UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE
SISTEMAS INFORMÁTICOS

PROYECTO FIN DE GRADO

openHASP: FIRMWARE PARA LA IMPLEMENTACIÓN DE GEMELOS DIGITALES

AUTOR:
IVÁN GARCÍA SALGADO

TUTOR:
VICENTE A. GARCÍA ALCÁNTARA

JULIO 2025

Universidad
Politécnica
de Madrid
ETSI SISTEMAS
INFORMÁTICOS

Agradecimientos

A la Universidad Politécnica de Madrid, y a la Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, mi "Escuela", la que me ha acogido durante todos estos años y que, sin duda, ha dejado una imborrable huella en mi vida.

A mi tutor, Vicente, con quien ya he compartido muchos trabajos, asignaturas y exámenes. Gracias a su valioso conocimiento, paciencia y consejos.

A todos esos compañeros que me han acompañado en este bonito camino aportando su granito de arena.

A mi familia, mi padre Javier, mi madre Yolanda y mi hermana Soraya, que siempre han aportado tranquilidad cuando las cosas no mostraban su mejor cara.

A mi mejor equipo, mi pareja, Diana, por su incansable ayuda y paciencia. Sin ella nada de esto hubiera sido posible. Los momentos no han sido fáciles pero juntos se recuerdan más bonitos.

A mi perrita Deena, por las horas de compañía y amor incondicional.

Gracias, de corazón, por este bonito final.

Universidad
Politécnica
de Madrid
ETSI SISTEMAS
INFORMÁTICOS

Índice

Índice de figuras

Índice de tablas

Objetivo

Objetivo general

El propósito principal de este PFG (Proyecto Fin de Grado) es conocer la herramienta openHASP [1], para el desarrollo de Gemelos Digitales [2] e interfaces gráficas, analizando sus ventajas limitaciones y propiedades, y defender por qué el *framework* (Entorno de trabajo) openHASP ha sido elegido protagonista del desarrollo de este proyecto. A través del mismo, se busca no solo entender la funcionalidad, y utilidad, de la plataforma en el contexto actual, cómo puede ayudar tecnológicamente como sociedad, sino también proporcionar una guía estructurada que permita a cualquier estudiante, grupo o empresa, independientemente de su nivel de conocimiento y su finalidad, aprender y aplicar estos conceptos en sus futuros proyectos.

Se ofrecerá una metodología práctica y progresiva, presentando ejemplos que facilitarán la comprensión teórica de la herramienta y, sobre todo, su aplicación con microcontroladores de bajo consumo y coste. Con esto, se pretende facilitar el acceso a estas tecnologías, permitiendo que cualquier estudiante, investigador o entusiasta pueda desarrollar soluciones innovadoras sin barreras económicas o técnicas.

El proyecto guiará al lector desde los conceptos más básicos y ejemplos funcionales iniciales hasta niveles más avanzados, permitiendo que el documento se adapte a un sin fin de escenarios. Por ejemplo, se explicará la configuración del entorno de desarrollo, incluyendo tanto la instalación y puesta en marcha de un servidor MQTT (*Message Queuing Telemetry Transport*, Cola de mensajes Transporte de telemetría) [3] como su integración con openHASP, detallando los pasos necesarios para su despliegue en diferentes entornos, entre otros.

Además, se incluirán secciones dedicadas a la elección y configuración del hardware, proporcionando una explicación de la elección del microcontrolador adecuado, sus compatibilidades, diferencias y capacidades según el tipo de implementación que se desee realizar. Se explorará el despliegue de estas soluciones tanto en servidores propios como en servidores externos, como podría ser el de la Universidad, cubriendo los aspectos clave para su integración en infraestructuras ya existentes.

Como parte del proyecto, se documentará la construcción práctica de un prototipo funcional basado en openHASP, mostrando paso a paso cómo convertir una idea en una aplicación completamente funcional. De este modo, se contará con una referencia clara y detallada para adaptar estos ejemplos a las propias necesidades individuales.

En definitiva, este trabajo no solo servirá como una introducción estructurada y accesible a la construcción de interfaces gráficas en microcontroladores y al framework openHASP, sino que también proporcionará las bases necesarias para que cualquier interesado pueda desarrollar sus propios proyectos, contribuyendo al avance y la divulgación de estas tecnologías en la comunidad académica y profesional.

Objetivos específicos

1. *Explorar las herramientas existentes para el desarrollo de interfaces gráficas en microcontroladores:* Comparar diferentes *frameworks* y justificar la elección de openHASP para el proyecto.

2. *Desarrollar una guía de implementación:* Crear un documento estructurado que permita a cualquier usuario comprender y aplicar el desarrollo de Gemelos Digitales en plataformas de bajo coste y consumo.

3. *Implementar ejemplos funcionales y prácticos:* Poner en marcha ejemplos prácticos que faciliten la aplicación de openHASP en distintos escenarios, desde lo más básico hasta configuraciones avanzadas.

4. *Configurar un entorno completo de despliegue:* Explicar la instalación y puesta en marcha de un servidor MQTT y la integración con openHASP en diferentes plataformas, incluyendo servidores locales y externos.

6. *Desarrollo final:* Crear, con toda la documentación anteriormente desarrollada, un modelo funcional completo. Este desarrollo, a modo de prototipo, será un pastillero inteligente, Pillbot, el cual de forma remota, a través de un servidor central, llevará a cabo acciones de monitorización de un paciente objetivo. Se emplearán todas las técnicas de creación de entornos, interfaces y objetos completamente funcionales. Asimismo, se cumplirán estándares de accesibilidad de interfaces para personas vulnerables, como se verá en apartados más adelante.

Resumen

Este Proyecto Fin de Grado desarrolla una guía completa y práctica sobre la utilidad de openHASP y su aplicación en entornos embebidos. Además, introduce el concepto de Gemelos Digitales, explicando qué son, cuál es su propósito y cómo pueden beneficiar a los usuarios. A lo largo del documento, se presentan ejemplos y técnicas que permiten implementar estas tecnologías utilizando el *firmware* (Soporte Lógico inalterable) seleccionado, culminando en el desarrollo de un sistema completamente funcional, listo para ser integrado en un dispositivo embebido.

El sistema se basa en una pantalla táctil de la marca Sunton, la cual incluye una pantalla TFT (*Thin Film Transistor*, Película fina de transistores) [4] y un microcontrolador ESP32-S3 [5] de última generación, junto con sensores y entradas GPIO (*General Purpose Input/Output*, Propósito general Entrada/Salida) [6]. Esta combinación proporciona una solución de bajo coste y fácil implementación, ya que el hardware viene listo para su instalación y uso inmediato. Gracias a la amplia compatibilidad de los ESP32 [7] con diversos protocolos de comunicación, es posible instalar múltiples *firmwares* y facilitar la integración de estos dispositivos en sistemas más complejos.

El desarrollo del proyecto se estructura en varias fases. En primer lugar, se proporciona un contexto tecnológico sobre la herramienta, openHASP, destacando sus ventajas y diferencias con otras alternativas. Se justifica la elección de openHASP como la mejor opción para determinados casos de uso. Posteriormente, se aborda el proceso de selección del hardware compatible, permitiendo que el usuario pueda adaptar la solución según sus necesidades específicas.

Una vez establecido el contexto teórico, la guía detalla los procesos de instalación del *firmware* seleccionado, ofreciendo tanto una instalación manual como una opción automatizada, dependiendo del caso de uso. Del mismo modo que se elige el hardware y software adecuados, se presentan los componentes esenciales para el desarrollo del entorno, incluyendo la configuración de un servidor MQTT. Se explican las opciones de configuración disponibles y se guía al usuario en la selección de la más adecuada según sus requerimientos.

Con el entorno completamente instalado y configurado, el documento profundiza en los principios fundamentales de la programación en openHASP, introduciendo sus elementos clave de forma progresiva. Se explican conceptos como la estructura de una página, la interacción con botones y la comunicación, avanzando hacia temas más complejos como la navegación por menús, la comunicación entre múltiples ESP32 y la interacción con otros dispositivos a través de MQTT.

Finalmente, se cierra la guía con un ejemplo funcional completo, aplicado a un caso de uso real, validando así los conocimientos adquiridos a lo largo del documento. Este caso práctico actúa como una prueba funcional del sistema, demostrando su aplicabilidad y reforzando la comprensión tanto teórica como técnica de la solución propuesta.

Este proyecto contribuye a la literatura existente sobre tecnologías embebidas y sostenibles, promoviendo el desarrollo de sistemas basados en microcontroladores de bajo coste y alto rendimiento, optimizando la integración y la automatización.

Abstract

This Final Degree Project aims to develop a comprehensive and practical guide to the usefulness of openHASP and its application in embedded environments. It also introduces the concept of Digital Twins, explaining what they are, what their purpose is and how they can benefit users. Throughout the document, examples and techniques are presented that allow these technologies to be implemented using the selected firmware, culminating in the development of a fully functional system, ready to be integrated into an embedded device.

The system is based on a Sunton touch screen, which includes an LCD display and a state-of-the-art ESP32-S3 microcontroller, together with sensors and GPIO inputs. This combination provides a low-cost and easy-to-implement solution, as the hardware comes ready for immediate installation and use. Thanks to the ESP32's wide compatibility with various communication protocols, it is possible to install multiple firmwares and facilitate the integration of these devices into more complex systems.

The development of the project is structured in several phases. First, a technological background on Digital Twins and existing control systems, including openHASP, is provided, highlighting their advantages and differences with other alternatives. It justifies the choice of openHASP as the best option for certain use cases. Subsequently, the process of selecting compatible hardware is addressed, allowing the user to tailor the solution to his or her specific needs.

Once the theoretical context has been established, the guide details the installation processes for the selected firmware, offering both a manual installation and an automated option, depending on the use case. As well as choosing the appropriate hardware and software, the essential components for the development of the environment are presented, including the configuration of an MQTT server. The available configuration options are explained and the user is guided in the selection of the most suitable one according to his requirements.

With the environment fully installed and configured, the document delves into the fundamentals of openHASP programming, introducing its key elements in a step-by-step manner. Concepts such as page structure, button interaction and communication are

explained, moving on to more complex topics such as menu navigation, communication between multiple ESP32 and interaction with other devices via MQTT.

Finally, the guide closes with a complete functional example, applied to a real use case, thus validating the knowledge acquired throughout the document. This case study acts as a functional test of the system, demonstrating its applicability and reinforcing both the theoretical and technical understanding of the proposed solution.

This project contributes to the existing literature on embedded and sustainable technologies, promoting the development of low-cost, high-performance microcontroller-based systems, optimising integration and automation.

Estructura de la memoria

Esta memoria ha sido estructurada con la intención de proporcionar una comprensión del desarrollo de interfaces gráficas en hardware haciendo uso de openHASP. Se pone especial énfasis en que se entienda de forma progresiva, es decir, desde los conceptos básicos hasta la creación de un sistema completo funcional, asegurando una formación integral en el uso de esta tecnología. A continuación, se describe cada uno de los capítulos que conforman esta memoria:

Capítulo 1, Introducción. Se presenta el contexto y la motivación que llevaron al desarrollo del proyecto. Se expone la relevancia de openHASP, destacando su utilidad en sistemas IoT (*Internet of things*, Internet de las cosas) [8], domótica y otras aplicaciones.

Capítulo 2, Estado de la técnica. Incluye una revisión de las soluciones actuales para el desarrollo de interfaces gráficas. Se analizan productos alternativos y estudios, donde se comparan con el presente desarrollado.

Capítulo 3, Marco Teórico. Declaración de los fundamentos técnicos y teóricos necesarios para la comprensión del proyecto.

Capítulo 4, Desarrollo e Implementación. Se detalla el proceso de instalación y configuración del entorno de desarrollo, incluyendo la instalación de un servidor MQTT y la implementación de openHASP en el hardware elegido. Además, se explica paso a paso el diseño y desarrollo de interfaces gráficas, desde elementos básicos hasta controles avanzados.

Capítulo 5, Pruebas y Validación. Se exponen las pruebas realizadas para garantizar el correcto funcionamiento del sistema. Se incluyen pruebas unitarias y pruebas de integración con sistemas IoT.

Capítulo 6, Herramientas utilizadas. Software usado en el desarrollo del proyecto.

Capítulo 7, Discusión de Resultados. Análisis de los resultados obtenidos, evaluando la eficacia del proyecto. Se comparan las expectativas iniciales con los resultados.

Capítulo 8, Impacto Social, Ambiental y Agenda 2030. Impacto del proyecto en términos sociales y ambientales, considerando los objetivos de la Agenda 2030 para el desarrollo sostenible.

Capítulo 9, Conclusión. Exposición de los logros del proyecto, reafirmando la contribución del sistema al campo de la agricultura inteligente y sostenible.

Capítulo 10, Líneas Futuras. Se proponen mejoras y posibles extensiones del proyecto.

Capítulo 11, Glosario de acrónimos. Explicación de acrónimos empleados.

Capítulo 12, Bibliografía. Fuentes consultadas y citadas a lo largo del documento, proporcionando un respaldo científico y técnico al proyecto.

Anexos. Material complementario, datos técnicos, fragmentos de código, configuraciones de ejemplo y otros documentos relevantes que complementan la memoria.

Esta estructura está diseñada para guiar al lector a través de todas las fases del proyecto, desde la instalación y configuración de openHASP hasta su implementación y evaluación, asegurando un entendimiento completo de la tecnología y sus aplicaciones en microcontroladores.

1. Introducción

Los productos tecnológicos son cada vez más frecuentes en nuestra vida cotidiana. Según el II Observatorio de la Unión de Créditos Inmobiliarios (UCI) [9], un 49,2% de los hogares en España cuentan con al menos un dispositivo inteligente, con una media de 1,78 dispositivos por vivienda. Por otro lado, un informe publicado por IT Reseller [10] estima que en 2025 más del 20% de las viviendas en España serán consideradas "inteligentes", lo que representaría aproximadamente 3,8 millones de hogares. A nivel global, la tendencia hacia la digitalización del hogar sigue en aumento, impulsada por la búsqueda de mayor eficiencia energética, seguridad y comodidad.

Estos dispositivos, cada vez más presentes en los hogares, asumen tareas en las actividades diarias, como la búsqueda en Internet, la consulta de pronósticos meteorológicos o el control de sistemas domóticos, incluyendo persianas, termostatos, electrodomésticos o incluso el cuidado de mascotas y plantas.

Sin embargo, la transformación tecnológica no se limita a los hogares, sino que también se ha expandido a la industria o el comercio con la llegada de la denominada Industria 4.0 [11]. En concreto, en la industria española, la adopción de dispositivos inteligentes y soluciones tecnológicas ha transformado significativamente los procesos productivos. Según un informe del Observatorio Nacional de Tecnología y Sociedad (ONTSI) [12], el 63% de las empresas de los sectores de energía y agua, el 54% en la industria de alimentación, textil, madera y artes gráficas, y el 53% en la fabricación de productos electrónicos, informáticos, material eléctrico, vehículos y muebles han incorporado tecnologías como IoT en algunas de sus actividades.

El éxito de estas tecnologías en distintos ámbitos de la sociedad se debe, en parte, a su facilidad de uso y accesibilidad. El Modelo de Aceptación de Tecnología (TAM, por sus siglas en inglés) sugiere que la "facilidad percibida de uso" es un factor determinante en la adopción de nuevas tecnologías, ya que los usuarios tienden a preferir sistemas que requieren menos esfuerzo para su utilización [13].

Esta teoría se relaciona con conceptos como la “expectativa de esfuerzo”, que se define como el grado de facilidad de uso asociado a una tecnología. Si el usuario percibe que

le será fácil utilizar una determinada herramienta o sistema, es más probable que la adopte [14].

A lo largo de la historia, diversos inventos han fracasado debido a su complejidad de uso, lo que ha limitado su aceptación por parte del público. Un ejemplo notable es el Apple Lisa, un ordenador personal lanzado en 1983 que, a pesar de su avanzada interfaz gráfica, resultó ser demasiado costoso y complicado para los usuarios de la época, lo que llevó a su fracaso comercial [15].

Es por ello que surgen tecnologías *Open Source* (código abierto) [16] para aquellos que quieran hacer más fácil, accesible y, por ende, atractivo con muchas posibilidades de ser exitoso alcanzando el objetivo de crear impacto para lo que fue creado. Un claro ejemplo de esto es openHASP, una plataforma de código abierto diseñada para la creación de interfaces gráficas intuitivas en dispositivos. Gracias a su flexibilidad y compatibilidad con hardware asequible, openHASP permite a los desarrolladores personalizar sus propios dispositivos controladores.

En este contexto, este proyecto no solo difunde el conocimiento sobre estas herramientas, sino que también ofrece una guía práctica y accesible para su implementación. A través de un enfoque didáctico, se abordan desde los fundamentos básicos hasta el desarrollo de un producto funcional, incluyendo la instalación de los elementos previos necesarios y la integración con hardware compatible. De este modo, se facilita el acceso a la tecnología para aquellos que deseen crear soluciones personalizadas y optimizar sus propios sistemas sin grandes barreras técnicas o económicas.

2. Estado de la Técnica

En este capítulo, se detallan las lecturas previas que se realizaron para el desarrollo del proyecto, así como cuáles fueron las ideas y conclusiones de las mismas aplicadas.

2.1. Introducción

El objetivo de este capítulo es revisar e investigar las contribuciones teóricas y las tecnologías previas relacionadas con el desarrollo de entornos gráficos en sistemas hardware de bajo coste, como microcontroladores, ya sea ESP-IDF [17], Arduino [18] o cualquier otra tecnología de desarrollo comúnmente usada en los sistemas embebidos.

2.2. Alcance del tema

El diseño y control de sistemas mediante interfaces gráficas desempeña un papel clave en la accesibilidad y operatividad de la tecnología. Un sistema con una interfaz no solo mejora la experiencia del usuario, sino que también facilita el aprendizaje y ayuda a conseguir el objetivo de la creación del producto.

Este campo de investigación no solo abarca investigaciones en la ingeniería y la tecnología, sino que también impacta áreas como la psicología y el comportamiento humano, marketing y comercio, puesto que una interfaz bien diseñada puede influir en el uso del sistema.

2.3. Revisión

Para esta revisión, se ha utilizado la herramienta Google Scholar [19] y el Archivo Digital UPM [20] como fuentes de documentación, donde se han seleccionado otros Proyectos Fin de Grado relacionados, libros y artículos científicos que abordan los siguientes temas:

1) *Diseño de interfaces de usuario para sistemas embebidos.*
2) *LVGL — Light and Versatile Embedded Graphics Library.*
3) *openHASP.*
4) *Técnicas educativas y de aprendizaje.*

2.3.1. Diseño de interfaces de usuario para Sistemas Embebidos

Para la discusión de este apartado se ha seleccionado el artículo de la revista online Medium [21] *Diseño de interfaces de usuario para Sistemas Embebidos*, debido a su formato revista, y de corta extensión, que hizo amena su lectura. El artículo del usuario *Luis Alberto Fernández Arnanz*, enumera algunas de las claves que más se destacan del momento del diseño de una interfaz gráfica, a rasgos muy generales.

En el artículo se enfatiza la importancia de comprender al cliente durante el proceso de diseño de interfaces gráficas para este tipo de sistemas. El autor destaca que el objetivo principal de cualquier interfaz de usuario debe ser adaptarse al producto, optimizando el uso de los recursos. Además, es esencial conocer en profundidad las necesidades y expectativas del cliente, lo que permite crear soluciones personalizadas y eficientes. Este enfoque centrado en el usuario asegura que el diseño no solo sea funcional, sino también intuitivo y satisfactorio para quienes interactúan con el sistema.

En resumen, una lectura corta y concisa para entender la finalidad y el objetivo final del desarrollo de una interfaz gráfica en un sistema embebido a rasgos generales.

2.3.2. LVGL - Light and Versatile Embedded Graphics Library

Para entender la base del producto a analizar y desarrollar hubo que comprender que era LVGL [22], una librería desarrollada en C++ enfocada en el diseño de interfaces gráficas en sistemas embebidos con una alta optimización de los recursos disponibles. Para ello se eligió el artículo *Comparison of TouchGFX and LVGL Embedded Hardware GUI Libraries* de la *Gazi University* [23] de la ciudad de Ankara, en Turquía.

En el artículo se menciona su eficiencia en el uso de memoria y procesamiento, permitiendo la creación de interfaces gráficas avanzadas sin necesidad de hardware dedicado. LVGL ofrece compatibilidad con microcontroladores como ESP32 [7], STM32 [24] y Raspberry Pi [25], y es capaz de ejecutarse tanto en sistemas operativos en tiempo real (RTOS) como en *bare metal*. Sus características incluyen gráficos vectoriales, soporte táctil, animaciones y componentes gráficos avanzados, todo optimizado para minimizar el consumo de recursos. Además, su arquitectura modular permite activar o desactivar funciones según las necesidades del hardware, lo que la hace ideal para su integración en plataformas como openHASP, donde facilita la interacción visual en sistemas de automatización embebidos.

2.3.3. openHASP.

Para entender la base del producto a analizar y desarrollar, se ha investigado sobre openHASP, una plataforma de código abierto que permite la creación de interfaces gráficas en sistemas embebidos, con especial énfasis en la automatización del hogar. Para ello, se ha seleccionado la información proporcionada en la página oficial de openHASP debido a su carácter técnico y detallado, lo que ha permitido comprender su arquitectura, compatibilidad y aplicación en dispositivos de bajo coste.

En la documentación oficial, se describe openHASP como una solución basada en la biblioteca LVGL, diseñada para operar en microcontroladores ESP32 y otros dispositivos embebidos de bajo consumo. Su principal ventaja es la posibilidad de integrar interfaces gráficas táctiles altamente configurables, las cuales se comunican a través de MQTT con sistemas de automatización como *Home Assistant* [26]. La arquitectura modular de openHASP permite a los usuarios personalizar su interfaz según sus necesidades, optimizando la usabilidad sin comprometer el rendimiento del sistema. Además, la plataforma ofrece compatibilidad con múltiples hardware y protocolos, facilitando su adopción en entornos domóticos y de control industrial. En resumen, openHASP es una herramienta flexible y eficiente que extiende las capacidades de los sistemas embebidos con interfaces gráficas intuitivas y conectividad avanzada.

2.3.4. Técnicas educativas y de aprendizaje

Por último, sobre la realización de un proyecto con fines educativos, que ayudarán a plasmar toda la información sobre el tema, a modo de tutorial, se han hecho algunas lecturas de otros proyectos similares del Archivo Digital UPM que hubieran abordado esta problemática en el ámbito tecnológico. En este sentido, la estructuración de un proyecto con una base didáctica requiere un enfoque que facilite el aprendizaje progresivo, permitiendo a los lectores adquirir conocimientos de manera clara y aplicable.

La lectura de proyectos como *Videotutoriales sobre el Entrenador analogico/digital ETS-7000A* de *Estrella Gutierrez Diaz* [27], en la que desarrolla ejemplos prácticos de forma progresiva de forma similar al objetivo de este proyecto.

De la misma temática que el anterior, se tiene D*ocumentación, tutorial multimedia del simulador Multisim (módulos digitales)* de *Jaime Martin Bradshaw* [28] que explica, de forma similar al anterior, y por ende, a la finalidad de este proyecto, una guía sobre el software Multisim, siguiendo un esquema progresivo y mezclando conocimientos tecnicos y teoricos.

En resumen, estas lecturas han servido como referencia para comprender la importancia de estructurar el contenido de forma progresiva y modular, asegurando que los conceptos se introduzcan de manera gradual, desde lo más básico hasta los aspectos más avanzados. La combinación de ejemplos prácticos, explicaciones teóricas y aplicaciones reales permite que el aprendizaje sea más dinámico y significativo, facilitando la asimilación de conocimientos técnicos.

3. Marco teórico

Este capítulo explica y revisa los conocimientos previos a los desarrollados en el proyecto. Dando a conocer todo el contexto al lector.

3.1. Pantallas de transistores de película delgada (TFT)

Las pantallas TFT incorporan, en cada píxel, un transistor. A diferencia de las LCD (*Liquid Crystal Display*, Pantalla de cristal liquido) [29] convencionales, las TFT permiten una representación de color más precisa y son ampliamente utilizadas en dispositivos electrónicos que requieren interfaces gráficas de alta resolución, como dispositivos portátiles, paneles de control y sistemas embebidos [4]. Su estructura, más detalladamente, se puede ver en la figura de Wikipedia a continuación.

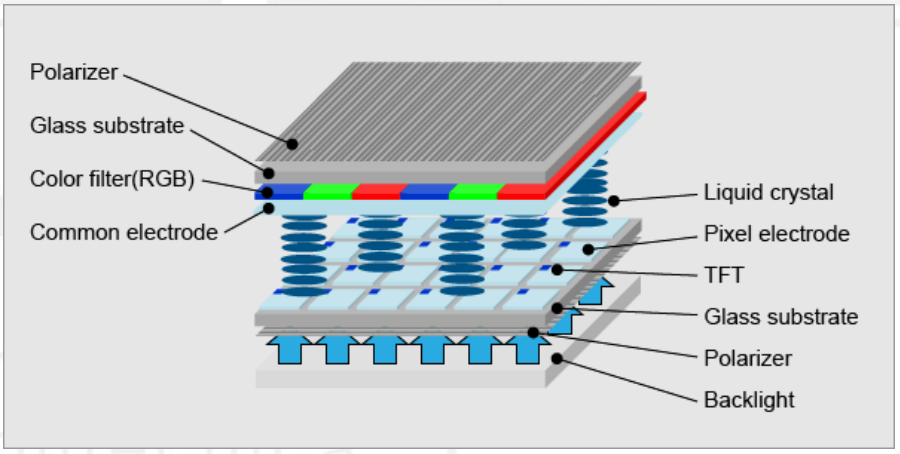


Figura 1. Componentes pantallas TFT

Cada píxel de una pantalla TFT está compuesto por un transistor y un capacitor que le permiten mantener su carga sin necesidad de recibir una actualización constante, lo que mejora la estabilidad de la imagen. La estructura de una pantalla TFT se compone de una capa de transistores que regula el paso de luz, un filtro de color para definir la imagen y una capa superior que muestra el contenido visual.

La resolución y el detalle de la imagen en una pantalla TFT dependen de la densidad de píxeles y del diseño de la matriz de color. A mayor cantidad de píxeles, transistores, mayor será la nitidez y precisión en la representación visual. Además, factores como el tamaño, el

consumo energético y el tipo de interfaz de conexión influyen en la funcionalidad de estos dispositivos.

A diferencia de tecnologías como OLED (*Organic Light-Emitting Diode*, Diodo orgánico de emisión de luz) [30], las pantallas TFT no emiten luz propia y requieren una fuente de retroiluminación para generar la imagen. En modelos más recientes, está retroiluminación se logra mediante LED (*Light-emitting Diode*, Diodo emisor de luz) [31].

3.2. Protocolo Serial Peripheral Interface (SPI)

La interfaz SPI (*Serial Peripheral Interface,* Interfaz serie de periféricos) [32] es un protocolo de comunicación síncrono utilizado para la transferencia de datos entre un microcontrolador y dispositivos periféricos, en este caso, una pantalla. A diferencia de otras interfaces de comunicación como I2C,UART,USB[33][34][35], SPI utiliza un enfoque basado en cuatro líneas principales para la transmisión de datos: MOSI (*Master Out Slave In,* Maestro fuera Esclavo dentro), MISO (*Master In Slave Out,* Maestro dentro Esclavo fuera), SCK (*Serial Clock,* Reloj Serial) y CS (*Chip Select,* Selector de Chip). En la figura siguiente, de la bibliografía Wikipedia, se puede observar las líneas de comunicación anteriormente mencionadas, prestando especial atención a la línea MISO de sincronización.

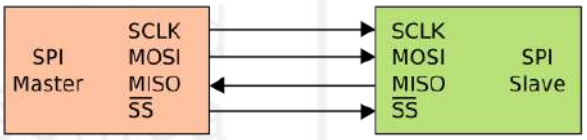


Figura 2. Funcionamiento SPI

SPI permite una transmisión de datos rápida y eficiente al utilizar un reloj de sincronización común entre el dispositivo maestro (el microcontrolador) y el dispositivo esclavo (la pantalla TFT). Este protocolo es ampliamente utilizado en aplicaciones donde se requiere alta velocidad y comunicación en tiempo real, ya que es capaz de transferir grandes volúmenes de datos rápidamente.

Una de las principales ventajas de SPI es su simplicidad de pines. Con solo cuatro líneas de conexión, se reduce la complejidad del cableado, lo que lo convierte en una opción atractiva para sistemas embebidos. Además, la configuración de SPI permite el uso de

múltiples dispositivos periféricos conectados en serie, mediante la utilización de una línea CS para seleccionar qué dispositivo debe recibir o enviar datos en cada momento.

SPI es también un protocolo *full-duplex*, lo que significa que puede enviar y recibir datos simultáneamente, lo que mejora la eficiencia en aplicaciones de alta velocidad. Aunque SPI no es tan flexible como otros protocolos en términos de cantidad de dispositivos conectados, su mayor velocidad de transmisión y su simplicidad lo convierten en una opción preferida para muchas aplicaciones embebidas.

3.3. Biblioteca de gráficos integrados ligera y versátil (LVGL)

LVGL *(Light and Versatile Graphics Library,* Libreria grafica ligera y versátil) es una biblioteca gráfica de código abierto diseñada para dispositivos con recursos limitados, como microcontroladores y sistemas embebidos. Su propósito es proporcionar una solución optimizada para la creación de interfaces gráficas modernas sin comprometer el rendimiento del hardware [22]. El logo representativo de LVGL puede verse a continuación en la figura.

Figura 3. Logo LVGL

A diferencia de otras bibliotecas gráficas que requieren una gran capacidad de procesamiento y memoria, LVGL está diseñada para operar con microcontroladores de bajo consumo, permitiendo la renderización eficiente de interfaces interactivas. Su arquitectura modular facilita la integración en diversas plataformas y soporta múltiples tipos de pantallas, incluyendo aquellas con interfaces SPI y paralelas.

Uno de los aspectos más destacables de LVGL es su sistema de *widgets,* o artilugios, y temas personalizables, que permite construir interfaces visuales complejas mediante elementos como botones, deslizadores, gráficos y listas.

Funciona en diferentes entornos y sistemas operativos. Puede ejecutarse de manera independiente en un microcontrolador o integrarse con sistemas operativos más completos

(Debian, Ubuntu, etc) [36]. Además, soporta la aceleración por hardware en plataformas que lo permitan.

LVGL se ha convertido en una de las bibliotecas más utilizadas en el desarrollo de interfaces gráficas para sistemas embebidos, facilitando la implementación de pantallas interactivas en aplicaciones como domótica, dispositivos médicos, paneles de control industrial y productos electrónicos de consumo.

3.4. openHASP, características principales

openHASP es una plataforma de código abierto, basada en LVGL (apartado anterior), diseñada para la creación de interfaces gráficas en sistemas embebidos. Su objetivo principal es proporcionar una solución flexible y eficiente para dispositivos con recursos limitados, como microcontroladores y placas de desarrollo, permitiendo el diseño de interfaces modernas sin necesidad de programación compleja [1]. Su logo, de la página oficinal de openHASP, se representa en la siguiente figura:

Figura 4. Logo openHASP

Una de las principales ventajas de openHASP es su capacidad para interactuar mediante protocolos de comunicación ligeros como MQTT y HTTP (*HyperText Transport Protocol*, Protocolo de transporte de Hipertexto) [37] facilitando la integración con sistemas de control remoto y plataformas IoT. Esto permite que las interfaces creadas con openHASP puedan recibir y enviar comandos desde servidores externos o aplicaciones móviles sin necesidad de modificar el código del microcontrolador. Además, su arquitectura modular permite configurar elementos gráficos como botones, deslizadores, indicadores y menús de manera sencilla a través de archivos de configuración JSON (*JavaScript Object Notation*, Notación de objetos JavaScript) [38] sin necesidad de compilar o reescribir código.

openHASP es compatible con microcontroladores de la familia *Espressif* [39], entre los más utilizados, ESP32 y ESP8266 [40], último ya menos común, pues fue el antecesor del ESP32, los cuales son ampliamente utilizados en aplicaciones embebidas debido a su bajo costo y capacidad de conectividad WiFi (*Wireless Fidelity*, Fidelidad inalámbrica) [41]. La combinación de openHASP con estos dispositivos permite crear paneles de control interactivos para el manejo de luces, sensores, actuadores y otros sistemas de automatización.

El *firmware* proporciona una solución eficiente para el desarrollo de interfaces gráficas. El uso de protocolos de comunicación ligeros, su alta compatibilidad con hardware y las facilidades de configuración, convierten a esta herramienta ideal para el desarrollo de entornos gráficos en sistemas embebidos en sectores como la automatización industrial.

3.5. Protocolo de comunicación MQTT

MQTT es un protocolo para la comunicación entre dispositivos en redes con recursos limitados [3], como los sistemas embebidos.

Este protocolo permite la transmisión de mensajes entre dispositivos y un servidor, de forma bidireccional, mediante un modelo de publicación/suscripción. Esto se logra a través de un agente (canales) de mensajes que gestiona la comunicación. Las características de MQTT, se basan en su ligereza, escalabilidad, fiabilidad y seguridad, lo que lo han convertido en un estándar para entornos de bajo consumo, como IoT. Además de su alta compatibilidad, MQTT facilita la detección y recuperación de errores en redes con baja fiabilidad y alta latencia, y ofrece varios niveles de calidad de servicio para asegurar la entrega de mensajes.

Figura 5. Logo de MQTT

Desde la versión lanzada en 2019, MQTT soporta la comunicación segura mediante cifrado y autenticación moderna, y su implementación en diversos lenguajes. Un ejemplo del flujo de intercambio de mensajes mediante el protocolo MQTT se puede ver en la siguiente figura, de SysTec [42]:

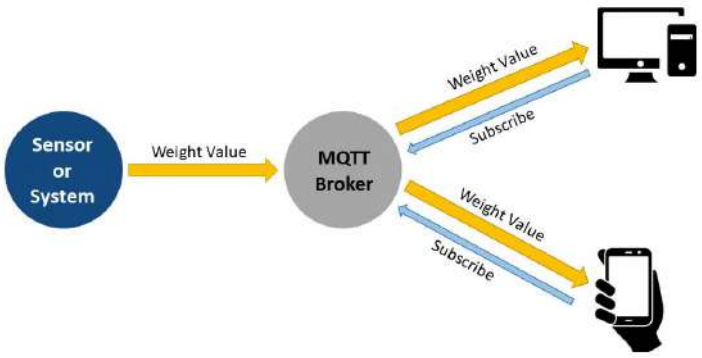


Figura 6. Protocolo MQTT

3.6. Microcontroladores ESP32

Los microcontroladores ESP32 [7] son una familia de chips de bajo coste y consumo de energía, con tecnología WiFi y Bluetooth [43], desarrollados por *Espressif Systems* [39]. Estos microcontroladores están diseñados principalmente para aplicaciones de IoT. El aspecto del microcontrolador es el mostrado en la siguiente figura del mismo fabricante *Espressif*:

Figura 7. Microcontroladores ESP32

El ESP32 DevKit-C [44] es una de las placas de desarrollo más populares de la serie ESP32, ya que ofrece una plataforma flexible y potente para el desarrollo de proyectos embebidos. Destacan por su capacidad de procesamiento y conectividad.

A continuación, se enumeran las principales características del ESP32 DevKit-C en la siguiente figura, también de *Espressif*:

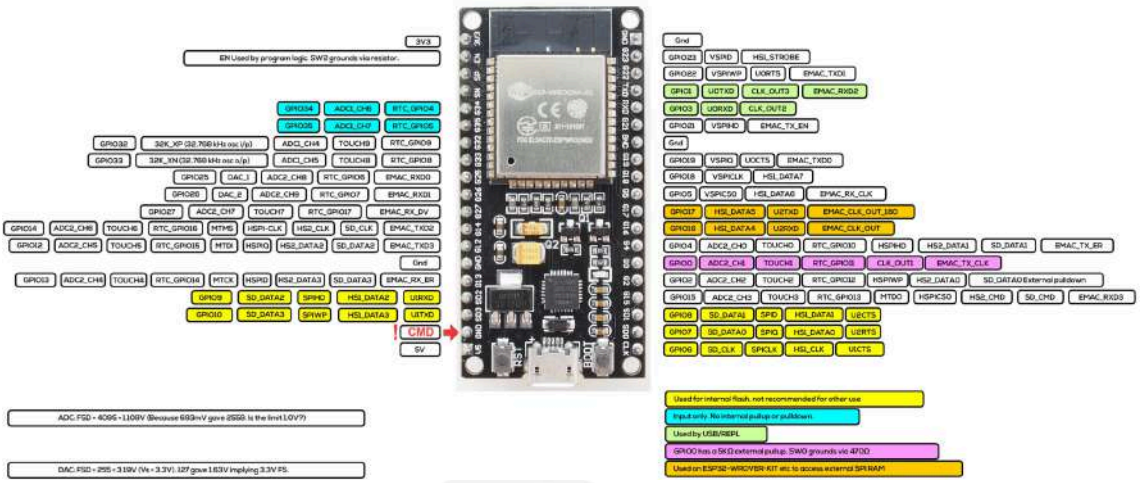


Figura 8. Características ESP32 DevKit-C

Y en una tabla, de fuente propia con este formato de aquí en adelante, la cantidad de cada componente en el microcontrolador:

Componente	Cantidad
ADC de 12 bits	Hasta 18 canales
DAC de 8 bits	2
SPI	4
I2S	2
I2C	2
UART	3
Controlador host SD/SDIO/etc.	1
Bus CAN 2.0	1

Tabla 1. Componentes microcontrolador ESP32

El ESP32 DevKit-C es una placa de desarrollo que incorpora un potente microprocesador Tensilica Xtensa de 32 bits, disponible en uno o dos núcleos, operando a velocidades de 160 o 240 MHz y ofreciendo un rendimiento de hasta 600 DMIPS, respectivamente. Además, cuenta con un co-procesador de ULP (*Ultra low power*, Ultra baja energia) que optimiza el consumo en aplicaciones de bajo rendimiento.

En cuanto a la memoria, el ESP32 dispone de 520 KB de SRAM. En el ámbito de la conectividad inalámbrica, soporta Wi-Fi 802.11 b/g/n y Bluetooth v4.2 BR/EDR y BLE, permitiendo una amplia gama de aplicaciones de comunicación inalámbrica.

El microcontrolador también ofrece múltiples interfaces periféricas, incluyendo un ADC de 12 bits con hasta 18 canales, 2 DACs de 8 bits, 4 interfaces SPI, 2 interfaces I2S, 2 interfaces I2C y 3 UARTs. Además, cuenta con un controlador host SD/SDIO/CE-ATA/MMC/eMMC, un controlador esclavo SDIO/SPI y un bus CAN 2.0.

El ESP32 también incluye un controlador (TX/RX) con soporte para hasta 8 canales y un controlador PWM. Para la administración de energía, el ESP32 permite el control individual para el RTC y puede operar con una corriente de apenas 5µA en modo de suspensión. Además, cuenta con la capacidad de despertarse mediante interrupciones de GPIO o temporizadores.

3.7. Gemelos digitales

Un Gemelo Digital [2] se define como una representación virtual de un objeto o sistema físico que muestra en tiempo real su estado y progreso de acciones, facilitando su gestión y control a distancia. Dentro del marco de los sistemas embebidos o microcontroladores, un gemelo digital no necesita siempre una simulación avanzada como comúnmente se conoce, sino que puede representarse también mediante una interfaz que reproduce con precisión las variables, estados y acciones del dispositivo físico, permitiendo al usuario supervisar y administrar su comportamiento sin necesidad de intervención directa en el sistema.

Es por eso que *frameworks* como openHASP, pueden facilitar la creación de estas herramientas que acerquen al usuario a la gestión y monitorización de sistemas de forma remota y virtual.

4. Desarrollo e Implementación

En este capítulo, uno de los más importantes del documento, se detalla de manera exhaustiva todo el proceso de desarrollo y configuración de la plataforma, y entorno necesario, openHASP.

4.1. Elección hardware compatible

Según los requisitos del proyecto, los desarrolladores pueden optar por diseñar la solución desde cero, seleccionando individualmente el microprocesador, la pantalla y otros componentes, o bien recurrir a una solución prácticamente integrada, que permita instalar el *firmware* de manera rápida y facilitar la integración con el dispositivo, agilizando el desarrollo. A continuación, se explican ambas opciones y los componentes recomendados para cada enfoque.

Microprocesador

Como se ha visto en capítulos anteriores, el hardware compatible debe cumplir una serie de requisitos mínimos para poder aprovechar el completo potencial de las funcionalidades de la plataforma.

La plataforma está desarrollada principalmente para ser instalada en microcontroladores de la familia ESP32, lo que la hace altamente compatible con la mayoría de microcontroladores del mercado. Actualmente es difícil calcular el número de modelos y versiones totales de la gama ESP32, pero a continuación se detallan los que cumplen con las características necesarias.

La comunidad de desarrolladores de openHASP enumeran que una placa de desarrollo debe tener al menos [45]:

- 520 kB de SRAM.
- 8 MB de memoria flash.
- 2 MB de PSRAM.

Dado el continuo desarrollo de la plataforma y sus mejoras constantes, se recomienda optar por un sistema que supere estos requisitos, asegurando así un margen suficiente para futuras actualizaciones sin comprometer el rendimiento.

Aun acotando el número de modelos ESP con estos datos, son aún incontables. Aunque la que presenta mayor compatibilidad con la plataforma es la placa de desarrollo ESP32-S3 [5], no solo por sus características técnicas, si no también por su precio competitivo, guías y comunidad disponible acerca del producto.

La ESP32-S3 es la placa de desarrollo con mayor compatibilidad con la plataforma. Sus principales ventajas incluyen:

ESP32-S3				
SRAM	**Flash**	**PSRAM**	**Precio**	**Ventaja**
512 kB	8 MB	2 MB	7.5 €/media	Soporte

Tabla 1. Características ESP32-S3

Aun teniendo características y aspecto similar, el modelo ESP32-S2 no es compatible con openHASP.

Otros modelos compatibles a mencionar serían:

ESP32-WROVER				
SRAM	**Flash**	**PSRAM**	**Precio**	**Ventaja**
520 kB	8 MB	4 MB	8.0 €/media	Potencia

Tabla 2. Características ESP32-WROVER

Entre las placas de desarrollo WROVER [46], destacan ESP32-WROVER, ESP32-WROVER-B y ESP32-WROVER-E, ya que cuentan con PSRAM integrada y suficiente memoria para garantizar un funcionamiento fluido.

Sin embargo, versiones más antiguas como ESP32-WROVER-A no cumplen con los requisitos y tendrán problemas de compatibilidad.

ESP32-PICO-D4				
SRAM	**Flash**	**PSRAM**	**Precio**	**Ventaja**
520 kB	4 MB	0 MB	6.0 €/media	Compacto

Tabla 3. Características ESP32-PICO-D4

Dentro de la gama ESP32-PICO [47], los modelos ESP32-PICO-D4, ESP32-PICO-V3 y ESP32-PICO-D1 son opciones compactas que cumplen con los requisitos especificados, además, si el proyecto requiere de un requisito físico (p.e una limitación de tamaño al ser incrustado en un armario hidroponico), es una de las mejores opciones debido a su tamaño.

ESP32-C3				
SRAM	**Flash**	**PSRAM**	**Precio**	**Ventaja**
400 kB	4 MB	0 MB	4.0 €/media	Eficiencia

Tabla 4. Características ESP32-C3

Dentro de la familia ESP32-C3 [48], basados en RISC-V de 32 bits, con 400 kB de SRAM, 4 MB de flash, lo que lo hace menos recomendable para openHASP.

En conclusión, los modelos pueden variar en el tiempo debido a su descontinuación en desarrollo por los fabricantes o por catálogo, además de la mejora de openHASP y otros entornos de desarrollo, lo que hace que sea mejor guiarse por las características mínimas del contexto en el que se encuentra el desarrollo del proyecto.

Pantalla

Una vez elegido el microprocesador donde se ejecutará el sistema operativo, es momento de seleccionar uno de los elementos más importantes del sistema: la pantalla.

Estas deben ser compatibles con el protocolo SPI, explicado en el capítulo del marco teórico. Además, una característica casi imprescindible, más allá de la calidad o el tamaño de

la imagen, es su capacidad táctil, ya que facilita la interacción con el dispositivo y refuerza el objetivo de este documento: crear tecnología accesible.

Existen pantallas en múltiples tamaños y formatos, desde modelos cuadrados de 2.3, 4 hasta 7 pulgadas, hasta formatos rectangulares o circulares, ideales para aplicaciones como *smartwatches* o controles de sistemas de ventilación. Algunos controladores de pantalla ampliamente utilizados incluyen el ILI9341, ILI9488, ST7796 y ST7789, cada una con diferentes niveles de precisión y sensibilidad [45].

La elección de la pantalla dependerá de los requisitos específicos del proyecto, priorizando compatibilidad, calidad de imagen y capacidad táctil para mejorar la experiencia de usuario.

Solución completa

Existen diversas opciones que integran el microcontrolador y la pantalla, ofreciendo soluciones completas y optimizadas. Entre las más destacadas se encuentra la ESP32 TouchDown, una placa de desarrollo que incorpora una pantalla táctil capacitiva de 3.5 pulgadas ILI9488. Cuenta con 4 MB de memoria flash. Con características similares se encuentran también los modelos M5Stack core2, la Makerfabs ESP32-S3 Parallel TFT, y AZ-Touch MOD [45].

A continuación se detallan las características de algunas de las más populares en el mercado y compatibles con el *firmware* de openHASP.

Como ya se citó la ESP32 TouchDown es una placa de desarrollo que incorpora una pantalla táctil de 3.5” ILI9488. Cuenta con 4 MB de memoria flash y está diseñada para facilitar la creación de interfaces gráficas interactivas. Su diseño compacto y la integración del ESP32 la convierten en una opción versátil para diversos proyectos.

El M5Stack Core2 [49] es un dispositivo modular que integra una pantalla táctil de 2 pulgadas con el controlador ILI9342C y el chip táctil FT6336U. Dispone de 16 MB de memoria flash y está orientado a aplicaciones de IoT y prototipado rápido. Su diseño modular y la amplia comunidad de soporte lo hacen ideal para desarrolladores que buscan una solución flexible y escalable.

La Makerfabs ESP32-S3 Parallel TFT [50] es una placa que combina el ESP32-S3 con una pantalla TFT de 3.5 pulgadas, utilizando el controlador ILI9488 y el chip táctil FT6236. Ofrece un alto rendimiento gráfico y soporte para múltiples interfaces, siendo adecuada para aplicaciones que requieren interfaces gráficas de alta calidad.

El AZ-Touch MOD [51] es una solución que integra una pantalla táctil de 2.4 pulgadas con el controlador ILI9341 y el chip táctil XPT2046, junto con el ESP32. Está diseñada para aplicaciones de domótica y control ambiental, facilitando la integración en entornos domésticos y permitiendo un control eficiente de dispositivos inteligentes.

La Lilygo T-Display-S3 [52] es una placa de desarrollo compacta que incorpora una pantalla a color de 1.9 pulgadas con el controlador ST7789 y el ESP32-S3. Es ideal para proyectos portátiles y wearables debido a su tamaño reducido y bajo consumo energético. A pesar de su tamaño, ofrece funcionalidades completas para el desarrollo de interfaces gráficas básicas.

El WT32-SC01 [53] es un módulo que combina una pantalla táctil de 3.5 pulgadas con el controlador ST7796 y el chip táctil FT6336U, junto con el ESP32. Facilita el desarrollo de interfaces de usuario interactivas y es adecuado para aplicaciones que requieren una pantalla de mayor tamaño sin comprometer la funcionalidad.

El fabricante Sunton ofrece pantallas de mayor tamaño, como la de 7.7 pulgadas táctil, con una resolución de 800x480 y una buena gama de colores. Estas pantallas son ideales para proyectos que requieren una visualización más amplia y detallada, manteniendo un equilibrio entre calidad y costo.

Si el precio es el factor determinante, existen opciones accesibles sin comprometer la funcionalidad. Modelos como la Lilygo T-Display-S3 o la WT32-SC01, ideal para proyectos sencillos y de bajo consumo permitiendo desarrollar interfaces gráficas sin un alto coste.

Si el tamaño es un requisito del sistema, Sunton, elegidas para el desarrollo de este proyecto, que a un bajo precio, ofrecen opciones de incluso 7,7” táctil, con amplios recursos y características. Estas pantallas presentan hasta una resolución de 800x480, y una buena gama de colores, suficiente para mostrar con una calidad aceptable contenidos.

La pantalla elegida, modelo Sunton 8048S070C, el cual su aspecto está representado en la siguiente figura, de su página de compra en el vendedor *Aliexpress*:

Figura 9. Pantalla Suunto 8040S070C

Es una fantástica opción, por sus características completas, como tamaño, opciones desde 3" hasta 7", precio, desde unos 15€ las opciones más asequibles, hasta 30€ las opciones con mayores recursos, que pueden llegar hasta los 16 MB, así como la calidad de su imagen y del panel táctil.

4.2. Instalación de openHASP en el hardware

De la misma forma en la que un entorno hardware preparado para openHASP podría construirse de varias formas: eligiendo uno a uno los componentes o adquiriendo un producto semi-integrado, la instalación en hardware puede producirse de la misma forma, manual o automática. Desde la versión 0.6.3 de openHASP, versión en la que se incluyen numerosos dispositivos más compatibles, se puede elegir la opción automática, mucho más cómoda si lo que se quiere es sencillez y comodidad en el proceso.

4.2.1. Instalación manual

La plataforma de desarrolladores de openHASP, de código abierto para la comunidad, mantiene un repositorio GitHub [55] en la que se pueden proponer cambios, discutir funcionalidades o estar al tanto de las últimas novedades *releases* (versiones). Estas Releases pueden encontrarse en el siguiente enlace del anexo.

En esta pestaña de GitHub aparecerán todas las versiones, junto con las diferencias reflejadas en los *commits* realizados por los distintos *contributors* (contribuidores). Es posible optar por la versión más reciente, que incluye las últimas novedades pero puede ser menos estable, o por una versión anterior, que suele ser más fiable. La elección dependerá de los requisitos del proyecto.

Por ejemplo, para el desarrollo de esta guía, se utiliza la última versión disponible, Release 0.7.0-rc13, que se puede obtener en el siguiente enlace:

https://github.com/HASwitchPlate/openHASP/releases/tag/0.7.0-rc13

Dentro de la propia *release* se verá con más detalles estas contribuciones de los diferentes desarrolladores, y en la parte final de la página los *assets* (activos). En la siguiente figura, de fuente propia, de aquí en adelante, sin referencia, se puede observar cómo la comunidad generó cada activo para un modelo de pantalla.

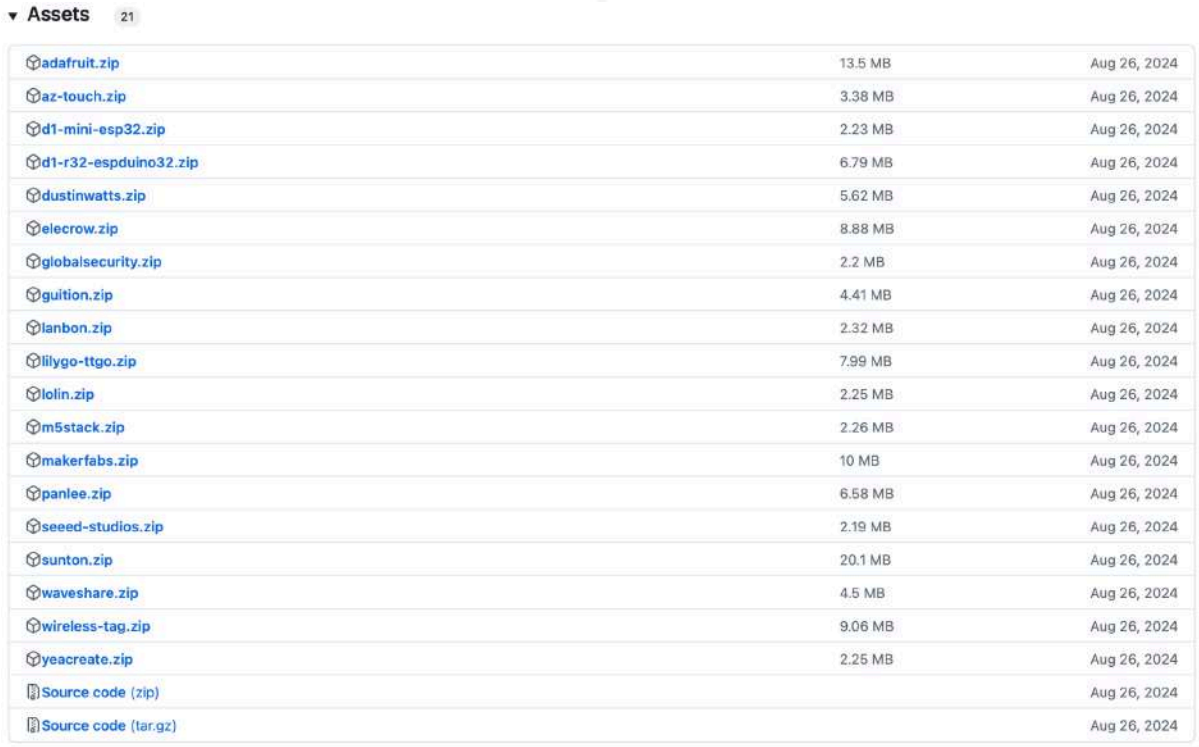
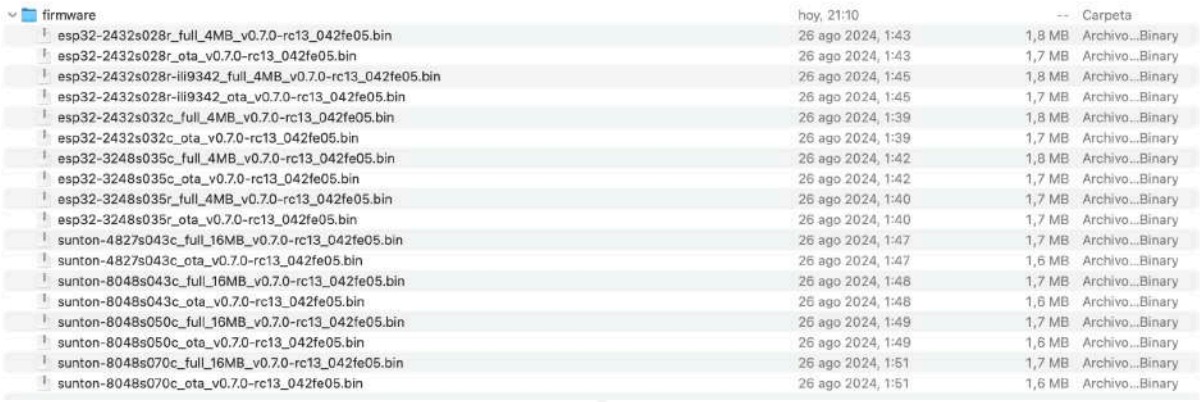


Figura 10. Releases Github openHASP

En esta sección, se deberá seleccionar el archivo correspondiente al dispositivo elegido para el proyecto (como se explicó en el apartado anterior). Para el desarrollo de esta guía, se seleccionará sunton.zip. Una vez extraído, el archivo tendrá la siguiente estructura dentro de la carpeta *firmware*.

```
└── sunton.zip
    └── build_output
        └── firmware
            └── binaries…
```



firmware	hoy, 21:10	--	Carpeta
esp32-2432s028r_full_4MB_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:43	1,8 MB	Archivo...Binary
esp32-2432s028r_ota_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:43	1,7 MB	Archivo...Binary
esp32-2432s028r-ili9342_full_4MB_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:45	1,8 MB	Archivo...Binary
esp32-2432s028r-ili9342_ota_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:45	1,7 MB	Archivo...Binary
esp32-2432s032c_full_4MB_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:39	1,8 MB	Archivo...Binary
esp32-2432s032c_ota_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:39	1,7 MB	Archivo...Binary
esp32-3248s035c_full_4MB_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:42	1,8 MB	Archivo...Binary
esp32-3248s035c_ota_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:42	1,7 MB	Archivo...Binary
esp32-3248s035r_full_4MB_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:40	1,7 MB	Archivo...Binary
esp32-3248s035r_ota_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:40	1,7 MB	Archivo...Binary
sunton-4827s043c_full_16MB_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:47	1,7 MB	Archivo...Binary
sunton-4827s043c_ota_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:47	1,6 MB	Archivo...Binary
sunton-8048s043c_full_16MB_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:48	1,7 MB	Archivo...Binary
sunton-8048s043c_ota_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:48	1,6 MB	Archivo...Binary
sunton-8048s050c_full_16MB_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:49	1,7 MB	Archivo...Binary
sunton-8048s050c_ota_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:49	1,6 MB	Archivo...Binary
sunton-8048s070c_full_16MB_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:51	1,7 MB	Archivo...Binary
sunton-8048s070c_ota_v0.7.0-rc13_042fe05.bin	26 ago 2024, 1:51	1,6 MB	Archivo...Binary

Figura 11. Contenido release openHASP

En esta carpeta se encuentran todos los archivos .bin (binarios) compilados para cada uno de los modelos compatibles. La estructura de carpetas y archivos es idéntica para todos los microcontroladores disponibles.

Para comenzar el proceso de carga del *firmware* en la memoria del microcontrolador, es necesario realizar el siguiente montaje con el módulo ESP32:

- VCC a 5V, ya sea a través de una entrada USB o una fuente de alimentación (se recomienda la conexión USB a PC).
- GPIO 0 a GND, junto con la conexión de GND para alimentar la placa.

Tras reiniciar el ESP32 (o forzar el reinicio desconectando y volviendo a conectar la alimentación), el microcontrolador entrará en modo escritura.

El montaje, anteriormente mencionado, realizado en KiCad [56], se muestra en la siguiente figura:

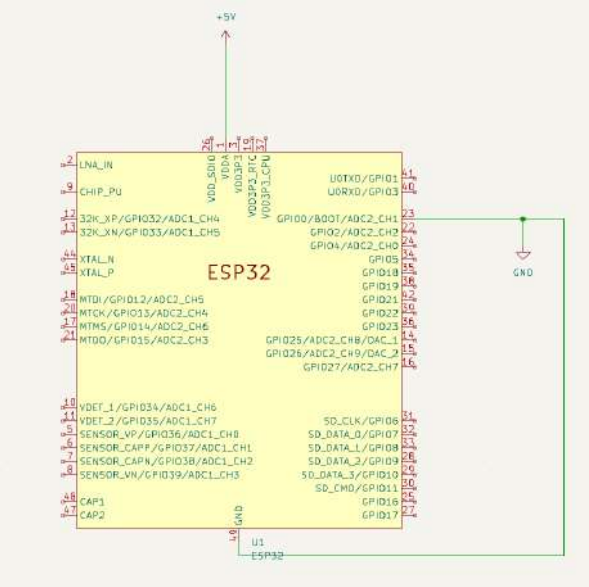


Figura 12. Esquema Kicad

Previamente a que el ESP32 arranque y esté conectado por USB al ordenador encargado de cargar el *firmware* es imprescindible contar con los drivers necesarios para su detección (ver anexo). Una vez detectado por el sistema (esto puede verificarse en el Administrador de Dispositivos), se recomienda anotar el puerto COM [1,2,3,4…] en el que ha sido reconocido.

También es necesario, previamente al *flasheo*, comprobar que el sistema disponga de la herramienta ESPtool.py, la cual suele instalarse junto con el entorno de desarrollo de *Espressif*, y que está accesible en el PATH del sistema (ver anexo).

Si se cumplen todos los requisitos anteriores, se puede ejecutar el siguiente comando:

```
esptool.py --port [COM] erase_flash
```

```
esptool.py --port [COM] --baud 921600 write_flash 0x0 [NOMBRE_FW].bin
```

Donde habrá que sustituir [COM] y [NOMBRE_FW] por el nombre del puerto que aparece en el administrador de dispositivos y el nombre del fichero *firmware* seleccionado, respectivamente. Los comandos para el caso de la guía, son de la siguiente forma.

```
esptool.py --port COM1 erase_flash
```

```
esptool.py --port COM1 --baud 921600 write_flash 0x0 sunton-8048s070c_full_16MB_v0.7.0-rc13_042fe05.bin
```

Estos comandos sobreescriben la memoria de nuestra ESP32, desde la posición 0x0, por el binario compilado descargado de GitHub.

Una vez el proceso termine, se puede volver a un montaje normal del ESP32, es decir, quitando el GPIO0 de GND y arrancando el ESP32.

4.2.2. Instalación automática

Si se utiliza un modelo compatible con openHASP, en versiones recientes es posible cargar los binarios directamente desde un navegador web, sin necesidad de instalar las herramientas de *Espressif*, aunque sigue siendo imprescindible contar con los drivers adecuados. Este método facilita significativamente la carga de una versión del *firmware*.

Para ello openHASP puso en marcha su plataforma *Nightly* [58], accesible desde:

https://nightly.openhasp.com/

La cual tiene el siguiente aspecto,

Version: v0.7.0-rc12 (cd34d89) 2024-05-24

Model	Flash	Display	Touch
AZ-Touch MOD 4 MB		ILI9341	XPT2046
Crowpanel 5"	4 MB	EK9716BD3	GT911
D1 R32 Espduino	4 MB	ILI9341	Analog
D1 R32 Espduino	4 MB	ILI9486	XPT2046
ESP32 D1-mini	4 MB	ILI9341	XPT2046
ESP32 One	4 MB	ILI9486	XPT2046
ESP32-Terminal SPI	16 MB	ILI9488	FT6236
ESP32 Touchdown	4 MB	ILI9488	FT6236
FreeTouchDeck 4 MB		ILI9488	XPT2046
Featherwing 2.4	4 MB	ILI9341	STMPE610
GS-T3E v2.3	16 MB	ST7701S	GT911
Guition ESP32-S3-4848S040	16 MB	ST7701S	GT911
Lanbon L8	8 MB	ST7789V	FT5206
Lilygo Lily Pi	16 MB	ILI9481	FT5206
Lolin D32 Pro v2	16 MB	ILI9341	XPT2046
M5Stack Core2	16 MB	ILI9342C	FT6336U
Makerfabs TFT Touch	16 MB	ILI9488	FT6236
Makerfabs S3 Touch 3.5 SPI	16 MB	ILI9488	FT6236
SenseCap Indicator D1	8 MB	ST7701	FT6336U
Panlee ZW3D95CE01S-AR	16 MB	GC9503V	FT6336U
Sunton 2432S028R	4 MB	ILI9341	XPT2046
TTGO T7 v1.5 4 MB		ILI9341	XPT2046
WT32-SC01 4 MB		ST7796	FT6336U
WT32-SC01 Plus 8 MB		ST7796	FT6336U
WT-86-32-3ZW1	16 MB	ILI9488	GSL1680
Yeacreate Nscreen32	16 MB	ST7796	GT911

Figura 13. Instalador automático nightly

En esta plataforma, de forma configurable, se debe seleccionar la versión deseada en el desplegable superior y elegir el dispositivo correspondiente de la lista. Si el sistema

reconoce un dispositivo compatible conectado al ordenador, aparecerá el botón Install, permitiendo la instalación del *firmware* de manera sencilla.

Siguiendo estos pasos y seleccionando la versión adecuada, openHASP quedará instalado en el ESP32, listo para su configuración.

4.2.3. Configuración inicial

Una vez que el ESP32 ha sido *flasheado*, ya sea mediante el proceso manual o automático, y se dispone de un montaje con una pantalla TFT con protocolo SPI, ya sea ensamblandose manualmente o adquiriendo una solución semi integrada, se puede proceder al arranque y posterior configuración inicial de openHASP.

La primera vez que openHASP se inicia, o en caso de desconfiguración, se generará un AP (*access point*, punto de acceso) al que será posible conectarse para configurar las credenciales de la red. Este acceso a la red será imprescindible tanto para la programación del sistema como para la comunicación con el servidor MQTT, que podrá estar en la misma red local o en un servidor externo, como se explicará en capítulos posteriores.

También, mediante el uso de un dispositivo móvil, es posible escanear el código QR para configurar. Esta pantalla puede observarse en la siguiente figura.

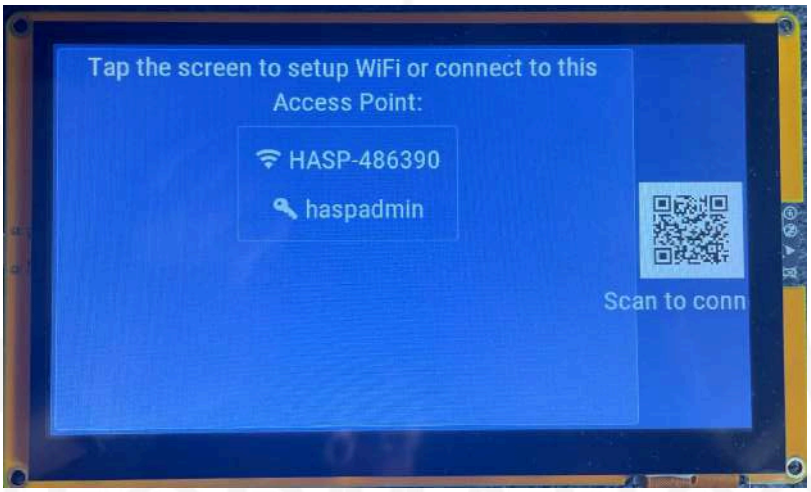


Figura 14. Primer arranque y configuración openHASP

Con lo anterior, se puede conectar el ESP32 a una red WiFi, por lo que el mencionado controlador tendrá asignado una dirección IP (*Internet Protocol*, Protocolo de Internet) por la cual el ESP32 obtendrá una IP por la cual podrá ser accedido al panel de control, como puede verse a continuación:

Figura 15. IP punto de acceso post-configuración

Desde un navegador web se podrá acceder al panel de control mediante, en este caso concreto, la URL (*Unified Resource Locator*, Localizador de recursos unificado):

http://192.168.1.32

Este enlace puede variar en función de la red WiFi a la que se ha conectado el sistema. Y puede variar en algún reinicio del ESP32. En este caso, se volvería a mostrar el mismo pop-up de la anterior figura.

Una vez accedido al ESP que tiene conectada la pantalla por WiFI, se muestra el panel de control, como puede observarse en la siguiente figura.:

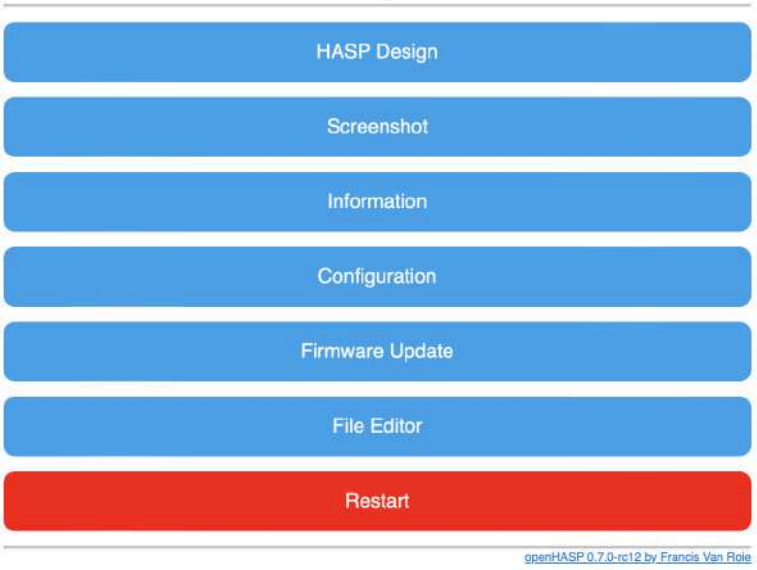


Figura 16. Menú principal openHASP

4.3. Vision general openHASP

En el mismo punto de partida que en la figura anterior se pueden observar todas las opciones disponibles que ofrece la plataforma. Se revisarán cada una de ellas viendo cómo pueden contribuir al desarrollo de un proyecto.

HASP Design: En esta opción del panel de control se pueden configurar los parámetros generales del diseño del SO. Como puede observarse en la siguiente figura:

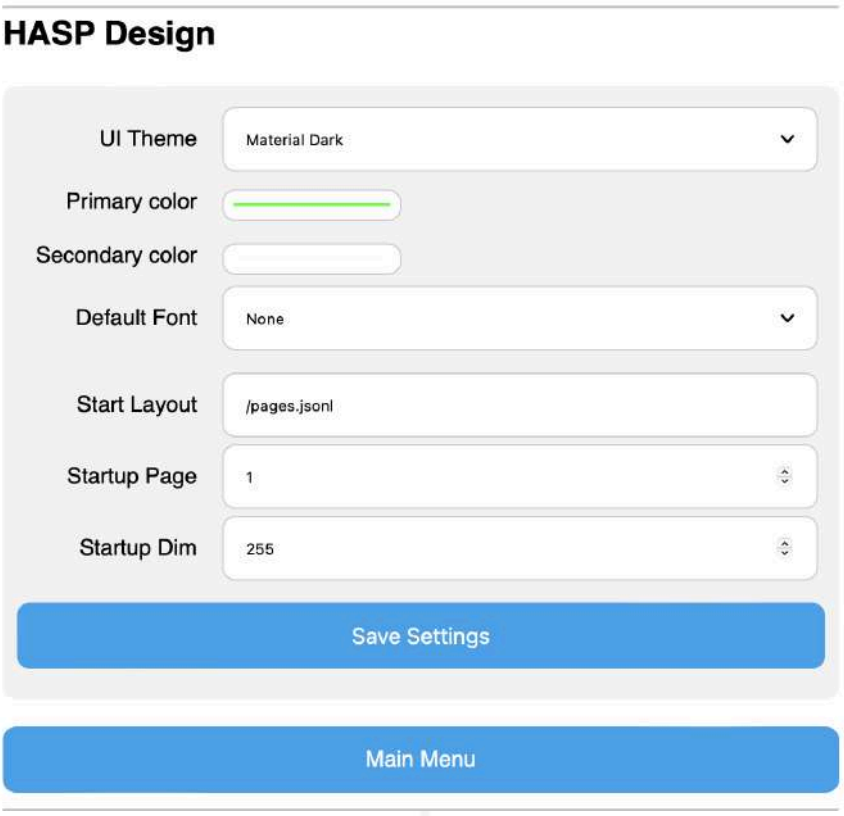


Figura 17. Pestaña de diseño del proyecto

Se pueden decidir parámetros como el tema principal del UI (*User interface*, Interfaz de usuario), eligiendo uno claro, monocromático u oscuro. Después, el color primario y secundario del proyecto, siendo esta selección de colores aplicada automáticamente a algunos elementos que no pueden ser personalizados. Otros parámetros como la fuente o la página .json inicial son importantes aunque pueden dejarse sus valores por defecto, así como el *Startup Page*, página inicial, que verá el usuario nada más encender el dispositivo.

Screenshot: En esta esta pestaña se podrán realizar capturas de pantalla o visualización desde el navegador de un diseño previo de una página. Esta captura se podrá descargar en el dispositivo que accede al panel de control.

Figura 18. Pestaña captura de pantalla (fuente propia)

Information: En esta opción se muestra un resumen detallado del estado del sistema y la configuración actual del dispositivo.

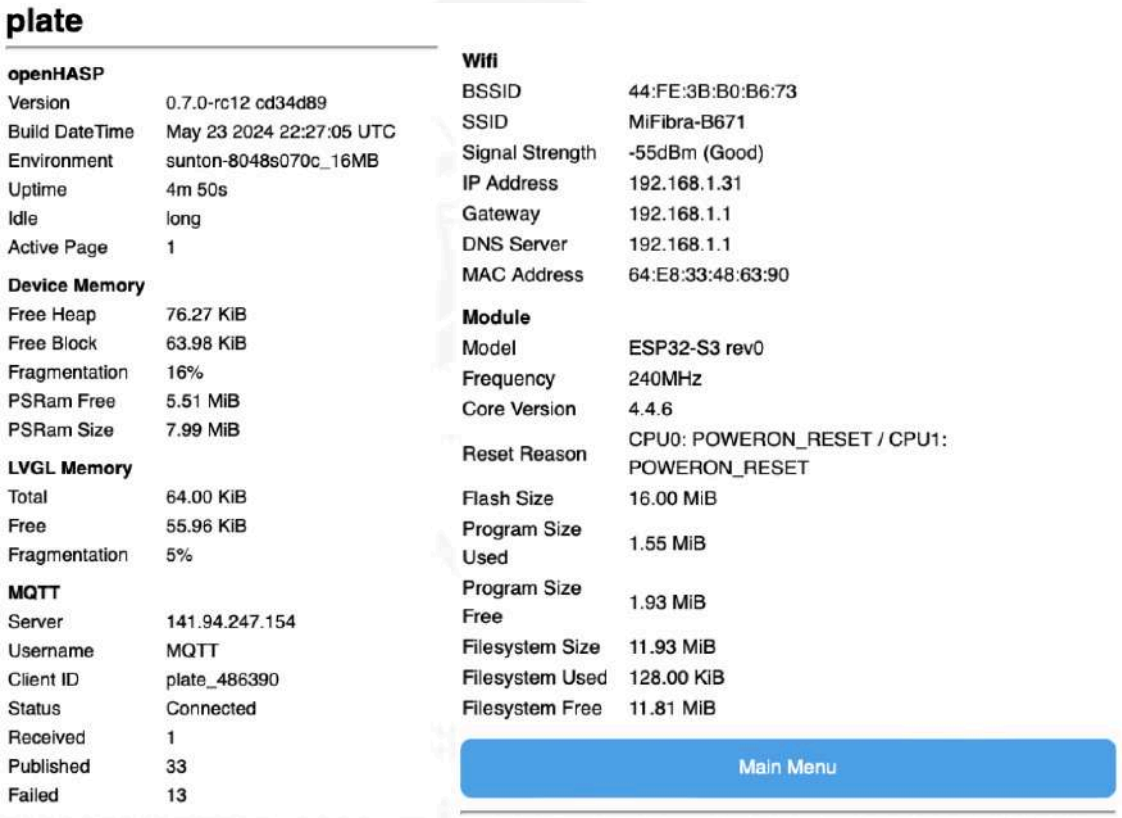


Figura 19. Pestaña logs

Aquí se puede consultar la versión del *firmware* instalada, la dirección IP (*Internet Protocol*, Protocolo de Internet) asignada y el estado de la conexión de red, lo que permite verificar si el dispositivo está correctamente comunicado. También proporciona información sobre el uso de memoria y almacenamiento, facilitando la supervisión del rendimiento del sistema. Además, en caso de contar con periféricos conectados, se podrá visualizar su estado.

Configuration: En este apartado del panel de control se pueden ajustar los distintos parámetros esenciales del sistema, organizados en varias secciones.

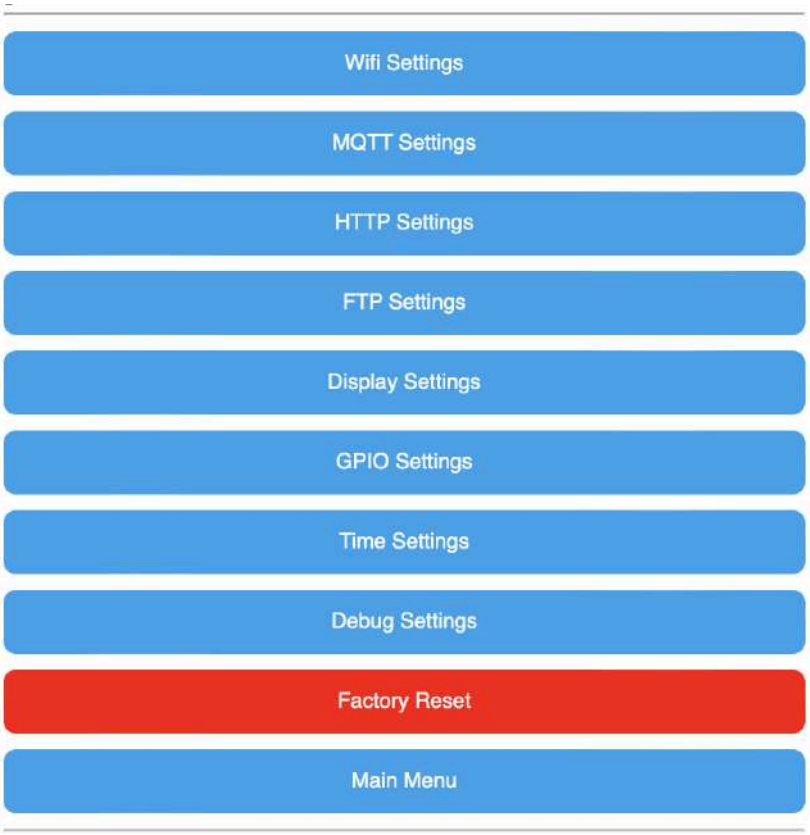


Figura 20. Pestaña menú configuración

Wifi Settings permite gestionar la conexión inalámbrica del dispositivo, configurando redes y credenciales. MQTT Settings se encarga de la configuración del protocolo de mensajería para la comunicación con servidores y dispositivos IoT. *HTTP Settings* y *FTP Settings* permiten definir las opciones de acceso remoto mediante estos protocolos. *Display Settings* ofrece opciones para modificar aspectos visuales como brillo, orientación y resolución. *GPIO Settings* permite gestionar los pines de entrada y salida del hardware para interactuar con sensores y actuadores. *Time Settings* ajusta la hora y sincronización del sistema mediante la configuración de un servidor NTP (*Network Time Protocol*, Protocolo de Hora por Internet) [58]. *Debug Settings* proporciona herramientas para el diagnóstico y resolución de problemas. Finalmente, *Factory Reset* permite restablecer el dispositivo a su configuración de fábrica, eliminando todos los ajustes previos.

Firmware Update: En esta sección del panel de control se puede gestionar la actualización del *firmware* del dispositivo. Se pueden cargar nuevas versiones manualmente o configurar la actualización automática si está disponible.

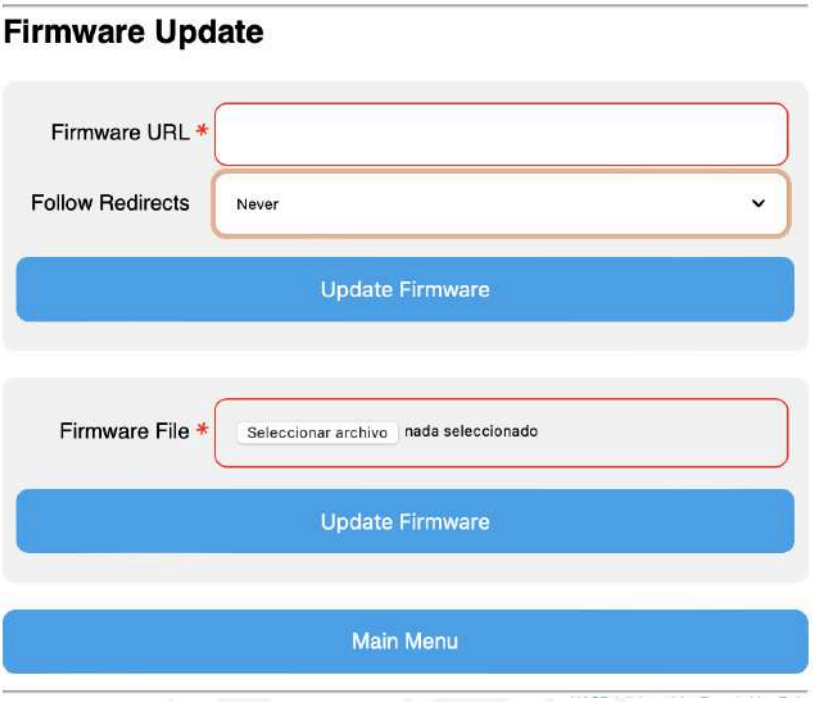


Figura 21. Pestaña actualizacion OTA

Como se puede ver en la anterior figura dependiendo del método de actualización elegido, el usuario puede subir un archivo desde su equipo o realizar la actualización vía OTA (*Over-The-Air*, Por el aire) indicando un enlace a un repositorio de binarios.

File editor: El editor de archivos es una herramienta clave dentro del panel de control, permitiendo la gestión y modificación de archivos de configuración y *scripts* esenciales para el funcionamiento del sistema.

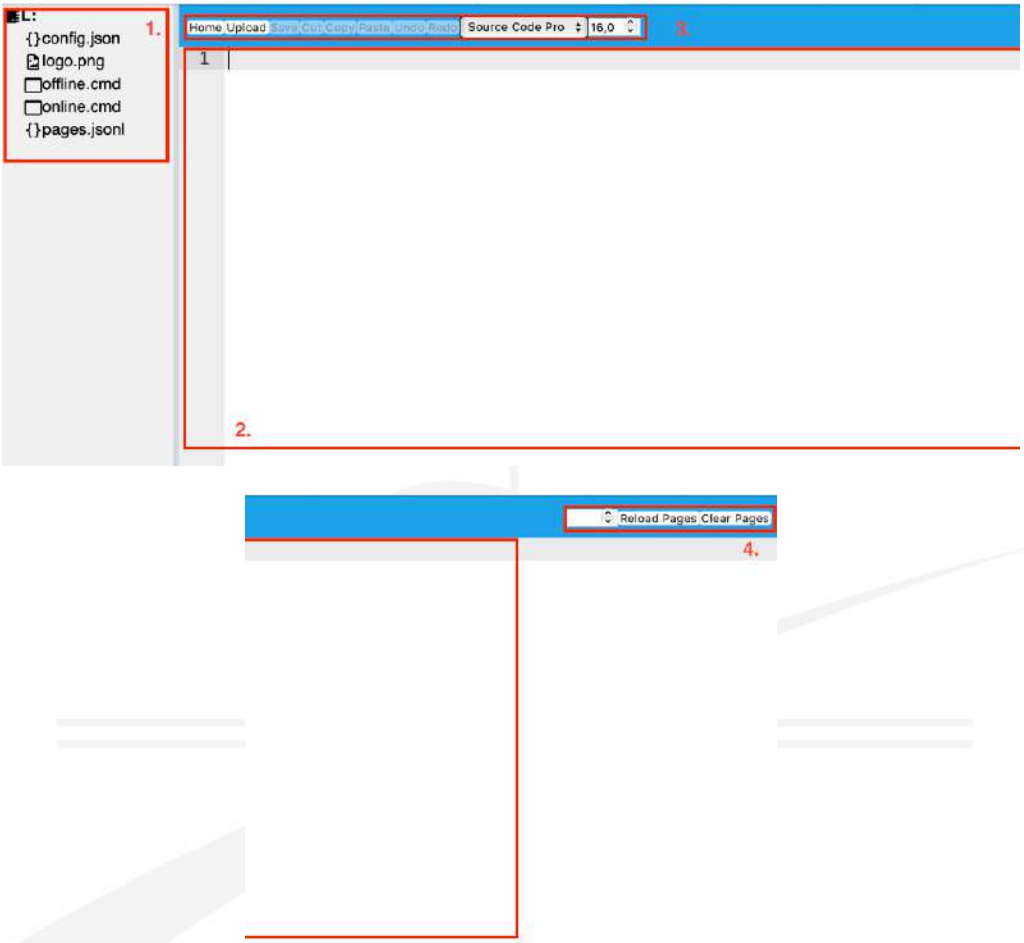


Figura 22. Pestaña File Editor

En la sección (1) se listan los archivos disponibles, como config.json y pages.jsonl, que contiene la configuración y diseño de las páginas del panel.

```
{"page":0,"comment":"---------- All pages ----------"}
{"page":0,"id":1,"obj":"label","x":620,"y":10,"h":60,"w":150,"text":"00.0°C","align":2,"bg_color":"#3
{"page":0,"id":4,"obj":"btn","action":{"down": "page prev"},"x":0,"y":420,"w":267,"h":60,"bg_color":"
{"page":0,"id":5,"obj":"btn","action":{"down": "page back"},"x":267,"y":420,"w":267,"h":60,"bg_color"
{"page":0,"id":6,"obj":"btn","action":{"down": "page next"},"x":534,"y":420,"w":266,"h":60,"bg_color"

{"page":1,"id":1,"obj":"btn","x":0,"y":0,"w":800,"h":60,"text":"UPMTwin","value_font":30,"bg_color":"
{"page":1,"id":2,"obj":"btn","x":20,"y":80,"w":150,"h":120,"toggle":true,"text":"\uE335","text_font":
    "down": "publish hasp/plate/event {\"button\":\"pressed\"}"
  }}
{"page":1,"id":3,"obj":"dropdown","x":20,"y":220,"w":300,"h":60,"options":"Configurar\nResetear"}


{"page":2,"id":34,"obj":"img","src":"L:/logo.png","auto_size":0,"w":800,"x":0,"y":0}

{"page":3,"id":1,"obj":"btn","x":0,"y":0,"w":800,"h":60,"text":"Mensaje","value_font":30,"bg_color":"
{"page":3,"id":24,"obj":"msgbox","text":"Un mensaje con dos botones","options":["Aceptar","Cerrar"]}

{"page":4,"id":3,"obj":"label","x":20,"y":220,"w":300,"h":60,"text":"0.0"}
```

```
{
    "wifi": {
        "ssid": "MiFibra-B671",
        "pass": "********"
    },
    "mqtt": {
        "host": "141.94.247.154",
        "user": "MQTT",
        "pass": "********",
        "topic": {
            "node": "hasp/%hostname%/%topic%",
            "group": "hasp/plates/%topic%",
            "broadcast": "hasp/broadcast/%topic%",
            "hass": "homeassistant/status"
        },
        "port": 1883,
        "name": "plate"
    },
    "telnet": {
        "enable": 1,
        "port": 23
    },
    "mdns": {
        "enable": 1
    },
    "http": {
        "enable": true,
        "port": 80,
        "user": "",
        "pass": "********"
    },
```

Figura 23. Contenido config.json

La zona central (2) del editor permite visualizar y editar el contenido de los archivos seleccionados, proporcionando una interfaz sencilla pero funcional para la personalización. Mientras que, en la parte superior (3), se encuentran herramientas básicas como guardar cambios, subir archivos y opciones de edición como cortar, copiar, pegar, deshacer y rehacer.

También es posible seleccionar el tipo de fuente y tamaño del texto para una mejor legibilidad. En la esquina superior derecha (4), se incluyen botones para recargar o limpiar las páginas editadas, facilitando la gestión de los cambios en tiempo real.

Restart: Por último, opción de reiniciar el dispositivo, en caso de que se desee aplicar una configuración que requiera de un reinicio, o el sistema se haya quedado en un estado de error o confuso.

4.4. Instalación y configuración del servidor MQTT

Uno de los elementos más importantes en el funcionamiento de un dispositivo con el *firmware* openHASP, junto con la conectividad WiFi, es su correcta configuración frente a un servidor MQTT.

Para ello, existen dos opciones principales:

1. Conectarse a un servidor MQTT externo (por ejemplo, upm).
2. Configurar un servidor MQTT propio (con una IP local o fija).

Si se elige la primera opción, se puede omitir la creación del entorno y conectarse directamente al servidor externo.

En esta guía, se ha optado por la segunda opción, configurando un servidor VPS (*Virtual Private Server*, Servidor Virtual privado) [59] con Debian 12.9 accesible remotamente en la IP 141.94.247.154. Esta dirección es accesible desde cualquier conexión WiFi en cualquier parte del mundo, lo que la convierte en una solución versátil para la conectividad de dispositivos IoT de forma distribuida.

Para más detalles sobre la adquisición y configuración de un servidor con estas características, ver el anexo correspondiente.

4.4.1. Configuración del servidor MQTT en VPS

A continuación, se describen los pasos seguidos para instalar y configurar el servidor MQTT en el VPS.

Primeramente se instaló Mosquitto [60], un *broker* (servidor central) MQTT ligero, con los siguientes comandos:

```
sudo apt update
```

```
sudo apt install -y mosquitto mosquitto-clients
```

En el cual primero se actualiza el repositorio Debian de paquetes a la última versión y posteriormente se instalan las últimas versiones de mosquitto y su cliente.

Después, una vez la instalación se ha completado con éxito, se ha editado el archivo de configuración para establecer las reglas de acceso y seguridad:

```
sudo nano /etc/mosquitto/mosquitto.conf

listener 1883

allow_anonymous false

password_file /etc/mosquitto/passwd
```

Estas opciones deshabilitan el acceso anónimo *(allow_anonymous=false)* y el servidor requerirá de autenticación para su uso.

El siguiente comando genera un usuario [MQTT] en el fichero de contraseña especificado en el anterior fichero de configuración y posteriormente su respectiva contraseña:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd MQTT
```

Por último, se ingresa la contraseña [TFG] y posteriormente se reinicia Mosquitto para aplicar los cambios:

```
sudo systemctl restart mosquitto
```

Para verificar que el servidor MQTT está funcionando correctamente, sobretodo el mecanismo de autentificación, se ha utilizado el siguiente comando de prueba:

```
mosquitto_sub -h localhost -t "#" -u MQTT -P TFG -v
```

El cual, en caso de que el servidor esté corriendo de forma correcta, seremos capaces con el comando anterior de suscribirnos y empezar a ver comunicaciones cuando las haya.

4.4.2. Configuración del servidor MQTT en openHASP

Dentro de la página principal de openHASP, accesible desde el navegador por la IP, como se explica en el apartado correspondiente, se seleccionará la siguiente opción, como se puede ver en la siguiente figura:

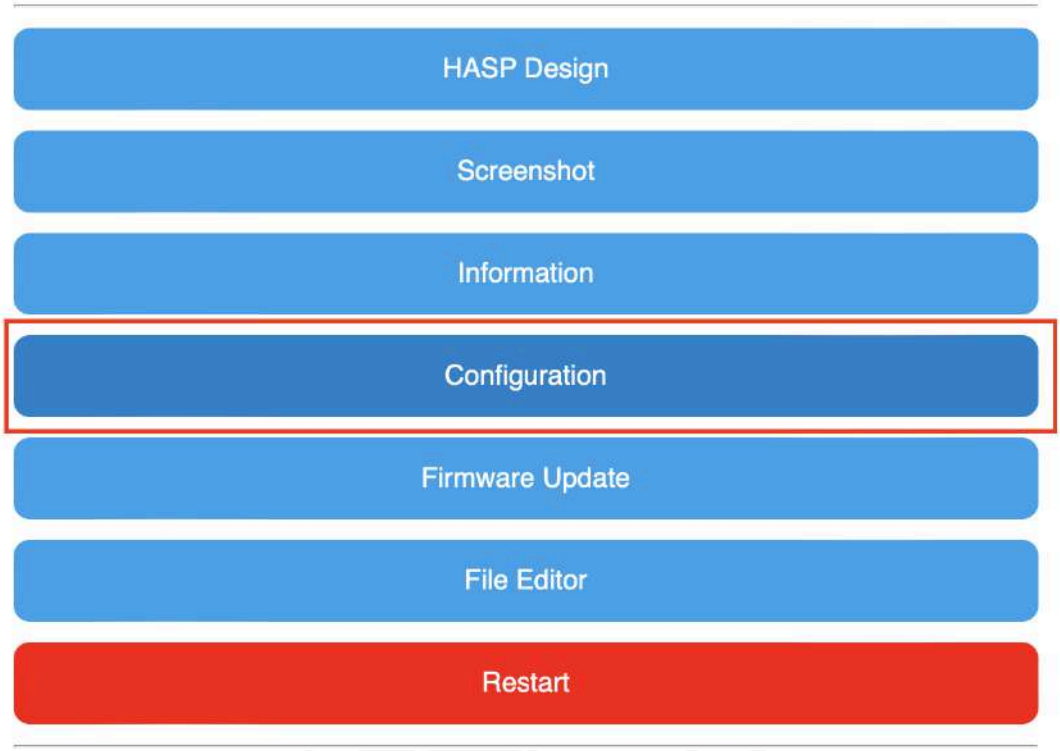


Figura 24. Seleccionar Configuration

También, es accesible desde la url:

http://<ip>/config

Una vez dentro, se observará la siguiente pestaña:

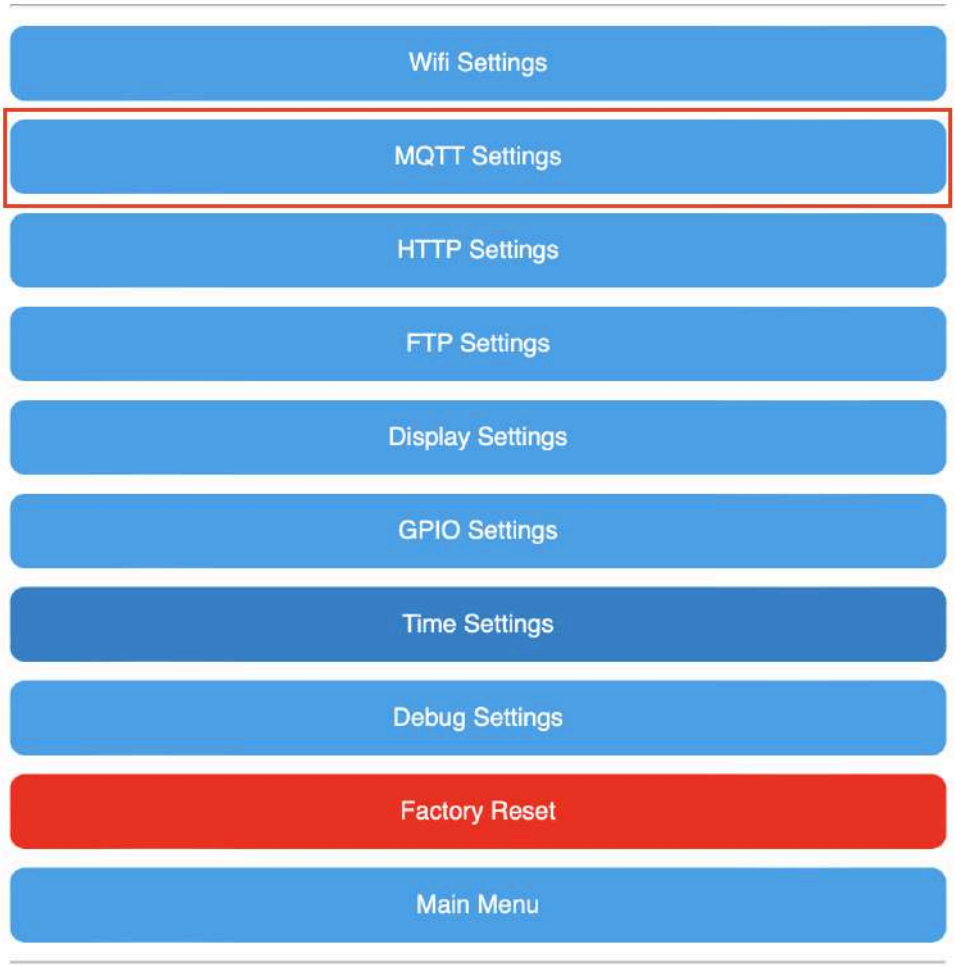


Figura 25. Seleccionar MQTT Settings

En la que se seleccionará, entre todas las opciones de configuración, que ya se han explicado, MQTT.

También accesible desde la URL:

http://<ip>/config/mqtt

Por último, una vez dentro, se configuran las siguientes opciones, el ejemplo puede verse en la siguiente figura:

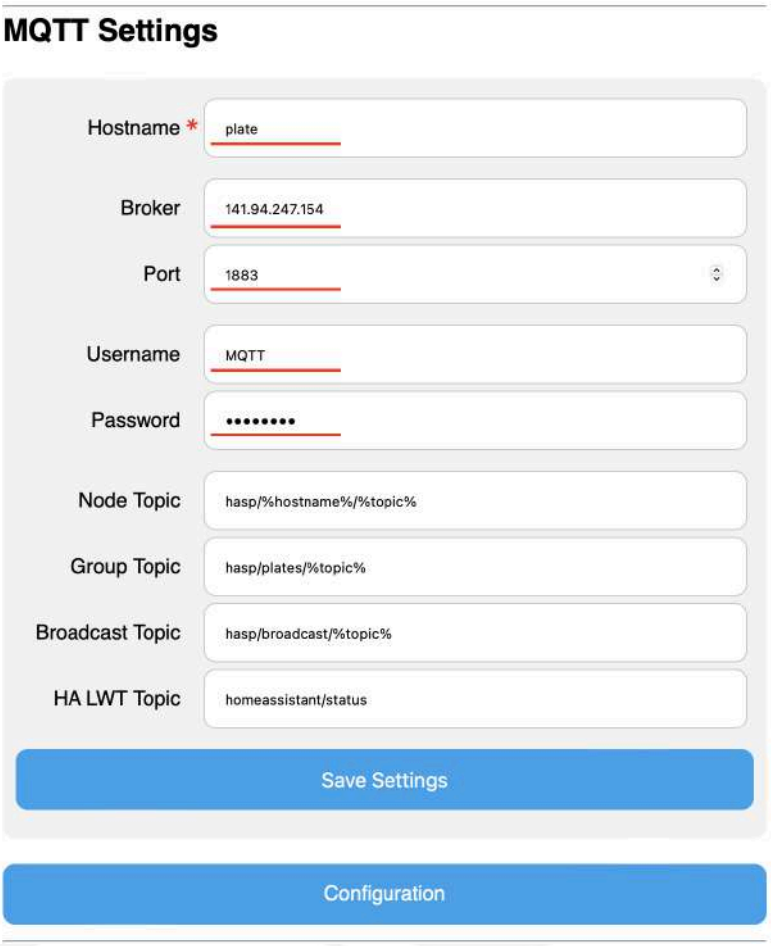


Figura 26. Contenido pestaña MQTT Settings

Que se corresponden a:

- *Hostname*: nombre identificativo dentro del *topic* y grupo de MQTT. Este nombre servirá para identificar el envío y recepción de mensajes, en caso de ser necesario, de más de 1 dispositivo.
- *Broker*: en esta opción se deberá indicar el *endpoint* del servidor MQTT, puede ser una IP local, estática o un nombre DNS por ejemplo mqtt.test.org.
- *Port*: Este es el puerto en el que está levantado el cliente MQTT, debe coincidir con el configurado en el fichero de configuración del apartado anterior.
- *Username*: nombre de usuario configurado.
- *Password*: contraseña de acceso.

Una vez configurado con los datos correspondientes, se guardará la configuración y posteriormente se procede el reinicio del dispositivo.

Para validar que el dispositivo se ha conectado correctamente al servidor se accede a la opción que se señala en la siguiente figura:

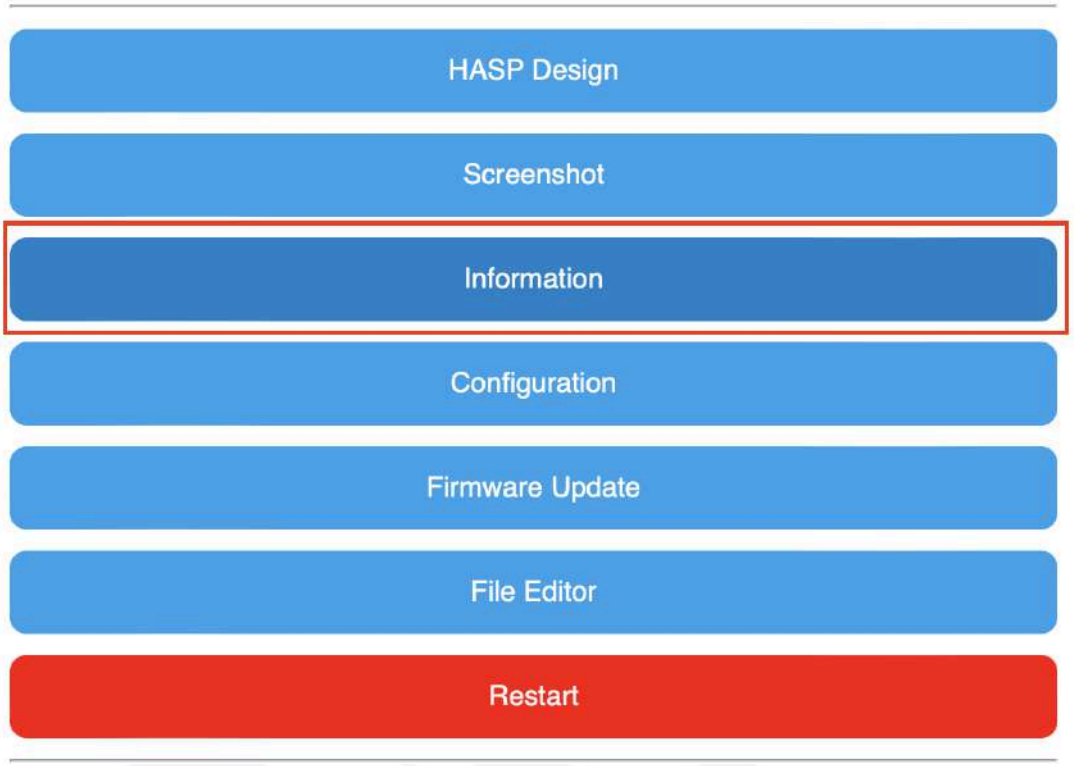


Figura 27. Seleccionar Information

También accesible, como el resto de opciones, a través de:

http://<ip>/info

En la pestaña, aparecerá la siguiente información de la siguiente figura:

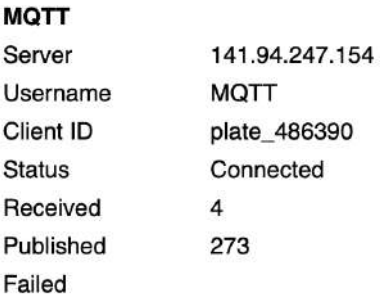


Figura 28. Contenido pestaña Information

Entre toda la información de diversos microservicios del *firmware*, debe incluir el Status de MQTT como Connected. Además aparecen otros datos, muy útiles, como mensajes recibidos, enviados o fallidos.

4.5. Implementación de interfaces básicas

En este apartado, se explora el diseño de interfaces en openHASP: su estructura y organización.

4.5.1. Introducción a la estructura de páginas y objetos en openHASP

El diseño de interfaces dentro de la plataforma tiene dos grandes protagonistas: las páginas y los objetos. Las páginas actúan como contenedores que permiten estructurar los objetos dentro de la interfaz. Los objetos, por su parte, definen acciones, textos y otros elementos visuales, encapsulando una única funcionalidad o diseño.

A continuación se explicará en detalle la funcionalidad de cada uno de ellos.

Páginas

OpenHASP funciona mediante la estructura de un fichero principal, llamado *pages*, con el formato .jsonl. Este formato de texto, con una amplia similitud al JSON estándar, se diferencia porque cada línea del archivo representa un objeto JSON independiente. En otras palabras, JSONL (JSON Lines) es un formato donde cada línea es un JSON autónomo, lo que facilita la carga y procesamiento de datos sin necesidad de leer el archivo completo en memoria.

Cada línea del archivo representa un objeto con un identificador único, y cada uno debe estar separado por un salto de línea (\n), lo que indica al sistema que debe procesar la siguiente línea. Si alguna de estas líneas contiene un error sintáctico, el procesamiento se detendrá en ese punto. En los logs del sistema puede aparecer la última línea procesada, lo que puede dar una pista de donde se encontró el error.

Para que una línea del fichero sea un objeto válido, además de ser sintácticamente correcta, debe contener dos propiedades esenciales: id y obj. El primero es un identificador de tipo entero único para cada objeto en una página. El segundo especifica el tipo de objeto.

Por ejemplo:

```
{"page":1,"id":3,"obj":"obj","x":40,"y":100,"w":160,"h":160,"
radius":100,"opacity":100,"border_opa":160,"border_width":4}
```

Si una línea procesada no tuviera las dos propiedades esenciales mencionadas se tomaría como un comentario de código. Estos comentarios, como en el resto de lenguajes de programación, sirven para hacer aclaraciones sobre el mismo.

En openHASP es útil para diferenciar de una página a otra, puesto que todas aparecen en el mismo fichero pages.jsonl.

```
{"comment":" ----------- Página 1 ------------"}
//objetos
{"comment":" ----------- Página 2 ------------"}
```

La extensión del archivo, así como sus funcionalidades, dependen de la memoria disponible en el microprocesador. En todo momento, es posible visualizar el consumo de memoria en la mencionada pestaña *Information*.

Este archivo pages.jsonl puede editarse directamente en el *File Editor*, como se explicó en el capítulo anterior, o en otro editor de código (por ejemplo, Visual Studio Code) [61], utilizando extensiones para el formateo. Posteriormente, se puede sobrescribir en el directorio raíz "/" del proyecto.

Sin embargo, editar el archivo no es la única forma de definir un objeto o una interfaz en una página. También es posible hacerlo de forma remota mediante comandos, como a través de MQTT, permitiendo la creación y actualización dinámica de elementos. Estos comandos serán abordados en capítulos posteriores, cuando se trate de interfaces más avanzadas.

Objetos

Como se explicó en el apartado anterior, los objetos se insertan en las páginas y encapsulan funcionalidades específicas. A continuación, se presenta una lista de todos los objetos disponibles para su uso, junto con algunas de sus características que pueden emplearse para definir su comportamiento.

En la parte izquierda de la lista se identifica el nombre del objeto, que se usa como referencia en el archivo JSONL. A la derecha, se indica su tipo de funcionamiento.

Los tipos de funcionamiento pueden ser los siguientes:

- ➢ *Value* (Valor): Representa un valor numérico que puede ser mostrado o modificado dentro de la interfaz, como un contador o un indicador de medición.
- ➢ *Toggle* (Alternar): Permite alternar entre dos estados, como un interruptor de encendido/apagado.
- ➢ *Selector* (Selección): Ofrece una lista de opciones entre las cuales el usuario puede elegir una, similar a un menú desplegable.
- ➢ *Range* (Rango): Define un rango de valores ajustables, como una barra deslizante para modificar un parámetro dentro de ciertos límites.
- ➢ *Binary* (Binario): Representa un estado binario (‘0’ o ‘1’), utilizado para variables que solo pueden estar activas o inactivas, tipo booleano.

A continuación se enumeran todos los disponibles desde la versión 0.6.3 a modo de cheatsheet [62].

Objeto	Tipo
btn	Binary
switch	Toggle
checkbox	Toggle
label	Visual
led	Visual
spinner	Visual
obj	Visual
line	Visual
img	Visual
cpicker	Selector
roller	Selector
dropdown	Selector

Objeto	Tipo
btnmatrix	Selector
msgbox	Selector
tabview	Selector
tab	Selector
bar	Range
slider	Range
arc	Range
linemeter	Range
gauge	Range
qrcode	Visual

Tabla 5. Lista de objetos openHASP

Todos estos objetos tienen propiedades únicas, pero la mayoría de las esenciales son comunes entre todos. En la tabla siguiente se especifican estas últimas.

Propiedad	Valor	Por defecto	Descripción
id	1..254	n/a	ID único del objeto.
obj	string	n/a	Nombre del objeto.
page	0..12	n/a	Localización en pagina del objeto
groupid	0..15	0 (none)	GPIO conectado al objeto.
x	int16	0	Posición horizontal.
y	int16	0	Posición vertical.

Propiedad	Valor	Por defecto	Descripción
w	int16	0	Ancho del objeto.
h	int16	0	Alto del objeto.
enabled	bool	true	Habilitar o deshabilitar su uso.
hidden	bool	false	Oculto.
opacity	uint8	255	Opacidad
swipe	JSONobject	null	Acción al deslizar.
action	JSONobject	null	Acción al activar el objeto.
click	bool	true	Habilitar el objeto como pulsable.

Tabla 6. Atributos comunes objetos

Se han eliminado algunas propiedades debido a su poco uso y compatibilidad, enumerando únicamente las más utilizadas en el desarrollo de interfaces.

A continuación se detalla cada uno de ellos, así como sus atributos únicos y, además, una figura de la guia oficial de objetos de openHASP.

Botón (obj=btn)

El objeto botón es útil cuando se quiere aceptar alguna solicitud desde el dispositivo a conectar o provocar alguna acción en este. Es el objeto más básico en la creación de interfaces con openHASP [1].

Un ejemplo de cómo sería la creación de un boton seria de la siguiente forma:

```
{"page":1,"id":2,"obj":"btn","x":10,"y":40,"w":105,"h":90,"toggle"
:true,"text":"Button","mode":"break","align":"center"}
```

El cual quedaría como se puede ver en la siguiente figura:

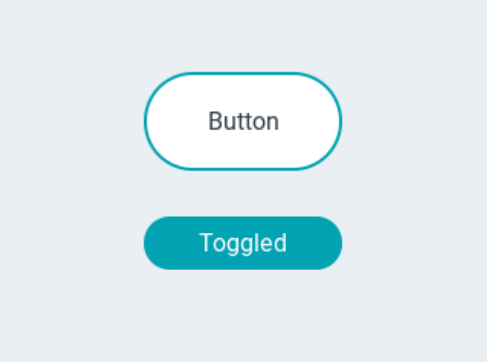


Figura 29. Objeto botón

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
mode	string	expand	Modo de visualización del texto del objeto. `expand` – El tamaño del objeto se ajusta al texto `break` – El ancho se mantiene fijo y el texto se ajusta horizontalmente `dots` - El tamaño se mantiene fijo y en caso de que el texto sea más largo añade puntos suspensivos (...) `scroll` - El texto, en caso de ser más grande, realiza un movimiento horizontal. `loop` - El movimiento es similar al del scroll pero circular. `crop` - Corta el texto sin ninguna acion.
align	string	left	Alineación del texto dentro del objeto

Tabla 7. Atributos objeto botón

Interruptor `(obj=switch)`

El interruptor, o *switch*, realiza una función similar a un interruptor físico. Puede utilizarse para el apagado o encendido de alguna acción, que puede tener efectos en un

dispositivo, por ejemplo, de una luz. A diferencia del botón, este estado se mantiene constante mientras que el interruptor este hacia el lado del valor que se haya indicado como "Encendido".

Un ejemplo de cómo sería la creación de un interruptor seria de la siguiente forma:

```
{"page":1,"id":4,"obj":"switch","x":125,"y":145,"w":105,"h":55,"radius":15}
```

El cual quedaría como se puede ver en la siguiente figura:

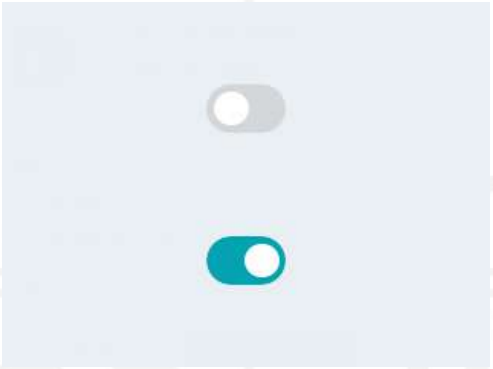


Figura 30. Objeto interruptor

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
bg_color10	color	0	Color del fondo del interruptor.
bg_color20	color	0	Color del "punto" o palanca del interruptor
radius	int16	Por defecto en tema.	Radio del interruptor.

Tabla 8. Atributos objeto Interruptor

Marcador `(obj=checkbox)`

El marcador, o *checkbox*, sirve para aceptar o marcar alguna condición. Suele usarse, como en el ejemplo de la imagen, para términos y condiciones, pero más comúnmente usado para marcar tareas, por ejemplo, en una lista.

Un ejemplo de cómo sería la creación de un marcador seria de la siguiente forma:

```
{"page":1,"id":5,"obj":"checkbox","x":10,"y":145,"w":105,"text":"
Checkbox"}
```

El aspecto de un marcador, como ya se acostumbra a ver en la mayoría de sitios web, se puede ver en la siguiente figura:

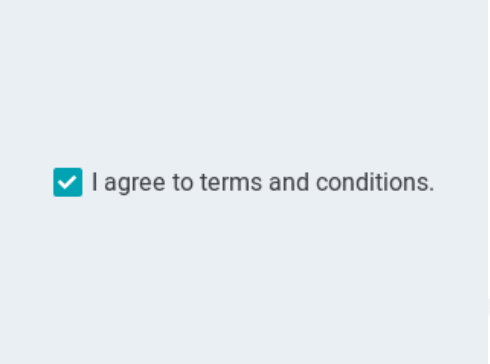


Figura 31. Objeto marcador

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
text	string	"Checkbox"	Texto a mostrar en el objeto.

Tabla 9. Atributos objeto Marcador

Barra de progreso `(obj=bar)`

La barra de progreso, como indica su nombre, sirve para indicar un valor, normalmente asociado a un progreso, de forma horizontal.

Un ejemplo de cómo sería la creación de una barra de progreso sería de la siguiente forma:

```
{"page":1,"id":5,"obj":"bar","x":10,"y":145,"w":105}
```

El cual quedaría como se puede ver en la siguiente imagen:

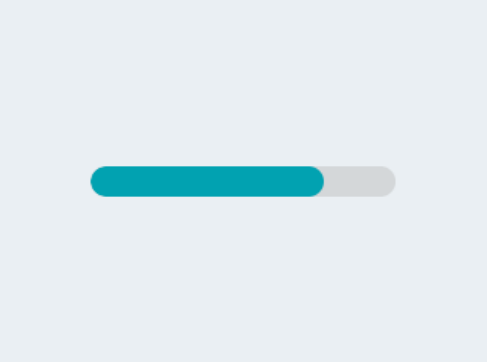


Figura 32. Objeto barra de progreso

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
min	int16	0	Valor mínimo
max	int16	100	Valor máximo
val	int16	0	Valor actual del progreso
start_value	int16	0	Valor inicial del progreso

Tabla 10. Atributos objeto Barra de progreso

Deslizador `(obj=slider)`

El deslizador, o *slider*, tiene una función similar a la barra de progreso, pero en este caso, es el usuario el cual, de forma táctil, selecciona un valor mediante un efecto visual de una barra horizontal. Normalmente suele ser para valores no muy precisos, puesto que el usuario no ve exactamente el valor seleccionado.

El objeto se podría crear de la siguiente manera:

```
{"page":1,"id":5,"obj":"slider","x":10,"y":145,"w":105}
```

Y este tendría el aspecto en la interfaz como se puede ver en la figura a continuación:

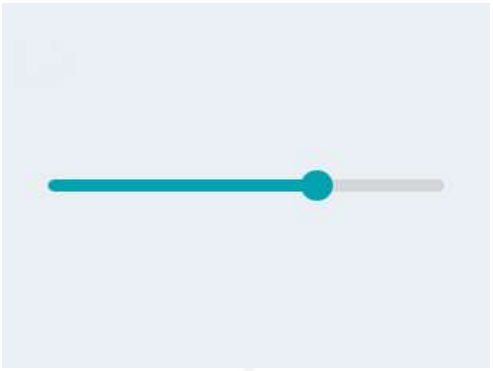


Figura 33. Objeto Deslizador

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
min	int16	0	Valor mínimo
max	int16	100	Valor máximo
val	int16	0	Valor actual del indicador
start_value	int16	0	Valor inicial del indicador

Tabla 11. Atributos objeto Deslizador

Arco `(obj=arc)`

El arco funciona de manera similar a un *slider*, se puede tanto mostrar valores, como una barra de progreso, cómo seleccionar valores. La diferencia es su forma circular en 360 grados.

Un ejemplo de cómo sería la creación de una deslizador sería de la siguiente forma:

```
{"page":1,"id":5,"obj":"arc","x":10,"y":145,"w":105}
```

El cual quedaría como se puede ver en la siguiente imagen:

Figura 34. Objeto Arco

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
min	int16	0	Mínimo valor
max	int16	100	Máximo valor
val	int16	0	Valor actual.
rotation	int16	0	Diferencia con posición inicial.
type	0-2	0	0 = normal, 1 = symmetrical, 2 = reverse
adjustable	bool	false	Añadir un seleccionador para habilitar cambiar el valor.
start_angle	0-360		Posición inicial
end_angle	0-360		Posición final.
start_angle10	0-360		Posición inicial del seleccionador.
end_angle10	0-360		Posición final del seleccionador.

Tabla 12. Atributos objeto Arco

Lista desplegable `(obj=dropdown)`

La lista desplegable sirve para habilitar al usuario de elegir una opción parametrizada. Posteriormente puede obtenerse esta opción para realizar alguna opción en el microcontrolador a conectar. Las opciones a elegir no se muestran, y puede seleccionarse

tanto un texto definido o la primera de la lista. Es una buena opción para ahorrar espacio y recursos de la interfaz pero tener opciones parametrizadas seguras.

Un ejemplo de cómo sería la creación de una deslizador sería de la siguiente forma:

```
{"page":1,"id":10,"obj":"dropdown","x":10,"y":205,"w":105,"h":30,"
options":"Apple\nBanana\nOrange\nMelon"}
```

El cual quedaría como se puede ver en la siguiente imagen:

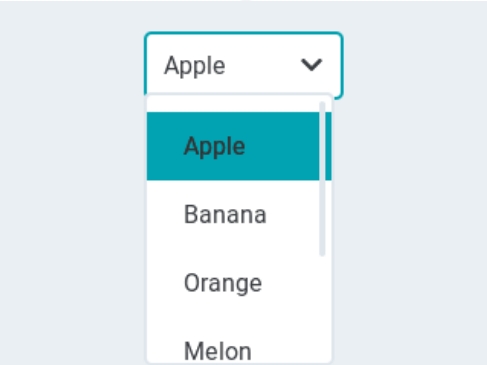


Figura 35. Objeto Lista desplegable

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
options	string	" "	Lista de opciones
val	int16	0	Número de la opción elegida
text	string	""	Texto de la opción elegida
direction	byte	0	Dirección a la que se expande (abre) el desplegable. 0 = down, 1 = up, 2 = left, 3 = right
show_selected	bool	true	Mostrar un texto por defecto o la primera opción en lista.
max_height	int16	3/4 of screen height	Tamaño de la lista desplegada.

Tabla 13. Atributos objeto Lista desplegable

Rodillo `(obj=roller)`

El rodillo, o roller, tiene el mismo objetivo que la lista desplegable, pero visualmente es distinto. Por defecto se muestra una lista desplegada de varias opciones, y el usuario puede deslizar de forma infinita entre las opciones.

Ejemplo de cómo sería el JSON Line del objeto rodillo:

```
{"page":1,"id":10,"obj":"dropdown","x":10,"y":205,"w":105,"h":30,"
options":"Enero\nFebrero\nMarzo\nAbril\n…"}
```

Obteniendo como resultado lo mostrado en la siguiente figura:

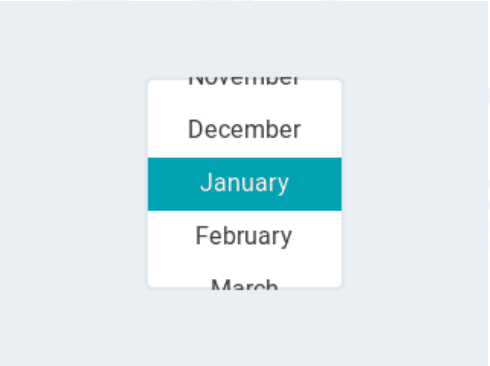


Figura 36. Objeto Rodillo

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
options	string	" "	Lista de opciones
val	int16	0	Número de la opción elegida
text	string	" "	Texto de la opción elegida
rows	in8	0	Número de opciones inicialmente visible
mode	0..1	0	0 modo normal, 1 modo infinito.
align	string	center	Alineación del texto.

Tabla 14. Atributos objeto Rodillo

Medidor en arco `(obj=linemeter)`

El medidor en forma de arco, *linemeter*, tiene propiedades similares al Arco, anteriormente visto, con la diferencia de que puede indicar valores de forma más precisa. La selección por parte del usuario en este objeto no es posible ni configurable, es decir, su único uso es para mostrar información no para introducirla.

El medidor puede referenciarse en el JSONL de la siguiente forma:

```
{"page":1,"id":12,"obj":"linemeter","x":20,"y":70,"w":200,"h":200,
"value_str":"Temp","val":75,"line_count":35,"line_rounded":1}
```

Quedando un aspecto como el de la siguiente figura:

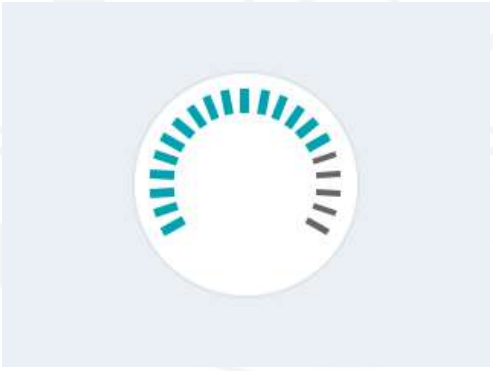


Figura 37. Objeto Medidor en arco

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
min	int16	0	Valor mínimo.
max	int16	100	Valor máximo.
val	int16	0	Valor actual.
angle	0-360	240	Ángulo entre inicio y final.
line_count	uint16	31	Conteo de ticks.
rotation	0-360	0	Diferencia de valor en rotación
type	0-1	0	0 para sentido horario 1 para antihorario.

Tabla 15. Atributos objeto Medidor en arco

Manómetro `(obj=gauge)`

El manómetro, como en su funcionamiento físico, sirve para indicar valores, por ejemplo presion de agua, temperatura, etc, con avisadores, en forma de color, de sectores criticos (una temperuta alta, presion baja/alta, etc…).

Un ejemplo de cómo sería la creación de un manómetro sería de la siguiente forma:

```
{"page":1,"id":13,"obj":"gauge","x":20,"y":70,"w":200,"h":200}
```

El cual quedaría como se puede ver en la siguiente imagen:

Figura 38. Objeto Manómetro

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
min	int16	0	Valor mínimo.
max	int16	100	Valor máximo.
val	int16	0	Valor actual.
critical_value	int16	80	Valor a activar el color crítico.

Tabla 16. Atributos objeto Manómetro

Visor de pestañas `(obj=tabview)`

El objeto pestaña, también llamado *tabview*, permite encapsular en un mismo espacio de la interfaz múltiples objetos pestañas en la misma página. Este objeto crea el espacio donde se añaden los objetos de tipo *Tab*.

Un ejemplo de cómo sería la creación de un visor de pestañas (y las derivadas pestañas) sería de la siguiente forma:

```
{"page":1,"id":14,"obj":"tabview","btn_pos":1,"y":180}
{"page":1,"id":51,"obj":"tab","parentid":14,"text":"Tab 1"}
{"page":1,"id":52,"obj":"tab","parentid":14,"text":"Tab 2"}
{"page":1,"id":53,"obj":"tab","parentid":14,"text":"Tab 3"}
{"page":1,"id":61,"obj":"switch","x":20,"y":10,"w":60,"h":30,"parentid":51,"radius":25,"radius20":25}
{"page":1,"id":71,"obj":"dropdown","x":15,"y":10,"w":110,"h":30,"parentid":52,"options":"Apple\nBanana\nOrange\nMelon"}
{"page":1,"id":81,"obj":"checkbox","x":15,"y":10,"w":110,"h":30,"parentid":53,"text":" Nice tabview"}
```

El cual quedaría como se puede ver en la siguiente imagen:

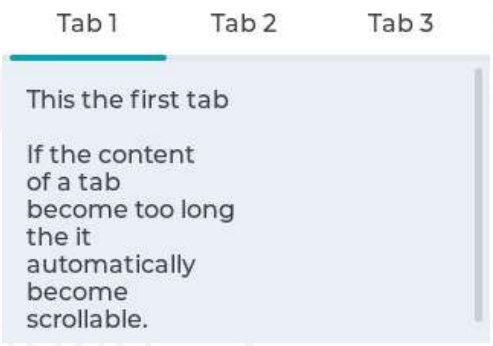


Figura 39. Objeto Visor de pestañas

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
Visor de pestañas			
val	int8	0	Pestaña inicial
text	string	“ “	Nombre de la pestaña activa.
btn_pos	0..4	1	Posición del botón seleccionador de pestañas.

Propiedad	Valor	Por defecto	Descripción
count	uint16	0	Devuelve el número de pestañas activas.
Pestañas			
parenid	int8	0	El ID del objeto Visor al que se adhiere la pestaña
text	string	"Tab"	Nombre de la pestaña.

Tabla 17. Atributos objeto Visor de pestañas

Seleccionador de color `(obj=cpicker)`

El seleccionador de color, *cpiker*, se usa en entornos normalmente luminosos. Este color se selecciona por el usuario y luego puede obtenerse para ajustar algún color en el elemento conectado.

Un ejemplo de cómo sería la creación de un seleccionador de color sería de la siguiente forma:

```
{"page":1,"id":13,"obj":"cpicker","x":20,"y":70}
```

El cual quedaría como se puede ver en la siguiente imagen:

Figura 40. Objeto Seleccionador de color

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
color	color	0	Color seleccionado en formato html #rrggbb

Propiedad	Valor	Por defecto	Descripción
scale_width	unit16	25	El ancho del gradiente.
pad_inner	int16	10	La separación entre el círculo seleccionador y la vista previa
mode	string	"hue"	El aspecto del color editado. "hue", "saturation", "value".

Tabla 18. Atributos objeto Seleccionador de color

Spinner `(obj=spinner)`

El *spinner* se conoce como un objeto giratorio. Normalmente se usa para indicar que un proceso está en marcha. Los usuarios lo asemejan a tiempos de carga, como una web, pero en entornos embebidos puede ser el tiempo desde que se ordena una acción hasta que el dispositivo embebido completa su trabajo.

Un ejemplo de cómo sería la creación de un *spinner* de color sería de la siguiente forma:

```
{"page":1,"id":15,"obj":"spinner","x":180,"y":50,"w":36,"h":36,"bg_opa":0,"border_width":0,"line_width":6,"line_width10":6,"angle":80,"line_color":"white","line_color10":"green"}
```

El cual quedaría como se puede ver en la siguiente imagen:

Figura 41. Objeto Spinner

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			

Propiedad	Valor	Por defecto	Descripción
speed	int16	1000	Velocidad de giro por segundo
direction	int16	0	0 = sentido horario, 1 = sentido antihorario
angle	0-360	60	Ángulo de giro.
type	0-2	0	0 = lento arriba, 1 = estrechamiento, 2 = velocidad constante
line_width	int16	20	Ancho de la línea de fondo..
line_width10	int16	20	Ancho del segmento giratorio.
line_color	color	Depende del tema	Color del fondo.
line_color10	color	Depende del tema	Color del segmento giratorio

Tabla 19. Atributos objeto Spinner

Indicador LED `(obj=led)`

El indicador LED es un objeto de solo input desde el microcontrolador acoplado, que puede indicar el estado de un accesorio mediante una intensidad de color.

Un ejemplo de cómo sería la creación de un indicador led sería de la siguiente forma:

```
{"page":1,"id":15,"obj":"led","x":180,"y":50,"w":36,"h":36}
```

El cual quedaría como se puede ver en la siguiente imagen:

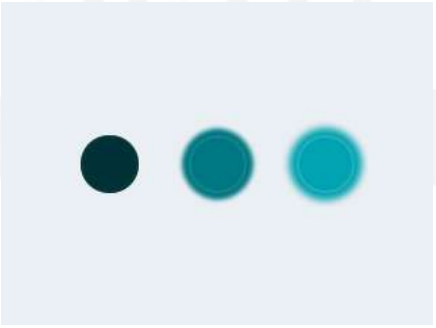


Figura 42. Objeto Indicador LED

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
val	byte	0	Intensidad lumínica [0..255]

Tabla 20. Atributos objeto Indicador LED

Matriz de botones `(obj=btnmatrix)`

La matriz de botones, o *btnmatrix*, sirve para que el usuario pueda introducir valores, de forma controlada. Puede usarse para seleccionar opciones alfanuméricas, una a la vez, o por ejemplo, seleccionar varias, como un código de acceso.

Un ejemplo de cómo sería la creación de un matriz de botones sería de la siguiente forma:

```
{"page":1,"id":20,"obj":"btnmatrix","x":10,"y":10,"w":220,"h":150,
"options":["#FF0000 Red Text#","#0000FF Cyan Text#","\n","#FFFF00
Yellow Text#"],"toggle":1,"one_check":1}
```

El cual quedaría como se puede ver en la siguiente imagen:

Figura 43. Objeto Matriz de botones

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			

Propiedad	Valor	Por defecto	Descripción
options	json array	"Text"	Opciones a introducir diferenciadas por \n
align	string	center	Alineación del texto.
toggle	bool	false	Los botones se mantienen pulsados o actúan como un botón normal.
one_check	bool	false	Solo se permite la activación de un botón a la vez.
val	int8	0	El número de botones activados al inicio.

Tabla 21. Atributos objeto Matriz de botones

Cuadro de mensaje `(obj=msgbox)`

El cuadro de mensaje, *msgbox*, suele usarse como *pop-up*, es decir, un cuadro de alerta que se activa mediante una acción. Este cuadro de mensaje puede usarse solo para información, por ejemplo, openHASP ha obtenido IP (viene por defecto) y se notifica, o para aplicar alguna opción ante la alerta.

Un ejemplo de cómo sería la creación de cuadro de mensaje sería de la siguiente forma:

```
{"page":1,"id":24,"obj":"msgbox","text":"A message box with two buttons","options":["Apply","Close"]}
```

El cual quedaría como se puede ver en la siguiente imagen:

Figura 44. Objeto Cuadro de mensaje

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
text	string	“ “	Texto a mostrar
options	json array	[“OK”]	Opciones a mostrar diferenciadas por \n
auto_close	int16	0	Después de los milisegundos seleccionados, el mensaje se cierra automáticamente.

Tabla 22. Atributos objeto Cuadro de mensajes

Línea `(obj=line)`

El objeto línea, *line*, es una ilustración básica. No tiene un uso concreto o definido, pero puede usarse para, por ejemplo, representar una variación de un dato de forma de gráfica.

Un ejemplo de cómo sería la creación de cuadro de mensaje sería de la siguiente forma:

```
{"page":1,"id":33,"obj":"line","points":[[10,25],[100,25],[100,0]]
,"auto_size":0,"y_invert":1}
```

El cual quedaría como se puede ver en la siguiente figura:

Figura 45. Objeto Línea

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
points	json array	[“OK”]	Array de las coordenadas de los puntos que forman la línea [[X,Y],[X,Y]].
auto_size	bool	true	El tamaño se ajusta de forma automática.
y_invert	bool	false	El eje Y invertido.

Tabla 23. Atributos objeto Línea

Imagen `(obj=img)`

El objeto imagen, *img*, sirve para presentar una imagen en una parte de la interfaz. Puede usarse para mostrar cualquier tipo de ilustración, como un logo, una imagen de una cámara de seguridad, etc...

Un ejemplo de cómo sería la creación de una imagen sería de la siguiente forma:

```
{"page":1,"id":34,"obj":"img","src":"L:/image.png","auto_size":0,"w":50}
```

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
src	string	" "	Dirección de la imagen.
auto_size	bool	true	Selección automática de imagen.
offset_x	int16	0	Aplicar recorte horizontal
offset_y	int16	0	Aplicar recorte vertical.
zoom	uint16	256	Aplicar zoom.
angle	int16	0	Rotar imagen.
pivot_x	int16	H center	Pivotar imagen desde un punto X horizontal.

Propiedad	Valor	Por defecto	Descripción
pivot_y	int16	V center	Pivotar imagen desde un punto Y vertical.
antialias	bool	false	Ajustar calidad de imagen a transformaciones como zoom o rotación.

Tabla 24. Atributos objeto Imagen

Codigo QR `(obj=qrcode)`

El objeto QR puede usarse para generar de forma automática un QR que puede ser escaneado por el usuario.

Un ejemplo de cómo sería la creación de código QR sería de la siguiente forma:

```
{"page":1,"id":42,"obj":"qrcode","text":"www.upm.es/0.7.0/design/objects/","x":520,"y":20,"size":200}
```

Propiedad	Valor	Por defecto	Descripción
Todas las anteriormente vistas en la tabla de propiedades comunes.			
text	string	“ “	El texto a generar el código QR. Normalmente un enlace.
size	int16	140	Tamaño del QR generado. Siempre será un cuadrado.

Tabla 25. Atributos objeto Codigo QR

Fuentes

OpenHASP permite subir fuentes personalizadas en cada proyecto. Esta característica puede ser útil cuando se requiere usar fuentes con caracteres especiales, tales como emoticonos, símbolos específicos, caracteres propios de un idioma, o simplemente fines decorativos para mejorar la estética [63].

Para añadir fuentes personalizadas, es necesario subir al dispositivo los archivos correspondientes, preferiblemente con nombres sencillos y fáciles de identificar. Las fuentes cargadas estarán luego disponibles para su uso en los objetos definidos en pages.jsonl.

Existen numerosas páginas web que ofrecen al desarrollador un buscador personalizado de fuentes, como puede ser, una de las mas famosas [64]:

https://www.1001fonts.com/

que ofrecen fuentes compatibles para openHASP.

El proceso para incluir una fuente personalizada en la pantalla es sencillo. Primero, se debe ir al apartado de *File Browser* en el panel web de openHASP y seleccionar la opción *Upload*, tal como se explicó en el apartado anterior.

Una vez se haya completado la subida del fichero con extensión *.ttf* aparecerá en la parte izquierda del *File Editor* en el visor de documentos.

Figura 46. Arbol de ficheros raíz

Además de fuentes, openHASP incluye un conjunto de iconos vectorizados que pueden utilizarse sin necesidad de consumir recursos adicionales, ya que vienen incorporados directamente en el *firmware*. Estos iconos están disponibles para el desarrollador y resultan útiles para distintos proyectos y escenarios.

El acceso a estos iconos se realiza mediante códigos específicos, como los que se muestran en la siguiente figura:

Figura 47. Logos UNICODE incluidos

Para utilizarlos, basta con referenciar su código correspondiente dentro de cualquier objeto con el atributo texto, como por ejemplo:

```
"text":"\uE438"
```

4.5.2. Creación de una página inicial simple

Una vez analizada la estructura del *firmware* openHASP, configurado su entorno básico y estudiados todos sus componentes agrupados en páginas y objetos, es momento de comenzar con el desarrollo de un espacio de trabajo básico.

Para ello, se emplearán elementos vistos en puntos anteriores, como la estructura del fichero pages.jsonl, páginas y objetos básicos, tales como *label* para textos o *btn* para botones simples.

Una página inicial básica debe contener algunos elementos clave, entre ellos comentarios que ayudan al desarrollador a organizar su proyecto, un *layout* identificativo del mismo (que puede incluir el logo, nombre e información relevante como hora, temperatura, etc.), y, si lo requieren las especificaciones, un navegador de páginas que permita al usuario desplazarse cómodamente entre las distintas páginas.

Estos elementos, que permanecerán estáticos a lo largo de las infinitas páginas que puede tener el proyecto, pertenecerán a la página 0, lo que lo convierte en una característica fija.

Navegación de Páginas

En caso de que el proyecto a desarrollar permita la inserción de inputs por parte del usuario (es decir, que no se trate simplemente de una pantalla destinada a mostrar información), el desarrollo de la página inicial puede comenzar por un navegador entre páginas. Esto ayudará a navegar fácilmente entre los distintos elementos de las páginas sin necesidad de utilizar el panel de control (durante el desarrollo) o dar al usuario un espacio digital amplio, simplemente haciendo uso de la función táctil de la pantalla.

Antes de iniciar el diseño e insertar objetos, es importante conocer las dimensiones exactas del dispositivo, ya que el desarrollador deberá calcular y decidir con precisión dónde ubicar cada objeto, así como sus dimensiones. En el caso de esta guía, como se mencionó en apartados anteriores, se eligió un modelo con dimensiones de 840x480.

Este elemento, para navegar entre las paginas, normalmente se sitúa en la parte inferior de la pantalla, ocupando todo su ancho, y se compone de tres botones. Para hacer su control lo más accesible posible, cada botón ocupa una tercera parte del ancho total. Por ejemplo, utilizando un dispositivo con un ancho de 800 píxeles, cada botón abarca aproximadamente 266 píxeles horizontalmente. De este modo, el primer botón ("page prev") comenzará en la posición 0 y finalizará en la posición 266, mientras que el siguiente botón comenzará justo en la posición 266. Este patrón se repetirá en los botones sucesivos.

En el plano vertical ocurre algo similar: si la pantalla tiene una altura máxima de 480 píxeles, es necesario decidir a qué altura se colocarán estos botones. Normalmente, se sitúan en la parte baja, reservando una altura determinada según las necesidades del diseño.

```
{"comment" : "Control de páginas"}
{"page":0,"id":1,"obj":"btn","action":{"down": "page
prev"},"x":0,"y":430,"w":266,"h":50,"bg_color":"#00A6FF","text":"
\uE141","text_color":"#FFFFFF","radius":0,"border_side":0,"text_f
ont":24}
{"page":0,"id":2,"obj":"btn","action":{"down": "page
back"},"x":266,"y":430,"w":268,"h":50,"bg_color":"#00A6FF","text"
:"\uE2DC","text_color":"#FFFFFF","radius":0,"border_side":0,"text
_font":24}
{"page":0,"id":3,"obj":"btn","action":{"down": "page
next"},"x":534,"y":430,"w":266,"h":50,"bg_color":"#00A6FF","text"
:"\uE142","text_color":"#FFFFFF","radius":0,"border_side":0,"text
_font":24}
```

Esto, una vez introducido en el *File Editor* -> pages.jsonl, se deberá ejecutar esta acción para cada cambio en el documento, Clear Page (para borrar la caché de la pantalla) lo que la volverá completamente blanca, y Reload Page (para subir el contenido actual de pages.jsonl).

Este elemento quedaría de la siguiente forma en la pantalla, cómo se puede ver en la siguiente figura:

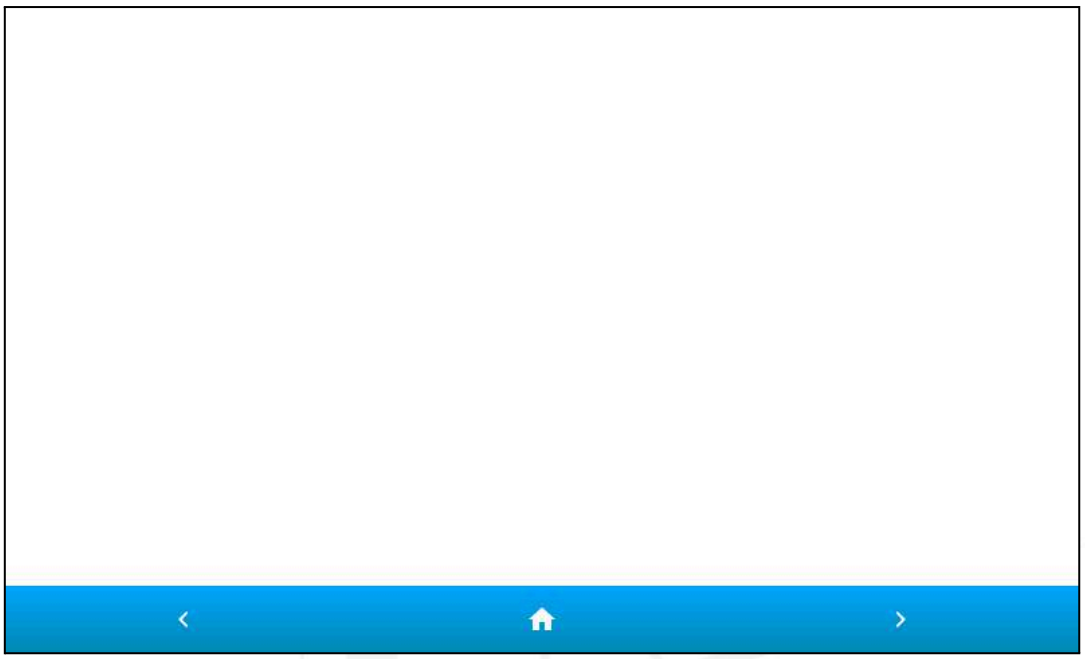


Figura 48. Página inicial con navegación de páginas

La elección de colores es el azul institucional UPM. Este atributo, como se ha visto en apartados anteriores sobre atributos en objetos, puede seleccionarse con "bg_color" en formato HTML (*HyperText Markup Language*, Lenguaje de marcado de hipertexto) RGB (*Red Green Blue*, Rojo Verde Azul). Para ayudar al desarrollador en la elección de colores, se puede hacer uso de webs como [65]:

https://htmlcolorcodes.com/

que devuelve el codigo de color seleccionado de una paleta de colores

Por último, este elemento permanecerá en todas las páginas existentes en el diseño, y se puede probar el uso de los botones ya que al pulsar, estos toman una tonalidad más oscura de su color natural

Encabezado (*Banner*)

Este elemento normalmente se sitúa en la parte superior de la pantalla y ocupa todo el ancho disponible, sirviendo como una barra de información o cabecera. Comúnmente llamado *banner*, este espacio puede contener información clave que el usuario necesite visualizar en todo momento, como el nombre del dispositivo, la hora actual, temperatura, o cualquier otro dato relevante para la interacción con el dispositivo.

Para lograr esto, se utilizará un pequeño “truco", ya que openHASP en ocasiones no cuenta con elementos específicos para funciones simples, como un contenedor de texto con fondo.

Por ello, se emplea un objeto básico (como un *label*) o una combinación sencilla de elementos para lograr este efecto. En definitiva, se trata de mantener siempre visible información crucial para el usuario durante el uso del dispositivo, garantizando así una mejor experiencia de usuario. Un ejemplo de lo comentado puede ser:

```
{"comment" : "Encabezado"}

{"page":0,"id":4,"obj":"btn","enabled":false,"x":0,"y":0,"w":840,
"h":50,"bg_color":"#00A6FF","text":"","text_color":"#000000","rad
ius":0,"border_side":0,"text_font":24}

{"page":0,"id":5,"obj":"label","x":0,"y":10,"w":800,"h":50,"bg_co
lor":"#00A6FF","text":"UPM","text_color":"#FFFFFF","align":1,"tex
t_font":24}
```

Siguiendo este ejemplo, es posible crear objetos que inicialmente contengan valores temporales (como {hora} o {tiempo}), los cuales se actualizarán posteriormente de forma remota. Por ejemplo, se podria mostrar la hora en un extremo de la barra superior y la información meteorológica (temperatura en ºC) en el extremo opuesto.

El mencionado “truco” o “atajo”, consiste en definir inicialmente los objetos con valores provisionales y luego actualizarlos dinámicamente mediante datos enviados desde un servidor de hora o una estación meteorológica a través de comunicación remota. Así, un dispositivo externo, como una estación meteorológica o un servidor NTP, puede transmitir periódicamente estos datos, manteniendo siempre actualizada la información visible en pantalla.

```
{"comment" : "Hora, actualizada dinamicamente"}
{"page":0,"id":5,"obj":"label","x":10,"y":10,"w":150,"h":30,"bg_c
olor":"#00A6FF","text":"{time}","text_color":"#FFFFFF","align":0,
"text_font":24}

{"comment" : "Informacion del tiempo actualizada dinamicamente"}
{"page":0,"id":6,"obj":"label","x":640,"y":10,"w":150,"h":30,"bg_
color":"#00A6FF","text":"{informacion relativa al
tiempo}","text_color":"#FFFFFF","align":2,"text_font":24,
"mode":"loop"}
```

Como se puede observar, la información del tiempo puede llegar en un formato largo y variado, es por eso que se aplica el "mode":"loop" para poder mostrar información de cualquier tamaño, sin afectar al resto de elementos, creando un efecto de *scroll* horizontal infinito.

El resultado, sin la información dinámica, quedaría como se puede observar en la siguiente figura:

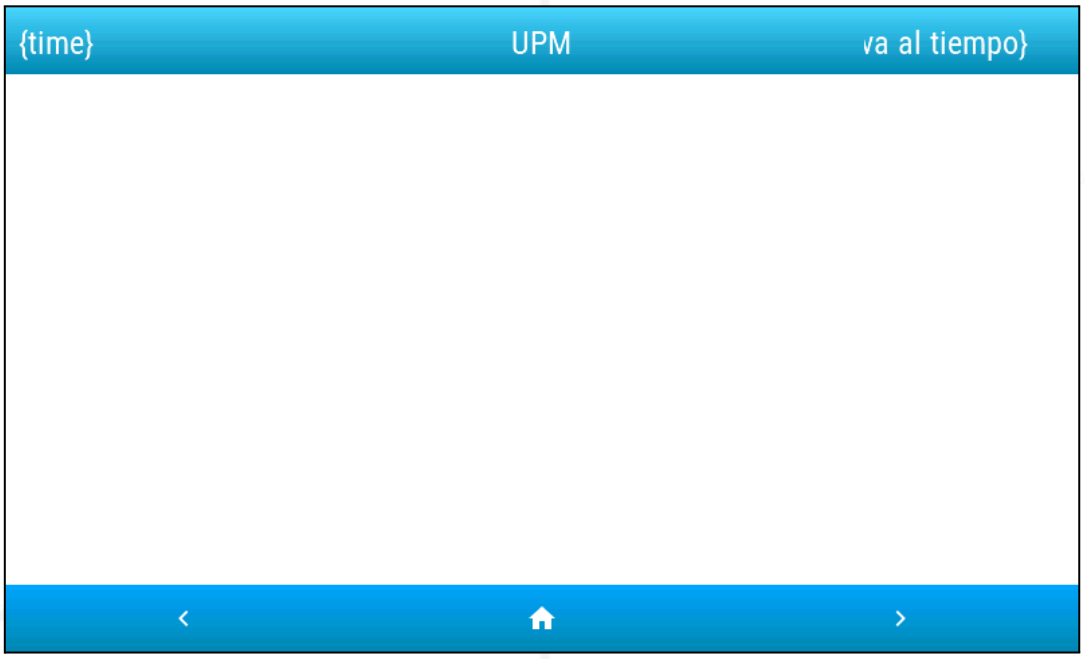


Figura 49. Página inicial con header incorporador

Este sería el resultado, y un buen punto desde el que partir, para el diseño del resto de páginas del proyecto.

4.5.3. Etiquetas para la presentación de información básica

Una etiqueta, denominada *label* en inglés dentro la categoría de objetos de openHASP, se emplea generalmente para mostrar texto plano y sin utilidad más allá de la presentación de información. Este texto puede ubicarse en cualquier localización definida en sus atributos. En este caso, dentro del prototipo, se crearán siete *labels* en siete páginas distintas, una para cada día de la semana. Esta es una manera de organizar la información, aunque existen otras formas de hacerlo.

Por ejemplo, la página 1 se reservará como punto de partida para información común, y a partir de ella, cada página incluirá la etiqueta correspondiente al día de la semana. Cada

página puede tener identificadores (ID) que van del 0 al 255. Las páginas no tienen un límite fijo, aunque su cantidad dependerá de la memoria disponible. En cada página, los identificadores, pueden tomar valores distintos y no es necesario que sigan un orden específico.

Como buena práctica, se reservará el ID = 1 para el *label* que mostrará el día de la semana, de la siguiente forma:

```
{"comment" : "Informacion comun"}
{"page":1,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":30,"bg_color":"#00A6FF","text":"Diario","text_color":"#000000","align":1,"text_font":30}
{"comment" : "======PÁGINA 1======"}
{"page":2,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":30,"bg_color":"#00A6FF","text":"Lunes","text_color":"#000000","align":1,"text_font":30}
{"comment" : "======PÁGINA 2======"}
{"page":3,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":30,"bg_color":"#00A6FF","text":"Martes","text_color":"#000000","align":1,"text_font":30}
{"comment" : "======PÁGINA 3======"}
{"page":4,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":30,"bg_color":"#00A6FF","text":"Miércoles","text_color":"#000000","align":1,"text_font":30}
{"comment" : "======PÁGINA 4======"}
{"page":5,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":30,"bg_color":"#00A6FF","text":"Jueves","text_color":"#000000","align":1,"text_font":30}
{"comment" : "======PÁGINA 5======"}
{"page":6,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":30,"bg_color":"#00A6FF","text":"Viernes","text_color":"#000000","align":1,"text_font":30}
{"comment" : "======PÁGINA 6======"}
{"page":7,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":30,"bg_color":"#00A6FF","text":"Sábado","text_color":"#000000","align":1,"text_font":30}
{"comment" : "======PÁGINA 7======"}
```

{"page":8,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":30,"bg_color":"#00A6FF","text":"Domingo","text_color":"#000000","align":1,"text_font":30}

Como se puede observar, este proceso es repetitivo y la diferencia entre otros objetos del mismo tipo, *label*, se limitaría únicamente a los atributos *text_color*, *font*, *text* y, por supuesto, su posición, *x*, *y*, y tamaño, *w*, *h,* además del texto a mostrar.

El resultado de una de estas páginas con un *label* sencillo quedaría de la siguiente forma:

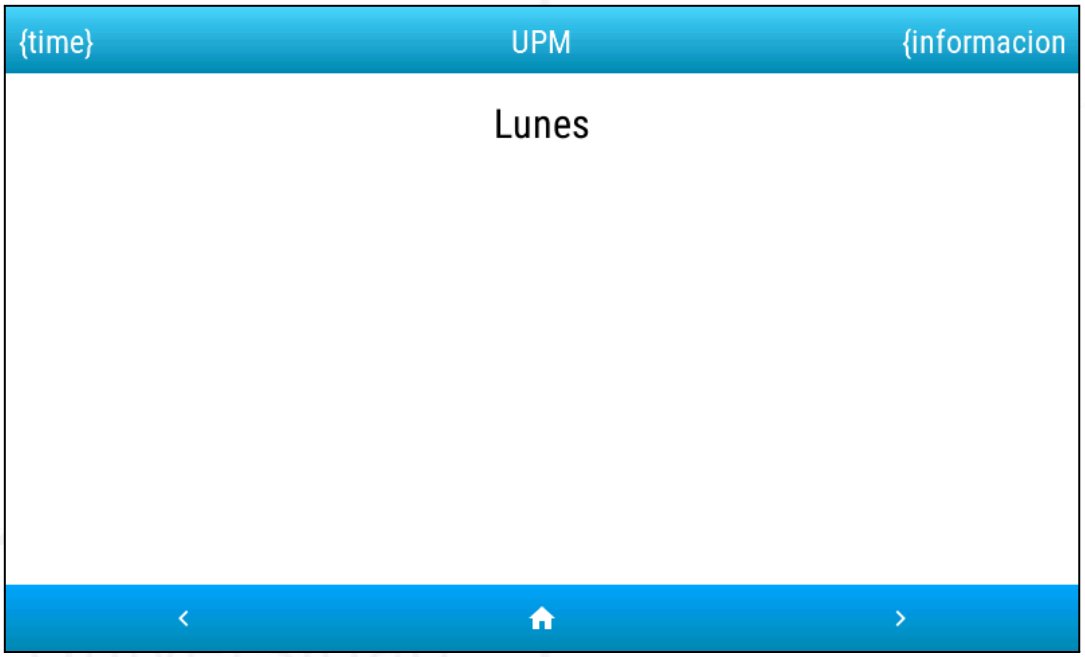


Figura 50. Aspecto etiqueta básica

Siendo el mismo resultado, pero para diferentes días, en cada una de las páginas.

Dado que se busca mantener esta información en la misma ubicación en cada una de las páginas, sólo será necesario modificar el atributo *text*. No obstante, cada proyecto puede ajustar estos parámetros según sus necesidades, permitiendo modificar cualquier atributo de acuerdo con las propiedades del objeto descritas anteriormente.

Por último, puede observarse, como se ha mencionado anteriormente, la estructura que puede tener *pages.jsonl* para facilitar su lectura y mantenimiento, indicando en cada caso la posición de la página en la que se encuentra. Otra forma implícita de hacerlo es mediante el uso del objeto *comment*, comentario, dónde se puede especificar directamente la página a la que pertenece, por ejemplo:

```
{“page” : 7,"comment" : "======PÁGINA 7======"}
```

Automáticamente, todos los objetos definidos después de este pertenecerán a la misma página sin necesidad de declarar el atributo en cada uno. Sin embargo, para mejorar la comprensión del código fuente por parte de futuros desarrolladores, es recomendable especificar siempre la página de cada objeto. Esto evita el uso de declaraciones implícitas que podrían generar confusión en el futuro.

Por último, se tendrá que actualizar dinámicamente la información común de la página 1 desde un servidor, como ya se mencionó. Este proceso se explicará más adelante

En primer lugar, se añadirá un texto similar a los vistos anteriormente: *"Tiempo para la próxima toma"*. Justo debajo, se incorporará un contador que servirá para mostrar otro tipo de texto actualizado en tiempo real, incluyendo incluso los segundos. Esto se implementa de la siguiente manera:

```
{"page":1,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":50,
"bg_color":"#00A6FF","text":"Tiempo para la próxima
toma","text_color":"#000000","align":1,"text_font":30}
```

```
{"page":1,"id":2,"obj":"label","x":100,"y":200,"w":600,"h":10
0,"bg_color":"#00A6FF","text":"00:00:00","text_color":"#000000","a
lign":1,"text_font":80}
```

Como se puede observar, la *label* que muestra el tiempo restante ha sido ajustada tanto en posición como en tamaño. Esto se debe a que, para garantizar la accesibilidad, el tamaño del texto debe ser lo suficientemente grande para que la mayoría de los usuarios puedan visualizarlo con facilidad.

El resultado sería el siguiente:

Figura 51. Etiqueta básica para prototipo

En resumen, el objeto *label* permite realizar una gran variedad de acciones y ofrece múltiples posibilidades creativas para su implementación. Su uso es sencillo y puede resultar altamente útil en distintos escenarios.

4.5.4. Botones y acciones simples

Siguiendo con el desarrollo del prototipo y la presentación de ejemplos, el siguiente objeto más utilizado después de una *label* es el botón, *btn*. Este elemento puede mostrar texto, al igual que una *label*, pero con la diferencia de que permite personalizar su fondo y, además, ejecutar diversas acciones. Las acciones pueden realizarse de forma local, es decir, dentro de la propia pantalla, o bien pueden emplearse para controlar dispositivos externos. Esto puede lograrse a través de MQTT, permitiendo una gestión remota, o mediante la activación de un GPIO conectado al ESP32 que controla la pantalla.

En este caso, se usará un botón en la pantalla inicial que conducirá automáticamente al usuario a la página del día actual. Este día se actualizará de forma dinámica.

```
{"page":0,"id":10,"obj":"btn","x":200,"y":200,"w":80,"h":80,"radius":40,"bg_color":"#00A6FF","text":\uE0ED","text_color":"#FFFFFF","text_font":24,"action":”p2”}
```

En este caso, el botón estará centrado en la pantalla, pero ubicado en un lateral, como se puede apreciar en el valor del atributo *x*.Para lograr que el botón tenga una forma redonda, se puede aplicar una sencilla artimaña: definir el mismo valor para *w* y *h*, y establecer *radius* como la mitad de ese valor. Además, en el atributo *text* se utiliza uno de los iconos disponibles en el almacenamiento de openHASP.

Por último, se define la acción *action.p2*, que, en caso de pérdida de conexión con el servicio MQTT, llevará automáticamente a la página 2. Este atributo será editado de forma dinámica según las necesidades del sistema.

El resultado es el siguiente:

Figura 52. Implementación Botón

Este ejemplo, al igual que los ya mostrados, puede aplicarse para cada uso personal simplemente editando los atributos según los requisitos de cada proyecto.

4.5.5. Alertas y notificaciones

De la misma forma en que openHASP notifica si el dispositivo está conectado a la red y, en caso afirmativo, nos muestra un *pop-up* o alerta, estas funcionan como notificaciones de un único aviso, *msgbox*. Las notificaciones pueden activarse de distintas maneras. La primera opción es habilitarlas al inicio y mostrarlas cuando el usuario accede por primera vez a la

página. La segunda opción es activarlas dinámicamente en respuesta a un evento de algún dispositivo de la red, ya sea el servidor *backend* o un dispositivo embebido que de la orden.

Estos avisos pueden configurarse para que el usuario deba cerrarlos manualmente, en cuyo caso permanecerán en pantalla hasta que se confirme su cierre, o bien establecer un *timeout* en uno de sus atributos para que se cierren automáticamente tras un período determinado. Sus aplicaciones son diversas. En el caso del prototipo, se utilizarán para notificar al usuario cuando el contador haya llegado a cero. En esta notificación, se indicará qué medicamento debe tomarse, en qué cantidad y, por último, el usuario deberá confirmar si ha realizado la toma.

Por ejemplo, como ya se ha comentado, se activa de forma dinámica, aunque la llamada en el pages.jsonl es la misma, el siguiente msgbox:

```
{"page":1,"id":24,"obj":"msgbox","text":"Ibuprofeno
400mg”,"options":["Aceptar","Cerrar"]}
```

En el atributo *options*, se pueden incluir las opciones a mostrar en forma de un array en formato JSON. Estos valores, y según la elección del usuario, como se verá más adelante en el apartado de conectividad, se enviarán a través de la red MQTT. Además, en este caso, no se ha incluido el atributo *auto_close,* por lo que la alerta no se eliminará hasta que el usuario haya elegido una de las dos opciones.

El resultado de la notificación es el que se muestra:

Figura 53. Implementación Notificacion

Esta opción es muy versátil y adaptable, pudiendo mandar todos los eventos necesarios. Como en el caso de uso del prototipo, en una misma toma pueden coexistir varios tratamientos. En ese caso solo habrá que crear objetos con IDs distintos y se irán mostrando de forma secuencial.

Este ejemplo puede adaptarse a distintos requisitos según las necesidades del sistema, como podría ser, mostrar una alerta al iniciar el sistema, ya sea como una advertencia o una precaución. También es útil para desplegar notificaciones diseñadas para captar la atención del usuario, con la ventaja adicional de poder registrar su respuesta.

4.5.6. Organización de pestañas y visor

Otro de los elementos más utilizados para la organización de objetos y funcionalidades dentro de las páginas, optimizando el espacio y los recursos del sistema, son las pestañas y su visor.

Este elemento permite establecer una relación jerárquica entre objetos, de modo que algunos puedan depender de otros. Además, facilita la creación de más elementos de los que normalmente se pueden incluir en una página estándar de openHASP y, sobre todo, permite agrupar funcionalidades de manera estructurada.

El elemento, *tabview*, posee los mismos atributos que cualquier otro objeto, como se mencionó anteriormente. Su declaración se realiza de la siguiente manera:

```
{"page":2,"id":14,"obj":"tabview","x":10,"y":120,"w":780,"h":
300, "bg_color":"#00A6FF80"}
```

En este momento, la interfaz mostrará un recuadro con el color definido en el atributo *bg_color* (aunque pueden definirse más, según los estilos), sin aplicar ningún efecto adicional.

La diferencia de este objeto con otros radica en que su identificador dentro de la página se utiliza como el valor del atributo *parentid* para las pestañas asociadas al visor. A su vez, estas pestañas también tendrán un *parentid* que indicará los objetos que les pertenecen.

Antes de utilizar el visor de pestañas, es necesario crear al menos una pestaña. Esto se realiza de la siguiente manera:

{"page":2,"id":51,"obj":"tab","parentid":14,"text":"Desayuno"}
{"page":2,"id":52,"obj":"tab","parentid":14,"text":"Comida"}
{"page":2,"id":53,"obj":"tab","parentid":14,"text":"Merienda"}
{"page":2,"id":54,"obj":"tab","parentid":14,"text":"Cena"}

En el caso del desarrollo del prototipo, Pillbot, para el proyecto, se crean estas 4 pestañas, "tab", que se verán de la siguiente forma.

Figura 54. Implementación visor de pestañas

Ahora, como se puede observar en las líneas del archivo .jsonl donde se declaran, las pestañas cuentan con un *id*, que a su vez será el *parentid* de los objetos que están asociados a las mismas.

En este punto, si se quiere que un objeto aparezca dentro de una pestaña, funcionando como una interfaz completamente independiente, deberá incluir el atributo *parentid*, de la siguiente manera:

{"page":2,"id":101,"obj":"label","x":10,"y":10,"w":700,"h":30,"parentid":51,"text":"Texto","text_color":"#000000","align":0,"text_font":24}

Una vez que el objeto se ha declarado con el *parentid* de la pestaña, sus dimensiones y posición serán relativas al objeto *tab*.. El resultado sería el mostrado en la siguiente figura:

Figura 55. Implementacion objeto hijo

4.5.7. Indicadores de progreso

Se puede necesitar, por los requisitos del proyecto, indicar el progreso de una actividad, ya sea referente al sistema o al usuario. Este progreso puede indicar la finalización de una acción, el lanzamiento de alguna alerta, o simplemente llevar un recordatorio de las tareas diarias.

Es posible indicar progreso con numerosos objetos disponibles en el *firmware*: barras de progreso o similares, o marcadores (*checkboxes*).

Barras de progreso

En esta primera opción, existen varios objetos como bar, arc, linemeter.. todas con la misma función, indicar un progreso de forma lineal, circular, etc… y con similares atributos.

Estos objetos muestran un progreso, en tiempo, de forma visual. Normalmente el máximo de tiempo, correspondiente al punto inicial, se corresponde con un valor alto del atributo *val,* donde el máximo por defecto será 100. Este valor puede ir cambiando de forma dinámica, como se verá posteriormente, y puede irse modificando por alguna acción exterior, o simplemente, con el paso del tiempo.

En el caso de Pillbot, el prototipo para el desarrollo de este proyecto, se eligió esta opción para representar el tiempo, junto con el contador de tipo *label* actualizado

dinámicamente, para la próxima toma de pastillas. La declaración, en el fichero pages, del objeto, es la siguiente:

```
{"page":1,"id":2,"obj":"bar","x":240,"y":197,"w":320,"h":100,
"bg_color":"#00A6FF80","border_color":"#FFFFFF","val":100,
"opacity":80, "radius":500}
```

Y el resultado, por ejemplo, en el punto de partida, se puede apreciar en la siguiente figura:

Figura 56. Implementacion indicadores de progreso

Para, posteriormente, en este proyecto, paso del tiempo, se vería de la siguiente forma:

Figura 57. Resultado final pagina inicial

En este punto, como puede apreciarse, la barra de progreso indica que el tiempo está agotado, representado de una forma más visual.

4.5.7. Imágenes

Las imágenes son uno de los objetos más a tener en cuenta positivamente, por su potencial en representar y mejorar interfaces gráficas, al igual que negativamente por volver pesados e inestables los sistemas. En openHASP pueden representarse imágenes solamente en formato .png, por su alta compresión de imagen, y en la resolución y tamaño que se requiera, siempre que se conozcan los límites físicos de la pantalla TFT. Por ejemplo, sería inutil cargar una imagen con una resolución en alta definición, 1920x1080, o resoluciones similares, si el dispositivo, como es el caso del prototipo, tiene una resolucion de 840x340.

Entre los atributos del objeto imagen, *img*, openHASP ofrecen herramientas para el redimensionamiento de las mismas, pero esto en tiempo de procesado es lento, además, que no se estaría aprovechando las propiedades de la imagen, por el innecesario exceso de peso.

Al estar en un sistema embebido de pocos recursos, se debe tener especial cuidado en las imágenes utilizadas. Se recomienda que, estas, tengan una resolución más o menos baja, suficiente para el espacio que tendrá el objeto. Por ejemplo, si una imagen se requiere que ocupe 64x64, para una foto de perfil de usuario, sería una buena práctica definirla

directamente con ese tamaño y, en caso de ser necesario un pequeño ajuste, poder usar las características de openHASP como *zoom*.

Por ejemplo, la declaración de una imagen en el fichero pages:

```
{"page":9,"id":1,"obj":"img","x":22,"y":80,"w":150,"h":150,"bg_col
or":"#FFFFFF","border_color":"#000000","src":"L:/profile-3.png"}
```

Para disponer de la imagen, primeramente hay que guardarla en el sistema de ficheros. Esto puede hacerse con los botones en el *File Editor -> Upload* como se vio en apartados anteriores. Una vez dentro de el sistema de ficheros, esta puede referenciarse en el *jsonl* bajo el directorio:

```
L:/<nombre-imagen>.png
```

Posteriormente, pueden definirse el resto de atributos comunes, tal como se muestra en la declaración anterior. Como se indicó previamente, las propiedades *w* y *h* deben coincidir con las dimensiones de la imagen. En caso de que el atributo *autosize* esté activado, la imagen se reescalará automáticamente, con lo que ello significa.

4.6. Implementación de configuracion avanzada

Una vez analizado exhaustivamente la construcción de interfaces en openHASP, gestión de objetos, pestañas, páginas, en este apartado se repasan las características más avanzadas del firmware que permiten crear sistemas robustos y duraderos en el tiempo.

4.6.1. Securizacion del panel administrador

Es común, que una vez acabadas las tareas de desarrollo y pruebas de un sistema en desarrollo, antes de desplegar la versión en producción y funcionamiento, una tarea importante en los proyectos modernos es la securización del sistema.

En openHASP, al tratarse de un *firmware* para un sistema embebido, la primera capa de seguridad es la característica inherente de la red interna, es decir, la red de openHASP a la que se conecta, normalmente WiFi, ya es privada. Pero en caso de que pudiera estar desplegado en un sistema con IP pública, desde acceso permitido a/desde Internet, es posible securizar el acceso al panel de administrador para evitar posibles ataques que, en caso de poder acceder al panel, serían altamente críticos.

Se puede securizar la conexión HTTP del panel de administrador. Para ello, habrá que acceder al panel de *Configuration*, desde el menú principal:

```
http://<ip>/config
```

Y una vez en el, seleccionar la opción http:

```
http://<ip>/config/http
```

Dentro, se puede observar un panel como el mostrado en la siguiente figura:

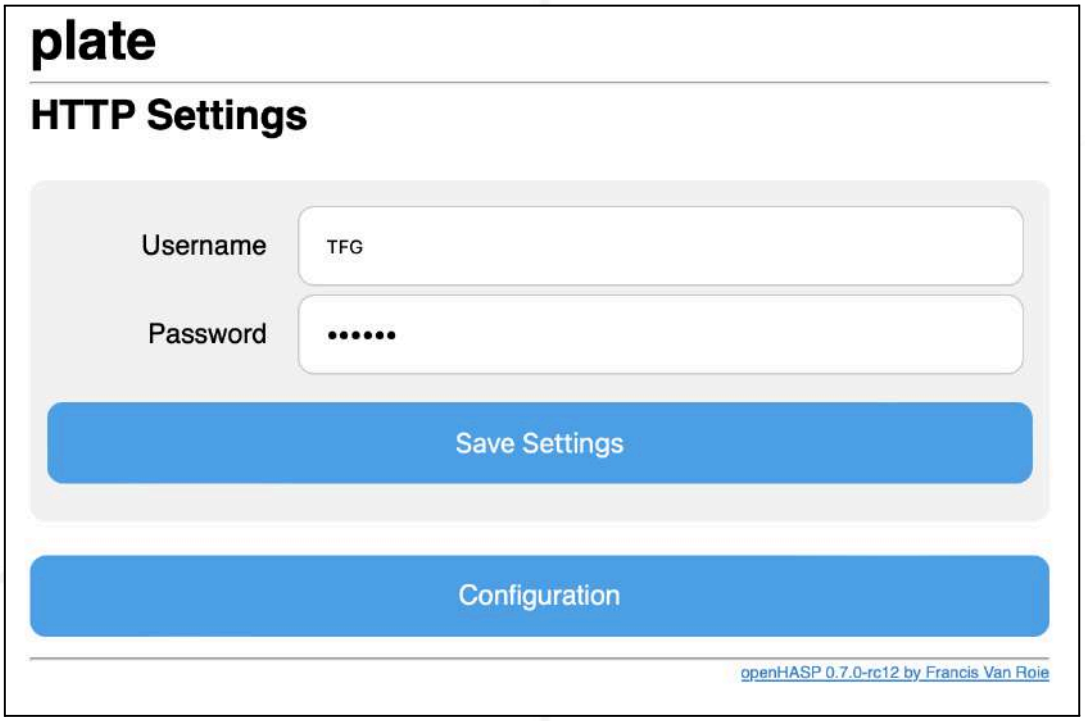


Figura 58. Configuracion securizacion HTTP

En las opciones *Username*, *Password,* respectivamente, pueden configurarse un nombre de usuario y contraseña para acceder al panel. A modo de prueba, se fijarán las claves TFG/ETSISI, de manera provisional.

La buena práctica, en un entorno real, sería generar un usuario y contraseña con herramientas de gestión de usuarios. Estas herramientas protegen estas credenciales con un nivel superior de seguridad, puesto que incluyen algunos pasos añadidos como la autenticación en dos pasos, las contraseñas temporales, entre otras.

Una vez aplicada la configuración, en los próximos accesos al panel de administrador, el usuario visualizará lo siguiente:

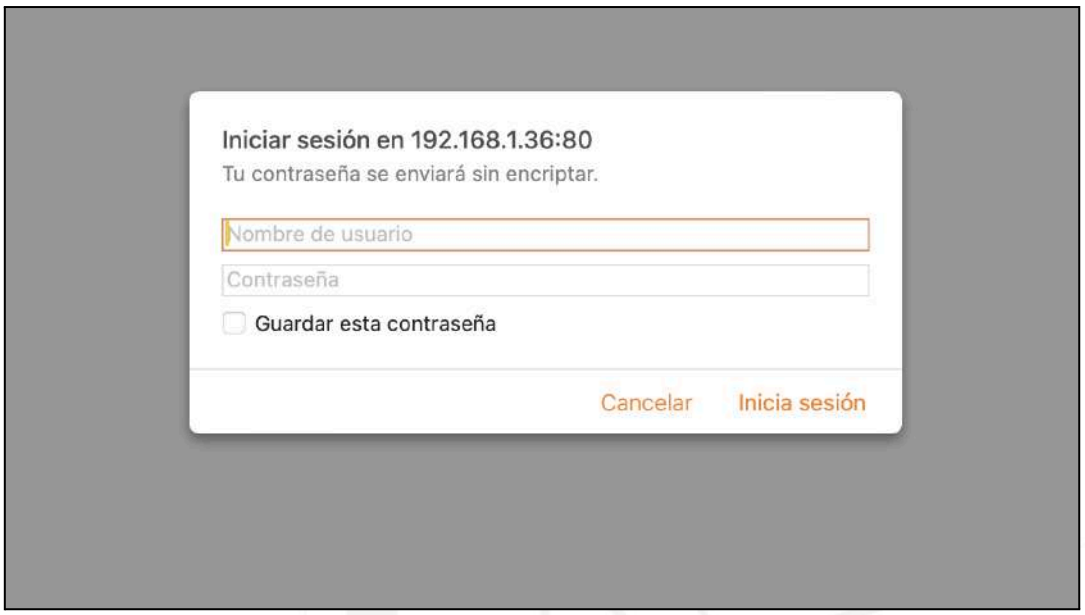


Figura 59. Petición de credenciales acceso web

Con esta pequeña configuración, nativa en el *firmware*, se puede añadir una capa de seguridad añadida al sistema embebido.

4.6.2. Servicio FTP (File Transfer Protocol) para la transferencia remota de ficheros

Como se ha visto en apartados anteriores, es posible subir ficheros a la memoria interna de del ESP32. Estos ficheros pueden ser, por ejemplo, una fuente para la mejora del diseño o la legibilidad, un nuevo fichero de pages.jsonl o, simplemente una imagen. Hasta ahora, con la configuración vista, esto solo podía hacerse desde el File Editor, pero habilitando el servicio FTP (*File Transfer Protocol,* Protocolo de transferencia de ficheros) [66], protocolo usado para la transferencia de archivos remota, es posible realizarlo sin necesidad de entrar en el panel de administrador.

La principal ventaja es la utilización de comandos en una futura automatización ya que el protocolo ftp viene integrado en línea de comandos en la mayoría de sistemas operativos conocidos. Por ejemplo, una foto de telegram, subida por el usuario, puede automatizar su subida con uso de *scripting* directamente al servidor habilitado FTP.

Para configurarlo, es necesario ir a la pestaña de configuración:

```
http://<ip>/config/ftp
```

Una vez dentro, se observa lo siguiente:

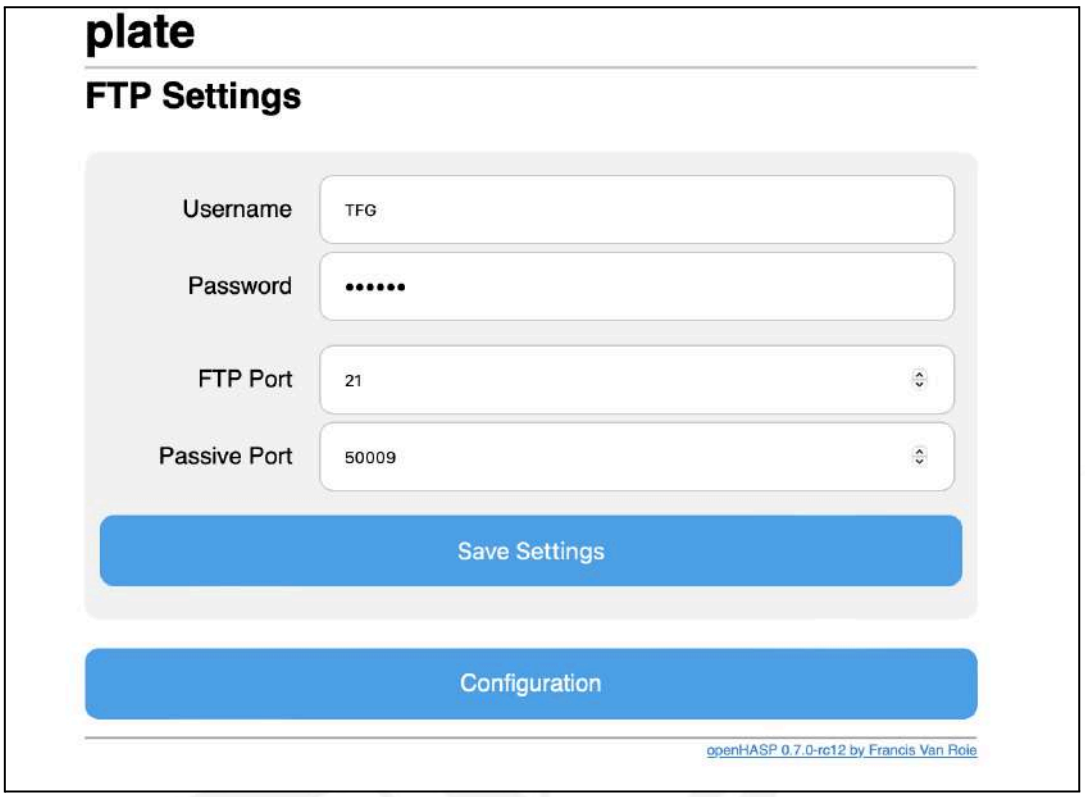


Figura 60. Configuración FTP

Y configurando simplemente las credenciales, el puerto y puerto pasivo, que pueden dejarse por defecto si no colisiona con otro dispositivo en el mismo equipo (poco probable) y reiniciando el equipo, se puede acceder con un cliente FTP (por ejemplo, FileZilla) [67] o por línea de comandos:

```
ftp <ip>
```

Un ejemplo de acceso sería el siguiente:

Figura 61. Acceso FTP

Y una vez establecida la conexión:

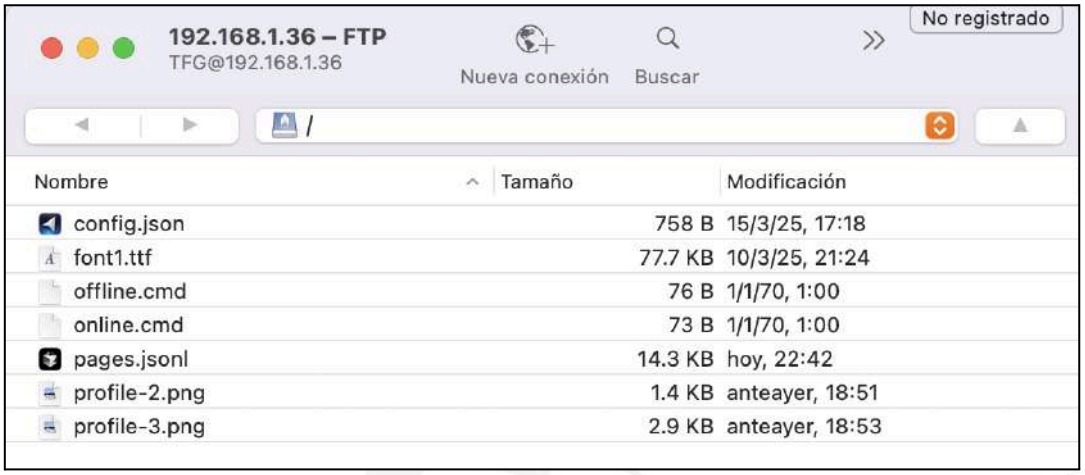


Figura 62. Explorador de archivos remoto vía FTP

Se puede obtener de forma automatizada acceso al almacenamiento interno del sistema.

4.6.3. Servicio NTP para gestión de hora remota

El servicio NTP es comúnmente usado en sistemas distribuidos para establecer una sincronización de hora remota con un servidor central. El *firmware* openHASP tiene la capacidad de sincronizar el reloj interno del microcontrolador para así tener un contexto horario a la hora de comunicarse con otros sistemas.

Disponer de un servidor NTP propio y privado suele ser costoso en tiempo y recursos, puesto que uno sincronizado, incluye algoritmos de cálculo de hora y sincronización. Es por ello que existen servidores NTP abiertos al uso, de forma gratuita, tanto Universales (estos devolverán una hora universal) o más regionales. La página:

https://www.ntppool.org/es/zone/es

ofrece un servicio de direcciones de servidores para proyectos personales [68].

Posteriormente, para su configuración en openHASP, habrá que dirigirse al apartado de configuración *Time config*:

http://<ip>/config/time

Y, dentro de la pestaña:

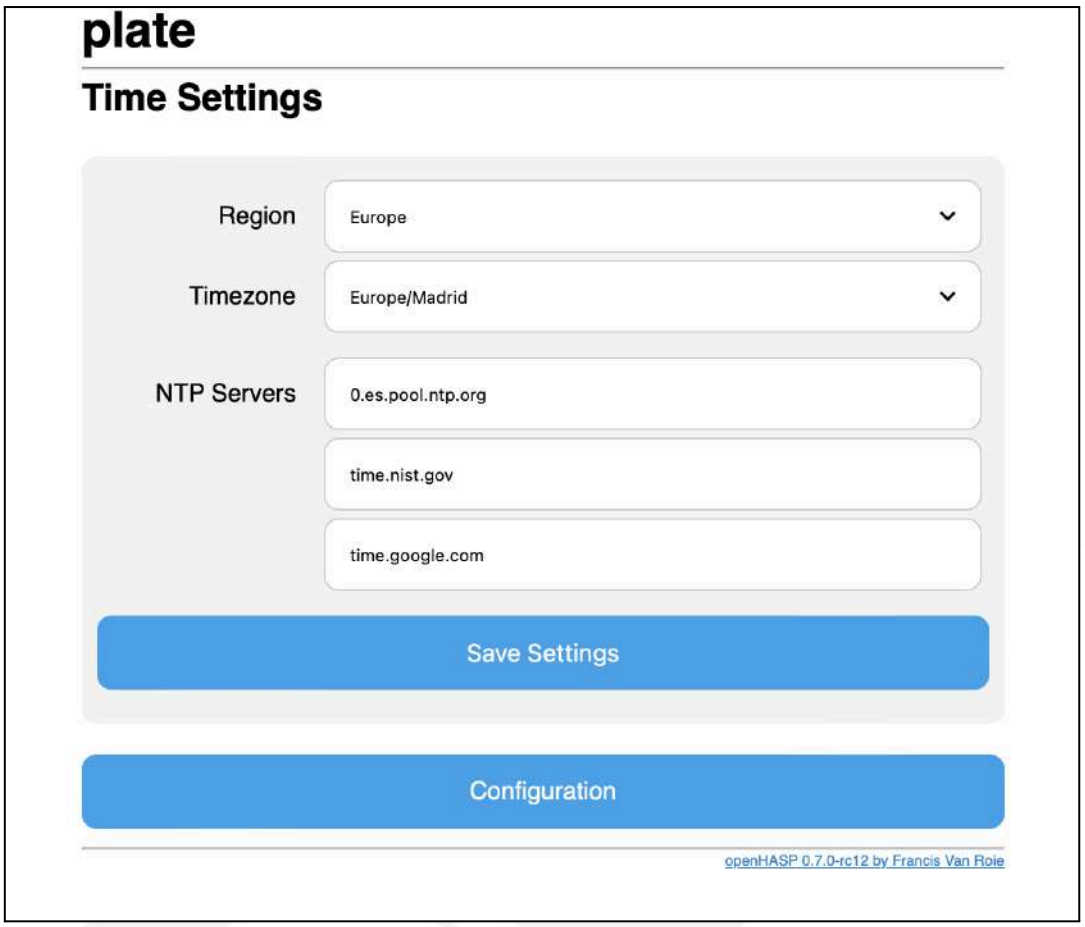


Figura 63. Configuración servidor NTP

Se puede configurar la región y la zona horaria, en el caso del prototipo Europa/Madrid, pero lo más importante, son las direcciones de servidores NTP, donde openHASP permite hasta 3 opciones. La primera de la lista es de ntp.org, y las otras 2 restantes se han dejado por defecto.

Esta petición de hora mediante NTP solo estará disponible si existe conectividad WiFi. Generalmente, la solicitud se realiza durante el arranque y, a partir de ese momento, el reloj interno del microcontrolador se encarga de mantener la sincronización.

Con la hora correctamente ajustada, los registros (*logs*) incluirán la hora, minuto y segundo exactos. Para mostrar esta información dinámicamente en el archivo pages.jsonl, es necesario añadir el atributo *template*, como se muestra en el siguiente ejemplo.

```
{"page":0,"id":5,"obj":"label","x":10,"y":10,"w":150,"h":30,"bg_co
lor":"#00A6FF","text_color":"#FFFFFF","align":0,"text_font":24,
"text": "98:80:00", "template": "%H:%M"}
```

En apartados anteriores, se definió la *label* con valor {*time*}, y, después de esta configuración y el nuevo atributo, *template,* se puede tener la hora actualizada como puede verse en el recuadro indicado en la siguiente figura:

Figura 64. Resultado servidor NTP

4.6.4. Gestión de logs remotos (syslog)

El servicio de *logs*, syslog, se utiliza en sistemas operativos distribuidos para almacenar *logs* (accesos, conexiones, acciones, etc…) de dispositivos remotos. Este servicio normalmente está configurado en un dispositivo con capacidad para almacenar *logs* y con recursos suficientes. Como ya se ha visto anteriormente, la configuración avanzada de openHASP está pensada para reducir el acceso a su panel de administración, para así poder automatizar tareas, lo que facilitará el mantenimiento y trazabilidad del sistema cuando este sea desplegado.

El servicio syslog de openHASP puede configurarse y así poder gestionar los *logs* de manera remota. Esta configuración puede hacerse desde el panel:

`http://<ip>/config/debug`

El aspecto del panel puede verse en la siguiente figura:

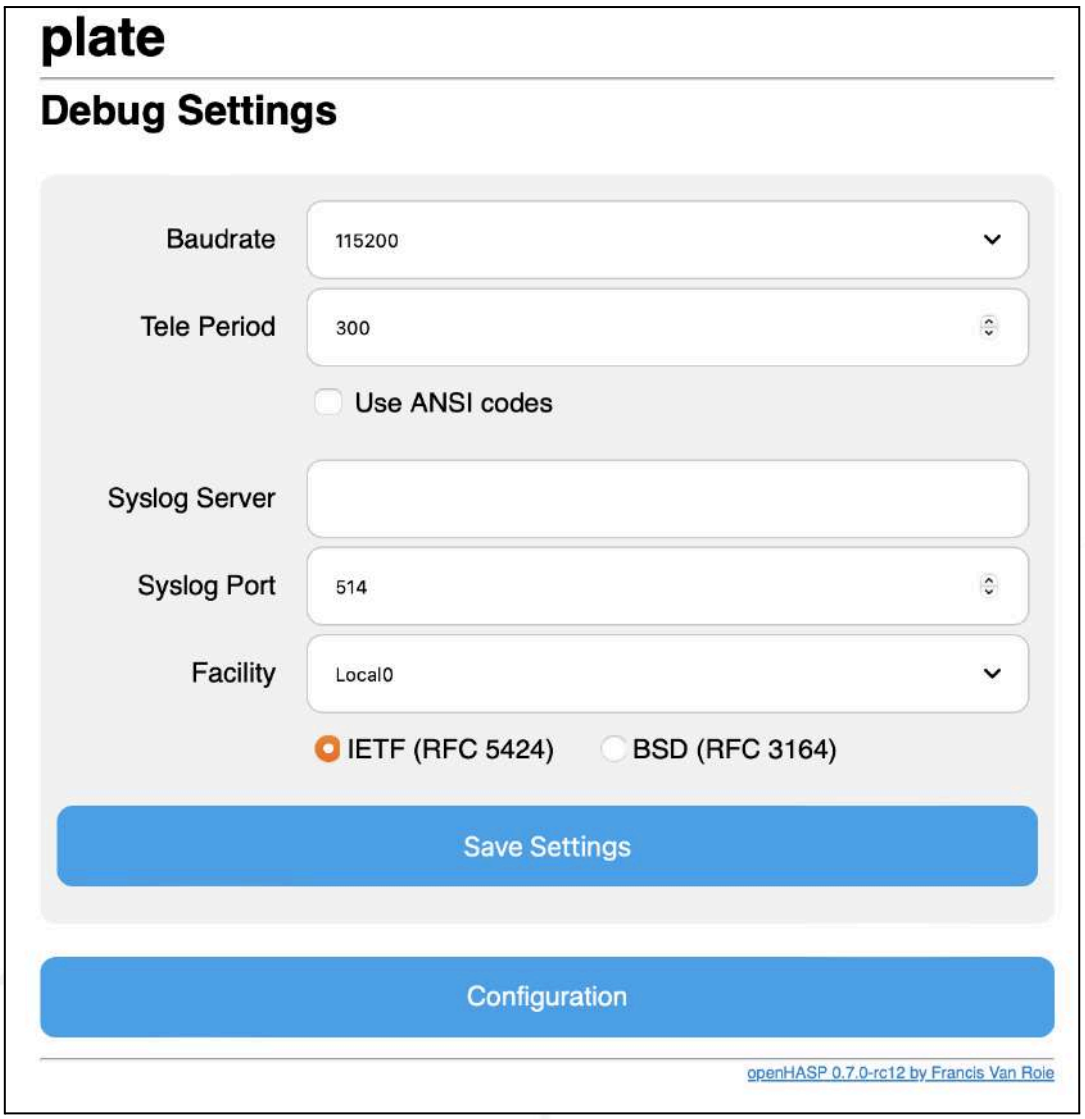


Figura 65. Configuración Syslog

En el cual, los parámetros críticos a configurar son Syslog *Server* y *Syslog Port,* en los que habrá que indicar la IP y puerto del servidor configurado para almacenar estos logs.

Para tener más detalle de la configuración de un servidor syslog (*ver Anexo*).

4.7. Integración con otros sistemas

Hasta ahora toda la lógica y configuración, salvo ligeras pinceladas en la configuración avanzada, como con el servidor SysLog o NTP, habían sido de forma local, es decir, dentro del mismo microcontrolador que contiene el *firmware* de openHASP. A lo largo del desarrollo del proyecto, se han definido algunos elementos que carecían de utilidad hasta su conexión con el entorno en el que el mencionado software puede verdaderamente explotar

su potencial. En este apartado se ven las diferentes vías que tiene el sistema operativo para comunicarse con los elementos de su entorno.

4.7.1. Comandos remotos por MQTT

Una vez configurada la conexión MQTT en el apartado de preparación del entorno, donde se estableció el servidor central y posteriormente su conexión con el microcontrolador en openHASP, es posible ejecutar comandos para dar funcionalidad y dinamismo al sistema. Desde MQTT, se pueden controlar todas las opciones disponibles para la gestión del archivo *pages.jsonl*.

Esta característica no solo permite modificar los elementos existentes en el archivo, sino que también permite añadir nuevos objetos y funcionalidades de forma completamente remota. Esto hace que la plataforma sea altamente versátil, facilitando su mantenimiento y mejora continua.

Para probar primero la conectividad del sistema, se puede ejecutar el comando anteriormente visto:

```
mosquitto_sub -h localhost -t "#" -u MQTT -P TFG -v
```

Donde puede verse como el microcontrolador manda los siguientes mensajes:

```
hasp/plate/LWT online
hasp/plate/state/idle off
hasp/plate/state/p1b3 {"event":"down"}
hasp/plate/state/p1b3 {"event":"up"}
hasp/plate/state/statusupdate {"node":"plate","idle":"off","version":"0.7.0-rc
12","uptime":3619,"ssid":"MiFibra-B671","rssi":-57,"ip":"192.168.1.36","mac":"
64:E8:33:48:63:90","heapFree":82528,"heapFrag":28,"core":"4.4.6","canUpdate":"
false","page":1,"numPages":12,"tftDriver":"Other","tftWidth":800,"tftHeight":4
80}
hasp/plate/state/sensors {"time":"2025-03-18T23:06:52","uptimeSec":3619,"uptim
e":"0T01:00:19"}
```

Figura 66. Suscripción topic MQTT

Estos mensajes indican distintos estados del sistema. *hasp/plate/LWT online* señala que el microcontrolador está correctamente conectado, es decir, en línea con el servidor MQTT. *hasp/plate/state/idle, off* indica que openHASP ha salido del modo *Idle* (bajo consumo automático después de un tiempo sin uso) debido a una interacción en la pantalla, como se puede ver en los dos mensajes siguientes: *hasp/plate/state/p1b3*. Por último, el mensaje *hasp/plate/state/statusupdate* se envía de forma automática y periódica con toda la

información del sistema, permitiendo que el servidor central conozca, en caso de ser necesario, el estado general del sistema.

Además, también envía el estado de todos los movimientos que se realizan en el sistema operativo, por ejemplo:

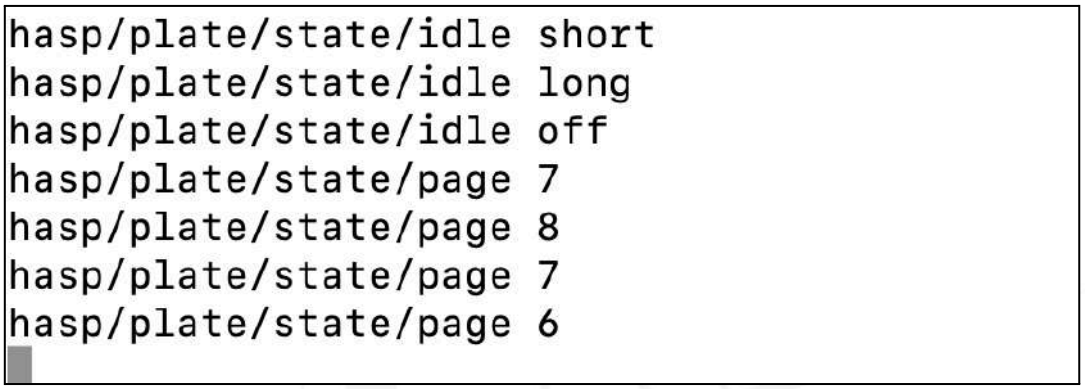


Figura 67. Mensajes de actividad en topic MQTT

Esta información es parte de la que el servidor MQTT central recibe tanto de forma automática, como el estado del microcontrolador, o en las acciones de parte del usuario. Como puede verse, openHASP gestiona los distintos *topics* siempre debajo de /hasp/plate, donde *plate,* es el nombre que se asignó al *firmware* en la configuración inicial. Estos datos pueden ser escuchados por microservicios en el servidor y realizar acciones en función de ellos.

Para lanzar comandos remotos solo hay que escribir en el *topic* /hasp/plate/command, donde el contenido del mensaje será un elemento de pages.jsonl, en estos casos:

1. Si el elemento ya existe, se sobreescriben los atributos con los mandados. Útil para modificar elementos existentes, como listas, etiquetas, acciones en botones…
2. Si el elemento no existe en pages.jsonl se crea. Para ello, el elemento debe estar configurado correctamente. Útil para crear alertas, solucionar errores o implementar mejoras y ampliaciones del sistema.

Por ejemplo, para crear una alerta mandada remotamente, podría hacerse con el siguiente comando:

```
mosquitto_pub -h 141.94.247.154 -t "hasp/plate/command" -u MQTT -P
TFG  -m  '{"page":0,"id":24,"obj":"msgbox","text":"Ibuprofeno  \n
400mg","options":["Aceptar","Cerrar"]}'
```

En el comando, se manda un mensaje al servidor MQTT en la IP 141.94.247.154, al *topic* hasp/plate/command, las credenciales y con el atributo -m, el mensaje crudo de la creacion de un objeto msgbox con el texto de un medicamento “Ibuprofeno 400mg” y las opciones “Aceptar”, “Cerrar”.

El efecto es el siguiente:

Figura 68. Notificación comandada remotamente via MQTT

En el caso, de modificar un elemento ya existente, puede tener varias utilidades, como: deshabilitar una característica momentánea, por ejemplo, el atributo *enabled* de un objeto *button*, *btn*, o un *label*.

Para el prototipo se modificará la *label* creada anteriormente con el texto {Información Relativa al tiempo}, se modificará su valor por la fecha y temperatura actual, con el siguiente comando:

```
mosquitto_pub -h 141.94.247.154 -t "hasp/plate/command" -u MQTT -P
TFG -m '{"page":0, "id":6, "text":"Lun 10 mar - 12°C \uE67E"}'
```

En este caso, se está modificando el atributo *text* del objeto con identificador 6, de ahí la importancia del identificador, en la página 0 (página común). El resultado es el siguiente:

Figura 69. Aspecto pagina principal actualizada remotamente

Como puede verse en la imagen, ahora todos los elementos del *banner*, están siendo actualizados dinámicamente. En la esquina superior derecha por MQTT e superior izquierda a través del servidor NTP.

Las opciones de personalización a través de MQTT son exactamente las mismas que las del archivo *pages.jsonl*, de ahí su versatilidad. Estos comandos pueden automatizarse con librerías de Python[69], como *paho-mqtt* [70], o directamente con lenguajes como *Bash* [71]. Por ejemplo, se puede ejecutar este comando de forma automática y diaria, con la información relativa de un servidor meteorológico. Lo que personaliza y dota de "vida" todo ello de manera distribuida.

En conclusión, al igual que un servidor central puede realizar esta gestión, también puede hacerse desde otros microcontroladores o dispositivos embebidos que permitan escuchar y enviar mensajes a través de MQTT. De esta forma, se puede realizar una acción ante eventos externos o intervenir en otros.

4.7.2. Configuración de GPIOs en firmware

Además de interactuar con eventos en la red MQTT, en ausencia de red, pueden mantenerse el control de sistema esenciales, como los GPIO integrados en el microcontrolador. Esta característica, además de dotar al microcontrolador de control sobre

periféricos conectados físicamente, aumenta la seguridad ante la pérdida de conexión por parte del servidor MQTT.

Esta configuración puede realizarse desde:

```
http://<ip>/config/gpio
```

En la pestaña, como puede observarse en la siguiente figura:

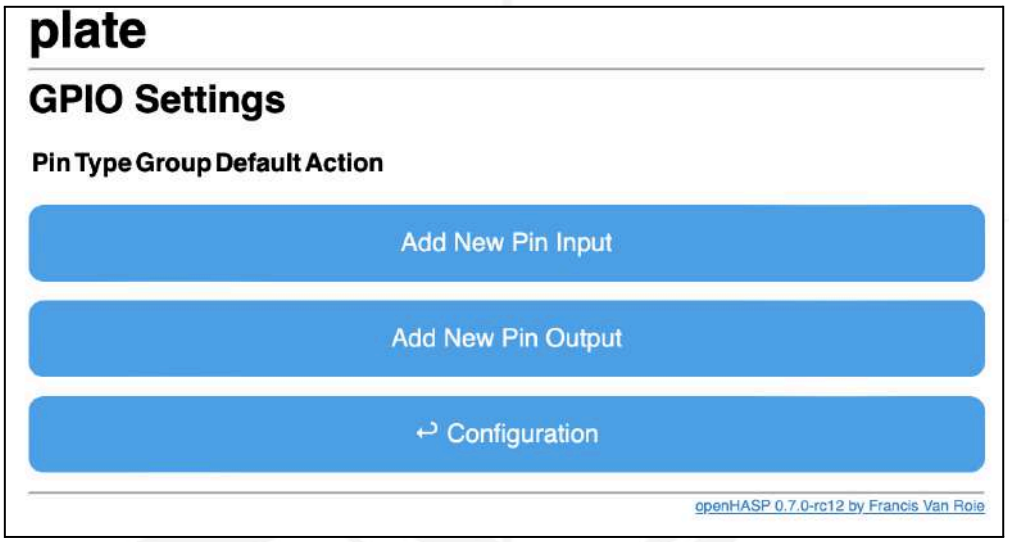


Figura 70. Configuración pines GPIO

da la opción al desarrollador de configurar un GPIO de entrada, para escuchar sus eventos, o un GPIO de salida, para realizar eventos desde la interfaz. Para el prototipo, se configura un GPIO de salida, que se realiza de la misma forma que uno de entrada.

Para ello, una vez dentro de *Add New Pin Output:*

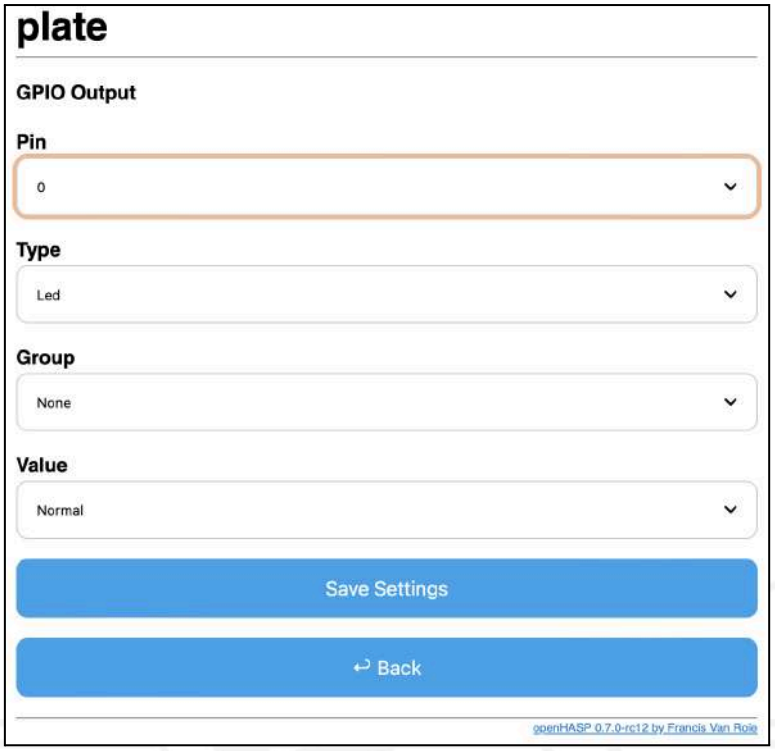


Figura 71. Configuración pines GPIO 2

Se selecciona, entre los desplegables el GPIO disponible, véase el ejemplo:

Figura 72. Selección de pin configuración GPIO

Después, el tipo, si no existiera, uno similar en características:

Figura 73 Selección de tipo GPIO

E importante, el *groupid*:

Figura 74. Selección de grupo GPIO

El *groupid*, será el identificador al que hacer referencia del GPIO en el pages.jsonl, por ejemplo:

```
{"page":1,"id":1,"obj":"switch","x":30,"y":40,"w":180,"h":75,"radi
us":40,"radius20":40,"groupid":1}
```

Este objeto en el pages.jsonl, activará el GPIO dentro del *groupid* 1 cuando se active el objeto de tipo *switch*.

En conclusión, esta forma de activar o escuchar eventos de periféricos conectados físicamente es una buena opción cuando se quiere tener el control sobre dispositivos esenciales. Por ejemplo, en el caso de una luz, si se pierde la conexión WiFi o hay una caída del servidor MQTT, el servicio seguiría estando disponible.

4.8. Creación de un producto final con openHASP

Durante el desarrollo de este proyecto se han ido detallando pequeños avances en la construcción de un prototipo funcional creado exclusivamente con las herramientas e instrucciones proporcionadas en las explicaciones de esta memoria. Es por ello que se han incluido pinceladas de su aspecto y funcionamiento a lo largo de los distintos puntos, a modo de respaldar técnicamente los conceptos teóricos.

4.8.1. Concepto

El propósito del prototipo, llamado *Pillbot,* es ayudar en la organización de la toma de medicamentos, principalmente en grupos vulnerables, como podrían ser: personas mayores, personas con discapacidades cognitivas, problemas visuales o psicomotrices. Este prototipo, completamente funcional, es capaz de cubrir una necesidad básica con apenas unas cuantas líneas de código, utilizando hardware de muy bajo coste, así como bajo consumo y mantenimiento. El logo de marca del prototipo es el que se muestra en la figura:

Figura 75. Logo pillbot

4.8.2. Diseño

En el diseño de la plataforma se pone el foco en la accesibilidad, no solo en la elección de colores, fuentes o tamaños, aspectos clave para personas con dificultades visuales, y un problema presente en un alto porcentaje de los grupos vulnerables, sino también en la sencillez de uso, como se detalló en los objetivos de openHASP, donde se ponía énfasis en la sencillez de los proyectos que usaban esta tecnología. En el capítulo *Impacto social, ambiental y Agenda 2030* se explicarán en detalle el cómo y el porqué de estas elecciones.

Software

El producto consta de tres elementos conectados: el dispositivo embebido, el servidor central y la interfaz móvil administradora.

El dispositivo embebido está formado, por el momento, por una pantalla TFT con un microcontrolador ESP32 integrado, en el cual se ejecuta el sistema operativo openHASP y cuya función es mostrar la interfaz gráfica. El servidor central se encarga de procesar toda la información entre el dispositivo embebido y el usuario final. En él se ejecutan los *scripts* responsables de los distintos microservicios del sistema. Por último, la interfaz móvil administradora abstrae todo su funcionamiento a través de la API (*Application Programing Interface*, Interfaz de programación de aplicaciones) de la aplicación de mensajería Telegram, lo que permite abaratar el desarrollo de una interfaz personalizada.

El esquema de la arquitectura del prototipo se resume en la siguiente figura:

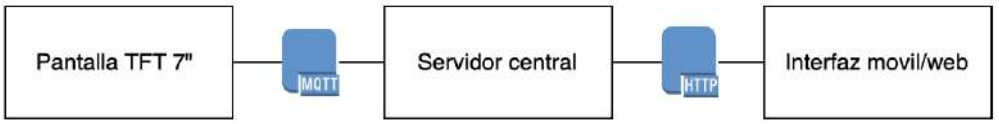


Figura 76. Dispositivos involucrados en prototipo

El microcontrolador, con openHASP, sirve únicamente para avisar y aceptar notificaciones relacionadas con las distintas tomas de medicación. El actor *Usuario*, en este caso, perteneciente al grupo objetivo vulnerable, puede consultar su calendario semanal de tomas, el tiempo restante hasta la siguiente, así como otra información relevante, como la hora del sistema, el estado de las tomas de días anteriores o datos relacionados con el clima. Una de las funcionalidades más importantes es permitir al usuario aceptar la notificación de una toma. Es decir, cuando llegue la hora programada por el administrador, el usuario puede indicar si ha tomado la pastilla o, en caso contrario, notificar una incidencia. Esta información se enviará mediante MQTT al servidor central, que, a su vez, la reportará al usuario administrador a través de la API de Telegram.

Además del actor *Usuario*, que será el principal protagonista, existe el actor *Técnico*, que engloba a un grupo de ayudantes como auxiliares domésticos, servicios sanitarios de emergencia o personal farmacéutico. Estos podrán interactuar con el paciente y acceder a

información clave en caso de emergencia o consulta, como su grupo sanguíneo, alergias, antecedentes médicos previos, así como datos de contacto.

El servidor central, a su vez, actualizará de forma periódica información adicional en el sistema. En concreto, actualizará diariamente los datos relativos al clima del día, así como el tiempo restante para la próxima toma organizada por el usuario administrador. Además, actuará como interfaz de comunicaciones, funcionando como un traductor entre la información proveniente de la API de Telegram (en HTTP) y el protocolo MQTT, generando un lenguaje entendible por openHASP. El servidor, mediante *scripts* en Python y Bash, también se encargará de actualizar información gestionada por el administrador, como el perfil del usuario, el calendario de tomas, y de enviar notificaciones de las tomas al administrador.

Por último, el actor *Administrador*, que puede ser un familiar del Usuario o incluso el propio actor *Técnico*, tiene la capacidad de gestionar toda la información relativa al pastillero a través de la API de Telegram, ya sea desde la aplicación móvil o desde la interfaz web. Por ejemplo, puede actualizar el perfil del usuario (una acción que también podrá realizar escaneando un código QR mostrado en la pantalla) o programar la toma de una pastilla, ya sea de forma puntual o periódica, apareciendo esta programación en el calendario mostrado en la pantalla TFT. Además, el servidor le notificará, a través del mismo canal de comunicación, información relativa a la última toma: si ha sido realizada exitosamente (es decir, confirmada por el Usuario) o si ha sido rechazada. De este modo, el Administrador puede estar al tanto de la situación y tomar acciones si es necesario, como contactar directamente con el Usuario.

A continuación, y con el objetivo de aportar mayor claridad sobre el funcionamiento del sistema y las acciones de cada actor, se muestra en la siguiente figura, en un diagrama UML (*Unified Modelling Language*, Lenguaje unificado de modelado) de casos de uso, donde se detallan las acciones y los sistemas implicados en cada una de ellas, como se han comentado anteriormente:

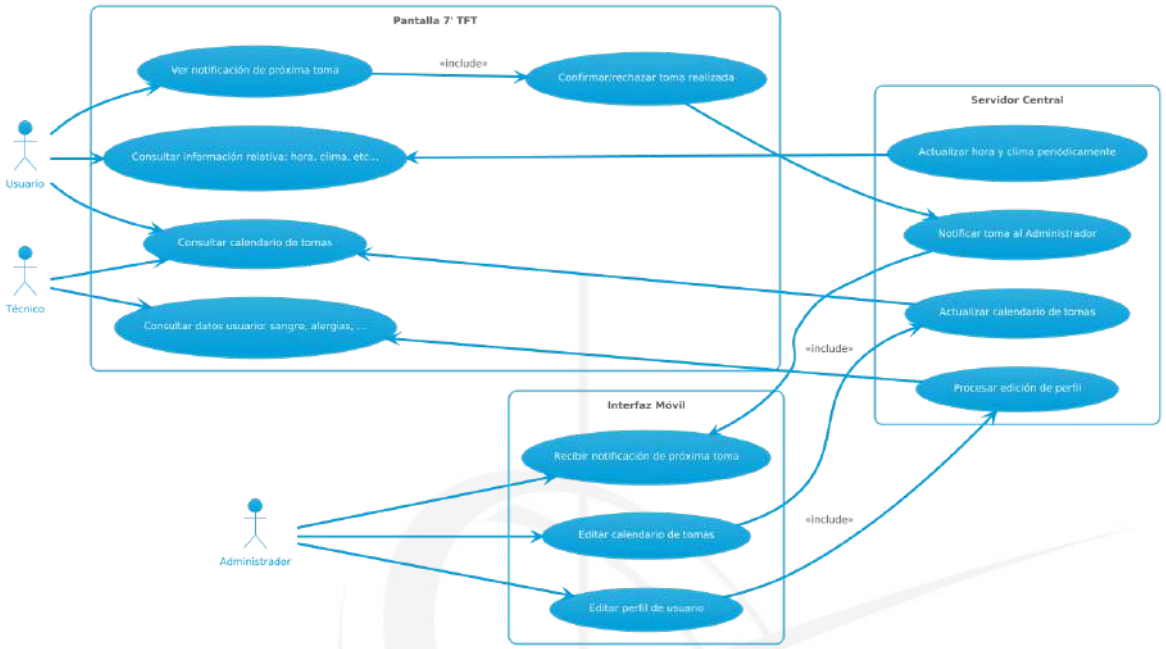


Figura 77. Diagrama casos de uso prototipo

En cuanto a los aspectos técnicos del diseño del sistema, se comenzará explicando el dispositivo embebido, encargado de ejecutar openHASP, y que ha sido objeto de las pruebas descritas en esta memoria.

La interfaz consta de un total de 10 páginas, de las cuales solo 9 son visibles para el actor *Usuario*.

La página 0 se muestra de forma persistente sobre todas las demás. Esta página incluye un *banner* que presenta, en su parte superior, información organizada de la siguiente manera: a la izquierda, la hora actualizada mediante un servidor NTP; en el centro, el nombre del producto; y a la derecha, datos relativos al clima, como la temperatura y el estado del cielo (nublado, soleado, lluvioso), así como la fecha en formato Día-Mes.

En la parte inferior de esta misma página se encuentra el controlador de navegación, que permite desplazarse entre las 9 páginas accesibles de la interfaz.

El aspecto de esta pagina 0 es el siguiente:

Figura 78. Explicación pagina 0 prototipo

Donde puede observarse, marcado en rojo, los elementos anteriormente explicados. En cuanto al código que definen los objetos del pages.jsonl, es el siguiente:

```
{"comment" : "======PAGINA 0======"}

{"comment" : "Encabezado"}

{"page":0,"id":4,"obj":"btn","enabled":false,"x":0,"y":0,"w":840,"
h":50,"bg_color":"#00A6FF","text":"","text_color":"#000000","radiu
s":0,"border_side":0,"text_font":24}

{"comment" : "Hora, actualizada dinamicamente"}

{"page":0,"id":5,"obj":"label","x":10,"y":10,"w":150,"h":30,"bg_co
lor":"#00A6FF","text_color":"#FFFFFF","align":0,"text_font":24,
"text": "98:80:00", "template": "%H:%M"}

{"comment" : "Informacion del tiempo actualizada dinamicamente"}

{"page":0,"id":6,"obj":"label","x":640,"y":10,"w":150,"h":30,"bg_c
olor":"#00A6FF","text":"{informacion relativa al
tiempo}","text_color":"#FFFFFF","align":2,"text_font":24,
"mode":"loop"}
```

```
{"comment" : "Informacion de marca"}

{"page":0,"id":7,"obj":"label","x":100,"y":10,"w":600,"h":30,"bg_c
olor":"#00A6FF","text":"Pillbot","text_color":"#FFFFFF","align":1,
"text_font":24}

{"comment" : "Control de paginas"}

{"page":0,"id":1,"obj":"btn","action":{"down": "page
prev"},"x":0,"y":430,"w":266,"h":50,"bg_color":"#00A6FF","text":"\
uE141","text_color":"#FFFFFF","radius":0,"border_side":0,"text_fon
t":24}

{"page":0,"id":2,"obj":"btn","action":{"down": "page
back"},"x":266,"y":430,"w":268,"h":50,"bg_color":"#00A6FF","text":
"\uE2DC","text_color":"#FFFFFF","radius":0,"border_side":0,"text_f
ont":24}

{"page":0,"id":3,"obj":"btn","action":{"down": "page
next"},"x":534,"y":430,"w":266,"h":50,"bg_color":"#00A6FF","text":
"\uE142","text_color":"#FFFFFF","radius":0,"border_side":0,"text_f
ont":24}
```

Se puede observar como, ya se explicó como una buena práctica en el diseño con openHASP, cada comentario encapsula un objeto y funcionalidad, como la hora, el control de páginas y el encabezado superior.

La página 1 se muestra al arrancar el sistema junto con la página 0, pero, a diferencia de esta última, no permanece visible en el resto de páginas. En ella se encuentra la información y el control más relevante para el *Usuario*. En esta página aparece un cuadro de texto fijo que indica la utilidad del contador y de la barra de progreso inferior. Este temporizador informa al usuario, de forma visual y clara, del tiempo restante hasta la próxima toma de la pastilla. Como complemento visual, y con el objetivo de reforzar la percepción del paso del tiempo, detrás del contador se encuentra una barra de progreso que cumple la misma función. Debajo de este conjunto de elementos se encuentra un botón que, en función del día actual, redirige a la página correspondiente con la preinscripción del día, ya tomada y aún pendiente, por el Usuario. Esto permite consultar qué es lo próximo que debe tomar, qué ha tomado ya y qué ha omitido. Estas páginas van desde la 2 hasta la 8, ambas incluidas, representando así cada día de la semana.

El aspecto de la página 1 es el siguiente:

Figura 79. Explicacion página 1 prototipo

Donde puede observarse, marcado en rojo, los elementos anteriormente explicados. En cuanto al código que definen los objetos del pages.jsonl, es el siguiente:

```
{"comment" : "======PAGINA 1======"}

{"comment" : "Informacion comun"}

{"page":1,"id":0,"prev":9}

{"page":1,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":50,
"bg_color":"#00A6FF","text":"Tiempo para la proxima
toma","text_color":"#000000","align":1,"text_font":30}

{"page":1,"id":2,"obj":"bar","x":240,"y":197,"w":320,"h":100,
"bg_color":"#00A6FF80","border_color":"#FFFFFF","val":32,
"opacity":80, "radius":500}

{"page":1,"id":3,"obj":"label","x":100,"y":200,"w":600,"h":10
0,"bg_color":"#00A6FF","text":"00:00:00","text_color":"#000000","a
lign":1,"text_font":80}

{"page":1,"id":10,"obj":"btn","x":300,"y":320,"w":200,"h":60,
"radius":35,"bg_color":"#00A6FF","text":"\uE0ED Ver
recetario","text_color":"#FFFFFF","text_font":24,"action":"p7"}
```

Como ya se ha mencionado en la explicación de la página 1, las páginas de la 2 a la 8 se utilizan para representar el calendario de tomas diarias, asignando una página a cada día de la semana. Aunque es posible navegar manualmente entre estas páginas, su contenido será actualizado semanalmente por el *Administrador*. No obstante, el *Usuario* accede directamente a la página correspondiente al día actual a través del botón descrito en la página 1.

Con el objetivo de mejorar la comprensión del prototipo, se tomará como ejemplo la página correspondiente al lunes.

En estas páginas, en la parte superior, se encuentra un objeto de texto fijo que indica de forma visible el día de la semana que el *Usuario* está consultando. Justo debajo, se presenta un visor de pestañas agrupadas en cuatro momentos del día, comúnmente utilizados por la comunidad sanitaria para la organización y prescripción de medicamentos: *Desayuno*, *Comida*, *Merienda* y *Cena*. Dentro de cada pestaña, los medicamentos se muestran en forma de lista, incluyendo el nombre del medicamento, con su principio activo, la dosis a tomar, tanto en miligramos como en unidades según el formato de presentación del medicamento y, en caso de que fuera necesario, una hora referencia. A la derecha de cada entrada, se dispone un *checkbox* que refleja el estado de la toma: marcado cuando ha sido confirmada y notificada al servidor, y desmarcado en caso contrario. El *Administrador* podrá editar esta lista en cualquier momento, añadiendo o eliminando medicamentos según sea la prescripción.

El aspecto de las páginas del calendario, es el siguiente:

Figura 80. Explicación páginas 2 a 8, prototipo

En la figura, se han marcado en rojo los elementos que han sido explicados anteriormente.

```
{"comment" : "======PAGINA 2======"}

{"page":2,"id":1,"obj":"label","x":100,"y":70,"w":600,"h":30,
"bg_color":"#00A6FF","text":"Lunes","text_color":"#000000","align"
:1,"text_font":30}

{"comment": "TabView para las tomas de pastillas"}

{"page":2,"id":14,"obj":"tabview","x":10,"y":120,"w":780,"h":
300, "bg_color":"#00A6FF80"}

{"comment": "Pestañas dentro del TabView"}

{"page":2,"id":51,"obj":"tab","parentid":14,"text":"Desayuno"
}

{"page":2,"id":52,"obj":"tab","parentid":14,"text":"Comida"}

{"page":2,"id":53,"obj":"tab","parentid":14,"text":"Merienda"
}

{"page":2,"id":54,"obj":"tab","parentid":14,"text":"Cena"}

{"comment": "Checkboxes para horarios en Mañana"}

{"page":2,"id":101,"obj":"label","x":10,"y":10,"w":700,"h":30
,"parentid":51,"text":"Adiro (ácido acetilsalicílico) | 100mg (1
cápsula) | 9:00","text_color":"#000000","align":0,"text_font":24}

{"page":2,"id":201,"obj":"checkbox","x":700,"y":10,"w":30,"h"
:30,"parentid":51,"text":"","checked":false}

{"page":2,"id":102,"obj":"label","x":10,"y":50,"w":700,"h":30
,"parentid":51,"text":"Seguril (Furosemida) | 60 mg (1+1/2
comprimidos) |
9:00","text_color":"#000000","align":0,"text_font":24}

{"page":2,"id":202,"obj":"checkbox","x":700,"y":60,"w":30,"h"
:30,"parentid":51,"text":"","checked":false}
```

Por último, el *Usuario*, *Técnico* o *Administrador* pueden consultar información médica relevante del Usuario. Esta información se encuentra en la última página accesible de la interfaz. En dicha página se muestran datos como: nombre, apellidos, DNI, contacto de emergencia, grupo sanguíneo y alergias. Esta información puede resultar de gran utilidad para

el personal médico encargado del Usuario, especialmente en situaciones de emergencia. Además, en caso de que se desee editar esta información, se incluye un botón que redirige al *Administrador* (o al propio *Usuario*) a una página oculta donde se muestra un código QR. Al escanearlo, se abre directamente la interfaz de Telegram, permitiendo realizar la edición de forma sencilla y segura.

El aspecto de perfil de Usuario, es el siguiente:

Figura 81. Explicacion página 9 prototipo

Los elementos previamente descritos se pueden ver resaltados en rojo en la figura. El código correspondiente a los objetos del pages.jsonl es el siguiente:

```
{"comment" : "Recuadro para la foto de perfil"}

{"page":9,"id":1,"obj":"img","x":22,"y":80,"w":150,"h":150,"bg_color":"#FFFFFF","border_color":"#000000","src":"L:/profile-3.png"}

{"comment" : "Etiquetas de información del usuario"}

{"comment": "Fila - Nombre"}

{"page":9,"id":52,"obj":"obj","x":210,"y":80,"w":550,"h":40,"click":0,"bg_color":"#00A6FFCC","radius":40, "opacity":100}

{"page":9,"id":3,"obj":"label","x":670,"y":85,"w":400,"h":30,"text":"Nombre","text_color":"#000000","align":0,"text_font":24,"opacity":120}
```

{"page":9,"id":4,"obj":"label","x":220,"y":85,"w":300,"h":30,
"text":"Iván","text_color":"#000000","align":0,"text_font":24}

{"comment": "Fila - Apellidos"}

{"page":9,"id":53,"obj":"obj","x":210,"y":130,"w":550,"h":40,
"click":0,"bg_color":"#00A6FFCC","radius":40, "opacity":100}

{"page":9,"id":5,"obj":"label","x":660,"y":135,"w":400,"h":30
,"text":"Apellidos","text_color":"#000000","align":0,"text_font":2
4, "opacity":120}

{"page":9,"id":6,"obj":"label","x":220,"y":135,"w":300,"h":30
,"text":"García
Salgado","text_color":"#000000","align":0,"text_font":24}

{"comment": "Fila - DNI"}

{"page":9,"id":54,"obj":"obj","x":210,"y":180,"w":550,"h":40,
"click":0,"bg_color":"#00A6FFCC","radius":40, "opacity":100}

{"page":9,"id":7,"obj":"label","x":710,"y":185,"w":400,"h":30
,"text":"DNI","text_color":"#000000","align":0,"text_font":24,
"opacity":120}

{"page":9,"id":8,"obj":"label","x":220,"y":185,"w":300,"h":30
,"text":"51141855K","text_color":"#000000","align":0,"text_font":2
4}

{"comment": "Fila - Grupo Sanguíneo"}

{"page":9,"id":55,"obj":"obj","x":210,"y":230,"w":550,"h":40,
"click":0,"bg_color":"#00A6FFCC","radius":40, "opacity":100}

{"page":9,"id":9,"obj":"label","x":585,"y":235,"w":400,"h":30
,"text":"Grupo
Sanguíneo","text_color":"#000000","align":0,"text_font":24,
"opacity":120}

{"page":9,"id":10,"obj":"label","x":220,"y":235,"w":300,"h":3
0,"text":"A+","text_color":"#000000","align":0,"text_font":24}

{"comment": "Fila - Alergias"}

{"page":9,"id":56,"obj":"obj","x":210,"y":280,"w":550,"h":40,
"click":0,"bg_color":"#00A6FFCC","radius":40, "opacity":100}

{"page":9,"id":11,"obj":"label","x":670,"y":285,"w":400,"h":3
0,"text":"Alergias","text_color":"#000000","align":0,"text_font":2
4, "opacity":120}

```
{"page":9,"id":12,"obj":"label","x":220,"y":285,"w":300,"h":3
0,"text":"Ninguna","text_color":"#000000","align":0,"text_font":24
}

{"comment": "Fila - Contacto de Emergencia"}

{"page":9,"id":57,"obj":"obj","x":210,"y":330,"w":550,"h":40,
"click":0,"bg_color":"#00A6FFCC","radius":40, "opacity":100}

{"page":9,"id":13,"obj":"label","x":545,"y":335,"w":400,"h":3
0,"text":"Contacto
Emergencia","text_color":"#000000","align":0,"text_font":24,
"opacity":120}

{"page":9,"id":14,"obj":"label","x":220,"y":335,"w":300,"h":3
0,"text":"+34 601 08 05
55","text_color":"#000000","align":0,"text_font":24}

{"page":9,"id":60,"obj":"btn","x":20,"y":280,"w":180,"h":60,"
radius":35,"bg_color":"#00A6FF","text":"\uE004
Editar","text_color":"#FFFFFF","text_font":24,"action":"p10"}

{"page":9,"id":0,"next":1}
```


En esta última página de perfil se hace referencia al QR para editar la información del Usuario. Esta página, en la navegación común por botones, está deshabilitada. El aspecto es el siguiente:

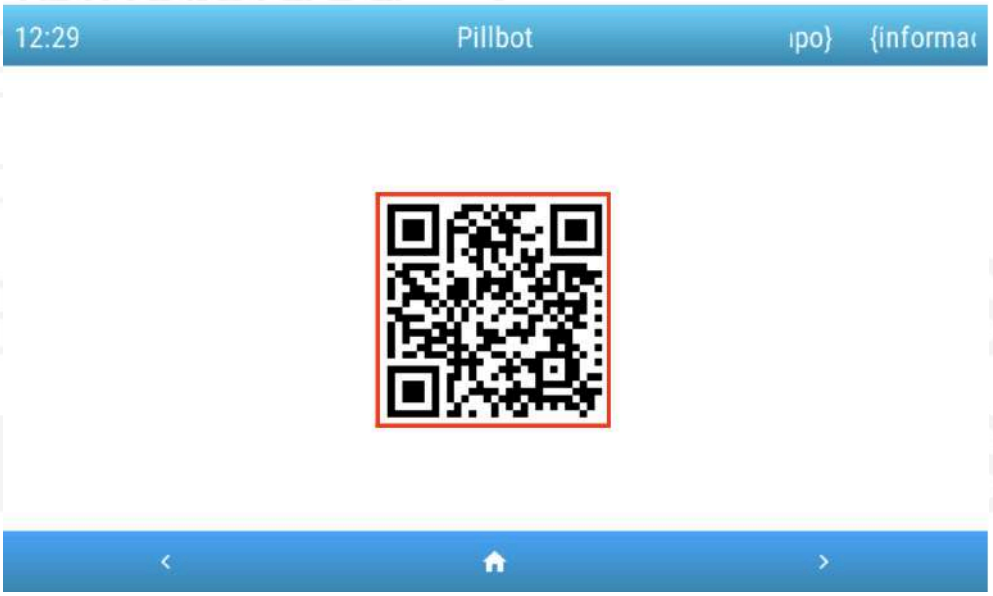


Figura 82. Explicacion página 10 prototipo

Donde puede observarse, marcado en rojo, el elemento explicado. En cuanto al código que definen los objetos del pages.jsonl, es el siguiente:

```
{"comment" : "======PAGINA 9======"}

     {"page":10,"id":1,"obj":"qrcode","text":"www.openhasp.com/0.7
.0/design/objects/","x":300,"y":150,"size":200}
```

Por último, además de las páginas, *Pillbot* cuenta con un sistema de notificaciones. Estas notificaciones pueden ser de dos tipos: periódicas y manuales. Las notificaciones periódicas se generan, por ejemplo, a partir de la programación de una toma de pastilla. Estas se reflejan en el temporizador de la página 1 y, llegado el momento, lanzan un evento que notifica al *Usuario*. Por otro lado, el *Administrador* puede, en cualquier momento, enviar manualmente un mensaje al sistema para que sea mostrado al *Usuario*, permitiendo así una comunicación directa y oportuna.

El aspecto de esta notificación es el siguiente:

Figura 83. Explicación notificación emergente prototipo

Siguiendo con el desarrollo del prototipo, una vez la interfaz, en el dispositivo embebido, está preparada para alojar y compartir los datos, es el momento de desarrollar el gestor para el actor *Administrador*. Este programa "gestor" corre en el servidor central, y tiene dos funciones principales: traducir desde y hacia openHASP la comunicación y permitir

desde un agente externo (móvil, web, etc…) con acceso a Internet la completa configuración del dispositivo, así como recibir notificaciones de su estado.

Sin entrar en detalle sobre la programación a bajo nivel de cada una de las funciones del programa, se hará un breve repaso a sus funcionalidades más importantes.

El programa, denominado pillbot.py, está desarrollado en Python 3.11 y se ejecutará, para garantizar su disponibilidad 24/7, en un VPS (*Virtual Private Server*, Servidor Privado Virtual). Este mismo servidor aloja el broker MQTT utilizado por openHASP, lo que permite optimizar recursos. Aprovechando las características del sistema operativo Ubuntu y el gestor de servicios systemd, el script se configura como un servicio mediante systemctl, permitiendo que, en caso de error o caída, el sistema lo reinicie automáticamente sin afectar a la experiencia del usuario.

El programa utiliza las siguientes librerías de python:

```
pip install python-telegram-bot schedule requests
```

La librería python-telegram-bot [72] se utiliza para implementar las funciones de escucha del chat de Telegram y permitir, mediante comandos parametrizables, la configuración por parte del *Usuario*. Por su parte, la librería schedule [73], junto con algunas funciones nativas de Python, se emplea para programar tareas periódicas que se ejecutan en un hilo de proceso independiente del utilizado para la escucha de Telegram. Por último, la librería requests [74] es la encargada de realizar todas las peticiones HTTP necesarias, por ejemplo, a una API de meteorología para posteriormente actualizarlo en la interfaz.

Comprender el uso de estas librerías permite identificar tres funcionalidades principales dentro de la aplicación *pillbot.py*. La primera de ellas corresponde a las funciones periódicas, que en concreto son dos: Una primera función, que se ejecuta únicamente al inicio de un nuevo día (a las 00:00), y realiza una petición a una API de meteorología de código abierto para obtener información sobre la temperatura, el viento y el estado del cielo, después estos datos se actualizan automáticamente en la interfaz gráfica del sistema. Una segunda función, que se ejecuta cada segundo, comprueba constantemente la hora del sistema para determinar cuánto tiempo falta para la próxima fase de medicación. Las horas por defecto para las tomas son: 9:00 (*Desayuno*), 14:00 (*Comida*), 18:00 (*Merienda*) y 21:00 (*Cena*), y pueden ser configuradas por el *Administrador* en cualquier momento. Ambas funciones se ejecutan en un hilo de proceso independiente (*thread*) para evitar sobrecarga en el hilo

principal, que se encarga de la funcionalidad de Telegram así como el envío y recepción de mensajes, aprovechando así la arquitectura multihilo del microcontrolador ESP32 empleado.

Otra de las funcionalidades clave, además de las tareas periódicas, es la gestión del bot de Telegram. Este bot, que puede implementarse siguiendo el ejemplo incluido en el Anexo (ver Anexo), permite al Administrador configurar el comportamiento del sistema. Las funciones del bot se agrupan en *callbacks*, los cuales actúan en respuesta a ciertos tipos de mensajes. Es decir, el sistema está en una escucha continua de los mensajes enviados por el *Administrador* y, si alguno de ellos coincide con un *callback* definido, se ejecuta la función correspondiente.

Para hacer este proceso lo más sencillo posible de usar, las opciones están organizadas y parametrizadas dentro de un menú incorporado en el chat. El actor *Administrador* solo debe seleccionar la opción deseada y, de forma guiada, el sistema solicita exclusivamente los datos necesarios, gestionando las respuestas del usuario de manera asíncrona. Esta arquitectura refuerza la importancia de ejecutar las tareas periódicas en un hilo independiente, evitando así bloqueos con la interacción en tiempo real del bot.

El aspecto del menú, puede verse en la siguiente imagen:

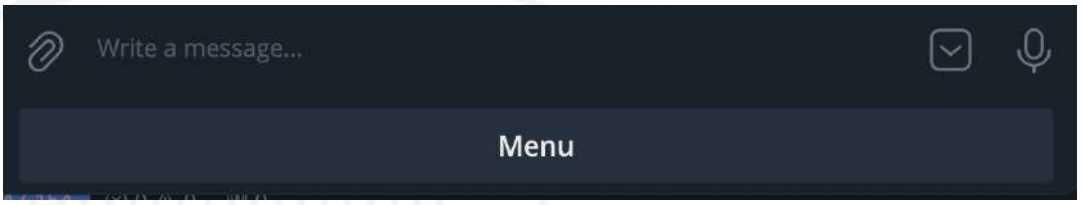


Figura 84. Aspecto Menú bot Telegram

En la primera ejecución de todo el sistema el perfil del Usuario no existirá, por lo que la única opción disponible en el Menú será Crear nuevo perfil, como se puede ver en la siguiente figura:

Figura 85. Opciones Menú en arranque

Además, puede observarse cómo, al pulsar la opción Menú, realmente la acción que se produce es la escritura en el chat "Menu", lo que ejecuta el *callback* de mostrar Menu, solo estando disponible, por el momento, esta opción.

Seleccionando esta opción, el programa pedirá al Administrador los datos de forma secuencial y asíncrona, como puede verse en la siguiente figura:

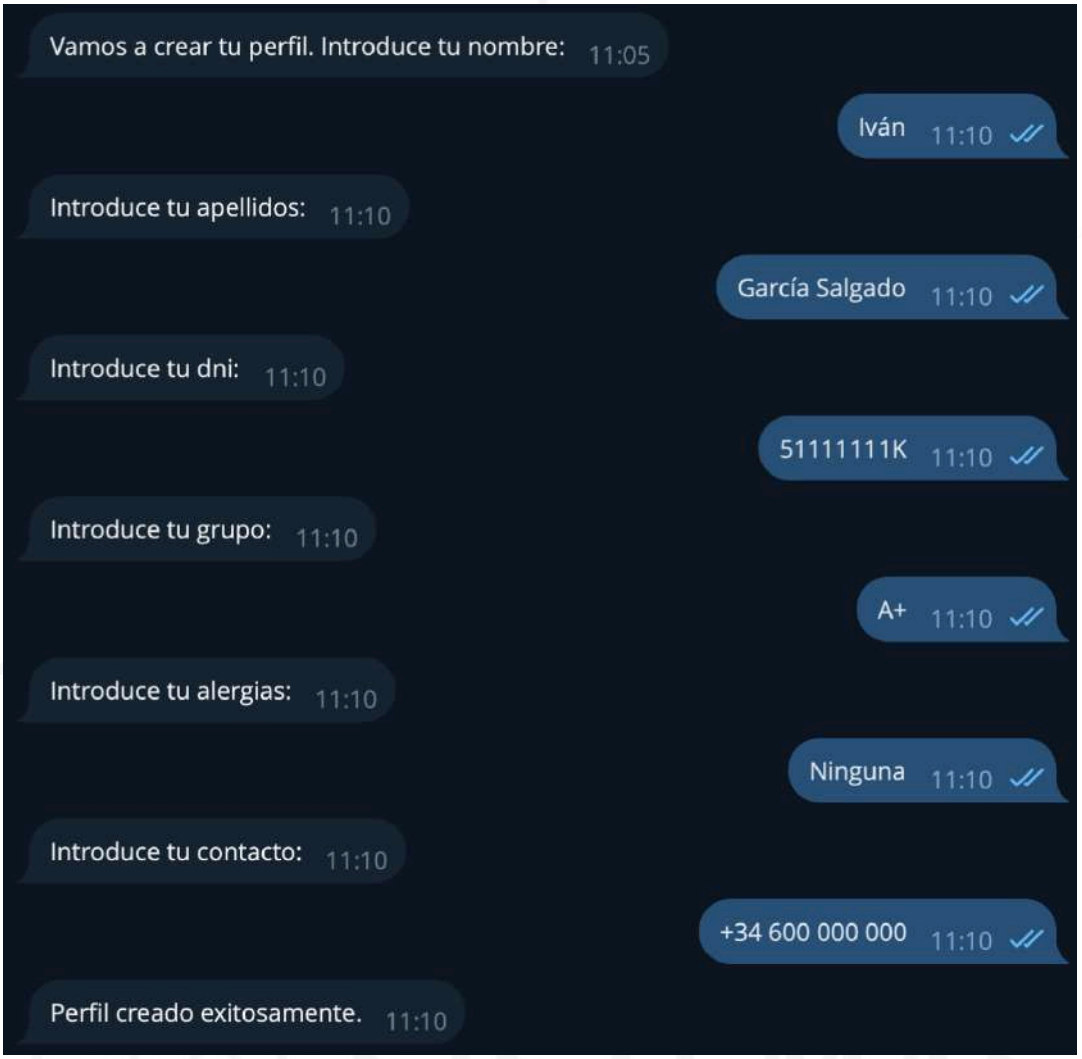


Figura 86. Creación perfil en Telegram

Este perfil se guardará en la memoria interna del *bot* mientras este siga en ejecución y, si el Administrador quisiera seleccionar otra opción del menú:

Figura 87. Opciones Menú con perfil existente

Y, como puede observarse, ahora existen más opciones disponibles.

Otra funcionalidad crítica del sistema, que funciona muy similar a la de Crear nuevo perfil, es Añadir pastilla. En esta opción, el programa pedirá la siguiente información:

Figura 88. Programador pastilla (fuente propia)

Siguiendo con el principio de operabilidad en sistemas accesibles, todas las opciones, excepto aquellas que requieren la introducción de texto, se presentan al Administrador de forma parametrizada. Esto significa que el Administrador solo deberá seleccionar, mediante botones interactivos, en qué franja horaria desea programar la toma de la pastilla, así como los días de la semana en los que esta debe repetirse.

Otro de los módulos presentes en el programa pillbot.py es el encargado de la comunicación vía MQTT con openHASP. Para ello, se han definido funciones que permiten enviar mensajes de edición de campos de forma sencilla tal y como se ha explicado en apartados anteriores. Estas funciones se encargan automáticamente del proceso de autenticación en el servidor MQTT y del envío del mensaje, lo que permite abstraer dicha lógica del resto del programa. De este modo, se simplifica el código de las funciones principales, haciendo que el sistema sea más escalable y abierto a futuras funcionalidades, al centralizar esta responsabilidad en un conjunto reducido de funciones especializadas.

Todas las funciones, así como los módulos, mencionados anteriormente, están disponibles en código abierto en el repositorio Github del proyecto [75]. Además, se incluye (ver Anexo) el nombre de cada función y un enlace para el estudio de cada una de ellas.

5. Pruebas y validación

Con el objetivo de demostrar la viabilidad y el correcto funcionamiento del sistema, se llevarán a cabo una serie de pruebas que permitan validar sus distintos componentes.

El proyecto consta de dos partes diferenciadas. La primera es la guía de implementación de interfaces con el *firmware* openHASP, y la segunda, relacionada con la anterior, es la creación de un prototipo funcional basado en los conocimientos teóricos anteriormente explicados.

Respecto a la primera parte, la guía de openHASP se considerará válida si, siguiendo sus explicaciones, el prototipo supera con éxito las pruebas y procesos de validación. Esto demostraría que la guía, desde un punto de vista teórico, cumple su objetivo principal: proporcionar una comprensión completa de la plataforma y dar la posibilidad al lector de construir un prototipo completo.

En cuanto a la segunda parte, centrada en el prototipo, este deberá cumplir con su propósito: ayudar en la organización y toma de medicación por parte de grupos vulnerables. Esta organización debe estar supervisada por un *Administrador*, quien podrá configurar el dispositivo de forma remota. Por su parte, el *Usuario* debe ser capaz de operar y comprender el dispositivo, y realizar el uso diario para gestionar la medicación.

Por tanto, las pruebas se centrarán, sobre todo, en la capacidad de configuración remota y en la visibilidad de esta información en la interfaz creada con openHASP.

El comportamiento esperado del dispositivo tras el arranque es actualizar los datos de hora, desde el servidor NTP, la información meteorológica, el calendario actual y el temporizador a la próxima toma.

Cuando el dispositivo arranca, este publica en el *topic* /hasp/plate/LWT online, donde el servidor central está escuchando, en caso de recibir un mensaje que coincide con este estado, actualiza esta información, de la siguiente forma.

Como se puede observar en la siguiente figura, el proceso de arranque de Pillbot es el siguiente:

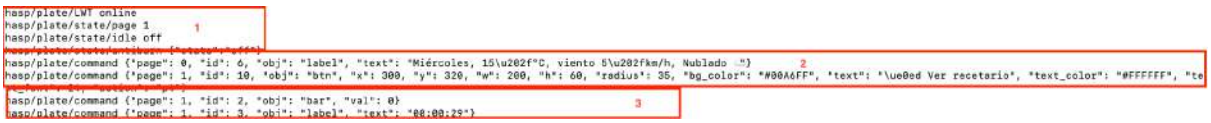


Figura 89. Comandos topic MQTT servidor-openHASP

Como se puede ver en la imagen, referente a los *logs* del servidor, en el recuadro indicado con un 1, openHASP publica su estado en el topic LWT. El servidor central, que está escuchando este *topic* inicia la escucha esta iniciativa de arranque y responde con la petición de la información meteorológica realizada a una API gratuita, open.meteo, utilizando la librería request, de python. Además, según el día del sistema, p.e, miércoles, actualiza el botón que redirige al Usuario a la página referente al día de hoy, estos mensajes pueden verse en el recuadro identificado con un 2. Por último, según la franja horaria obtenida del servidor NTP, calcula el tiempo restante para la siguiente toma y actualiza el cronómetro y la barra de progreso que lo precede, tal y como se aprecia en el recuadro 3.

Con ello, en la pantalla TFT se podrá observar lo que se muestra en la siguiente figura:

Figura 90. Página 1 prototipo finalizada

donde toda la información está siendo actualizada de forma dinámica. Desde este punto, el sistema está preparado y conectado para terminar su configuración.

Por otro lado, en la parte de la vista de *Administrador,* en la interfaz Telegram, el dispositivo no dispondrá de ningún perfil, y la única opción que debe ofrecer al *Administrador* es esta misma, cómo se puede observar en la siguiente figura:

Figura 91. Opción menú y start

Donde, una vez detectada la no existencia de un perfil de Pillbot, pedirá al usuario la información necesaria para crear el perfil *Usuario,* de la siguiente forma:

Figura 92. Configurando perfil

El servidor procesa esta información y la traduce en instrucciones listas para ser enviadas por MQTT y entendidas por openHASP, como se puede ver en la siguiente figura:

```
hasp/plate/command {"page": 9, "id": 4, "obj": "label", "text": "Iván"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:39"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:38"}
hasp/plate/command {"page": 9, "id": 6, "obj": "label", "text": "García"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:37"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:36"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:35"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:34"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:33"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:32"}
hasp/plate/command {"page": 9, "id": 8, "obj": "label", "text": "51141855K"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:31"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:30"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:29"}
hasp/plate/command {"page": 9, "id": 10, "obj": "label", "text": "A+"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:28"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:27"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:26"}
hasp/plate/command {"page": 1, "id": 2, "obj": "bar", "val": 100}
hasp/plate/command {"page": 1, "id": 3, "obj": "label", "text": "11:19:25"}
hasp/plate/command {"page": 9, "id": 12, "obj": "label", "text": "Ninguna"}
```

Figura 93. Mensajes enviados configuración

Donde, entre los mensajes de actualización de cada segundo para el cronómetro, se indica en rojo los mensajes que se han procesado y posteriormente enviado para editar la página de información médica de *Usuario*. En openHASP quedaría de la siguiente forma:

Figura 94. Aspecto perfil de usuario tras configuracion

Una vez el perfil está creado y asociado a un dispositivo Pillbot, el *Administrador* puede comenzar a organizar las tomas diarias divididas en franjas horarias. Para ello, ahora tendrá disponible en la interfaz de Telegram, específicamente en el menú, la opción, como se puede ver en la siguiente figura:

Figura 95. Opciones menú tras crear perfil

A la cual, seleccionando, Añadir pastilla, comenzará el proceso guiado para la petición de información de la prescripción. En concreto se pedirá el nombre más una información, como puede ser la dosis, el número de unidades o el objetivo de la misma, totalmente personalizable para facilidad del *Usuario.* Además, habrá que añadir la franja horaria de la toma así como el día, o días, en los que deberá hacerlo. La petición de esta información se muestra en las siguientes figuras:

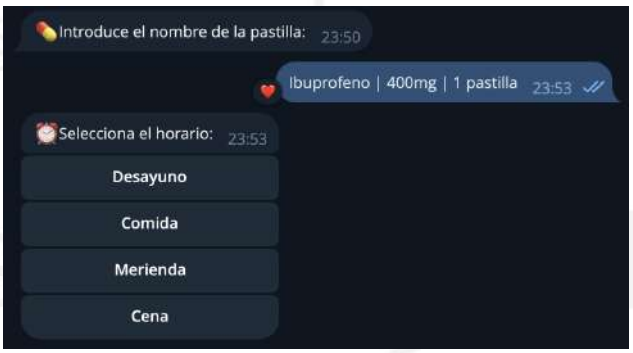


Figura 96. Opción añadir pastilla I

En la selección de franja horaria solo será posible la selección de uno a uno. En cambio, si una pastilla se repite en los días, en la misma franja horaria, pueden seleccionarse todos los días de la semana que se requieran, como se puede comprobar a continuación.

Figura 97. Opción añadir pastilla II

Por último, el *Administrador* debe confirmar su selección y el servidor responderá si ha sido añadida correctamente. A continuación, el servidor, de la misma forma que ha hecho con anterioridad, procesa esta información y la actualizará en el TFT, de la siguiente forma:

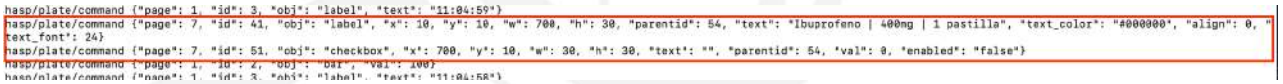


Figura 98. Vista creacion objeto en servidor

Para cada nueva prescripción el servidor procesa dos objetos, el texto para la información y el marcador para indicar cuando esta ha sido tomada. Además, esta información se guarda de forma persistente en el servidor para, más adelante, poder procesar esta información para el envío de notificaciones y mensajes.

En openHASP quedaría de la siguiente forma:

Figura 99. Aspecto calendario un horario y día

Donde se puede observar que, de momento, la pastilla se ha programado para la franja horaria de la Cena y el marcador está deshabilitado y esperando a activarse en la toma correspondiente.

En función de qué fecha devuelve el servidor NTP, el servidor calcula el tiempo para la próxima franja horaria, las cuales se dividen en 4, desayuno, comida, merienda y cena, de acuerdo con las 4 franjas en las que un paciente medicado suele agrupar su medicación. Cada franja se corresponde con las 9, 14, 18 y 21 horas, respectivamente, por lo que, cuando el cálculo de esta diferencia, apreciable en el cronómetro y en la barra de progreso llegue a 0, aparecerá en la pantalla, bloqueando el uso del TFT hasta su respuesta, una notificación donde el *Usuario* debe informar sobre el estado de la toma de la medicación, de la siguiente manera:

Figura 100. Aspecto notificación

Y una vez el usuario acepte, o rechace, esta notificación, su estado se recogerá tanto en openHASP, para el estudio de *Técnicos o Auxiliares*, o incluso el propio *Usuario*, y en la interfaz Telegram para que *Administrador*, remotamente, pueda conocer el estado de Usuario.

En la siguiente figura puede observarse esto en la interfaz implementada:

Figura 101. Aspecto calendario tras aceptar toma

Donde ambas medicaciones aparecen como marcadas, es decir, el *Usuario* asegura que se las tomó, y el *Administrador* recibe:

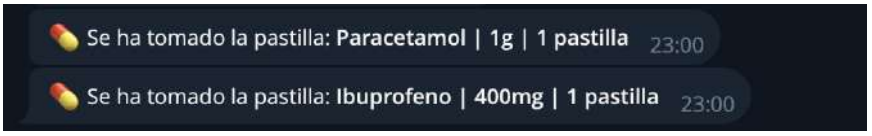


Figura 102. Mensaje enviado tras aceptar toma

En caso de que estas hayan sido correctamente tomadas. Esta característica puede conseguirse gracias a el uso ordenado de identificadores en el diseño de openHASP, así como el almacenamiento en memoria interna del servidor de identificadores de objetos, información, etc… para posteriormente, incluso 1 semana después, de forma periódica, pueda utilizarse para notificar a *Usuario*.

En conclusión, tanto el entorno de pruebas como el prototipo embebido *Pillbot*, basado en openHASP, han sido desarrollados siguiendo la guía y la documentación oficial de la plataforma, así como las directrices recogidas en este mismo documento. Esto reafirma y valida la aplicación práctica de los conocimientos teóricos adquiridos, trasladándose con éxito a un plano técnico y demostrando su utilidad y fiabilidad.

Por ende, quedan cumplidos ambos objetivos fundamentales del proyecto:

1. Crear un prototipo completamente funcional, preparado para asistir a personas en su organización y toma de medicación.
2. Lograrlo sin salir del marco definido por este documento, lo que garantiza la coherencia y aplicabilidad del contenido presentado desde el inicio del mismo.

6. Herramientas utilizadas

A continuación, se hace un breve repaso a las **herramientas de software** que han ayudado a la elaboración y desarrollo del proyecto.

6.1. Python 3.12

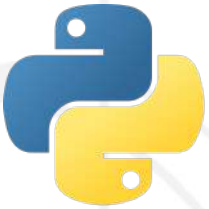


Figura 103. Logo Python

Python es un lenguaje de programación de alto nivel que destaca por su claridad y sencillez [69]. La versión 3.12 proporciona mejoras significativas en términos de rendimiento y seguridad.

Python ha sido utilizado para desarrollar el *script* del servidor *back-end* para la conexión con el dispositivo. Sirve de puente entre la API de Telegram y el usuario *Administrador*.

6.2. OVH Cloud

Figura 104. Logo OVHCloud

OVH Cloud es una plataforma de servicios en la nube que ofrece soluciones para aplicaciones IoT [77].

En este proyecto, OVH Cloud ha sido utilizado para alojar el servidor del sistema.

6.3. Telegram

Figura 105. Logo Telegram

Telegram es una plataforma de mensajería que ofrece una API gratuita [78] que ha sido integrada para proporcionar notificaciones en tiempo real, alertas automatizadas del sistema a los usuarios y configuración del dispositivo.

Esto permite a los familiares, técnicos auxiliares y usuario, recibir actualizaciones instantáneas sobre el estado, configurar perfiles y tomas.

6.4. GitHub

Figura 106. Logo Github

GitHub [54] es la aplicación más extendida y estandarizada en el mercado que, gracias a su versatilidad, y la inclusión de herramientas como **GitHub Desktop** y **GitHub Actions**, se ha escogido para el control de versiones del código desarrollado.

6.5. Cursor.AI IDE

Figura 107. Logo Cursor.ai

Cursor.AI [79] es un entorno de desarrollo, de sus siglas en inglés IDE (*Integrated Development Environment*, Entorno de Desarrollo Integrado), diseñado específicamente para el uso de diversas herramientas de Inteligencia Artificial como Github Copilot, o ChatGPT, que facilitan el desarrollo y la detección de errores, sobretodo trabajando en ficheros sensibles.

6.6. StarUML

Figura 108. Logo StarUML

StarUML [80] es una herramienta software para la creación de diagramas UML, utilizados en proyectos de software para representar los "planos" del sistema. Estos diagramas se emplean habitualmente por su capacidad para explicar modelos complejos de forma sencilla a usuarios que no necesariamente cuentan con un perfil técnico.

La herramienta, de código abierto, desarrollada por **MKLabs** [81], fue seleccionada por su compatibilidad con distintos tipos de diagramas.

6.7. draw.io

Figura 109. Logo draw.io

draw.io [82] es una plataforma web para la creación de esquemas. Estos esquemas no necesariamente siguen un modelo formal, sino que se utilizan para representar gráficamente situaciones concretas del comportamiento del sistema.

En el desarrollo del proyecto, la plataforma se utilizó específicamente para representar los flujos de comunicación entre los distintos equipos conectados.

6.8. Adobe Photoshop

Figura 110. Logo Adobe Photoshop

Adobe Photoshop [83], del grupo de programas de diseño Adobe [84], sirve para diseñar imágenes complejas, como un logo.

En concreto se utilizó para el diseño del logo del prototipo Pillbot.

7. Discusión de resultados

En su conjunto, el proyecto ha cumplido con los objetivos propuestos inicialmente. El objetivo principal, como se mencionó en el capítulo correspondiente, era crear un documento que sirviera como una guía completa sobre todos los aspectos teóricos del *firmware* openHASP.

Otro objetivo, secundario pero igualmente importante, era el desarrollo de un ejemplo completamente funcional que respaldara los conocimientos teóricos expuestos. El prototipo desarrollado, **"Planificador de pastillero inteligente | Pillbot"**, es plenamente usable y cubre la necesidad real de un grupo de personas afectadas por una problemática, por lo que junto a esto despeja dudas sobre una de las dudas iniciales, la elección de este *firmware*.

A lo largo del proceso de documentación se ha puesto, en todo momento, a disposición del lector todo el material necesario, incluyendo ejemplos de código parametrizables, lo que facilita su compatibilidad con distintos casos particulares. Además, se han incorporado imágenes del proceso, permitiendo al lector seguir la guía sin necesidad de recurrir a fuentes externas debido a posibles lagunas de información.

No obstante, con el objetivo de realizar una autocrítica constructiva, es importante reconocer que algunos aspectos podrían haberse abordado con mayor profundidad. Por ejemplo, habría sido útil incluir una explicación más detallada sobre conceptos relacionados, como el manejo de datos en **JSON y JSONL**, ya que estos formatos desempeñan un papel clave en la gestión de objetos dentro de openHASP.

Esta decisión de obviar ciertos conocimientos del lector, como el mencionado anteriormente, permitió abordar todos los aspectos de openHASP con profundidad y con un orden que se estima adecuado. Se comenzó con una explicación teórica y una introducción al contexto de las pantallas y el software LVGL, nativo de openHASP, seguido de un análisis de opciones y visión de mercado, hasta llegar a los conceptos más básicos de openHASP y, finalmente, a la construcción completa de un prototipo funcional.

Respecto al objetivo **técnico** del proyecto, a nivel general, ha alcanzado sus objetivos con la ayuda del respaldo teórico. La elección del hardware, en particular la pantalla **Sunton**,

cuyo modelo se explicó en el apartado correspondiente, cumple con las expectativas de tamaño para el ámbito doméstico y proporciona un rendimiento adecuado en términos de memoria y velocidad de procesamiento, lo que se traduce en una experiencia fluida para el usuario.

Sin embargo, manteniendo el enfoque autocrítico, la calidad de la pantalla varía según el ángulo de visión, lo que puede representar un inconveniente en ciertas ubicaciones. Dado que la accesibilidad es uno de los puntos clave del proyecto, este aspecto se considera una desventaja en la selección de hardware para futuros desarrollos.

Por el lado del sistema *backend*, se ha realizado un gran esfuerzo en su integración y usabilidad para gestionar la comunicación entre el hardware y el software. Este sistema garantiza una conexión permanente y reduce al mínimo la intervención del usuario *Administrador*, mejorando así la accesibilidad y facilidad de uso del dispositivo.

Además, se ha diseñado el sistema gestor de manera que no requiera de una configuración técnica, sino que sea completamente configurable de forma visual. Esto permite que usuarios con conocimientos tecnológicos mínimos puedan interactuar y configurar el sistema a través de interfaces ya conocidas y accesibles, como chats de texto en Telegram, similar a las aplicaciones de mensajería más utilizadas, como WhatsApp [76], plataformas con las que ya están familiarizados.

Finalmente, todas estas conclusiones sobre el resultado reflejan que el documento cumple con sus requisitos fundamentales: proporcionar una explicación detallada y guiada de la plataforma, la cual facilita la comprensión de su complejidad y completitud a través del estudio del proyecto, y presentar un prototipo funcional que corrobora el objetivo de accesibilidad que busca la plataforma. Este enfoque ha permitido que el *firmware* sea elegido como caso de estudio y que el sistema resultante sea aplicable a múltiples casos de uso.

8. Impacto social, ambiental y Agenda 2030

La sostenibilidad se ha convertido en un pilar fundamental que se ha de tener en cuenta a la hora de desarrollar nuevas tecnologías. En la actualidad, es crucial que los proyectos no solo busquen la eficiencia y la productividad, sino que también se consideren sus repercusiones en el entorno social y ambiental.

Este proyecto representa un avance significativo hacia un desarrollo tecnológico más sostenible y, sobre todo, accesible, orientado sus beneficios directamente a toda la comunidad. Por un lado, el proyecto busca acercar el conocimiento a una tecnología accesible, y de bajo consumo, pudiendo desarrollar dispositivos tecnológicos mucho más accesibles en su uso, potenciando así los beneficios de los mismos y su accesibilidad. Socialmente, se asegura que el documento sea entendido y usado por toda la comunidad científica, augurando la proliferación del uso de tecnologías similares y abriendo un abanico de posibilidades que mejorarán el uso, y por ende, la vida, de las tecnologías y las personas.

El proyecto busca colaborar junto a las líneas de actuación de los **Objetivos de Desarrollo Sostenible (ODS)** [85] de la Agenda 2030. Los ODS fueron aprobados en 2015 por los Estados miembros de las Naciones Unidas con la finalidad de alcanzar para 2030, proteger el planeta y sus recursos naturales, entre otros objetivos.

Figura 111. Resumen ODS

Dentro de los ODS se encuentran 17 objetivos orientados a poner fin a la pobreza, proteger el planeta y mejorar las vidas y las perspectivas de las personas en todo el mundo. Cada objetivo tiene unas metas específicas que buscan alcanzarse en los próximos 6 años. Estos propósitos se recogen en la Agenda 2030 y con ellos se busca la colaboración de toda la comunidad, desde las grandes entidades hasta cada individuo en particular.

8.1. ODS

El proyecto se alinea directamente con las ODS 9 "Industria, innovación e infraestructura" y el ODS 10 "Reducción de las desigualdades" e indirectamente con otras como ODS 4 "Educación de calidad" y ODS 11 "Ciudades y comunidades sostenibles", haciendo mención especial a ODS 12 "Producción y consumo responsables".

- ***ODS 9, Industria, innovación e infraestructura*** [86]. Este objetivo tiene como fin construir infraestructuras inteligentes, promover la industrialización inclusiva y sostenible, y fomentar la innovación.

- ***ODS 10, Reducción de las desigualdades*** [87]. Este objetivo busca garantizar la equidad en el acceso a oportunidades y recursos, reduciendo brechas económicas, sociales y tecnológicas. Promueve la inclusión digital, el acceso a soluciones asequibles y el desarrollo equitativo a nivel global.

Gracias al desarrollo del proyecto, se busca facilitar el acceso a tecnologías innovadoras, mejorando la infraestructura en diversos sectores clave para la sociedad. El principal objetivo es generar un impacto positivo en la vida de las personas, promoviendo el uso de soluciones tecnológicas que no eran accesibles.

Además, se prioriza la accesibilidad, no sólo en términos de facilidad de acceso a los recursos, sino también en la usabilidad de uso de estas tecnologías. Se busca que puedan ser utilizadas por personas con diversas discapacidades, como dificultades visuales, como la pérdida parcial o total de visión, discapacidad auditiva, entre otras. La implementación de un *firmware* capaz de gestionar dispositivos de forma visual contribuirá a mejorar la calidad de vida de estos usuarios y de su entorno.

Por otra parte, el intento de mejorar la inclusividad en la tecnología impulsa la igualdad digital, reduciendo la brecha digital, permitiendo que más personas puedan beneficiarse de los avances tecnológicos sin barreras. Esto no solo favorece la integración de colectivos vulnerables, sino que también fomenta la innovación en el diseño de dispositivos más intuitivos, adaptables y eficientes para todos los usuarios.

- ***ODS 4, Educación de calidad*** [88]. Este objetivo busca garantizar una educación inclusiva, equitativa y de calidad, promoviendo oportunidades de aprendizaje para todos. Se enfoca en mejorar el acceso a la educación, reducir las brechas de desigualdad en el aprendizaje.

- ***ODS 11, Ciudades y comunidades sostenibles*** [89]. Este objetivo pretende lograr que las ciudades y comunidades sean más sostenibles. Promueve la planificación urbana responsable, la reducción de la contaminación y el desarrollo de infraestructuras que minimicen el impacto ambiental, mejorando así la calidad de vida de sus habitantes.

El desarrollo de un documento sobre una plataforma de código abierto y con el objetivo de mejorar la accesibilidad a sistemas tecnológicos complejos facilita el aprendizaje y la formación en entornos educativos. Al proporcionar una guía de la plataforma a desarrolladores experimentados en este tipo de productos se permite, en un futuro, que estudiantes puedan explorar nuevas soluciones tecnológicas que hasta entonces no eran conocidas por su complejidad en algunos niveles educativos.

La implementación de soluciones tecnológicas eficientes contribuye a la creación de entornos más sostenibles e inteligentes. Mediante el uso de hardware de bajo consumo, se reduce el gasto energético. Además, la integración de interfaces accesibles permite mejorar el uso de plataformas de automatización, las cuales están preparadas para un uso inteligente de los recursos. Es el caso de equipamiento domótico o tecnología para el día a día de las personas.

- ***ODS 12,*** **Producción y consumo responsables** [90]. Este objetivo busca garantizar modalidades de consumo y producción sostenibles, promoviendo el uso eficiente de los recursos y la reducción del impacto ambiental. Fomenta la

reutilización, el reciclaje y la optimización de los procesos productivos para minimizar los residuos y la contaminación.

El aprovechamiento de dispositivos obsoletos, o cerca, a través de su modernización es un claro ejemplo de consumo circular, alineado con este objetivo. Al actualizar dispositivos con estas soluciones, se pretende extender su vida útil y reducir los desechos electrónicos. Esto no solo reduce el impacto ambiental, sino que también permite acceder a tecnología moderna sin necesidad de adquirir nuevos dispositivos, fomentando un modelo de desarrollo más sostenible y responsable.

En definitiva, este documento no solo busca optimizar el uso de la tecnología, sino también contribuir a un futuro más sostenible, donde los recursos sean gestionados de manera eficiente y accesibles para todos. Al abordar de manera directa los **ODS 6 y 12**, y generar un impacto positivo en los **ODS 2, 9, 13 y 15**, se traza un camino hacia un modelo en el que la tecnología y la sostenibilidad se complementan para mejorar la calidad de vida y reducir la huella ambiental.

Este proyecto se compromete con los **Objetivos de Desarrollo Sostenible** y la **Agenda 2030**. A través de la innovación tecnológica responsable y el aprovechamiento eficiente de los recursos tecnológicos, se impulsa un cambio hacia un mundo donde el progreso tecnológico esté en armonía con el bienestar ambiental y social.

8.2. Accesibilidad en Pillbot y proyectos openHASP

La accesibilidad de un sistema se entiende como la capacidad de ser utilizado, y comprendido, por el mayor número posible de personas, incluidas aquellas con discapacidades físicas, sensoriales o cognitivas. En el contexto de *Pillbot*, el sistema está orientado a un usuario objetivo que se encuentra en situaciones médicas complejas, normalmente asociado a una etapa de edad avanzada, especialmente debido al propósito principal de la plataforma: agilizar y asegurar la correcta toma de todos sus medicamentos.

Para asegurar que la interfaz del sistema sea inclusiva, se han seguido los principios establecidos por las **WCAG** (*Web Content Accessibility Guidelines*, Pautas de Accesibilidad para el Contenido Web), impulsadas por la **WAI** [91] (*Web Accessibility Initiative,* Iniciativa Sostenibilidad Web), que forma parte del **W3C (***World Wide Web Consortium,* Consorcio

World Wide Web*)* [92]. Estas pautas, vigentes desde enero de 1995 con su primera versión WCAG 1.0, y actualizadas por última vez en la WCAG 2.1, en junio de 2018, proponen que los sistemas deben cumplir con cuatro principios fundamentales [93].

Estas pautas se corresponden a la WCAG 2.0, publicada el 11 de diciembre de 2008:

1. **Perceptible**: La información debe presentarse de forma que pueda ser percibida por todos los usuarios. Esto implica proporcionar alternativas a un mismo modo de comunicación, como el texto, en otros esquemas o imágenes visuales. Esto podría ser el uso de barras de progreso, que apoyan también el dato numérico en un valor.
2. **Operable**: La interfaz debe permitir que todos los usuarios puedan interactuar con ella de manera sencilla. Para ello, se incorporan elementos como botones de gran tamaño, menús simplificados y una navegación clara e intuitiva. Además, hay que evitar el contenido basado en tiempos cortos, dando al usuario margen disponible para la lectura de contenido.
3. **Comprensible**: Los contenidos deben ser fáciles de entender. La información debe presentarse de forma clara, mediante textos comprensibles, utilizando una tipografía legible y un contraste adecuado; es decir, empleando colores de texto que se distingan claramente del fondo para evitar que se difuminen o dificulten la lectura.
4. **Robusto**: El sistema debe ser compatible con una amplia variedad de tecnologías, tanto actuales como futuras.

Figura 112. Logo WCAG 2.2

Aunque esta normativa está orientada originalmente al entorno web, también puede aplicarse a la interfaz gráfica de proyectos generados con el *firmware* openHASP. Seguir estas pautas no solo mejora la accesibilidad para personas con discapacidades, sino que también beneficia a todos los usuarios, al hacer el sistema más intuitivo y fácil de utilizar en una amplia variedad de contextos y condiciones.

9. Conclusión

En conclusión, este Proyecto Fin de Grado materializa uno de los pilares fundamentales en el proceso educativo de cualquier profesional: la docencia e investigación.

Con el desarrollo de este trabajo espero demostrar que la investigación en un área con un gran potencial por explorar, como este *firmware* y similares, puede beneficiar a toda la comunidad de usuarios. Este documento perdurará en el tiempo y servirá de apoyo para que otros proyectos alcancen sus metas. Sobre todo, espero que esta tecnología llegue a las personas con el mismo propósito con el que nació este proyecto: ayudar.

A lo largo de su desarrollo, se han aplicado conocimientos técnicos adquiridos durante la carrera, como la Administración de Sistemas Operativos, en el despliegue y gestión del servidor, y la Administración y Gestión de Redes, en la instalación y puesta en marcha de un servicio MQTT similar a los estudiados en clase. Sin embargo, también existen inspiraciones en la labor de los docentes que han marcado mi etapa universitaria. Explicar y garantizar el aprendizaje del lector era un objetivo fundamental del proyecto y, a veces, lograrlo no ha sido fácil.

Más allá del aprendizaje técnico, este proyecto ha sido una oportunidad de crecimiento personal, enseñándome la importancia de la perseverancia, la autocrítica y la capacidad de adaptación frente a los retos que surgen en cualquier proceso de investigación y desarrollo. Confío en que esta experiencia sirva también como inspiración para futuros desarrolladores que decidan aventurarse en proyectos ambiciosos, motivados por el deseo de generar un impacto real y positivo en la sociedad.

Creo que el documento revisa todas las principales vías de investigación de la plataforma, dando una visión completa y amplia para un punto de partida autónomo, sentando la bases teóricas, y puesta en marcha técnica, para la creación de plataformas similares. Espero sentar precedente en la investigación de más plataformas interesantes como openHASP, y que estas sirvan para la mejora de los dispositivos desarrollados, por el bien común.

10. Líneas futuras

A lo largo de la memoria del Proyecto se ha explorado en profundidad las distintas vías de investigación que abre la plataforma. No obstante, el campo de estudio es dinámico, y más en un entorno de desarrollo tecnológico, que como ya bien se sabe, sigue una constante evolución y tendencias, cada vez más inexploradas, lo que abre la puerta a un sinfín de mejoras o linea futuras en las que los conocimientos explicados en el proyecto puedan incorporarse. En este apartado, se presentan algunas líneas futuras que serían enriquecedoras en el proyecto.

10.1. Incrustación a sistemas embebidos reales

Primeramente, como línea de mejora, el ejemplo de desarrollo en el proyecto actualmente no incluye un microcontrolador real, si no, se utilizaron otros métodos, también válidos, como un bot y servidor. Conectarse a un microcontrolador, a ser posible, ya existente, refuerza el objetivo que persigue el documento de mejorar dispositivos casi o prácticamente obsoletos, pudiendo acercar, o revivir, esta tecnología a mucha más gente.

Un ejemplo de esto podría ser mi Proyecto Final de Grado, en Ingeniería de Computadores, *Sistema integrado y distribuido para el análisis de agua en cultivos hidropónicos – CROP.AI* [94], el cual se tiene acceso al código fuente para poder mejorar la comunicación vía MQTT y habilitar su control desde una pantalla con *firmware* openHASP. Al igual que este proyecto, se podían hacer mejora de un sin fin de ellos, sobre todo en la Escuela, de los que puedan o quieran mejorar sus productos y se tenga acceso a la modificación de código fuente.

10.2. Librería para programación en ESP-IDF

La línea de desarrollo de la mayoría de los proyectos actuales se basa en el entorno de desarrollo nativo de los microcontroladores ESP32, ESP-IDF. Sería interesante ayudar a la comunidad a utilizar el *firmware* de openHASP, no solo ofreciendo este documento explicativo sobre la programación del *firmware*, sino también proporcionando una librería o un archivo fuente .h que encapsule algunas de las funciones principales de conexión con la plataforma.

Este módulo actuará principalmente a través de MQTT, facilitando al desarrollador la implementación de funcionalidades mediante una llamada, por ejemplo, pasando únicamente una función a modo de *callback* y un *topic*. Esto podría aplicarse tanto para el envío como para la recepción de datos: ya sea para activar una opción en openHASP o para ejecutar una acción en el entorno embebido.

Un ejemplo de esta librería sería por ejemplo

```
typedef void (*CallbackFunction)(const char* mensaje);
void escucharTopic(const char* topic, CallbackFunction function);
void publicarMensaje(const char* topic, const char* mensaje);
```

y su llamada algo como:

```
escucharTopic("mi/topic", miFuncionCallback);
publicarMensaje("mi/topic", "ON");
```

Con estas facilidades muchos desarrolladores de dispositivos tomarían la iniciativa de actualizar o crear nuevos proyectos basados en esta tecnología.

10.3. Análisis de código fuente openHASP

Al ser openHASP un proyecto *open source*, su código está disponible para su estudio exhaustivo [55], así como para su modificación con fines personales. Este código, escrito en su mayoría en C++, puede descargarse y ajustarse en ciertos aspectos para, por ejemplo, desarrollar una alternativa a openHASP adaptada a necesidades específicas.

Esta línea de investigación sería el siguiente paso tras una guía, como la de este proyecto, sobre su uso. Además, complementará el proceso formativo iniciado con este proyecto.

11. Glosario de acrónimos

ADC: *Conversor Analógico a Digital (Analog-to-Digital Converter).*

API: *Interfaz de Programación de Aplicaciones (Application Programming Interface).*

CAN: *Red de Área de Controladores (Controller Area Network).*

CS: *Selección de Chip (Chip Select).*

DAC: *Conversor Digital a Analógico (Digital-to-Analog Converter).*

ESP-IDF: *Framework de Desarrollo de IoT para ESP32 (Espressif IoT Development Framework).*

FTP: *Protocolo de Transferencia de Archivos (File Transfer Protocol).*

GND: *Tierra (Ground).*

GPIO: *Entrada/Salida de Propósito General (General Purpose Input/Output).*

GUI: *Interfaz Gráfica de Usuario (Graphical User Interface).*

HTML: *Lenguaje de Marcado de Hipertexto (HyperText Markup Language).*

HTTP: *Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol).*

I2C: *Circuito Inter-integrado (Inter-Integrated Circuit).*

I2S: *Interfaz de Sonido Integrado (Integrated Interchip Sound).*

IP: *Protocolo de Internet (Internet Protocol).*

IT: *Tecnología de la Información (Information Technology).*

JSON: *Notación de Objetos de JavaScript (JavaScript Object Notation).*

JSONL: *Notación de Objetos de JavaScript en Línea (JSON Lines).*

LCD: *Pantalla de Cristal Líquido (Liquid Crystal Display).*

LED: *Diodo Emisor de Luz (Light Emitting Diode).*

LVGL: *Biblioteca Gráfica de Bajo Nivel (Light and Versatile Graphics Library).*

MISO: *Entrada Serial del Maestro (Master In Slave Out).*

MOSI: *Salida Serial del Maestro (Master Out Slave In).*

MQTT: *Protocolo de Telemetría de Cola de Mensajes (Message Queuing Telemetry Transport).*

NTP: *Protocolo de Tiempo en Red (Network Time Protocol).*

OLED: *Diodo Orgánico Emisor de Luz (Organic Light Emitting Diode).*

ODS: *Objetivos de Desarrollo Sostenible.*

ONTSI: *Observatorio Nacional de las Telecomunicaciones y la Sociedad de la Información.*

OTA: *Actualización por el Aire (Over The Air).*

PSRAM: *Memoria Pseudo-Estática de Acceso Aleatorio (Pseudo Static Random Access Memory).*

PWM: *Modulación por Ancho de Pulso (Pulse Width Modulation).*

QR: *Código de Respuesta Rápida (Quick Response Code).*

RGB: *Rojo, Verde, Azul (Red, Green, Blue).*

RTOS: *Sistema Operativo en Tiempo Real (Real-Time Operating System).*

RTC: *Reloj en Tiempo Real (Real-Time Clock).*

SCK: *Reloj Serial (Serial Clock).*

SPI: *Interfaz Periférica Serial (Serial Peripheral Interface).*

SRAM: *Memoria de Acceso Aleatorio Estática (Static Random Access Memory).*

SSID: *Identificador de Conjunto de Servicios (Service Set Identifier).*

SSH: *Acceso Seguro a Sistemas Remotos (Secure Shell).*

TAM: *Tecnología de Apoyo a la Monitorización.*

PFG: *Proyecto Fin de Grado.*

TFT: *Pantalla de Transistor de Película Fina (Thin-Film Transistor).*

UART: *Transmisor/Receptor Asíncrono Universal (Universal Asynchronous Receiver-Transmitter).*

UI: *Interfaz de Usuario (User Interface).*

UPM: *Universidad Politécnica de Madrid.*

URL: *Localizador Uniforme de Recursos (Uniform Resource Locator).*

VCC: *Voltaje de Alimentación (Voltage Common Collector).*

V&V: *Verificación y Validación (Verification & Validation).*

VPS: *Servidor Privado Virtual (Virtual Private Server).*

W3C: *Consorcio World Wide Web (World Wide Web Consortium).*

WAI: *Iniciativa de Accesibilidad Web (Web Accessibility Initiative).*

WCAG: *Pautas de Accesibilidad para el Contenido Web (Web Content Accessibility Guidelines).*

WiFi: *Conectividad Inalámbrica (Wireless Fidelity).*

12. Bibliografía

[1] F. V. Roie, "Overview - openHASP," *Openhasp.com*, 2025. https://www.openhasp.com/0.7.0/ .

[2] IBM, "gemelo digital," *Ibm.com*, Aug. 05, 2021. https://www.ibm.com/es-es/think/topics/what-is-a-digital-twin

[3] "MQTT - The Standard for IoT Messaging," *mqtt.org*. https://mqtt.org

[4] "¿Qué es una pantalla TFT? Tecnología LCD de transistores de película fina | Orientación de la pantalla," *Orient Display*, Jun. 21, 2022. https://www.orientdisplay.com/es/knowledge-base/tft-basics/what-is-thin-film-transistor-tft/ .

[5] "ESP32-S3 Wi-Fi & Bluetooth 5 (LE) MCU | Espressif Systems," *www.espressif.com*. https://www.espressif.com/en/products/socs/esp32-s3

[6] "GPIO (General-Purpose Input/Output) - definición | Distribuidor de componentes electrónicos. Tienda en línea: Transfer Multisort Elektronik España," *TME*, 2025. https://www.tme.eu/es/news/library-articles/page/61888/gpio-general-purpose-inputoutput-definicion/ .

[7] "ESP32," *Wikipedia*, Aug. 05, 2022. https://es.wikipedia.org/wiki/ESP32

[8] "Acelera tus operaciones con IOT," *Oracle.com*, 2024. https://www.oracle.com/es/internet-of-things/

[9] "Hogares conectados: 8 de cada 10 españoles quiere invertir en domótica para su vivienda," *Uci.com*, 2023. https://uci.com/es/sala-de-comunicacion/nota-de-prensa/hogares-conectados-8-de-cada-10-espanoles-quiere-invertir-en-domotica-para-su-vivienda/

[10] D. Media, "Para 2025 más del 20% de las casas en España serán inteligentes," *Itreseller.es*, Apr. 15, 2024. https://www.itreseller.es/en-cifras/2024/04/para-2025-mas-del-20-de-las-casas-en-espana-seran-inteligentes

[11] "¿Qué es la Industria 4.0? | Deloitte España," *www.deloitte.com*. https://www.deloitte.com/es/es/Industries/industrial-construction/analysis/que-es-la-industria-4-0.html

[12]
"Uso de tecnologías digitales por empresas en España. 2022," Jan. 2022, doi: https://doi.org/10.30923/094-22-006-6.

[13] C. de, "Modelo de aceptación de tecnología," *Wikipedia.org*, Oct. 24, 2016. https://es.wikipedia.org/wiki/Modelo_de_aceptaci%C3%B3n_de_tecnolog%C3%ADa

[14] Colaboradores de los proyectos Wikimedia, "Expectativa de esfuerzo," *Wikipedia.org*, Apr. 06, 2010. https://es.wikipedia.org/wiki/Teor

[15] C. de, "Apple Lisa," *Wikipedia.org*, Nov. 11, 2005. https://es.wikipedia.org/wiki/Apple_Lisa

[16] RedHat, "What is open source?," *Redhat.com*, Oct. 24, 2019. https://www.redhat.com/en/topics/open-source/what-is-open-source

[17] "ESP-IDF Getting Started | Espressif Systems," *Espressif.com*, 2025. https://idf.espressif.com

[18] ARDUINO, "Arduino - Home," *Arduino.cc*, 2019. https://www.arduino.cc

[19] "Google Scholar," *scholar.google.es*. https://scholar.google.es

[20] "Archivo Digital UPM - Archivo Digital UPM," *oa.upm.es*. https://oa.upm.es

[21] L. A. Fernandez Arnanz, "Medium," *Medium*, 2025. https://medium.com/idean-spain/dise

[22] "LVGL — Light and Versatile Embedded Graphics Library," *Lvgl.io*, 2025. https://lvgl.io

[23] G. İŞNAS and N. ŞENYER, "Comparison of TouchGFX and LVGL Embedded Hardware GUI Libraries," *Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji*, vol. 9, no. 3, pp. 373–384, Sep. 2021, doi: https://doi.org/10.29109/gujsc.915163.

[24] "STM32 32-bit Arm Cortex MCUs - STMicroelectronics," *STMicroelectronics*. https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html

[25] R. P. Ltd, "Raspberry Pi," *Raspberry Pi*. https://www.raspberrypi.com

[26] H. Assistant, "Home Assistant," *Home Assistant*. https://www.home-assistant.io

[27] G. Diez, "Videotutoriales sobre el entrenador analógico/digital ETS-7000A | Archivo Digital UPM," *Oa.upm.es*, Oct. 2023, doi: https://oa.upm.es/76517/.

[28] M. Bradshaw, "Documentación, tutorial multimedia del simulador Multisim (módulos digitales) | Archivo Digital UPM," *Oa.upm.es*, 2018, doi: https://oa.upm.es/51897/.

[29] "¿Qué es una pantalla LCD? Tecnología LCD y tipos de pantalla | Orientar pantalla," *Orient Display*. https://www.orientdisplay.com/es/knowledge-base/lcd-basics/what-is-lcd-liquid-crystal-display/?fbclid=IwAR3MT8DqiG6ehTGLwZKNp11pLBifaIaHeEnNVtgEODLrEAZrYPgS8NeRTgA

[30] "¿Qué es una TV OLED? I LG ESPAÑA," *LG ES*, 2025. https://www.lg.com/es/soporte/guias-y-soluciones/television/que-es-tv-oled/?srsltid=AfmBOooNySRK95GlMM-1CITh_7TEPvK2mD6K1TtuV1EjaCao_tNdF1Zq

[31] "¿Qué es un led?," *Visual Led*, Aug. 02, 2018. https://visualled.com/glosario/que-es-un-led/

[32] "Serial Peripheral Interface," *Wikipedia*, Apr. 22, 2022. https://es.wikipedia.org/wiki/Serial_Peripheral_Interface

[33] C. de, "I2C," *Wikipedia.org*, Aug. 14, 2003. https://es.wikipedia.org/wiki/I

[34] Colaboradores de los proyectos Wikimedia, "Universal Serial Bus," *Wikipedia.org*, Jan. 04, 2004. https://es.wikipedia.org/wiki/Universal_Serial_Bus

[35] R. & S. International, "Qué es UART," *www.rohde-schwarz.com*. https://www.rohde-schwarz.com/es/productos/test-y-medida/essentials-test-equipment/digital-oscilloscopes/que-es-uart_254524.html

[36] "19 distribuciones de Linux populares," *Stackscale*, Jul. 20, 2021. https://www.stackscale.com/es/blog/distribuciones-linux-populares/

[37] "HTTP | MDN," *MDN Web Docs*, Mar. 27, 2025. https://developer.mozilla.org/es/docs/Web/HTTP#

[38] JSON.org, "JSON," *www.json.org*, 2023. https://www.json.org/json-en.html

[39] "Overview | Espressif Systems," *www.espressif.com*. https://www.espressif.com

[40] "ESP8266 Overview | Espressif Systems," *www.espressif.com*. https://www.espressif.com/en/products/socs/esp8266

[41] "¿Qué es el wifi? - Tipos de conexiones wifi y seguridad | Proofpoint ES," *Proofpoint*, Jun. 02, 2023. https://www.proofpoint.com/es/threat-reference/wifi

[42] "Protocolo MQTT para terminales de peso," *Systecnet.com*, 2025. https://www.systecnet.com/es/productos/software/mqtt.html

[43] Bluetooth, "Bluetooth Technology Overview," *Bluetooth® Technology Website*, 2024. https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[44] "Development Boards | Espressif Systems," *www.espressif.com*. https://www.espressif.com/en/products/devkits/esp32-devkitc/overview

[45] F. V. Roie, "Getting Started - openHASP," *Openhasp.com*, 2025. https://www.openhasp.com/0.7.0/hardware/

[46] "ESP32WROVERE & ESP32WROVERIE Datasheet." Available: https://www.espressif.com/sites/default/files/documentation/esp32-wrover-e_esp32-wrover-ie_datasheet_en.pdf

[47] "ESP32-PICO Data sheet," *Mouser.es*, 2025. https://www.mouser.es/ProductDetail/Espressif-Systems/ESP32-PICO-V3?qs=GBLSl2Akirs311o8knsnZA%3D%3D&srsltid=AfmBOooT4RROX3zhlwhmDAyA6SCgIG0GTwemLRKaGy7To5I8ge2hB_4Q

[48] "ESP32-C3 Wi-Fi & Bluetooth 5 (LE) MCU | Espressif Systems," *www.espressif.com*. https://www.espressif.com/en/products/socs/esp32-c3

[49] "m5-docs," *docs.m5stack.com*. https://docs.m5stack.com/en/core/core2

[50] F. V. Roie, "ESP32-S3 Parallel TFT - openHASP," *Openhasp.com*, 2025. https://www.openhasp.com/0.7.0/hardware/makerfabs/esp32-s3-parallel-tft-touch/

[51] F. V. Roie, "AZ-Touch MOD - openHASP," *Openhasp.com*, 2025. https://www.openhasp.com/0.7.0/hardware/az-delivery/az-touch/

[52] "T-Display S3," *LILYGO®*, 2022. https://lilygo.cc/products/t-display-s3?srsltid=AfmBOooQL26j8sIeC2m9HAGQXwBiGeBrlP71kGgBv5Q_m89ciAxJnBAs

[53] F. V. Roie, "Wireless-Tag WT32-SC01 - openHASP," *Openhasp.com*, 2023. https://www.openhasp.com/0.6.1/devices/wt32-sc01/

[54] Github, "GitHub," *GitHub*, 2013. https://github.com

[55] HASwitchPlate, "GitHub - HASwitchPlate/openHASP: HomeAutomation Switchplate based on lvgl for ESP32," *GitHub*, May 30, 2022. https://github.com/HASwitchPlate/openHASP .

[56] "KiCad EDA," *www.kicad.org*. https://www.kicad.org

[57] "Install openHASP," *Openhasp.com*, 2024. https://nightly.openhasp.com

[58] C. de, "Network Time Protocol," *Wikipedia.org*, Nov. 23, 2005. https://es.wikipedia.org/wiki/Network_Time_Protocol

[59] "¿Qué es un servidor privado virtual (VPS)? | Google Cloud," *Google Cloud*, 2025. https://cloud.google.com/learn/what-is-a-virtual-private-server?hl=es-419

[60] "Eclipse Mosquitto," *Eclipse Mosquitto*, Jan. 08, 2018. https://mosquitto.org

[61] MICROSOFT, "Visual Studio Code," *Visualstudio.com*, Apr. 14, 2016. https://code.visualstudio.com

[62] F. V. Roie, "Objects Cheatsheet - openHASP," *Openhasp.com*, 2025. https://www.openhasp.com/0.7.0/design/objects/#cheatsheet

[63] F. V. Roie, "Fonts - openHASP," *Openhasp.com*, 2022. https://www.openhasp.com/0.7.0/design/fonts/

[64] 1001 Fonts, "1001 Fonts," *1001 Fonts*. https://www.1001fonts.com

[65] "HTML Color Codes," *HTML Color Codes*, 2017. https://htmlcolorcodes.com

[66] Y. Fernández, "FTP: qué es y cómo funciona," *Xataka*, Jul. 15, 2021. https://www.xataka.com/basics/ftp-que-como-funciona

[67] "FileZilla - The free FTP solution," *filezilla-project.org*. https://filezilla-project.org

[68] "pool.ntp.org: the internet cluster of ntp servers," *Ntppool.org*, 2025. https://www.ntppool.org/es/

[69] "Python Release Python 3.12.0," *Python.org*. https://www.python.org/downloads/release/python-3120/

[70] "paho-mqtt," *PyPI*, Sep. 02, 2018. https://pypi.org/project/paho-mqtt/

[71] "Bash Reference Manual," *www.gnu.org*, Sep. 19, 2022. https://www.gnu.org/software/bash/manual/bash.html

[72] "python-telegram-bot," *Python-telegram-bot.org*, 2015. https://python-telegram-bot.org

[73] "schedule — schedule 0.4.0 documentation," *schedule.readthedocs.io*. https://schedule.readthedocs.io/en/stable/

[74] K. Reitz, "requests: Python HTTP for Humans.," *PyPI*, May 22, 2023. https://pypi.org/project/requests/

[75] ivangarsalgado, "GitHub - ivangarsalgado/pillbot," *GitHub*, 2025. https://github.com/ivangarsalgado/pillbot

[76] WhatsApp, "WhatsApp," *WhatsApp.com*, 2019. https://www.whatsapp.com

[77] "OVHcloud España: Cloud Computing y Web Hosting," *www.ovhcloud.com*. https://www.ovhcloud.com/es-es/

[78] "Bots: An introduction for developers," *core.telegram.org*. https://core.telegram.org/bots

[79] "Cursor," *Cursor.com*, 2024. https://www.cursor.com

[80] "StarUML," *Staruml.io*, 2014. https://staruml.io

[81] "MKLabs," *MKLabs*, 2016. https://mklaabs.com

[82] Draw.io, "Flowchart Maker & Online Diagram Software," *app.diagrams.net*, 2024. https://app.diagrams.net

[83] "Adobe Photoshop | Líder en edición y diseño fotográfico," *Adobe.com*, 2025. https://www.adobe.com/es/products/photoshop/landpa.html?mv=search&s_kwcid=AL

[84] Adobe, "Adobe: Creative, marketing and document management solutions," *Adobe: Creative, marketing and document management solutions*, 2019. https://www.adobe.com

[85] ONU, "Objetivos y Metas de Desarrollo Sostenible," *Naciones Unidas*, 2015. https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

[86] Naciones Unidas, "Infraestructura - Desarrollo Sostenible," *Desarrollo Sostenible*, 2015. https://www.un.org/sustainabledevelopment/es/infrastructure/

[87] "Objetivo 10," *Estrategia 2030, Agencia Multidisciplinar*. https://estrategia2030.es/objetivo-10-reduccion-de-las-desigualdades/

[88] Naciones Unidas, "Educación - Desarrollo Sostenible," *Desarrollo Sostenible*, 2015. https://www.un.org/sustainabledevelopment/es/education/

[89] M. Moran, "Ciudades," *Desarrollo Sostenible*, 2020. https://www.un.org/sustainabledevelopment/es/cities/

[90] Naciones Unidas, "Consumo y producción sostenibles - Desarrollo Sostenible," *Desarrollo Sostenible*, 2015. https://www.un.org/sustainabledevelopment/es/sustainable-consumption-production/

[91] W3C, "Home," *Web Accessibility Initiative (WAI)*, 2019. https://www.w3.org/WAI/

[92] W3C, "World Wide Web Consortium (W3C)," *W3.org*, 2019. https://www.w3.org

[93] "Web Content Accessibility Guidelines," *Wikipedia*, Oct. 13, 2021. https://es.wikipedia.org/wiki/Web_Content_Accessibility_Guidelines

[94] I. García Salgado, "Sistema integrado y distribuido para el análisis de agua en cultivos hidropónicos – CROP.AI | Archivo Digital UPM," *Oa.upm.es*, Oct. 2024, doi: https://oa.upm.es/84844/

Anexos

Anexo I. Instalar Drivers microcontroladores ESP32 (o similares)

Normalmente los microcontroladores de la gama ESP32, del fabricante Espressif, incorporan el módulo puente USB-to-UART CP210x de Silicon Labs el cual traduce las instrucciones UART del ordenador a lenguaje entendible por el microcontrolador, permitiendo la comunicación, sobretodo, para el flasheo, entre PC y microcontrolador.

No tener disponibles en el dispositivo estos controladores pueden producir que, al conectar el microcontrolador a un puerto COM (USB) del ordenador personal, este no sea reconocido y produzca una notificación de error en el Sistema Operativo.

La instalación del controlador puede realizarse desde la web oficial del fabricante del módulo:

https://www.silabs.com/developer-tools/usb-to-uart-bridge-vcp-drivers?tab=downloads

Donde, como se puede ver en la siguiente figura, se deberá seleccionar el sistema operativo destino.

Figura 113. Portal descarga drivers CP210x

Anexo II. Instalar herramientas ESPtool

El fabricante *Espressif* ofrece una librería basada en Python y *open-source*, que permite vía línea de comandos interactuar con una amplia gama de microcontroladores compatibles.

Esta librería, normalmente, se usa para flashear dispositivos desde línea de comandos, en los ejemplos como el que se explica en este documento.

Para instalarla, solo será necesario cumplir el requisito de disponer de Python en el ordenador personal, donde se debe ejecutar el siguiente comando en la terminal:

```
pip install esptool
```

Una vez el proceso de instalación se haya completado, se puede comprobar su funcionamiento con:

```
esptool.py -p <PORT> flash-id
```

Comando el cual nos devolverá la información, en caso de que lo reconozca, del microcontrolador conectado al puerto COM <PORT>.

Anexo III. Adquisición servidor VPS

Un VPS (Virtual Private Server, Servidor Virtual Privado) como dicen sus siglas en inglés, es un servidor privado con una IP fija la cual puede accederse desde cualquier parte del mundo con acceso a internet. Son numerosos los operadores que te ofrecen soluciones, de más o menos presupuesto, para su implementación.

Existen opciones con más memoria, con más o menos procesadores, con opciones de soporte en remoto, entre otros.

Para contratar uno de estos servidores, habrá simplemente que acceder a su web:

https://www.ovhcloud.com/es-es/

Una vez dentro, es necesario abrir una cuenta de forma totalmente gratuita y adjuntar un método de pago y, una vez rellenado esta información, se puede proceder a contratar el servicio de VPS y configurarlo tal y como se puede ver en la siguiente figura:

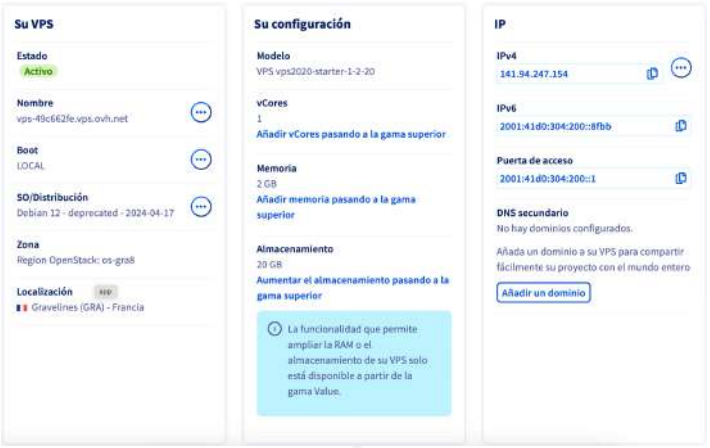


Figura 114. Panel de control VPS OVHCloud

Después de contratar el servicio, éste estará disponible en escasos 5 minutos, y desde entonces se dispone de una IPv4 pública de acceso SSH. Además, dispone de una contraseña de único uso que debe ser modificada en el primer acceso. Desde la terminal, es posible acceder como se muestra en la siguiente figura:

```
Last login: Mon Aug  5 09:18:39 on ttys323
ivangarcia@MacBook-Air-de-Ivan ~ % ssh root@141.94.247.154
root@141.94.247.154's password:
Permission denied, please try again.
root@141.94.247.154's password:
Permission denied, please try again.
root@141.94.247.154's password:
Linux vps-49c662fe 6.1.0-23-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.99-1 (
2024-07-15) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Aug  5 07:21:59 2024 from 85.52.230.99
root@vps-49c662fe:~#
```

Figura 115. Terminal acceso SSH VPS

Una vez dentro, se dispone de una máquina virtual con distribución Debian, con las características de la figura anteriormente mencionada, accesible desde cualquier parte del mundo, tanto por los desarrolladores, como por los microcontroladores conectados a internet.

Anexo IV. Configurar servidor syslog (rsyslog)

El protocolo Syslog permite centralizar los mensajes (*logs*) de distintos dispositivos en un único servidor, facilitando el seguimiento de errores y monitorización de red.

Para configurar un servidor Syslog, se pueden utilizar herramientas *open-source* como rsyslog, que suele venir preinstalada en muchas distribuciones de Linux, como en el servidor Debian VPS explicado en el Anexo anterior.

Para instalarlo basta con ejecutar el siguiente comando en la terminal:

```
sudo apt install rsyslog
```

Una vez instalado, se debe editar su archivo de configuración, generalmente ubicado en /etc/rsyslog.conf o en los archivos dentro del directorio /etc/rsyslog.d/, y cerciorarse de que las siguientes líneas estén habilitadas para permitir la recepción remota en el puerto especificado:

```
module(load="imudp") # para habilitar UDP
      input(type="imudp" port="514")
module(load="imtcp") # para habilitar TCP
      input(type="imtcp" port="514")
```

Tras realizar los cambios, será necesario reiniciar el servicio para que se apliquen correctamente:

```
sudo systemctl restart rsyslog
```

A partir de ese momento, el servidor Syslog está disponible para su configuración desde <ip_servidor>:514. Ahora, una vez que los sistemas comienzan a mandar sus *logs*, se puede ver el contenido de los mismos en el siguiente fichero:

```
/var/log/syslog
```

Anexo V. Crear bot Telegram (BotFather)

Para poder hacer uso de la API gratuita de Telegram es necesario disponer de un *Token*. Este "token", que es un código generado por Telegram, que permitirá acceder a la API desde un agente externo, por ejemplo, un bot. Importante no compartir con terceros y proteger su acceso

Para facilitar este proceso Telegram incorpora un bot el cual permite crear, gestionar, consultar o eliminar el estado de los bots de los cuales el usuario es propietario.

Este bot es accesible desde cualquier interfaz de Telegram, ya sea Web o App, a través de su cuenta @BotFather, de la siguiente forma como se puede ver en la figura:

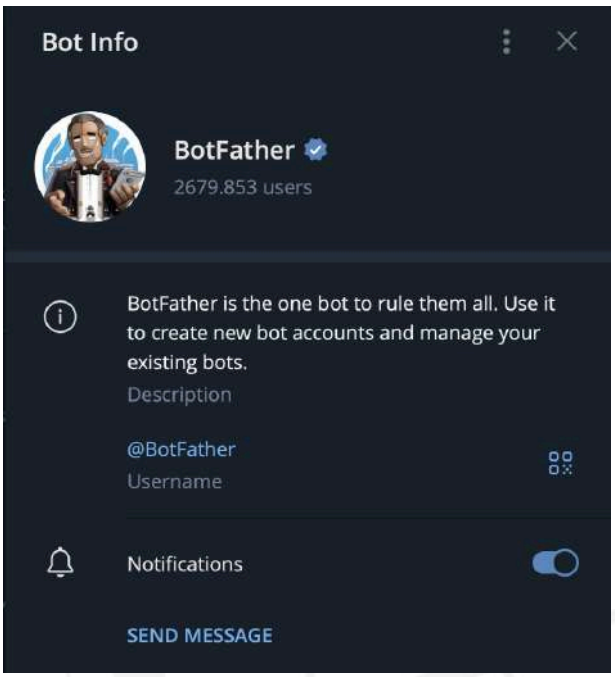


Figura 116. Perfil botfather Telegram

Dentro de su chat, un usuario tiene las siguientes opciones:

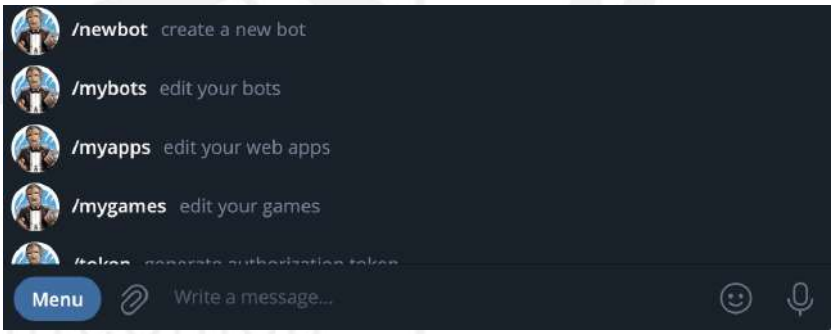


Figura 117. Comandos perfil botfather

Y seleccionando una de sus numerosas opciones, por ejemplo, /newbot, guiará en el proceso de creación de la siguiente forma:

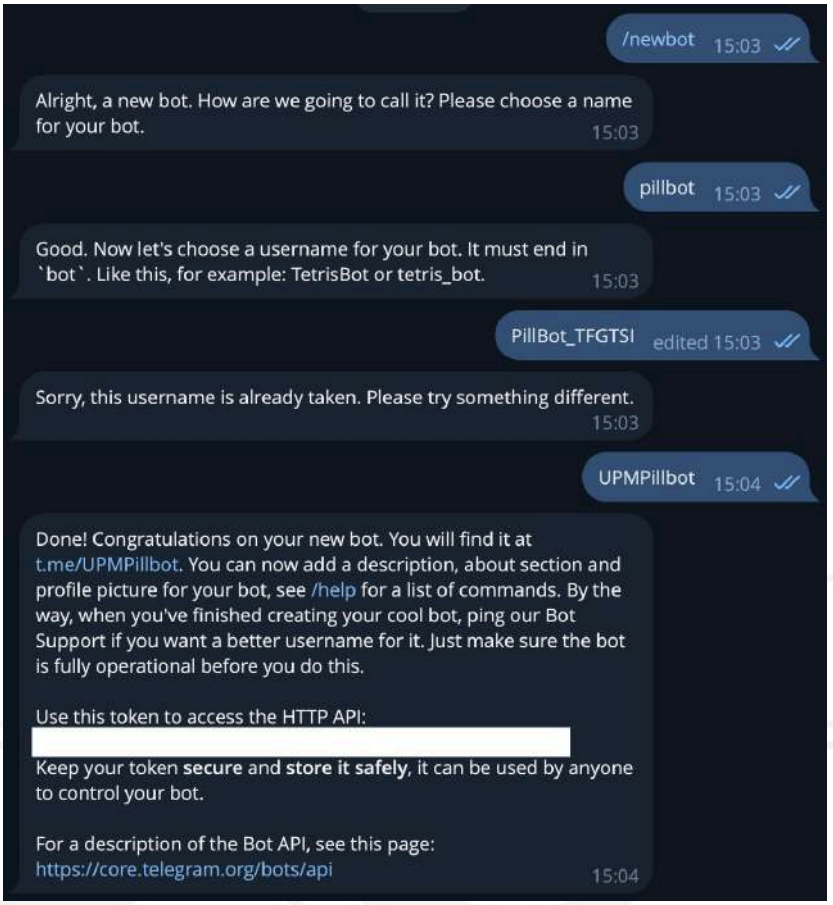


Figura 118. Creación bot

Donde después de pasar por un proceso de configuración en el cual habrá que seleccionar su nombre, un identificador, *BotFather* devolverá su *token* con el cual se podrá controlar el bot.

Después, es posible editar cierta información como, descripción, imagen, nombre, permisos… desde el comando /editbot, seleccionando del desplegable el bot objetivo.

Anexo VI. Tabla de código, openHASP

Enlace repositorio openHASP
https://github.com/ivangarsalgado/pillbot/blob/main/embedded/pages.jsonl

Funcion	Descripción	Enlace a línea
Header	Header página 0. Hora, nombre de marca e información meteorológica.	https://github.com/ivangarsalgado/pillbot/blob/b2f9e85b98735c58e0a0d77e0039858ccfc1b043/embedded/pages.jsonl#L3-L10
Control de páginas	Navegador de páginas que se sitúa en la parte inferior de la página 0	https://github.com/ivangarsalgado/pillbot/blob/b2f9e85b98735c58e0a0d77e0039858ccfc1b043/embedded/pages.jsonl#L12-L15
Página 1	Temporizador próxima toma, barra de progreso y botón de calendario	https://github.com/ivangarsalgado/pillbot/blob/b2f9e85b98735c58e0a0d77e0039858ccfc1b043/embedded/pages.jsonl#L17-L23
Página 2-8	Páginas correspondientes a días de la semana. Tabview para agrupar las pestañas con cada franja horaria. Label para cada nombre de pastilla y checkbox para confirmar su toma	https://github.com/ivangarsalgado/pillbot/blob/b2f9e85b98735c58e0a0d77e0039858ccfc1b043/embedded/pages.jsonl#L25-L142
Página 9	Visualización de perfil e información médica	https://github.com/ivangarsalgado/pillbot/blob/b2f9e85b98735c58e0a0d77e0039858ccfc1b043/embedded/pages.jsonl#L144-L180
Página 10	Código QR	https://github.com/ivangarsalgado/pillbot/blob/b2f9e85b98735c58e0a0d77e0039858ccfc1b043/embedded/pages.jsonl#L182-L183

Tabla 26. Tabla anexo código openHASP

Anexo VII. Tabla de código, servidor

Enlace repositorio servidor
https://github.com/ivangarsalgado/pillbot/blob/main/server/pillbot.py

Funcion	Descripción	Enlace a línea
imports	Librerías utilizadas.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L1-L19
Variables globales y estructuras	Valores y variables globales. Estructuras como traductores entre IDs y elementos textuales	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L21-L81
send_mqtt_json	Función que se encarga de la autenticación de cada mensaje en el servidor MQTT. Las funciones, en caso de querer mandar un mensaje, solo deberán invocar esta función.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L83-L85
run_scheduler	Función que se encarga de ejecutar las tareas pendientes. Por ejemplo, a las 00:00 la actualización meteorológica y cada segundo el cronómetro de tomas.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L87-L92
notificar_telegram	Función que notifica la toma de una pastilla pasado por parámetro.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L94-L97
on_connect	Callback para la conexión MQTT. Se suscribe a los topics.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L99-L102

Funcion	Descripción	Enlace a línea
on_message	Callback para la llegada de un mensaje MQTT de los topics suscritos. Según su contenido llama a una u otra función.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L104-L120
procesar_respuesta_pastilla	Función que procesa la respuesta a una notificación de una toma. Analiza su respuesta y toma acción según esta.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L122-L167
iniciar_listener_mqtt	Inicializador de las funciones MQTT.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L169-L175
send_mqtt_text_update	Función que actualiza el texto de un objeto pasado por parámetro	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L177-L180
send_mqtt_bar_update	Función que actualiza el valor de la barra de progreso de las tomas.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L182-L185
tarea_contador_toma	Tarea periódica ejecutada 1 segundo que actualiza y calcula el tiempo para las siguientes tomas. En caso de lanzar notificaciones, lo hará	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L188-L246
mostrar_info_diaria	Tarea periódica que realiza una petición a un servidor meteorológico y actualiza la información y controles según la fecha actual	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L248-L333
start	Callback para el mensaje /start	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L335-L342

Funcion	Descripción	Enlace a línea
mostrar_menu	Callback para el mensaje Menu. Muestra el menú.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L344-L354
campo_seleccionado	Callback para procesar los mensajes de edición de perfil	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L356-L385
crear_perfil	Callback para la función crear perfil, la cual se ejecuta en la primera ejecución.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L387-L409
guardar_valor	Función que guarda los valores introducidos por el usuario.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L411-L411
comenzar_agregar_pastilla	Callback para la funcion Añadir Pastilla	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L451-L455
recibir_pastilla_texto	Función que procesa el texto introducido por el usuario y lo almacena	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L457-L462
recibir_pastilla_horario	Función que procesa el horario de la pastilla y la almacena.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L464-L476
registrar_dia	Función que procesa el día de la pastilla y la almacena.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L478-L482

Funcion	Descripción	Enlace a línea
lanzar_notificacion_pastilla	Función encargada de la funcionalidad de lanzar una notificación para las pastillas organizadas.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L484-L492
recibir_pastilla_dia	Funcion que tras procesar toda la información del texto, horario y día, crea el objeto correspondiente en openHASP.	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L494-L576
main	Función main, inicio de los callbacks y la escucha en telegra,	https://github.com/ivangarsalgado/pillbot/blob/f4785bdd3c07ac3840c40f12fe18111a0c90c443/server/pillbot.py#L579-L617

Tabla 27. Tabla anexo código embebido

Anexo VIII. Vídeo demostración

Existe la posibilidad de ver una demostración del prototipo creado únicamente con openHASP, el enlace es el siguiente:

https://drive.google.com/file/d/1AEpEe6mGr9SJD3CfQwEpt6YXh7f450r9/view?usp=share_link