document.getElementById("messages").innerHTML ="Connected to "+h
ost +"on port "+port;
    connected_flag=1
    document.getElementById("status").innerHTML = "Conectado";
    console.log("on Connect "+connected_flag);
      }

    function MQTTconnect() {
    document.getElementById("messages").innerHTML ="";
    var s = document.forms["connform"]["Serv"].value;
    var p = document.forms["connform"]["Port"].value;
```

```
var SSL= document.getElementById("SSL_act").checked;
//var s="192.168.1.133";
//var p=9001;
if (p!="")
{
console.log("ports");
    port=parseInt(p);
    console.log("port" +port);
    }
if (s!="")
{
    host=s;
    console.log("host");
    }


console.log("connecting to "+ host +" "+ port);
mqtt = new Paho.MQTT.Client(host,port,"clientjsaaa");
var options = {
    timeout: 3,
    onSuccess: onConnect,
    onFailure: onFailure,
    useSSL:SSL,

 };

mqtt.onConnectionLost = onConnectionLost;
mqtt.onMessageArrived = onMessageArrived;
mqtt.onConnected = onConnected;
mqtt.connect(options);
return false;


}


function sub_generico(){
    document.getElementById("messages").innerHTML ="";
    if (connected_flag==0){
        out_msg="<b>Not Connected so can't subscribe</b>"
        console.log(out_msg);
        document.getElementById("messages").innerHTML = out_msg;
        return false;
    }
    var stopic= document.forms["subs"]["Stopic"].value;
    console.log("Subscribing to topic1 ="+stopic);
    mqtt.subscribe(stopic);
```

```
    return false;
}



function Topic_generico(){
    document.getElementById("messages").innerHTML ="";
    if (connected_flag==0){
    out_msg="<b>Not Connected so can't send</b>"
    console.log(out_msg);
    document.getElementById("messages").innerHTML = out_msg;
    return false;
    }
    var msg = document.forms["smessage"]["message"].value;
    console.log(msg);

    var topic = document.forms["smessage"]["Ptopic"].value;

    message = new Paho.MQTT.Message(msg);
    if (topic=="")
        message.destinationName = "test-topic"
    else
        message.destinationName = topic;
    mqtt.send(message);
    return false;
}
function pub_topic(topics,mensajetopic){
    document.getElementById("messages").innerHTML ="";
    if (connected_flag==0){
    out_msg="<b>Not Connected so can't send</b>"
    console.log(out_msg);
    document.getElementById("messages").innerHTML = out_msg;
    return false;
    }
    var msg = mensajetopic;
    console.log(msg);

    var topic =topics;
    message = new Paho.MQTT.Message(msg);
    if (topic=="")
        message.destinationName = "test-topic"
    else
        message.destinationName = topic;
    mqtt.send(message);
    return false;
}
```

```
function sub_topics(topico){
document.getElementById("messages").innerHTML ="";
    if (connected_flag==0){
    out_msg="<b>Not Connected so can't subscribe</b>"

    console.log(out_msg);
    document.getElementById("messages").innerHTML =out_msg;
    return false;
    }
var stopic= topico;
console.log("Subscribing to topic ="+stopic);
mqtt.subscribe(stopic);
return false;
}
function coordenadas(mensaje){
    document.getElementById("messages").innerHTML ="";
    if (connected_flag==0){
    out_msg="<b>Not Connected so can't send</b>"
    console.log(out_msg);
    document.getElementById("messages").innerHTML = out_msg;
    return false;
    }
    var msg = mensaje;
    console.log(msg);

    var topic ="zbos/slam/interaction/moveto";
    message = new Paho.MQTT.Message(msg);
    if (topic=="")
        message.destinationName = "test-topic"
    else
        message.destinationName = topic;
    mqtt.send(message);
    return false;
}
function refrescarmapa(){
    suscripcionmapa();
    document.getElementById("messages").innerHTML ="";
    if (connected_flag==0){
        out_msg="<b>Not Connected so can't send</b>"
        console.log(out_msg);
        document.getElementById("messages").innerHTML = out_msg;
        return false;
    }
    var topic ="zbos/slam/mapview/current/get";
    var msg = "{“key”: “ABCxyz”}";
    console.log(msg);
```

```
        message = new Paho.MQTT.Message(msg);
        if (topic=="")
            message.destinationName = "test-topic"
        else
            message.destinationName = topic;
        mqtt.send(message);
        return false;
    }
    function obtencionmapa(datosmapa){

        var res = datosmapa.slice(9,-29);
        res= res.replace(/\\n/g, '');
        res= res.replace(/\\u/g, '');

        imagen=""+res;
        document.getElementById("mimapa").src = imagen;
        document.getElementById("messages").innerHTML ="Mapa actuali
zado";
    }
    function suscripcionmapa(){

            if (connected_flag==0){
            out_msg="<b>Not Connected so can't subscribe</b>"

            console.log(out_msg);
            document.getElementById("messages").innerHTML =out_msg;
            return false;
            }

        var stopic= "zbos/slam/mapview/current/response/all";
        console.log("Subscribing to topic ="+stopic);
        mqtt.subscribe(stopic);
        return false;
        }

    function robotxy(){

        if (connected_flag==0){
                out_msg="<b>Not Connected so can't subscribe</b>"

                console.log(out_msg);
                document.getElementById("messages").innerHTML =out_m
sg;
                return false;
        }
```

```
        var stopic= "zbos/slam/location/current";
        console.log("Subscribing to topic ="+stopic);
        mqtt.subscribe(stopic);

        return false;

    }
    function obtencioncoor(datoscoor){
        datoscoor=datoscoor.slice(26);
        var xi = datoscoor.search(/x/i);
        xi=xi+3;
        var xf=xi+2;
        var yi = datoscoor.search(/y/i);
        yi=yi+3;
        var yf=yi+2;
        x = datoscoor.slice(xi,xf);
        y = datoscoor.slice(yi,yf)
        var coorx = parseInt(x)*5;
        var coory= parseInt(y)*10.3-100;

        document.getElementById("punto").style.position= "absolute";
        document.getElementById("punto").style.top= coory+"px";
        document.getElementById("punto").style.left= coorx+"px";

        document.getElementById("messages").innerHTML =document.getE
lementById("messages").innerHTML+"<br>"+"X: "+x+" Y:  "+y;

    }
    document.addEventListener("click", function(){
        if(event.clientX>220 && event.clientX<470){
            if(event.clientY>320 && event.clientY<590){
                var x= (event.clientX -90)/11.5;
                var y=(event.clientY+100)/10.3;
                var coorx = parseInt(x);
                var coory= parseInt(y);
                var topic ="zbos/slam/interaction/moveto";
                var msg = "{\"coordinate\": {\"x\":"+ coorx + ",\"y\
":"+coory+" }, \"mapName\": \"Estudio miguel\"}";
                document.getElementById("messages").innerHTML =  doc
ument.getElementById("messages").innerHTML + msg+"\n";
                message = new Paho.MQTT.Message(msg);
                message.destinationName = topic;
                mqtt.send(message);


                return false;
            }
```

```
    }
});
```

mystyle.css

```
#messages
{

background-color:yellow;
font-size:3;
font-weight:bold;
line-height:140%;
word-wrap: break-word;
}
#status
{
background-color:red;
font-size:4;
font-weight:bold;
color:white;
line-height:140%;
}


.titulo{
    background-color: teal;
    border: 1px solid black;
    padding: 20px;
    margin-bottom: 0px;
    margin-top: 0px;
}
#esquema{
    grid-template-columns: 40% 30% 30%;
    grid-template-rows: 200px 350px;
    display: grid;
}
#inicio{
    background-color:#888c56;
    border: 1px solid black;
    padding: 5px;

}
#topicogenerico{
    background-color:#70ac84;
    border: 1px solid black;
    padding: 5px;

}
#publicar{
    text-align: left;
```

```
}
#topicopre{
    background-color:#c4d495;
    border: 1px solid black;
    padding: 5px;
}
#zonamapa{
    grid-row: 1 / 3;
    background-color:#9fb680;
    border: 1px solid black;
    padding: 5px;

}
#zonasuscripciones{
    grid-column: 2 / 4;
    background-color:#9fb680;
    border: 1px solid black;
    padding: 5px;
    width: inherit;
    overflow: auto;
}
#mapa{
    display: block;
    margin: 20px auto;
}

#Titulomapa{
    font-size:4;

    line-height:140%;
}

#logo{

    display: block;
    margin: 20px auto;
}
body{
    margin: 0px 10%;
    width: 80%;
    border: 1px solid black;
    text-align: center;
    font-family: sans-serif;
}
footer{
    background-color:#63c5da;
    padding: 10px;
```

```
    border: 1px solid black;
    text-align: center;
    font-family: sans-serif;
    display: block;

}
```

Código App Android

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/md_green_50"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="180dp"
        android:layout_marginLeft="180dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="180dp"
        android:layout_marginRight="180dp"
        android:layout_marginBottom="600dp"
        android:text="@string/nombre"
        android:textColor="@color/md_black_1000"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0" />

    <TableLayout
        android:id="@+id/tableLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="0dp"
        android:layout_marginLeft="0dp"
        android:layout_marginEnd="180dp"
        android:layout_marginRight="180dp"
        android:layout_marginBottom="262dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Button
                android:id="@+id/msg1"
                android:layout_width="wrap_content"
                android:background="@color/md_green_200"
                android:layout_height="wrap_content"
                android:onClick="handleClick"
                android:text="@string/boton1"
                android:textColor="@color/md_black_1000"
                tools:layout_editor_absoluteX="43dp"
                tools:layout_editor_absoluteY="341dp" />
        </TableRow>
```

```
<TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="5dp">

    <Button
        android:id="@+id/act1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/md_green_100"
        android:textColor="@color/md_black_1000"

        android:onClick="handleClick"
        android:text="@string/boton2"
        tools:layout_editor_absoluteX="258dp"
        tools:layout_editor_absoluteY="343dp" />
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="5dp">

    <Button
        android:id="@+id/vac1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/md_green_200"
        android:textColor="@color/md_black_1000"
        android:onClick="handleClick"
        android:text="@string/boton3"
        tools:layout_editor_absoluteX="44dp"
        tools:layout_editor_absoluteY="452dp" />
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="25dp"
    android:layout_marginTop="5dp">

    <Button
        android:id="@+id/vac2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/md_green_100"
        android:textColor="@color/md_black_1000"

        android:onClick="handleClick"
        android:text="@string/boton4"
        tools:layout_editor_absoluteX="255dp"
        tools:layout_editor_absoluteY="449dp" />
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="25dp"
    android:layout_marginTop="5dp">
```

```
        <Button
            android:id="@+id/beep"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@color/md_green_200"
            android:textColor="@color/md_black_1000"

            android:onClick="handleClick"
            android:text="@string/boton5"
            tools:layout_editor_absoluteX="255dp"
            tools:layout_editor_absoluteY="449dp" />
    </TableRow>
</TableLayout>


<androidx.cardview.widget.CardView
    android:id="@+id/card1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/tableLayout"
    android:layout_marginTop="24dp"

    app:cardCornerRadius="20dp"
    android:layout_marginStart="9dp"
    android:layout_marginEnd="9dp"
    android:layout_marginRight="9dp"
    android:layout_marginBottom="39dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tableLayout">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Dialogo con James"
            android:textStyle="bold"
            android:textAlignment="center"
            android:textSize="20dp"/>
        <EditText
            android:id="@+id/editartexto"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginHorizontal="10dp"
            android:hint="Escribe el texto aqui"/>
        <Button
            android:id="@+id/enviartexto"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginHorizontal="10dp"
            android:layout_marginBottom="10dp"
            android:text="Enviar"/>



</LinearLayout>
```

```
    </androidx.cardview.widget.CardView>

    <androidx.cardview.widget.CardView
        android:id="@+id/card2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"

        app:cardCornerRadius="20dp"
        android:layout_marginStart="180dp"
        android:layout_marginEnd="9dp"
        android:layout_marginRight="900dp"
        android:layout_marginBottom="550dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tableLayout"
        android:layout_marginLeft="180dp">


    <LinearLayout
        android:layout_width="223dp"
        android:layout_height="242dp"
        android:orientation="vertical"
        tools:layout_editor_absoluteX="172dp"
        tools:layout_editor_absoluteY="210dp">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Volumen"
            android:textAlignment="center"
            android:textSize="20dp"
            android:textStyle="bold" />
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

        <TextView
            android:id="@+id/textVol"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="____"
            android:textAlignment="center"
            android:textSize="20dp"
            android:textStyle="bold" />


        </RelativeLayout>

    </LinearLayout>

    </androidx.cardview.widget.CardView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

APP

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"
    defaultConfig {
        applicationId "com.pdhn.mqttdemotfg2"
        minSdkVersion 15
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}
repositories {
    maven {
        url "https://repo.eclipse.org/content/repositories/paho-
releases/"
    }
}
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation
'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.2.0'
    implementation
'androidx.localbroadcastmanager:localbroadcastmanager:1.0.0'

implementation('org.eclipse.paho:org.eclipse.paho.android.service:
1.0.2') {
        exclude module: 'support-v4'
    }
    implementation 'com.android.support:cardview-v7:28.0.0'
}
```

MainActivity.java

```
package com.pdhn.mqttdemotfg2;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import android.view.View;

import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.IMqttActionListener;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.IMqttToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;

import java.io.UnsupportedEncodingException;

public class MainActivity extends AppCompatActivity {
    //variables globales
    static String MQTTHOST ="tcp://192.168.1.133:1883";
    static String topicStr="zbos/dialog/set/message";
    static String topicStr2="zbos/audio/beep";
    static String topicStr3="zbos/composition/start/id";
    static String USERNAME ="";
    static String PASSWORD ="";

    //Card 1
    EditText editText;
    String payload;
    Button btnenviar;

    //Card 2
    TextView textVol;


    MqttAndroidClient client;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        String clientId = MqttClient.generateClientId();
        client = new
MqttAndroidClient(this.getApplicationContext(), MQTTHOST,
clientId);
        MqttConnectOptions options = new MqttConnectOptions();

//options.setMqttVersion(MqttConnectOptions.MQTT_VERSION_3_1);

        //Card 1
        editText = (EditText) findViewById(R.id.editartexto);
        btnenviar =(Button) findViewById(R.id.enviartexto);
```

```
        btnenviar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String topic= "zbos/dialog/set/message";
                payload= editText.getText().toString();
                try {
                    client.publish(topic, payload.getBytes(),0,
false);
                }
                catch (Exception e)
                {
                    e.printStackTrace();
                }

            }
        });

        //Fin Card 1

        //Card 2
        textVol=(TextView) findViewById(R.id.textVol);


        //Fin card 2

        try {
            IMqttToken token = client.connect(options);
            token.setActionCallback(new IMqttActionListener() {
                @Override
                public void onSuccess(IMqttToken asyncActionToken)
{
                    // Si mqtt conecto
                    Toast.makeText(MainActivity.this, "Conectado a
Mqtt", Toast.LENGTH_LONG).show();
                    suscripcionTopics();
                }

                @Override
                public void onFailure(IMqttToken asyncActionToken,
Throwable exception) {
                    // Si hay error de conexion
                    Toast.makeText(MainActivity.this, "Error de
conexion", Toast.LENGTH_LONG).show();


                }
            });
        } catch (MqttException e) {
            e.printStackTrace();
        }

        //Parte del callback

        client.setCallback(new MqttCallback() {
            @Override
            public void connectionLost(Throwable cause) {

            }
```

```
            @Override
            public void messageArrived(String topic, MqttMessage
message) throws Exception {
                //Aqui cuando los mensajes lleguen
                //Card 2

                if(topic.matches("zbos/audio/volume/event")){
                    textVol.setText(new
String(message.getPayload()));
                }
            }

            @Override
            public void deliveryComplete(IMqttDeliveryToken token)
{

            }
        });

    }

    public void handleClick(View v) {
        if (v.getId() == R.id.msg1) {
            String topic = topicStr;
            String message = "Hola mundo, soy james";
            try {
                client.publish(topic, message.getBytes(),0,
false);
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }

        } else if(v.getId() == R.id.act1) {
            String topic = topicStr3;
            String message = "{\"id\":\"_sqwya9yjsw\"}";
            try {
                client.publish(topic, message.getBytes(),0,
false);
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }

        }else if(v.getId() == R.id.vac1) {
            String topic = topicStr;
            String message = "";
            try {
                client.publish(topic, message.getBytes(),0,
false);
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
```

```
        }else if(v.getId() == R.id.vac2) {
            String topic = topicStr;
            String message = "";
            try {
                client.publish(topic, message.getBytes(),0,
false);
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }

        }else if(v.getId() == R.id.beep) {

            String topic = topicStr2;
            String payload = "";
            byte[] encodedPayload = new byte[0];
            try {
                encodedPayload = payload.getBytes("UTF-8");
                MqttMessage message = new
MqttMessage(encodedPayload);
                client.publish(topic, message);
            } catch (UnsupportedEncodingException | MqttException
e) {
                e.printStackTrace();
            }

        }
    }

    private void suscripcionTopics(){
        try {
            client.subscribe("zbos/audio/volume/event", 0);

client.subscribe("zbos/system/robot/identification/event", 0);
            client.subscribe("zbos/status/battery/event", 0);
            client.subscribe("zbos/slam/charging/goto/started",
0);
        }catch (MqttException e){
            e.printStackTrace();
        }
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.pdhn.mqttdemotfg2">

    <uses-permission android:name="android.permission.WAKE_LOCK"
/>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.READ_PHONE_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.NoActionBar">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service
android:name="org.eclipse.paho.android.service.MqttService" >
        </service>
    </application>

</manifest>
```

Strings.xml

```
<resources>
    <string name="app_name">MQTTdemoTFG2</string>
    <string name="nombre">App prueba TFG Miguel</string>
    <string name="boton1">Msg Inical</string>
    <string name="boton2">Act Inicial</string>
    <string name="boton3">Vacio</string>
    <string name="boton4">Vacio</string>
    <string name="boton5">Beep</string>
</resources>
```

Colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#fafafa</color>
    <color name="colorPrimaryDark">#fff</color>
    <color name="colorAccent">#D81B60</color>


    <array name="reds">
        <item>@color/md_red_500</item>
        <item>@color/md_red_50</item>
        <item>@color/md_red_100</item>
        <item>@color/md_red_200</item>
        <item>@color/md_red_300</item>
        <item>@color/md_red_400</item>
        <item>@color/md_red_600</item>
        <item>@color/md_red_700</item>
        <item>@color/md_red_800</item>
        <item>@color/md_red_900</item>
        <item>@color/md_red_A100</item>
        <item>@color/md_red_A200</item>
        <item>@color/md_red_A400</item>
        <item>@color/md_red_A700</item>
    </array>
    <color name="md_red_50">#ffebee</color>
    <color name="md_red_100">#ffcdd2</color>
    <color name="md_red_200">#ef9a9a</color>
    <color name="md_red_300">#e57373</color>
    <color name="md_red_400">#ef5350</color>
    <color name="md_red_500">#f44336</color>
    <color name="md_red_600">#e53935</color>
    <color name="md_red_700">#d32f2f</color>
    <color name="md_red_800">#c62828</color>
    <color name="md_red_900">#b71c1c</color>
    <color name="md_red_A100">#ff8a80</color>
    <color name="md_red_A200">#ff5252</color>
    <color name="md_red_A400">#ff1744</color>
    <color name="md_red_A700">#d50000</color>
    <color name="md_red_500_25">#40f44336</color>
    <color name="md_red_500_50">#80f44336</color>
    <color name="md_red_500_75">#c0f44336</color>
    <!-- pinks -->
    <array name="pinks">
        <item>@color/md_pink_500</item>
        <item>@color/md_pink_50</item>
        <item>@color/md_pink_100</item>
        <item>@color/md_pink_200</item>
        <item>@color/md_pink_300</item>
        <item>@color/md_pink_400</item>
        <item>@color/md_pink_600</item>
        <item>@color/md_pink_700</item>
        <item>@color/md_pink_800</item>
        <item>@color/md_pink_900</item>
        <item>@color/md_pink_A100</item>
        <item>@color/md_pink_A200</item>
        <item>@color/md_pink_A400</item>
        <item>@color/md_pink_A700</item>
    </array>
    <color name="md_pink_50">#fce4ec</color>
```

```
<color name="md_pink_100">#f8bbd0</color>
<color name="md_pink_200">#f48fb1</color>
<color name="md_pink_300">#f06292</color>
<color name="md_pink_400">#ec407a</color>
<color name="md_pink_500">#e91e63</color>
<color name="md_pink_600">#d81b60</color>
<color name="md_pink_700">#c2185b</color>
<color name="md_pink_800">#ad1457</color>
<color name="md_pink_900">#880e4f</color>
<color name="md_pink_A100">#ff80ab</color>
<color name="md_pink_A200">#ff4081</color>
<color name="md_pink_A400">#f50057</color>
<color name="md_pink_A700">#c51162</color>
<color name="md_pink_500_25">#40e91e63</color>
<color name="md_pink_500_50">#80e91e63</color>
<color name="md_pink_500_75">#c0e91e63</color>
<!-- purples -->
<array name="purples">
    <item>@color/md_purple_500</item>
    <item>@color/md_purple_50</item>
    <item>@color/md_purple_100</item>
    <item>@color/md_purple_200</item>
    <item>@color/md_purple_300</item>
    <item>@color/md_purple_400</item>
    <item>@color/md_purple_600</item>
    <item>@color/md_purple_700</item>
    <item>@color/md_purple_800</item>
    <item>@color/md_purple_900</item>
    <item>@color/md_purple_A100</item>
    <item>@color/md_purple_A200</item>
    <item>@color/md_purple_A400</item>
    <item>@color/md_purple_A700</item>
</array>
<color name="md_purple_50">#f3e5f5</color>
<color name="md_purple_100">#e1bee7</color>
<color name="md_purple_200">#ce93d8</color>
<color name="md_purple_300">#ba68c8</color>
<color name="md_purple_400">#ab47bc</color>
<color name="md_purple_500">#9c27b0</color>
<color name="md_purple_600">#8e24aa</color>
<color name="md_purple_700">#7b1fa2</color>
<color name="md_purple_800">#6a1b9a</color>
<color name="md_purple_900">#4a148c</color>
<color name="md_purple_A100">#ea80fc</color>
<color name="md_purple_A200">#e040fb</color>
<color name="md_purple_A400">#d500f9</color>
<color name="md_purple_A700">#aa00ff</color>
<color name="md_purple_500_25">#409c27b0</color>
<color name="md_purple_500_50">#809c27b0</color>
<color name="md_purple_500_75">#c09c27b0</color>
<!-- deep purples -->
<array name="deep_purples">
    <item>@color/md_deep_purple_500</item>
    <item>@color/md_deep_purple_50</item>
    <item>@color/md_deep_purple_100</item>
    <item>@color/md_deep_purple_200</item>
    <item>@color/md_deep_purple_300</item>
    <item>@color/md_deep_purple_400</item>
    <item>@color/md_deep_purple_600</item>
```

```
    <item>@color/md_deep_purple_700</item>
    <item>@color/md_deep_purple_800</item>
    <item>@color/md_deep_purple_900</item>
    <item>@color/md_deep_purple_A100</item>
    <item>@color/md_deep_purple_A200</item>
    <item>@color/md_deep_purple_A400</item>
    <item>@color/md_deep_purple_A700</item>
</array>
<color name="md_deep_purple_50">#ede7f6</color>
<color name="md_deep_purple_100">#d1c4e9</color>
<color name="md_deep_purple_200">#b39ddb</color>
<color name="md_deep_purple_300">#9575cd</color>
<color name="md_deep_purple_400">#7e57c2</color>
<color name="md_deep_purple_500">#673ab7</color>
<color name="md_deep_purple_600">#5e35b1</color>
<color name="md_deep_purple_700">#512da8</color>
<color name="md_deep_purple_800">#4527a0</color>
<color name="md_deep_purple_900">#311b92</color>
<color name="md_deep_purple_A100">#b388ff</color>
<color name="md_deep_purple_A200">#7c4dff</color>
<color name="md_deep_purple_A400">#651fff</color>
<color name="md_deep_purple_A700">#6200ea</color>
<color name="md_deep_purple_500_25">#40673ab7</color>
<color name="md_deep_purple_500_50">#80673ab7</color>
<color name="md_deep_purple_500_75">#c0673ab7</color>
<!-- indigos -->
<array name="indigos">
    <item>@color/md_indigo_500</item>
    <item>@color/md_indigo_50</item>
    <item>@color/md_indigo_100</item>
    <item>@color/md_indigo_200</item>
    <item>@color/md_indigo_300</item>
    <item>@color/md_indigo_400</item>
    <item>@color/md_indigo_600</item>
    <item>@color/md_indigo_700</item>
    <item>@color/md_indigo_800</item>
    <item>@color/md_indigo_900</item>
    <item>@color/md_indigo_A100</item>
    <item>@color/md_indigo_A200</item>
    <item>@color/md_indigo_A400</item>
    <item>@color/md_indigo_A700</item>
</array>
<color name="md_indigo_50">#e8eaf6</color>
<color name="md_indigo_100">#c5cae9</color>
<color name="md_indigo_200">#9fa8da</color>
<color name="md_indigo_300">#7986cb</color>
<color name="md_indigo_400">#5c6bc0</color>
<color name="md_indigo_500">#3f51b5</color>
<color name="md_indigo_600">#3949ab</color>
<color name="md_indigo_700">#303f9f</color>
<color name="md_indigo_800">#283593</color>
<color name="md_indigo_900">#1a237e</color>
<color name="md_indigo_A100">#8c9eff</color>
<color name="md_indigo_A200">#536dfe</color>
<color name="md_indigo_A400">#3d5afe</color>
<color name="md_indigo_A700">#304ffe</color>
<color name="md_indigo_500_25">#403f51b5</color>
<color name="md_indigo_500_50">#803f51b5</color>
<color name="md_indigo_500_75">#c03f51b5</color>
```

```
<!--blues-->
<array name="blues">
    <item>@color/md_blue_500</item>
    <item>@color/md_blue_50</item>
    <item>@color/md_blue_100</item>
    <item>@color/md_blue_200</item>
    <item>@color/md_blue_300</item>
    <item>@color/md_blue_400</item>
    <item>@color/md_blue_600</item>
    <item>@color/md_blue_700</item>
    <item>@color/md_blue_800</item>
    <item>@color/md_blue_900</item>
    <item>@color/md_blue_A100</item>
    <item>@color/md_blue_A200</item>
    <item>@color/md_blue_A400</item>
    <item>@color/md_blue_A700</item>
</array>
<color name="md_blue_50">#e3f2fd</color>
<color name="md_blue_100">#bbdefb</color>
<color name="md_blue_200">#90caf9</color>
<color name="md_blue_300">#64b5f6</color>
<color name="md_blue_400">#42a5f5</color>
<color name="md_blue_500">#2196f3</color>
<color name="md_blue_600">#1e88e5</color>
<color name="md_blue_700">#1976d2</color>
<color name="md_blue_800">#1565c0</color>
<color name="md_blue_900">#0d47a1</color>
<color name="md_blue_A100">#82b1ff</color>
<color name="md_blue_A200">#448aff</color>
<color name="md_blue_A400">#2979ff</color>
<color name="md_blue_A700">#2962ff</color>
<color name="md_blue_500_25">#402196f3</color>
<color name="md_blue_500_50">#802196f3</color>
<color name="md_blue_500_75">#c02196f3</color>
<!-- light blues-->
<array name="light_blues">
    <item>@color/md_light_blue_500</item>
    <item>@color/md_light_blue_50</item>
    <item>@color/md_light_blue_100</item>
    <item>@color/md_light_blue_200</item>
    <item>@color/md_light_blue_300</item>
    <item>@color/md_light_blue_400</item>
    <item>@color/md_light_blue_600</item>
    <item>@color/md_light_blue_700</item>
    <item>@color/md_light_blue_800</item>
    <item>@color/md_light_blue_900</item>
    <item>@color/md_light_blue_A100</item>
    <item>@color/md_light_blue_A200</item>
    <item>@color/md_light_blue_A400</item>
    <item>@color/md_light_blue_A700</item>
</array>
<color name="md_light_blue_50">#e1f5fe</color>
<color name="md_light_blue_100">#b3e5fc</color>
<color name="md_light_blue_200">#81d4fa</color>
<color name="md_light_blue_300">#4fc3f7</color>
<color name="md_light_blue_400">#29b6f6</color>
<color name="md_light_blue_500">#03a9f4</color>
<color name="md_light_blue_600">#039be5</color>
<color name="md_light_blue_700">#0288d1</color>
```

```
<color name="md_light_blue_800">#0277bd</color>
<color name="md_light_blue_900">#01579b</color>
<color name="md_light_blue_A100">#80d8ff</color>
<color name="md_light_blue_A200">#40c4ff</color>
<color name="md_light_blue_A400">#00b0ff</color>
<color name="md_light_blue_A700">#0091ea</color>
<color name="md_light_blue_500_25">#4003a9f4</color>
<color name="md_light_blue_500_50">#8003a9f4</color>
<color name="md_light_blue_500_75">#c003a9f4</color>
<!-- cyans -->
<array name="cyans">
    <item>@color/md_cyan_500</item>
    <item>@color/md_cyan_50</item>
    <item>@color/md_cyan_100</item>
    <item>@color/md_cyan_200</item>
    <item>@color/md_cyan_300</item>
    <item>@color/md_cyan_400</item>
    <item>@color/md_cyan_600</item>
    <item>@color/md_cyan_700</item>
    <item>@color/md_cyan_800</item>
    <item>@color/md_cyan_900</item>
    <item>@color/md_cyan_A100</item>
    <item>@color/md_cyan_A200</item>
    <item>@color/md_cyan_A400</item>
    <item>@color/md_cyan_A700</item>
</array>
<color name="md_cyan_50">#e0f7fa</color>
<color name="md_cyan_100">#b2ebf2</color>
<color name="md_cyan_200">#80deea</color>
<color name="md_cyan_300">#4dd0e1</color>
<color name="md_cyan_400">#26c6da</color>
<color name="md_cyan_500">#00bcd4</color>
<color name="md_cyan_600">#00acc1</color>
<color name="md_cyan_700">#0097a7</color>
<color name="md_cyan_800">#00838f</color>
<color name="md_cyan_900">#006064</color>
<color name="md_cyan_A100">#84ffff</color>
<color name="md_cyan_A200">#18ffff</color>
<color name="md_cyan_A400">#00e5ff</color>
<color name="md_cyan_A700">#00b8d4</color>
<color name="md_cyan_500_25">#4000bcd4</color>
<color name="md_cyan_500_50">#8000bcd4</color>
<color name="md_cyan_500_75">#c000bcd4</color>
<!-- teals -->
<array name="teals">
    <item>@color/md_teal_500</item>
    <item>@color/md_teal_50</item>
    <item>@color/md_teal_100</item>
    <item>@color/md_teal_200</item>
    <item>@color/md_teal_300</item>
    <item>@color/md_teal_400</item>
    <item>@color/md_teal_600</item>
    <item>@color/md_teal_700</item>
    <item>@color/md_teal_800</item>
    <item>@color/md_teal_900</item>
    <item>@color/md_teal_A100</item>
    <item>@color/md_teal_A200</item>
    <item>@color/md_teal_A400</item>
    <item>@color/md_teal_A700</item>
```

```
</array>
<color name="md_teal_50">#e0f2f1</color>
<color name="md_teal_100">#b2dfdb</color>
<color name="md_teal_200">#80cbc4</color>
<color name="md_teal_300">#4db6ac</color>
<color name="md_teal_400">#26a69a</color>
<color name="md_teal_500">#009688</color>
<color name="md_teal_600">#00897b</color>
<color name="md_teal_700">#00796b</color>
<color name="md_teal_800">#00695c</color>
<color name="md_teal_900">#004d40</color>
<color name="md_teal_A100">#a7ffeb</color>
<color name="md_teal_A200">#64ffda</color>
<color name="md_teal_A400">#1de9b6</color>
<color name="md_teal_A700">#00bfa5</color>
<color name="md_teal_500_25">#40009688</color>
<color name="md_teal_500_50">#80009688</color>
<color name="md_teal_500_75">#c8009688</color>
<!-- greens -->
<array name="greens">
    <item>@color/md_green_500</item>
    <item>@color/md_green_50</item>
    <item>@color/md_green_100</item>
    <item>@color/md_green_200</item>
    <item>@color/md_green_300</item>
    <item>@color/md_green_400</item>
    <item>@color/md_green_600</item>
    <item>@color/md_green_700</item>
    <item>@color/md_green_800</item>
    <item>@color/md_green_900</item>
    <item>@color/md_green_A100</item>
    <item>@color/md_green_A200</item>
    <item>@color/md_green_A400</item>
    <item>@color/md_green_A700</item>
</array>
<color name="md_green_50">#e8f5e9</color>
<color name="md_green_100">#c8e6c9</color>
<color name="md_green_200">#a5d6a7</color>
<color name="md_green_300">#81c784</color>
<color name="md_green_400">#66bb6a</color>
<color name="md_green_500">#4caf50</color>
<color name="md_green_600">#43a047</color>
<color name="md_green_700">#388e3c</color>
<color name="md_green_800">#2e7d32</color>
<color name="md_green_900">#1b5e20</color>
<color name="md_green_A100">#b9f6ca</color>
<color name="md_green_A200">#69f0ae</color>
<color name="md_green_A400">#00e676</color>
<color name="md_green_A700">#00c853</color>
<color name="md_green_500_25">#404caf50</color>
<color name="md_green_500_50">#804caf50</color>
<color name="md_green_500_75">#c04caf50</color>
<!--light greens-->
<array name="light_greens">
    <item>@color/md_light_green_500</item>
    <item>@color/md_light_green_50</item>
    <item>@color/md_light_green_100</item>
    <item>@color/md_light_green_200</item>
    <item>@color/md_light_green_300</item>
```

```
    <item>@color/md_light_green_400</item>
    <item>@color/md_light_green_600</item>
    <item>@color/md_light_green_700</item>
    <item>@color/md_light_green_800</item>
    <item>@color/md_light_green_900</item>
    <item>@color/md_light_green_A100</item>
    <item>@color/md_light_green_A200</item>
    <item>@color/md_light_green_A400</item>
    <item>@color/md_light_green_A700</item>
</array>
<color name="md_light_green_50">#f1f8e9</color>
<color name="md_light_green_100">#dcedc8</color>
<color name="md_light_green_200">#c5e1a5</color>
<color name="md_light_green_300">#aed581</color>
<color name="md_light_green_400">#9ccc65</color>
<color name="md_light_green_500">#8bc34a</color>
<color name="md_light_green_600">#7cb342</color>
<color name="md_light_green_700">#689f38</color>
<color name="md_light_green_800">#558b2f</color>
<color name="md_light_green_900">#33691e</color>
<color name="md_light_green_A100">#ccff90</color>
<color name="md_light_green_A200">#b2ff59</color>
<color name="md_light_green_A400">#76ff03</color>
<color name="md_light_green_A700">#64dd17</color>
<color name="md_light_green_500_25">#408bc34a</color>
<color name="md_light_green_500_50">#808bc34a</color>
<color name="md_light_green_500_75">#c88bc34a</color>
<!-- limes -->
<array name="limes">
    <item>@color/md_lime_500</item>
    <item>@color/md_lime_50</item>
    <item>@color/md_lime_100</item>
    <item>@color/md_lime_200</item>
    <item>@color/md_lime_300</item>
    <item>@color/md_lime_400</item>
    <item>@color/md_lime_600</item>
    <item>@color/md_lime_700</item>
    <item>@color/md_lime_800</item>
    <item>@color/md_lime_900</item>
    <item>@color/md_lime_A100</item>
    <item>@color/md_lime_A200</item>
    <item>@color/md_lime_A400</item>
    <item>@color/md_lime_A700</item>
</array>
<color name="md_lime_50">#f9fbe7</color>
<color name="md_lime_100">#f0f4c3</color>
<color name="md_lime_200">#e6ee9c</color>
<color name="md_lime_300">#dce775</color>
<color name="md_lime_400">#d4e157</color>
<color name="md_lime_500">#cddc39</color>
<color name="md_lime_600">#c0ca33</color>
<color name="md_lime_700">#afb42b</color>
<color name="md_lime_800">#9e9d24</color>
<color name="md_lime_900">#827717</color>
<color name="md_lime_A100">#f4ff81</color>
<color name="md_lime_A200">#eeff41</color>
<color name="md_lime_A400">#c6ff00</color>
<color name="md_lime_A700">#aeea00</color>
<color name="md_lime_500_25">#40cddc39</color>
```

```
<color name="md_lime_500_50">#80cddc39</color>
<color name="md_lime_500_75">#c0cddc39</color>
<!--yellows -->
<array name="yellows">
    <item>@color/md_yellow_500</item>
    <item>@color/md_yellow_50</item>
    <item>@color/md_yellow_100</item>
    <item>@color/md_yellow_200</item>
    <item>@color/md_yellow_300</item>
    <item>@color/md_yellow_400</item>
    <item>@color/md_yellow_600</item>
    <item>@color/md_yellow_700</item>
    <item>@color/md_yellow_800</item>
    <item>@color/md_yellow_900</item>
    <item>@color/md_yellow_A100</item>
    <item>@color/md_yellow_A200</item>
    <item>@color/md_yellow_A400</item>
    <item>@color/md_yellow_A700</item>
</array>
<color name="md_yellow_50">#fffde7</color>
<color name="md_yellow_100">#fff9c4</color>
<color name="md_yellow_200">#fff59d</color>
<color name="md_yellow_300">#fff176</color>
<color name="md_yellow_400">#ffee58</color>
<color name="md_yellow_500">#ffeb3b</color>
<color name="md_yellow_600">#fdd835</color>
<color name="md_yellow_700">#fbc02d</color>
<color name="md_yellow_800">#f9a825</color>
<color name="md_yellow_900">#f57f17</color>
<color name="md_yellow_A100">#ffff8d</color>
<color name="md_yellow_A200">#ffff00</color>
<color name="md_yellow_A400">#ffea00</color>
<color name="md_yellow_A700">#ffd600</color>
<color name="md_yellow_500_25">#40ffeb3b</color>
<color name="md_yellow_500_50">#80ffeb3b</color>
<color name="md_yellow_500_75">#c0ffeb3b</color>
<!-- ambers -->
<array name="ambers">
    <item>@color/md_amber_500</item>
    <item>@color/md_amber_50</item>
    <item>@color/md_amber_100</item>
    <item>@color/md_amber_200</item>
    <item>@color/md_amber_300</item>
    <item>@color/md_amber_400</item>
    <item>@color/md_amber_600</item>
    <item>@color/md_amber_700</item>
    <item>@color/md_amber_800</item>
    <item>@color/md_amber_900</item>
    <item>@color/md_amber_A100</item>
    <item>@color/md_amber_A200</item>
    <item>@color/md_amber_A400</item>
    <item>@color/md_amber_A700</item>
</array>
<color name="md_amber_50">#fff8e1</color>
<color name="md_amber_100">#ffecb3</color>
<color name="md_amber_200">#ffe082</color>
<color name="md_amber_300">#ffd54f</color>
<color name="md_amber_400">#ffca28</color>
<color name="md_amber_500">#ffc107</color>
```

```
<color name="md_amber_600">#ffb300</color>
<color name="md_amber_700">#ffa000</color>
<color name="md_amber_800">#ff8f00</color>
<color name="md_amber_900">#ff6f00</color>
<color name="md_amber_A100">#ffe57f</color>
<color name="md_amber_A200">#ffd740</color>
<color name="md_amber_A400">#ffc400</color>
<color name="md_amber_A700">#ffab00</color>
<color name="md_amber_500_25">#40ffc107</color>
<color name="md_amber_500_50">#80ffc107</color>
<color name="md_amber_500_75">#c0ffc107</color>
<!-- oranges -->
<array name="oranges">
    <item>@color/md_orange_500</item>
    <item>@color/md_orange_50</item>
    <item>@color/md_orange_100</item>
    <item>@color/md_orange_200</item>
    <item>@color/md_orange_300</item>
    <item>@color/md_orange_400</item>
    <item>@color/md_orange_600</item>
    <item>@color/md_orange_700</item>
    <item>@color/md_orange_800</item>
    <item>@color/md_orange_900</item>
    <item>@color/md_orange_A100</item>
    <item>@color/md_orange_A200</item>
    <item>@color/md_orange_A400</item>
    <item>@color/md_orange_A700</item>
</array>
<color name="md_orange_50">#fff3e0</color>
<color name="md_orange_100">#ffe0b2</color>
<color name="md_orange_200">#ffcc80</color>
<color name="md_orange_300">#ffb74d</color>
<color name="md_orange_400">#ffa726</color>
<color name="md_orange_500">#ff9800</color>
<color name="md_orange_600">#fb8c00</color>
<color name="md_orange_700">#f57c00</color>
<color name="md_orange_800">#ef6c00</color>
<color name="md_orange_900">#e65100</color>
<color name="md_orange_A100">#ffd180</color>
<color name="md_orange_A200">#ffab40</color>
<color name="md_orange_A400">#ff9100</color>
<color name="md_orange_A700">#ff6d00</color>
<color name="md_orange_500_25">#40ff9800</color>
<color name="md_orange_500_50">#80ff9800</color>
<color name="md_orange_500_75">#c0ff9800</color>
<!-- deep oranges -->
<array name="deep_oranges">
    <item>@color/md_deep_orange_500</item>
    <item>@color/md_deep_orange_50</item>
    <item>@color/md_deep_orange_100</item>
    <item>@color/md_deep_orange_200</item>
    <item>@color/md_deep_orange_300</item>
    <item>@color/md_deep_orange_400</item>
    <item>@color/md_deep_orange_600</item>
    <item>@color/md_deep_orange_700</item>
    <item>@color/md_deep_orange_800</item>
    <item>@color/md_deep_orange_900</item>
    <item>@color/md_deep_orange_A100</item>
    <item>@color/md_deep_orange_A200</item>
```

```
    <item>@color/md_deep_orange_A400</item>
    <item>@color/md_deep_orange_A700</item>
</array>
<color name="md_deep_orange_50">#fbe9e7</color>
<color name="md_deep_orange_100">#ffccbc</color>
<color name="md_deep_orange_200">#ffab91</color>
<color name="md_deep_orange_300">#ff8a65</color>
<color name="md_deep_orange_400">#ff7043</color>
<color name="md_deep_orange_500">#ff5722</color>
<color name="md_deep_orange_600">#f4511e</color>
<color name="md_deep_orange_700">#e64a19</color>
<color name="md_deep_orange_800">#d84315</color>
<color name="md_deep_orange_900">#bf360c</color>
<color name="md_deep_orange_A100">#ff9e80</color>
<color name="md_deep_orange_A200">#ff6e40</color>
<color name="md_deep_orange_A400">#ff3d00</color>
<color name="md_deep_orange_A700">#dd2c00</color>
<color name="md_deep_orange_500_25">#40ff5722</color>
<color name="md_deep_orange_500_50">#80ff5722</color>
<color name="md_deep_orange_500_75">#c0ff5722</color>
<!-- browns -->
<array name="browns">
    <item>@color/md_brown_500</item>
    <item>@color/md_brown_50</item>
    <item>@color/md_brown_100</item>
    <item>@color/md_brown_200</item>
    <item>@color/md_brown_300</item>
    <item>@color/md_brown_400</item>
    <item>@color/md_brown_600</item>
    <item>@color/md_brown_700</item>
    <item>@color/md_brown_800</item>
    <item>@color/md_brown_900</item>
</array>
<color name="md_brown_50">#efebe9</color>
<color name="md_brown_100">#d7ccc8</color>
<color name="md_brown_200">#bcaaa4</color>
<color name="md_brown_300">#a1887f</color>
<color name="md_brown_400">#8d6e63</color>
<color name="md_brown_500">#795548</color>
<color name="md_brown_600">#6d4c41</color>
<color name="md_brown_700">#5d4037</color>
<color name="md_brown_800">#4e342e</color>
<color name="md_brown_900">#3e2723</color>
<color name="md_brown_500_25">#40795548</color>
<color name="md_brown_500_50">#80795548</color>
<color name="md_brown_500_75">#c0795548</color>
<!--grey-->
<array name="greys">
    <item>@color/md_grey_500</item>
    <item>@color/md_grey_50</item>
    <item>@color/md_grey_100</item>
    <item>@color/md_grey_200</item>
    <item>@color/md_grey_300</item>
    <item>@color/md_grey_400</item>
    <item>@color/md_grey_600</item>
    <item>@color/md_grey_700</item>
    <item>@color/md_grey_800</item>
    <item>@color/md_grey_900</item>
    <item>@color/md_black_1000</item>
```

```
    <item>@color/md_white_1000</item>
</array>
<color name="md_grey_50">#fafafa</color>
<color name="md_grey_100">#f5f5f5</color>
<color name="md_grey_200">#eeeeee</color>
<color name="md_grey_300">#e0e0e0</color>
<color name="md_grey_400">#bdbdbd</color>
<color name="md_grey_500">#9e9e9e</color>
<color name="md_grey_600">#757575</color>
<color name="md_grey_700">#616161</color>
<color name="md_grey_800">#424242</color>
<color name="md_grey_900">#212121</color>
<color name="md_grey_500_25">#409e9e9e</color>
<color name="md_grey_500_50">#809e9e9e</color>
<color name="md_grey_500_75">#c09e9e9e</color>
<color name="md_white_1000">#ffffff</color>
<color name="md_white_1000_10">#1affffff</color>
<color name="md_white_1000_15">#22ffffff</color>
<color name="md_white_1000_20">#33ffffff</color>
<color name="md_white_1000_25">#40ffffff</color>
<color name="md_white_1000_50">#80ffffff</color>
<color name="md_white_1000_60">#99ffffff</color>
<color name="md_white_1000_75">#c0ffffff</color>
<color name="md_black_1000_10">#1a000000</color>
<color name="md_black_1000_15">#26000000</color>
<color name="md_black_1000_20">#33000000</color>
<color name="md_black_1000_25">#40000000</color>
<color name="md_black_1000_50">#80000000</color>
<color name="md_black_1000_75">#c0000000</color>
<color name="md_black_1000">#000000</color>
<!--blue grey-->
<array name="blue_greys">
    <item>@color/md_blue_grey_500</item>
    <item>@color/md_blue_grey_50</item>
    <item>@color/md_blue_grey_100</item>
    <item>@color/md_blue_grey_200</item>
    <item>@color/md_blue_grey_300</item>
    <item>@color/md_blue_grey_400</item>
    <item>@color/md_blue_grey_600</item>
    <item>@color/md_blue_grey_700</item>
    <item>@color/md_blue_grey_800</item>
    <item>@color/md_blue_grey_900</item>
</array>
<color name="md_blue_grey_50">#eceff1</color>
<color name="md_blue_grey_100">#cfd8dc</color>
<color name="md_blue_grey_200">#b0bec5</color>
<color name="md_blue_grey_300">#90a4ae</color>
<color name="md_blue_grey_400">#78909c</color>
<color name="md_blue_grey_500">#607d8b</color>
<color name="md_blue_grey_600">#546e7a</color>
<color name="md_blue_grey_700">#455a64</color>
<color name="md_blue_grey_800">#37474f</color>
<color name="md_blue_grey_900">#263238</color>
<color name="md_blue_grey_500_25">#40607d8b</color>
<color name="md_blue_grey_500_50">#80607d8b</color>
<color name="md_blue_grey_500_75">#c0607d8b</color>
<!--Texts-->
<array name="texts_white">
    <item>@color/md_text_white</item>
```

```
        <item>@color/md_text_white_87</item>
        <item>@color/md_secondary_text_icons_white</item>
        <item>@color/md_disabled_hint_text_white</item>
        <item>@color/md_divider_white</item>
    </array>
    <color name="md_text_white">#ffffffff</color>
    <color name="md_text_white_87">#dfffffff</color>
    <color name="md_secondary_text_icons_white">#b3ffffff</color>
    <color name="md_disabled_hint_text_white">#4dffffff</color>
    <color name="md_divider_white">#1fffffff</color>
    <array name="texts_black">
        <item>@color/md_text</item>
        <item>@color/md_secondary_text_icons</item>
        <item>@color/md_disabled_hint_text</item>
        <item>@color/md_divider</item>
    </array>
    <color name="md_text">#df000000</color>
    <color name="md_secondary_text_icons">#8a000000</color>
    <color name="md_disabled_hint_text">#4c000000</color>
    <color name="md_divider">#1f000000</color>
    <!--Falcon-->
    <array name="falcon">
        <item>@color/md_falcon_400</item>
        <item>@color/md_falcon_500</item>
        <item>@color/md_falcon_700</item>
    </array>
    <color name="md_falcon_400">#ff38628b</color>
    <color name="md_falcon_500">#384e77</color>
    <color name="md_falcon_700">#2b3e5f</color>
    <color name="md_falcon_A400">#598bae</color>
    <color name="md_falcon_500_25">#40384e77</color>
    <color name="md_falcon_500_50">#80384e77</color>
    <color name="md_falcon_500_75">#c0384e77</color>
    <color name="md_falcon_A400_25">#40598bae</color>

    <color name="md_statusbar_translucent">#4000</color>
</resources>
```