UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

Escuela de Ingenierías Industriales

Ingeniería de Sistemas y Automática

Trabajo Fin de Grado

# Sistema Automático de Control de Asistencia en el laboratorio de RoboRescue UMA

Automatic attendance control system in the RoboRescue UMA laboratory

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Autor: Miguel Fuentes Hervás

Tutor: Antonio José Muñoz Ramírez

Málaga, Junio de 2023

UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

# Sistema Automático de Control de Asistencia en el laboratorio de RoboRescue UMA


- **Autor:** Miguel Fuentes Hervás.
- **Tutor:** Antonio José Muñoz Ramírez.
- **Departamento:** Ingeniería de Sistemas y Automática.



- **Titulación:** Grado en Ingeniería Electrónica, Robótica y Mecatrónica.
- **Palabras clave** RoboRescue, Control de Presencia, Bluetooth, Bluetooth Low Energy, Automatización, Código Libre, Home Assistant, RaspBerry Pi.


## Resumen


En este proyecto se presenta el prototipado e implementación de un sistema de control de presencia para el laboratorio de RoboRescue UMA. Dicho sistema debe permitir a los participantes del programa consultar en tiempo real qué compañeros se encuentran en el laboratorio a través de la página web de RoboRescue.

Como núcleo del proyecto se presenta Home Assistant OS, un sistema operativo de código abierto enfocado a las aplicaciones IOT, automatización y domótica a nivel de usuario. Se realiza por tanto un análisis de las diferentes tecnologías que se pueden emplear para realizar el control de presencia en esta plataforma, valorando su idoneidad en el contexto de las necesidades particulares del proyecto.

Se prioriza durante todo el desarrollo el empleo de herramientas software de código abierto que garanticen que el proyecto sea ampliable y replicable, poniendo en valor la importancia de la existencia de estas herramientas.

UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

# Automatic attendance control system in the

# RoboRescue UMA laboratory


- **Author:** Miguel Fuentes Hervás.
- **Tutor:** Antonio José Muñoz Ramírez.
- **Department:** Automation and Systems Engineering.


- **Degree:** Electronics, Robotics and Mechatronics Engineering.

- **Keywords** RoboRescue, Presence Tracking, Bluetooth, Bluetooth Low Energy, Automation, Open Source, Home Assistant, RaspBerry Pi.


## Abstract


This project presents the prototyping and implementation processes of a presence control system for the RoboRescue UMA laboratory. The system's purpose is to allow program participants to check in real-time which colleagues are present in the lab by accessing the RoboRescue website.

The relies on Home Assistant OS, an open-source operating system designed for IoT applications, automation, and user-level home automation. Various technologies are analyzed to determine their suitability for implementing the presence control feature within this platform, taking into consideration the specific needs of the project.

Throughout the development process, utilizing open-source software tools to ensure scalability and reproducibility of the project is prioritized, emphasizing the importance of these tools' existance.

# Glosario de Términos

**Hub** Dispositivo central de conexión en una red. Permite que varios dispositivos se conecten entre sí y compartan información. Actúa como un concentrador pasivo que replica y distribuye los datos a todos los dispositivos conectados.

**Wearable** Dispositivo electrónico portátil empleado como accesorio corporal, incorporando funcionalidades relacionadas con la salud, la comunicación y la interacción con otros dispositivos. Estos dispositivos suelen estar equipados con sensores y se conectan a otros dispositivos para compartir información y realizar tareas adicionales. Se presentan normalmente en forma de reloj (smartWatch) o pulsera (smartBand)

**Open Source** Software de código abierto. Su código fuente es accesible, utilizable, modificable y distribuible por cualquier persona.

**Máquina Virtual** Software que permite ejecutar sistemas operativos diferentes al anfitrión en una misma máquina física.

**Contenido Estático** Información mostrada en las páginas web, normalmente presentada como código HTML que no se actualiza según la interacción del usuario. Se envían al navegador del cliente tal como se almacenan en el servidor.

**Contenido Dinámico** Contenido web generado en tiempo real según la interacción del usuario con la web. Suele estar basado en lenguajes de programación del lado del servidor, es decir, cuyo costo computacional recae enteramente en el servidor, realizando un preprocesamiento para luego enviarlo al usuario.

**GitHUB** Plataforma web basada en repositorios que permite el desarrollo colaborativo de software. Los desarrolladores pueden tanto compartir código como colaborar con aportes en diferentes versiones de códigos ajenos.

**Dirección estática** En el contexto del proyecto, relacionado con direcciones MAC e IP. Dirección que no cambia en el tiempo.

**Dirección dinámica** En el contexto del proyecto, relacionado con direcciones MAC e IP. Dirección asociada a un usuario de forma temporal. Se renueva algorítmicamente o según recursos disponibles.

**Proxy** Intermediario entre las peticiones de recursos que realiza un cliente a un servidor. Las peticiones pasan a ser realizadas por el proxy, en nombre del cliente, sin necesidad de que el servidor conozca su identidad.

# Acrónimos

| | |
|---|---|
| **IOT** | Internet of Things, Internet de las Cosas |
| **HASS/HassOS** | Home Assistant / Home Assistant Operating System |
| **BLE** | Bluetooth Low Energy |
| **GPS** | Global Positioning System |
| **HTTP** | Hypertext Transfer Protocol |
| **HTML** | HyperText Markup Lenguage |
| **REST** | Representational State Transfer |
| **IRK** | Identity Resolving Key |
| **WP** | WordPress |
| **JSON** | JavaScript Object Notation |
| **cmd** | Command promp |
| **API** | Application Programming Interface |
| **PHP** | Hypertext Preprocessor |
| **MAC** | Media Access Control |
| **PC** | Personal Computer |
| **MQTT** | Message Queuing Telemetry Transport |
| **IP** | Internet Protocol |
| **ICMP** | Internet Control Message Protocol |
| **TCP** | Transmision Control Protocol |

| | |
|---|---|
| **UDP** | User Datagram Protocol |
| **RHEL** | Red Hat Enterprise Linux |
| **UUID** | Universally Unique IDentifier |

# Índice general

# Índice de figuras

# Capítulo 1

# Introducción

## 1.1 Introducción

El Internet de las Cosas (IOT), entendido este como una red local de dispositivos inteligentes interconectados, forma progresivamente parte de la cotidianidad de cada vez más personas. Lejos de ser algo reservado al ámbito comercial existen una variedad de alternativas apoyadas en el esfuerzo de la comunidad de desarrolladores que se presenta en forma de código abierto, documentación y plataformas de desarrollo, entre otras.

En la coyuntura expuesta se presenta Home Assistant [1], una plataforma gratis, de código abierto que nace en 2013 para acercar la automatización del hogar a un mayor público, permitiendo centralizar en un mismo servidor la coordinación de la mayoría de los dispositivos IOT comerciales, así como de los dispositivos IOT creados por los propios usuarios. Home Assistant es el eje sobre el que se construyen las diferentes aproximaciones que permitirán cumplir los objetivos del presente documento.

Con el desarrollo de este Trabajo de Fin de Grado se pretende realizar un acercamiento al campo de la automatización e interconexión de dispositivos inteligentes siguiendo la filosofía *open-source*. Para ello, se estudian diversas alternativas para finalizar proponiendo una solución a la problemática del control de presencia en el laboratorio del programa RoboRescue UMA. En el proceso se dotará a dicho laboratorio de una plataforma de desarrollo enfocada en este campo, favoreciendo posteriores ampliaciones de la red de dispositivos y funcionalidades que la componen.

## 1.2 Antecedentes

El laboratorio de desarrollo de robot de rescate del equipo RoboRescue UMA cuenta con un sistema de control de presencia manual, llevado a cabo mediante la herramienta Shifts integrada en Microsoft Teams [2].

Empleado el sistema actual cada uno de los participantes del equipo debe indicar a través de la aplicación que su turno ha comenzado una vez ha accedido al laboratorio. Posteriormente, cuando abandona el laboratorio, debe indicar entonces que finaliza su turno. Sin embargo, es habitual encontrar desfases entre el tiempo de estancia en el taller de los participantes del programa y el tiempo que se registra como válido, principalmente debido a errores humanos.

Durante toda duración del turno el usuario permite que los miembros responsables del proyecto puedan consultar su historial de ubicaciones GPS. Pese a que Shifts actualmente cuenta con la opción de monitorización de la ubicación de los miembros del equipo durante sus turnos, estos pueden comenzar el turno desde cualquier emplazamiento, así como durante un turno abandonar el taller sin que la aplicación lo notifique. Es responsabilidad por tanto de los coordinadores del equipo verificar la presencia de los miembros en el taller de forma activa, cosa que es indeseable debido al esfuerzo añadido que supone y la sensación implícita de desconfianza que puede llegar a generar. Adicionalmente, olvidar interrumpir los turnos implica contabilizar horas extra que no representan el trabajo real del participante del programa, tiempo en el que dicho participante está, generalmente de forma no consciente, compartiendo su historial de ubicaciones permitiendo que posteriormente pueda ser consultado.

Por otra parte, y como factor determinante debido a la naturaleza presencial y colaborativa del equipo RoboRescue UMA, es de alto interés para los participantes conocer quién se encuentra en el laboratorio en cada momento, de forma que les motive a acudir al mismo al verificar que se encuentren las personas con las que están colaborando de forma más estrecha.

En este contexto surge la necesidad por parte de los coordinadores del equipo en consonancia con las peticiones de los participantes de automatizar/agilizar el sistema de control de presencia, de forma que sea intuitivo, más sencillo, y facilite las tareas organizativas y la coordinación de la evolución del proyecto.

## 1.3 Objetivos

El objetivo principal de este Trabajo de Fin de Grado es prototipar e implementar un sistema distribuido que permita la detección e identificación de dispositivos móviles, tales como Smartphones y Smartwatches asociados a usuarios únicos. De esta forma, el sistema debe permitir la visualización en forma de listado en tiempo real de estos dispositivos en la página web de RoboRescue UMA [3]. Este listado será visible exclusivamente para los miembros que tengan credenciales de acceso en la página web anteriormente mencionada.

La correcta implementación del sistema mencionado depende de la consecución de las siguientes subtareas:

- Instalación y configuración apropiada del sistema operativo Home Assistant.
- Identificación e implementación del método óptimo para realizar el seguimiento de los dispositivos asociados a cada persona.
- Estudio e implementación de técnicas que permitan exportar diferentes conjuntos de datos desde el sistema operativo Home Assistant a la web.

- Configuración del servidor que actuará de receptor de los datos que necesiten ser exportados.

- Despliegue bajo solicitud del listado en tiempo real de los asistentes en la página web del programa RoboRescue.

Adicionalmente, se implementará un sistema de conteo que permitirá almacenar y consultar en una base de datos propia el cómputo de horas mensuales de cada miembro. La base de datos que almacena dicha información comparte servidor con la página web, pero su acceso y consulta no es accesible a todos los miembros del laboratorio desde dicha página, sino que necesita previo inicio de sesión en el servidor para ser llevado a cabo.

## 1.4 ¿Por qué es interesante este trabajo?

En primer lugar, el trabajo aquí documentado responde a una petición expresa por parte del entorno de RoboRescue, proponiendo una solución implementable que se ajusta a unos requisitos concretos. Además, mediante su desarrollo se pretenden poner en valor y materializar las aptitudes y conocimientos adquiridos por el alumno que finaliza el Grado en Ingeniería Electrónica, Robótica y Mecatrónica. Durante todo el proceso se abordan cuestiones que, si bien no son materia inmediata de estudio en el plan docente del grado anteriormente mencionado, justifican la labor del ingeniero como agente versátil, no limitado a la aplicación inmediata de los conocimientos técnicos que posee, sino capaz de adaptarse y desempeñar correctamente en entornos multidisciplinares cambiantes.

Por otro lado, se hace un énfasis en la disponibilidad de herramientas de código abierto gratuitas, creadas por y para la comunidad. La filosofía *open-source* universaliza el acceso multitud de herramientas software muy potentes, fomentando la colaboración entre desarrolladores y generando en el proceso un ecosistema en el que se promueve el avance tecnológico. Cualquier persona, independientemente de su nivel de recursos puede emplear estas herramientas, reduciendo la brecha digital y democratizando el acceso al conocimiento tecnológico. Es importante poner esto en valor, ya que, en de no ser estas las circunstancias, este proyecto no habría podido ser llevado a cabo.

Por último, respondiendo a las tendencias de desarrollo de los últimos tiempos y a la expansión de los ecosistemas interactivos que facilitan y optimizan la interacción humano-máquina, se proporciona al laboratorio un punto de partida para seguir explorando en esta dirección. El laboratorio de RoboRescue UMA podrá así dar cabida a proyectos personales o grupales en materia de IOT y automatización que pueden llegar a ser muy útiles y didácticos aprovechando la implementación aquí presentada y los recursos disponibles. Se hace así del laboratorio un posible entorno de aprendizaje y experimentación complementario y compatible con las actividades que actualmente se llevan allí a cabo.

## 1.5 Contenido de la Memoria

Se realizará a continuación un listado y breve resumen de los capítulos y documentos anexos que componen la memoria de este TFG

- ***Capítulo 2. Estudio Inicial del Ecosistema Hardware y Software***. En este capítulo se realizará una descripción contextual de las herramientas que han resultado más relevantes a la hora de desarrollar el proyecto.
- ***Capítulo 3. Diseño de la Arquitectura.*** Este capítulo desarrolla la valoración que se hace de cada una de las herramientas consideradas oportunas para la realización del trabajo. Concluye proponiendo un diseño conformado por algunas de estas herramientas cuya implementación se completará en el capítulo siguiente.
- ***Capítulo 4. Implementación.*** Se realiza una descripción en detalle de cómo se ha implementado la solución propuesta en el capítulo anterior. Se diferencia en su desarrollo entre la implementación realizada en el dispositivo físico que realiza el control de presencia y el servidor que ejecuta la página web del programa RoboRescue.
- ***Capítulo 5. Resultados.*** Se analiza el grado de consecución de los objetivos de este proyecto según la implementación propuesta. Se finaliza con el análisis de resultados durante la implementación en el laboratorio de una solución alternativa, propuesta para evitar las limitaciones existentes debidas a la configuración de las redes de la UMA.
- ***Capítulo 6. Conclusiones.*** Este capítulo concluye el cuerpo de la memoria, reflexionando sobre el trabajo realizado y el alcance del proyecto. Finalmente se aportan una serie de valoraciones acerca del futuro del mismo.
- ***Apéndice A. Configuración y primeros pasos en Home Assistant.*** A lo largo de este apéndice se detalla con fin informativo el proceso de instalación y configuración del entorno de Home Assistant en el dispositivo RaspBerry Pi 3B.
- ***Apéndice B. Configuración del Servidor Local.*** Se anexiona debido a su extensión el proceso de recreación de las condiciones del servidor que ejecuta la página web en la máquina local.
- ***Apéndice C. Configuración de Solución basada en Node-RED y Telegram.*** Desarrollo de la solución alternativa propuesta durante la implementación en el laboratorio.

# Capítulo 2

# Estudio incial del ecosistema hardware y software

## 2.1 Introducción

Este capítulo tiene como objetivo realizar una breve introducción a las características de los componentes software y hardware empleados en la realización del proyecto. Se destacarán de dichos componentes sus características principales así como su relevancia en el contexto del trabajo realizado. El propósito principal es contextualizar el contenido que se desarrolla en los capítulos subsiguientes, permitiendo obtener una comprensión más extensa de los elementos fundamentales que conforman el proyecto.

## 2.2 Hardware

El diseño del controlador de presencia en su totalidad es un sistema distribuido. Esto implica que existen diferentes nodos de procesamiento independientes, funcionando de manera conjunta para lograr un objetivo común. Estos nodos son el servidor sobre el que se ejecuta la página web de RoboRescue UMA y el equipo físico que realiza el control de presencia.

El servidor que ejecuta la página web y contiene las bases de datos se encuentra alojado en los servidores de la Universidad de Málaga. Su dirección web es pública y es accesible desde [3]. Sin embargo, y pese a necesitar una implementación de servidor local tal como se detalla en el Apéndice B, se omite esta descripción de esta Sección, debido a que la instalación se realizará de forma local en una máquina virtual del equipo personal.

En cuanto al terminal físico que gestiona el control de presencia, se opta por emplear una RaspBerry Pi3B. Ver figura 2.1.

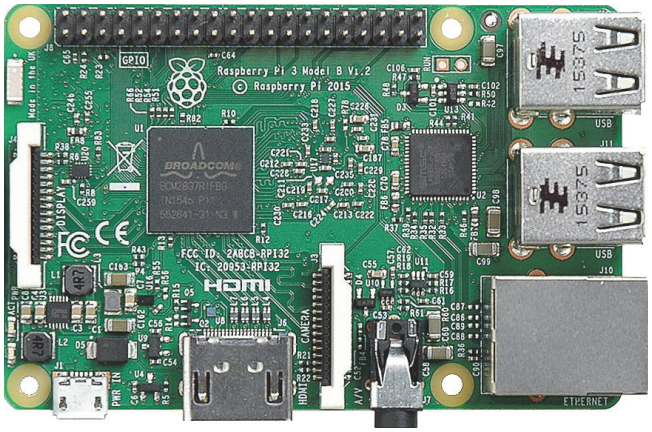



**Figura 2.1:** Vista de RaspBerry Pi 3B

Las RaspBerry Pi son una gama de ordenadores de bajo coste ideales para proyectos de domótica y automatización a pequeña escala[4]. Estos ordenadores monoplaca cuentan con una potencia de cómputo algo limitada, pero pueden ejecutar sin mayor problema sistemas operativos, siempre que cumplan con las restricciones que este hardware impone [5]. Entre estos sistemas operativos podemos encontrar mayoritariamente sistemas basados en Linux, como Raspbian, Kali Linux, Arch Linux ARM y Home Assistant OS [6]. Esta relación de compatibilidad es clave para el resto del desarrollo del proyecto, como se verá más adelante.

Las características que hacen que este miniPC sea ideal para nuestra aplicación son las siguientes:

**Disponibilidad:** Pese a que estos equipos suelen ser muy económicos (históricamente conocidos como el ordenador de 35 dólares [7]), la alta demanda [8] y la escasez mundial de semiconductores, han derivado en un aumento considerable de precio en los distribuidores oficiales. Siendo estas las condiciones de partida, en caso de no poseer uno de estos dispositivos, el empleo de un mini PC [9], en el mismo rango de precio actual que una Raspberry Pi 3B (100-150€) sería preferente. No obstante, en el laboratorio hay disponibilidad de sendos de estos dispositivos, por lo que el problema del desabastecimiento queda solventado.

**Bajo consumo:** El dispositivo empleado tiene unos picos de consumo de 2.5W [10], lo que lo hace increíblemente eficiente al realizar la comparativa con ordenadores convencionales. Pueden observarse las tasas de consumo en diferentes condiciones en 2.2.

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

*Pi Power Usage table adding Zero W (at 5.19V)*

**Figura 2.2:** Tabla comparativa del consumo de diferentes dispositivos de la familia RaspBerry [10]

**Conectividad:** La Raspberry Pi 3B cuenta con un puerto Ethernet, así como con Wi-Fi y Bluetooth on-board (directamente integrado). Se hace un énfasis especial en la compatibilidad directa con Wi-Fi y Bluetooth ya que, como se explora más adelante (Sección 3.3), serán factores determinantes a la hora de diseñar una solución.

**Posibilidad de ejecutar sistemas operativos:** A priori, la integración que se desea realizar como fin último de este proyecto es implementable en amplia variedad de equipos, como las placas ESP32, ESP8266 [11][12] que verifican las características requeridas. Más si cabe, mejorando algunas de ellas como el consumo y precio. En esta situación, hay un factor determinante que decanta la balanza de elección del hardware hacia la Raspberry Pi 3B: se puede ejecutar un sistema operativo con interfaz de usuario gráfica.

Se pretende que, una finalizada la implementación, el procedimiento de configuración sea lo más amigable posible para la persona encargada de gestionar el sistema. Esto cobra vital importancia durante las renovaciones de plantilla, en las que las personas que forman parte del equipo varían considerablemente. Reconfigurar el sistema de forma manual en entornos de bajo nivel de abstracción, como puede ser la reprogramación de placas tipo ESP, puede convertirse en algo tedioso y por tanto poco deseable.

## 2.3 Software

Se detallan los componentes software tanto del equipo local en el que se realizará el control de presencia como del servidor en el que se ejecuta la página web.

### 2.3.1 Software en RaspBerry Pi. Home Assistant

En el equipo Raspberry PI 3B se instalará Home Assistant OS. Home assistant será el eje central del proyecto, mediante el que se generarán todas las funcionalidades que conciernen a la detección de personas por diferentes vías.

#### Definición de Home Assistant

Home Assistant es un software libre de control domótico de que ejecuta la licencia Apache. Está basado en Python y hospedado en GitHub, [13] lo que lo hace software comunitario, sirviéndose de lo cual recibe numerosas contribuciones por parte de desarrolladores independientes.

Este software actúa como controlador central permitiendo a raíz de su función más básica, la automatización, interconectar e interoperar dispositivos IOT de forma local y privada, independientemente del fabricante que suministre dichos dispositivos. A diferencia de un *hub*, Home Assistant no es un controlador físico, sino un conjunto de elementos software sin plataforma definida que en su ejecución conforman un servidor local que permite el control y automatización de dispositivos inteligentes en el hogar u otro ámbito [1].

#### Bases de Home Assistant

**Versiones:** Home Assistant se presenta en varios formatos, dependiendo de las necesidades del usuario. Se introducen a continuación las cuatro versiones (ver Figura 2.3 )disponibles: [14].

1. Home Assistant Core: es el núcleo del software de Home Assistant. Proporciona la funcionalidad básica de Home Assistant en cuanto a labores de automatización e interacción. La instalación se realiza directamente y de forma obligatoria sobre un sistema operativo, como podrían ser Windows o Linux. La interacción con Home Assistant en este formato se realiza principalmente vía ventana de comandos, ya que no cuenta con la interfaz de supervisión que ofrecen otras versiones más extensas.

2. Home Assistant Container: Es idéntico en cuanto a prestaciones a Home Assistant core, pero se presenta en formato de imagen para Docker, por lo que debe ser instalado en un contenedor Docker.

3. Home Assistant Supervised: Cuenta con todas las funcionalidades de Home Assistant, incluyendo el supervisor, es decir, la interfaz de usuario desde la que controlar y supervisar el sistema. Requiere ser instalado en un sistema operativo Linux.

4. Home Assistant OS: Cuenta con todas las funcionalidades de Home Assistant Supervised pero es un Sistema Operativo autónomo basado en Linux, que requiere de un soporte físico (un ordenador). Está altamente optimizado para ser ejecutado en ordenadores tipo Mini-PC y RaspBerry Pi.

| | OS | Container | Core | Supervised |
|---|---|---|---|---|
| Automations | ✓ | ✓ | ✓ | ✓ |
| Dashboards | ✓ | ✓ | ✓ | ✓ |
| Integrations | ✓ | ✓ | ✓ | ✓ |
| Blueprints | ✓ | ✓ | ✓ | ✓ |
| Uses container | ✓ | ✓ | ✗ | ✓ |
| Supervisor | ✓ | ✗ | ✗ | ✓ |
| Add-ons | ✓ | ✗ | ✗ | ✓ |
| Backups | ✓ | ✓[1] | ✓[1] | ✓ |
| Managed Restore | ✓ | ✗[2] | ✗[2] | ✓ |
| Managed OS | ✓ | ✗ | ✗ | ✗ |

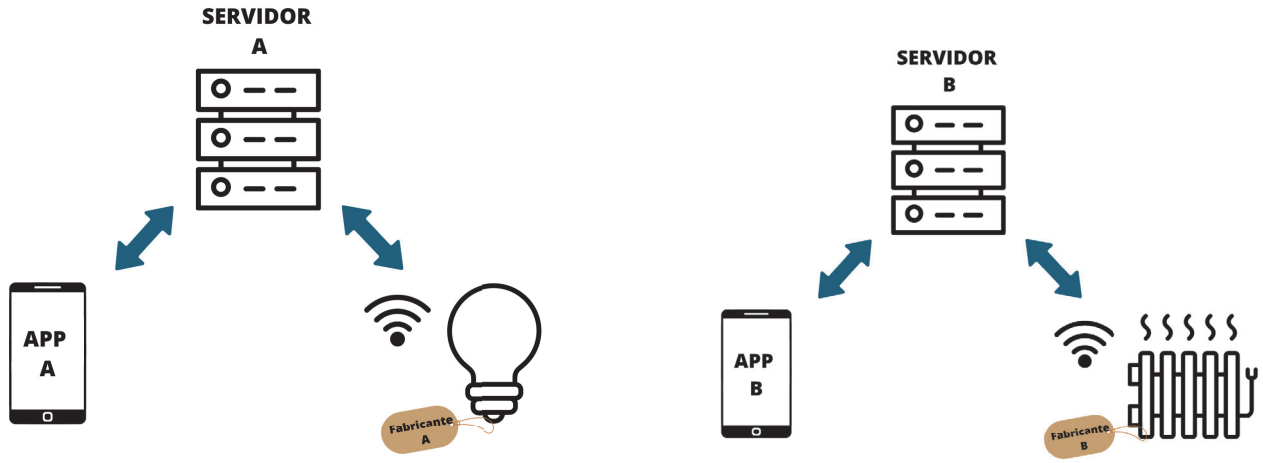
**Figura 2.3:** Versiones con soporte oficial de Home Assistant junto con sus características. (Recuperada desde: [15])

El Hardware Raspberry Pi 3B cumple los requisitos exigidos por los desarrolladores para instalar la versión de Home Assistant OS [16].

**Funcionamiento:** La función principal de Home Assistant es la de actuar como servidor intermediario, que centraliza en un mismo terminal la gestión e interacción con elementos y aplicaciones IOT. Esta gestión se realiza en una red local, en la que el usuario es propietario tanto del hardware que ejecuta el sistema como de todos los datos que se almacenan en él. Este último factor tiene importantes implicaciones, como que la información de actividad de los dispositivos inteligentes, credenciales y otros tipos de datos privados no se exponen en internet, sino que sólo son accesibles de forma local (a menos que se habilite de forma explícita lo contrario, como se expone en la Sección 3.5).

Por norma general, cada dispositivo IOT tiene una companion-app asociada, mediante la cual pueden ser controlados y supervisados. Estos dispositivos se encuentran habitualmente conectados a la nube de cada fabricante. Tener los dispositivos IOT conectados a una nube ajena puede acabar comprometiendo la seguridad, revelando información en algunos casos sensible [17] , y por otro lado requiriendo necesariamente una aplicación única por cada fabricante. Esto

limita las posibilidades de creación de un ecosistema personalizado que funcione tal como el usuario desea, viéndose este limitado a depender de una gama cerrada de dispositivos, que son únicamente controlables según las opciones predeterminadas a través de cada aplicación (ver Figura 2.4).




**(a)** Control de dispositivo de fabricante A **(b)** Control de dispositivo de fabricante B

**Figura 2.4:** Interacción desde smartphone con dispositivos IOT de fabricantes diferentes

Cuando se centralizan los servicios domóticos en Home Assistant, es este sistema el encargado de almacenar y gestionar los datos de los dispositivos, así como las peticiones, comandos y demás acciones que se realizan sobre ellos (comportamiento ilustrado gráficamente en Figura 2.5). Cuenta con una amplia compatibilidad con estándares y protocolos de comunicación, por lo que Home Assistant puede coordinar de forma paralela dispositivos de diferentes fabricantes simultáneamente. Asimismo, tanto de forma nativa como derivada de la labor de la comunidad de desarrolladores independientes, en este sistema se pueden instalar más de 2400 integraciones [18], que permiten la compatibilidad con la mayoría de dispositivos inteligentes que se pueden encontrar en el mercado. Sumado a la mencionada compatibilidad, algunas de estas integraciones servirán para resolver el problema de la detección e identificación de dispositivos y, por tanto, control de presencia; objetivo último del proyecto.

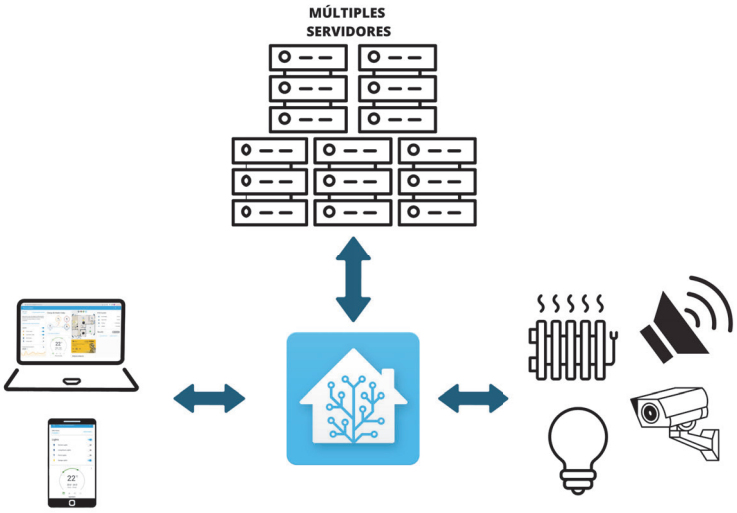



**Figura 2.5:** Centralización en Home Assistant del control domótico de varios dispositivos

En conclusión, Home Assistant no sólo permite la ejecución del cometido propuesto (detección e identificación de dispositivos por cercanía) de forma nativa como se detalla más adelante, sino que es capaz de integrar otros dispositivos inteligentes, tales como las placas tipo ESP, por lo que es modular y ampliable. Esto implica que si bien Home Assistant per se ofrece una plataforma que brinda varias alternativas para resolver la tarea impuesta, es además compatible y puede coordinar la gran mayoría de soluciones basadas en otras plataformas.

Finalmente, una vez implementado, el sistema puede servir como punto de partida para la modernización y automatización de muchas de las tareas del laboratorio, haciendo de este un entorno más interactivo en el que poder desarrollar otras ideas a la par que continúa el programa vigente (RoboRescue).

### 2.3.2 Software en el Servidor

Uno de los objetivos principales del presente proyecto es la posibilidad de consultar en tiempo real qué miembros se encuentran en el laboratorio en la página web de RoboRescue[3]. La página web basada en WordPress se ejecuta en una máquina virtual en los servidores de la UMA. Por prudencia, las características del servidor deberán ser recreadas de forma local de modo que las pruebas pertinentes no generen riesgo de corromper la base de datos de la que depende el sitio web del programa.

Siendo este el caso, el software cuya mención es más relevante se lista a continuación, aunque cabe mencionar que algunos de los componentes listados no se encuentran de forma natural en el servidor de la web. Su instalación y configuración se detallarán en el Apéndice B.

#### CentOS Linux

CentOS Linux (Figura 2.6) es un sistema operativo de código abierto. Es una redistribución creada por la comunidad basada en RHEL (Red Hat), en la que se han eliminado todas las

referencias al branding de esta compañía. CentOS comparte código base con RHEL, ya que esta compañía liberó gran parte del código fuente de su sistema operativo. Por sus características, ambos sistemas operativos están optimizados para servidores [19][20][21].


**Figura 2.6:** Logotipo CentOS, [22]

Se utilizará CentOS en su versión 7-2009, ya que es el sistema que ejecuta la máquina virtual sobre la que se ejecuta la página web del programa RoboRescue.

### Apache

Apache (Figura 2.7) es un servidor web HTTP de código abierto, gratuito, que permite a los usuarios alojar múltiples sitios web, servir contenido estático y dinámico en forma de páginas y aplicaciones web, entre otras[23]. Es el servidor que hospeda el dominio web de RoboRescue. Fue desarrollado por la Apache Software Foundation y es compatible con una amplia gama de sistemas operativos, incluyendo CentOS 7.


**Figura 2.7:** Logotipo Apache, [24]

Apache es modular, multi-plataforma, extensible y muy popular, lo que facilita la obtención de ayuda y soporte en caso de ser necesario.

### MariaDB

MariaDB (Figura 2.8)es un sistema de gestión de bases de datos relacional [25] o RDBMS originado a partir del código fuente de una versión anterior del popular MySQL. Su versión MariaDB Community server es de código abierto y gratuita. [26]


**Figura 2.8:** Logotipo MariaDB, [27]

Este software gestor será el encargado de administrar las bases de datos necesarias para llevar a cabo el proyecto: por un lado, la página web basada en WordPress requiere contenida en una base de datos; por otro lado, se necesitará otra base de datos independiente para almacenar los históricos de tiempo de estancia en el laboratorio de cada uno de los miembros del equipo.

### PHP

PHP (Figura 2.9) es un lenguaje de programación de código abierto y en constante desarrollo enfocado a la programación web. Es utilizado principalmente para crear páginas web dinámicas, puesto que puede ser incrustado en un código HTML. Uno de los factore que hace sobresalir a PHP por encima de otros lenguajes en que su ejecución es del lado del servidor, esto es, las tareas computacionales que requiere para su compilación y ejecución se realizan en el servidor en lugar de en la máquina del cliente. Una vez compilado, se traduce a contenido HTML y el servidor puede mostrar la página al cliente, estando la página cargada antes de que el cliente lo solicite. [28][29]


**Figura 2.9:** Logotipo PHP [30]

PHP es fundamental para este proyecto, ya que es el lenguaje en el que se basa el código de Wordpress.

### phpMyAdmin

Esta herramienta software (Figura 2.10) escrita en PHP (de ahí su nombre) proporciona una interfaz web para administrar los gestores de bases de datos, como MySQL o MariaDB, como es el caso[31].[28][29]


**Figura 2.10:** Logotipo phpMyAdmin [30]

Desde la amigable interfaz gráfica de phpMyAdmin se permite realizar amplia variedad de operaciones sobre las bases de datos, incluyendo la mayoría de las características de MySQL, así como la capacidad de importar y exportar datos en múltiples formatos. Esta herramienta facilitará en gran medida la administración de las bases de datos con las que se trabajan en este proyecto, ya que prescindir de ella implica la necesidad de trabajar con ellas en la ventana de comandos del sistema operativo CentOS 7, lo cual es mucho menos ágil y eficiente.

### WordPress

Wordpress (Figura 2.11) es un sistema de gestión de contenidos o CMS de código abierto programado en PHP. Permite crear y gestionar sitios web de diversos tipos sin la necesidad directa de programar a bajo nivel.[32]


**Figura 2.11:** Logotipo WordPress [33]

La herramienta Wordpress funciona en sistemas que ejecuten MySQL (en este caso MariaDB) y Apache. Su funcionamiento se puede clasificar en tres niveles: núcleo, template y plugins. [34] El núcleo es el core, de la herramienta, que contiene el código, comandos e instrucciones que hacen que funcione. Los templates o plantillas son capas de personalización que modifican la apariencia a la hora de visualizar las páginas web. Finalmente, WordPress incorpora una serie de plugins o extensiones, gran parte de ellos gratuitos; estos plugins añaden gran variedad de herramientas al conjunto inicial que proporciona WordPress, ampliando su versatilidad y funcionalidad. La página web del laboratorio está realizada con esta herramienta, en su versión WordPress 5.0.4.

#### Node-RED

Este software (Figura 2.12) proporciona un entorno de programación gráfico ligero basado en Node.js accesible desde navegadores web. La programación en Node-RED es gráfica; se lleva a cabo mediante la interconexión de abstracciones de código denominadas Nodos. Un conjunto de nodos conectados puede formar un flujo, y a su vez se pueden interconectar varios flujos, dando lugar a aplicaciones funcionales. Los nodos en Node-RED pueden realizar un gran abanico de funciones, ya que sumado a que se pueden crear nodos que contengan funciones de JavaScript, la comunidad contribuye de forma continua con colecciones de nodos que desempeñan tareas muy variadas. [35]

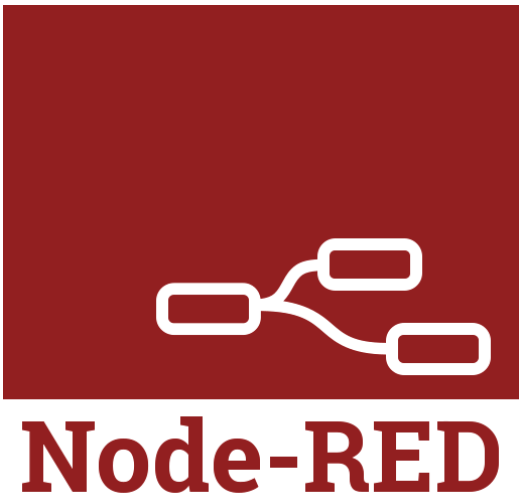



**Figura 2.12:** Logotipo Node-RED [36]

Node-RED es ideal para prototipar aplicaciones y realizar aplicaciones a pequeña escala muy rápido. Estas aplicaciones, debido a la programación gráfica, son prácticamente autoexplicativas; es sencillo entender el funcionamiento viendo cómo se interconectan los flujos. En cuanto a la utilidad dentro del proyecto, esta herramienta cuenta con nodos integrados que permiten convertirla en un cliente MQTT. Este aspecto es vital, ya que abre una vía de comunicación directa al servidor con otros dispositivos que utilicen este mismo protocolo.

### 2.3.3 Software Compartido

Para intercomunicar Home Assistant con el servidor en el que se ejecuta la página Web se necesita emplear software con el que ambos terminales sean compatibles. Se introducen dos componentes fundamentales: el protocolo MQTT y RESTful API.

#### Protocolo MQTT

MQTT es un protocolo de comunicación basado en publicación/suscripción de unas estructuras de datos denominadas *mensajes* entre cliente y servidor. Es ligero, abierto, simple y

está diseñado para ser fácilmente implementable. Estas características lo hacen ideal para ser aplicado en entornos limitados, como la comunicación en el contexto del Internet de las Cosas (IoT), donde se requiere un código pequeño y/o el ancho de banda de la red es valioso. Es un protocolo robusto y ligero que proporciona herramientas para la calidad de servicio y seguridad en las conexiones [37].

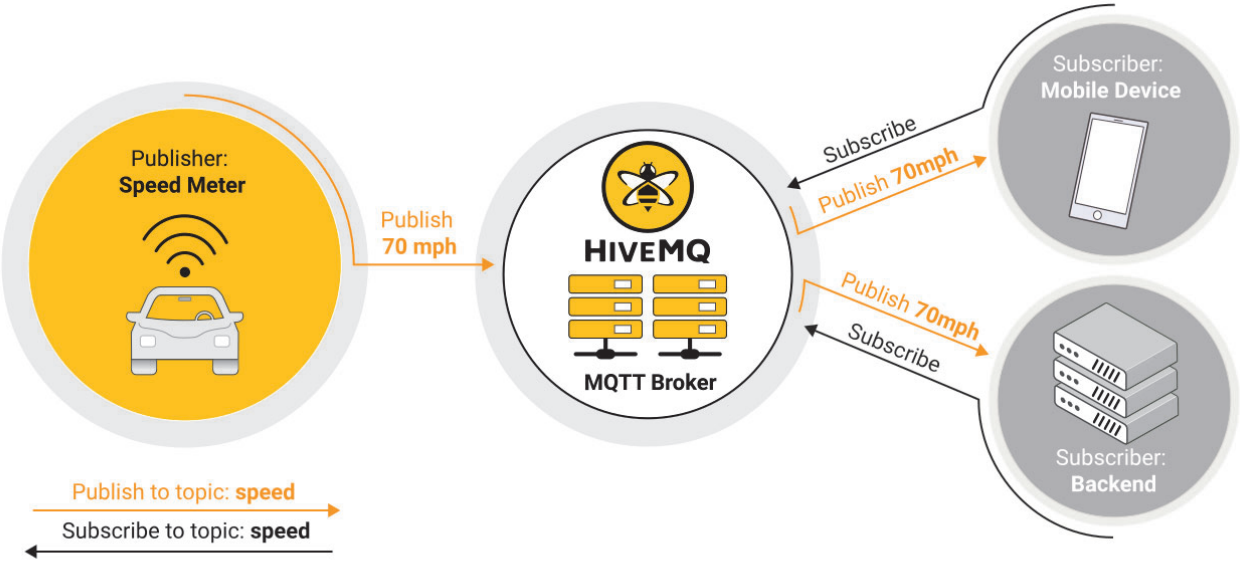



**Figura 2.13:** Esquema del intercambio de mensajes utilizando MQTT [38]

Los clientes publican y reciben mensajes mediante *topics*, que son campos con estructura jerárquica establecida mediante el uso de *slash (/)* . Un cliente puede publicar y estar suscrito a múltiples *topics*, delegando la coordinación de recepción y gestión de datos en el denominado Broker. Los clientes se interconectan entre sí indirectamente a través de este broker mediante las mencionadas reglas de publicación/suscripción. Este comportamiento aparece ilustrado en la Figura 2.13

Es importante hablar en este contexto de **Mosquitto Broker** (Figura 2.14), un broker ligero MQTT de código abierto ideal para equipos de baja potencia, compatible con Home Assistant [39][40].

**Figura 2.14:** Logotipo Eclipse Mosquitto. Broker MQTT de código abierto.

### RESTful API

Se denomina RESTful API a las API web que verifican la estructura REST (*REpresentional State Transfer*). Esta plataforma proporciona una interfaz que coordina los mecanismos para establecer conexión e intercambiar información mediante solicitudes HTTP entre cliente y servidor desacoplados (las aplicaciones ejecutadas en cliente y servidor deben ser completamente independientes). Toda la información asociada a las consultas se basa en transferencias sin estado, es decir, los mensajes intercambiados deben contener todas las instrucciones necesarias para llevar a cabo la instrucción deseada, sin necesidad de mantener una sesión abierta para el cliente del lado del servidor o viceversa. [41]

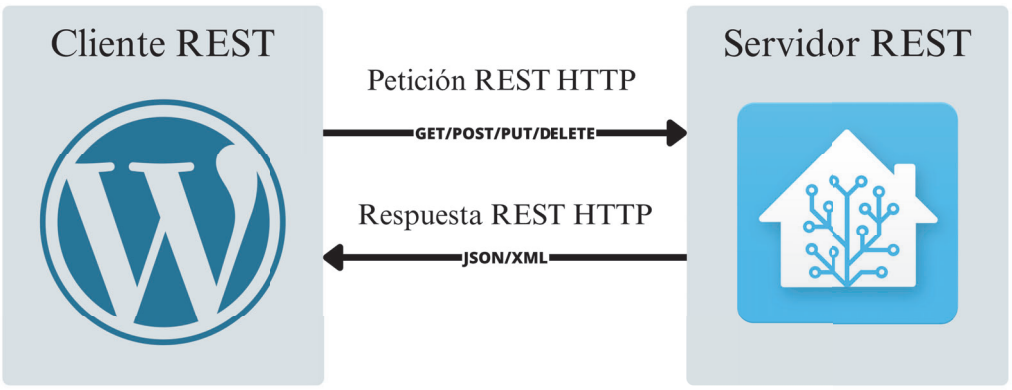



**Figura 2.15:** Esquema del intercambio de mensajes utilizando RESTful API

RESTful API está integrada de forma nativa en WordPress [42] y en Home Assistant [43], por lo que proporciona una herramienta muy útil para la interconexión de ambos terminales en el contexto del presente proyecto (ilustrado en Figura 2.15).

# Capítulo 3

# Diseño de la Arquitectura

## 3.1 Introducción

Una vez sentadas las bases del ecosistema con el que se va a trabajar, se plantean las diferentes ideas de diseño, valorando para cada alternativa la viabilidad dentro del contexto en el que se realiza este trabajo.

Para ello, se han propuesto dos características fundamentales que debe verificar el sistema. Por un lado, se evalúa la capacidad y fiabilidad que los diferentes métodos ofrecen a la hora de detectar e identificar a las personas. Por otro lado, se estudiarán las alternativas para establecer la conexión y consecuente flujo de datos entre Home Assistant y el servidor que aloja la página web de RoboRescue.

## 3.2 Control de Presencia en Home Assistant

Home Assistant dispone de una integración denominada ***device_tracker*** [44] utilizada para realizar seguimiento de los dispositivos que sean configurados para ser rastreables. Esta integración es compatible con varias plataformas que emplean diferentes tecnologías para realizar el mencionado seguimiento. Los dispositivos rastreados se almacenan en forma de entidades del tipo *device_tracker.{device_name}*, que pueden alternar entre los estados *"home"* y *"not_home"* según el dispositivo sea o no detectado en el rango que ofrezcan las diferentes configuraciones.

Por otro lado, la integración ***person*** [45] permite crear y configurar de forma intuitiva entidades a las que se le pueden asociar una o varias entidades del tipo *device_tracker.{device_name}*. Esto posibilita complementar varias técnicas de rastreo para la detección de personas. Ejemplificando lo anterior, una misma persona *person.luis* puede tener asociada una entidad *device_tracker.movil_luis* y otra entidad *device_tracker.pulsera_luis*, actualizando el estado binario *"home"* o *"not_home"* según la actualización más reciente de las

entidades del tipo *device_tracker*. Esta distinción se aprecia gráficamente en la Figura 3.1.

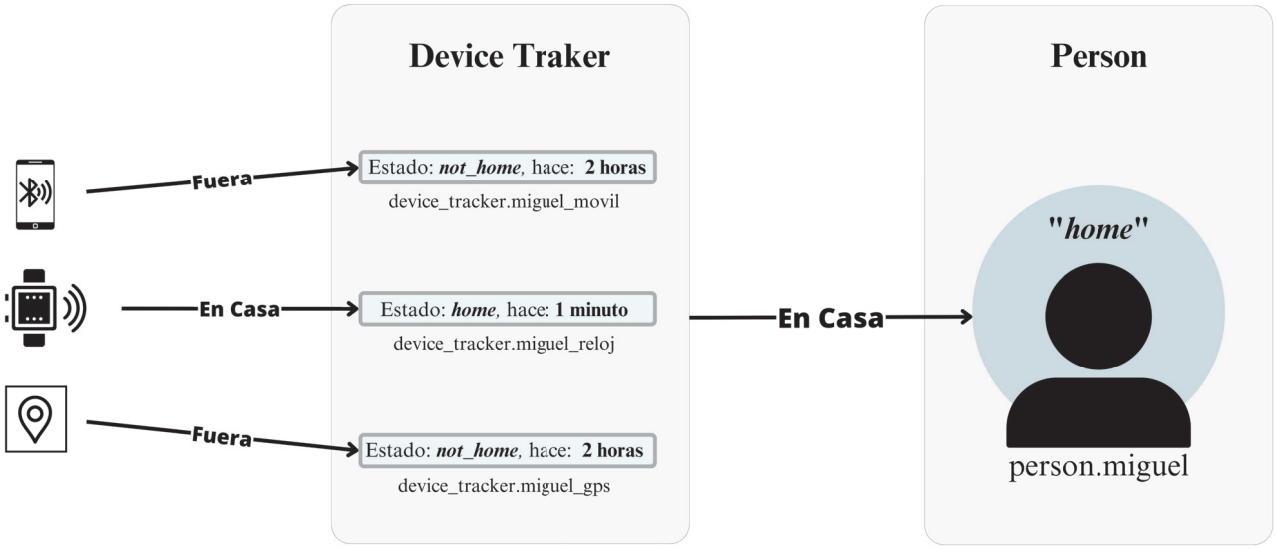



**Figura 3.1:** Distinción entre las entidades *device_tracker.device* y *person.nombre*

Para la aplicación que compete a este TFG, que la configuración de estas entidades personales sea sencilla y compatible con diferentes técnicas de detección es ideal, ya que posibilita crear una entidad *person.miembro* por cada uno de los miembros que forman el equipo, que se verá inalterada independientemente del método que se emplee para su detección.

Entre las posibilidades existentes puede hacer una una distinción básica: aquellas alternativas que requieren de hardware externo para funcionar, y aquellas que funcionan con componentes integrados de forma nativa en el equipo RaspBerry Pi 3B

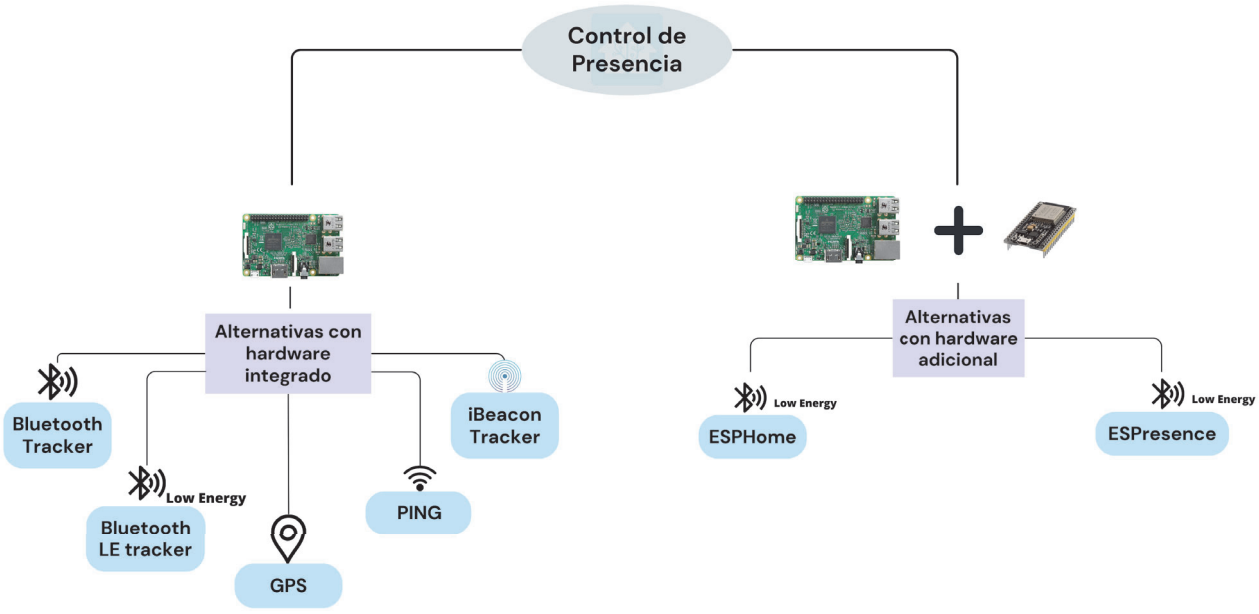



**Figura 3.2:** Esquema global de las alternativas exploradas

## 3.3 Control de Presencia con Hardware integrado

A continuación se evaluarán las diferentes alternativas que sólo requieren de RaspBerry Pi 3B para funcionar, ya que se sirven de las propias características de este sistema para realizar el control de presencia en Home Assistant.

### 3.3.1 Control de Presencia mediante Bluetooth Tracker

Home Assistant cuenta con una integración nativa de Bluetooth basada en BlueZ [46]. Desde la versión HassOS 8, esta integración es automáticamente detectada y configurable desde el asistente de instalación del sistema operativo en dispositivos RaspBerry Pi. Gracias a ella, se pueden detectar y configurar conexiones con dispositivos Bluetooth dentro del rango de alcance. Empleando esta compatibilidad, la integración encargada de realizar el seguimiento de los dispositivos (*device_tracker*) puede utilizar como plataforma la integración *bluetooth_tracker* [47] para la detección de dispositivos según su MAC asociada, la cual es individual y única.

La clase IOT de esta integración según la guía proporcionada por los desarrolladores [48] es *Local Polling*. El polling en este contexto consiste en el muestreo repetitivo del estado del dispositivo Bluetooth por parte de Home Assistant de todos los dispositivos que se encuentran en la lista de dispositivos conocidos *known_devices.yaml* (ver Figura 3.3). Al ser un muestreo local, la automatización que verifica el estado del dispositivo no depende de internet. Consecuentemente, para la correcta detección del estado del dispositivo, este debe estar siempre encendido. Esto sería un impedimento para conocer el estado de dispositivos que, con el objetivo de ahorrar energía, por ejemplo, desactivan periódicamente sus funcionalidades de forma temporal en ciclos de encendido-apagado. Esta problemática no es influyente para el caso de dispositivos móviles, siempre y cuando estos se mantengan encendidos durante el tiempo de estancia en el laboratorio, aunque se debe valorar en qué medida la comprobación constante puede llegar a saturar la banda 2.4GHz.

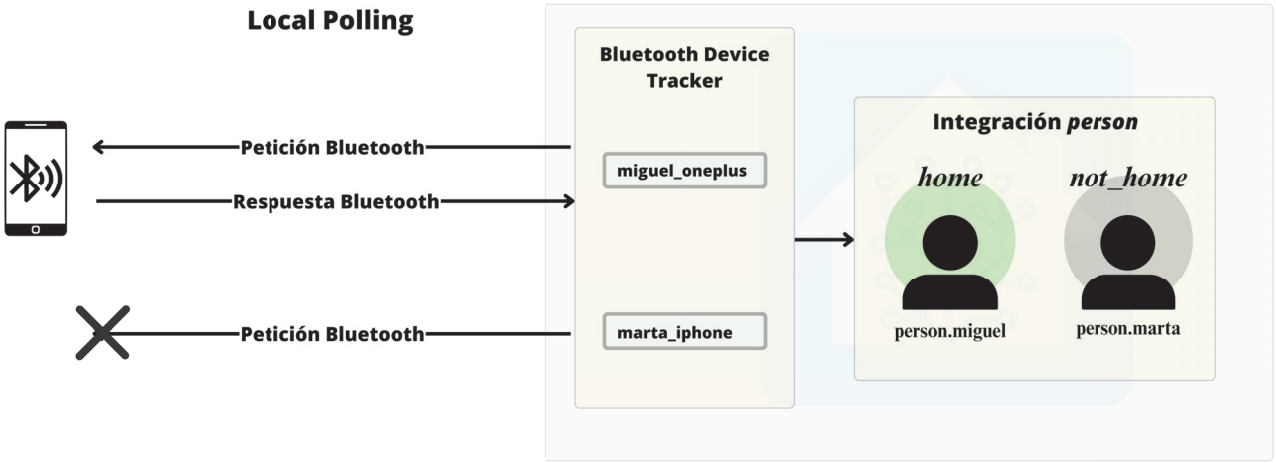



**Figura 3.3:** Detección de dispositivos mediante *Bluetooth Tracker*. Local Polling.

Para utilizar esta integración se debe especificar en el fichero raíz de configuración *confi-*

*guration.yaml* que se usará la integración *device_tracker* con esta plataforma como se muestra en el Código 3.1

**Código 3.1:** Líneas agregadas al fichero de configuración para habilitar la integración Bluetooth Tracker

```
1 device_tracker:
2   - platform: bluetooth_tracker
3     new_device_defaults:
4     track_new_devices: false      #Se debe activar esta opción para añadir nuevos dispositivos
5     interval_seconds: 10          #Tiempo entre escaneos al detectar dispositivos
6     consider_home: 60             #Tiempo para considerar un dispositivo offline
```

Los dispositivos que se identifiquen se almacenarán en el fichero *known_devices.yaml* como se muestra en el output Código 3.2.

**Código 3.2:** Dispositivos detectados añadidos automáticamente a *known_devices.yaml*

```
1  tv_samsung_q60ba_43_tv:
2    name: '[TV] Samsung Q60BA 43 TV'
3    mac: BT_A0:D0:5B:C7:21:AD
4    icon:
5    picture:
6    track: false      #Haciendo este valor True se realizará seguimiento del dispositivo
7
8  mi_portable_bt_speaker_16w:
9    name: Mi Portable BT Speaker 16W
10   mac: BT_08:EB:ED:AA:9A:DE
11   icon:
12   picture:
13   track: false      #Haciendo este valor True se realizará seguimiento del dispositivo
```

Es sencillo asociar a una persona alguno de los dispositivos almacenados (Figura 3.4) en el fichero *known_devices.yaml*, ya que los nombres con los que se almacenan suelen evidenciar el dispositivo del que se trata, además de ser personalizables en caso de que se desee.

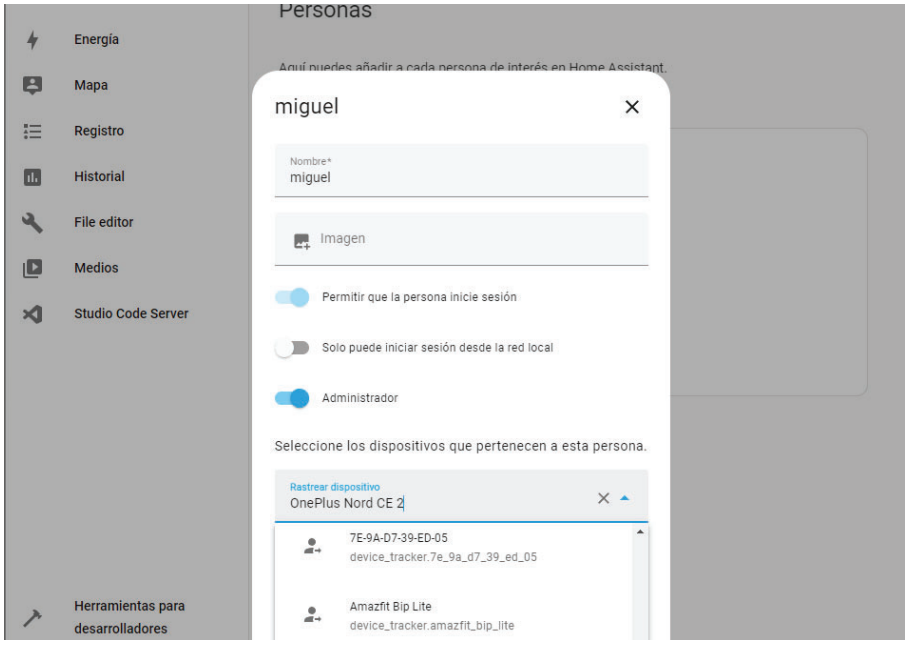



**Figura 3.4:** Vista del asistente de configuración para asociar dispositivos a las entidades tipo *person*

A continuación, la entidad *person.miguel* del ejemplo mostrado en la Figura , mostrará su estado como *"home"* o *"not_home"* según se detecte o no via bluetooth.

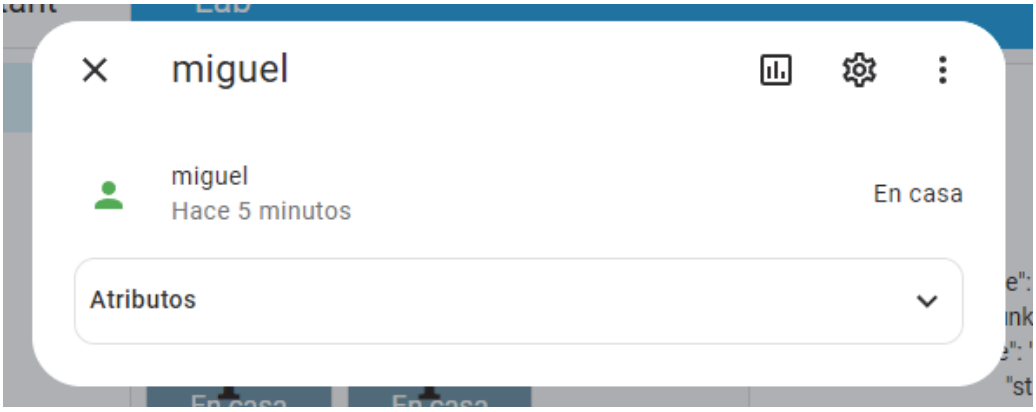



**Figura 3.5:** Vista del asistente de configuración para asociar dispositivos a las entidades tipo *person*

Tras exponer el funcionamiento de esta integración, se realiza un listado de los factores considerados ventajosos y negativos a la hora de considerarla como la solución final.

Ventajas de la integración Bluetooth Tracker:

1. Se encuentra disponible de forma nativa en el sistema operativo, por lo que la configuración adicional es mínima.
2. Los dispositivos se almacenan con nombres que facilitan su identificación para asociarlos a las entidades personales de cada usuario. La persona encargada de configurar, añadir o

eliminar dispositivos en el sistema tendrá mayor facilidad a la hora de identificarlos. Alternativamente, otras soluciones exigen conocer y trabajar con la dirección MAC asociada a cada uno, lo cual puede ralentizar el proceso para un número elevado de usuarios.

3. La práctica totalidad de los asistentes al laboratorio cuentan con Smartphone u otros dispositivos móviles que cuentan con tecnología Bluetooth.

Desventajas del método:

1. No es compatible con dispositivos que emplean tecnología de Bluetooth de baja energía (BLE), que habitualmente son mas prácticos debido que esta tecnología no requiere ser activada de forma explícita (siempre está en funcionamiento en segundo plano).

2. Requiere prestar especial atención a que Bluetooth esté activado en los dispositivos móviles para que puedan ser identificados. En caso contrario, será considerado como *"not_home"*. Esta característica cuando se encuentra activa supone un incremento en el consumo de batería, factor que es poco deseable

3. El fichero que contiene los dispositivos conocidos *known_devices.yaml* crece de forma incontrolada, registrando todos los dispositivos que encuentra a su alrededor, por lo que el rastreo de nuevos dispositivos se debe activar con prudencia, y sólo cuando se quieran incorporar nuevos dispositivos al sistema [49].

4. No es ampliable; es decir, el rango de detección queda limitado al rango en el que el adaptador Bluetooth de la RaspBerry Pi tenga influencia, en contraposición a otros métodos modulares que permiten incluso diferenciación por habitaciones o módulos.

Las primeras pruebas realizadas parecen confirmar el correcto funcionamiento de esta integración. A pesar de ello, el crecimiento descontrolado del fichero *known_devices.yaml* llega complicar la interacción con esta integración en cierta medida

### 3.3.2 Alternativas basadas en Bluetooth de Baja Energía (BLE)

Las siglas BLE (Bluetooth low energy) referencian a un subconjunto del estándar Bluetooth 4.0 orientado a aplicaciones sencillas de muy bajo consumo energético. Pese a contar con especificaciones similares en cuanto a rango de alcance (100m), a diferencia del Bluetooth convencional, que es originalmente diseñado para transmitir tramas de datos de tamaño considerable, BLE está orientado a la transmisión intermitente de paquetes de datos de tamaño más reducido. La tecnología BLE es tendencia en las líneas de desarrollo de aplicaciones basadas en IOT, ya que al basarse en conexiones intermitentes que se establecen con gran velocidad en lugar de conexiones continuas que requieren flujo ininterrumpido de energía, puede funcionar en dispositivos alimentados por pequeñas fuentes de energía, como pilas o baterías, durante años. El factor de tendencia se ha visto potenciado en gran medida por la acogida que BLE ha tenido por parte de las grandes tecnológicas, dueñas de plataformas como iOS y Android entre otras, que han proporcionado las condiciones necesarias para compatibilizar BLE con algunos de los dispositivos más extendidos del mercado. En las conexiones BLE, existe un dispositivo

central, encargado de iniciar y normalmente controlar las conexiones, y uno o varios dispositivos periféricos, que son aquellos que proporcionan los datos de interés. Los dispositivos periféricos están anunciando continuamente mediante el envío de paquetes su presencia y los servicios que ofrecen, y será el dispositivo central el encargado de gestionar dichos paquetes una vez establezca la conexión. El dispositivo central es en este caso la RaspBerry Pi ejecutando Home Assistant o placas tipo ESP [12][11] compatibles con este software [50][51][52].

Cuando se habla de control de presencia basado en BLE, los dispositivos periféricos que transmiten su identidad reciben el nombre de beacons o balizas que suelen emplear el protocolo iBeacon de Apple [53] . Las balizas son dispositivos hardware ligeros específicamente diseñados para esta labor. Suelen funcionar con pilas y tener una esperanza de vida media de varios años. Estas características pueden ser apreciadas en la Figura 3.6. Esta tecnología de beacon, si bien suele presentarse en formato de pequeño hardware, puede ser simulada por software en aquellos dispositivos que integran Bluetooth 4.0 en adelante, como relojes y pulseras inteligentes o teléfonos inteligentes, factor que es más interesante para la aplicación que se pretende implementar, ya que elimina la necesidad de tener que adquirir hardware nuevo.


**Figura 3.6:** Beacon BLE comercial (Recuperado desde [54]

Si bien utilizar esta tecnología para rastrear dispositivos móviles puede parecer la alternativa óptima, habrá que enfrentar un factor determinante a la hora de implementarla. Varias de las técnicas de identificación de dispositivos dependen su dirección MAC asociada. Que los dispositivos móviles emisores de BLE sean identificables y rastreables abre una brecha de en términos de seguridad importante debido a la expansión de la tecnología BLE, por lo que los fabricantes de Smartphones emplean métodos algorítmicos de aleatorización de las direcciones MAC de sus dispositivos [55].

Por otro lado, beacons tradicionales y otros dispositivos *wearables* como pulseras y relojes inteligentes convencionales (a excepción de los modelos de Apple y últimos modelos de Samsung SmartWatch) no presentan ningún inconveniente a la hora de ser rastreados, pero no están tan extendidos como podrían estarlo los Smartphones.

### Control de presencia mediante Bluetooth LE Tracker

Al igual que la integración nativa para el seguimiento con Bluetooth convencional, Home Assistant cuenta con la integración *bluetooth_le_tracker*. A diferencia de la intregración *bluetooth_tracker*, la clase IOT es *Local Push* (ver Figura 3.7). Esto implica que el dispositivo actualiza su estado informando directamente al servidor, sin necesidad de responder a una petición del mismo, aliviando la carga en la banda 2.4GHz. Home Assistant revisa entonces de entre el listado de dispositivos conocidos si el anuncio BLE corresponde con alguno de ellos. En caso contrario, si es detectado un número suficiente de veces, lo añade a la lista .

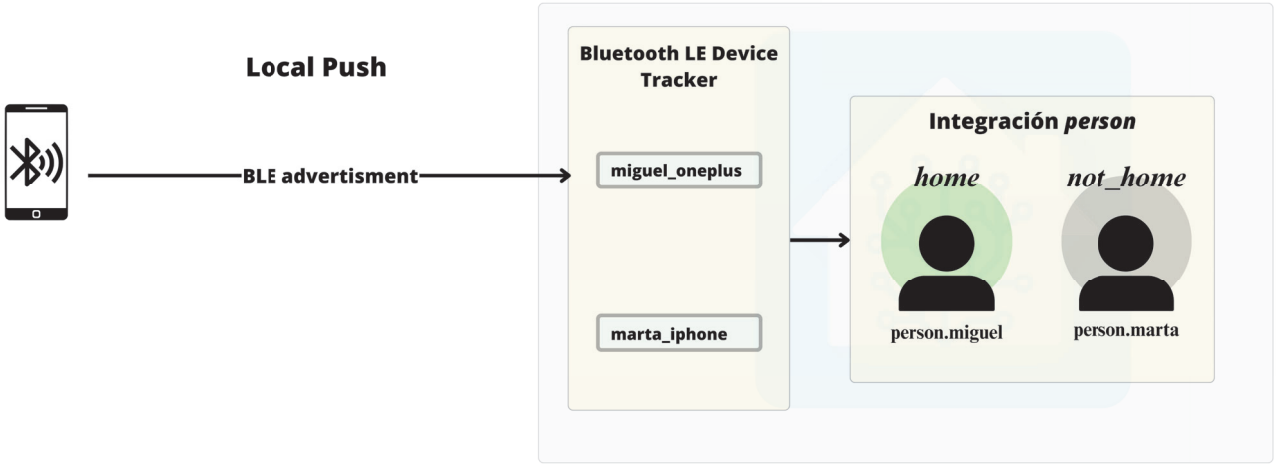



**Figura 3.7:** Detección de dispositivos mediante *Bluetooth LE Tracker*. Local Push.

El seguimiento se realiza para dispositivos con dirección MAC estática, por lo que no es funcional para Smartphones. La configuración es idéntica a la realizada para la integración *bluetooth_tracker*, (3.3) debiendo modificar ligeramente el código de configuración. Los dispositivos se almacenan en el mismo fichero *known_devices.yaml*.

**Código 3.3:** Líneas agregadas al fichero de configuración para habilitar la integración Bluetooth LE Tracker

```
1 device_tracker:
2   - platform: bluetooth_le_tracker
3     new_device_defaults:
4     track_new_devices: false      #Se debe activar esta opción para añadir nuevos dispositivos
5     interval_seconds: 10          #Tiempo entre escaneos al detectar dispositivos
6     consider_home: 60             #Tiempo para considerar un dispositivo offline
```

Al igual que para la solución anterior, se realiza un listado de ventajas e inconvenientes a la hora de considerar esta solución para el diseño final.

Ventajas de la integración Bluetooth LE Tracker:

1. Esta solución está basada en BLE, que se prevé será la nueva norma para aplicaciones de este estilo en una ventana temporal bastante reducida.

2. No requiere de ningún tipo de verificación activa por parte de los asistentes al laboratorio, ya que las balizas BLE se identifican continuamente sin necesidad de intervención.

3. El cambio de paradigma *Local Polling* a *Local Push* implica un alivio en la actividad de la banda 2.4GHz, minimizando el riesgo de interferencias.

4. Resto de ventajas mencionadas para la integración nativa de Bluetooth estándar.

Desventajas del método:

1. Comportamiento errático a largo plazo en las pruebas realizadas, lo que no permite asegurar una fiabilidad mínima para la tarea impuesta.

2. Incompatibilidad con SmartPhones y wearables de última generación al depender únicamente de sus direcciones MAC asociadas, las cuales suelen ser dinámicas.

3. Resto de desventajas mencionadas para la integración nativa de Bluetooth estándar.

Las primeras pruebas realizadas con esta integración fueron aparentemente exitosas. No obstante, al poco tiempo de funcionamiento, normalmente oscilante aleatoriamente entre los diez y cincuenta minutos, Home Assistant deja de registrar los cambios de estado de los dispositivos. Este problema es común, habiendo sido reportado numerosas veces por usuarios de la comunidad en los foros de soporte oficial [56][57].

Se trató de solventar el funcionamiento siguiendo la solución propuesta [58] por un usuario de la comunidad, que descubrió un error en el código fuente de la integración. La integración comprueba si un dispositivo está o no presente ejecutando un servicio que retorna un listado de los dispositivos almacenados en cache en la API de Bluetooth. El problema radica en que estos dispositivos permanecen en cache por un tiempo no definido, que oscila entre minutos y horas, y la integración considera como válidos, es decir, como presentes, todos los dispositivos que se encuentran en este listado. Al consultar sólo la existencia dispositivos almacenados en caché en lugar de la información, que de hecho contienen y se actualiza correctamente, acerca de la última vez que el dispositivo fue visto por el sistema, falla en actualizar el estado del mismo a *"not_home"*.

Conociendo esta problemática, se procede a realizar una actualización del código fuente según la corrección propuesta. Esta corrección se encuentra verificada por varios desarrolladores, así como por parte del equipo desarrollador de Home Assistant. Al no resultar efectiva la modificación se desiste en esta alternativa de forma temporal.

La solución propuesta finalmente se integra en el repositorio de GitHub [59] de Home Assistant, y se hace disponible una nueva actualización. Tras obtener la última versión del núcleo de Home Assistant (2023.4) [60] a fecha de 15 de Abril de 2023 se comprueba que la solución propuesta mejora discretamente el comportamiento errático de la integración, pero no desaparecen definitivamente estos problemas.

Reportes acerca de versiones posteriores, 2023.4.2 en adelante, [61] aseguran que el problema ha desaparecido definitivamente, por lo que debe ser realizada una verificación más exhaustiva.

**Control de presencia mediante iBeacon Tracker**

El protocolo iBeacon [53], es el más extendido en cuanto a balizas o beacons en sistemas de posicionamiento en interiores basados en Bluetooth de baja energía. El rastreador iBeacon Tracker es una integración para Home Assistant que permite la localización de dispositivos basados en esta tecnología, mayoritariamente beacons comerciales figura 3.6, aunque extiende su cobertura a un listado más amplio de equipos [62]. Al ser compatible con software que emplea hardware externo, como ESPHome, su funcionalidad y posibilidades se extienden. Esta integración registra todos los dispositivos que se encuentran en el rango BLE de Home Assistant de forma automática requiriendo posteriormente configuración adicional por cada uno de los dispositivos que se reconozcan como propios.

Ofrece mejoras significativas con respecto a la integración nativa BLE Tracker, como la desaparición de la problemática lista *known_devices.yaml* y la posibilidad de rastrear dispositivos con aleatorización de direcciones MAC, ya que el rastreo ahora se produce basandose en la dirección pública mostrada (UUID) en lugar de la dirección MAC estática. Por cada dispositivo se registra una combinación única de *UUID, major y minor* (ver Figura 3.8). Una vez se haya registrado dicha idéntica combinación proveniente desde 10 direcciones MAC diferentes se reconoce el dispositivo, ignorando su MAC (aleatoria), escuchando únicamente a la combinación de estos parámetros

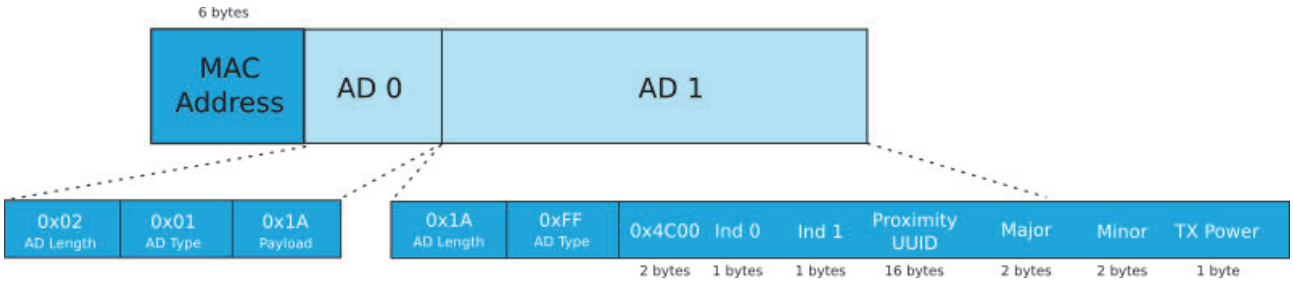



**Figura 3.8:** Estructura de paquete *advertisement* iBeacon. Imagen obtenida de [63]

Si bien es una alternativa que merece ser contemplada, se reportan varios problemas de funcionamiento en cuanto a la robustez de este sistema para hacer *tracking* de Smartphones. Durante las pruebas realizadas la integración se muestra incapaz de detectar algunos de los dispositivos.

### 3.3.3 Control de presencia mediante Ping Tracker

Además de las alternativas expuestas anteriormente existen opciones que no dependen del Bluetooth para detectar dispositivos. Herramientas como la integración Ping de Home Assistant basan su funcionamiento en características de la red WiFi local.

Para su funcionamiento, basta con agregar al fichero *configuration.yaml* la plataforma *ping* para posteriormente identificar cada dispositivo con el nombre deseado y su dirección IP en la red WiFi local, tal como se muestra en el Código 3.4.

**Código 3.4:** Líneas agregadas al fichero de configuración para habilitar el seguimiento basado en ICMP

```
1 device_tracker:
2   - platform: ping
3     hosts:
4       miguel_oneplus: 192.168.2.10
5       marta_iphone: 192.168.2.25
6
7       #Agregar tantas líneas como dispositivos a rastrear
```

El funcionamiento de esta integración es similar al de otras aplicaciones que utilizan este mismo comando de diagnóstico de redes. Ping hace eco de solicitudes ICMP (protocolo de mensajes de control de Internet, protocolo IP) a una dirección IP conocida, para posteriormente evaluar el estado, velocidad y calidad de la red. Para la detección de presencia, algunas plataformas como Android bloquean mediante firewall las solicitudes TCP y UDP, sin embargo, sí se encuentran respondiendo a ICMP, por lo que, ignorando el resto de datos de diagnóstico que ofrecen, es posible configurar un *device_tracker* empleando la plataforma *ping* que considere a los dispositivos como *"home"* en caso de que reciba cualquier respuesta y *"not_home"* al no recibirla.

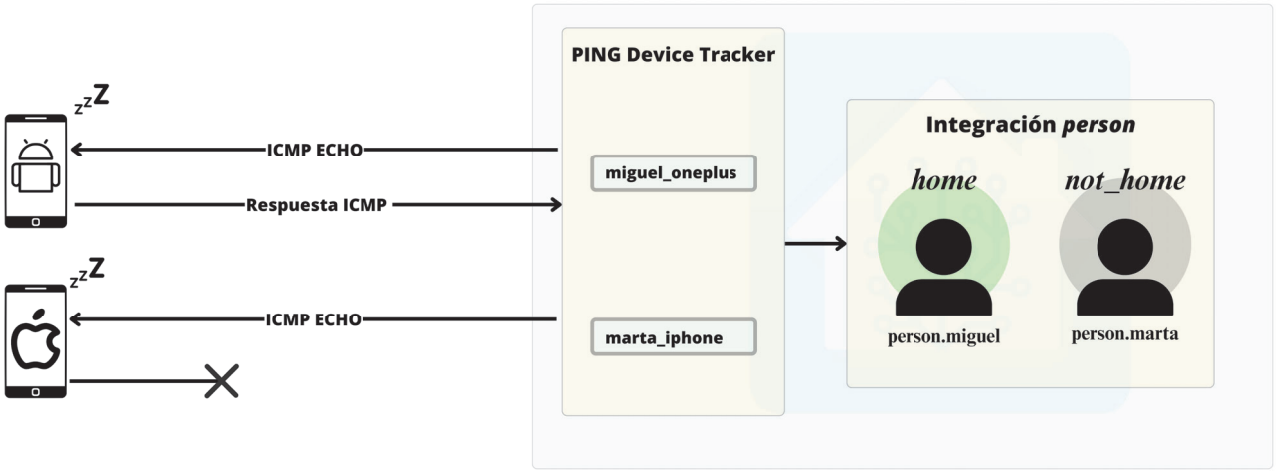



**Figura 3.9:** Comparativa entre el funcionamiento de la plataforma *ping* de *device_tracker* entre dispositivos Android e iOS mientras se encuentran suspendidos.

Esta integración no puede emplearse para dispositivos iOS, ya que mientras se encuentran en estado de *stand-by* no responden a solicitudes ICMP (ver Figura 3.9). Al encontrarse tan extendidos, siendo que numerosas personas del programa RoboRescue los utilizan, se descarta la viabilidad del empleo de esta integración para la labor propuesta.

### 3.3.4 Control de presencia mediante Geolocalización

Home Assistant cuenta con la opción de monitorizar en tiempo real la ubicación de los Smartphones que habiliten esta característica desde la aplicación que ofrecen. Si la ubicación estimada del dispositivo es suficientemente cercana a la ubicación predefinida durante la configuración de Home Assistant como *"home"*, se considerará que está en el laboratorio. Esta opción

requiere adquirir un servicio de pago que permite utilizar Home Assistant de forma remota denominado Home Assistant Cloud [64]. Este factor junto con la evidente invasión de privacidad que supone compartir y permitir visualizar la ubicación en tiempo real hacen descartar esta alternativa de forma permanente.

## 3.4 Control de Presencia con Hardware adicional

Se evalúan a continuación las posibles soluciones que requieren de hardware externo, es decir, que además del equipo RaspBerry Pi 3B, requieren de la instalación de hardware adicional. Las aplicaciones disponibles comunmente se aprovechan las capacidades y recursos de conectividad de las placas ESP (ESP8266/ESP32), enfocadas mayoritariamente a aplicaciones IOT. Estas placas además destacan por tener un precio muy económico, lo que las hace ideales para la tarea. Estas alternativas sirven para distribuir los recursos de la RaspBerry Pi, ampliando además sus características según la filosofía IOT, es decir, varios dispositivos especializados en tareas diferentes interconectados en una red común, en este caso una red local WiFi.

### 3.4.1 ESPHome

ESPHome es un sistema de control orientado a las placas ESP8266/ESP32 que se sirve de simples pero a la vez potentes ficheros YAML de configuración, permitiendo controlar dichas placas de forma remota desde Home Assistant [65]. En Home Assitant se presenta en forma de *add-on,* que provee de una herramienta que es capaz de interpretar archivos de configuración YAML y crear custom firmware que se instalará en las placas ESP. Esto permite hacer de Home Assistant un centro de gestión y control para las diferentes aplicaciones, sensores y actuadores que pueden ser integrados en placas ESP.

Para el seguimiento de dispositivos, ESPHome permite programar cada ESP 32 como un nodo de una red de dispositivos detectores de Bluetooth, permitiendo generar un entramado que permite la detección por habitaciones. Al solo contar con un espacio en el laboratorio, esta tarea se simplifica, necesitando solo una placa para realizar esta labor. La aplicación ESPHome cuenta con una serie de componentes específicos realizar seguimiento de beacons BLE [66] que ofrecen funcionalidades ampliadas en relación a la integración nativa, además de ser compatible con IBeacon Tracker.

Si bien un entramado de ESP32 haciendo de proxy Bluetooth para Home Assistant coordinado desde ESPHome amplía las posibilidades en términos de rango, precisión y funcionalidades extra con respecto a las integraciones nativas a Home Assistant, no ofrece un incentivo suficiente para la aplicación concreta en el contexto que se está estudiando, ya que presenta las mismas limitaciones que las integraciones nativas a pesar de requerir de hardware adicional.

### 3.4.2 ESPresence

ESPresence [67] es un firmware para las placas ESP32 que funciona en consonancia con la integración *mqtt_room* [68]. Al igual que ESPHome, está orientado a establecer una red Bluetooth capaz de diferenciar presencia por zonas. Sin embargo, este software presenta técnicas que lo hacen compatible con la detección de señales BLE generadas por smartphones, ya que está basado en la detección de ID en lugar de direcciones MAC.

En Android, se puede utilizar la companion-app [69] de Home Assistant para simular una baliza BLE[70].

La información que los dispositivos de Apple envían, normalmente está encriptada. Esta familia de direcciones se denomina RPA (*Resolvable Private Address*), que es resoluble mediante un hash denominado IRK (identity resolving key) [55]. El proceso de obtención del IRK así como su configuración suceden de forma automática en las versiones más recientes de este software[71].

Este software solventa las limitaciones que tenían los métodos basados en BLE anteriormente descritos, permitiendo rastrear tanto Smartphones, wearables o balizas iBeacon convencionales. Para ello, se puede emplear la integración MQTT Device Tracker, actualizando el valor de cada entidad asociada a cada dispositivo según el topic en el que se publican sus actualizaciones de estado. Pese a las ventajas que proporciona frente a otras soluciones estudiadas, el proceso de configuración[72] para cada usuario es lento y manual, lo que puede suponer un problema a medida que el número de usuarios crece.

## 3.5 Comunicación Home Assistant-Servidor

La capacidad de poder exportar datos obtenidos en durante el control de presencia es crucial para la ejecución del proyecto, en tanto en cuanto uno de los requisitos fundamentales es poder comprobar qué personas se encuentran en el laboratorio desde la página web hospedada en un servidor externo a Home Assistant. Por otro lado, se deberá poder exportar información suficiente para determinar qué cantidad de horas permanece cada miembro en el taller según cada mes consultado.

Se valoran dos tecnologías para establecer esta conexión: el protocolo MQTT y la RESTful API (ver Figura 3.10). Ambas tecnologías se encuentran integradas en Home Assistant. Este sistema operativo ofrece la posibilidad de instalar un add-on de **Mosquitto Broker**, integrando en la propia RaspBerry Pi el broker que coordinará el envío de mensajes MQTT entre los diferentes clientes.

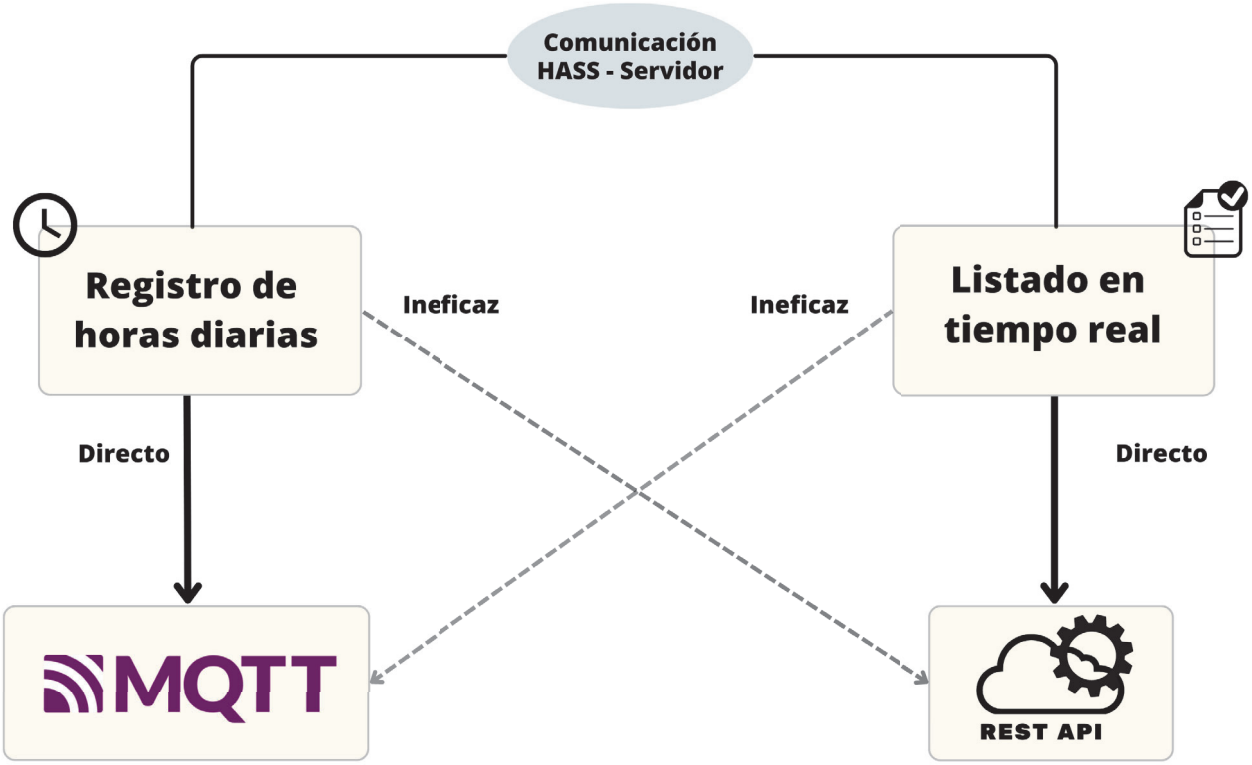



**Figura 3.10:** Esquema de interconexión HASS - Servidor. Alternativas estudiadas

## 3.6 Comunicación empleando el protocolo MQTT

Se evalúa la aptitud de las herramientas que utilizan el protocolo MQTT en la efectividad de llevar a cabo las dos tareas anteriormente descritas.

### 3.6.1 Enviar el tiempo de estancia en el laboratorio vía MQTT

Se propone una automatización que envía el tiempo que ha estado cada miembro en el laboratorio a una hora fija cada día (22:00, una vez nadie se encuentra en el laboratorio). Estos datos deben ser después almacenados en una base de datos local al servidor de la página Web.

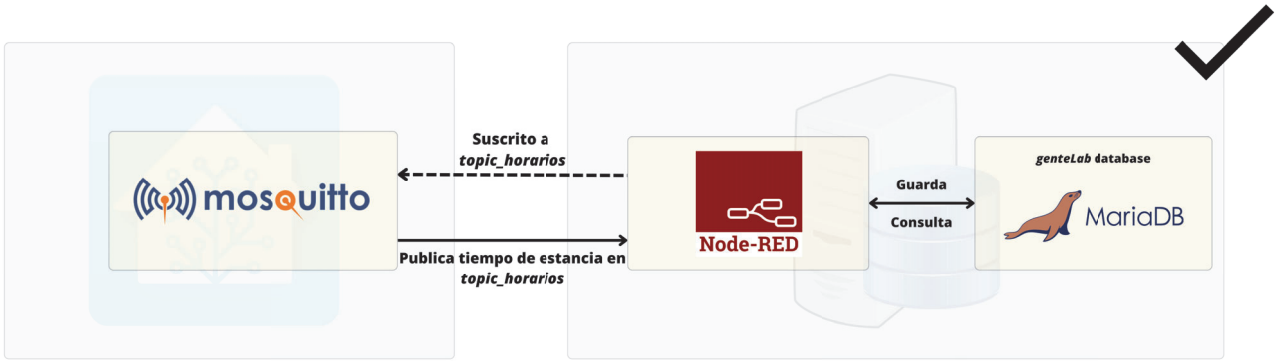



**Figura 3.11:** Esquema de interconexión HASS - Servidor. Cómputo de horas enviado vía MQTT.

Node-RED, el cliente MQTT instalado en el servidor tiene herramientas específicas para almacenar y consultar estos datos, generando un flujo visual intuitivo junto con una interfaz que permite consultar estos valores. Es una opción ideal para desempeñar esta tarea (ver Figura 3.11).

### 3.6.2 Enviar listado de asistentes en tiempo real vía MQTT

A la hora de enviar el listado de personas presentes en el laboratorio a Wordpress se debe considerar que pese a estar ejecutándose en el mismo servidor, es ajeno a MQTT, y que los plugins ofrecidos para esta labor ya no reciben soporte oficial. Existen algunas soluciones estudiadas: desde el cliente Node-RED se puede generar un archivo de texto local al servidor cada vez que se reciban los datos de asistencia. Posteriormente cuando se consulte, desde WordPress se cargará este fichero (formato *.txt* o *.csv*) y se filtrarán los resultados para mostrar quién se encuentra en el laboratorio.

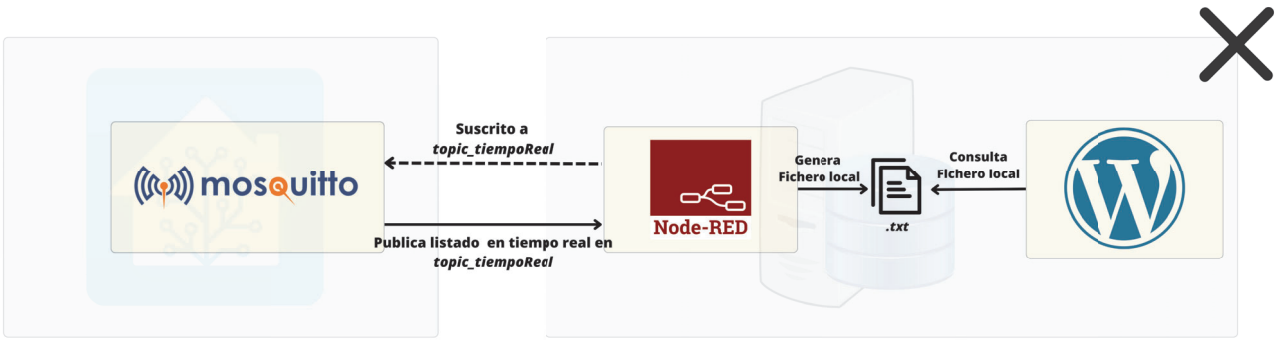



**Figura 3.12:** Esquema de interconexión HASS - Servidor. Cómputo de horas enviado vía MQTT.

Si bien se puede solucionar de esta forma, supone lo que se conoce como un *workaround* (del inglés, paliativo, Figura 3.12) al existir posibilidad de interconectar Wordpress y Home Assistant de forma directa como se explora a continuación.

## 3.7 Comunicación mediante RESTful API

Se valora en esta sección la idoneidad de emplear la API REST, que puede ser consumida tanto desde WordPress como desde Home Assistant lo que abre una vía de comunicación directa entre la página web y la RaspBerry instalada en el laboratorio.

### 3.7.1 Enviar el tiempo de estancia en el laboratorio vía RESTful API

La aplicación de conteo de horas debe ser independiente a la página web. Dado que los endpoints de esta API se ejecutan en los terminales de Home Assistant y Wordpress, aún siendo posible almacenar los datos deseados bases de datos que comparten servidor desde Wordpress, **no es coherente** con la filosofía que se pretende seguir. Idealmente se necesitará una base de datos independiente a la base de datos sobre la que se ejecuta Wordpress, ya que al tratarse de aplicaciones diferentes, es mejor praxis organizar sus respectivos contenidos en bases de datos diferentes. La propuesta aparece reflejada en la Figura 3.13

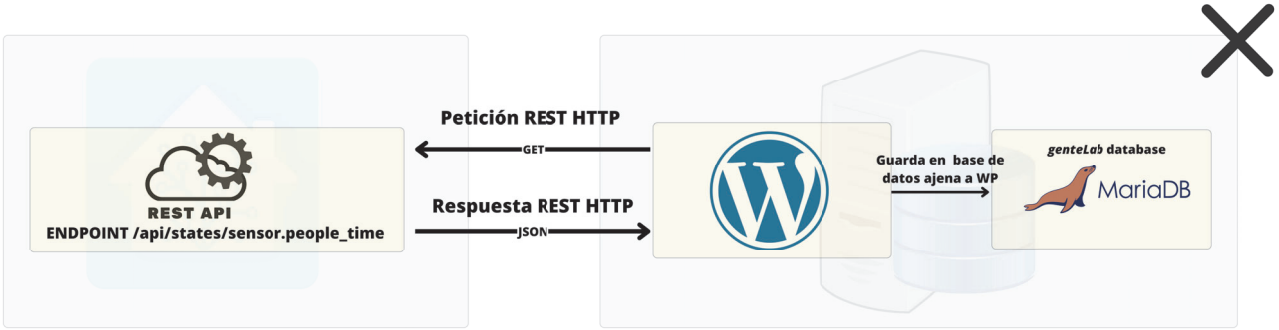



**Figura 3.13:** Esquema de interconexión HASS - Servidor. Cómputo de horas enviado vía RESTful API.

### 3.7.2 Enviar listado de asistentes en tiempo real vía RESTful API

La característica de mostrar el mencionado listado es exclusiva de la página web WordPress, es decir, la forma preferente de consulta es únicamente a través de la página web. Siendo el caso en que Wordpress integra de forma nativa compatibilidad con RESTful API, se puede establecer una conexión directa entre el servidor de detección de presencia y la página Web del laboratorio, comportamiento ilustrado en la Figura 3.14. Home Assistant genera endpoints personalizados por cada uno de los sensores configurados. Estos endpoints pueden ser consultados directamente desde Wordpress, obteniendo la información requerida sin necesidad de software intermediario, hecho que sí sucede con la propuesta que emplea el protocolo MQTT.

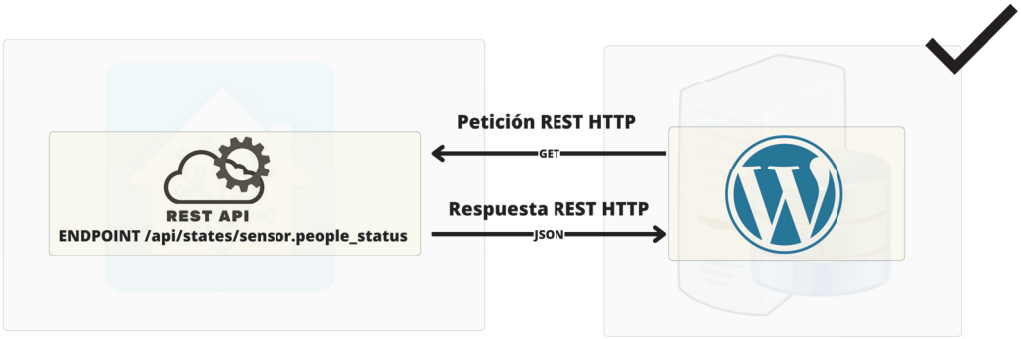



**Figura 3.14:** Esquema de interconexión HASS - Servidor. Listado de gente en tiempo real vía RESTful API.

Se debe crear un sensor cuyo estado sea el conjunto de miembros y su ubicación asociada (*"home", "not_home"*) para que se genere el endpoint que posteriormente se pretende consultar desde la página web.

## 3.8 Conclusiones y Diseño Elegido

Sintetizando la información recopilada y los resultados de extensivos periodos de prueba previos a la implementación final el diseño de la arquitectura propuesto es el que se expone a continuación.

Por un lado, se utilizará la integración nativa de Bluetooth Tracker para la detección e identificación de dispositivos, en este caso smartphones, debido a su sencillez, fiabilidad demostrada y facilidad de interacción con los miembros del equipo. Idealmente, una solución basada en varias integraciones funcionando conjuntamente sería ideal. Específicamente la solución basada ESPresence, que según lo observado y la amplia aceptación de la comunidad, en consonancia con la tendencia creciente del uso de Bluetooth de baja energía en múltiples dispositivos tecnológicos, sería una solución óptima en caso de querer complementar la integración de Bluetooth Tracker nativa. No obstante, la configuración es necesariamente más compleja para cada usuario, y dado que el número de usuarios será considerablemente alto, es importante optimizar la tarea de configuración para ser lo más ágil posible.

El listado de dispositivos detectados en el laboratorio se enviará a Wordpress vía RESTful API, aprovechando los endpoints que genera Home Assistant de forma automática para cada sensor [43] . Este listado será provisto bajo demanda de la página web, es decir, cada vez que se acceda se realizará una solicitud de actualización del listado. El comportamiento descrito puede observarse ilustrativamente en la Figura 3.15.

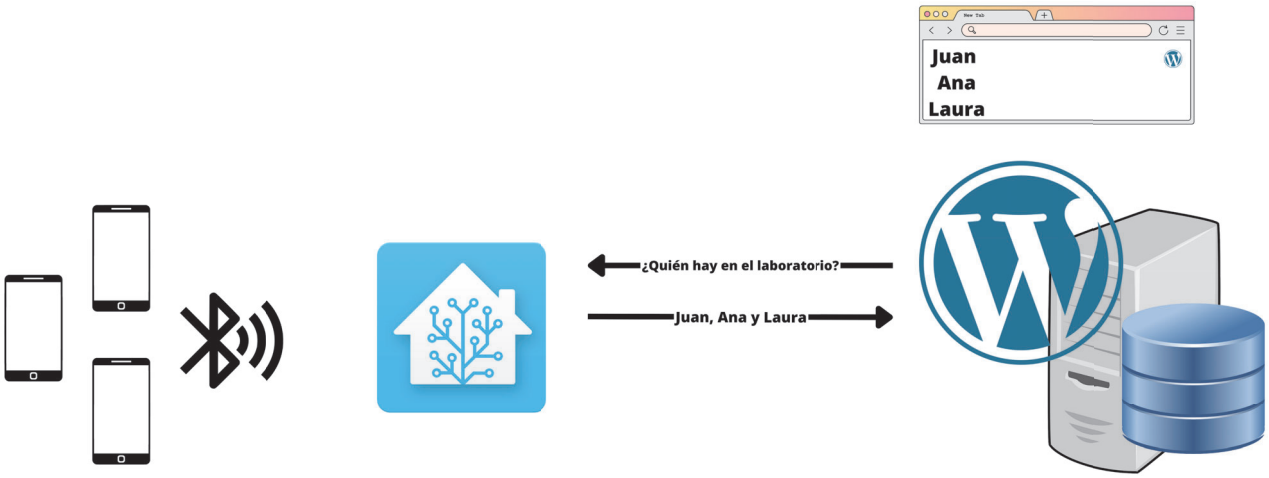



**Figura 3.15:** Vista ilustrativa del envío del listado de dispositivos detectados por Bluetooth en tiempo real vía RESTful API según solicitud

Por otro lado, en cuanto a la contabilización de horas mensuales de cada uno de los miembros, se empleará Home Assistant para realizar los cálculos necesarios para cada uno de los miembros del equipo. Posteriormente, una automatización programada para ser enviada a una hora determinada (22:00) todos los días, será enviada vía MQTT al cliente Node-RED instalado en el servidor. Node-RED almacenará estos datos en una base de datos ajena a la de WordPress creada en MariaDB y posteriormente podrá mostrarlos en forma de listado ordenado según nombre, mes y horas, fácilmente legible. Este comportamiento se ilustra gráficamente en la Figura 3.16.

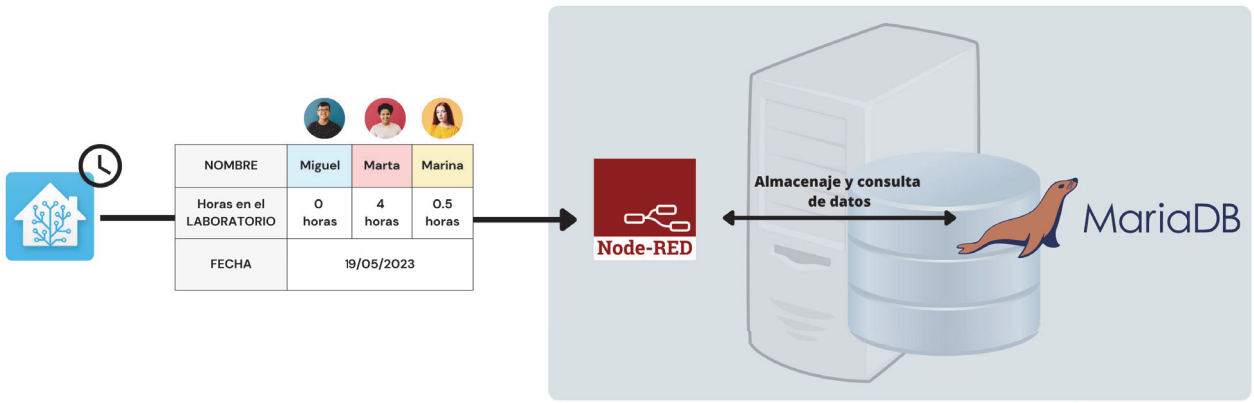



**Figura 3.16:** Vista ilustrativa del envío programado del tiempo de estancia vía MQTT

# Capítulo 4

# Implementación

## 4.1 Introducción

A lo largo de este capítulo se detallará cómo se ha implementado la solución propuesta en el Capítulo 3, distinguiendo entre las dos tareas principales: El control de presencia en Home Assistant **??** y el envío de los datos recopilados al servidor que hospeda la Web del proyecto. Este comportamiento aparece ilustrado gráficamente en la Figura 4.1.

En primer lugar se describe el proceso de implementación de las funcionalidades de Home Assistant. Será desde este sistema operativo donde se configuren los mecanismos para detectar mediante Bluetooth convencional qué dispositivos se encuentran en el laboratorio. Posteriormente, se generará un listado exportable bajo consulta vía RESTful API de todos los miembros registrados junto a su estado actual. Finalmente, se recopila el tiempo de permanencia en el laboratorio de cada uno de los miembros registrados, comprobando cuanto tiempo han permanecido sus entidades asociadas cada día en estado *"home"* . Para exportar estos datos se generará una automatización programada a una hora fija que los publicará vía MQTT al cliente Node-RED instalado en el servidor.

Finalmente, se detalla la configuración que ha sido necesario realizar del lado del servidor. Se detallará primeramente cómo se realiza la solicitud HTTP desde WordPress a Home Assistant para recibir el listado en tiempo real de los miembros que se encuentran en el laboratorio y mostrarlo en una página web. Adicionalmente, se profundiza en la implementación del flujo del cliente MQTT en Node-RED para la recepción del cómputo de horas diarias por miembros, permitiendo almacenarlo en una base de datos previamente configurada y consultar los totales mensuales siempre que se solicite.

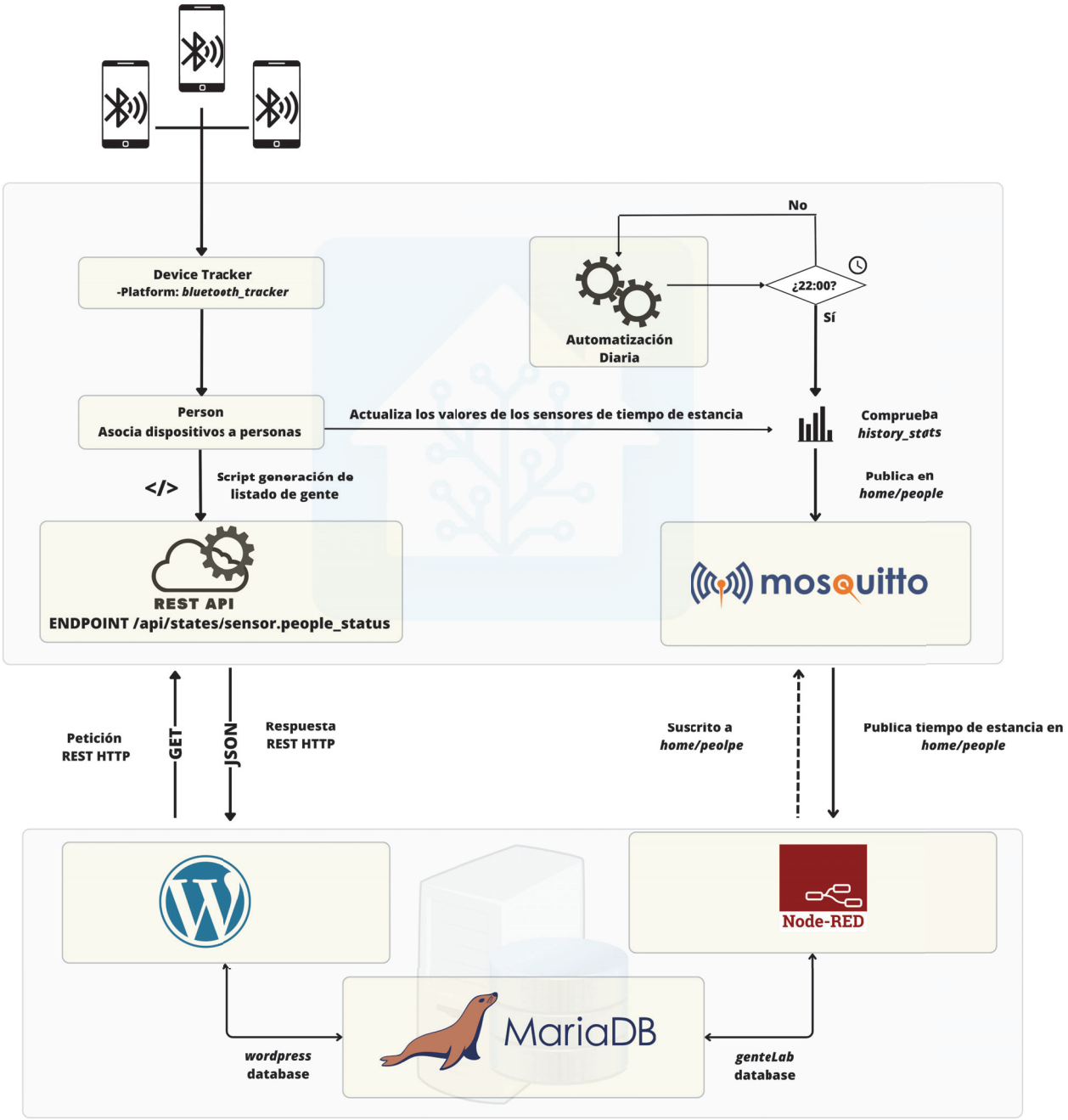



**Figura 4.1:** Esquema general de la implementación

## 4.2 Implementación en Home Assistant

Como paso previo al resto de la implementación se debe realizar la instalación del sistema operativo Home Assistant OS y el software necesario según las indicaciones del ApéndiceA. Posteriormente, se configura la integración Bluetooth Tracker como rastreador de dispositivos junto a las entidades personales asociadas a cada uno de ellos, tal como se muestra en las el

Código 3.1 y figuras 3.4 y 3.5.

Home Assistant permite crear grupos de entidades y hacer estos grupos visibles generando un endpoint de la RESTful API. Sin embargo, en la creación de grupos se deben añadir cada una de las entidades de forma manual. Para automatizar la tarea, se programa un sensor cuyo estado será un listado de todas las personas junto con su estado (*"home"/"not_home"*), iterando sobre todas las entidades miembros del dominio *person.{nombre_persona}*. Se añade al archivo *configuration.yaml* el código 4.1.

**Código 4.1:** Código para generar el cuyo estado es el listado de personas

```
sensor:
  - platform: template
    sensors:
        people_status:
            value_template: >
              {% set data = namespace(person_list = []) %}
                {% for state in states.person %}
                  {% set person = {'name': state.name, 'state': state.state}%}
                  {% set data.person_list = data.person_list + [person] %}
                {% endfor %}
                {{ data.person_list | tojson }}
```

Posteriormente, tal como se muestra en la Figura 4.2, se verifica que se genera el sensor cuyo estado es un objeto formateado en JSON que contiene el listado de entidades de personas creado junto a su estado


**Figura 4.2:** Estado del sensor creado en el código 4.1

Asociado a este sensor se generará un endpoint de la RESTful API, desde el que se podrá consultar el estado del sensor, obteniendo el listado requerido. En primer lugar, se debe obtener el *Long Lived Acess Token* (ver Figura 4.3), token que debe ser incluido como cabecera al realizar peticiones desde otro terminal a Home Assistant, siendo el método de autenticación predeterminado.

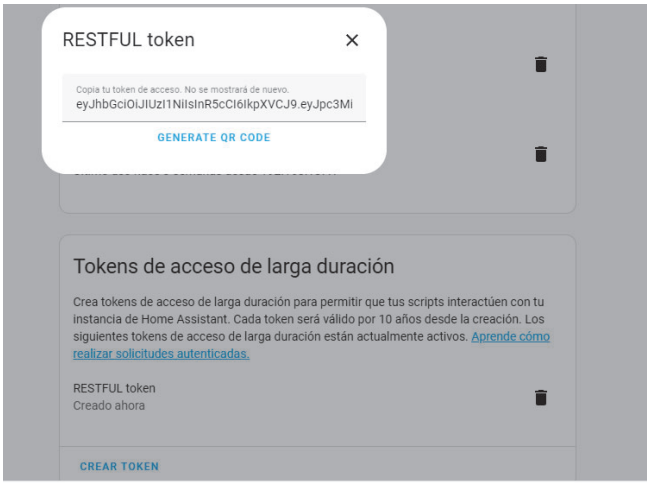



**Figura 4.3:** Generación del *Long Lived Acess Token* desde los ajustes de Home Assistant

Se verifica el correcto funcionamiento haciendo una solicitud al endpoint generado y comprobando que devuelve el listado. Es posible realizar peticiones de este estilo desde la consola de comandos CMD de Windows (Figura 4.4).

```
C:\Users\Miguel>curl -X GET -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI2NmV
jMGY2NWFlODI0NGQ3YWQyYWFjNGJiYWJkMGI4OSIsImlhdCI6MTY4NDg1MTAxOCwiZXhwIjoyMDAwMjExMDE4fQ.SRY1m50xjoeA3fFpHF
nNZBwv-1mP5nffq9W-TVwyf0E" homeassistant.local:8123/api/states/sensor.people_status
```

**Figura 4.4:** Solicitud al endpoint /api/states/sensor.people_status

La respuesta obtenida (Figura 4.5) indica que se encuentra funcionando correctamente, es decir, el endpoint es accesible y ofrece los datos que se desean obtener.

```
{"entity_id":"sensor.people_status","state":"[{\"name\": \"roborescueuma\", \"state\": \"unknown\"}, {\"na
me\": \"miguel\", \"state\": \"home\"}, {\"name\": \"Pascu\", \"state\": \"unknown\"}, {\"name\": \"prueba
1\", \"state\": \"home\"}, {\"name\": \"prueba2\", \"state\": \"home\"}]","attributes":{"friendly_name":"p
eople_status"},"last_changed":"2023-05-25T18:25:17.348856+00:00","last_updated":"2023-05-25T18:25:17.34885
6+00:00","context":{"id":"01H1A2GX0ZDF9NNB7871GN32VH","parent_id":null,"user_id":null}}
C:\Users\Miguel>
```

**Figura 4.5:** Respuesta del endpoint /api/states/sensor.people_status

Para contabilizar cuánto tiempo permanece cada persona cada día en el laboratorio, podemos aprovechar la plataforma History Stats [73]. Esta plataforma de la integración sensor permite generar sensores de tiempo que indican cuánto tiempo permanece alguna entidad en el estado especificado empleando los datos de la base de datos local, según la integración History [74]. Se generará un sensor de tiempo asociado a cada uno de los miembros del equipo, prestando especial atención a la estructura de los nombres, ya que posteriormente será utilizada para conformar el mensaje del cómputo total de horas. Se deben añadir las líneas correspondientes a cada entidad en el fichero de configuración configuration.yaml, ver Código 4.2.

**Código 4.2:** Creación de una entidad temporal por cada persona

```
sensor:
  - platform: history_stats
    name: time_today_miguel
    entity_id: person.miguel
    state: "home"
    type: time
    start: "{{ now().replace(hour=0, minute=0, second=0, microsecond=0) }}"
    end: "{{ now() }}"

  - platform: history_stats
    name: time_today_prueba1
    entity_id: person.prueba1
    state: "home"
    type: time
    start: "{{ now().replace(hour=0, minute=0, second=0, microsecond=0) }}"
    end: "{{ now() }}"

  - platform: history_stats
    name: time_today_prueba2
    entity_id: person.prueba2
    state: "home"
    type: time
    start: "{{ now().replace(hour=0, minute=0, second=0, microsecond=0) }}"
    end: "{{ now() }}"
```

En el código 4.2, se programa una entidad asociada a una persona y dispositivo real junto a dos personas ficticias que tienen asociado el mismo dispositivo. Se hace de esta manera para generar listas que contengan más de un individuo y poder verificar el correcto funcionamiento y gestión de los datos en los distintos terminales. La plataforma history_stats permite establecer la ventana temporal en la que se calculará cuánto tiempo la entidad indicada person.{nombre_persona} ha permanecido en estado "home". La función now() retorna un objeto del tipo DATETIME con la hora y fecha actual. Sustituyendo la hora por las 00 00, comienzo del día actual y fijando el tiempo de fin de la ventana al momento presente now(), este sensor retorna con una resolución de 30 segundos la cantidad de tiempo en formato hh mm ss, tal como se muestra en la Figura 4.6

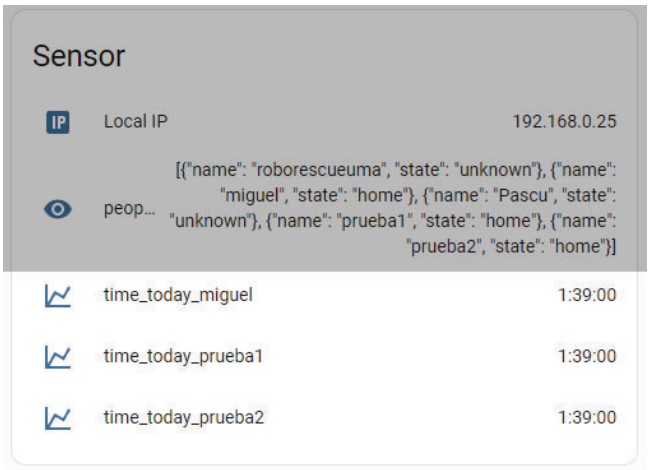



**Figura 4.6:** Datos recopilados por los sensores temporales asociados a cada persona

A continuación, para exportar estos datos, se genera una automatización (Figura 4.7)

programada a las 22:00 cada día, que envía como un objeto JSON el listado de todas las entidades person.{nombre_persona} junto con el tiempo que han permanecido ese día en el laboratorio y la fecha en la que se envía el mensaje.

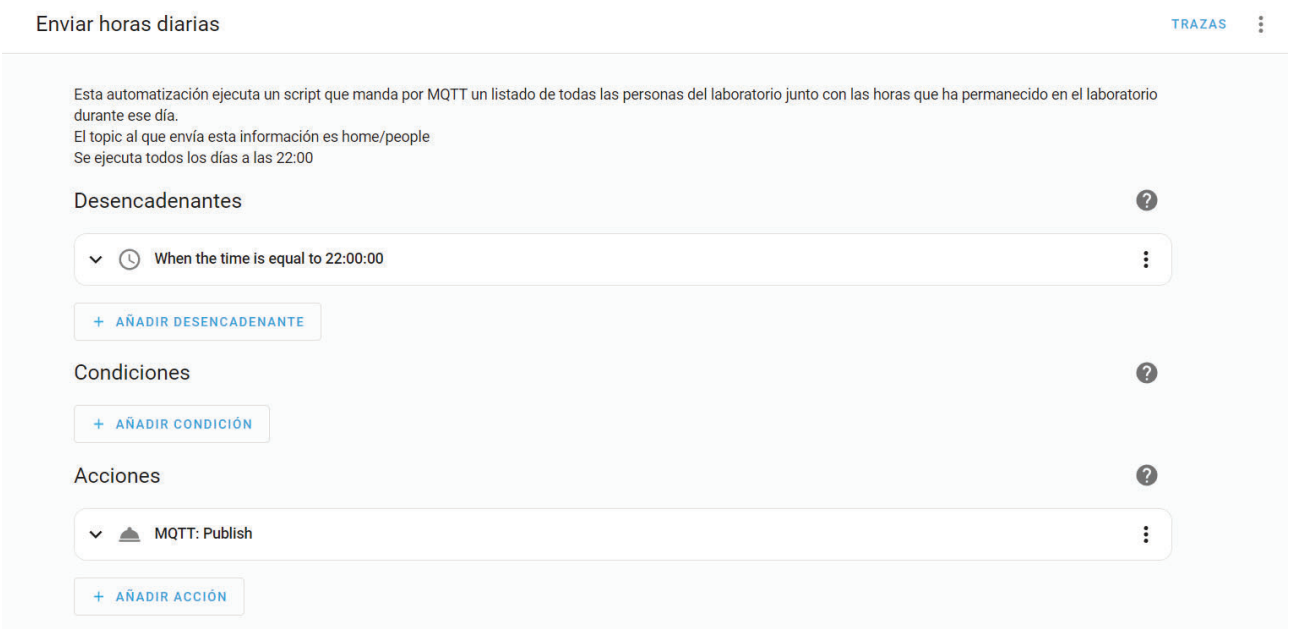



**Figura 4.7:** Automatización diaria programada a las 22:00

En la Figura 4.7 se observa cómo es posible asociar acciones a la automatización, que se ejecutarán según se verificque el desencadenante. En este caso, se llama al servicio *mqtt.publish*, para publicar vía *MQTT* el conjunto de datos.

**Código 4.3:** Acción asociada a la automatización. Publica el tiempo de estancia de cada persona en el laboratorio

```
1
2  service: mqtt.publish
3  data:
4    qos: 1
5    retain: true
6    topic: home/people
7    payload_template: |
8      {
9        "people": [
10         {% for person in states.person %}
11           {
12             "name": "{{ person.name }}",
13             "time_home": "{{ states('sensor.time_today_' ¬ person.name) }}"
14           }
15         {% if not loop.last %},{% endif %}
16       {% endfor %}
17       ],
18     "date":
19         {{'"'¬as_timestamp(now(),default)¬'"' }}
20     }
```

En el Código 4.3, el contenido del payload del mensaje se genera mediante un *template* que itera sobre los estados de todos los miembros del dominio *person*, generando un JSON con

dos campos. Por un lado el campo *people* contiene objetos con el nombre de cada persona y la cantidad de horas registradas ese día. El campo date contiene el timestamp que determina el momento en el que se ha enviado el mensaje. Home Assistant provee de una herramienta que permite comprobar el resultado de los *templates* programados. Mediante esta herramienta se puede verificar que el resultado de la ejecución del *template* (Figura 4.8) genera el mensaje JSON con la estructura deseada.

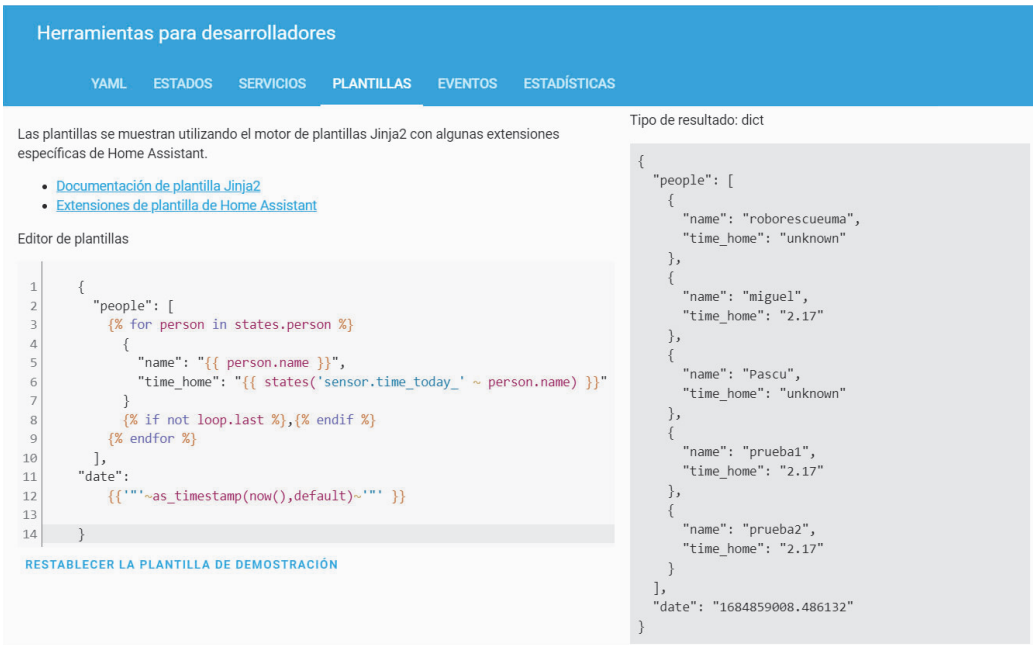



**Figura 4.8:** Verificador de plantillas de Home Assistant. Mensaje JSON generado

## 4.3 Implementación en el servidor

Se parte de la base del servidor local en las mismas condiciones que el servidor de la página web, completamente configurado y con todas las herramientas software adicionales necesarias correctamente instaladas tras seguir el procedimiento detallado en el Apéndice B

Continuando con la estructura presentada en la Sección 3, se comenzará detallando cómo se obtiene en el servidor el listado de las personas presentes en el laboratorio en tiempo real. Como se comenta en capítulos anteriores (Subsección 2.3.3), Wordpress cuenta con RESTful API integrada de forma nativa. Tras verificar que el endpoint del sensor creado es accesible (Figura 4.5), se genera un código en PHP (código 4.4)que consulta este endpoint y filtra los datos para finalmente mostrar quién se encuentra en el laboratorio en tiempo real. Wordpress cuenta con el plugin WPCode que permite insertar fragmentos de código en las páginas deseadas.

**Código 4.4:** Petición HTTP a Home Assistant para recibir el listado en tiempo real. PHP.

```php
<?php

if( is_user_logged_in() ){

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, '192.168.0.25:8123/api/states/sensor.people_status');
    curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiI2NmVjMGY2NWFlODI0N...
    GQ3YWQyYWFjNGJiYWJkMGI4OSIsImlhdCI6MTY4NDg1MTAxOCwiZXhwIjoyMDAwMjExMDE4fQ.SRY1...
    m50xjoeA3fFpHFnNZBwv-1mP5nffq9W-TVwyf0E'
    ));

    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $result = curl_exec($ch);
    curl_close($ch);

    echo '<br>';

    if (!empty($result)){
        $data = json_decode($result,true);
        // echo($data['state']);
        echo '<br>';

        $list = json_decode($data['state'],true);

        foreach($list as $valor){
            if($valor['state']=='home'){

                echo ($valor['name']);
                echo '<br>';

            }

        }

    }


}


?>
```

Al igual que en el ejemplo ejecutado desde la ventana de comandos, desde el programa PHP (Código 4.4) hacemos una solicitud *cURL* al *endpoint* del sensor *people_status* de Home Asssistant. Tras comprobar que el contenido del mensaje no es vacío (causística que puede suceder en caso de apagar la RaspBerry Pi, generando errores al intentar mostrar contenido vacío), se extrae y decodifica el contenido del campo del mensaje recibido. Se recuerda que, como se muestra en la Figura 4.5, el campo *state* contiene el JSON con la información buscada, por lo que será necesario decodificar este contenido. A continuación se itera sobre cada uno de los objetos que contiene la lista obtenida al decodificar el JSON, imprimiendo por pantalla el parámetro *name* asociado a cada uno de los que verifiquen la condición de que su estado sea *"home"*.

El plugin WPCode permite generar un *shortcode* (fragmento de código insertable) asociado al código PHP 4.4, que se puede insertar en cualquier página o post de Wordpress. Tal como se aprecia en la Figura 4.9, tras insertar este código en la página nombrada **Gente en el Laboratorio**, al acceder a dicha página obtenemos un listado de las personas que efectivamente se encuentran presentes.

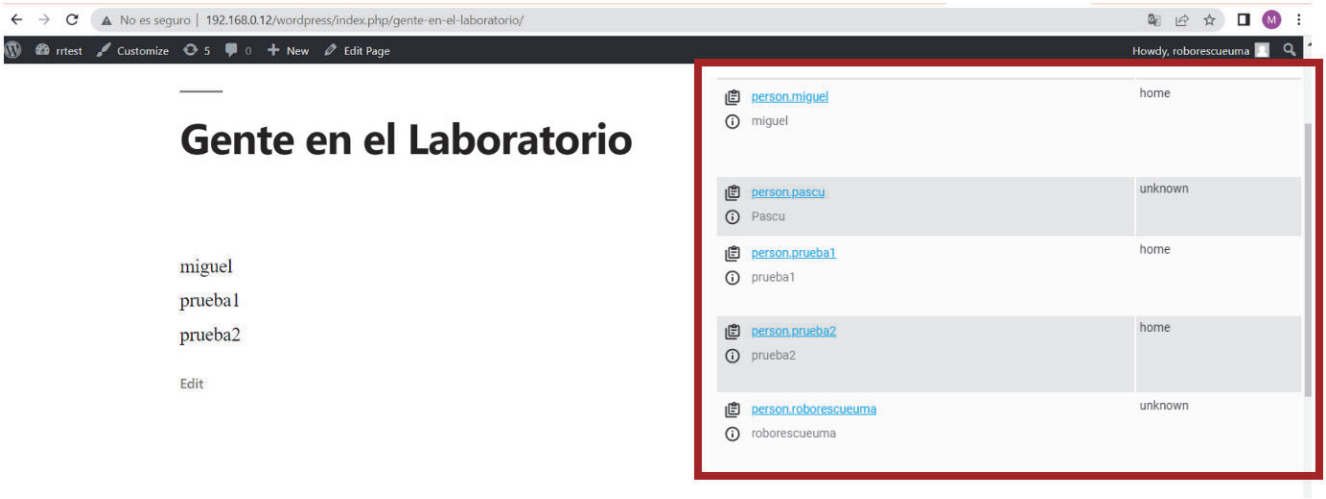



**Figura 4.9:** Página de Wordpress que muestra el listado de génte enviado por Home Assistant. A la derecha estado de las entidades asociadas a cada persona en Home Assistant en el momento de la captura

Para implementar la recepción del cómputo de horas diario se emplea Node-RED como cliente MQTT. Se programa un sencillo flujo (Figura 4.10) que recibe a través del topic *home/-people* el listado de todos los miembros junto con el tiempo que han estado en el laboratorio ese día, para posteriormente almacenarlo en la base de datos de MariaDB *genteLab*.

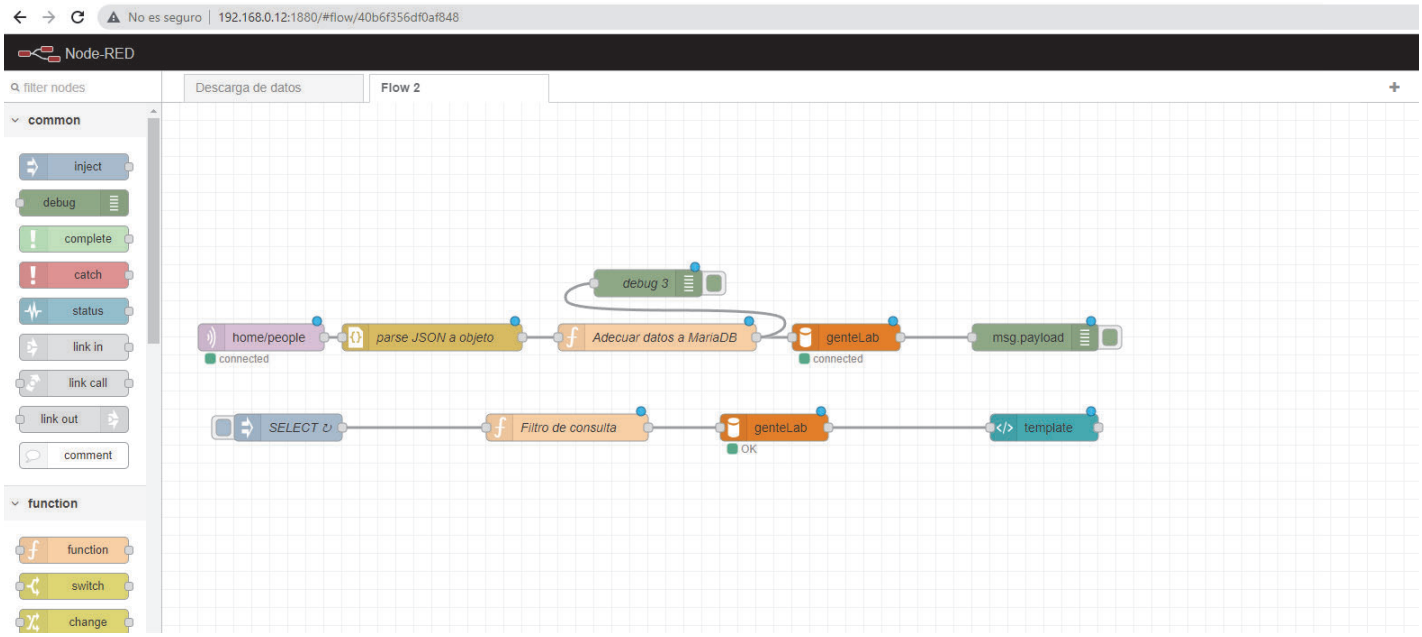



**Figura 4.10:** Vista gráfica del flujo programado en Node-RED

El mensaje recibido se encuentra formateado en JSON, por lo que se debe realizar un *parseo* para obtener objeto/estructura de datos convencional con el objetivo de hacer accesibles sus campos y establecer la jerarquía de los datos que contiene (Ver Figura 4.8). Posteriormente, se emplea una función de JavaScript para realizar el *query* de inserción en la base de datos. La función se muestra en el Código 4.5 como el nodo *."decuar datos a MariaDB"* y transforma la estructura de datos que recibe como entrada en una cadena de caracteres interpretable por la base de datos. Según la especificación del paquete de nodos *node-mySQL*, el contenido de la *query* debe estar contenido en el campo *topic* de la estructura de datos que devuelve la función. Consultar Código 4.5 para ver con detalle el funcionamiento de este algoritmo.

Debido a incompatibilidades con el formato, la fecha debe ser transformada a un objeto de tipo *DATETIME*. Por ello, como el *timestamp* que contiene representa el número de milisegundos tanscurridos desde el inicio de 1970, se ajusta al formato *YY-MM-DD-mm-ss*. Se inyectarán en orden de aparición en el mensaje tomado como input todos los nombres junto a las horas asociadas al día en el que se recibe el mensaje según lo dispuesto en el Código 4.5.

**Código 4.5:** Creación de la solicitud para almacenar los datos en MariaDB. JavaScript.

```
1  var query = "INSERT INTO horarios (nombre, horas, fecha) VALUES ";
2  var values = [];
3
4
5  var fecha = new Date(msg.payload.date*1000);;
6  fecha.setHours(fecha.getHours() + 6);
7  var fechaFormateada = fecha.getFullYear() + "-" +
8      ("0" + (fecha.getMonth() + 1)).slice(-2) + "-" +
9      ("0" + fecha.getDate()).slice(-2) + " " +
10     ("0" + fecha.getHours()).slice(-2) + ":" +
11     ("0" + fecha.getMinutes()).slice(-2) + ":" +
12     ("0" + fecha.getSeconds()).slice(-2);
13
14 for (var i = 0; i < msg.payload.people.length; i++) {
15     var nombre = msg.payload.people[i].name;
16     var horas = parseFloat(msg.payload.people[i].time_home);
17     if (isNaN(horas)){
18         horas= 0;
19     }
20
21     values.push("('" + nombre + "', '" + horas + "', '" + fechaFormateada + "')");
22 }
23
24
25 query += values.join(", ");
26 msg.topic = query;
27 return msg;
```

El segundo hilo de la Figura 4.10 tiene programado cada intervalos de 1 segundo una consulta a la base de datos. El contenido del *query* se especifica en la función JavaScript *Filtro de Consulta*, (Código 4.6).El código muestra un ejemplo de consulta que permite visualizar de forma organizada el cómputo total de horas por mes asociado a cada persona registrada desde Home Assistant.

**Código 4.6:** Consulta a MariaDB. JavaScript.

```
var consulta = {
    "topic": "SELECT nombre, DATE_FORMAT(fecha, '%Y-%m') AS Mes, SUM(horas) AS Horas_totales FROM ...
        horarios GROUP BY Nombre, Mes ORDER BY Horas_totales DESC;"
}
return consulta;
```

Finalmente se muestra mediante una plantilla HTML el resultado de la consulta. Para ello se emplea el nodo *HTML Template* de Node-RED, que permite incrustar fragmentos de HTML en la interfaz. Este código organiza un listado según nombres, meses y horas de estancia por mes, tal como se aprecia en el Código 4.7

**Código 4.7:** Template HTML

```
<table style="width:100%">
  <tr>
    <th>Nombre</th>
    <th>Mes</th>
    <th>Horas</th>
  </tr>
  <tr ng-repeat="x in msg.payload ">
    <td><center>{{msg.payload[$index].nombre}}</center></td>
    <td><center> {{msg.payload[$index].Mes}} </center></td>
    <td><center>{{msg.payload[$index].Horas_totales}}</center></td>
  </tr>
</table>
```

Node-RED proporciona un DashBoard amigable en el que se puede verificar el resultado de esta consulta en tiempo real (Figura 4.11). Si bien esta consulta tiene como objetivo ejemplificar el correcto funcionamiento del sistema, es posible realizar peticiones más específicas, filtrando por meses concretos, personas concretas, entre otras, y ordenando según se prefiera.

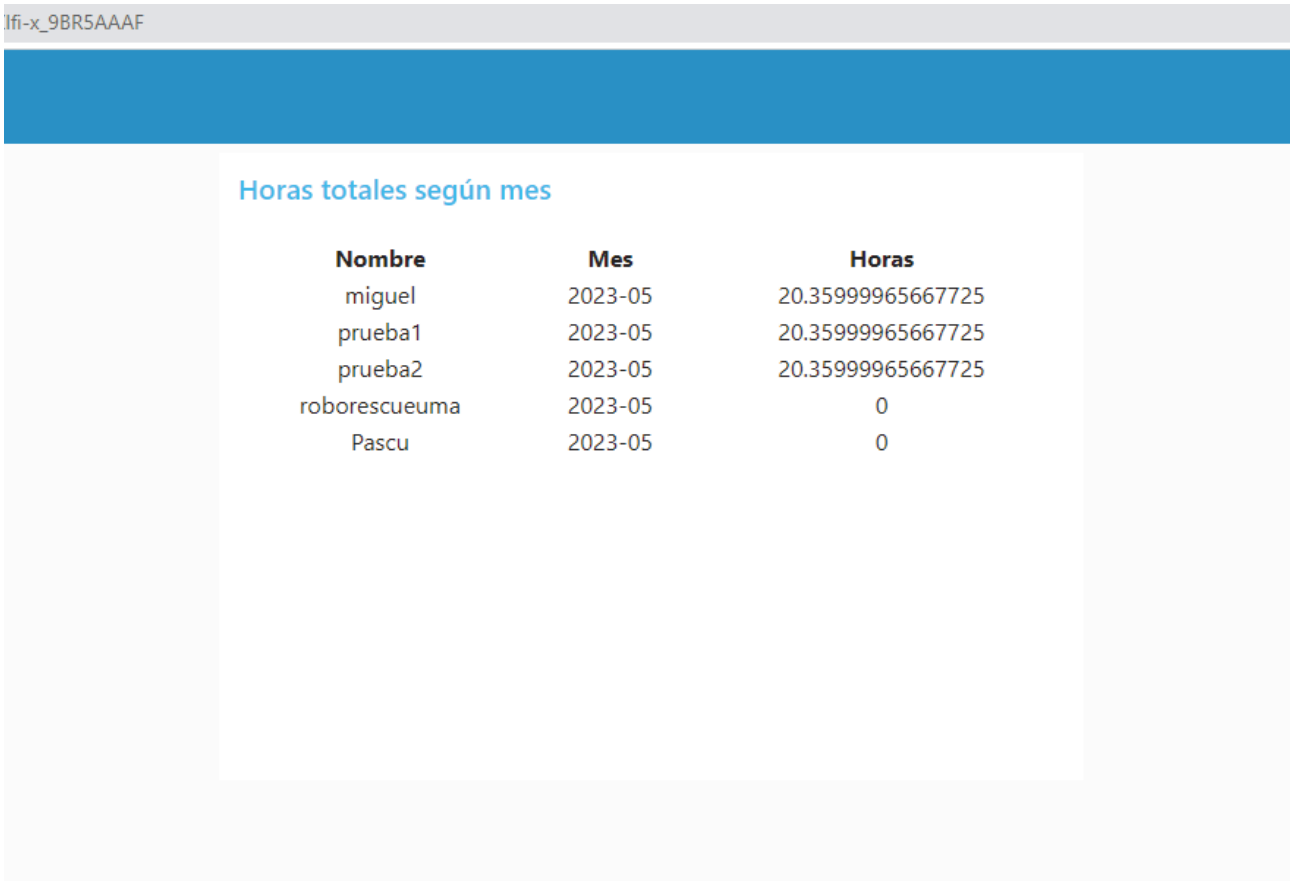



**Figura 4.11:** DashBoard de Node-RED mostrando la plantilla HTML

## 4.4 Conclusiones

En esta sección se ha detallado el proceso de instalación que debe seguirse, tanto desde el punto de vista de Home Assistant como desde el servidor para que el sistema se encuentre completamente implementado. Una vez finalizado el proceso se verifican los resultados obtenidos en el Capítulo 5.

# Capítulo 5

# Resultados

## 5.1 Introducción

Se recopila en este capítulo la información más relevante de forma posterior a la implementación del sistema.

En primer lugar se ha realizado una instalación completa en el entorno del hogar. Esta instalación se ha sometido a prueba durante el periodo de cinco días de funcionamiento ininterrumpido. Se registran como dispositivos conocidos los de los habitantes de la casa, que de forma voluntaria mantienen durante todo el periodo la señal Bluetooth de sus Smartphones activada para poder evaluar la fiabilidad del sistema. El servidor con *CentOS 7* se encuentra ejecutándose, previa configuración, en una máquina virtual en el ordenador personal del autor del TFG.

Por último se analiza la situación tras realizar el montaje en el laboratorio. La imposibilidad hallada a la hora de establecer la conexión desde Home Assistant en la red del laboratorio al servidor que se encuentra en una subred distinta, aún existiendo comunicación con el Servicio Central de Informática, deriva en la propuesta de una solución alternativa temporal, basada en un bot de Telegram [75]. Esta solución permite obtener el listado en tiempo real de los usuarios presentes en el laboratorio, en previsión de solucionar los problemas de compatibilidad existentes en la intercomunicación entre ambos terminales.

## 5.2 Resultados en entorno Personal

Tras el periodo de cinco días de funcionamiento se veerifica el grado de consecución de los objetivos según los reusltados obtenidos. En esta sección se analiza el resultado del funcionamiento global del sistema en el entorno personal del autor. Esto corresponde a una instalación en el hogar, empleando Home Assistant y CentOS 7 instalado en una máquina virtual según el Apéndice B.

### 5.2.1 Control de Presencia en HASS

Las tareas de control de presencia asociadas a Home Assistant se realizan correctamente. En la Figura 5.1 se muestran durante todo el periodo tanto los cambios de estado como el cómputo de horas asociado a cada uno de los usuarios (*Miguel, Angel, Pascu*). Los periodos de ausencia, marcados en color gris, corresponden con franjas horarias en las que, efectivamente, el usuario catalogado como fuera se encontraba fuera de la vivienda. Se puede observar cómo el tiempo acumulado evoluciona correctamente con pendiente constante siempre que un usuario se encuentra en casa. Cada día, a las 00:00, este cómputo se reinicia.

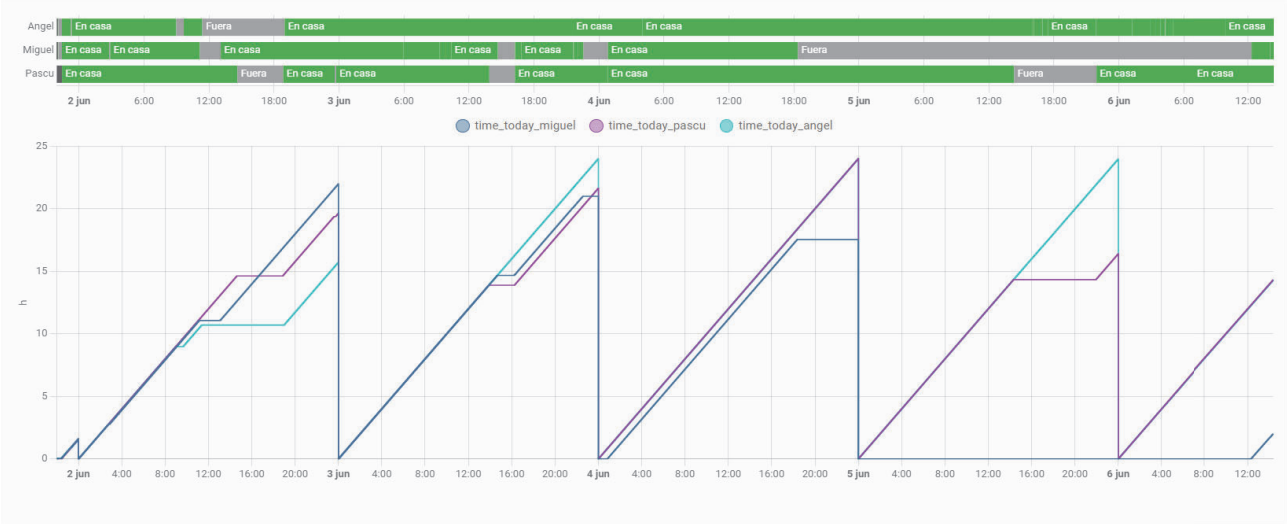



**Figura 5.1:** Evolución del estado de los usuarios. En la parte inferior se muestra el total de horas de estancia acumuladas cada día.

Destaca al observar en detalle la Figura 5.1, como durante algunas noches se produce un parpadeo en el los estados. Este comportamiento se visualiza de forma más clara en la Figura 5.2. Si bien no es muy representativo frente al tiempo total, en caso de que existiesen automatizaciones ligadas a la actualización de estado de algunos miembros del equipo, este comportamiento errático puede llegar a suponer un problema.

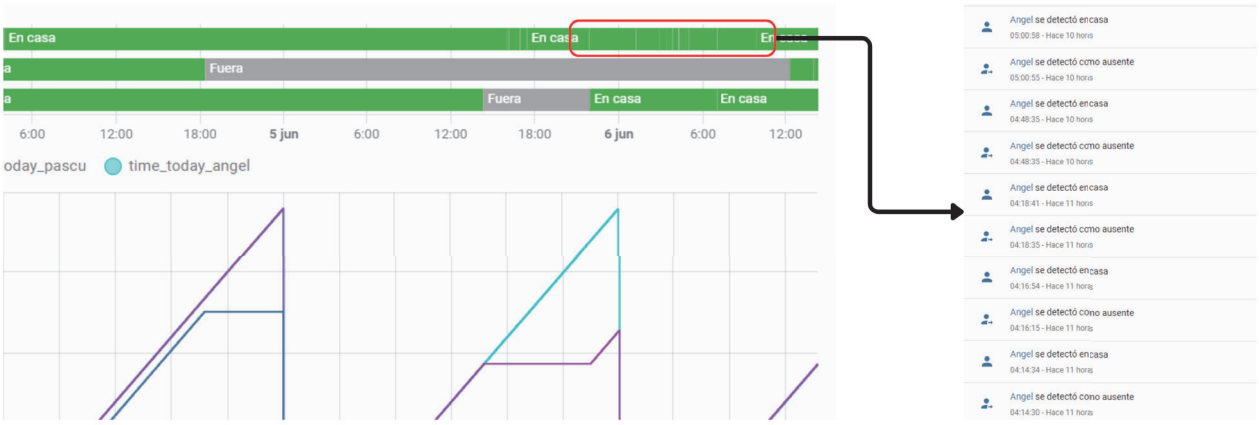



**Figura 5.2:** Comportamiento parpadeante del detector de presencia durante la noche .

El error mostrado en la Figura 5.2 sucede debido a que la ventana para considerar un usuario como ausente es demasiado corta ( ver Código 3.1), hecho que se evidencia por la rápida reconexión observada en la figura. La solución es sencilla; basta con modificar el parámetro indicado, aumentándolo, de forma que no se registren falsos negativos (Ver Línea 6 del Código 5.1).

**Código 5.1:** Líneas agregadas al fichero de configuración para habilitar la integración Bluetooth Tracker

```
1 device_tracker:
2   - platform: bluetooth_tracker
3     new_device_defaults:
4     track_new_devices: false      #Se debe activar esta opción para añadir nuevos dispositivos
5     interval_seconds: 10          #Tiempo entre escaneos al detectar dispositivos
6     consider_home: 120            #AUMENTADO de 60 a 120
```

### 5.2.2 Visualización en tiempo real en WordPress

El desempeño de la función que despliega el listado de gente en el laboratorio en la página de WordPress es correcto. En la Figura 5.3 se muestra cómo este listado se carga correctamente. A continuación se deshabilita la señal bluetooth de uno de los dispositivos (*Miguel*) y dicho cambio se refleja tanto en el estado de la entidad *person.miguel* como en el listado mostrado en la página Web.

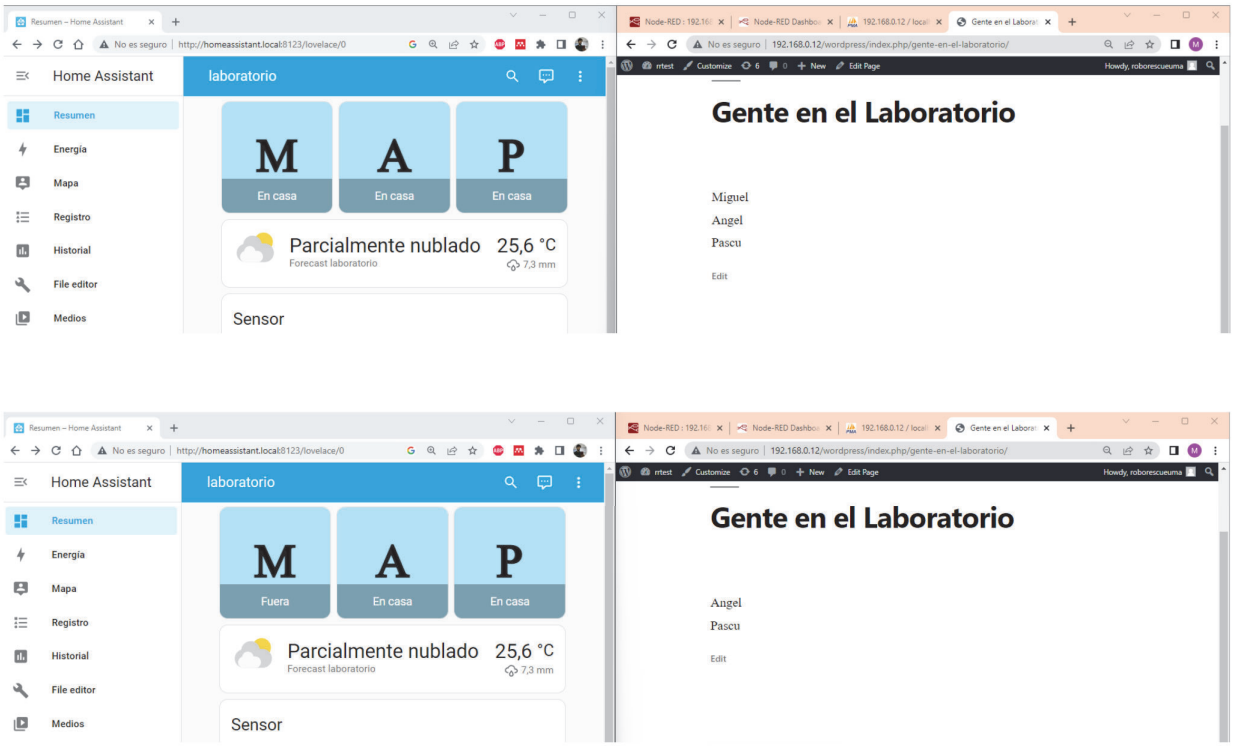



**Figura 5.3:** Comportamiento parpadeante del detector de presencia durante la noche .

### 5.2.3 Comunicación con Node-RED y cómputo de horas

Según lo mostrado en la Figura 5.1, el tiempo total de estancia se calcula correctamente. Se debe verificar si la comunicación con el servidor a la hora de exportar estos datos es correcta. Ejecutando la automatización 'programada a las 22:00 se comprueba que los mensajes MQTT se reciben correctamente en el terminal del servidor, tal como se muestra en la Figura 5.4. Se emplea un nodo *debug* para verificar el mensaje de salida tras recibir y adecuar los datos al formato compatible con MariaDB.

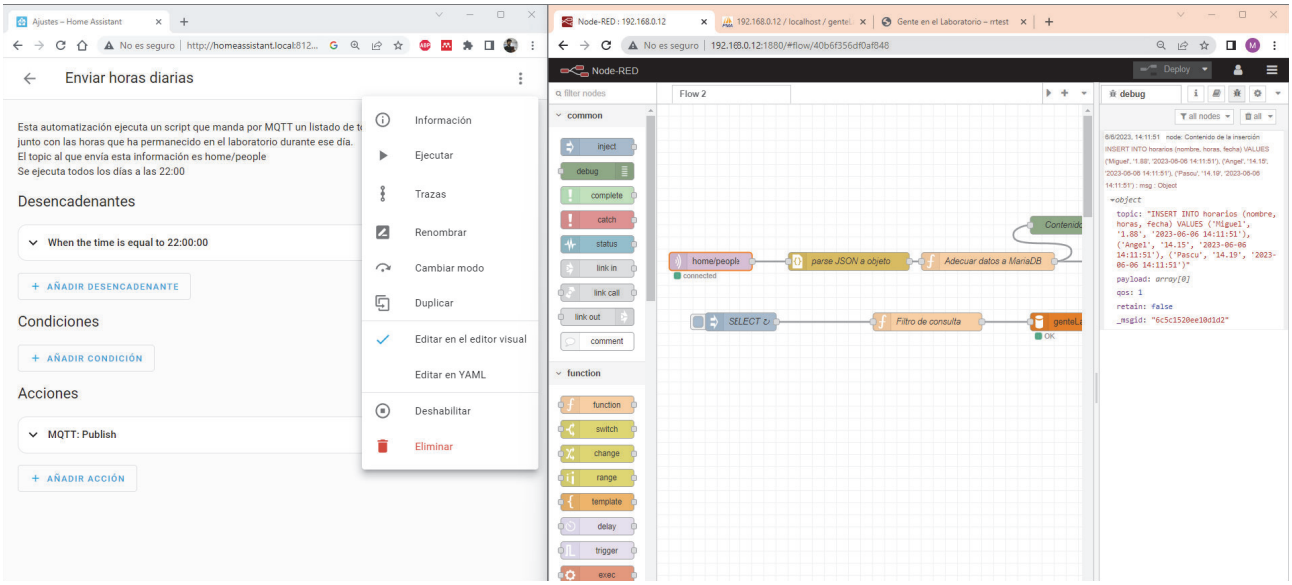



**Figura 5.4:** Ejecución de la automatización que envía el cómputo de horas. Del lado izquierdo Home Assistant, a la derecha Node-RED en el servidor, mostrando el mensaje construido para realizar la inserción en la base de datos.

Desde la herramienta *phpMyAdmin* se puede verificar que la inserción en la base de datos es correcta, tal como se muestra en la Figura 5.5. En esta misma figura se puede comprobar cómo las inserciones programadas a las 22:00 de cada día se han realizado correctamente, excepto la noche del 4 de Mayo, ya que como se muestra en la Figura 5.1, esa noche *Miguel* se encuentra fuera de casa. Al encontrarse el servidor ejecutándose en una máquina virtual del ordenador personal y este no estar en la red, no se almacenan los datos. Por otro lado, como los mensajes MQTT tienen el flag *retained*, al recuperar la conexión, independientemente de la hora, se recibe el último mensaje que desde Home Assistant se hubiera mandado. El correspondiente a la noche del 4 de Mayo se sobrescribe por tanto con el de la noche del 5 de Mayo y al recuperar la conexión (revisar de nuevo historial en Figura 5.1), el mensaje retenido es recibido por Node-RED y almacenado en la base de datos.

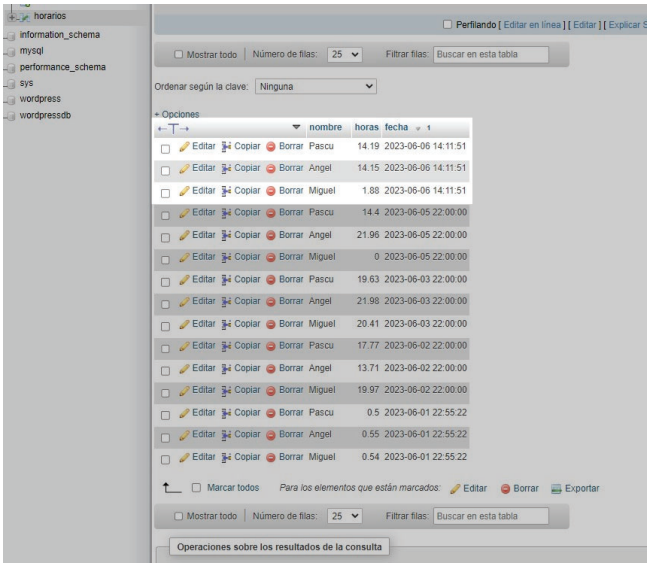



**Figura 5.5:** Inserción correcta en la base de datos del mensaje recibido según la Figura 5.4. Se muestran el resto de inserciones programadas a las 22:00 de cada día, exceptuando la primera realizada de forma manual a las 22:55 y la correspondiente al día 4 de Mayo.

La interfaz gráfica de Node-RED muestra en un *template HTML* el resultado de la consulta a la base de datos en la que se realiza el cómputo según meses de las horas correspondientes al tiempo de estancia de cada usuario. En la Figura 5.6 se puede observar la comparativa entre antes y después de la inserción de los datos enviados por la automatización ejecutada manualmente según la Figura 5.4.


**Figura 5.6:** Interfaz gráfica de Node-RED. Resultado ordenado según meses de la consulta realizada a la base de datos de los horarios en la que se suman las horas totales de cada miembro. A la derecha resultado posterior a la inserción mostrada en la Figura 5.4

## 5.3 Resultados en el laboratorio

Una vez realizado todo el proceso de configuración detallado en el Capítulo 4 y Apéndices A, B no se consigue realizar de forma exitosa la interconexión entre Home Assistant y el Servidor. En la red de la Universidad de Málaga, las conexiones entre subredes diferentes deben ser habilitadas de forma explícita por medios que no están al alcance de un alumno. Desde el Servicio Central de Informática, se habilita en el proxy HTTP del servidor que ejecuta la página web la conexión a la dirección de Home Assistant en el puerto requerido. No obstante, las pruebas realizadas muestran que la comunicación no es exitosa. por lo que se propone una solución alternativa basada en Node-RED y Telegram. El desarrollo de esta solución, al tratarse de una solución alternativa, aparece indexado en el Apéndice C.

### 5.3.1 Interacción con el bot de Telegram

El bot de Telegram es accesible tanto desde terminales web como desde la aplicación de Telegram disponible para Smartphones. Tras iniciar el bot, hecho que sucede la primera vez que se inicializa o mediante comando, este muestra un mensaje introductorio con las posibilidades que ofrece. Este mensaje aparece en la Figura 5.7, en la que se muestra el inicio de la conversación con el bot.


**Figura 5.7:** Bot de Telegram. Mensaje de bienvenida y comandos disponibles.

Se verifica el correcto funcionamiento alternando entre diferentes estados asociados a las entidades *person.miguel* y *person.victor_manuel*. En la Figura 5.8 se muestra la respuesta correcta del bot ante el comando que requiere el listado, proporcionando el nombre de ambos miembros del equipo.

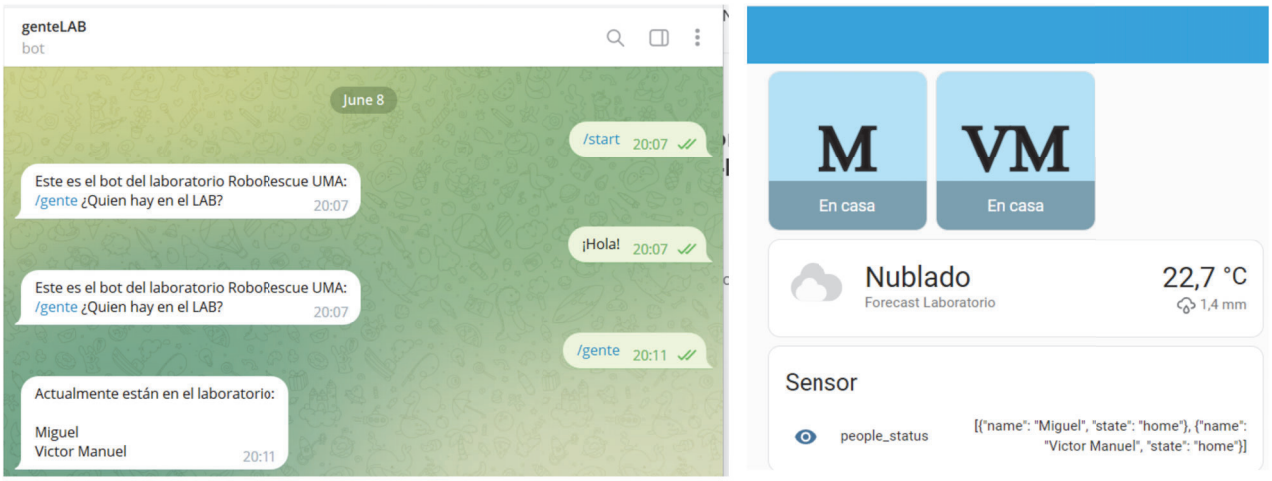



**Figura 5.8:** Bot de Telegram y *dashboard* de HASS. A la izquierda respuesta del bot de Telegram. A la derecha interfaz de HASS mostrando a dos personas en estado "*en casa*"

A continuación se ausenta *Miguel*, tal como se observa en la Figura 5.9. Una vez HASS lo detecta como "*fuera*"se vuelve a ejecutar el comando en Telegram, obteniendo la respuesta esperada.

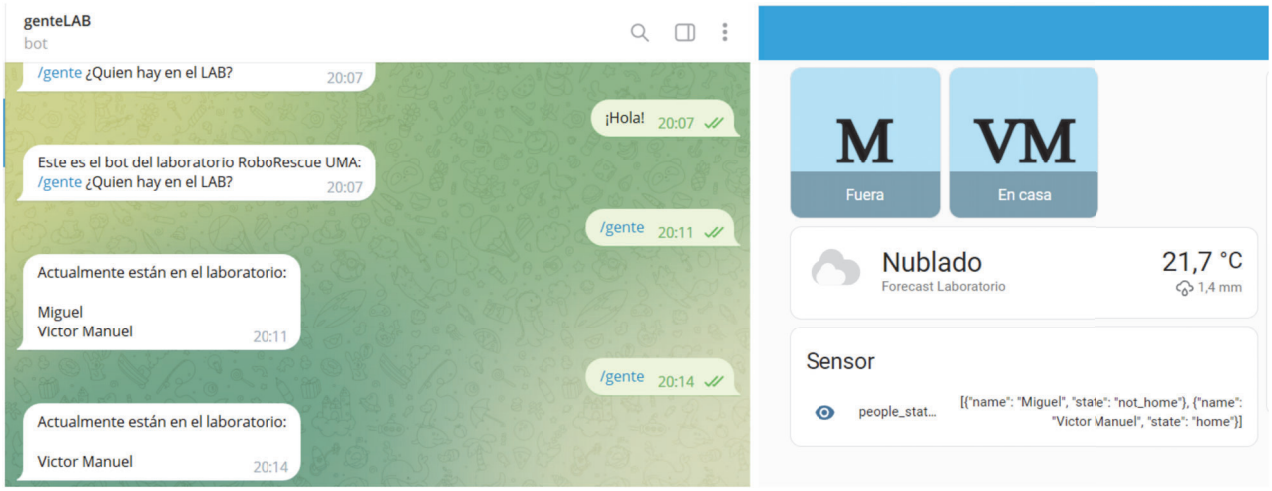



**Figura 5.9:** Bot de Telegram y *dashboard* de HASS. A la izquierda respuesta del bot de Telegram. A la derecha interfaz de HASS mostrando sólo una persona .*en casa*"

Finalmente se verifica la respuesta cuando ninguna persona se encuentra en el laboratorio, tras desactivar la señal Bluetooth en el smartphone asociado a *Victor Manuel*, tal como se observa en la Figura 5.10.

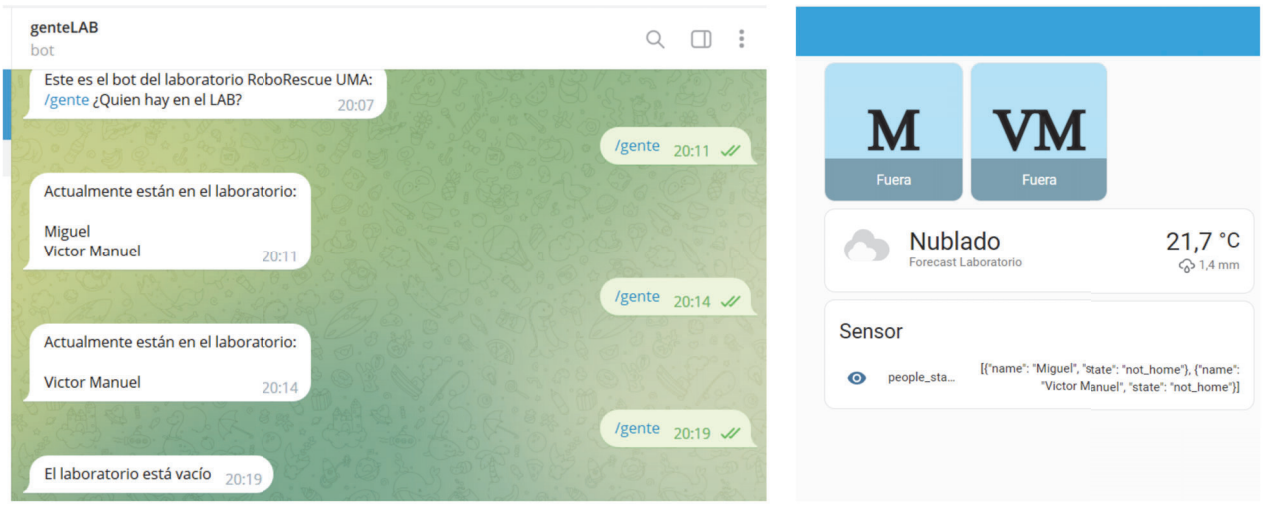



**Figura 5.10:** Bot de Telegram y *dashboard* de HASS. A la izquierda respuesta del bot de Telegram. A la derecha interfaz de HASS mostrando nadie "*en casa*"

## 5.4 Conclusiones

Los resultados mostrados en este capítulo son satisfactorios en la instalación realizada para una red controlada en el hogar, respondiendo a las expectativas del funcionamiento del sistema.

La detección de presencia mediante la integración *Bluetooth Tracker* responde según lo esperado durante los días de prueba. Ocurren algunas desconexiones puntuales momentáneas (del orden de segundos) debido a la corta duración de la ventana temporal en la que se considera un dispositivo en casa. Aumentando este valor se solventa dicho comportamiento irregular.

El cómputo de horas totales es realizado de forma exitosa, ligando un sensor temporal cuyos valores resultados son correctos a cada una de las personas configuradas en el sistema.

La comunicación entre Home Assistant y el servidor se realiza de manera efectiva. Por un lado, el listado solicitado vía RESTFul API desde WordPress se visualiza correctamente y actualizado en tiempo real en la página web. Finalmente, las inserciones en la base de datos asociadas al tiempo de permanencia de cada usuario son realizadas correctamente por Node-RED, permitiendo visualizar de forma posterior dichos datos de forma ordenada.

La comunicación entre Home Assistant y el servidor alojado en los dominios de la UMA no se realiza correctamente. La conexión debe ser habilitada de forma explícita desde el SCI, por lo que, aún habiéndose habilitado un proxy para ello, tras las primeras pruebas no satisfactorias se propone una solución alternativa basada en Telegram. Esta solución permite de manera efectiva conocer el listado de personas que se encuentran en el laboratorio mediante el empleo de un bot. Es necesario por tanto explorar más detenidamente si la intercomunicación entre el servidor y Home Assistant es posible, con la asistencia del SIC, para finalmente utilizar la solución propuesta en el Capítulo 4.

# Capítulo 6

# Conclusión

## 6.1 Introducción

Con este capítulo se concluye el Trabajo de Fin de grado, aportando una serie de valoraciones tanto de carácter funcional como de carácter personal en relación al contenido desarrollado y los resultados obtenidos. De forma complementaria, se añaden unos comentarios acerca de las posibles ampliaciones que pueden realizarse tomando como punto de partida tanto este documento como la implementación del sistema propuesto.

## 6.2 Conclusiones

En el presente documento se ha ejecutado la implementación de un sistema de control de presencia basado en Bluetooth para el laboratorio de RoboRescue UMA. La detección se realiza con el único requisito por parte de los usuarios de habilitar la señal Bluetooth en sus smartphones. Este sistema mejora en flexibilidad y comodidad al empleado actualmente, proponiendo una serie de mecanismos con funcionalidades ampliables de cara a extender las posibilidades de dicho sistema.

La solución final propuesta, es decir, el empleo de *Bluetooth Tracker* integrado en Home Assistant ha superado las pruebas realizadas, demostrando ser suficientemente fiable. Se ha priorizado en la implementación propuesta la sencillez en términos de configuración, la robustez y la universalidad de la solución (el Smartphone es el dispositivo más extendido). El funcionamiento global del sistema, tanto en términos de detección de presencia como de comunicación con el servidor a nivel local ha sido satisfactorio a lo largo de las pruebas a las que ha sido sometido.

Se ha implementado una solución alternativa funcional basada en Node-RED y Telegram para el despliegue del sistema en el laboratorio. Mediante esta implementación se pueden realizar consultas al listado de personas en tiempo real desde cualquier Smartphone, interactuando con

un *bot* a través de la aplicación de Telegram. Se ha instalado en el terminal que ejecuta Home Assistant la aplicación de Node-RED. Esta aplicación permite realizar una amplia variedad de funciones y automatizaciones de forma visual, entre las que se encuentra esta solución propuesta.

Es destacable la capacidad que tiene Home Assistant de ofrecer un ecosistema en el que múltiples aproximaciones a esta problemática, tanto comerciales como *user-made*, tienen cabida. La posibilidad de combinar varias soluciones con tareas de automatización abre un enorme abanico de opciones a la hora de automatizar el laboratorio. No obstante, el ecosistema de Home Assistant es muy rígido. De igual forma que se facilitan determinadas tareas que empleando otro software serían enormemente complejas, es posible encontrarse con muchas limitaciones a la hora de desarrollar funcionalidades específicas para unos requisitos concretos, como es el caso de la generación y exportación de los listados de gente en el laboratorio y horarios. La rigidez en la arquitectura de Home Assistant, debido a su enfoque basado en estados y automatizaciones puede llegar a dificultar según qué tareas. No obstante, la existencia de numerosas aplicaciones complementarias como Node-RED y ESPHome no sólo contrarrestan esta limitación, sino que aumentan las caracetrísticas de que ofrece este ecosistema en gran medida.

Finalmente, como apreciación personal, el proyecto ha supuesto una iniciación en la domótica, campo del que anteriormente poseía un conocimiento muy limitado, motivándome a realizar proyectos similares basados en la automatización inteligente en el entorno personal. Adicionalmente, el proceso de configuración y comunicación con el servidor ha supuesto una dificultad añadida debido a mi desconocimiento en la materia, pero ser capaz de llevarlo a cabo me ha proporcionado una perspectiva más amplia de estas tecnologías, contribuyendo enormemente a mi formación.

## 6.3 Ampliaciones futuras

Tal como se menciona anteriormente, si el balance funcionalidad-esfuerzo se considera positivo, es interesante complementar la técnica de control de presencia propuesta con alguna de las alternativas exploradas.

Es posible además implementar sendas funciones de notificación a los usuarios aprovechando la compatibilidad del propio Home Assistant con herramientas como Node-RED. En este contexto se podría, por ejemplo, programar funcioalidades añadidas en forma de comandos para el Bot de Telegram creado, que permitan notificar de forma personalizada a cada usuario según las preferencias que elija, o envíen mensajes de difusión cada vez que un coordinador del programa o jefe de equipo se encuentre en el laboratorio.

Por otro lado, se debe profundizar en la interconexión entre la RaspBerry ejecutando HASS en el laboratorio y el servidor ejecutando la página web del programa y la base de datos que contabiliza las horas totales. Queda por tanto pendiente, con asistencia del SIC, establecer las reglas que finalmente permitan la comunicación fluida entre Home Assistant y el servidor que ejecuta la página web según las disposiciones del Capítulo 4. Si bien esta es la implementación propuesta, de terminar no siendo viable, se puede considerar el flujo propuesto en Node-RED y alternativas como bots personalizados de Telegram para realizar las tareas propuestas, además de bases de datos a nivel local, en la misma subred, de modo que sean inmediatamente visibles

para HASS, para almacenar los históricos de horas totales.

Existen multitud de dispositivos inteligentes que pueden aprovechar la presencia como desencadenante de automatizaciones varias. Los dispositivos *sonoff* y enchufes inteligentes podrían programarse para que, iterando sobre todos los miembros del dominio *person*, al detectar que ninguno de ellos se encuentra presente, apague las luces.

En términos generales, no limitándose exclusivamente a las automatizaciones basadas en la detección de presencia, Home Assistant ofrece una plataforma para el desarrollo de aplicaciones basadas en IOT que complementen la interactividad con el laboratorio. La compatibilidad con las placas ESP mediante software como ESPHome permite automatizar una enorme variedad de actuadores y coordinar la lectura de igual variedad de sensores. Sumado a las prácticamente ilimitadas posibilidades que ofrece la compatibilidad con ESP, los *add-on*s disponibles y la variedad de dispositivos encontrados en el mercado, mediante Home Assistant se ofrecen herramientas para tareas como: Control inteligente del alumbrado, visualización de imágenes por cámara WiFi basada en desencadenantes (ausencia de personal en el laboratorio, detección de presencia, etc.), control inteligente de actuadores basados en WiFi mediante comandos de voz, entre tantas otras.

# Bibliografía

[1] *What is Home Assistant, and what can it do? - HASS guide*. dirección: https://home-assistant-guide.com/guide/what-is-home-assistant-and-what-can-it-do/ (visitado 27-05-2023).

[2] *What is Shifts? - Microsoft Support*. dirección: https://support.microsoft.com/en-us/office/what-is-shifts-f8efe6e4-ddb3-4d23-b81b-bb812296b821 (visitado 23-05-2023).

[3] *RoborRescue - UMA*. dirección: https://roborescue.uma.es/ (visitado 23-05-2023).

[4] *¿Que es Raspberry Pi? - Raspberry Pi*. dirección: https://raspberrypi.cl/que-es-raspberry/ (visitado 23-05-2023).

[5] *Raspberry Pi 3B+ — MagPi*. dirección: https://magpi.raspberrypi.com/articles/raspberry-pi-3bplus-specs-benchmarks (visitado 23-05-2023).

[6] *Operating Systems for Raspberry Pi - RaspBerry Pi Stater Kits*. dirección: https://www.raspberrypistarterkits.com/products/operating-systems-raspberry-pi/ (visitado 23-05-2023).

[7] *¿Por qué ha subido el precio de la RaspBerry Pi? - El Output*. dirección: https://eloutput.com/noticias/tecnologia/raspberry-pi-subida-precio-temporal/ (visitado 23-05-2023).

[8] *La Raspberry Pi sube de precio, y la culpa la tiene la escasez de chip - Xataka*. dirección: https://www.xataka.com/ordenadores/raspberry-pi-sube-precio-culpa-tiene-escasez-chips (visitado 23-05-2023).

[9] *Mini PC | REVIEWBOX España*. dirección: https://www.reviewbox.es/mini-pc/ (visitado 23-05-2023).

[10] *Power consumption Pi Zero W – RasPi.TV*. dirección: http://raspi.tv/2017/how-much-power-does-pi-zero-w-use (visitado 23-05-2023).

[11] *ESP8266 - Wikipedia*. dirección: https://es.wikipedia.org/wiki/ESP8266 (visitado 27-05-2023).

[12] *ESP32 - Wikipedia*. dirección: https://es.wikipedia.org/wiki/ESP32 (visitado 27-05-2023).

[13] *GitHub - home-assistant/core*. dirección: https://github.com/home-assistant/core (visitado 23-05-2023).

[14] *Home Assistant - GitHub*. dirección: https://github.com/home-assistant (visitado 23-05-2023).

[15] *Installation Methods - Home Assistant*. dirección: https://www.home-assistant.io/installation/#compare-installation-methods (visitado 23-05-2023).

[16] *Raspberry Pi - Home Assistant*. dirección: https://www.home-assistant.io/installation/raspberrypi#suggested-hardware (visitado 23-05-2023).

[17] *Security leak using WiFi cameras - HASS guide*. dirección: https://home-assistant-guide.com/news/2021/05/18/pull-the-plug-on-your-eufy-security-cameras-eufycam-right-now/ (visitado 23-05-2023).

[18] *Integrations - Home Assistant*. dirección: https://www.home-assistant.io/integrations/ (visitado 23-05-2023).

[19] *The CentOS Project - CentOS*. dirección: https://www.centos.org/ (visitado 27-05-2023).

[20] *CentOS - Wikipedia*. dirección: https://es.wikipedia.org/wiki/CentOS (visitado 27-05-2023).

[21] *Sistema operativo Red Hat Enterprise Linux - Red Hat*. dirección: https://www.redhat.com/es/technologies/linux-platforms/enterprise-linux (visitado 27-05-2023).

[22] *Imagen: ArtWork/Brand/Logo - CentOS Wiki*. dirección: https://wiki.centos.org/ArtWork/Brand/Logo (visitado 23-05-2023).

[23] *Servidor HTTP Apache - Wikipedia*. dirección: https://es.wikipedia.org/wiki/Servidor_HTTP_Apache (visitado 23-05-2023).

[24] *Imagen: Apache Software Foundation Graphics - Apache*. dirección: https://www.apache.org/foundation/press/kit/#logo (visitado 23-05-2023).

[25] *Introduction to Relational Databases - MariaDB Knowledge Base*. dirección: https://mariadb.com/kb/en/introduction-to-relational-databases/ (visitado 23-05-2023).

[26] *MariaDB en resumen - MariaDB.org*. dirección: https://mariadb.org/es/ (visitado 23-05-2023).

[27] *Official MariaDB Logos - MariaDB*. dirección: https://mariadb.com/es/about-us/logos/ (visitado 27-05-2023).

[28] *PHP: ¿Qué es PHP? - Manual*. dirección: https://www.php.net/manual/es/intro-whatis.php (visitado 23-05-2023).

[29] *PHP: ¿qué es, para qué sirve y cuáles son sus características?* Dirección: https://rockcontent.com/es/blog/php/ (visitado 27-05-2023).

[30] *PHP, Download Logos - PHP*. dirección: https://www.php.net/download-logos.php (visitado 23-05-2023).

[31] *phpMyAdmin*. dirección: https://www.phpmyadmin.net/ (visitado 23-05-2023).

[32] *WordPress - Wikipedia*. dirección: https://es.wikipedia.org/wiki/WordPress (visitado 23-05-2023).

[33] *Gráficos y logotipos - WordPress España*. dirección: https://es.wordpress.org/about/logos/ (visitado 23-05-2023).

[34] *Funcionamiento de Wordpress - ideandoazul*. dirección: https://ideandoazul.com/wordpress/como-funciona-wordpress/ (visitado 23-05-2023).

[35] *Node-RED*. dirección: https://nodered.org/ (visitado 23-05-2023).

[36] *Resources - Node-RED*. dirección: https://nodered.org/about/resources/ (visitado 27-05-2023).

[37] *MQTT Version 3.1.1 - OASIS*. dirección: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html (visitado 27-05-2023).

[38] *Publish and Subscribe in MQTT Part - HiveMQ*. dirección: https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/ (visitado 27-05-2023).

[39] *MQTT - Home Assistant*. dirección: https://www.home-assistant.io/integrations/mqtt/#choose-a-mqtt-broker (visitado 27-05-2023).

[40] *GitHub - eclipse/mosquitto*. dirección: https://github.com/eclipse/mosquitto#eclipse-mosquitto (visitado 27-05-2023).

[41] *¿Qué es una API REST? - IBM*. dirección: https://www.ibm.com/es-es/topics/rest-apis (visitado 27-05-2023).

[42] *REST API Handbook | WordPress Developer Resources*. dirección: https://developer.wordpress.org/rest-api/ (visitado 26-05-2023).

[43] *REST API - Home Assistant Developer Docs*. dirección: https://developers.home-assistant.io/docs/api/rest/ (visitado 23-05-2023).

[44] *Device Tracker - Home Assistant*. dirección: https://www.home-assistant.io/integrations/device_tracker/ (visitado 23-05-2023).

[45] *Person - Home Assistant*. dirección: https://www.home-assistant.io/integrations/person/ (visitado 23-05-2023).

[46] *BlueZ*. dirección: http://www.bluez.org/ (visitado 23-05-2023).

[47] *Bluetooth Tracker - Home Assistant*. dirección: https://www.home-assistant.io/integrations/bluetooth_tracker/ (visitado 23-05-2023).

[48] *Classifying the Internet of Things - Home Assistant*. dirección: https://www.home-assistant.io/blog/2016/02/12/classifying-the-internet-of-things/#classifiers (visitado 23-05-2023).

[49] *Same device multiple times in known$_d$evices−Configuration−HomeAssistantCommunity*. dirección: https://community.home-assistant.io/t/same-device-multiple-times-in-known-devices/111258 (visitado 23-05-2023).

[50] *BLE based Proximity Control using ESP32 – circuitdigest*. dirección: https://circuitdigest.com/microcontroller-projects/ble-based-proximity-control-using-esp32 (visitado 23-05-2023).

[51] *BLE (Bluetooth Low Energy) - Elt*. dirección: https://www.elt.es/ble-bluetooth-low-energy (visitado 23-05-2023).

[52] *Bluetooth de baja energía - Wikipedia*. dirección: https://es.wikipedia.org/wiki/Bluetooth_de_baja_energ%C3%ADa (visitado 23-05-2023).

[53] *iBeacon - Wikipedia*. dirección: https://es.wikipedia.org/wiki/IBeacon (visitado 23-05-2023).

[54] *Imagen: Keyfob beacon - Keyfob*. dirección: https://www.global-tag.com/wp-content/uploads/2017/06/Keyfob_beacony_2_big.jpg.

[55] J. K. Becker, D. Li y D. Starobinski, «Tracking Anonymized Bluetooth Devices», *Proceedings on Privacy Enhancing Technologies*, vol. 2019, págs. 50-65, 3. DOI: 10.2478/popets-2019-0036.

[56] *BLE device only appears 'away' after reboot host · Issue 79539 · home-assistant/core - GitHub*. dirección: https://github.com/home-assistant/core/issues/79539 (visitado 23-05-2023).

[57] *Bluetooth LE Tracker reports offline device as home · Issue 84802 · home-assistant/core - GitHub*. dirección: https://github.com/home-assistant/core/issues/84802#issuecomment-1492984766 (visitado 23-05-2023).

[58] *Fix bluetooth_le_tracker reporting devices Home when they leave by ProtoxiDe22 · Pull Request 90641 · home-assistant/core - GitHub*. dirección: https://github.com/home-assistant/core/pull/90641/commits/124bc8482b20a471a7c82bad3bf9ff9b81eeef11 (visitado 23-05-2023).

[59] *Merged Fix bluetooth_le_tracker reporting devices Home when they leave by ProtoxiDe22 · Pull Request 90641 · home-assistant/core - GitHub*. dirección: https://github.com/home-assistant/core/pull/90641/commits/6d5e942e448dd1ce7e78dae9b2bdd6352d0a9487 (visitado 23-05-2023).

[60] *2023.4 Release - Home Assistant*. dirección: https://www.home-assistant.io/blog/2023/04/05/release-20234/ (visitado 23-05-2023).

[61] *Bluetooth LE Tracker reports offline device as home · Issue 84802 · Issue Comment 1506451417 · home-assistant/core - GitHub*. dirección: https://github.com/home-assistant/core/issues/84802#issuecomment-1506451417 (visitado 23-05-2023).

[62] *iBeacon Tracker - Known Working Devices - Home Assistant*. dirección: https://www.home-assistant.io/integrations/ibeacon/#known-working-devices (visitado 26-05-2023).

[63] *BLE Advertising Primer - Argenox*. dirección: https://www.argenox.com/library/bluetooth-low-energy/ble-advertising-primer/ (visitado 27-05-2023).

[64] *Home Assistant Cloud - Home Assistant*. dirección: https://www.home-assistant.io/cloud/ (visitado 23-05-2023).

[65] *ESPHome — ESPHome*. dirección: https://esphome.io/ (visitado 23-05-2023).

[66] *ESP32 Bluetooth Low Energy Tracker Hub — ESPHome*. dirección: https://esphome.io/components/esp32_ble_tracker.html (visitado 23-05-2023).

[67] *Home - ESPresense*. dirección: https://espresense.com/ (visitado 23-05-2023).

[68] *MQTT Room Presence - Home Assistant*. dirección: https://www.home-assistant.io/integrations/mqtt_room/ (visitado 23-05-2023).

[69] *Home Assistant Companion Docs - Home Assistant Companion Docs*. dirección: https://companion.home-assistant.io/ (visitado 23-05-2023).

[70] *Android - ESPresense*. dirección: https://espresense.com/beacons/android (visitado 23-05-2023).

[71] *Apple - ESPresense*. dirección: https://espresense.com/beacons/apple#ios-manual-setup (visitado 23-05-2023).

[72] *Configuration - ESPresense*. dirección: https://espresense.com/configuration (visitado 23-05-2023).

[73] *History Stats - Home Assistant*. dirección: https://www.home-assistant.io/integrations/history_stats/ (visitado 25-05-2023).

[74] *History - Home Assistant*. dirección: https://www.home-assistant.io/integrations/history/ (visitado 25-05-2023).

[75] *Telegram Messenger - Telegram*. dirección: https://telegram.org/ (visitado 27-05-2023).

# Apéndice A

# Configuración y primeros pasos en Home Assistant

## A.1 Instalación de Home Assistant

Para instalar Home Assistant en un dispositivo RaspBerry Pi 3B o similares se requiere un lector de tarjetas MicroSD y una tarjeta MicroSD de al menos 8GB (más es recomendado).

La imagen del sistema operativo se puede obtener directamente desde el repositorio oficial de GitHub de Home Assistant, asegurando así que se obtiene la última versión disponible. El enlace de descarga de la última versión a fecha de redacciónn del presente documento es el siguiente: https://github.com/home-assistant/operating-system/releases/download/10.1/haos_rpi3-64-10.1.img.xz. Una vez obtenido el sistema operativo, se debe *flashear* en la tarjeta MicroSD, de forma que RaspBerry Pi reconozca dicha tarjeta como *booteable*, arrancando el sistema operativo desde ella. Para realizar esta tarea, se emplea el software *Balena Etcher*, disponible en https://etcher.balena.io/. En primer lugar se selecciona la imagen del sistema operativo de Home Assistant.

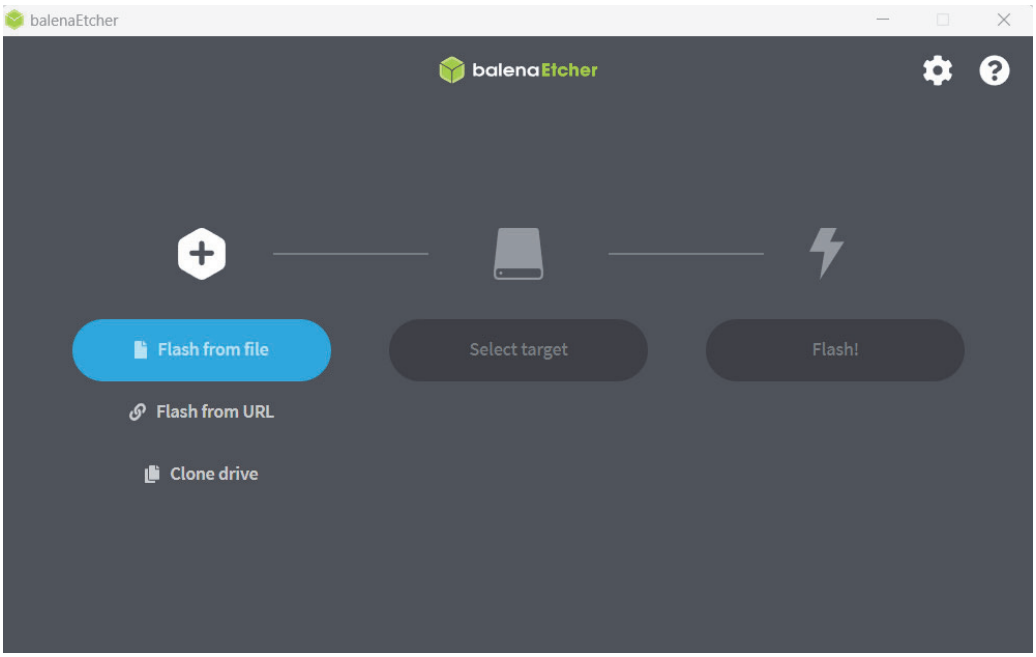



**Figura A.1:** Balena Etcher. Selección de imagen.

Posteriormente, se debe introducir la tarjeta MicroSD con el adaptador en alguno de los puertos USB del PC, y seleccionarla desde el asistente de configuración de *Balena Etcher*.

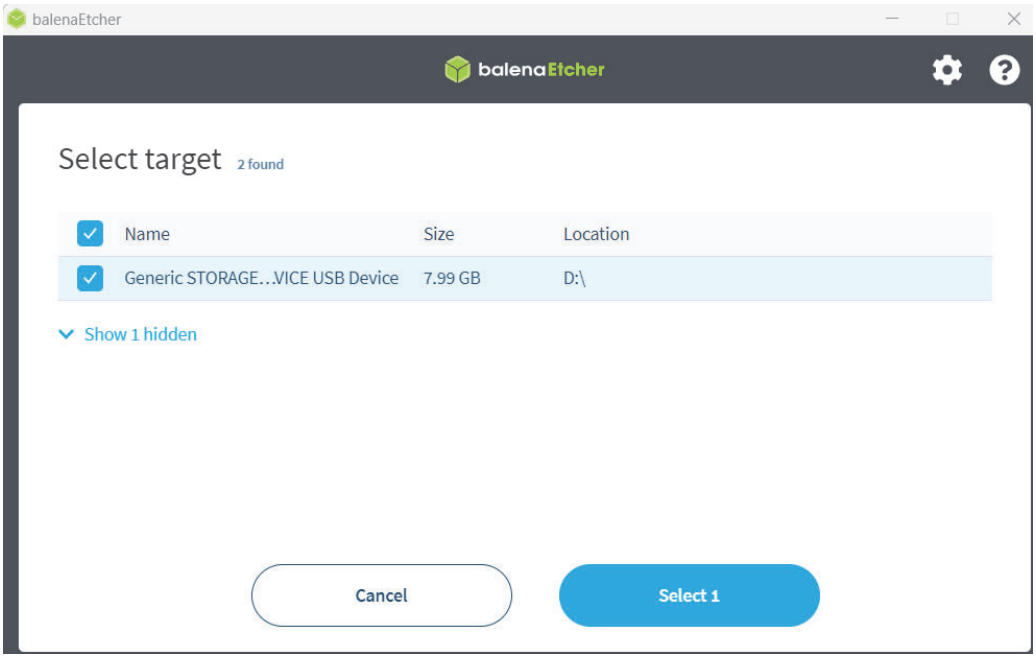



**Figura A.2:** Balena Etcher. Selección de dispositivo.

A continuación, se debe hacer click en la opción *Flash* y esperar que se complete el proceso.

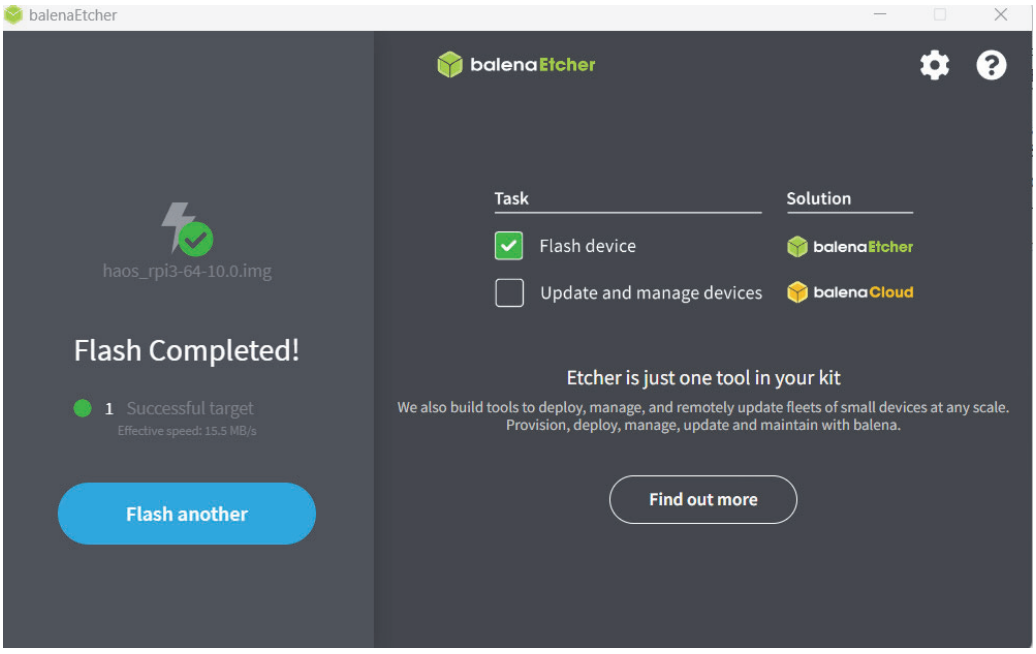



**Figura A.3:** Balena Etcher. Flash completado.

Una vez completado, se debe introducir la SD en la ranura de RaspBerry Pi 3B y, aún sin conectar la alimentación, conectar la RaspBerry Pi 3B a internet mediante un cable Ethernet. La primera configuración de Home Assistant requiere de conexión por cable, aunque una vez configurado es posible establecer conexión a cualquier red WiFi disponible.

**Figura A.4:** Primera conexión de RaspBerry Pi 3B

Desde un PC conectado a la misma red se puede acceder desde el navegador una vez conectada la RaspBerry Pi 3B a la interfaz de usuario de Home Assistant, que en este caso mostrará el asistente de primera configuración. La interfaz de usuario para operar con Home Assistant estará disponible en la dirección:

http://homeassistant.local:8123

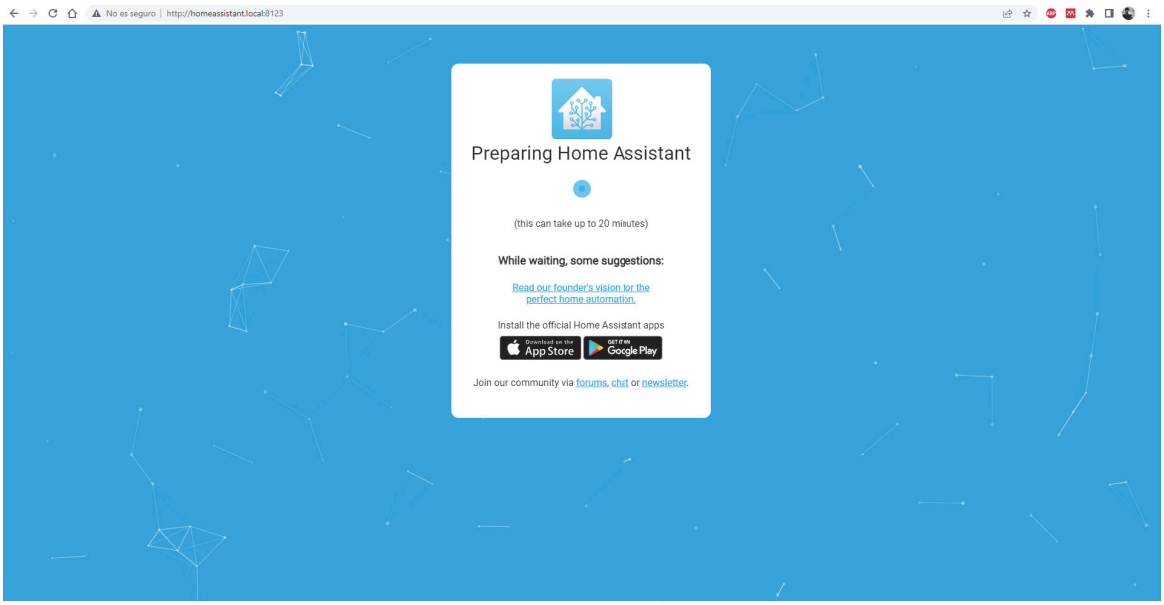



**Figura A.5:** Home Assistant. Primera ejecución.

El asistente de configuración permite establecer de forma intuitiva algunos parámetros básicos que el sistema necesita, como la ubicación, contraseña, nombre de usuario, entre otras. Es importante seleccionar la integración de Bluetooth cuando el asistente pregunte si se desea instalar alguna integración.

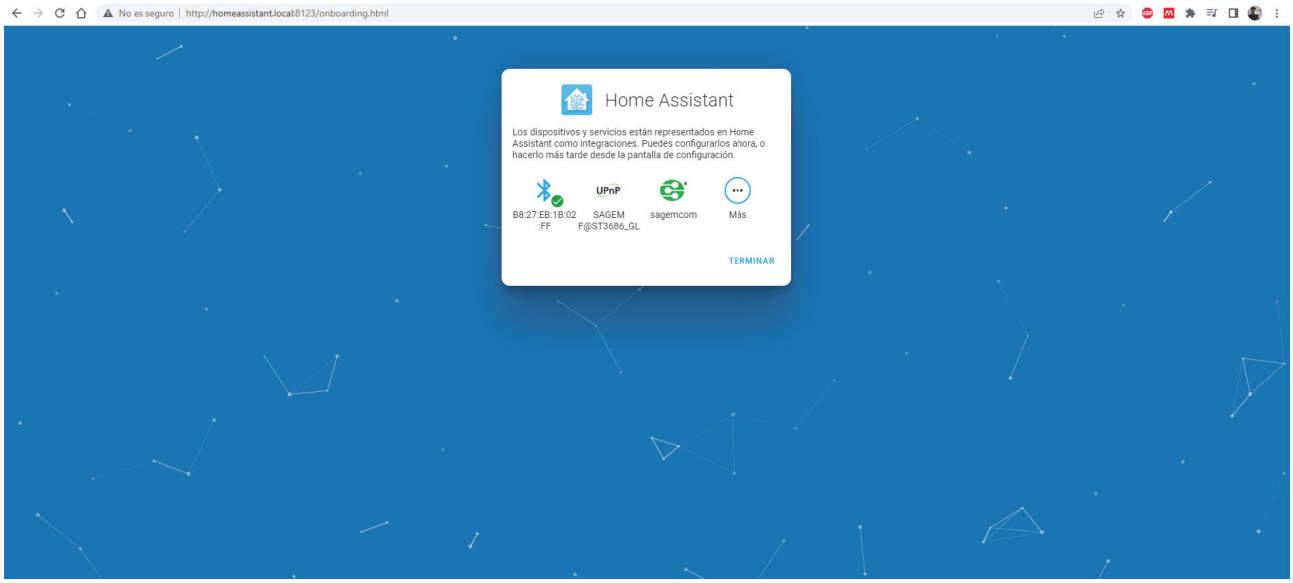



**Figura A.6:** Home Assistant. Dispositivos y servicios detectados automáticamente durante la instalación

Tras finalizar la configuración se ofrece la interfaz de usuario general de Home Assistant.

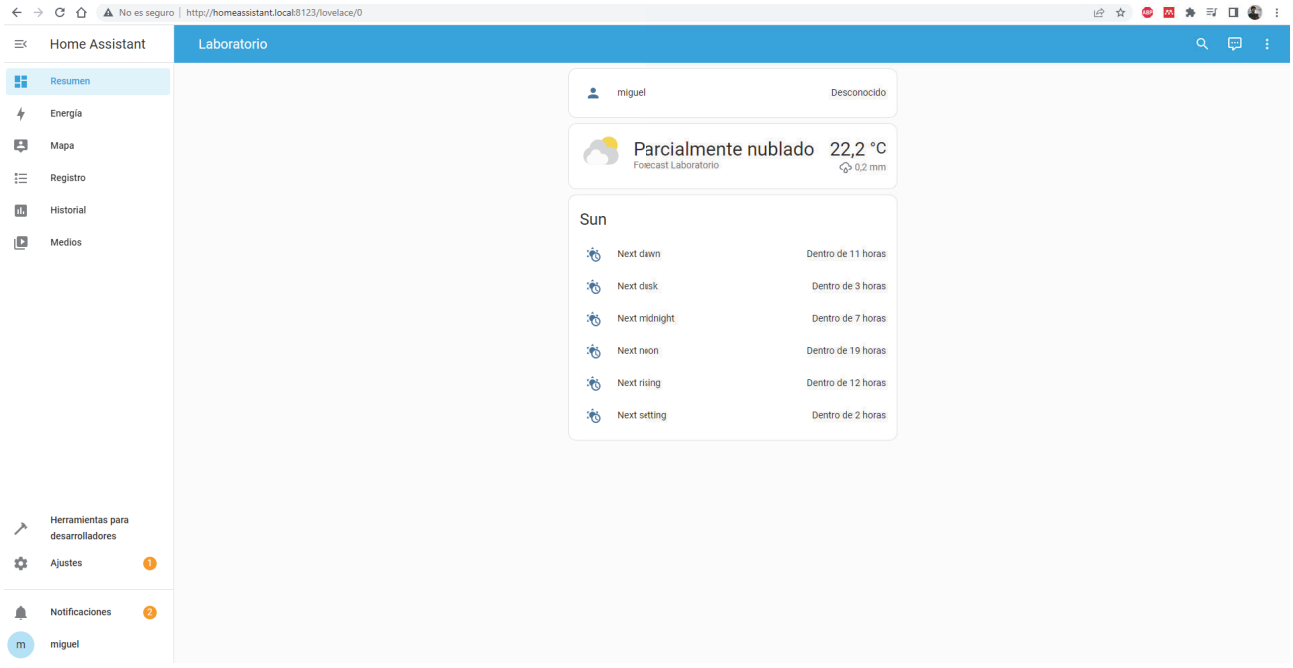



**Figura A.7:** Home Assistant. Interfaz de usuario de Home Assistant OS

## A.2 Instalación de complementos necesarios

A continuación se detalla el proceso de obtención del software requerido para el proyecto.

En primer lugar se debe descargar *File Editor*. Para obtener este y otros complementos se debe ir a *Ajustes ->Complementos ->Tienda de Complementos* y buscar el nombre del complemento deseado, en este caso *File Editor*.

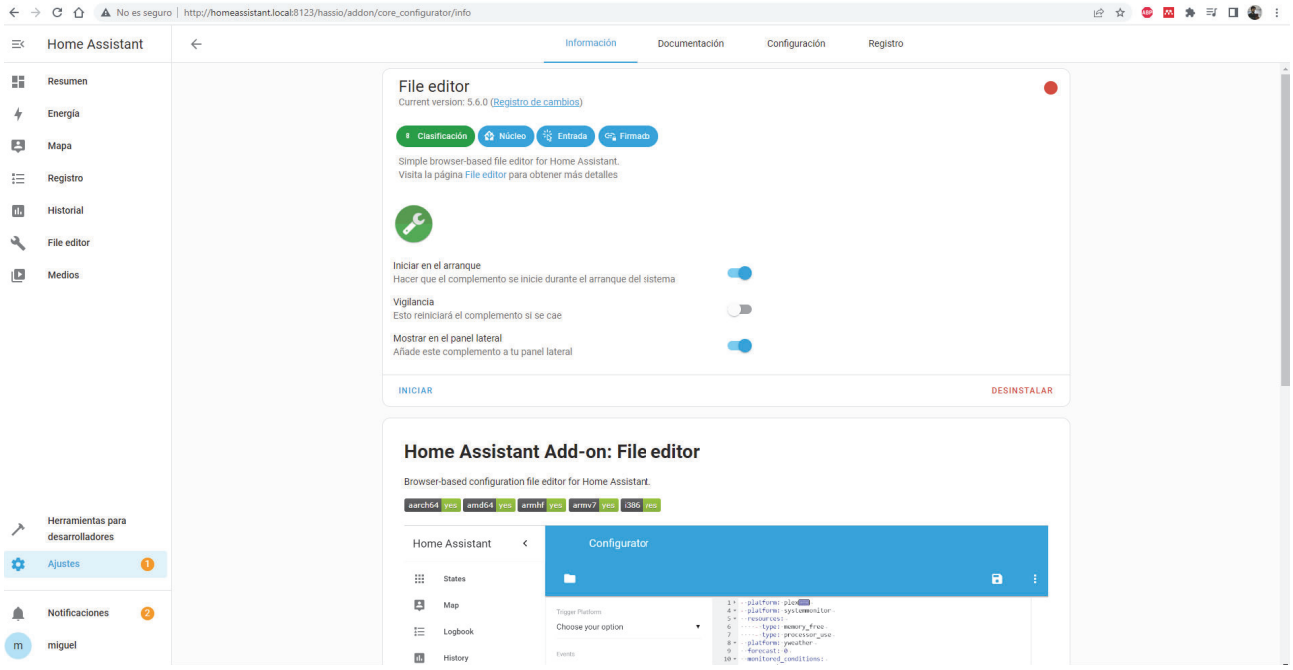



**Figura A.8:** Home Assistant. Instalación del editor de texto File Editor

Este complemento permite navegar entre los archivos de Home Assistant y tanto acceder como editar los ficheros de configuración que se desee.

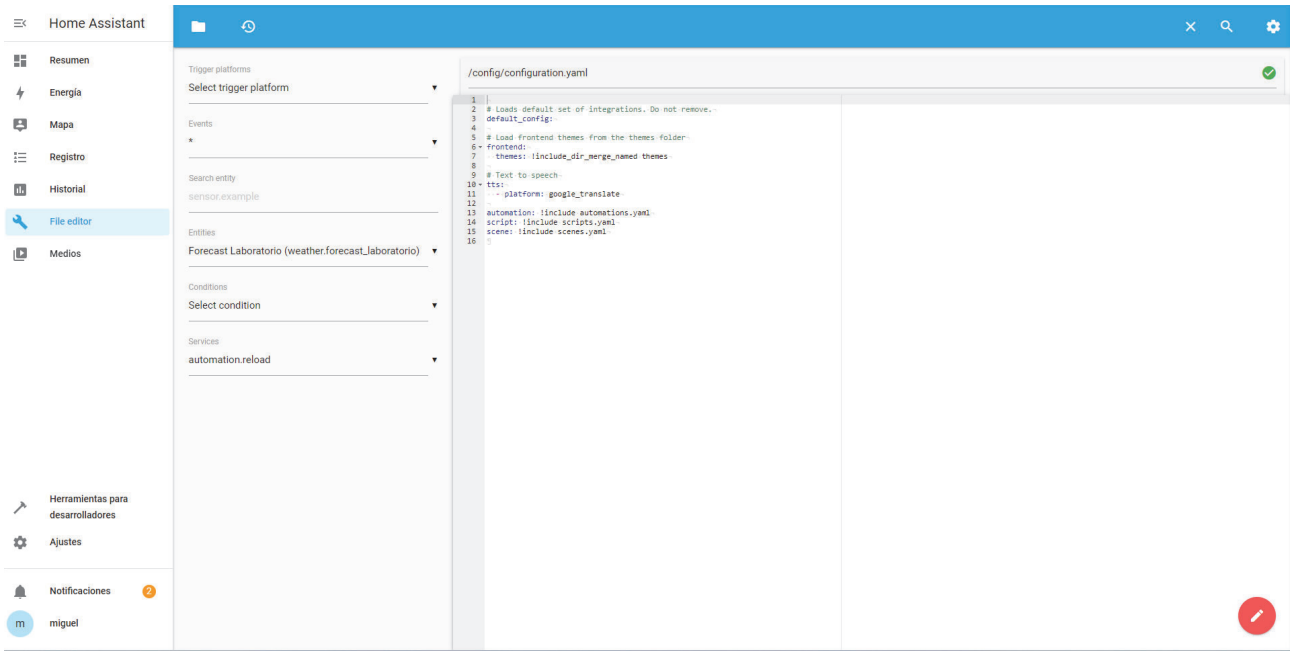



**Figura A.9:** Home Assistant. Fichero *configuration.yaml* editable desde File Editor

A continuación se instala siguiendo el mismo procedimiento *Mosquitto Broker*, que permite integrar en el mismo dispositivo que ejecuta Home Assistant un broker MQTT. Si bien las características del hardware lo hacen algo limitado y la ejecución de la plataforma de Mosquitto solicita (en estos términos) bastantes recursos, la carga que sufre el sistema debido a esta ejecución es perfectamente asumible. Este costo computacional es más tolerable si cabe al contemplar la posibilidad de aumentar las prestaciones hardware de las que disponemos, hecho que deriva inevitablemente en un incremento considerable del consumo eléctrico.

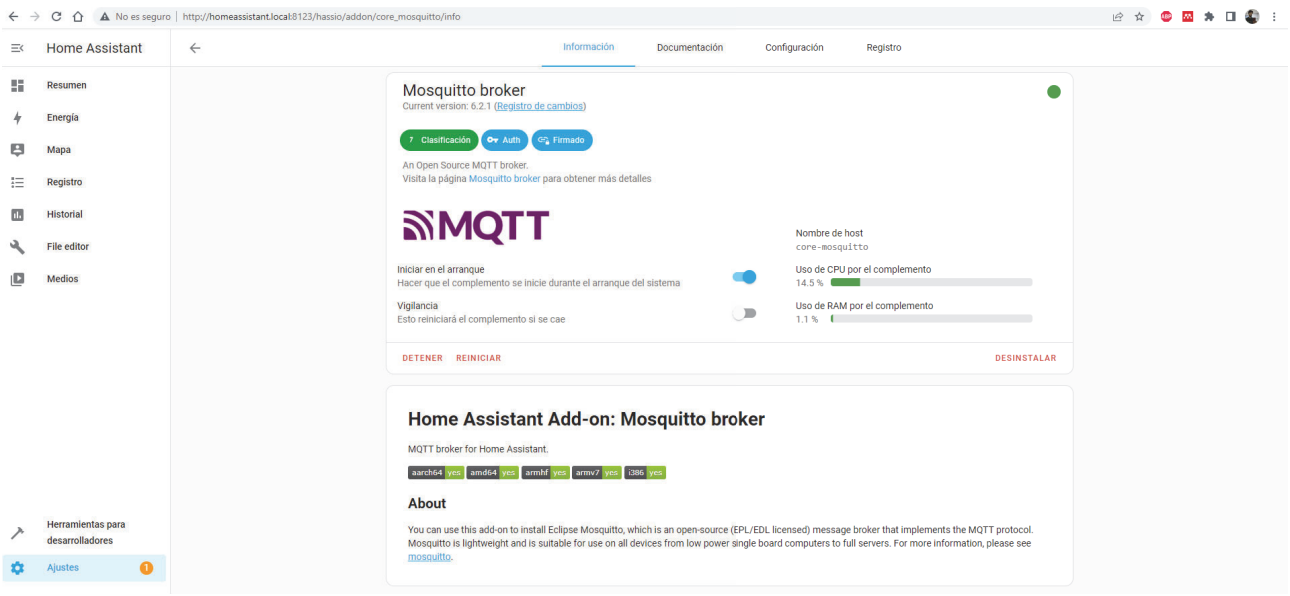



**Figura A.10:** Home Assistant. Instalación de Mosquitto Broker

A continuación se debe crear un usuario para MQTT. Estas credenciales serán utilizadas desde otros terminales para establecer la conexión con Mosquitto Broker.

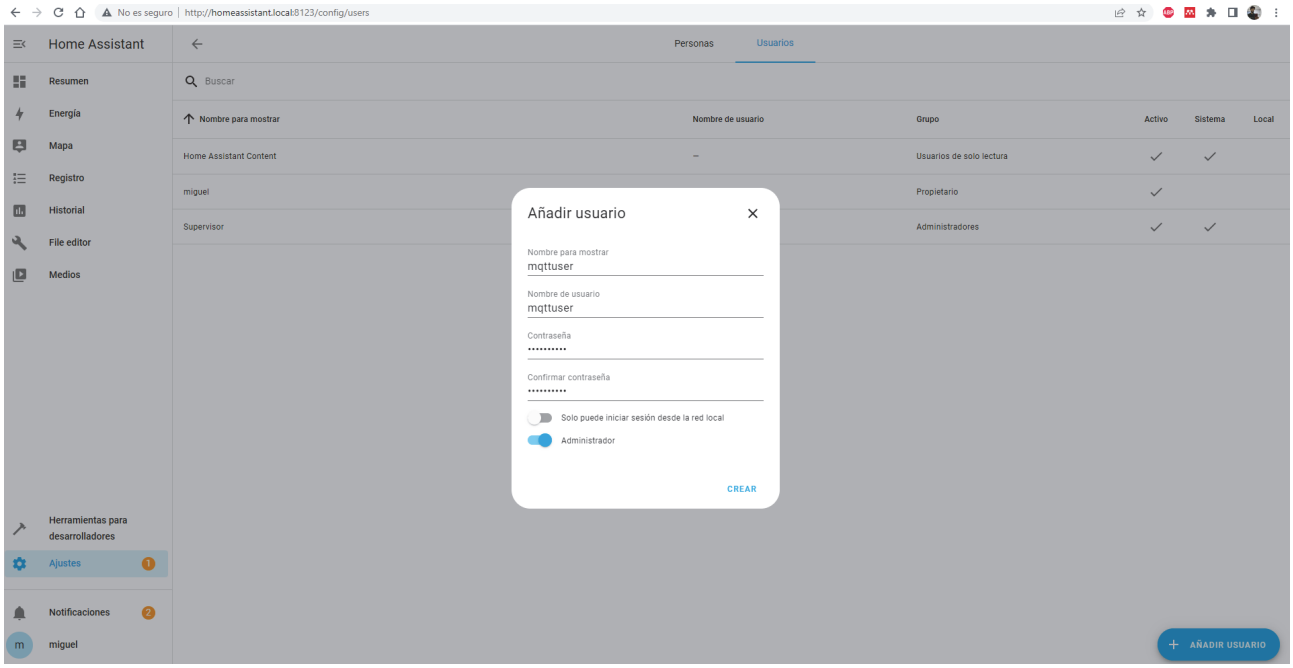



**Figura A.11:** Home Assistant. Creación de un usuario para MQTT

Por último, se debe añadir este usuario al listado de *logins* de la configuración de Mosquitto Broker

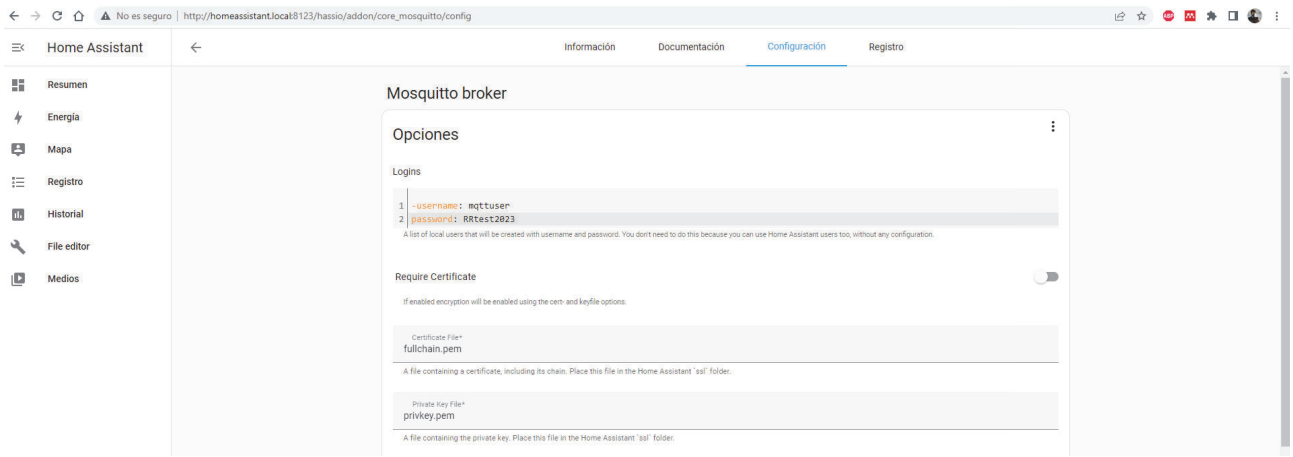



**Figura A.12:** Home Assistant. Adición de las credenciales del usuario MQTT a la lista *logins* del plugin Mosquitto Broker

# Apéndice B

# Configuración del Servidor Local

El servidor que aloja la página web Wordpress del programa RoboRescue se ejecuta en una máquina virtual en los ordenadores de la UMA destinados a esta finalidad. Esta máquina ejecuta CentOS 7 y Wordpress sobre MariaDB. Si bien existen backups de la máquina virtual, el correcto funcionamiento de la página web está sujeto a la integridad del sistema, por lo que, por precaución, es pertinente recrear las condiciones a nivel local para poder realizar pruebas y verificaciones sin riesgo, exportando posteriormente al sistema real las conclusiones obtenidas de este periodo de prueba.

## B.1 Instalación del sistema operativo CentOS 7

Se utiliza el software de visualización *Oracle Virtual Box* para la creación de la máquina virtual en la que se instalará CentOS 7. La imagen del sistema operativo puede obtenerse desde el enlace: http://isoredirect.centos.org/centos/7/isos/x86_64/.

Una vez descargada se debe configurar una nueva máquina virtual en Virtual Box, de tipo Linux, para software de Red Hat 64 bits, a la que se deben asignar recursos suficientes para ejecutar dicho sistema operativo.

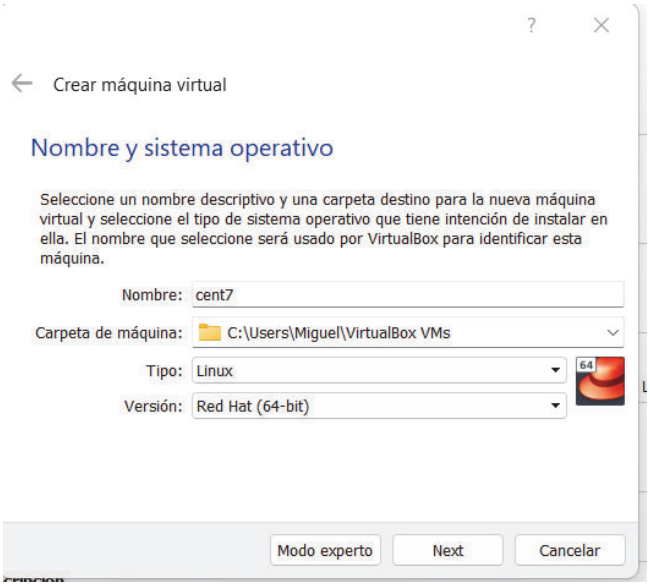



**Figura B.1:** Virtual Box. Asistente de creación de una nueva máquina.

Es importante modificar desde los ajustes de red la conexión de Red, de modo que quede habilitada la opción de Cable Conectado. Las especificaciones elegidas para la máquina se detallan a continuación.

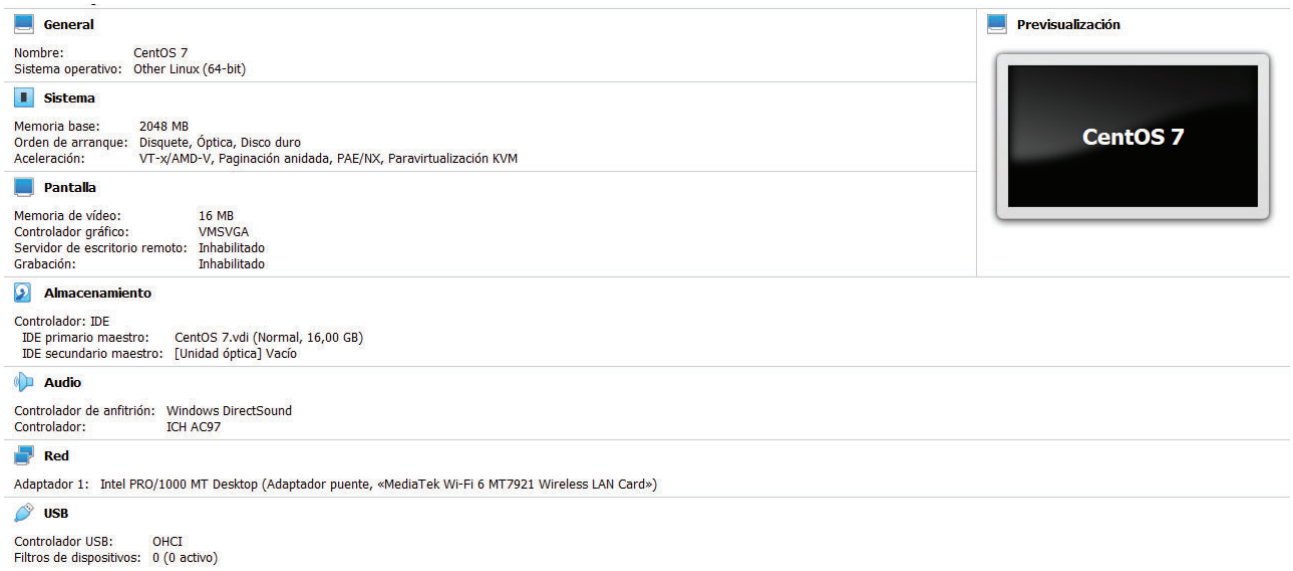



**Figura B.2:** Virtual Box. Especificaciones de la máquina virtual.

Una vez configurada la máquina, en su primer inicio permitirá elegir la imagen ISO del sistema operativo que se ha descargado con anterioridad.

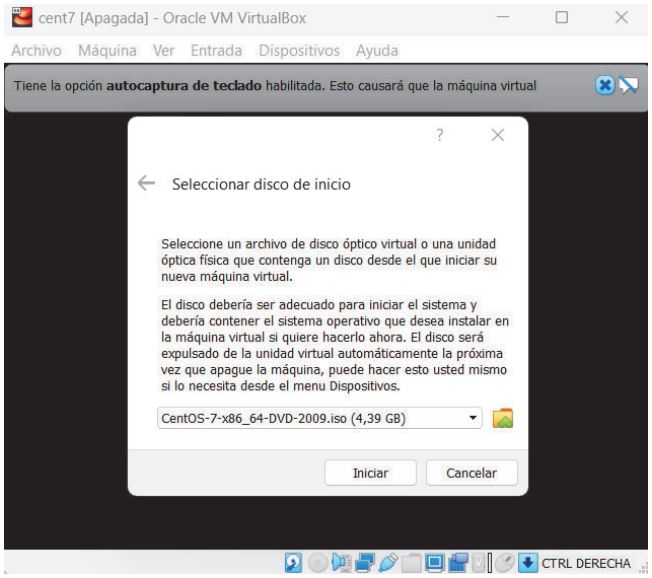



**Figura B.3:** Virtual Box. Selección de imagen ISO

## B.2 Configuración de CentOS 7

Una vez instalado y ejecutando el sistema operativo se sigue el procedimiento expuesto a continuación para su configuración

### B.2.1 Configuración de Red

La conexión a la red del sistema debe ser realizada de forma manual. Se accede a la interfaz de configuración red de CentOS 7 mediante el comando

```
1 $ nmtui
```

Desde esta interfaz se debe activar la conexión de red. Un asterisco sobre el nombre del adaptador de red indica que se ha activado la conexión.

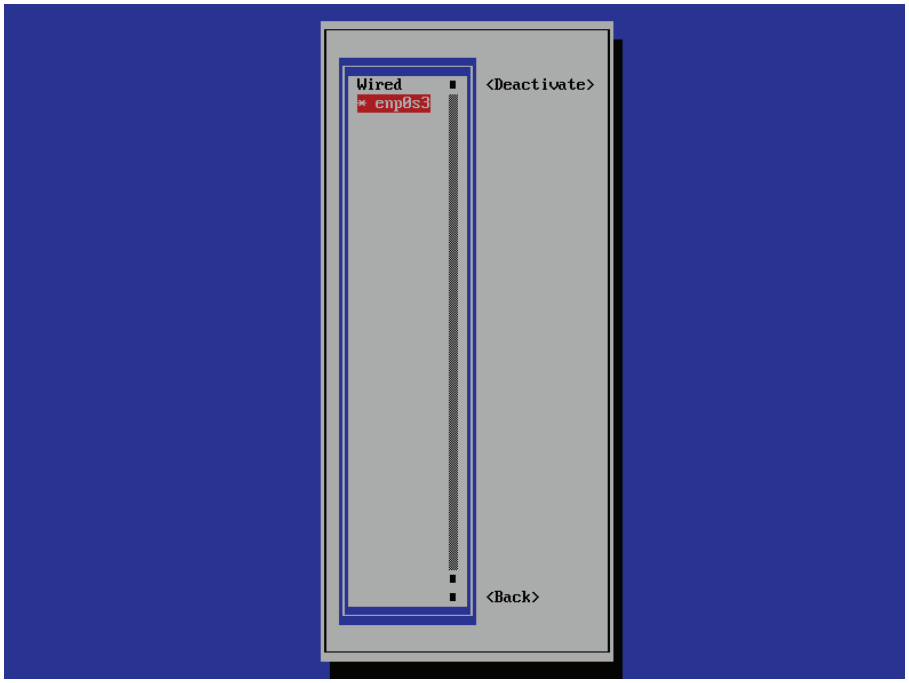



**Figura B.4:** CentOS 7. Activar Conexión

Ahora, una vez teniendo red, es necesario fijar la dirección IP. La dirección IP de la máquina es por defecto IPV4 dinámica, por lo que debería consultarse cada vez que se quiera establecer una conexión. Para evitar esto, forzaremos la opción de IP fija, editando los ajustes de conexión ethernet para especificar que se establecerá esta IP de forma manual.

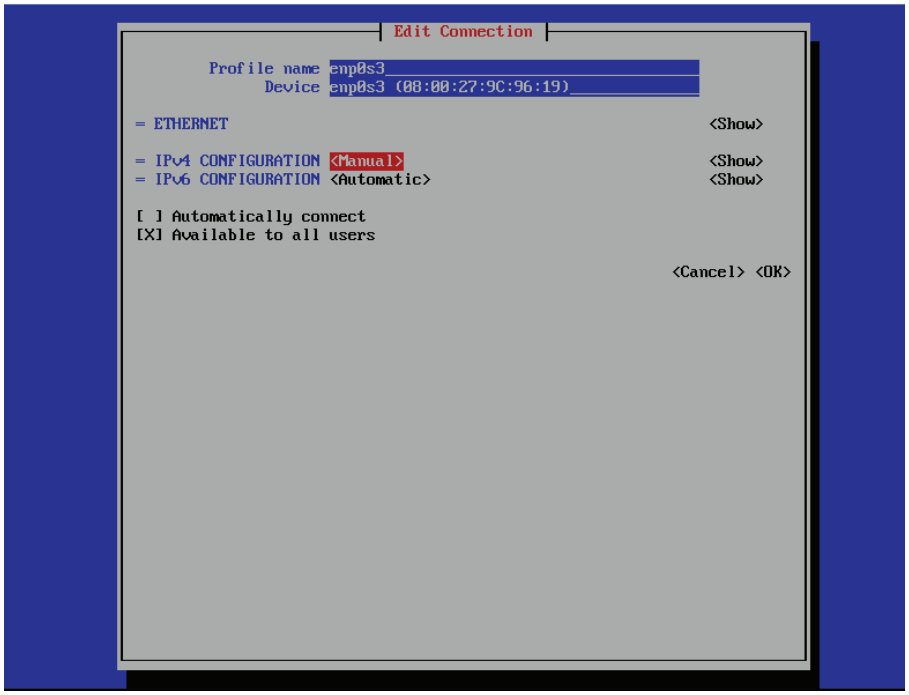



**Figura B.5:** CentOS 7. Fijar IPv4 a manual

Para asignarle una dirección IP a la máquina con CentOS 7 se debe conocer qué direcciones están disponibles en la distribución de red de la máquina fija sobre la que se ejecuta. Es posible acceder al servidor DHCP para comprobar este listado de direcciones disponibles introduciendo en cualquier navegador la dirección IP asociada a este servidor. Se deben conocer las credenciales de del router para poder realizar esta consulta. Estas credenciales se encuentran habitualmente en el lateral o parte trasera de los routers comerciales.

Se puede consultar desde la consola de comandos de Windows la dirección del servidor DHCP mediante el comando *ipconfig -all*

```
Adaptador de LAN inalámbrica Wi-Fi:

   Sufijo DNS específico para la conexión. . :
   Descripción . . . . . . . . . . . . . . . : MediaTek Wi-Fi 6 MT7921 Wireless LAN Card
   Dirección física. . . . . . . . . . . . . : [redacted]
   DHCP habilitado . . . . . . . . . . . . . : sí
   Configuración automática habilitada . . . : sí
   Vínculo: dirección IPv6 local. . . : fe80::fcae:294d:f28f:ba4%18(Preferido)
   Dirección IPv4. . . . . . . . . . . . . . : 192.168.0.14(Preferido)
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Concesión obtenida. . . . . . . . . . . . : jueves, 4 de mayo de 2023 12:21:15
   La concesión expira . . . . . . . . . . . : viernes, 5 de mayo de 2023 12:21:14
   Puerta de enlace predeterminada . . . . . : 192.168.0.1
   Servidor DHCP . . . . . . . . . . . . . . : 192.168.0.1
   IAID DHCPv6 . . . . . . . . . . . . . . . : [redacted]
   DUID de cliente DHCPv6. . . . . . . . . . : [redacted]
   Servidores DNS. . . . . . . . . . . . . . : 212.166.132.110
                                       212.166.132.104
   NetBIOS sobre TCP/IP. . . . . . . . . . . : habilitado
```

**Figura B.6:** Windows. Direción del servidor DHCP

Si bien cada distribuidor de red proporciona una interfaz diferente es sencillo localizar en cualquier caso las direcciones IP disponibles. Se debe elegir una de entre las direcciones libres para ser asignada a la máquina CentOS 7.


**Figura B.7:** Interfaz de usuario del Router. Direcciones IP disponibles

Se elige una dirección comprendida entre los valores mínimo y máximo. En este caso *192.168.0.12* y se asigna desde el menú de configuración de red de CentOS 7.

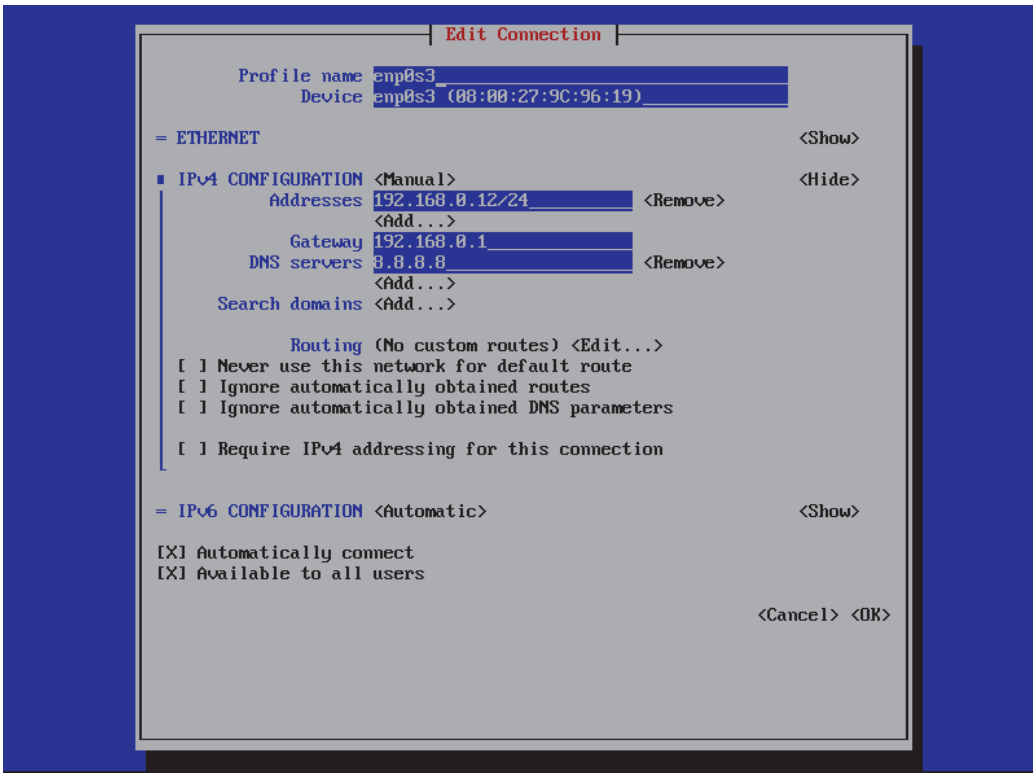



**Figura B.8:** CentOS 7. Establecimiento de IP fija

## B.3 Instalación de software necesario

Tras finalizar la configuración del sistema operativo CentOS 7 se debe realizar la instalación y configuración del software que permitirá realizar las funcionalidades requeridas del sistema: Apache, MariaDB, PHP, phpMyAdmin, Node-RED y WordPress.

### B.3.1 Instalación de Apache

Una vez configurada la conexión a la red de la máquina virtual que ejecuta CentOS 7, se debe configurar para hacerla funcionar como servidor web. Con este fin se instala en la máquina el servidor de Apache. Se deben ejecutar los siguientes comandos. En primer lugar se actualizan los paquetes de CentOS 7. Posteriormente se instala y se habilita la opción de inicio en el arranque de Apache.

```
1 $ sudo yum update
2 $ sudo yum install httpd
3 $ sudo systemctl enable httpd
```

Una vez instalado Apache se debe añadir una regla al firewall para que permita las conexiones *http* y *https*.

```
1 $ sudo firewall-cmd --permanent --add-service=http
2 $ sudo firewall-cmd --permanent --add-service=https
```

Se puede verificar la correcta instalación de todos los servicios accediendo a la dirección del servidor desde un navegador en la máquina local. Apache mostrará una página de prueba que indica el estado de correcto funcionamiento.

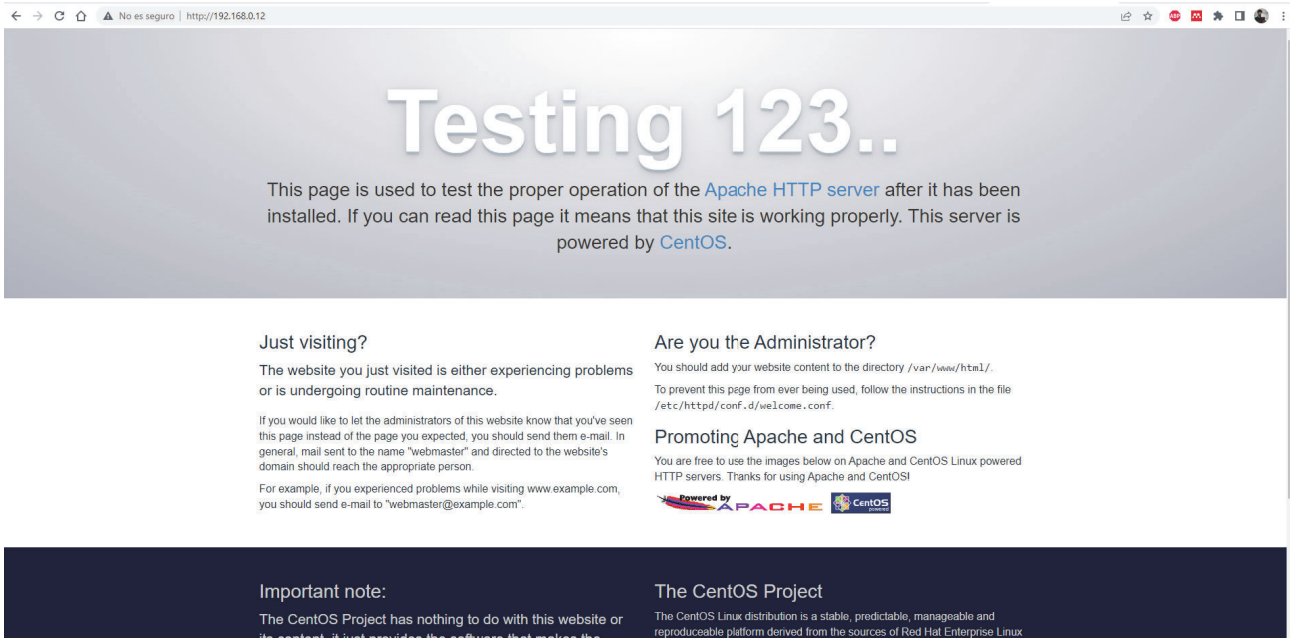



**Figura B.9:** Web. Página de prueba por defecto indicando la correcta instalación de Apache

## B.3.2 Instalación de PHP

En primer lugar se añaden los repositorios de *EPEL* y *remi-repo*

```
1 $ sudo yum install epel-release
2 $ sudo yum install http://rpms.remirepo.net/enterprise/remi-release-7.rpm
```

Se habilita la instalación de la versión *PHP 7.4* y posteriormente se instalan tanto PHP como algunas dependencias necesarias para compatibilizarlo con el resto de aplicaciones.

```
1 $ sudo yum-config-manager --enable remiphp74
2 $ sudo yum install php php-common php-opcache php-mcrypt php-cli php-gd ...
    php-curl php-mysql -y
```

Se puede verificar el correcto funcionamiento de PHP añadiendo un ejecutable PHP de prueba en la ruta */var/www/html/* de la máquina CentOS, en el que se lame a la función *phpinfo()*. Esta función muestra información básica acerca del estado de la instalación de PHP. Accediendo a esta dirección desde el navegador de la máquina local introduciendo *dirección_IP/testphp.php* se verifica la correcta instalación.


**Figura B.10:** Web. Ruta del archivo de prueba PHP

### B.3.3 Instalación de MariaDB y phpMyAdmin

Para instalar MariaDB primero ejecutamos las siguientes líneas desde el terminal para obtener los repositorios necesarios.

```
1 $ sudo yum install wget
2 $ wget https://downloads.mariadb.com/MariaDB/mariadb_repo_setup
3 $ chmod +x mariadb_repo_setup
4 $ sudo ./mariadb_repo_setup
```

A continuación se debe instalar y habilitar en el inicio MariaDB.

```
1 $ sudo yum install MariaDB-server -y
2 $ sudo systemctl enable mariadb.service
```

Posteriormente se configuran las características y credenciales de acceso a MariaDB mediante.

```
1 $ sudo mariadb-secure-installation
```

Para instalar la herramienta phpMyAdmin, que permite una interacción mucho más amigable con MariaDB se deben ejecutar las siguientes líneas

```
1 $ sudo yum install phpmyadmin
```

Como el acceso a phpMyAdmin está restringido por defecto sólo a la IP local y se desea acceder desde el PC en el que se está ejecutando la máquina virtual debemos modificar el fichero de configuración de phpMyAdmin. Esto se puede hacer abriendo directamente el archivo desde la ruta con el editor de texto mediante el comando *vi*.

```
1 $ sudo vi /etc/httpd/conf.d/phpMyAdmin.conf
```

Las modificaciones se muestran resaltadas en un recuadro rojo, donde se ha sustituido la IP correspondiente a la máquina local *127.0.0.1* por *all granted* y All para permitir el acceso externo so autenticación según las credenciales generadas anteriormente.

```
#
# Allows only localhost by default
#
# But allowing phpMyAdmin to anyone other than localhost should be considered
# dangerous unless properly secured by SSL

Alias /phpMyAdmin /usr/share/phpMyAdmin
Alias /phpmyadmin /usr/share/phpMyAdmin

<Directory /usr/share/phpMyAdmin/>
   AddDefaultCharset UTF-8

   <IfModule mod_authz_core.c>
     # Apache 2.4
     <RequireAny>
       Require all granted
       Require ip ::1
     </RequireAny>
   </IfModule>
   <IfModule !mod_authz_core.c>
     # Apache 2.2
     Order Deny,Allow
     Deny from All
     Allow from All
     Allow from ::1
   </IfModule>
</Directory>

<Directory /usr/share/phpMyAdmin/setup/>
   <IfModule mod_authz_core.c>
     # Apache 2.4
     <RequireAny>
       Require all granted
       Require ip ::1
     </RequireAny>
"phpMyAdmin.conf" 96L, 2239C written
```

**Figura B.11:** CentOS 7. Modificaciones en la configuración de phpMyAdmin

## B.3.4 Instalación de WordPress

Previo a la instalación de wordpress, se debe crear una base de datos que lo contenga desde la herramienta phpMyAdmin.


**Figura B.12:** Web. Creación de la base de datos de WordPress desde phpMyAdmin

La versión de wordpress que se instale en la máquina virtual debe ser la misma que la que se está ejecutando en el servidor de la UMA. De este modo se asegura la compatibilidad con el sistema real de todos los plugins y configuraciones realizadas. Debido a que el asistente de instalación de WordPress provee un enlace que descarga la última versión disponible, se debe descargar y transferir de forma manual la versión 5.0.4 del repositorio de versiones de wordpress.

Tras descargar el archivo comprimido en la máquina física se puede usar un cliente SSH FTP como *bitvise* para transferir los archivos a la máquina CentOS 7. Se debe habilitar este servicio desde CentOS 7 instalando el siguiente paquete.

```
1 $ sudo yum install vsftpd ftp -y
```

Tras configurar el acceso al puerto 22 del servidor se transfiere el comprimido que contiene WordPress.

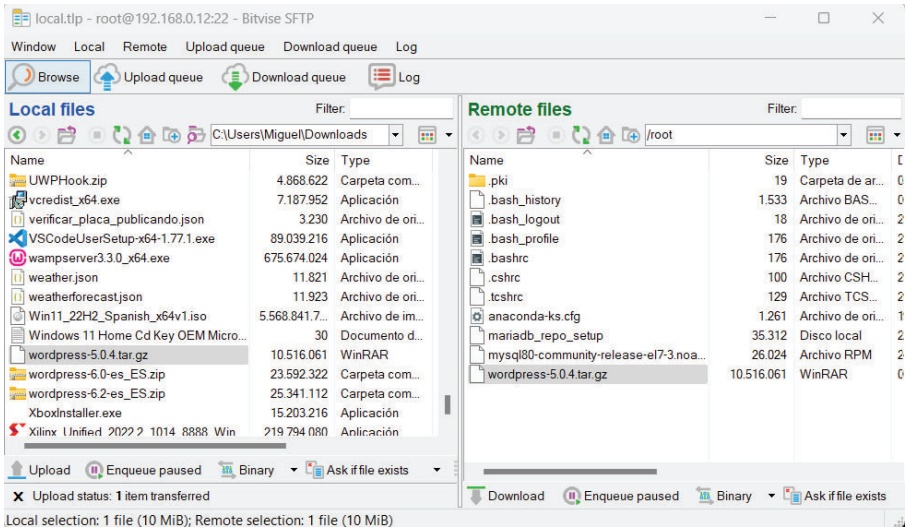



**Figura B.13:** Bitvise. Transferencia del comprimido con WordPress 5.0.4 a CentOS 7

Se debe descomprimir el fichero *wordpress-5.0.4.tar.gz* en la ruta */var/www/html/*. A continuación se puede iniciar el asistente de configuración de WordPress desde la ruta *ip_servidor/wordpress*.

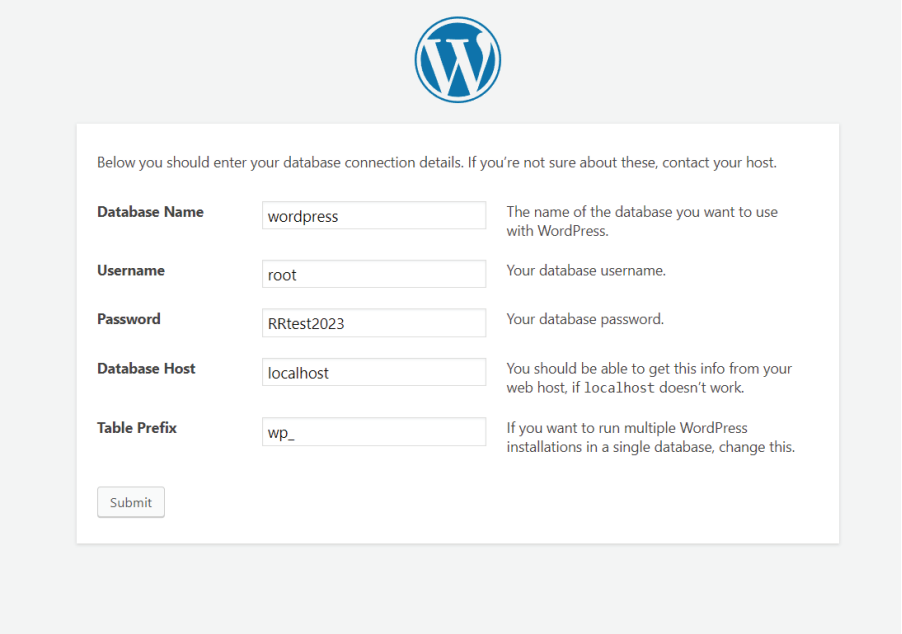



**Figura B.14:** Web. Asistente de configuración de Wordpress

Desde este momento WordPress es accesible desde *ip_servidor/wordpress*.

### B.3.5 Instalación de Node-RED

Para la instalación de Node-RED es necesario instalar antes *Node.js*. Desde el repositorio GitHub de Node-RED se proporcionan herramientas para descargar e instalar tanto *Node.js* como Node-RED en un único comando.

```
1 $ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers...
2         /master/rpm/update-nodejs-and-nodered)
```

Durante el proceso de instalación se debe añadir la regla al firewall que hace el puerto 1880. del servidor público. Esto es necesario para poder acceder a Node-RED desde la máquina local, ya que CentOS 7 no tiene interfaz de usuario, por lo que no se puede lanzar desde el mismo un navegador web.

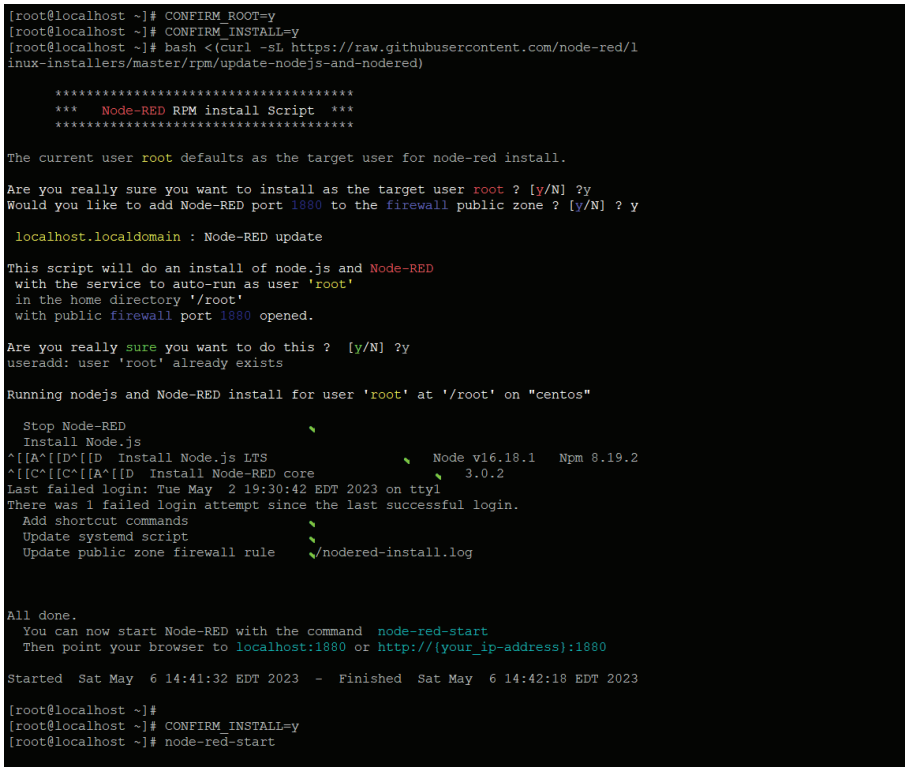



**Figura B.15:** CentOS 7. Detalle de la instalación de Node-RED

A partir de la instalación y tras iniciar Node-RED es posible acceder a la interfaz gráfica desde *ip_servidor:1880*

A continuación se debe proteger mediante credenciales el acceso a Node-RED. Para ello, Node-RED porporciona una herramienta que permite crear un hash para una contraseña elegida. Desde la ruta *.node-red*

```
1 $ node-red admin hash-pw
```

Tras introducir la contraseña deseada se generará el hash que podremos almacenar en el fichero de configuración de Node-RED, asegurando que no se almacenan contraseñas, incrementando la seguridad. Este hash será empleado para verificar que la contraseña introducida es correcta al iniciar sesión en Node-RED. Se debe abrir el fichero de configuración desde la ruta *.node-red* y añadir el hash de autenticación.


**Figura B.16:** CentOS 7. Introducción del hash generado en los archivos de configuración de Node-RED

Es necesario a continuación instalar las librerías que permiten la comunicación entre Node-RED y MariaDB. Desde el gestor de *palettes* de Node-RED se instala *node-red-node-mysql*. Para ello se accede desde el navegador en el ordenador local y se procede con la instalación.

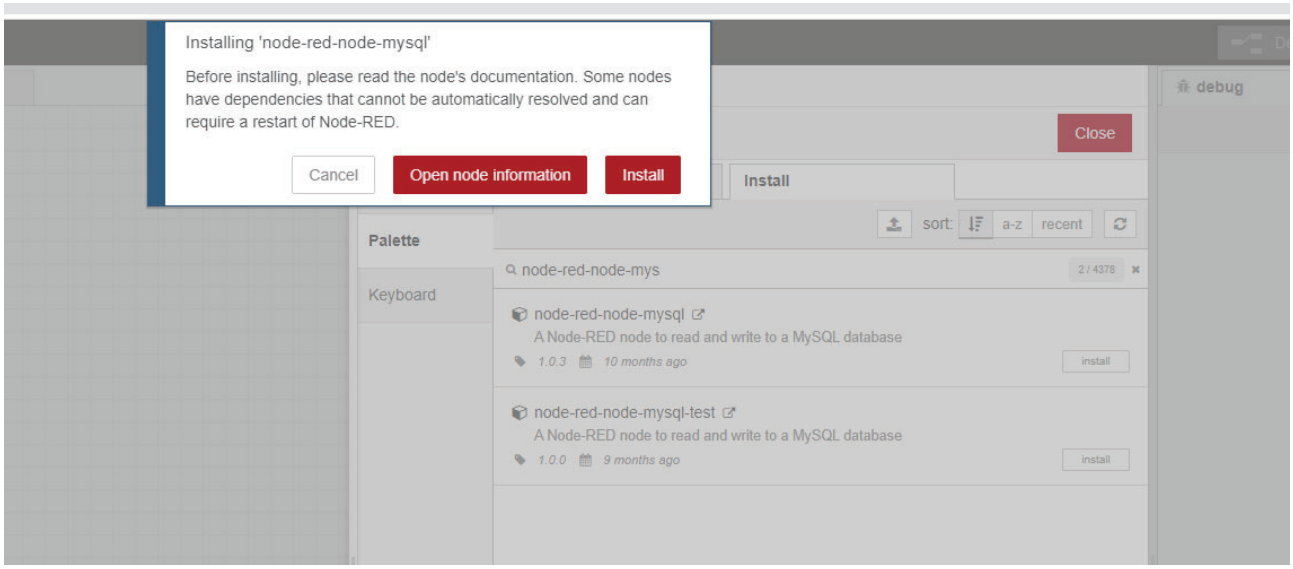



**Figura B.17:** Web. Instalación del paquete de nodos de comunicación con bases de datos en Node-RED

A continuación, desde phpMyAdmin, se crea la base de datos *genteLab*, en la que se almacenarán los datos recibidos en Node-RED junto con la tabla *horarios*, en la que se especifican los campos de los datos que se almacenarán.

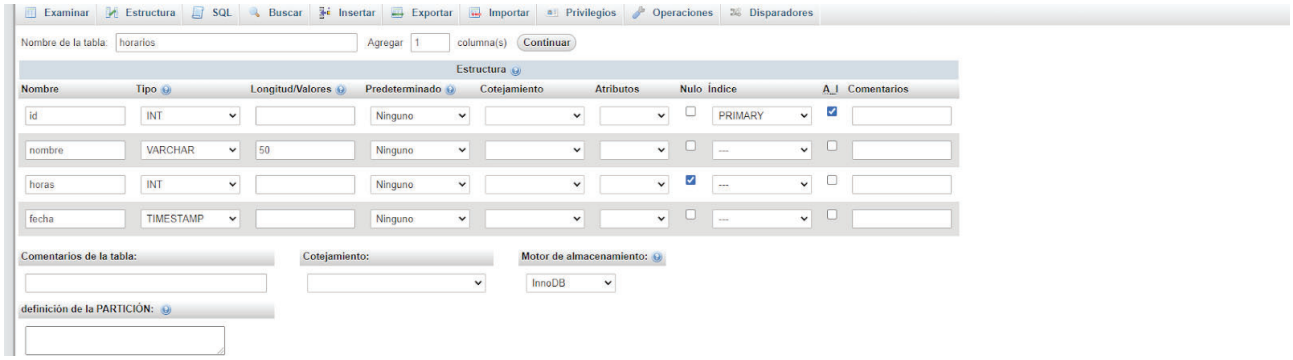



**Figura B.18:** Web. Tabla de la base de datos en la que se almacenarán los datos recibidos en Node-RED, creada desde phpMyAdmin

# Apéndice C

# Configuración de solución basada en Node-RED y Telegram

A lo largo de este apéndice se desarrollan los detalles de la implementación de la solución alternativa propuesta para exportar el listado de gente en el laboratorio en tiempo real desde Home Assistant, de forma que posteriormente puedan ser consultados desde la aplicación Telegram mediante la interacción con un bot.

## C.1 Creación del bot *RRgenteLABbot*

En primer lugar, se utiliza el asistente *Botfather* para la creación del bot que se va a utilizar desde Node-RED. Desde el chat con *Botfather* se pueden configurar los aspectos esenciales como nombre e identificación del bot que se desea crear (ver Figura C.1). Además, este asistente proporcionará el token de acceso a la HTTP API de Telegram, mediante el cual será posible interactuar desde aplicaciones externas (Node-RED) en la conversación con el bot.

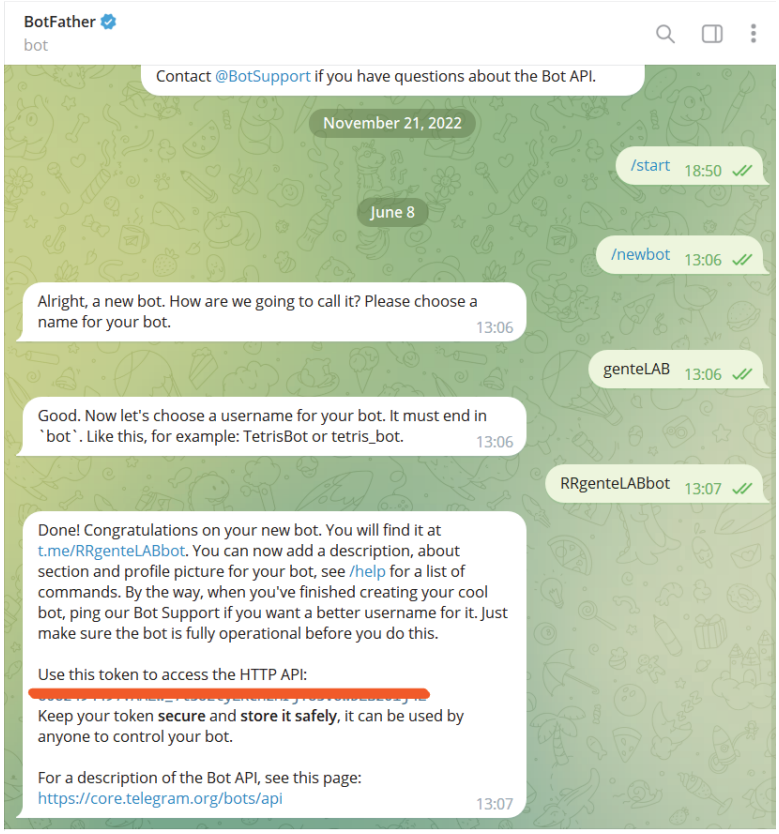



**Figura C.1:** Telegram Desktop. Creación de un nuevo bot mediante el asistente *Botfather*

Una vez creado, el bot es accesible desde la URL t.me/RRgenteLABbot

## C.2 Configuración de Home Assistant

Node-RED cuenta con una aplicación compatible con el ecosistema de HASS. Es posible encontrar esta aplicación en la *store* de *add-ons*, tal como se muestra en la Figura C.2

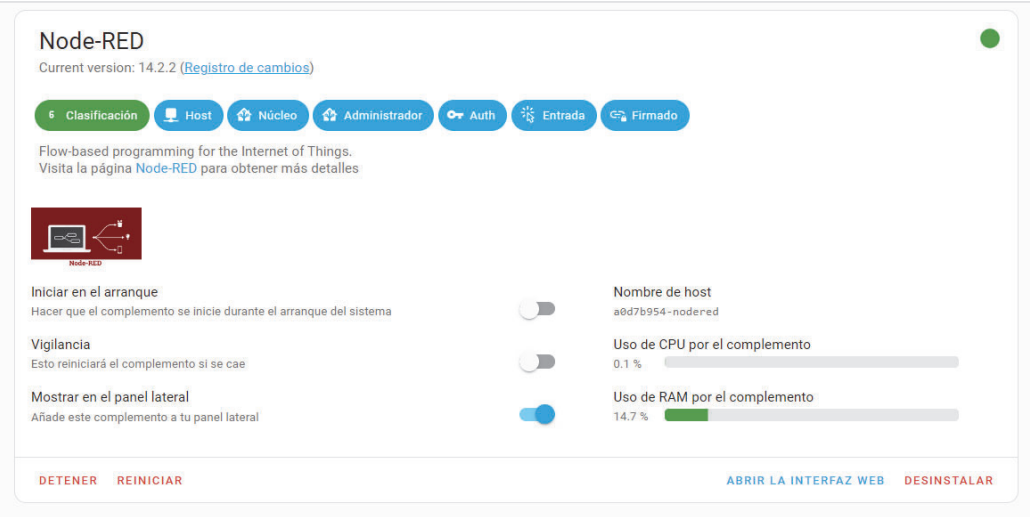



**Figura C.2:** Home Assistant. Instalación de Node-RED desde la tienda de complementos

A continuación, una vez habiendo accedido al *add-on* instalado, se debe obtener desde el gestor de *palettes* de Node-RED el paquete que añade nodos que permiten la compatibilidad con los bots de Telegram, *node-red-contrib-telegrambot* (ver Figura C.3).

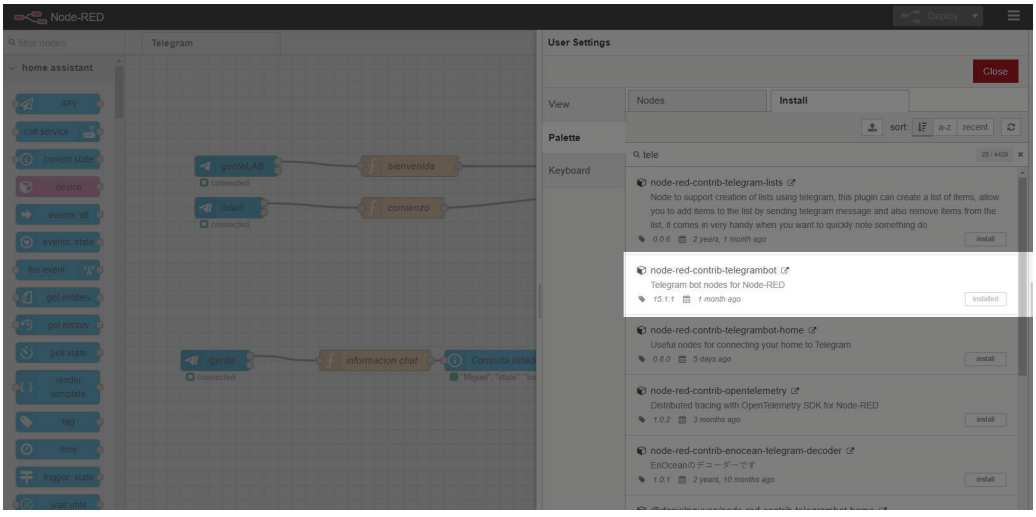



**Figura C.3:** Node-RED. Instalación del paquete de nodos *node-red-contrib-telegrambot*

A continuación, introduciendo al flujo el nodo *Telegram Receiver* se debe crear y configurar el perfil para el bot en Node-RED. Es necesario introducir tanto el nombre con el que se crea el bot como el token de acceso generado, tal como referencia la Figura C.1. A continuación se introducen estos parámetros en la configuración del nodo (ver Figura C.4).

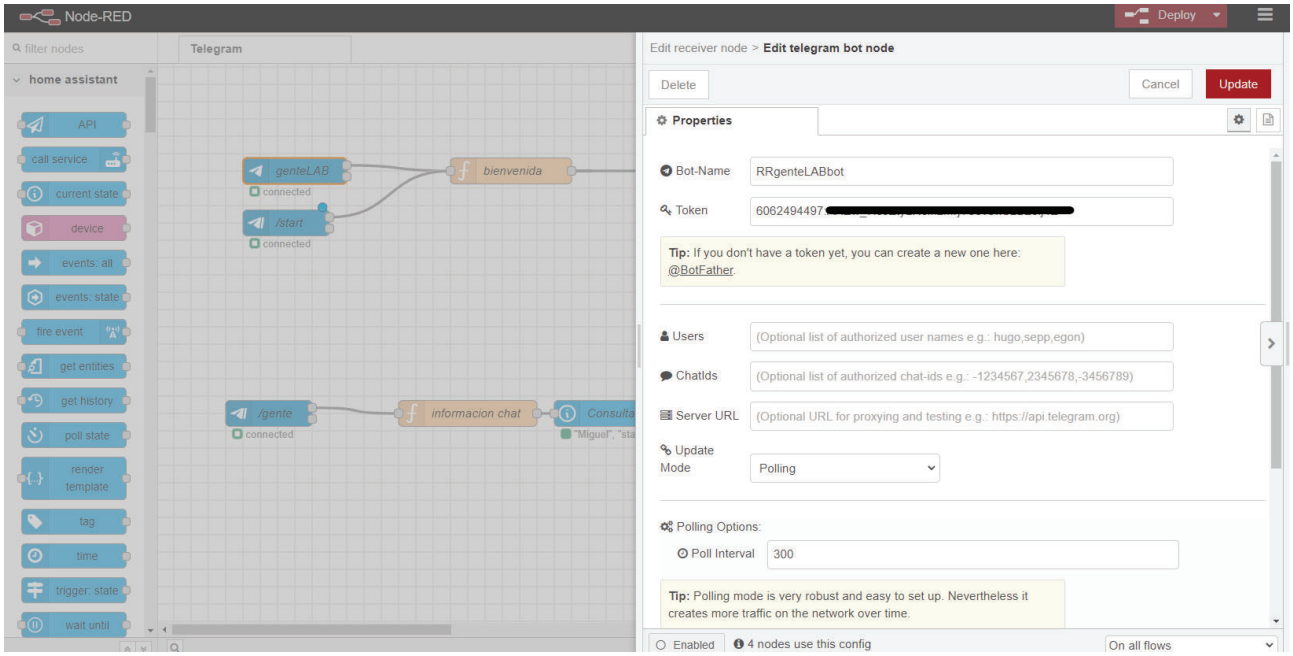



**Figura C.4:** Node-RED. Configuración del perfil del bot creado según la Figura C.1

Finalmente se programa el flujo propuesto en la Figura C.5. En él, tanto los comandos de */start* como cualquier otro texto que no sea un comando devuelven un mensaje de bienvenida que contiene los comandos existentes, en este caso */gente*. Estas líneas de código se muestran en los Códigos C.1, C.2.

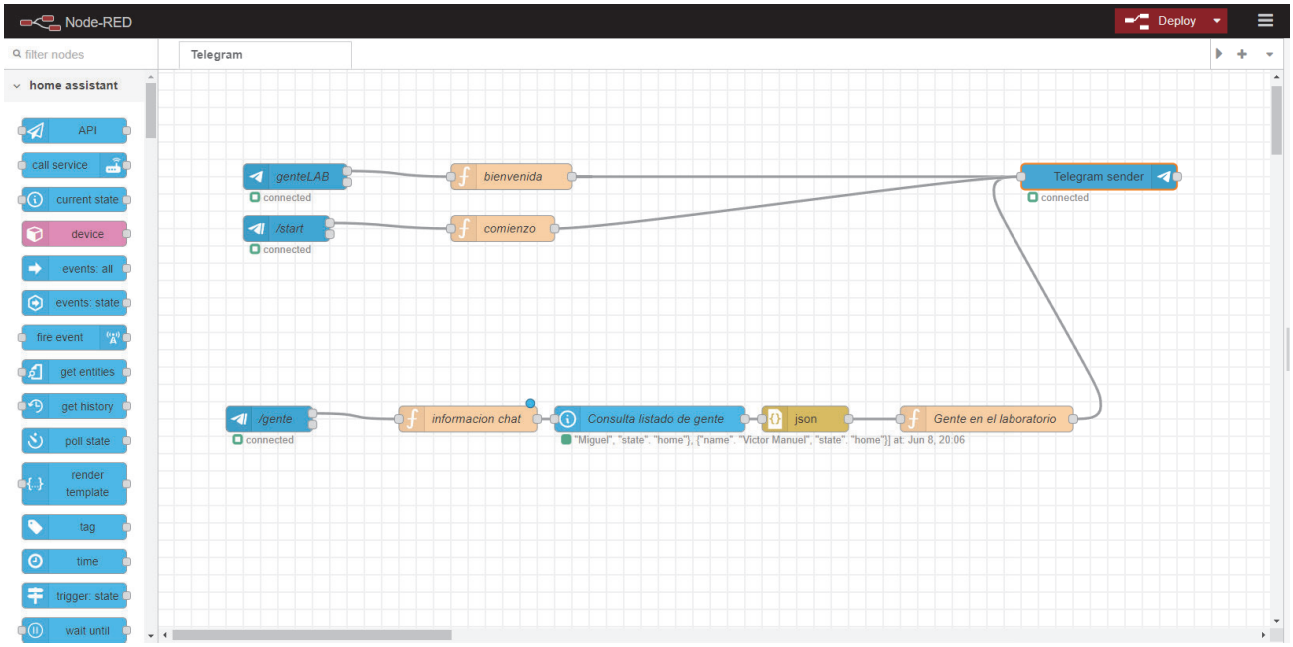



**Figura C.5:** Node-RED. Vista global del flujo implementado

La recepción del comando */gente* sirve como desencadenante para realizar una consulta

en los registros de Home Assistant al estado del sensor *sensor.people_status*, que contiene el listado de todos los miembros guardados junto a su estado actual. Previo a esta consulta, se almacenan los datos del mensaje que posteriormente permiten identificar el chat destinatario mediante la función *informacion chat* (Código C.3).

Finalmente se realiza una conversión a objeto de JavaScript del listado de gente en el laboratorio y se filtra el mensaje de salida según el estado asociado a cada miembro iterando sobre el dicho listado para conformar el mensaje que se envía al chat del usuario que ha introducido el comando */gente*. El mensaje de salida forma parte del campo *msg.payload.content*, y es formado empleando la función *Gente en el laboratorio*. El detalle del código JavaScript puede ser consultado en Código C.4

**Código C.1:** Función bienvenida. Se genera la respuesta predefinida en caso de recibir el comando /start. JavaScript

```
if (msg.payload.content.substring(0, 1) != "/") {
    msg.payload.content = "Este es el bot del laboratorio RoboRescue UMA:\n/gente ¿Quien hay en el ...
        LAB?";
    return msg;
}
```

**Código C.2:** Función comienzo. Se genera la respuesta predefinida en caso de recibir cualquier mensaje no considerado comando. JavaScript

```
msg.payload.content = "Este es el bot del laboratorio RoboRescue UMA:\n/gente ¿Quien hay en el ...
    LAB?";
return msg;
```

**Código C.3:** Función informacion chat. Se crea un campo nuevo en el mensaje para conservar los datos del cliente que indica el comando. JavaScript

```
msg.mensaje=msg.payload; //Se almacena para usar despues
return msg;
```

**Código C.4:** Función Gente en el laboratorio. Creación del mensaje cuyo contenido es el listado de gente en el laboratorio. JavaScript.

```

var listado = msg.payload;  //se recupera todo el listado
var listadoIn = []; //listado vacío
var output = "Actualmente están en el laboratorio: " + "\n\n"; //Output, por lo pronto vacío
var cont= 0;
msg.payload=msg.mensaje;  //se recuperan los datos del destinatario

for(var i=0; i<listado.length; i++){
    if(listado[i].state=='home'){
        listadoIn.push(listado[i].name );
        cont ++;
    }
}
if(cont>0){     //si hay alguien
    output += listadoIn.join("\n"); // Añade al string de salida un miembro por linea
}else{          //si laboratorio vacio
    output="El laboratorio está vacío"
}
```

```
19
20 msg.payload.content=output;
21
22 return msg;
```