



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Diseño e implementación de un sistema de control y
monitorización remoto de bajo coste mediante Raspberry
Pi y gestión en la nube con Firebase

Trabajo Fin de Grado

Grado en Ingeniería en Tecnologías Industriales

AUTOR/A: Martínez Martínez, Darío

Tutor/a: Pérez Jiménez, Alberto José

CURSO ACADÉMICO: 2021/2022



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



ESCUELA TÉCNICA
SUPERIOR INGENIERÍA
INDUSTRIAL VALENCIA

Curso Académico:

A mi familia
En especial, a mi tío

Resumen:

El objetivo de este TFG es el diseño de un proyecto de instalación informatizada de bajo coste para controlar y monitorizar a distancia procesos industriales, concebido para pequeñas empresas. Podemos destacar tres apartados fundamentales.

En la instalación a controlar/monitorizar se empleará un ordenador SBC (Single Board Computer), concretamente una Raspberry Pi, que recibirá información de sensores y/o controlará acciones mediante actuadores ubicados en dicha instalación. El software con el que se realizará la gestión de los sensores y actuadores será Python junto con ciertas librerías específicas para gestionar cómodamente estos periféricos (`sense_hat`) así como las comunicaciones con la nube (`pyrebase`). Los sensores y actuadores serán emulados mediante el producto Sense HAT.

Además, una aplicación web se encargará de visualizar las variables medidas por los sensores acoplados a la Raspberry Pi y de controlar actuadores gestionados remotamente. Asimismo, se diseñará con soporte para varios usuarios potenciales y distintos perfiles, planteando la necesidad de un esquema de autenticación de usuarios. Los lenguajes a utilizar en el diseño de la aplicación web serán HTML5, CSS3 y JavaScript. La aplicación web estará alojada en un *hosting* gratuito proporcionado por la tercera parte del proyecto, Firebase, que se describe a continuación.

Una tercera entidad ubicada en la nube se encargará de gestionar la autenticación de los usuarios y transferir información entre la aplicación web y la Raspberry Pi en ambas direcciones. Se utilizará un producto comercial denominado Firebase de la firma Google, completamente gratuito bajo ciertas condiciones de tráfico de datos.

Esta instalación informática industrial de bajo coste y altamente accesible estará concebida como una introducción a la automatización y digitalización para empresas tradicionales que desean mejorar su eficiencia y capacidad de control de su producción.

Palabras clave: IoT, control remoto, monitorización remota, Raspberry PI, Firebase, automatización de bajo coste, PYME, web, Python.

Abstract

The objective of this TFG is the design of a low-cost computerized installation project to remotely control and monitor industrial processes, conceived for small companies. We can highlight three fundamental aspects.

In the installation to be controlled/monitored, an SBC (Single Board Computer) will be used, specifically a Raspberry Pi, which will receive information from sensors and/or control actions by means of actuators located in the installation. The software used to manage the sensors and actuators will be Python together with some specific libraries to comfortably manage these peripherals (`sense_hat`) as well as the communications with the cloud (`pyrebase`). The sensors and actuators will be emulated using the Sense HAT product.

In addition, a web application will be in charge of visualizing the variables measured by the sensors attached to the Raspberry Pi and of controlling remotely managed actuators, and it will be designed with support for multiple users and different profiles, it will rethink a user authentication scheme. The languages to be used in the design of the web application will be HTML5, CSS3 and JavaScript. The web application will be hosted on the free hosting provided by the third party of the project, Firebase, described below.

A third party entity located in the cloud will be responsible for managing user authentication and transferring information between the web application and the Raspberry Pi in both directions. A commercial product called Firebase from Google, which is completely free under certain data traffic conditions, will be used.

This low-cost and highly accessible industrial computing installation will be conceived as an introduction to automation and digitalization for traditional companies wishing to improve their efficiency and production control capacity.

Keywords: IoT, remote control, remote monitoring, Raspberry PI, Firebase, low-cost automation, SME, web, Python.

Resum

L'objectiu d'este TFG és el disseny d'un projecte d'instal·lació informatitzada de baix cost per a controlar i monitoritzar a distància processos industrials, concebut per a xicotetes empreses. Podem destacar tres apartats fonamentals.

En la instal·lació a controlar/monitorizar s'emprarà un ordinador SBC (Single Board Computer) , concretament una Raspberry Pi, que rebrà informació de sensors y/o controlarà accions per mitjà d'actuadors ubicats en la dita instal·lació. El programari amb què es realitzarà la gestió dels sensors i actuadors serà Python junt amb certes llibreries específiques per a gestionar còmodament estos perifèrics (sense_hat) així com les comunicacions amb el núvol (pyrebase) . Els sensors i actuadors seran emulats per mitjà del producte Sense HAT.

A més, una aplicació web s'encarregarà de visualitzar les variables mesures pels sensors acoblats a la Raspberry Pi i de controlar actuadors gestionats remotament. Asimismo, es dissenyarà amb suport per a diversos usuaris i distints perfils, replantejarà un esquema d'autenticació d'usuaris. Els llenguatges a utilitzar en el disseny de l'aplicació web seran HTML5, CSS3 i JavaScript. L'aplicació web estarà allotjada en l'hosting gratuït proporcionat per la tercera part del projecte, Firebase, que es descriu a continuació.

Una tercera entitat ubicada en el núvol s'encarregarà de gestionar l'autenticació dels usuaris i transferir informació entre l'aplicació web i la Raspberry Pi en ambdós direccions. S'utilitzarà un producte comercial denominat Firebase de la firma Google, completament gratuït davall certes condicions de tràfic de dades.

Esta instal·lació informàtica industrial de baix cost i altament accessible estarà concebuda com una introducció a l'automatització i digitalització per a empreses tradicionals que desitgen millorar la seua eficiència i capacitat de control de la seua producció.

Paraules clau: IoT, control remot, monitorització remota, Raspberry PI, Firebase, automatització de baix cost, PIME, web, Python.

ÍNDICE:

- I. Memoria
- II. Presupuesto
- III. Anexo

ÍNDICE DE LA MEMORIA:

1.	Introducción.....	9
1.1.	Objetivos.....	10
1.2.	Alcance del proyecto y ámbito de aplicación.....	11
1.3.	Introducción al software empleado.....	12
2.	Visión general del proyecto.....	13
2.1.	Descripción general del proyecto	14
2.2.	Selección de alternativas.....	16
2.3.	Antecedentes.....	17
2.4.	Motivación	18
2.5.	Justificación.....	18
2.6.	Acciones necesarias a priori.....	19
2.7.	Configurar la Raspberry pi.....	21
2.7.1.	Reseña histórica de la RPI.....	21
2.7.2.	Definición de SBC y funcionamiento básico.....	23
2.7.3.	Listado de sensores viables.....	25
2.7.4.	Interfaz de los sensores.....	29
2.7.5.	Librerías de los sensores.....	29
2.7.6.	Sistema Operativo y Software.....	30
2.7.7.	Arranque automático del Script.....	31
2.8.	Configurar la nube Firebase.....	32
2.8.1.	Configuración de la base de datos en tiempo real.....	33
2.8.2.	Autenticación en la base de datos.....	36
3.	Ejemplos prácticos y puesta en marcha.....	38
3.1.	Descripción del ejemplo que hemos propuesto.....	38
3.2.	Preparación del hardware: Raspberry pi y SenseHat.....	39
3.3.	Descripción de scripts de python empleados.....	40
3.4.	Descripción de la web: HTML, JS y CSS.....	46
3.5.	Demostración de funcionamiento.....	49
4.	Conclusiones.....	50
4.1.	Ampliaciones y mejoras del proyecto.....	50
5.	Bibliografía y referencias.....	51

Índice de tablas de la memoria:

Tabla 1: Versiones de Raspberry Pi.....	22
Tabla 2: Características del sensor de temperatura.....	26
Tabla 3: Características del sensor de presión.....	26
Tabla 4: Características del sensor de humedad.....	27
Tabla 5: Características del sensor de aceleración.....	27
Tabla 6: Características del sensor de luz solar.....	28
Tabla 7: Características del sensor de CO2.....	28

Índice de figuras de la memoria:

Figura 1: Esquema del proyecto.....	13
Figura 2: Esquema de la Zona controlada.....	14
Figura 3: Ejemplo de varias zonas controladas.....	15
Figura 4: Ejemplo de la raíz en Firebase con varias zonas controladas.....	16
Figura 5: Guardado de mediciones en la nube.....	20
Figura 6: Límites del uso gratuito de Firebase.....	20
Figura 7: Comunicación mediante redes móviles.....	22
Figura 8: Raspberry pi 4 Model B.....	22
Figura 9: Carcasa para Raspberry pi para ambientes industriales explosivos....	22
Figura 10: Componentes de la Raspberry pi.....	24
Figura 11: Puerto para tarjeta microSD.....	25
Figura 12: Sensor de temperatura.....	26
Figura 13: Sensor de Presión.....	26
Figura 14: Sensor de Humedad.....	27
Figura 15: Sensor de aceleración.....	27
Figura 16: Sensor de luz solar.....	28
Figura 17: Sensor de CO2.....	28
Figura 18: Raspberry pi imager.....	30
Figura 19: Inicio de Firebase.....	32
Figura 20: Ventana de proyectos en Firebase.....	33
Figura 21: Base de datos en tiempo real de Firebase.....	33
Figura 22: Estructura de la base de datos.....	34
Figura 23: Históricos en la base de datos.....	35
Figura 24: Las reglas de la base de datos.....	35
Figura 25: Ventana de autenticación ...	36
Figura 26: Especificar el esquema de autenticación	37
Figura 27: Montaje del Sense Hat.....	39
Figura 28: Vista general de la aplicación web.....	46

Figura 29: Detalle de la fila del sensor de temperatura.....	47
Figura 30: Actuadores.....	48
Figura 31: Barra de navegación.....	48
Figura 32: Modal “Entrar”.....	48
Figura 33: Créditos.....	49
Figura 34: Demostración de funcionamiento.....	49

ÍNDICE DEI PRESUPUESTO:

1. Cuadro de Mano de Obra.....	54
2. Cuadro de Materiales.....	54
3. Cuadro de Maquinaria.....	54
4. Cuadro de Unidades de Obra.....	55
5. Presupuesto Total.....	57

ÍNDICE DEI ANEXO:

1. FICHEROS HTML.....	59
2. FICHEROS CSS.....	68
3. FICHEROS JavaScript.....	71

Índice de figuras del anexo:

Figura 1: Contenedor div principal marcado en la aplicación web.....	61
Figura 2: Contenedor div del sensor de temperatura.....	63
Figura 3: Detalle del contenedor div que contiene el selector.....	64
Figura 4: Ejemplos de la propiedad flex-container.....	69

MEMORIA

1 INTRODUCCIÓN

La automatización y control digital es una de las revoluciones tecnológicas más importantes que ha experimentado la industria, tanto es así que tiene un papel imprescindible en la conocida cuarta revolución industrial [1]. Sus capacidades de mejora de la producción y mayor eficiencia ha llevado a la mayoría de empresas a adoptar esta rama de la Ingeniería.

En paralelo, las redes de comunicación, como la Internet y la Informática han revolucionado la forma en la que funciona el mundo. Dentro de este, destaca para el interés de este trabajo fin de grado los “Single Board Computers” [2]: Pequeños ordenadores con todos sus componentes imprescindibles soldados en una única placa. Estos equipos informáticos de bajo coste son extremadamente versátiles al ser capaces de realizar prácticamente cualquier tarea que un ordenador convencional pueda hacer, tanto a nivel de cálculo como de comunicaciones, con las lógicas limitaciones debido a sus relativamente modestas prestaciones.

Combinando la portabilidad y bajo coste de los SBC y las capacidades de comunicación y conexión de internet, obtenemos un producto con una gran funcionalidad, practicabilidad y accesibilidad y que usaremos para nuestro trabajo.

Para este trabajo fin de grado se demostrará que es posible implementar un sistema completo de control digital a distancia empleando software y hardware libre y/o gratuito. Esta instalación combinará un bajo coste y una gran flexibilidad pensado para ser adoptado por pequeñas industrias que no sean capaces de instalar un equipo de automatización comercial de grandes dimensiones y prestaciones.

La memoria del trabajo final de grado consta de las siguientes partes principales:

- Introducción:

Se presenta brevemente el proyecto, explicando los objetivos, mencionando el contexto económico y social en el que se desarrolla. Se introducen y explican de forma breve y concisa herramientas de software que se emplean. También se indica el alcance y ámbito de aplicación al que está pensado el trabajo final de grado.

- Vista general del proyecto:

Se describen las tres entidades en las que se basa el sistema de control. Para cada entidad se discuten las mejores alternativas disponibles y se escoge una. Se indican antecedentes de anteriores trabajos finales de grado, se indican las motivaciones académicas del trabajo y se explican las justificaciones del mismo.

También se introducen los requisitos previos necesarios para llevar a cabo el proyecto del sistema de control y monitorización. En el apartado de la Raspberry Pi, se incluye una breve reseña histórica y un detalle de las partes que la conforman, se incluye una lista de sensores compatibles y explicaciones de las librerías e interfaces y su conexión a internet. En el apartado de Configuración de la Nube, Firebase, se describen las características que emplearemos de este producto, así como su configuración para obtener toda la funcionalidad necesaria del mismo.

- Ejemplos prácticos y puesta en marcha:

Como demostración de funcionamiento del sistema al completo se ha montado un ejemplo práctico con sensores reales. Cada paso de la realización de esta prueba de concepto se encuentra explicada detalladamente.

- Conclusiones:

Comentarios del trabajo y del resultado del sistema final. Aportaciones de posibles mejoras para líneas futuras que continúen con el trabajo mostrado.

- Bibliografía y referencias:

Se aportan numeradas todas las referencias bibliográficas que han sido mencionadas a lo largo de la memoria.

1.1 Objetivos

El objetivo principal es el diseño e implementación de una prueba de concepto que permita el control y monitorización de un sistema a distancia, de bajo coste, orientado para pequeñas empresas. Para este sistema se emplearán en todo lo posible software libre y gratuito y componentes electrónicos económicos (como la Raspberry Pi, sensores fácilmente accesibles, etc.).

Para lograr el objetivo principal se deberá cumplir una ruta con los siguientes sub-objetivos: la configuración de la Raspberry Pi, elaborar los scripts en python necesarios para dotar de su comunicación, la configuración de la base de datos en la nube y la creación de una aplicación web básica como interfaz de monitorización y control para probar el resultado. Debido a la amplia posibilidad de configuración de sensores y actuadores que depende de cada caso de aplicación, vamos a emplear en el desarrollo de nuestra prueba de concepto un producto compatible con la SBC Raspberry Pi que incluye varios sensores y lo que podríamos considerar como actuadores (matriz de leds); el producto se denomina comercialmente Sensor Hat. La intención es que en una instalación en un ambiente real, el uso de otros sensores u otro tipo de actuadores no altere la validez de la idea.

Los objetivos mostrados de forma desglosada son:

- Configurar un sistema de entradas y salidas (sensores y actuadores) junto con un ordenador que muestree las entradas y actúe sobre las salidas..
- Configurar un servicio de almacenamiento en la nube.
- Crear una aplicación web fácil de usar para representar los datos recogidos y controlar los actuadores.
- Establecer la comunicación entre los tres agentes del proyecto para transmitir datos y órdenes.

1.2 Alcance del proyecto y ámbito de aplicación

Desde que la crisis energética afectó a los negocios y empresas, en la sociedad hay una mayor concienciación en el uso de la energía. Además, la preocupación de la sociedad sobre los futuros impactos del cambio climático también han llevado a una mayor atención a la forma en la que consumimos energía[3]. Sumando a esto, las consecuencias de vivir en un mundo cada vez más globalizado, exige a las empresas ser más competitivas y eficientes. Para lograr consumir únicamente lo mínimo imprescindible, no desaprovechar energía y mantenerse competitivas, las empresas se han volcado en un proceso de digitalización, automatización y control a distancia de muchos de sus procesos.

Hoy en día es muy común en las industrias la presencia de sistemas automáticos de control y/o monitorización de procesos. Los PLC (Programmable Logic Controller) o Controlador Lógico Programable es el instrumento más habitual para instalar los sistemas de automatización en las industrias[4]. Gracias al aumento de productividad que supone automatizar y monitorear procesos industriales, las industrias y grandes empresas pueden permitirse instalar sistemas automáticos comerciales de alto coste. Pero lo mismo no ocurre con pequeñas empresas (PYMEs) que no son capaces de costear la inversión inicial pese a las evidentes mejoras que supondría su instalación.

El sistema que se desarrolla en este trabajo fin de grado está pensado para satisfacer en cierto modo, una mínima demanda de digitalización, modernización y control de las pequeñas empresas y negocios familiares que componen la mayoría del tejido productivo en nuestro país[5], pero no cuentan con el suficiente capital inicial para adquirir un sistema de control/monitorización comercial. El sistema que proponemos debe ser lo suficientemente capaz como para realizar acciones de control y monitorización remota simples y complejas, mientras que al mismo tiempo mantiene un coste reducido.

Esta prueba de concepto también puede ser utilizada para empresas que no se atreven a introducirse a la digitalización o no lo creen necesario,

desincentivados por un alto coste inicial. Pueden probar a utilizar este sistema más económico como prueba piloto para familiarizarse con los conceptos y el funcionamiento para más adelante dar el salto a un sistema más profesional/comercial.

1.3 Introducción al software empleado

- HTML: “HyperText Markup Language” es el lenguaje informático estándar diseñado para crear páginas web y mostrarlas en un navegador de internet. El funcionamiento del lenguaje se basa en etiquetar y marcar piezas de código para que los navegadores web puedan interpretarlo.
- JavaScript: Es un lenguaje de programación orientado en objetos en sus últimas versiones, que se utiliza para otorgar funcionalidad a las páginas web. Es el encargado de que al pulsar en un botón o pestaña, estos tengan un comportamiento definido.
- CSS: “Cascading Style Sheets” es un lenguaje de hojas de estilo en cascada usado para proporcionar estilo a las páginas web desarrolladas con HTML.
- Python: Es un lenguaje de programación orientado a objetos , interpretado, dinámico y multiplataforma y se clasifica como uno de los lenguajes más populares y usados. Es un lenguaje de programación interpretado que destaca por su filosofía de tener una buena legibilidad, fácil interpretación y uso de sangrado para estructurar el control de flujo. Ha sido utilizado para el desarrollo de parte de aplicaciones como Netflix, Spotify o Instagram.
- SSH: SSH es el nombre de un protocolo y del programa utilizado en la terminal de linux que concede el uso de una máquina donde toda la transferencia de información es cifrada.
- Visual Studio Code: Es un editor de texto gratuito con un gran espectro de personalización y configuración. Dispone de diversas extensiones que pueden ampliar sus funcionalidades y facilitar su uso a la hora de escribir código o archivos. Está desarrollado para una gran cantidad de lenguajes así como Java, JavaScript, PHP, C++, Python entre otros. Los desarrolladores del editor son Microsoft.
- GIT/ GitHub: es un sistema de control de versiones con el que se puede acumular en el tiempo y en el espacio la evolución de un código de programación, por lo que se puede volver a versiones anteriores en el tiempo, plantear alternativas, fusionarlas y un largo etcétera de funcionalidades relacionadas con el diseño tanto personal como sobre todo en equipo de software. GitHub es la versión en la nube de parte de la funcionalidad de Git así como funcionalidades adicionales.

2 VISIÓN GENERAL DEL PROYECTO

Para lograr los objetivos que se proponen en este trabajo final de grado se deben cumplir los requisitos siguientes: el producto debe ser configurable para un amplio abanico de situaciones, debe ser sencillo de usar de cara a un usuario sin experiencia informática avanzada, debe funcionar de forma autónoma sin necesidad de supervisión humana, debe continuar funcionando aunque ningún usuario se encuentre conectado...

Teniendo en cuenta estas necesidades se llega a las siguientes conclusiones:

- 1) La zona controlada encargada de recolectar datos y controlar actuadores debe incluir un procesador configurado y conectado a sensores y actuadores y a una red de comunicaciones con el exterior.
- 2) Para que el usuario sea capaz de controlar y supervisar la instalación remotamente, debe desarrollarse una aplicación web compatible con cualquier dispositivo (ordenador, móvil, tablet, etc) mediante conexión a internet.
- 3) Para que el sistema esté sensando y almacenando información continuamente debe estar conectado a una nube donde almacenará la información en una base de datos. Desde la aplicación web se consultará esta base de datos.

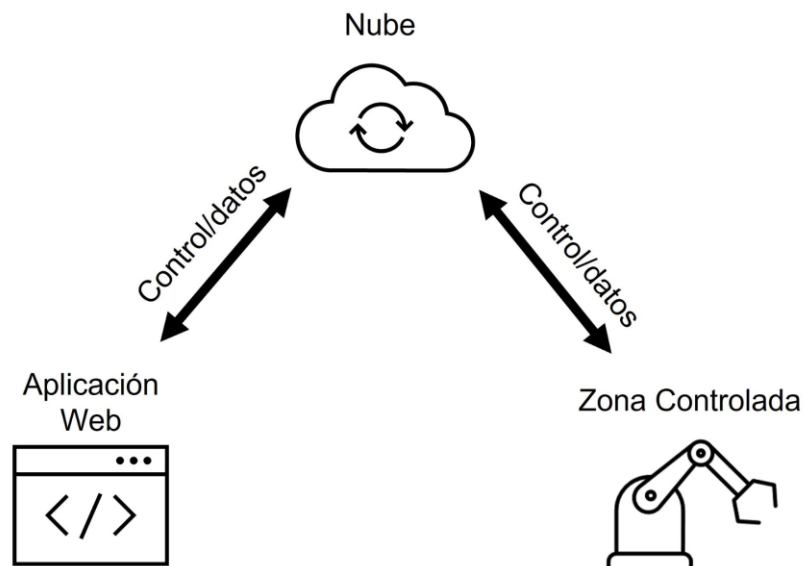


Figura 1: Esquema del proyecto

2.1 Descripción general del proyecto

El sistema consta de 3 partes básicas: la zona controlada, la aplicación web y la nube donde se almacena la información. A continuación se va a profundizar más en el funcionamiento, diseño y misión de cada una.

I. Zona controlada

La zona controlada y sus sensores/actuadores conforman la parte física del sistema. En esta zona se encuentran las entradas y las salidas a controlar. Estas pueden ser sensores de cualquier tipo y actuadores capaces de llegar a conmutar alimentación de cualquier máquina eléctrica. En el ordenador de la zona controlada se ejecutará un script (escrito en lenguaje Python) encargado de ordenar a los sensores tomar lecturas y determinar a los actuadores cuándo deben conectarse empleando las librerías propias de cada sensor y actuador. Se ha escogido este lenguaje programación por varios motivos: es un lenguaje relativamente fácil de aprender y además las librerías encargadas de llevar a cabo la comunicación con los sensores y con la nube son fácilmente conseguibles y utilizables en dicho lenguaje de programación, existiendo numerosas fuentes de información que ayudan a asimilar rápidamente los conceptos necesarios para desarrollar la parte relacionada con la programación en el SBC del proceso a controlar y/o monitorizar.

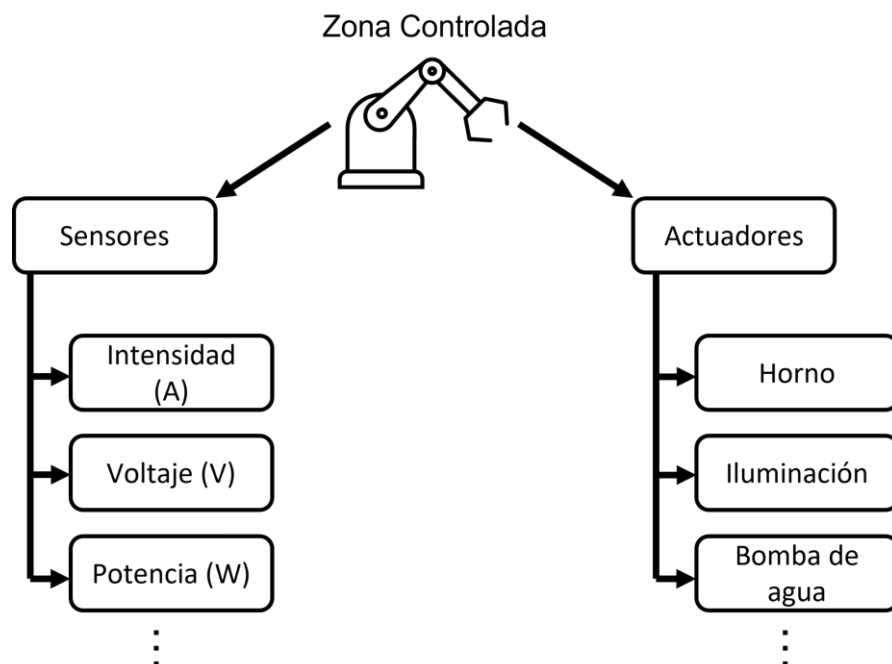


Figura 2: Esquema de la Zona controlada

II. Aplicación Web

Para crear la aplicación web se va a programar en el lenguaje HTML una web de estilo sencillo. Para dar estilo y personalización al contenido se usa CSS.

Junto con los archivos HTML se emplean scripts en JavaScript para añadir funcionalidades como la conexión con la nube, representar los valores de las lecturas de los sensores con diales o relojes, interruptores para los actuadores y la representación de gráficos temporales interactivos.

JavaScript también permitirá configurar un menú contextual con campos para introducir la identificación y uso de contraseña para disponer de una primera aproximación a la parte de autenticación y seguridad que requiere cualquier aplicación que de alguna forma pase a través de internet.

III. Nube

Una de las cualidades de usar este tipo de configuración para nuestro sistema es la facilidad con la que se puede implementar nuevas zonas controladas, sin límite de distancia, únicamente necesitando conexión a internet. Para lograr esto simplemente es necesario conectar diferentes SBC (Raspberries) con la nube y organizar en la base de datos distintos subdirectorios para cada una, por lo que la escalabilidad, a priori y dentro de un orden, parece perfectamente asumible.

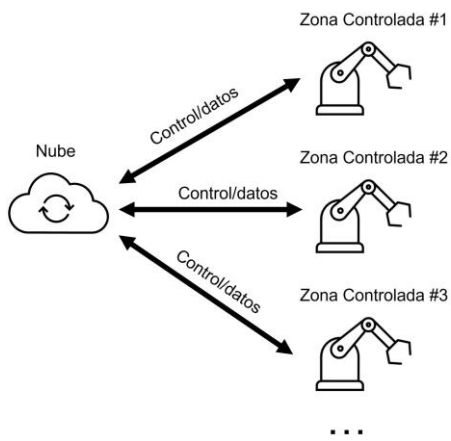


Figura 3: Ejemplo de varias zonas controladas



Figura 4: Ejemplo de la raíz en Firebase con varias zonas controladas

Además la nube también puede almacenar lecturas de los sensores o estados/posiciones de actuadores ordenados en el tiempo para luego representarlas gráficamente respecto al mismo.

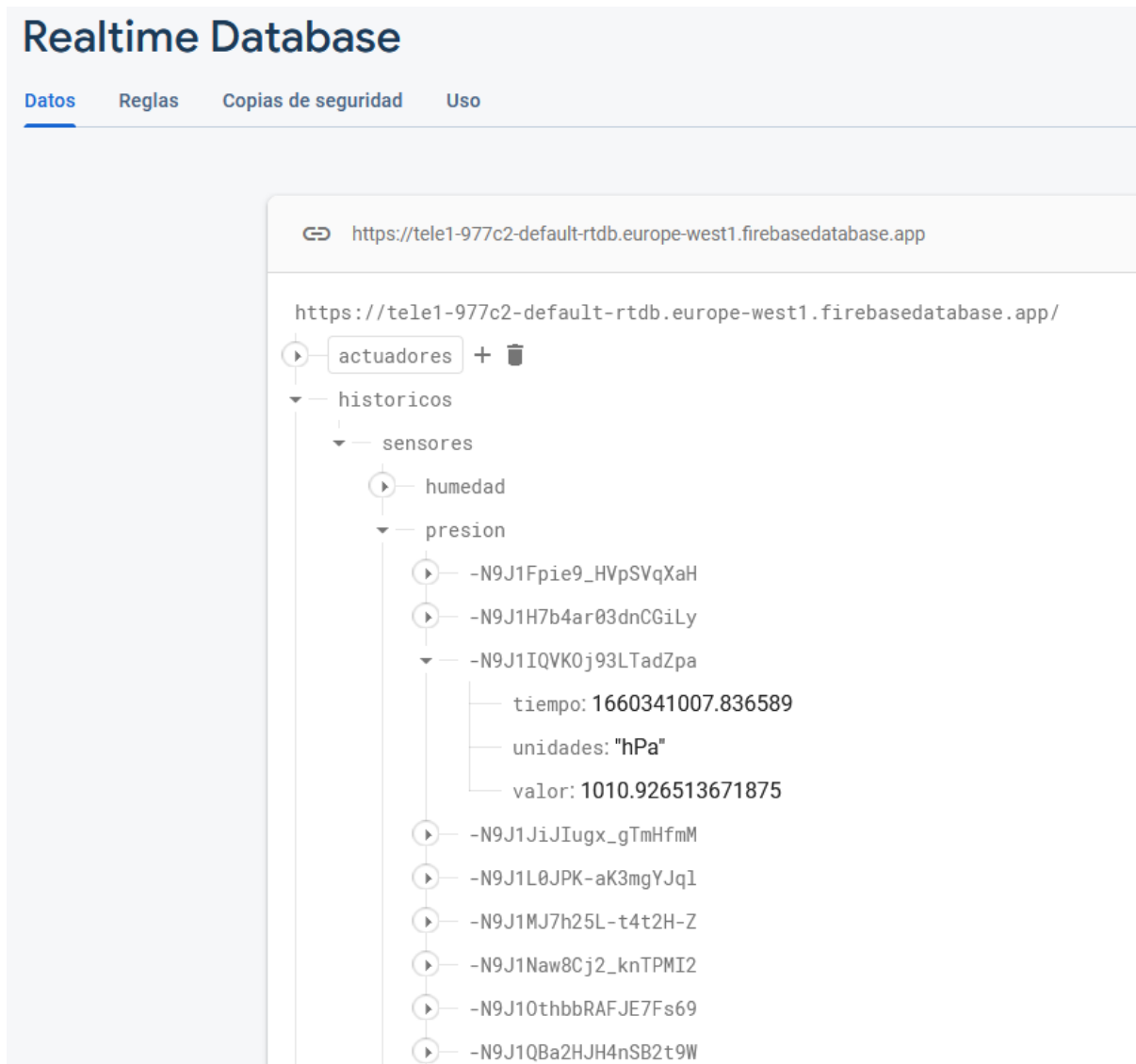


Figura 5: Guardado de mediciones en la nube

2.2 Selección de alternativas

Una vez ya se ha presentado la estructura general del sistema y descrito con detalle los componentes es necesario determinar qué productos y servicios online se adapta mejor a nuestras necesidades.

1) Procesador para la zona controlada

Para las necesidades de nuestro proyecto necesitamos un ordenador con unos requisitos de potencia de procesador relativamente bajos. En este caso la mejor opción es emplear un ordenador del tipo SBC, con potencia de procesador más que suficiente para nuestras necesidades y que además tiene un precio inferior a un ordenador de sobremesa o portátil.

Dos de los ordenadores SBC más famosos son la Raspberry Pi y el Arduino. Arduino es fundamentalmente un microcontrolador que ejecuta un único programa. No utiliza sistema operativo, por lo tanto no se puede hacer uso de aplicaciones y herramientas que sí se encuentran en Linux. [6]

Raspberry es un pequeño ordenador completamente funcional con un sistema operativo basado en Linux. Esto le permite realizar varias tareas en paralelo al mismo tiempo y tiene acceso a todos los programas y aplicaciones profesionales desarrolladas para el sistema Linux.

Considerando las amplias posibilidades que ofrece una raspberry frente a Arduino, merece la pena utilizarlo aunque su coste sea ligeramente superior.

2) Aplicación web

La aplicación que emplea el usuario para controlar y monitorizar la zona controlada se puede desarrollar de varias maneras. La aplicación puede basarse en una aplicación Android, descargable desde la Play Store de Google. Este método es habitual entre trabajos fin de grado por su facilidad de creación con las herramientas gratuitas de Google pero trae inconvenientes como por ejemplo la falta de compatibilidad con usuarios de iPhone u otros dispositivos Windows etc. Lo mismo ocurre si se emplean aplicaciones destinadas a las tiendas de aplicaciones de Apple o Microsoft.

En nuestro caso se opta por elaborar una aplicación web desde la cual se puede acceder con cualquier navegador en cualquier dispositivo sin distinguir SO o plataforma, de esta manera se garantiza la compatibilidad universal. El inconveniente es que puede ser ligeramente menos eficiente que las opciones anteriores.

3) Nube

Existen numerosas posibilidades para plantear el almacenamiento y control a través de la nube. El protocolo MQTT (Message Queue Telemetry Transport) íntimamente ligado con el paradigma IoT, puede ser utilizado en este contexto en el que tendríamos una serie de suscriptores abonados a ciertos datos retransmitidos por un dispositivo denominado broker. Hemos preferido utilizar la opción proporcionada por una plataforma distinta (Firebase) ya que permite una mayor flexibilidad y control sobre los procesos que se están monitorizando o controlando.

2.3 Antecedentes

Anteriormente se presentaron otros trabajos final de grado que pueden enmarcarse en temáticas parecidas al del presente TFG y que aportan funcionalidades interesantes.

Desarrollo de un Controlador PID Industrial de bajo coste mediante Raspberry Pi para control de temperatura: Ofrece la posibilidad de incorporar un controlador PID programado en una Raspberry Pi para automatizar el control de una variable. También crea una aplicación web donde configurar los parámetros deseados para el empleo del PID. [7]

Desarrollo de un sistema de localización y aplicación móvil para vehículos en aparcamientos: Desarrolla un sistema de geolocalización con una Raspberry Pi y módulos GPS con el que puede determinar la posición con una precisión de menos de 3 metros. [8]

Estos sistemas pueden fácilmente enlazarse a la nube Firebase de un modo similar al explicado en el presente trabajo final de grado y enviar los datos para ser visualizados y controlados desde nuestra aplicación web. Son adiciones muy interesantes que quedan a la voluntad del usuario final llevarlas a realizar o no, dependiendo de sus necesidades.

2.4 Motivación

Promover la posibilidad de mejora de control y digitalización de procesos a las PYME mejoraría notablemente su eficiencia y les ayudaría a hacer sus funciones de una forma más cómoda y práctica. El TFG, una vez implementado, lograría dar un mayor control a pequeños negocios sobre sus procesos.

Se va a diseñar el sistema para que la aplicabilidad del mismo sea para cualquier negocio. La gran variedad de sensores y actuadores disponibles para la Raspberry Pi hace que las posibilidades de implantación sean muy grandes y para todo tipo de negocio interesado. Ayudar a estas pequeñas empresas supone una motivación suficiente para completar este proyecto, además de la oportunidad que me brinda la Escuela Técnica Superior de Ingeniería Industrial para investigar sobre el internet de las cosas (IoT), la digitalización y la programación web.

2.5 Justificación

El sistema de control tiene implicaciones que puede volverlo inadecuado en determinadas situaciones. Al estar formado por varios servicios aumenta la posibilidad de que el sistema en conjunto falle. Los casos en los que la nube de Firebase pierde su conexión son muy infrecuentes pero bien documentados [9]. La conexión a internet de la zona controlada también puede verse interrumpida

por cuestiones externas al ser una conexión convencional que no dispone de métodos de comunicación alternativos o adicionales. Por estos motivos el sistema propuesto no se considera “seguro”, haciendo referencia a que no hay una defensa en profundidad para mantener la conexión y funcionamiento correcto del sistema. Por ello, las aplicaciones de las cuales sea estrictamente obligatorio que el sistema funcione pueden no ser adecuadas para este tipo de sistema, por ejemplo: actividades de las cuales la vida de una persona dependa del correcto funcionamiento del sistema o actuaciones que en caso de emergencia deben realizarse y no pueden permitirse fallar, como un sistema de aspersores anti-incendios. Todas estas situaciones tienen estipulado normas y procedimientos de obligado cumplimiento en las cuales nuestro sistema no reúne las condiciones necesarias para ser válido por los motivos expuestos al principio de estos párrafos.

2.6 Acciones necesarias a priori

Antes de decidirse a usar el sistema es conveniente tener en consideración una serie de cuestiones expuestas a continuación.

El servicio de la nube Firebase es gratuito si no se superan unos determinados umbrales de uso de recursos. Es importante conocer si el uso que se le pretende dar se encuentra dentro de estos umbrales para mantener la gratuidad o en caso contrario, si merece la pena abonar el coste mensual que supondría la utilización de un recurso como es una base de datos con las características de este sistema.

Por ejemplo, en nuestro proyecto se emplea principalmente la base de datos en tiempo real. Esta tiene una capacidad de almacenamiento gratuita de 1 GB y un límite de transferencia de datos de 10 GB mensuales los cuales se ha comprobado más que suficiente para la demostración de funcionamiento de este trabajo final de grado.



			Los proyectos se facturan mensualmente
 Realtime Database	GB almacenados	1 GB acerca de 20 mensajes de chat M	No cost
	GB transferidos	10 GB acerca de 200 mensajes de chat M	No cost
 Cloud Firestore	GiB almacenado	1 GiB acerca de 20 mensajes de chat M	No cost
	Escrituras de documentos	600,000 operaciones de escritura cantidad de veces que se escriben los datos	No cost
	Lecturas de documentos	1,500,000 operaciones de lectura cantidad de veces que se leen los datos	No cost

Figura 6: Límites del uso gratuito de Firebase. Fuente: <https://firebase.google.com/pricing>

También es importante mencionar que es necesario disponer de un ordenador desde donde primeramente configurar el sistema y después desarrollar y escribir los scripts y la aplicación web.

Por último, también hay que disponer de conexión a internet en la zona controlada. La Raspberry Pi puede conectarse a internet mediante wifi o cable ethernet cuando exista un *router* en la zona controlada. En los casos en los que no se dispone de acceso a un *router* se puede proveer de internet mediante un módulo de comunicaciones móviles con una tarjeta SIM. En ese caso es necesario que la zona controlada esté en un sitio con cobertura telefónica móvil.



Figura 7: Comunicación mediante redes móviles. Fuente: <https://magpi.raspberrypi.com/articles/raspberry-pi-cellular>

2.7 Configurar la Raspberry Pi

2.7.1 Reseña histórica de la Raspberry Pi

Eben Upton fundó Raspberry Pi Foundation en mayo de 2009 con la intención de introducir a jóvenes estudiantes en el mundo de la informática y la programación. Eben observó que el público general estaba perdiendo interés en estudiar informática y que los nuevos estudiantes no contaban con la preparación suficiente. Un método anticuado de enseñanza estaba alejando a posibles futuros científicos informáticos en una época en la que la tecnología estaba en auge y el sector necesitaba más trabajadores que nunca. También el alto coste de los ordenadores dificultaba que la mayoría de los alumnos pudieran experimentar y aprender con ellos. [10]

Para solucionar estos problemas, Eben y otros apasionados de la tecnología informática se propusieron desarrollar un ordenador sencillo y económico para que los institutos puedan enseñar informática y programación universalmente. El nombre del ordenador es Raspberry siguiendo la tendencia de las marcas de ordenadores de llamar a sus productos con nombres de comida (Apple, Acorns, Apricot...) y Pi haciendo alusión, según algunos, al lenguaje Python que empleaban las primeras versiones o como símil fonético a *pie* (pastel)... Los primeros prototipos comenzaron en 2006 basados en el microcontrolador ATmega644 hasta completar la primera versión definitiva a la venta en 2012. Desde entonces, la fundación benéfica Raspberry Foundation ha logrado llevar ordenadores a miles de escuelas en todos los continentes, promoviendo la enseñanza informática en países en desarrollo como en Afganistán y Sudáfrica y llegando a las 45 millones de unidades vendidas.

La primera Raspberry Pi en ponerse en venta al público fué la Raspberry Pi Model B en 2012 y costaba 35€. Tenía un procesador single-core, 512 MB de RAM con 2 puertos USB y un HDMI. Su factor de forma SBC (Single Board Computer) del tamaño de una tarjeta de crédito le hizo destacar entre el resto de ordenadores.

La segunda actualización de la computadora Raspberry llegó en 2015 con un aumento de la memoria RAM hasta 1GB y aumentando su procesador a un *quad-core* (*procesador de cuatro núcleos*). Esta versión era 4 veces más potente que su predecesora.

Una de las últimas versiones es la Raspberry 4 Model B y cuenta con hasta 8 GB de RAM, Wi-Fi y Bluetooth incorporado, Gigabit ethernet, 4 puertos USB (dos de ellos 3.0) 2 puertos microHDMI preparados para definición 4K entre otros.



Figura 8: Raspberry pi 4 Model B. Fuente: <https://linuxhint.com/raspberry-pi-history/>

Raspberry Pi Platform	RAM	Processor
Raspberry Pi B +	512MB	700 MHz ARM11
Raspberry Pi 2 B	1GB	900 MHz Quad-Core ARM Cortex-A7
Raspberry Pi 3 B	1GB	1.2 GHz, Quad-Core 64-bit ARM Cortex A53
Raspberry Pi 3 B +	1GB	1.4 GHz 64-bit ARM Cortex A53
Raspberry Pi 4B	Multiple options	Quad core Cortex-A72 (ARM v6) 64-bit SoC @ 1.5GHz.
Raspberry Pi 400	4 Gb	Cortex-A72 (ARM v8) 64-bit
Raspberry Pi Pico	264 Kb	RP2040 dual-core Arm Cortex-M0+

Tabla 1: Versiones de Raspberry Pi. Fuente: <https://linuxhint.com/raspberry-pi-history/>

La versatilidad, facilidad de uso y precio económico de la Raspberry Pi la llevó a encontrar sus aplicaciones en las industrias. La gran comunidad de voluntarios que han construido un ambiente tecnológico accesible y desarrollado variadas aplicaciones con objetivos educacionales, ha servido a la industria para desarrollar diseños adaptados a sus necesidades. Para muchas aplicaciones la potencia que proporciona la Raspberry es más que suficiente, necesitando un consumo mucho menor que los ordenadores comerciales cotidianos. [11]

En la actualidad podemos encontrar carcasas industriales [12] para la Raspberry adecuadas para un ambiente industrial, algunas incluso a prueba de explosiones. [13]



Figura 9: Carcasa para Raspberry pi para ambientes industriales explosivos. Fuente: <https://www.jeffgeerling.com/blog/2022/industrial-raspberry-pi-computers-one-explosion-proof>

2.7.2 Definición de SBC y funcionamiento básico

Los ordenadores de placa única SBC por sus siglas en inglés son ordenadores completos montados en una única placa de circuitos. El procesador, la memoria, el almacenamiento, la GPU y todos los conectores de entrada y salida se encuentran soldados en la misma placa de circuitos.

Presentan varias ventajas sobre los ordenadores convencionales: Al tener todos los componentes soldados en un mismo espacio, su producción es más sencilla, son más ligeros y compactos. Al necesitar menos cantidad de placas de circuitos y eliminar los conectores entre ellas el coste es menor. Los procesadores de los SBC más pequeños y menos potentes requieren una alimentación eléctrica menor, lo que los hace muy eficientes energéticamente, en comparación con los ordenadores multi placa. No obstante, también presentan desventajas, como por ejemplo la dificultad de reparación y su menor potencia, que aunque para la mayoría de aplicaciones es suficiente, no lo son para las actividades intensas en el uso de la CPU. [14]

Los usos más comunes de los SBC se encuentran en equipos con fines educativos, demostración de prototipos, paneles de control o aplicaciones integradas en un sistema más complejo. Su reducido tamaño y portabilidad los hacen la mejor opción para aplicaciones que requieren de estas características como por ejemplo sistemas robóticos u ordenadores de abordaje en barcos, camiones, sondas atmosféricas, etc.

Para nuestro proyecto vamos a utilizar un SBC como centro de procesamiento en nuestra instalación. De nuestro SBC obtendremos la información ambiental que deseemos gracias a los sensores acoplados y realizaremos acciones en nuestra instalación gracias a actuadores para encender/apagar/regular máquinas y equipos eléctricos.

A continuación se describe las partes que componen la Raspberry Pi 4 Model B.

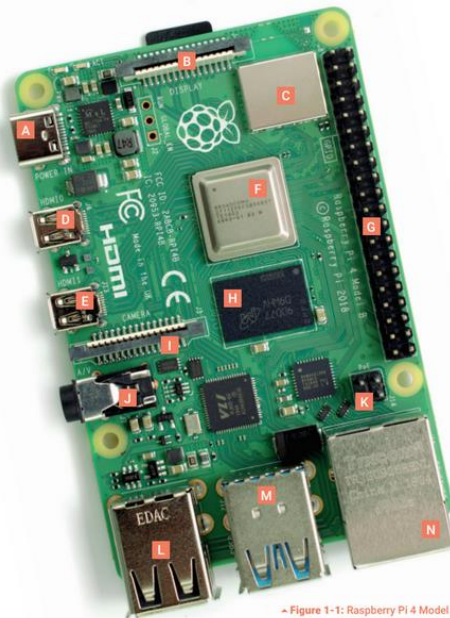


Figura 10: Componentes de la Raspberry pi. Fuente: <https://magpi.raspberrypi.com/books/beginners-guide-4th-ed>

A: Puerto de alimentación USB tipo-C: Mediante este puerto la Raspberry Pi recibe la energía que necesita para funcionar. La fundación Raspberry recomienda emplear sus transformadores oficiales ya que adaptan la alimentación a las repentinas variaciones de consumo que tienen los diferentes componentes y proporcionan la cantidad de amperaje necesaria (alrededor de los 3 A).

B: Puerto de video DSI (Display Serial Interface): Puerto diseñado para emplear el módulo “Raspberry Pi Touch Display” con el que se puede incorporar una pantalla táctil a la Raspberry.

C: Chip bluetooth y WiFi: Permite conectarse a otros dispositivos de manera inalámbrica.

D y E: Micro-HDMI: Dos puertos con la interfaz micro High Definition Multimedia Interface usado para conectar monitores, televisiones, proyectores, etc. a la Raspberry Pi. Soportan salida a 4K.

F: SoC (System on Chip): Bajo una cubierta de metal se encuentra un circuito integrado que compone la CPU y la GPU.

G: GPIO (General-Purpose Input/Output): Puerto específico de la Raspberry Pi con la función de comunicarse con otras placas de circuitos: Sensores, botones, detectores de movimiento, etc.

H: RAM: Disponible de 2GB hasta 8GB en la Raspberry 4 Model B

I: Puerto para cámara CSI (Camera Serial Interface): Puerto diseñado para emplear el módulo “Raspberry Pi Camera Module” con el que se puede capturar video.

J: Puerto 3.5mm AV: Puerto de salida de audio, puedes conectar unos altavoces. También puede usarse para transmitir una señal de video empleando un adaptador TRRS.

K: PoE (Power over Ethernet): Puerto que permite recibir alimentación eléctrica a partir de una red de comunicación de tipo Ethernet.

L: USB 2.0: Dos puertos normalmente utilizados para teclado y ratón.

M: USB 3.0: Más rápidos que la versión 2.0 se emplean para conectar a otros dispositivos, como por ejemplo Pendrives USB.

N: Puerto Ethernet: Usado para incorporar la Raspberry en una red de ordenadores usando el conector RJ45. Incluye dos LEDs para monitorizar la conexión.

En la parte inferior de la placa encontramos el conector de tarjetas microSD. Aquí guardaremos el almacenamiento de la Raspberry Pi, los archivos, los programas y el sistema operativo Raspberry OS estarán en este puerto en una tarjeta microSD.



Figura 11: Puerto para tarjeta microSD. Fuente: <https://magpi.raspberrypi.com/books/beginners-guide-4th-ed>

2.7.3 Listado de sensores viables

La versatilidad y variedad que ofrece la Raspberry Pi es uno de los motivos de su elección para este trabajo. En internet podemos encontrar sensores y actuadores de cualquier tipo y para cualquier situación. A continuación se presenta una lista de sensores/actuadores compatibles con nuestro proyecto que podrán ser utilizados por los futuros usuarios.

-Temperatura: MCP9808 High Accuracy I2C Temperature Sensor Breakout Board:

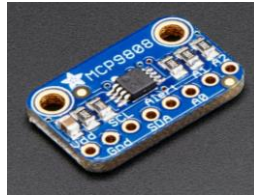


Figura 12: Sensor de temperatura. Fuente: <https://www.adafruit.com/product/1782>

Sensor	MCP9808
Precisión	$\pm 0.25^{\circ}\text{C}$
Rango de medidas	-40°C to $+125^{\circ}\text{C}$
Resolución	0.0625°C
Voltaje de alimentación	2.7V to 5.5V
Interfaz	I2C

Tabla 2: Características del sensor de temperatura. Fuente: <https://www.adafruit.com/product/1782>

-Presión: BMP180



Figura 13: Sensor de Presión. Fuente: <https://tutorials-raspberrypi.com/raspberry-pi-and-i2c-air-pressure-sensor-bmp180/>

Sensor	BMP180
Precisión	$\pm 0.12\text{hPa}$
Rango de medidas	300 to 1100 hPa
Voltaje de alimentación	1.3V – 3.6V
Interfaz	I2C

Tabla 3: Características del sensor de presión. Fuente: <https://tutorials-raspberrypi.com/raspberry-pi-and-i2c-air-pressure-sensor-bmp180/>

-Humedad: DHT11/DHT22



Figura 14: Sensor de Humedad. Fuente: <https://tutorials-raspberrypi.com/raspberry-pi-measure-humidity-temperature-dht11-dht22/>

Sensor	DHT11
Precisión	±1% (humedad relativa)
Rango de medidas	20% to 90%
Voltaje de alimentación	3.5V to 5.5V
Interfaz	I2C

Tabla 4: Características del sensor de humedad. Fuente: <https://tutorials-raspberrypi.com/raspberry-pi-measure-humidity-temperature-dht11-dht22/>

-Giroscopio/aceleración: MPU6050

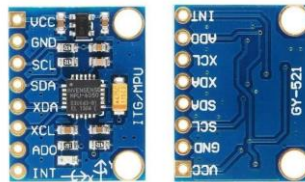


Figura 15: Sensor de aceleración. Fuente: <https://www.raspberrypistarterkits.com/guide/raspberry-pi-accelerometer-gyroscope/>

Sensor	MPU6050 Giroscopio	MPU6050 Acelerometro
Rango de medidas	250 to 2000 ° / s	2g to 16g
Voltaje alimentación	3V-5V	3V-5V
Interfaz	ICC	ICC

Tabla 5: Características del sensor de aceleración. Fuente: <https://www.raspberrypistarterkits.com/guide/raspberry-pi-accelerometer-gyroscope/>

-Luz solar: ALS-PT19

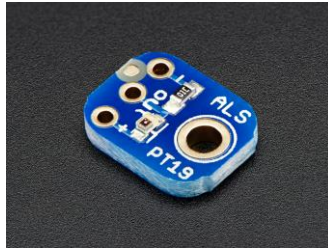


Figura 16: Sensor de luz solar. Fuente: <https://www.adafruit.com/product/2748>

Sensor	ALS-PT19
Rango de medidas	0-10.000 Lux
Voltaje de alimentación	2.5V to 5.5V
Interfaz	I2C

Tabla 6: Características del sensor de luz solar. Fuente: <https://www.adafruit.com/product/2748>

-Detector de CO2: SCD30



Figura 17: Sensor de CO2. Fuente: <https://sensirion.com/products/catalog/SCD30/>

Sensor	SCD30
Precisión	±30 ppm
Rango de medidas	400 to 10000 ppm
Voltaje de alimentación	3.3V – 5.5V
Interfaz	I2C

Tabla 7: Características del sensor de CO2. Fuente: <https://sensirion.com/products/catalog/SCD30/>

2.7.4 Interfaz de los sensores

La comunicación entre los sensores y la Raspberry Pi debe realizarse mediante una interfaz. Podemos interpretar la interfaz como el canal físico de comunicación, así como el software que respeta cierto protocolo para la comunicación con el sensor. La interfaz gestiona el envío y recepción de datos a través de cables mediante señales eléctricas. Existen diversos tipos de interfaces. A continuación se describen algunos de ellos:

I2C: Formado por las siglas de “Inter-Integrated circuit” es una interfaz muy utilizada por su simplicidad y fácil integración. La interfaz está formada por dos cables con comunicación bidireccional: SDL (Serial Data Line) que transmite los datos y SCL (Serial Clock Line) que establece el tiempo y sincroniza la comunicación. I2C soporta hasta 128 dispositivos usando direcciones de 7 bits y 1024 dispositivos si se aumentan las direcciones hasta los 10 bits. Las señales siguen un patrón de “*reconocimiento/no reconocimiento*” (acknowledge/not-acknowledged pattern) con un bit de señal de *reconocimiento* para cada bit de datos. En I2C puede designarse cualquier dispositivo como el maestro o como receptor [15].

SDI-12: Serial Digital Interface a 1200 baudios es un protocolo de comunicaciones asíncrono y en serie (comunicación serial) para sensores inteligentes que monitorean los datos del entorno. Estos instrumentos suelen ser de baja potencia (12 voltios), se utilizan en ubicaciones remotas y generalmente se comunican con un data logger u otro dispositivo de adquisición de datos. El protocolo sigue una configuración cliente-servidor mediante la cual un registrador de datos (grabador SDI-12) solicita datos de los sensores inteligentes (sensores SDI-12), cada uno identificado con una dirección única. [16]

SPI: Serial Peripheral Interface en el que la línea MOSI (Master Output Serial Input) se utiliza para enviar datos del master al esclavo y la línea MISO (Master Input Serial Output) en sentido opuesto, todo ello coordinado por la señal de reloj SCLK. Líneas individuales de selección se emplean para requerir la atención de uno u otro esclavo desde el maestro [17].

2.7.5 Librerías de los sensores

Gestionar las interfaces de los sensores, detectar pulsos eléctricos por las líneas de datos y traducirlo a información es la tarea de la que se encargan las librerías o bibliotecas de programación de los sensores. Las librerías sirven para aliviar la complejidad de la programación para conseguir la comunicación con los sensores, simplificando notablemente el uso del hardware por parte del programador.

En nuestro caso empleamos la librería de la placa de sensores Sense Hat llamada sense-hat [18] . Para utilizarla primero hay que descargar de internet mediante los comandos “sudo apt-get install sense-hat” en una terminal dentro de la propia Raspberry Pi, y luego llamarla desde el script de python usando la orden, cargándola previamente con “from sense_hat import SenseHat”, tal y como se observará en ejemplos posteriores.

2.7.6 Sistema Operativo y Software

Para hacer funcionar el ordenador Raspberry es necesario emplear su sistema operativo: Raspberry Pi OS. Para ello debemos instalar el sistema operativo en una tarjeta de memoria microSD empleando el software oficial de la marca llamado Raspberry Pi Imager [19], con descarga gratuita disponible en su web: <https://www.raspberrypi.com/software/>

Para emplear el Raspberry Pi Imager necesitaremos otro ordenador que disponga de un lector de tarjetas SD/microSD.

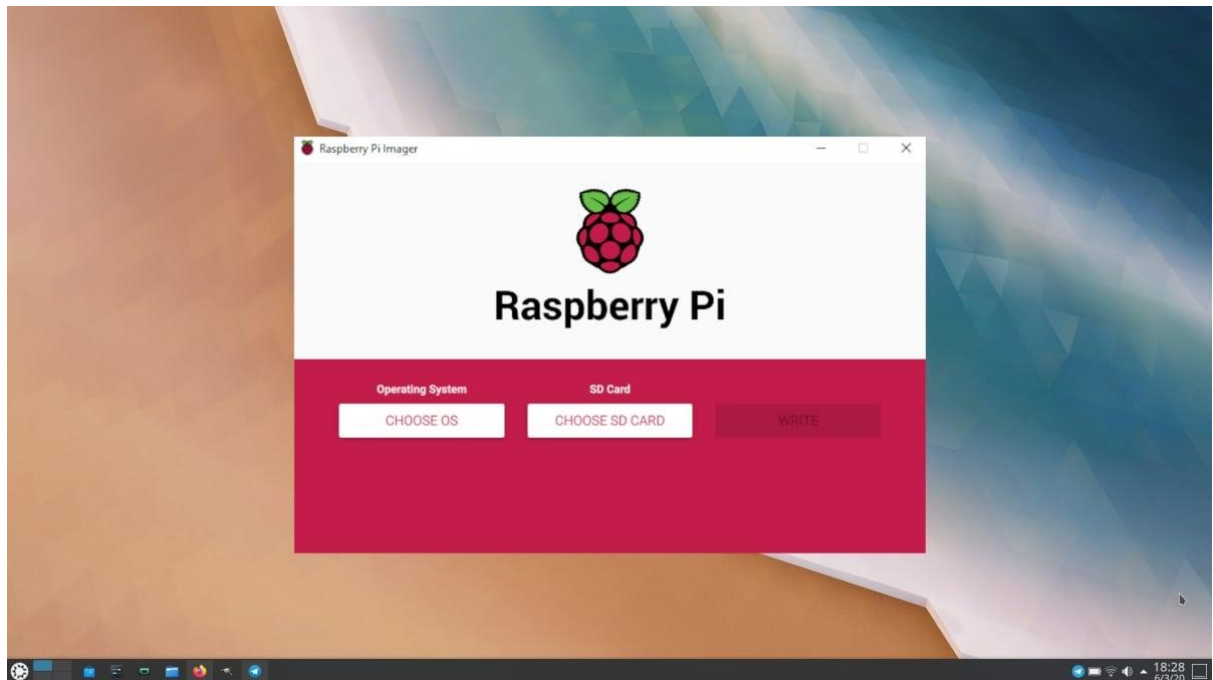


Figura 18: Raspberry pi imager. Fuente: <https://linux-os.net/raspberry-pi-imager-es-la-herramienta-oficial-de-raspberry-para-instalar-imagenes-en-sus-famosas-placas/>

Una vez abierto el programa Raspberry Pi Imager en otro ordenador y conectada la memoria microSD, se selecciona la versión que se desea instalar, se selecciona la tarjeta microSD y pulsamos el botón “Write”. El programa nos preguntará el nombre de usuario y una contraseña para el inicio de sesión cada vez que encendamos la Raspberry. Una vez finalizado el proceso de instalación

ya podemos introducir la memoria microSD en el puerto microSD de la Raspberry Pi y encenderla conectándola a la alimentación eléctrica.

Una vez encendido e introducido nuestro usuario y contraseña, aparece el escritorio de la raspberry Pi. La forma de crear carpetas, navegar por los archivos, utilizar el escritorio es idéntica a los sistemas operativos comunes como Windows y MacOS. El dispositivo está listo para guardar archivos o scripts para ser ejecutados.

El lenguaje de programación Python se descarga junto con el sistema operativo de la Raspberry Pi automáticamente, no es necesario instalarlo explícitamente.

2.7.7. Arranque automático de script

En muchas ocasiones interesa que el script que deba estar continuamente ejecutándose arranque de forma automática al proporcionar tensión al sistema; de ese modo, frente a una pérdida eventual de la misma, cuando todo el sistema arranque, todo volverá a funcionar. En el sistema Linux de la Raspberry Pi podemos hacerlo de varias maneras; nosotros emplearemos el comando `crontab` para lograrlo.

Primeramente crearemos un fichero de script que es un fichero ejecutable directamente por el intérprete de comandos de línea del sistema operativo. En este caso lo llamaremos `arranque.sh` y tendrá el siguiente contenido

```
python/home/dario/teler/Software/Raspberry/sensores2FBLEDS  
_push.py
```

donde observamos el comando `python` que invoca al intérprete de Python y después el nombre del fichero Python con el script que debe ejecutarse (con un direccionamiento absoluto, desde la raíz del sistema de ficheros).

El fichero `arranque.sh` deberá tener permisos de ejecución:

```
chmod +x arranque.sh
```

A continuación lo que haremos será indicar mediante el citado comando `crontab` que el script se ejecute tras arrancar el sistema:

```
sudo crontab -e
```

Este comando permitirá elegir previamente el editor de texto con el que añadir la siguiente línea al fichero de configuración, con el que se indicará que se ejecute el script en cuestión cuando arranque el sistema.

```
@reboot sh  
/home/dario/Documentos/TeleNube/Planta/pruebasSense/arranque.sh
```

2.8 Configurar la nube Firebase

En <https://firebase.google.com> podemos encontrar la página web con la que establecemos un primer contacto con el producto de Google denominado Firebase.

Firebase es una plataforma de desarrollo de aplicaciones de Google que permite añadir a nuestra aplicación bases de datos en tiempo real, esquemas de autenticación, obtención de analíticas de comportamiento de nuestro producto, y un largo etcétera a unos precios contenidos, si no gratuitos cuando el uso de recursos es esporádico o relativamente pequeño.

El uso del producto se lleva a cabo interactuando con la denominada “consola” de Firebase que no es más que una interfaz de control del producto permitiendo la configuración y el ajuste según nuestras necesidades.

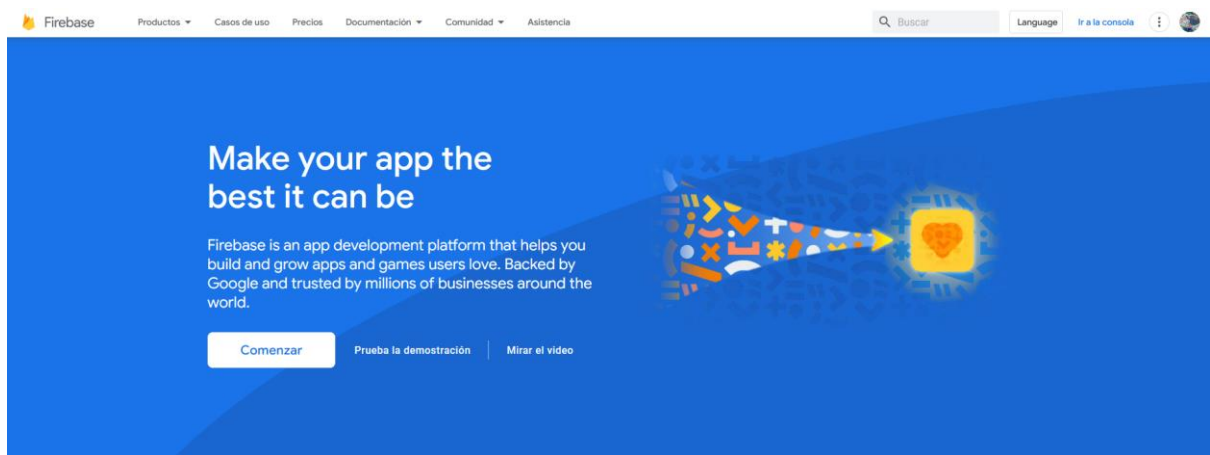


Figura 19: Inicio de Firebase

De la gran cantidad de funcionalidades que el producto ofrece, nosotros solo vamos a emplear la utilización de la base de datos en tiempo real para almacenar y acceder a los resultados de las medidas y controles llevados a cabo en nuestro sistema, así como, parcialmente, el sistema de autenticación para permitir validar la prueba de concepto al utilizar esta plataforma.

Para ello, lo primero que debemos hacer, una vez dentro de la consola es crear un proyecto Firebase.

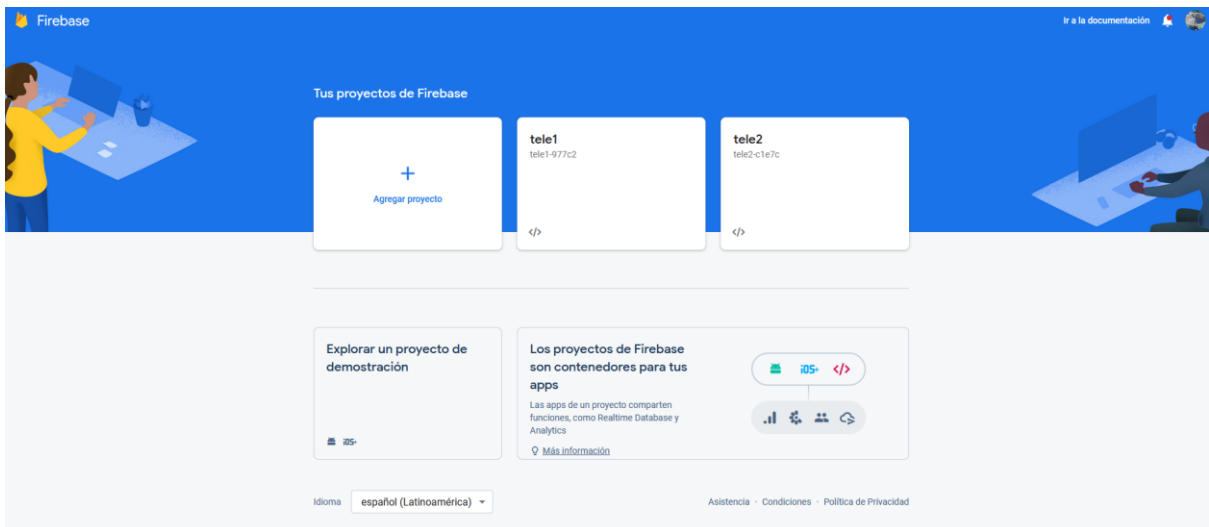


Figura 20: Ventana de proyectos en Firebase

2.8.1. Configuración de la base de datos en tiempo real

Una vez indicado el nombre del proyecto con el que se va a trabajar, deben configurarse los componentes básicos de Firebase. En primera instancia configuraremos las características necesarias para la base de datos en tiempo real donde serán almacenadas las muestras que los sensores captan en la instalación y sean convenientemente enviadas a Firebase.

La pantalla de configuración de la base de datos en tiempo real posee el aspecto de la siguiente figura donde observamos 4 pestañas que permiten configurar cuatro aspectos de la misma: Datos, Reglas, Copias de seguridad y Uso.

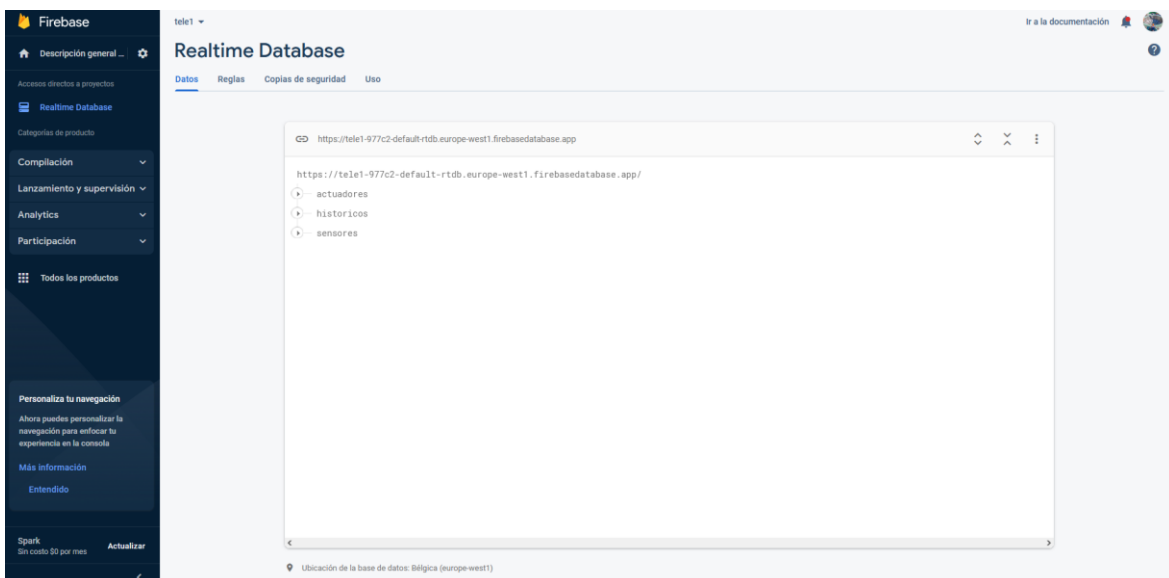


Figura 21: Base de datos en tiempo real de Firebase

Las que resultan de nuestro interés son Datos y Reglas. La primera de ellas permite tanto observar cómo configurar una estructura de datos de tipo arborescente siguiendo una filosofía de notación de tipo JSON (JavaScript Object Notation). En principio no sería necesario crear ninguna estructura inicial ya que con los primeros accesos de escritura a la base de datos se iría conformando la estructura de la misma. En tiempo real podemos ver cómo se van modificando las entradas de la misma. A continuación se muestra la estructura:

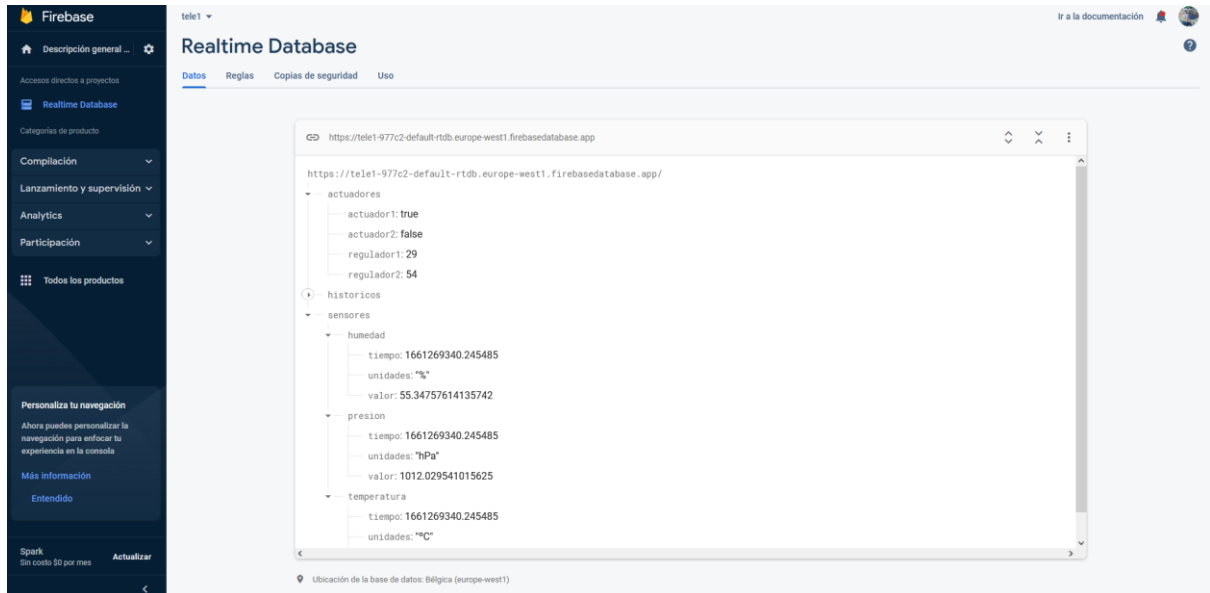


Figura 22: Estructura de la base de datos

Desde el punto de vista de una aplicación (web o en Raspberry), el acceso (lectura o escritura) tendría como referencias, por ejemplo

`<REF>/actuadores/actuador1`

para el actuador1, denotando `<REF>` la referencia al nodo inicial o raíz de la base de datos. O bien

`<REF>/sensores/humedad/tiempo`

o

`<REF>/sensores/humedad/unidades`

o

`<REF>/sensores/humedad/valor`

para la última muestra del sensor de humedad.

En el siguiente ejemplo se muestra la estructura de los históricos de una medida, donde se han ido acumulando en el tiempo todas las muestras ordenadamente:

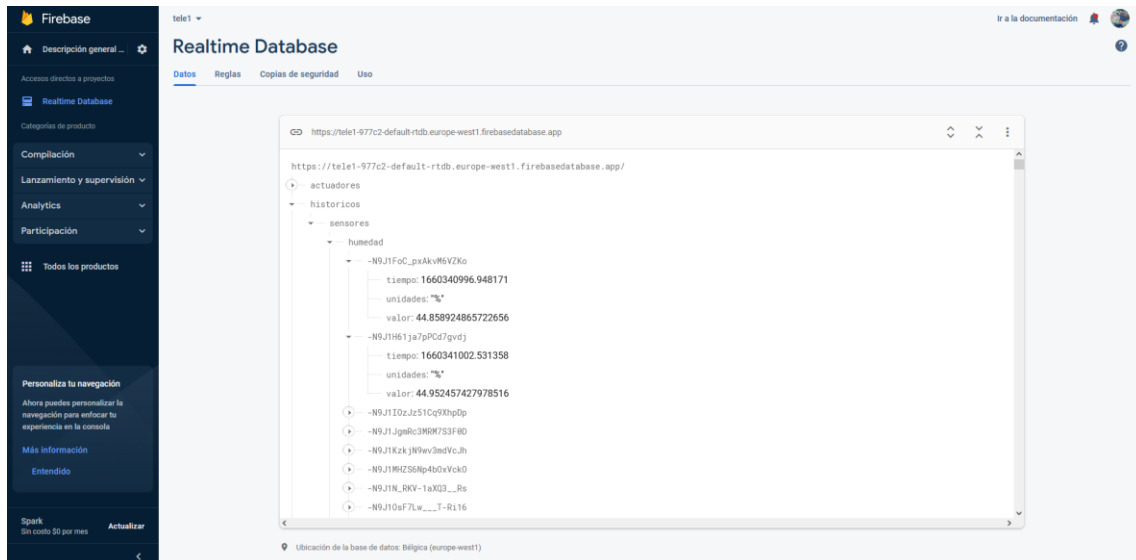


Figura 23: Históricos en la base de datos

La otra pestaña interesante nos permite imponer reglas de acceso a la base de datos, distinguiendo claramente reglas para escritura y reglas para lectura. En nuestro caso particular, ya que dejamos como línea futura el diseño del software para el control de acceso, las reglas de escritura y lectura se circunscriben a poner una fecha tope en la que, en principio, no planteamos ningún tipo de control. Una vez correctamente establecido el control de acceso, cada individuo tendrá un identificador con el que se podrá describir mediante cierta sintaxis propia de Firebase la capacidad de leer o escribir en dicha base de datos, proporcionando ya las características propias de sitios seguros: privacidad, no suplantación, autenticidad, etc.

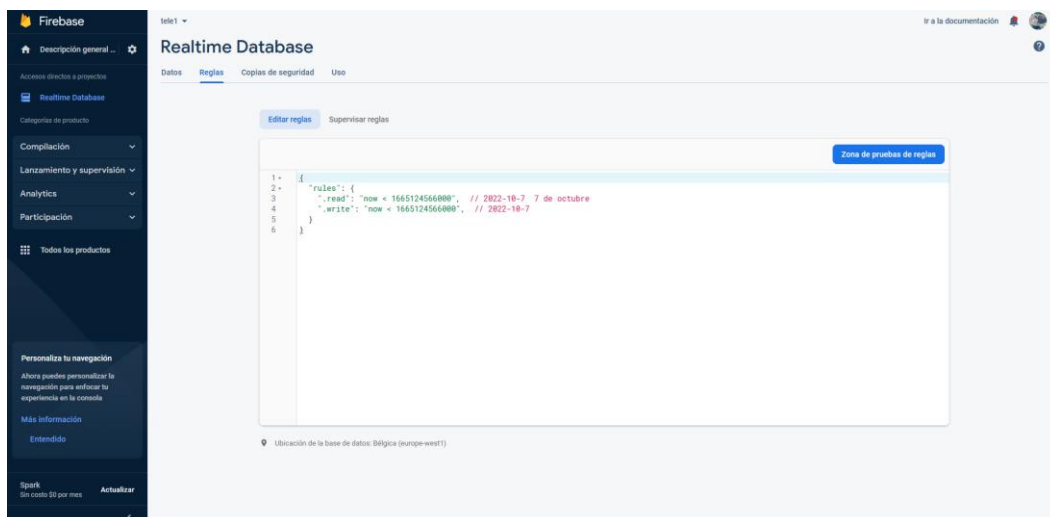


Figura 24: Las reglas de la base de datos

En el ejemplo anterior, las reglas, tanto de lectura como de escritura, se circunscriben a permitirla, siempre y cuando el instante de acceso (“now”) sea inferior a cierto tiempo (tiempo UNIX: cantidad de tiempo en ms desde el 1 de enero de 1970). Por lo tanto, se puede acceder a la base de datos (lectura y escritura) solo por tiempo limitado.

2.8.2. Autenticación en la base de datos

La ventana de autenticación de la base de datos se corresponde con la siguiente figura:

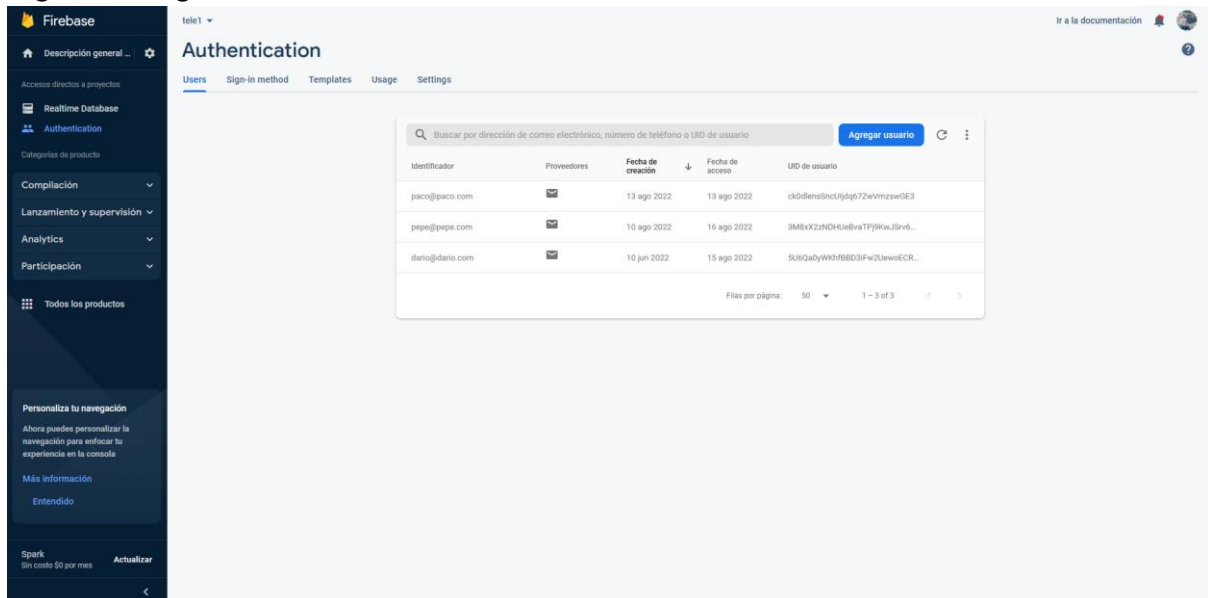


Figura 25: Ventana de Autenticación

En la cual podemos observar las pestañas Users, Sign-in method, Templates, Usage y Settings. Serán de nuestro interés Users y Sign-in method.

Recordemos que hemos dejado como línea futura la parte de autenticación debido a que no formaba parte de la prueba de concepto y necesitábamos focalizar el esfuerzo en aspectos que fueran realmente operativos. Ni que decir tiene que en un producto comercial es intrínsecamente necesario proporcionar un control de acceso que permita que sólo los usuarios autorizados puedan acceder a controlar/monitorizar el sistema (asimismo se podrían plantear cierta jerarquía en el acceso a dichos valores del sistema).

En la primera pestaña, Users, se muestra la lista de usuarios que ya están autorizados de alguna manera, junto con su esquema de acceso, fechas de alta y acceso y el identificador del usuario. Estos usuarios se pueden agregar

manualmente o bien mediante un esquema de auto-registro. Hemos implementado este último caso por simplicidad, para poder crear usuarios arbitrariamente y realizar mínimas pruebas funcionales.

En la segunda pestaña, se puede especificar el esquema de autenticación. En todos nuestros casos, por simplicidad, se ha decidido que sea un identificador personal en forma de dirección de correo electrónico personal junto con una contraseña, que es la que aparece a continuación:

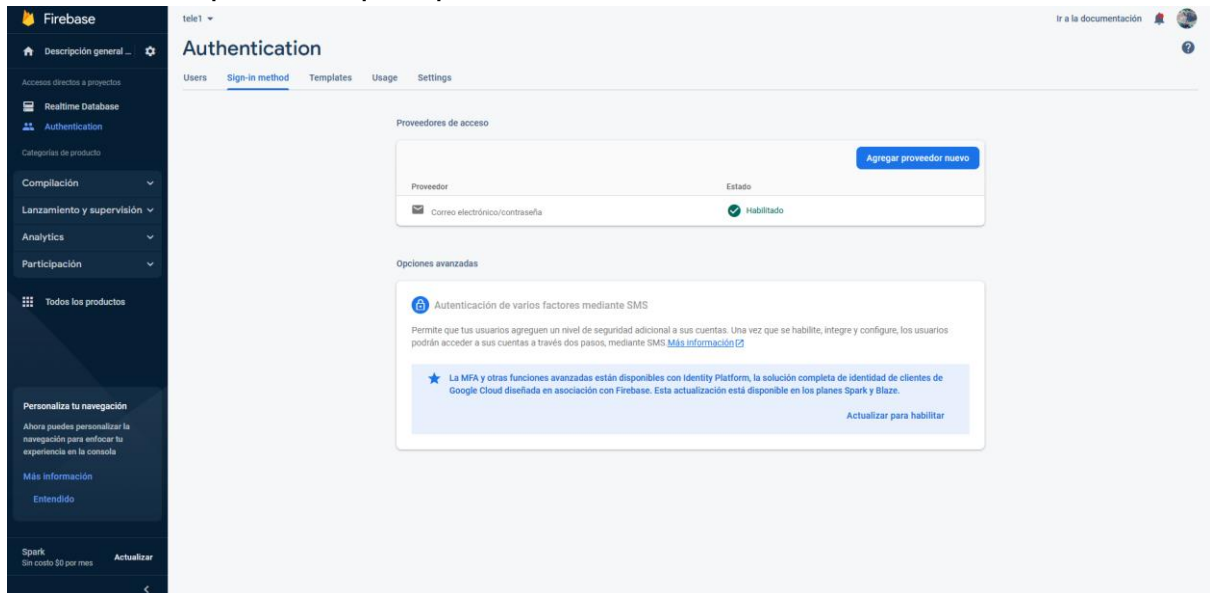


Figura 26: Especificar el esquema de autenticación

Existen numerosos esquemas alternativos que requieren un tratamiento particular en la aplicación web asociada, pero insistimos que por simplicidad vamos a dejarlos para una línea futura de extensión del proyecto, como pudiera ser: por usuario de GMail, de Facebook, Twitter, Github, ...

3 Ejemplos prácticos y puesta en marcha

3.1 Descripción del ejemplo que hemos propuesto

El sistema propuesto en este trabajo final de grado ofrece muchas y muy diversas aplicaciones en distintos ámbitos. Para ejemplificar, un agricultor dueño de un invernadero que desea poder supervisar y controlar sus cultivos a distancia para no tener que desplazarse podría dar un buen uso a el sistema que se propone. Con sensores de humedad de la tierra, sensores de radiación solar, sensores de temperatura puede monitorizar las variables que le interesen y añadiendo actuadores puede controlar a distancia un ventilador para enfriar, encender luces para que las plantas crezcan por la noche, encender una bomba de agua para regar, etc. Otro ejemplo sería una pequeña empresa que maduran quesos en cuevas (por ejemplo el queso de cabrales) pero debido a problemas con variaciones de la humedad sus quesos no han resultado ideales. Usando el sistema propuesto en este TFG, se podría vigilar que los niveles de humedad ambiental estuvieran en determinados intervalos de forma remota.

Otro ejemplo más común puede ser el de un particular o un negocio que quiere monitorizar el consumo eléctrico de su vivienda o local. Con sensores de potencia aplicados al sistema descrito pueden monitorizar máquinas eléctricas individuales o de todo el local/vivienda, a un precio económico.

Como se puede observar hay muchas aplicaciones que pueden justificar el estudio de nuestro proyecto. En nuestro caso vamos a demostrar el funcionamiento del sistema con un ejemplo abstracto utilizando una placa llamada SenseHat [20] , desarrollada por la fundación Raspberry Pi. La utilidad de esta placa a añadir a Raspberry Pi es que contiene una gran cantidad y variedad de sensores tal que permite tener una visión general de potencial comportamiento en una situación real.

El Sense Hat tiene una matriz de 8x8 LED, un joystick y sensores de temperatura, humedad, presión atmosférica, magnetómetro, acelerómetro y un giroscopio. La organización de Raspberry Pi ha creado una librería [18] en Python desde la cual es sencillo controlar sus funcionalidades.

Para la prueba de concepto vamos a utilizar los sensores de temperatura y de humedad como inputs de nuestra zona controlada y la matriz led como output simulando los actuadores. Los actuadores 1 y 2 encenderán y apagarán como un interruptor bombillas LED simulando que controlan cualquier mecanismo eléctrico. Los reguladores 1 y 2 también encenderán un número determinado

de bombillas led en una fila en función de su valor entre 0 y 100 simulando una máquina eléctrica con control variable, como la apertura de una válvula.

Desde la aplicación web se controla dos interruptores con dos posiciones: encendido y apagado, y dos reguladores variables. Desde la web también se muestran las lecturas de los sensores de temperatura y humedad, con opción a representar gráficamente un histórico eligiendo la fecha a representar. De esta manera se demuestra el funcionamiento del proyecto.

3.2 Preparación del hardware: Raspberry pi y SenseHat

Para encender la Raspberry Pi conectamos el cable de alimentación micro-USB a una toma de corriente con un transformador adecuado, siendo el más recomendado el transformador propio de la organización Raspberry Pi [21].

Para acoplar el Sense Hat a la Raspberry Pi hay que colocarlo sobre los pins del GPIO de forma que el Sense Hat quede tapando la Raspberry, como si fuera un sombrero, de ahí su nombre.

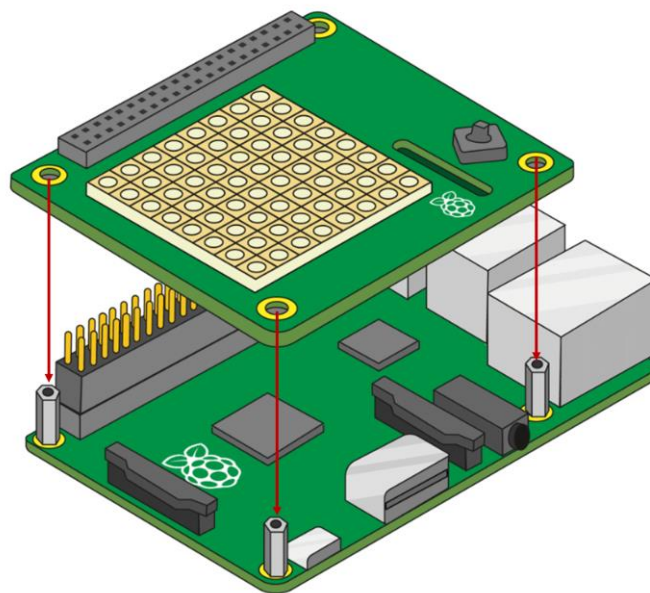


Figura 27: Montaje del Sense Hat. Fuente:

<https://projects.raspberrypi.org/en/projects/getting-started-with-the-sense-hat/2>

Tenemos varias alternativas para trabajar con la Raspberry Pi para instalar el software necesario:

- Conectar una pantalla al puerto HDMI, y un teclado y un ratón a los puertos USB disponibles.
Esto requiere de estos tres dispositivos hardware.

- Si la Raspberry Pi está conectada a nuestra subred y se ha activado la escucha del puerto asociado al servicio SSH, podremos conectarnos a ella mediante un ordenador convencional mediante la aplicación ssh.
- En las mismas condiciones que en el caso anterior, el editor de textos Visual Studio Code puede hacer las funciones, tanto de editor de textos convencional como de terminal ssh (instalando cierta extensión), proporcionando una gran versatilidad y comodidad en el trabajo.

En nuestro caso emplearemos la citada extensión del programa Visual Studio Code.

Una vez conectado a la Raspberry Pi mediante SSH/Visual Studio Code y acoplado el conjunto Raspberry y Sensor Hat podemos empezar a añadir el script cuyo objetivo es la comunicación entre los sensores y la nube Firebase.

3.3 Descripción de scripts de Python empleados

Tal y como comentábamos anteriormente, el editor de texto empleado ha sido Visual Studio Code, desarrollado por Windows, aunque disponible para cualquier plataforma (Windows, Linux y Mac-OS. Para poder conectarse de forma inalámbrica a la Raspberry descargamos el complemento gratuito llamado “Remote - SSH” directamente en la pestaña de extensiones.

Una vez ya, editando las primeras líneas del script de control, realizamos la importación de los paquetes necesarios con el comando “import”. Necesitamos la librería “pyrebase” (API wrapper) para enviar y recibir información de la nube Firebase [22]. La librería “SenseHat” la necesitamos para poder manejar la placa con sensores que tenemos acoplada a nuestra Raspberry Pi y su funcionamiento sea el correcto. La librería “time” la utilizamos para pausar el código momentáneamente y simular de alguna manera cierta frecuencia de muestreo y la librería datetime para trabajar con unidades de tiempo UNIX (cantidad de segundos/o milisegundos transcurridos desde el 1 de enero de 1970)

```
1 import time
2 from sense_hat import SenseHat
3 import pyrebase
4
```

En la línea 5 guardamos la función SenseHat() en la variable “sense” para hacer más cómodo su uso a lo largo del código. Seguido, en la línea 6 usamos “sense” y le aplicamos el método “.clear()”: esto hace apagar los LED en caso de que alguno estuviera encendido.

```
5 sense = SenseHat()
6 sense.clear()
```

En las líneas 8-16 escribimos los datos necesarios para identificar el proyecto de Firebase con el que vamos a trabajar (estos datos podemos conseguirlos en la ventana de configuración del proyecto Firebase, ubicado en la consola de Firebase).

En la línea 15 se inicializan los programas que permiten establecer comunicación con Firebase y la guardamos en un objeto con nombre “firebase” para facilitar su manejo en adelante. En la línea 16 guardamos en otro objeto llamado “db” la función que referencia a la base de datos en tiempo real integrada en Firebase.

```
7
8 config = {
9     "apiKey": "AIzaSyBXrFGQtK6YQZ5hivip1jCiSsF8nqWz1Hw ",
10    "authDomain": "tele1-977c2.firebaseio.com",
11    "databaseURL": "https://tele1-977c2-default-rtdb.europe-west1.firebaseio.com",
12    "storageBucket": "tele1-977c2.appspot.com"
13 }
14
15 firebase = pyrebase.initialize_app(config)
16 db = firebase.database()
```

En estas líneas definimos colores para poder iluminar los leds del color que queramos más fácilmente. En este caso negro también es apagado.

```
17
18 negro = [0, 0, 0]
19 rojo = [255, 0, 0]
20 verde = [0, 255, 0]
21 azul = [0, 0, 255]
```

En las líneas 23-28 definimos la función fila_leds. Esta función ilumina leds en una fila representando un número del 0 al 100. Si el número es 50, se encenderán la mitad inferior de los leds de la fila, si el número es 100 se encenderá toda la fila. Con un bucle for, recorre cada led de la fila y si el número es mayor que la división en 8 partes, se enciende el led en rojo.

```

22
23 def fila_leds(fila,x):
24     print("datos bucle")
25     for i in range(8):
26         sense.set_pixel(i, fila, negro)
27         if (x>100./8.*i):
28             sense.set_pixel(i, fila, rojo)
29
30

```

De las líneas 31 hasta 75 se define la función `stream_handler`. El objetivo de esta función es ordenar a la Raspberry que realice las órdenes que le llegan desde la nube. Por ejemplo, si el usuario cambia el estado de algún actuador la nube enviará un mensaje mostrando la actualización. Esta función recibe el mensaje y ordena a la Raspberry que active/desactive dicha salida.

El mensaje que recibe de la nube es un diccionario con entradas `path` y `data`. `path` nos indica la RAÍZ del actuador que ha sido actualizado y `data` nos indica su valor. Cuando se inicia el *script*, el mensaje llega actualizando el estado de todos los actuadores al mismo tiempo, en tal caso `path` contiene el directorio raíz, (una barra) y en `data` otro diccionario con los nombres de cada actuador y su valor. Cuando se actualiza más tarde un único actuador, en “`path`” aparece la ruta del actuador manipulado y en “`data`” su valor.

```
{'path': '/', 'data': {'actuador1': False, 'actuador2': False, 'regulador1': 29, 'regulador2': 54}, 'event': 'put'}
```

Ejemplo de mensaje, Actualización inicial

```
{'path': '/actuador1', 'data': True, 'event': 'put'}
```

Ejemplo de mensaje, Actualización individual

tal y como puede comprobarse en el apartado de 2.8.1. Configuración de la base de datos en tiempo real.

La función primero comprueba la ruta del mensaje. Si la ruta es '/' tal y como ocurre al principio de ejecutar el script, la función evalúa el estado de los actuadores dentro del diccionario “data”. Si el valor es `True`, se enciende su correspondiente led. En caso contrario se mantiene apagado. Esto lo hace tanto para el `actuador1` como para el `actuador2`.

Para los actuadores llamados “`regulador1`” y “`regulador2`” la función recibe su valor y lo envía a la función “`fila_leds`” antes descrita. Como la función “`fila_leds`” solo admite números enteros como argumentos, y el valor del

regulador está en formato cadena de caracteres “char” en nuestro mensaje, es necesario convertirlo a un entero usando la función `int()`.

Después de recibir el mensaje inicial, la función también debe gestionar las actualizaciones individuales de cada actuador. En este caso en “`path`” nos indica la ruta del único actuador actualizado y en `data` su valor. La función comprueba qué actuador ha sido actualizado leyendo el “`path`” y actúa de manera similar al caso anterior encendiendo o apagando LED.

Ante el hipotético caso en el que el usuario actualice el estado de varios actuadores al mismo tiempo, el valor de “`path`” sería '/' y “`data`” sería otro diccionario con los nombres de los actuadores y sus valores. Si la función recibe este mensaje se produce un error “`KeyError`” e ignora los cambios realizados en los actuadores. Para evitar esto usamos “`try:`” y “`except`”. Esto hace que python ejecute determinadas líneas de código excepto cuando se produzcan excepciones (en este caso cuando ocurre un `KeyError`)

```

31 def stream_handler(message):
32     print(message)
33     if (message['path']=='/'):
34
35         try:
36             if (message['data']['actuador1']):
37                 sense.set_pixel(0, 0, rojo)
38             else:
39                 sense.set_pixel(0, 0, negro)
40         except KeyError:
41             print("Excepcion cazada individual")
42
43         try:
44             if (message['data']['actuador2']):
45                 sense.set_pixel(0, 1, rojo)
46             else:
47                 sense.set_pixel(0, 1, negro)
48         except KeyError:
49             print("Excepcion cazada individual")
50
51         try:
52             fila_leds(2, int(message['data']['regulador1']))
53         except KeyError:
54             print("Excepcion cazada individual")
55
56         try:
57             fila_leds(3, int(message['data']['regulador2']))
58         except KeyError:
59             print("Excepcion cazada individual")
60
61     elif (message['path']=='/actuador1'):
62         if (message['data']):
63             sense.set_pixel(0, 0, rojo)
64         else:
65             sense.set_pixel(0, 0, negro)
66     elif (message['path']=='/actuador2'):
67         if (message['data']):
68             sense.set_pixel(0, 1, rojo)
69         else:
70             sense.set_pixel(0, 1, negro)
71     elif (message['path']=='/regulador1'):
72         fila_leds(2, int(message['data']))
73     elif (message['path']=='/regulador2'):
74         fila_leds(3, int(message['data']))
75     print("fin stream_handler")
76

```

En el campo de las comunicaciones, un *stream* representa una conexión de comunicación entre dos entidades. En la línea 77 se emplea la función `stream()` en la rama de nuestra base de datos llamada “actuadores” mediante la función `child()`. La función `stream` recibe información de la base de datos de la nube y la trae al *script* únicamente cuando detecta que se ha producido un cambio. De esta forma, solo cuando el usuario actualiza en la aplicación web el estado de algún actuador se detecta una variación y se transmite el mensaje. En nuestro *script* enviamos el mensaje a la función `stream_handler` antes descrita.

```
77 my_stream = db.child("actuadores").stream(stream_handler)
78
```

Por último, dentro de un bucle `while` que se repetirá indefinidamente, se leen las variables temperatura, presión y humedad con las funciones de la librería del SenseHat `get_temperature()`, `get_pressure()` y `get_humidity()`.

Seguidamente se envían los valores de las variables a la nube con la función `.set()` y las guardará en la base de datos usando `child()`, en el nodo de la variable que corresponda (“temperatura”, “presion”, “humedad”) que a su vez se encuentran en el nodo “sensores”.

```
79 while(True):
80
81     temp = sense.get_temperature()
82     pressure = sense.get_pressure()
83     humidity = sense.get_humidity()
84
85     print("temperatura: %5.2f °C, presion: %7.2f mm Hg, humedad rel.: %5.2f %" % (temp, pressure, humidity))
86
87     datoTemperatura = {"valor": temp, "unidades": "°C"}
88     db.child("sensores").child("temperatura").set(datoTemperatura)
89     db.child("sensores_push").child("temperatura").push(datoTemperatura)
90
91     datoPresion = {"valor": pressure, "unidades": "mm Hg"}
92     db.child("sensores").child("presion").set(datoPresion)
93     db.child("sensores_push").child("presion").push(datoPresion)
94
95     datoHumedad = {"valor": humidity, "unidades": "%"}
96     db.child("sensores").child("humedad").set(datoHumedad)
97     db.child("sensores_push").child("humedad").push(datoHumedad)
98
99     time.sleep(5)
```

Tras usar la función `.set()`, se emplea la función `.push()` de la misma manera para guardar los valores junto con una marca temporal en una base de datos a la que accederemos más tarde.

Con la función **“time.sleep(segundos)”** emulamos cierta frecuencia de muestreo en la recopilación de los datos, que en nuestro caso hemos elegido arbitrariamente con 5 segundos.

3.4 Descripción de la aplicación web: HTML, JS y CSS

La aplicación web que el usuario utilizará para monitorizar la instalación/zona controlada está organizada gráficamente en forma de filas. La primera fila de la aplicación contiene la información relacionada con la medición de temperatura y la segunda fila contiene la información relacionada con el sensor de humedad. En una tercera fila se incluyen los botones y reguladores que controlan las salidas en la zona controlada. Hemos escogido un subconjunto de sensores y controles arbitrarios con tal de que las ideas sean fácilmente generalizables a otro número y tipo de sensores/actuadores.

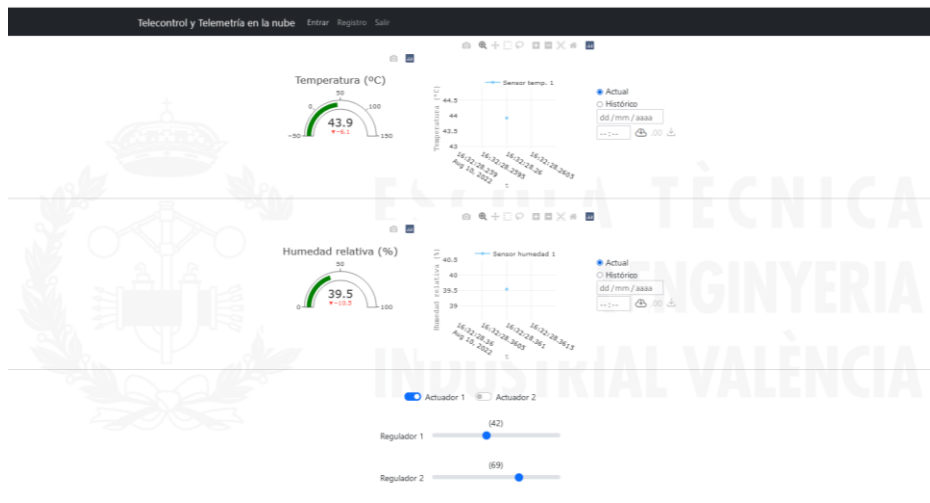


Figura 28: Vista general de la aplicación web

En cada fila de los sensores se incluyen 3 apartados principales: un reloj (o *gauge*, utilizando una terminología anglófona) que indica el valor de la última medición del sensor. A su derecha aparece una gráfica interactiva obtenida de la librería `plotly` [23] que añade automáticamente entradas para ir dibujando una representación temporal de las medidas. Más a la derecha aparece un menú selector donde puede conmutarse el comportamiento de la gráfica dinámica entre “Actual” o “Histórico”.

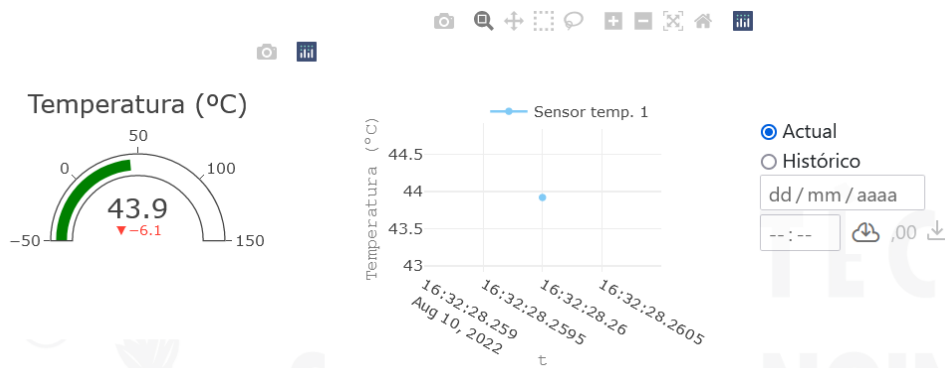


Figura 29: Detalle de la fila del sensor de temperatura

El modo “Actual” activa el funcionamiento antes descrito: el gráfico se va actualizando conforme se envían nuevas medidas y las va representando en un eje de tiempo. El modo “Histórico” activa un formulario adyacente que permite seleccionar un rango temporal de medidas de 60 minutos, con la fecha y hora de inicio indicada. Al pulsar en el icono de la *nube con la fecha* se seleccionan los valores en el tiempo seleccionado de la base de datos de la nube y se representan en el gráfico interactivo del centro. También incluye una opción para descargar un archivo CSV con los valores temporales seleccionados. En este archivo CSV puede personalizarse para que aparezca la coma o el punto como separador decimal (ya que suele ser un problema típico e incómodo de formato en algunas hojas de cálculo, con lo que permitimos que a priori se formateen adecuadamente con un punto o con una coma, el separador de decimales de los números reales correspondientes a las medidas).

La potente librería `Plotly` permite modificar la visualización de los datos de su gráfica, fácilmente se puede aumentar zoom o disminuir zoom, desplazarse entre la serie de datos y realizar una captura de pantalla entre otras opciones, siguiendo las pequeñas indicaciones pictóricas que se muestran en la parte superior de la propia gráfica.

En la tercera y cuarta fila de la web se encuentran los controles de los actuadores. Su posición también es ordenada siguiendo una malla, matriz o “*Grid*”, la primera fila para los interruptores de los actuadores 1 y 2 y las filas 2 y 3 para los reguladores 1 y 2 respectivamente.

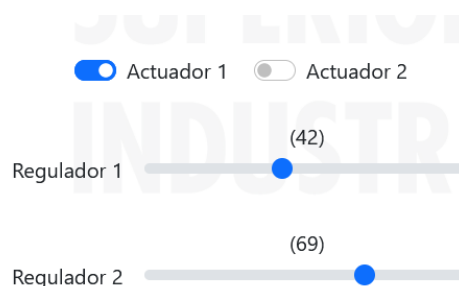


Figura 30: Actuadores

En la parte superior de la web se encuentra una barra de navegación con elementos como “Entrar” para iniciar sesión con la cuenta de correo registrada, “Registro” para añadir una nueva cuenta de correo a la base de datos de autenticaciones y “Salir” para cerrar sesión.

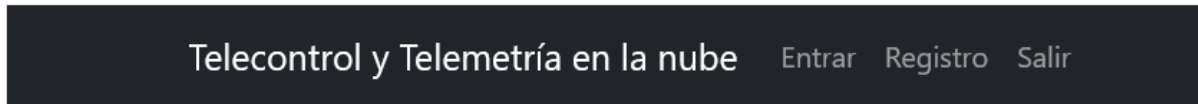


Figura 31: Barra de navegación

En los elementos “Entrar” y “Registro” invocan a dos ventanas modales con los formularios correspondientes con el correo y la contraseña. Esta información es dinámica, ya que si hemos “entrado” en la aplicación, ya no se muestra el ítem “Entrar”. Asimismo, también se añade el nombre del usuario para poder identificarse fácilmente.

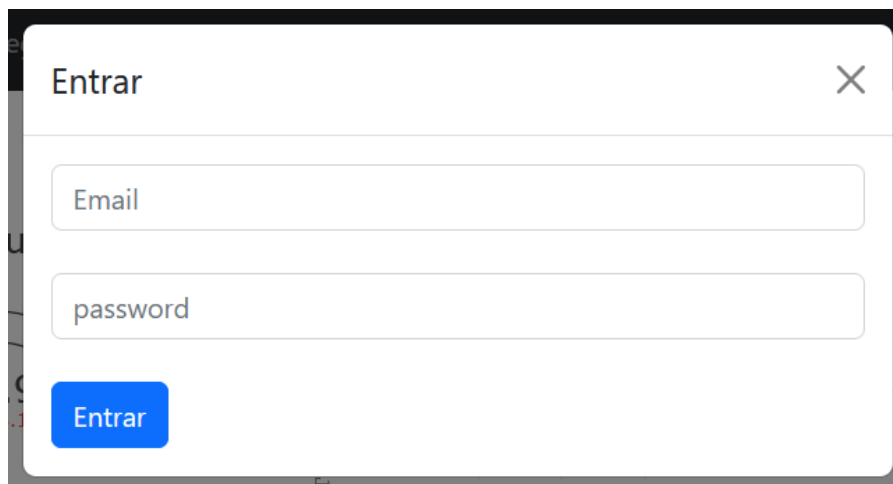


Figura 32: Modal “Entrar”

Por último, se ha añadido un fondo transparente en el que se puede ver el logo de la escuela ETSII y unos créditos desplegados con información sobre el creador de la web y otra información de interés.

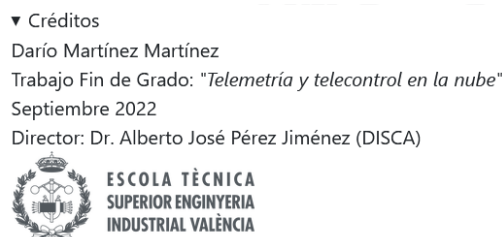


Figura 33: Créditos

3.5 Demostración de funcionamiento

Una vez completado todas las diferentes partes del proyecto se incluye una prueba demostrativa del correcto funcionamiento. Se emplea la función de alojamiento de proyectos web de GitHub para visualizar la web en internet mediante el siguiente enlace: <https://darioupv.github.io/AplicacionWeb/> .

Se puede comprobar, una vez puesto en marcha la zona controlada, la visualización de las medidas de los sensores en tiempo real y de los estados de los actuadores. Para entrar en la aplicación es posible registrarse con un correo electrónico y una contraseña o iniciar sesión con un correo de prueba llamado prueba1@prueba.com y la contraseña 123456.

Para comprobar las funciones de visualización de medidas históricas se registró datos en la fecha 13/08/2022 a las 00:00 durante una hora para tener datos que representar. Pulsando sobre los botones de descarga del archivo se puede comprobar el correcto funcionamiento de esta característica.

Toda la parte del proyecto basado en la aplicación web puede encontrarse en el repositorio público <https://github.com/DarioUPV/AplicacionWeb>.

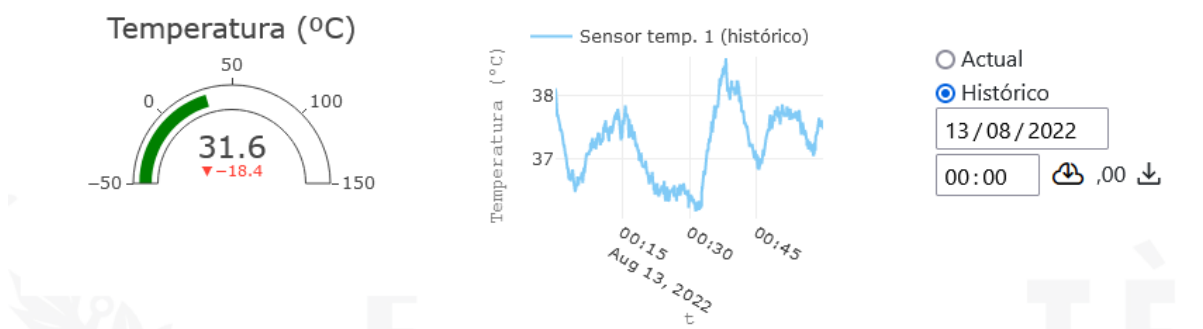


Figura 34: Demostración de funcionamiento

4. Conclusiones

Como se puede comprobar en las demostraciones de funcionamiento el sistema es válido para controlar y monitorizar actividades a distancia. Hemos diseñado el proyecto basado en una zona controlada formada por una Raspberry pi y sensores de temperatura y humedad y una matriz led que ejemplifica unos actuadores, la adición de una nube basada en el servicio Firebase añade funcionalidades como el guardado de medidas históricas para posterior representación y la autenticación, y por último se ha completado una aplicación web universalmente accesible basada en HTML, CSS y JavaScript para la visualización y monitorización de las variables de la zona controlada. La zona en que más tiempo se ha dedicado en este proyecto ha sido la creación de la aplicación web, sobre todo en la implementación de funcionalidades en JavaScript, pero para futuros trabajos es posible reutilizar gran parte de este proyecto ahorrando una considerable cantidad de tiempo. La versatilidad y facilidad del sistema anima a ser instalado en situaciones profesionales y domésticas, como un sistema de monitorización de producción de los paneles solares instalados en el tejado de una vivienda o incluso otras actividades de domótica. El sistema brinda la oportunidad de conocer y aprender la dinámica entre el mundo digital y el mundo físico, entre los sensores hasta los procesos del script de una forma accesible y de bajo coste.

La prueba de concepto es el primer paso para determinar la viabilidad de un sistema. Una vez queda demostrada la funcionalidad, el proyecto puede optar a futuras mejoras y expansiones que dejamos para futuros proyectos.

4.1 Ampliaciones y mejoras del proyecto

Un apartado que no se ha tenido en cuenta es la seguridad que necesitaría un sistema como el propuesto. Si se produce una conexión indeseada a la web de control alguien podría hacer un uso indebido de ella y/o acceder a información y datos en un principio privados. Como trabajo futuro se propone elaborar un sistema de protección y control de acceso a la base de datos de la nube.

Siguiendo la intención de crear un sistema fácilmente configurable se optó por utilizar un servicio en la nube de Firebase. Su sencillez, gratuidad y facilidad resultan atractivas, pero lo que es gratuito hoy tal vez no lo siga siendo en el futuro. También depender de una empresa externa para una nube/servidor tal vez sea motivo de desconfianza para el usuario. Para ello se propone como línea de trabajo futura la creación de un servidor web con almacenamiento propio, calibrando previamente la fiabilidad del sistema, internalizando esta parte del sistema.

Previendo que el control y la supervisión de la zona controlada puede llevarse a cabo con más de un usuario en la aplicación web se desarrolló una autenticación con correo y contraseña. Una de las opciones que se dejaron fuera del proyecto era los controles con diferentes grados de privilegios, que determinados controles solo sean accesibles a usuarios seleccionados o que variables sensibles solo puedan monitorizarse por usuarios determinados. Se propone como otra posible mejora a implementar en el futuro.

Otra posible ampliación podría ser, una vez que se haya proporcionado un control completo de autenticación, graficar igualmente en el dominio del tiempo la evolución de las señales de control por parte de los usuarios, ya que al llevar a cabo la autenticación, se pueden asignar responsabilidades sobre quién activó qué control en qué instante de tiempo.

5. Bibliografía y referencias

Referencias:

[1] Cuarta revolución industrial:

https://es.wikipedia.org/wiki/Revoluciones_industriales#Tercera_revoluci%C3%B3n_industrial

[2] Single board computers: https://en.wikipedia.org/wiki/Single-board_computer

[3] Empresas concienciadas con la energía y clima:

<https://www.leanpio.com/es/blog/las-empresas-espanolas-concienciadas-con-el-medio-ambiente>

[4] PLC: https://en.wikipedia.org/wiki/Programmable_logic_controller

[5] Pymes en España: <https://gdempresa.gesdocument.com/noticias/la-evolucion-de-las-pymes>

[6] Diferencias entre Arduino y Raspberry: <https://www.educba.com/raspberry-pi-vs-arduino/>

[7] Desarrollo de un controlador PID industrial de bajo coste mediante Raspberry Pi para control de temperatura. (Esperanza, 2015):

<https://riunet.upv.es/handle/10251/67637>

[8] Desarrollo de un sistema de localización y aplicación móvil para vehículos en aparcamientos (Héctor, 2016): <https://riunet.upv.es/handle/10251/88241>

- [9] Histórico de las veces que falla Firebase: <https://status.firebase.google.com/summary>
- [10] Historia de la Raspberry Pi: <https://linuxhint.com/raspberry-pi-history/>
- [11] Aplicaciones en la industria de la Raspberry pi: <https://www.raspberrypi.com/for-industry/>
- [12] Carcasas industriales para la Raspberry pi: <https://www.recantha.co.uk/blog/?p=17723>
- [13] Raspberry Pi a prueba de explosiones: <https://www.jeffgeerling.com/blog/2022/industrial-raspberry-pi-computers-one-explosion-proof>
- [14] Ordenadores SBC: <https://descubrearduino.com/sbc/>
- [15] Interfaz I2C: <https://wyldnetworks.com/blog/what-are-sensor-interfaces>
- [16] Interfaz SDI: <https://en.wikipedia.org/wiki/SDI-12>
- [17] Interfaz SPI: https://es.wikipedia.org/wiki/Serial_Peripheral_Interface
- [18] Librería sense-hat: <https://pythonhosted.org/sense-hat/>
- [19] Imager del SO de la Raspberry: <https://www.raspberrypi.com/software/>
- [20] Placa Sense Hat para la raspberry pi: <https://www.raspberrypi.com/products/sense-hat/>
- [21] Transformador recomendado por la Asociación Raspberry: <https://www.raspberrypi.com/products/raspberry-pi-universal-power-supply/>
- [22] Pyrebase: <https://github.com/thisbejim/Pyrebase>
- [23] Librería Plotly para JavaScript: <https://plotly.com/javascript/>

Bibliografía:

- Official Raspberry Pi Beginner's Guide (Gareth Halfacree, 2019)
Python 3 Curso Práctico (Cuevas Álvarez, Alberto, 2016)
Curso de Programación Python (Montejo Ráez, Arturo, 2019)

PRESUPUESTO

El objetivo del presupuesto es indicar el coste económico que requiere realizar el proyecto, el cual se incluye a continuación:

1. Cuadro de Mano de Obra

Código	Empleado	Salario Mensual (€/mes)	Salario (€/hora)
MO.01	Graduado Ingeniero de Tecnologías Industriales	3200	20

2. Cuadro de Materiales

Código	Ud	Nombre	Precio (€)	Rend.	Precio (€)
MT.01	u	Raspberry Pi Model B 3+	48.95	1	48.95
MT.02	u	Sense Hat	39.49	1	39.49
MT.03	u	Carcasa de protección	7.57	1	7.57
MT.04	u	Cable Ethernet 20 cm	1.3	1	1.3
		Precio Total			97.31

3. Cuadro de Maquinaria

Suponiendo una amortización de maquinaria de un período de un año, donde la vida útil estimada es de 2400 h/año, equivalente a 300 días laborales con un horario de 8 horas de trabajo.

Código	Unidades	Nombre	Total (€)
M.01	1	Laptop Lenovo IdeaPad 500	1000

4. Cuadro de Unidades de Obra

Parte 1: Creación de la aplicación web:

UO1-01		Diseño del contenido de la web			
Código	Ud	Descripción	Precio (€)	Rend.	Precio (€)
		Diseño de la estructura y los estilos de la aplicación web			
MO.01	h	Graduado Ingeniero de Tecnologías Industriales	20	40	800
M.01	h	Laptop Lenovo IdeaPad 500	0.42	40	16.8
	%	Costes directos complementarios		2	16.3
		Total unidad de obra			833.1

UO1-02		Diseño del funcionamiento de web			
Código	Ud	Descripción	Precio (€)	Rend.	Precio (€)
		Programación del funcionamiento y comunicación con la nube			
MO.01	h	Graduado Ingeniero de Tecnologías Industriales	20	160	3200
M.01	h	Laptop Lenovo IdeaPad 500	0.42	160	67.2
	%	Costes directos complementarios		2	65.34
		Total unidad de obra			3332.54

Parte 2: Configuración de la Nube en Firebase

UO1-01		Diseño del contenido de la web			
Código	Ud	Descripción	Precio (€)	Rend.	Precio (€)
		Crear de un proyecto en Firebase			
MO.01	h	Graduado Ingeniero de Tecnologías Industriales	20	15	300
M.01	h	Laptop Lenovo IdeaPad 500	0.42	15	6.3
	%	Costes directos complementarios			6.1
		Total unidad de obra			312.4

Parte 3: Instalación de la zona controlada.

UO3-01		Instalación de la Raspberry Pi			
Código	Ud	Descripción	Precio (€)	Rend.	Precio (€)
		Instalación de la RPI y programación de los scripts			
MO.01	h	Graduado Ingeniero de Tecnologías Industriales	20	85	1700
M.01	h	Laptop Lenovo IdeaPad 500	0.42	75	31.5
MT.01	u	Raspberry Pi Model B 3+	48.95	1	48.95
MT.02	u	Sense Hat	39.49	1	39.49
MT.03	u	Carcasa de protección	7.57	1	7.57
MT.04	u	Cable Ethernet 20 cm	1.3	1	1.3
	%	Costes directos complementarios		2	36.6
		Total unidad de obra			1865.4

5. Presupuesto Total

Código	Descripción	Subtotal (€)	Precio (€)
1	Creación de la aplicación web	4165.68	
2	Configuración de la Nube en Firebase	312.4	
3	Instalación de la zona controlada	1865.4	
	Presupuesto de Ejecución Material		6343.49
	Gastos generales 13%		824.65
	Beneficio industrial 6%		380.61
	Total antes de IVA		7548.76
	IVA 21%		1585.24
	Total presupuesto		9133.99

El importe del proyecto asciende a NUEVE MIL CIENTO TREINTA Y TRES CON NOVENTA Y NUEVE.

ANEXO

Se dispone en el siguiente anexo de una explicación del código empleado para crear la aplicación web con la descripción de los archivos HTML, CSS y JavaScript.

1 Ficheros HTML

La estructura principal de la web se define en un archivo html y los estilos visuales se definen en un archivo CSS. Vamos a proceder a explicar estos archivos de manera conjunta.

El archivo en el que se basa la web es el index.html y se compone de dos etiquetas fundamentales: `<head>` y `<body>`

En head añadimos el título de la página web a mostrar en la pestaña del navegador (línea 7). En esta sección se debe enlazar la web con los elementos para enriquecer la web, como lo son los estilos importados de la web bootstrap (línea 10), la importación de la librería de Plotly y todos sus elementos (línea 13) y los estilos propios de nuestro archivo CSS (línea 16). También se importan los estilos de varios iconos desarrollados por google (línea 17).

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>TeleCloud</title>
8
9   <!-- Bootstrap -->
10  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-0evHe/X
+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfhxawutUpFmBp4vmVor" crossorigin="anonymous">
11
12  <!-- Plotly -->
13  <script src="https://cdn.plot.ly/plotly-2.12.1.min.js"></script>
14
15  <!-- Mis estilos -->
16  <link href="estilos.css" rel="stylesheet">
17  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@48,400,0,0" />
18
19 </head>
20 > <body>...
311 </body>
312 </html>
```

En el *body* del archivo html hay varios elementos. Los más relevantes son: el contenedor `<div>` principal que engloba la estructura de la web de los sensores y actuadores y varias etiquetas `<script>` donde se enlaza la web con los varios scripts de JavaScript que aportan funcionalidad.

En las líneas 266, 268 y 311 se encuentran las funciones que cargan los códigos en JavaScript en la aplicación web. Más adelante se describirán el contenido de estos archivos.

```
265 <!-- Enlazando con JavaScript-->
266 <script src="relojes\_graficas.js"></script>
267
268 <script src='controlPagina.js'></script>
269
310 <!-- Código propio -->
311 <script src="comunicacionFB.js"></script>
312
```

Para el correcto funcionamiento de los componentes de la web reutilizados de Bootstrap se requiere escribir estas líneas de código (Líneas 274 y 275) siguiendo las instrucciones de la web de bootstrap: [<https://getbootstrap.com/docs/4.3/getting-started/introduction/>].

```
273 <!-- Scripts de Bootstrap -->
274 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.5/dist/umd/popper.min.js"
  integrity="sha384-Xe+8cL9oJa6tN/veChSP7q+mnSPaj5Bcu9mPX5F5xIGE0DVittaqT5lorf0EI7Vk"
  crossorigin="anonymous"></script>
275 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.min.js"
  integrity="sha384-kjU+l4N0Yf4ZOJErLsIcvOU2qSb74wXp0hqTvwVx30ElZRweTnQ6d31fXEoRD1Jy"
  crossorigin="anonymous"></script>
276
```

Del mismo modo, es necesario añadir los siguientes scripts para la correcta comunicación que permiten el acceso e inicialización de los componentes Firebase de la cual se crean los objetos correspondientes a la base de datos en tiempo real y la parte implementada de autenticación.

```

280 <!-- Scripts de Firebase -->
281 <!-- Insert this script at the bottom of the HTML, but before you use any Firebase services -->
282 <script src="https://www.gstatic.com/firebasejs/9.8.2/firebase-app-compat.js"></script>
283 <script src="https://www.gstatic.com/firebasejs/9.8.2/firebase-firestore-compat.js"></script>
284 <script src="https://www.gstatic.com/firebasejs/9.8.2/firebase-auth-compat.js"></script>
285 <script src="https://www.gstatic.com/firebasejs/9.8.2/firebase-database-compat.js"></script>
286
287
288 <script>
289
290   const firebaseConfig = {
291     apiKey: "AIzaSyBXrfGQtK6YQZ5hivip1jCiSsF8nqWz1Hw",
292     authDomain: "tele1-977c2.firebaseio.com",
293     databaseURL: "https://tele1-977c2-default-rtdb.europe-west1.firebaseio.com",
294     projectId: "tele1-977c2",
295     storageBucket: "tele1-977c2.appspot.com",
296     messagingSenderId: "1086367011073",
297     appId: "1:1086367011073:web:85c749e53e08874c9fcb00"
298   };
299
300   const firebaseApp = firebase.initializeApp( firebaseConfig );
301   //const fs = firebaseApp.firestore();
302   const db = firebaseApp.database();
303   const auth = firebaseApp.auth();
304
305 </script>

```

La estructura que forma la parte principal del contenido de la aplicación web se ha incluido en un contenedor `<div>` con el identificador `#principal`. En este contenedor se encuentran los gráficos de temperatura, humedad y los diferentes botones para los actuadores y alcanza desde la línea 87 hasta la línea 194.

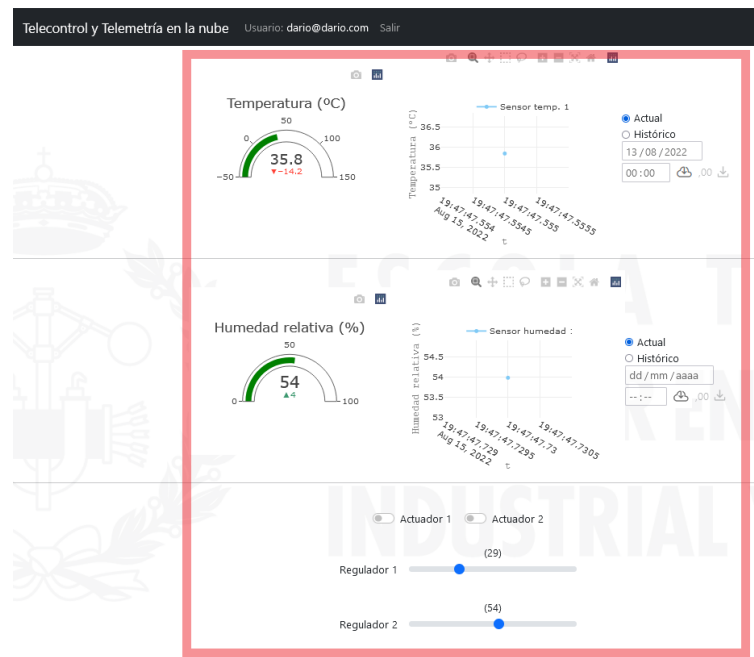


Figura 1: Contenedor `div` principal marcado en la aplicación web

```

87 <div id="principal" class="desaparecido">
88
89   <div class="flex-container" >
90     <div id='divReloj_temperatura' ></div>
91     <div id='divGrafica_temperatura' ></div>
92     <div id='divGrafica_temperatura_historico' class ="oculto"></div>
93
94 >   <div id='selector_temperatura'>...
120   </div>
121
122 </div>
123
124 <hr>
125
126   <div class="flex-container" >
127     <div id='divReloj_humedad' ></div>
128     <div id='divGrafica_humedad' ></div>
129     <div id='divGrafica_humedad_historico' class ="oculto"></div>
130
131 >   <div id='selector_humedad'>...
156   </div>
157 </div>
158
159 <hr>
160
161   <br>
162
163   <div class="flex-container-controles" id="controles">
164 >   <div class="form-check form-switch">...
167   </div>
168
169 >   <div class="form-check form-switch">...
172   </div>
173 </div>
174
175 <br>
176
177 > <div class="contenedor_grid" id="reg1">...
181 </div>
182
183 <br>
184
185 > <div class="contenedor_grid" id="reg2">...
190 </div>
191 <br>
192
193 <hr>
194 </div>

```

Dentro del `<div>` principal, desde la línea 89 hasta la 122 se escribe otro contenedor `<div>` que corresponde a la fila del sensor de temperatura como se aprecia resaltado en la imagen siguiente.

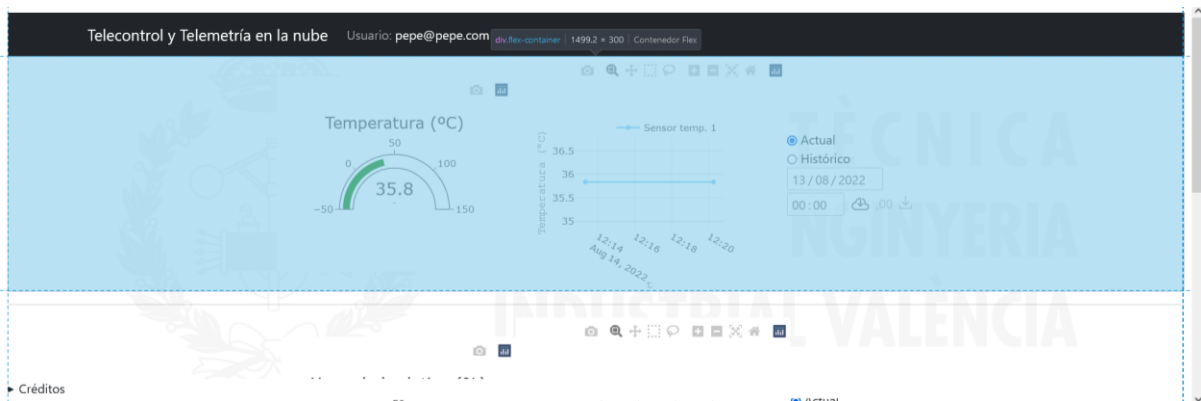


Figura 2: Contenedor div del sensor de temperatura

Los elementos que forman este <div> (el reloj, la gráfica y el menú selector de opciones) están separados y organizados por otros contenedores <div> (Líneas 90, 91, 92 y 94-120). Para cambiar la representación gráfica de las medidas actuales a las medidas históricas se crea otra gráfica dedicada a representar datos históricos y se mantiene oculta. Tras obtener las medidas históricas se elimina el atributo “oculto” a la gráfica histórica y se añade este atributo a la gráfica de datos actuales mediante programación en JavaScript. Por último, se incluye un contenedor <div> que incluye dos selectores tipo “radio” o circulares para cambiar entre datos actuales e históricos. Para seleccionar los datos históricos con su fecha y hora correspondiente se incorpora un formulario (Líneas 104-119) y se establece un botón con un logo de una nube como botón de “submit”.

```

89   <div class="flex-container" >
90     <div id='divReloj_temperatura' ></div>
91     <div id='divGrafica_temperatura' ></div>
92     <div id='divGrafica_temperatura_historico' class ="oculto"></div>
93
94     <div id='selector_temperatura'>
95
96       <input type="radio" id="temperatura_actual" name="sel_temperatura" value="actual" checked>
97       <label for="temperatura_actual">Actual</label>
98       <br>
99
100      <input type="radio" id="temperatura_historico" name="sel_temperatura" value="historico">
101      <label for="temperatura_historico">Histórico</label>
102      <br>
103
104      <form id="temperatura_dia_hora" class="form-actual-historico" >
105
106        <input type="date" id="dia_temperatura" name="dia" disabled required<<br>
107
108        <div class="flex-container">
109          <input type="time" id="hora_temperatura" name="hora" list="lista_horas" disabled required> &nbsp;
110          <button type="submit" id="boton_nube_temperatura" disabled>
111            <span class="material-symbols-outlined icono cloud">cloud_download</span>
112          </button>
113          &nbsp;
114          <span class="icono csv deshabilitado" id="punto_coma_decimal_csv_temperatura">,00</span>
115          &nbsp;
116          <span class="material-symbols-outlined icono csv deshabilitado" id="descargar_csv_temperatura">file_download</span>
117        </div>
118      </form>
119    </div>
120  </div>
121
122 </div>

```

Adicionalmente, también se añaden iconos mediante `` que indican la descarga de un fichero CSV con las medidas representadas en el gráfico histórico. A su lado, un icono para seleccionar el separador decimal entre el punto y la coma. Estos iconos empleados provienen de la librería de google “Material Symbols” [<https://fonts.google.com/icons>]

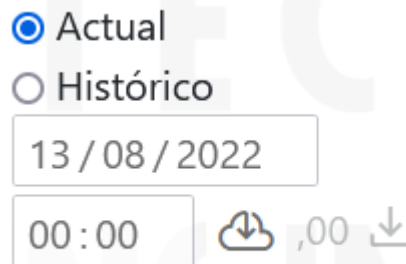


Figura 3: Detalle del contenedor div que contiene el selector

Para el sensor de la humedad y sus medidas se ha seguido la misma estructura y código html que la empleada para la magnitud de temperatura.

La última sección de la aplicación web es la fila de los actuadores, estructurada en tres contenedores `<div>`: uno para los actuadores tipo interruptores (Líneas 163-173) que contienen a “Actuador 1” y “Actuador 2” y otros dos para los reguladores con rango variable respectivamente (Líneas 177-181 y 185-190).

```

163 <div class="flex-container-controles" id="controles">
164   <div class="form-check form-switch">
165     <input class="form-check-input" type="checkbox" role="switch" id="actuador1" onclick="actuacion(this);">
166     <label class="form-check-label" for="flexSwitchCheckChecked">Actuador 1</label>
167   </div>
168
169   <div class="form-check form-switch">
170     <input class="form-check-input" type="checkbox" role="switch" id="actuador2" onclick="actuacion(this);">
171     <label class="form-check-label" for="flexSwitchCheckChecked">Actuador 2</label>
172   </div>
173 </div>
174
175 <div class="contenedor_grid" id="reg1">
176   <div></div> <span id="val_regulador_1"></span>
177   <label for="customRange1" class="form-label">Regulador 1</label>
178   <input type="range" class="form-range regulador" id="regulador1" onchange="regulador(this);" min="0" max="100">
179 </div>
180
181 <br>
182
183 <div class="contenedor_grid" id="reg2">
184   <div></div> <span id="val_regulador_2"></span>
185   <label for="customRange2" class="form-label">Regulador 2</label>
186   <input type="range" class="form-range regulador " id="regulador2" onchange="regulador(this);" min="0" max="100">
187 </div>
188
189
190 </div>

```

Una vez finalizada la explicación del contenido principal se describe brevemente el contenido secundario de la web. La barra de navegación en la parte superior de la página web se ha creado empleando los contenidos de Bootstrap siguiendo las indicaciones de la web <https://getbootstrap.com/docs/5.2/components/navbar/>. Únicamente es necesario introducir los componentes de navegación como “Entrar”, “Salir”, etc añadiendo “elementos de lista” o “List Item” con la etiqueta y los atributos indicados por Bootstrap.

```
49 <!-- Barra de Navegacion -->
50 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
51   <div class="container">
52     <a class="navbar-brand" href="#">Telecontrol y Telemetría en la nube</a>
53     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
54       aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
55       <span class="navbar-toggler-icon"></span>
56     </button>
57     <div class="collapse navbar-collapse" id="navbarNav">
58       <ul class="navbar-nav ml-auto">
59         <li class="nav-item" id="usuario_entrada">
60           <a class="nav-link" href="#" data-bs-toggle="modal" data-bs-target="#loginModal">Entrar</a>
61         </li>
62
63         <li class="nav-item" id="usuario_registro">
64           <a class="nav-link" href="#" data-bs-toggle="modal" data-bs-target="#signupModal">Registro</a>
65         </li>
66
67
68         <li class="nav-item oculto" id="usuario_nombre" >
69           <a class="nav-link" href="#" >Usuario: <span id="usuarioEmail"></span></a>
70         </li>
71
72         <li class="nav-item" id="usuario_salir">
73           <a class="nav-link" href="#" id="logout">Salir</a>
74         </li>
75
76
77       </ul>
78     </div>
79   </div>
80 </nav>
81
```

Del mismo modo que la barra de navegación, se incluyen unas ventanas flotantes o “modal” que aparecen cuando pulsamos sobre los elementos de la barra de navegación. Para completarlos se añaden títulos en el encabezado del modal y formularios en su cuerpo. Para la creación de estos elementos se ha seguido las indicaciones de la web <https://getbootstrap.com/docs/5.2/components/modal/>.

```

202 <!-- Modal Registrarse -->
203 <div class="modal fade" id="signupModal" tabindex="-1" aria-labelledby="exampleModallabel" aria-hidden="true">
204   <div class="modal-dialog">
205     <div class="modal-content">
206       <div class="modal-header">
207         <h5 class="modal-title" id="exampleModallabel">Registro</h5>
208         <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
209       </div>
210       <div class="modal-body">
211         <form id="signup-form">
212           <div class="form-group cajetin">
213             <input type="text" id="signup-email" class="form-control" placeholder="Email" required>
214           </div>
215           <div class="form-group cajetin">
216             <input type="password" id="signup-password" class="form-control" placeholder="password" required>
217           </div>
218           <button type="submit" class="btn btn-primary">Registro</button>
219         </form>
220       </div>
221     </div>
222   </div>
223 </div>
224
225
226 <!-- Modal Entrar-->
227 <div class="modal fade" id="loginModal" tabindex="-1" aria-labelledby="exampleModallabel" aria-hidden="true">
228   <div class="modal-dialog">
229     <div class="modal-content">
230       <div class="modal-header">
231         <h5 class="modal-title" id="exampleModallabel">Entrar</h5>
232         <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
233       </div>
234       <div class="modal-body">
235         <form id="login-form">
236           <div class="form-group">
237             <input type="text" id="login-email" class="form-control" placeholder="Email" required>
238           </div><br>
239           <div class="form-group">
240             <input type="password" id="login-password" class="form-control" placeholder="password" required><br>
241           </div>
242           <button type="submit" class="btn btn-primary">Entrar</button>
243         </form>
244       </div>
245     </div>
246   </div>
247 </div>

```

Para referenciar la Escuela Técnica Superior de Ingeniería Industrial se ha incluido un fondo con el logo de la escuela.

```

250 <!-- Escudo de la escuela de fondo-->
251 <div class="centrar">
252   
253 </div>

```

Por último para finalizar el archivo index.html se incluye un botón desplegable con datos de autoría y mención al tutor del trabajo final de grado.

```
255 <!-- Detalles desplegados -->
256 <details class="abajo" class="detalles">
257   <summary>Créditos</summary>
258   Darío Martínez Martínez <br>
259   Trabajo Fin de Grado: <i>"Telemetría y telecontrol en la nube"</i><br>
260   Septiembre 2022<br>
261   Director: Dr. Alberto José Pérez Jiménez (DISCA)<br>
262   
263 </details>
```

2 FICHEROS CSS

El archivo css es el responsable del estilo visual de los elementos de la web. A continuación se describen las propiedades CSS más relevantes.

Esta propiedad establece una opacidad del 5% y una anchura del 80% de la anchura total de la página, sea cual sea. Está aplicada al logo de la ETSII del fondo.

```
12  /* logo fondo central de la página web */
13  .logofondo{
14      width: 80vw;
15      opacity: 0.05;
16  }
```

La propiedad `contenedor_grid` modifica el espaciado de los elementos en su interior para que se distribuyan de una forma matricial. Se aplican a los contenedores de “Regulador 1” y “Regulador 2”.

```
18  /* contenedor grid para reguladores */
19  .contenedor_grid{
20      display: grid;
21      grid-template-columns: repeat(2, auto);
22      grid-template-rows: repeat(2, auto);
23      grid-row-gap: .1rem;
24      grid-column-gap: 1rem;
25      justify-items: center;
26      justify-content: center;
27  }
```

Esta propiedad CSS logra que un objeto se encuentre centrado en lo alto y ancho de la pantalla. Se aplica al logo del fondo de la ETSII.

```
29  /* centrado horizontal y vertical: para logo semitransparente ETSII */
30  .centrar{
31      margin: 0;
32      position: fixed;
33      top: 50%;
34      left: 50%;
35      -ms-transform: translate(-50%, -50%);
36      transform: translate(-50%, -50%);
37      z-index: -10;
38  }
```

Las propiedades css “desaparecido” y “oculto” son similares pero no iguales. La propiedad “desaparecido” se emplea para no mostrar el contenido principal mientras no se autentifica el usuario, mientras que la propiedad “oculto” esconde el contenedor de las gráficas actual o histórico según se pulse en el menú selector. La propiedad “visibility: hidden;” oculta a la vista el elemento al que se le aplica, pero el espacio que ocupa el elemento se mantiene

inalterado, por lo que aparece un hueco vacío. La propiedad “display: none;” no muestra el elemento y además elimina el espacio que ocuparía, con lo que no aparecen espacios vacíos.

```
50 /* oculto mientras no se autentique como usuario */
51 .desaparecido {
52 |     visibility: hidden;
53 | }
54
55 .oculto {
56 |     display:none;
57 | }
```

La propiedad “flex-container” es similar a “contenedor_grid” pues su objetivo es ordenar la posición de los elementos que contienen en el interior del elemento al que se le aplica. En este caso se organizan los elementos horizontalmente con “flex-direction: row;”. Con “flex-wrap: wrap” se consigue que si los elementos no caben en una misma fila en la pantalla, se disponen en una fila inferior. Esto es especialmente útil en los dispositivos móviles donde las pantallas son más estrechas y tienen menos espacio horizontalmente. “align-content” es para que la anchura de los elementos se adapte dinámicamente y ocupe el espacio horizontal disponible. El resto de atributos son para centrar los elementos a lo alto y a lo ancho.

Esta propiedad está aplicada al contenedor que incluye los relojes y gráficos de temperatura y humedad y al conjunto del selector de hora e iconos de la nube, descarga y separador decimal.

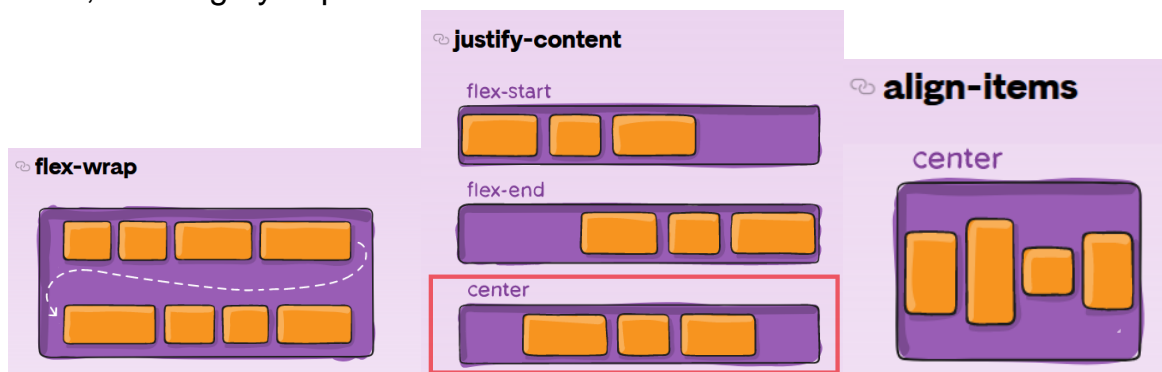


Figura 4: Ejemplos de la propiedad flex-container. Fuente: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

```
87  /* contenedor de tipo flex para organización de elementos */
88  .flex-container {
89      display: flex;
90      flex-direction: row;
91      flex-wrap: wrap;
92      justify-content: center;
93      align-items: center;
94      align-content: stretch;
95  }
```

La propiedad “flex-container-controles” es idéntica a “flex-container” excepto que añade un espacio “gap” de 20 píxeles que envuelve los elementos.

```
97  /* contenedor de tipo flex para los controles on-off de actuadores */
98  .flex-container-controles {
99      display: flex;
100     flex-direction: row;
101     flex-wrap: wrap;
102     justify-content: center;
103     align-items: center;
104     align-content: stretch;
105     gap: 20px;
106 }
```


3 FICHEROS JavaScript

El código javascript de la aplicación web se ha repartido en los siguientes ficheros:

1. `controlPagina.js`: funciones relacionadas con la interacción con la página de la aplicación web.
2. `comunicacionFB.js`: funciones relacionadas con la comunicación bidireccional con Firebase.
3. `relojes_graficas.js`: funciones relacionadas con el dibujo de los relojes y las gráficas de las magnitudes consideradas.

Fichero `controlPagina.js`:

Contiene todo el software que me permite interactuar con la página de la aplicación. El software asociado a la exposición de la temperatura es idéntico al de la humedad, por lo que simplemente indicando con un parámetro si nos referimos a una magnitud u otra se ha conseguido generalizar el software, consiguiendo con ello una fiabilidad mejor.

La primera labor realizada es la programación de los *listeners* que se van a encargar de controlar la graficación de los datos actuales (desde que arrancó la aplicación web hasta el momento actual) o de datos pertenecientes a un histórico (datos recogidos durante 60 minutos, durante la fecha seleccionada y la hora indicada). Los respectivos *listeners* (para la temperatura y para la humedad) son arrancados con las invocaciones



```
32  activar_listeners_actual_historico("temperatura");  
33  activar_listeners_actual_historico("humedad");
```


Las dos siguientes funciones simplemente llevan a cabo la conmutación funcional entre la graficación de los datos actuales o los datos históricos, habilitando y deshabilitando la posibilidad de interactuar con las mismas funciones `activar_historico` y `desactivar_historico`.

Téngase en cuenta que mientras que las gráficas pueden corresponderse a un intervalo actual o a un intervalo histórico, los relojes siempre muestran los valores actuales instantáneos.

Los tiempos que manejamos están referenciados a tiempo UNIX (cantidad de milisegundos, o microsegundos o segundos, según el caso, transcurridos desde el 1 de enero de 1970). JavaScript permite obtener el tiempo UNIX a

partir de la creación de un objeto `Date()` y el uso del método `getTime()`. Para nuestra comodidad, codificaremos el tiempo en unidades de segundos transcurridos desde el 1 de enero de 1970.

Los iconos de descarga desde la nube , y de la descarga local , son iconos de Google con criterios de diseño Material utilizados para indicar la descarga desde Firebase a la aplicación web del histórico tras lo cual se mostrará su graficación. Con la descarga local, se descarga los valores en un fichero CSV; adicionalmente se muestra como icono el símbolo **,00** o **.00**, seleccionando con ello un formato numérico con una coma como separador decimal o un punto respectivamente, para facilitar la interpretación numérica de los resultados de hojas de cálculo tipo Excel o LibreOffice, por ejemplo.

El icono de descarga desde la nube  está asociado a la acción `submit` de un formulario para que controle automáticamente los valores requeridos obligatoriamente de la fecha y hora del histórico. Por lo tanto, cuando se cliquen, será invocada la función `traer_desde_nube(<MAGNITUD>)`, donde `<MAGNITUD>` es el string `"temperatura"` o `"humedad"` (intentando generalizar las funciones), donde a partir de la fecha y hora indicada en los selectores temporales se calcula el tiempo UNIX mínimo con el que buscar resultados en la base de datos de Firebase y sumándole 3600 segundos (constante `intervalo_descarga`) obtendremos el tiempo máximo para definir el intervalo de búsqueda.

Con la instrucción `let valores = db.ref(...)`

```
112 | let valores = db.ref('historicos/sensores/' + medida).orderByChild("tiempo").startAt(tmin).endAt(tmax)
```

seleccionamos todos valores implicados y con el método `once`, asíncronamente se reciben. En el código, el parámetro `snap` recoge los resultados y si la aplicación del método `val()` al mismo nos proporciona un valor nulo es porque no existen registros con las características temporales indicadas. En caso contrario, se recorre el array con todos los resultados reuniéndolos y fabricando un string que representará el fichero CSV; `ficheroCSV_prov` es el nombre de la variable que representa a dicho *string*. El string se va formando con todas las entradas del fichero CSV respetando las separaciones necesarias entre las columnas, eligiendo el carácter `“;”` (punto y coma) como carácter separador, así como la separación entre filas, añadiendo un carácter de fin de línea.

Paralelamente a la formación provisional del fichero CSV se “apaga” la gráfica correspondiente a la situación temporal actual y se conmuta a una nueva, dibujando los datos recibidos:

```
154 | document.querySelector("#divGrafica_" + medida).classList.add("oculto");
155 | document.querySelector("#divGrafica_" + medida + "_historico").classList.remove("oculto");
156 | Plotly.newPlot("divGrafica_" + medida + "_historico", data, layout2, config2);
```

Obsérvese que con la aparición de la gráfica del histórico, no se borra la de los datos actuales, simplemente se oculta. Sendas gráficas ocupan sendos divs cuyos ids respectivos son `divGrafica_temperatura` y `divGrafica_temperatura_historico`, para el caso de la magnitud temperatura.

Adicionalmente se realizan los controles oportunos para que la interfaz tenga un aspecto y comportamiento coherente con el estado al que se llega.

Quedan dos aspectos pendientes y son la elección de punto o coma como separador decimal del fichero CSV y propiamente la descarga del fichero CSV.

Por defecto, al ir creando el string con el contenido del fichero CSV los números tendrá un formato con un punto decimal como separador; solo en el caso de que queramos que sea una coma dicho separador, entonces aplicaremos un método a todo el string con el que reemplazamos cualquier aparición del punto por una coma (método `replaceAll()`). Dada la sintaxis empleada no hay posibilidad de ambigüedad en la elección de esta estrategia de conversión de formatos:

```
158 | ficheroCSV = (comaCSV)? ficheroCSV_prov.replaceAll(".", ",") : ficheroCSV_prov;
```

Por último, la descarga se lleva a cabo añadiendo un elemento de tipo ancla al DOM (Document Object Model: estructura o perspectiva en forma de objeto del propio fichero HTML) con el ajuste siguiente con cuanto a los atributos de la misma,

```
172 | element.setAttribute('href', 'data:text/plain;charset=utf-8,' + encodeURIComponent(text));
173 | element.setAttribute('download', filename);
```

de modo que al clicar automáticamente el ancla sin necesidad de intervención humana (`element.click()`), provoca la descarga dada la presencia del atributo `download` (automáticamente el ancla se elimina del DOM con `document.body.removeChild(element)`):

Fichero comunicacionFB.js:

En este fichero hemos incluido prácticamente todas las funciones JavaScript que hacen referencia de alguna manera a Firebase.

Se ha implementado una parte mínima de autenticación para hacer operativo el uso de Firebase. Es una parte realmente importante de todo sistema de comunicación y se deja como línea futura de trabajo abordarla en toda su dimensionalidad.

La parte que hemos considerado mínima para que el sistema sea operativo y pueda mostrar la viabilidad de la prueba de concepto consiste en autenticarse mediante un usuario (sugerimos que sea un correo electrónico por si en el futuro se desea realizar algún tipo de realimentación de información a través de esta vía) y una contraseña. Una vez conseguida la autenticación, tendremos acceso a toda la base de datos. Este es otro aspecto que hay que cubrir cuando se contemple plenamente la parte de seguridad y autenticación de usuarios.

Para que previamente el sistema conozca el nombre del usuario y la contraseña se ha establecido un mecanismo de prueba con el que se añaden estos datos, denominándolo **Registro**. Una vez registrado un par usuario/contraseña ya se puede acceder al sistema mediante la acción **Entrada**.

En primera instancia se ajusta la persistencia de la autenticación a la sesión; es decir, si se finaliza la sesión (se cierra el navegador, la pestaña o se recarga la página), la autenticación debería perderse.

Se procede de la siguiente forma al acceder al Registro, cuando es clicado el ítem correspondiente al menú principal: se muestra el código correspondiente a la adición del listener del evento submit del modal del registro

```
signupForm.addEventListener("submit", ...
```

. La intención es crear de manera formal en la base de datos de Firebase las entradas correspondientes a un usuario (su email) junto con su contraseña. Estos datos son leídos desde el modal correspondiente y añadidos al método `createUserWithEmailAndPassword()` del objeto `auth` creado en el fichero `index.html` (accediendo a la parte de autenticación de Firebase).

A continuación, se muestra el código asociado al menú **Entrada** mediante la acción de envío o *submit* de un formulario correspondiente a la ventana modal

correspondiente al acto de *login* o Entrada; se lleva a cabo la lectura del contenido del cajetín de usuario (correo electrónico tal y como hemos sugerido anteriormente) y de la contraseña y se envía a Firebase para verificar su corrección. Para ello emplearemos el método

`signInWithEmailAndPassword()` del objeto `auth` (proveniente del acceso a las facilidades de autenticación de Firebase, tal y como se puede observar en el fichero `index.html`). Si se cumple la *promesa* (método *then*) entonces el usuario está correctamente autenticado, se oculta el modal de Entrada y se activan todas las partes gráficas correspondientes ya a la interacción con las medidas y controles de la planta. En caso de que haya un error en el usuario/email o la contraseña saldrá una ventana de alerta indicándonos el suceso.

Por lo tanto mediante la primera ventana modal (**Registro**) incluimos los nuevos usuarios y mediante la segunda (**Entrada**) cualquier usuario previamente registrado accede a todos los recursos de la aplicación. Cuando la parte de autenticación esté plenamente resuelta, también podrá abordarse sin fisuras la parte de registro, ya que en principio, debería existir una única autoridad encargada de llevar a cabo el registro de los usuarios autorizados.

El menú principal es dinámico por lo que en función del estado en el que se encuentre la relación entre el usuario y la app aparecerán unos ítems u otros. Del mismo modo, cuando un usuario esté en el sistema, aparecerá su nombre en el menú principal.

Cuando el usuario finalice la sesión, al pulsar **Salir** se lleva a cabo el proceso de desenlace con la base de datos y la autenticación, tal y como se muestra en el código asociado al evento de clicar la citada opción en el menú principal.

Quedan por describir cómo se llevan a cabo las acciones de actualización de los indicadores o medidores o gráficas de la aplicación web con el estado de la base de datos en Firebase, así como las acciones oportunas a realizar cuando en la aplicación web modificamos algún control o regulador y cómo se refleja ello en la base de datos de Firebase.

Los cambios que se produzcan como consecuencia de la interacción del usuario con la aplicación web, deberán ser enviados a Firebase para que se almacenen oportunamente y tengan el reflejo oportuno en la planta a controlar (Raspberry Pi en nuestro caso). Para ello es necesario referenciar el nodo en el que queremos que se refleje el cambio y después aplicar el método `set`. Por ejemplo, el siguiente código envía a Firebase un cambio en algún actuador:

```

135 // Accion sobre actuadores: indicación a FireBase
136 function actuacion(checkbox) {
137     if (checkbox.id=="actuador1") {
138         console.log("actuador 1: " + checkbox.checked);
139         db.ref('actuadores/actuador1').set(checkbox.checked)
140     } else {
141         console.log("actuador 2: " + checkbox.checked);
142         db.ref('actuadores/actuador2').set(checkbox.checked)
143     }
144 }

```

Adicionalmente, ya que planteamos una estructura potencialmente jerárquica de estos controles, deseamos que si un valor es modificador en la app (actuadores o control) esto se vea reflejado, no solo en la app del usuario que ha provocado la modificación, si no en el resto también, de manera que se dé la sensación de control replicado, en aquellas aplicaciones en las que así interese.

Para atender a los cambios que se proponen desde la aplicación web respecto a los actuadores o reguladores, se hace referencia al *nodo* en cuestión dentro de la base de datos (camino identificativo dentro de la base de datos desde la raíz de la misma hasta el punto que contiene la información en el que se está interesado). Para ello se programan acciones asíncronas, de modo que cuando haya alguna alteración se dispare la acción oportuna. Por ejemplo:

```

80 db.ref('actuadores/actuador1').on('value', (instantanea) => {
81     const valor = instantanea.val();
82     document.querySelector("#actuador1").checked = valor;
83 })

```

Con las anteriores instrucciones, cuando el 'valor' del nodo `actuadores/actuador1` varíe (es decir, varíe cualquier valor que dependa jerárquicamente de él), entonces se ejecutará la función indicada en el segundo parámetro del método `on`, extrayendo el nuevo valor a partir del método `val()` del parámetro `instantanea`.

Se procedería de forma análoga con los reguladores.

Fichero `relojes_graficas.js`:

En este fichero se describen los parámetros que permitirán el dibujo de las curvas así como el dibujo de los valores instantáneos en los relojes.

Se ha estudiado la API JavaScript de Plotly [<https://plotly.com/javascript/>] y se han configurado los parámetros:

- id del elemento `DIV` de `HTML` en el que se dibujará la gráfica. En nuestro caso tenemos cuatro posibilidades:
 - Para temperatura:
 - `divReloj_temperatura`
 - `divReloj_temperatura_historico`
 - Para humedad:
 - `divReloj_humedad`
 - `divReloj_humedad_historico`
- Datos a mostrar en relojes o graficar de forma acumulativa
- `layout`: dimensiones, ejes, etc.
- configuración: responsividad del gráfico, herramientas, ...

Estos datos se configuran adecuadamente en función de si estamos haciendo referencia al reloj que nos muestra el valor instantáneo o la gráfica que nos muestra los valores en función del tiempo.

En las funciones `actualizar_indicador()` y `actualizar_grafica()` se actualizan respectivamente los valores instantáneos de los relojes y de las gráficas respectivamente. Podemos observar que se utilizan los métodos `Plotly.animate()` y `Plotly.extendTraces()` para llevar a cabo estas acciones respectivamente.

A continuación se incluye la totalidad del código en javascript de los tres ficheros descritos.

1. controlPagina.js

```
13 // Conmutación actual / histórico
14
15 const activar_listeners_actual_historico = function (medida) {
16     let selectores = document.querySelectorAll('input[name="sel_" + medida + "']');
17     for (let i=0; i< selectores.length; i++) {
18         selectores[i].addEventListener("click", function() {
19             let val = this.value;
20             if (val=="historico") {
21                 activar_historico(medida);
22             } else {
23                 desactivar_historico(medida);
24                 document.querySelector("#divGrafica_" + medida).classList.remove("oculto");
25                 document.querySelector("#divGrafica_" + medida + "_historico").classList.add("oculto");
26             }
27         });
28     }
29 }
30
31
32
33
34
35
36 // Activa el historico: invocado en activar_listeners_actual_historico
37 const activar_historico = function(medida) {
38     document.querySelector("#dia_" + medida).removeAttribute("disabled");
39     document.querySelector("#hora_" + medida).removeAttribute("disabled");
40     document.querySelector("#boton_nube_" + medida).removeAttribute("disabled");
41
42     let hay_csv = (medida=="temperatura")? hay_csv_temperatura : hay_csv_humedad;
43     if (hay_csv) {
44         document.querySelector("#punto_coma_decimal_csv_" + medida).classList.remove("deshabilitado");
45         document.querySelector("#descargar_csv_" + medida).classList.remove("deshabilitado");
46         document.querySelector("#divGrafica_" + medida).classList.add("oculto");
47         document.querySelector("#divGrafica_" + medida + "_historico").classList.remove("oculto");
48     }
49 }
50
51 // Desactiva historico: activa el actual: invocado en activar_listeners_actual_historico
52 const desactivar_historico = function(medida) {
53     document.querySelector("#dia_" + medida).setAttribute("disabled", "");
54     document.querySelector("#hora_" + medida).setAttribute("disabled", "");
55     document.querySelector("#boton_nube_" + medida).setAttribute("disabled", "");
56     document.querySelector("#punto_coma_decimal_csv_" + medida).classList.add("deshabilitado");
57     document.querySelector("#descargar_csv_" + medida).classList.add("deshabilitado");
58     document.querySelector("#divGrafica_" + medida).classList.remove("oculto");
59     document.querySelector("#divGrafica_" + medida + "_historico").classList.add("oculto");
60 }
```



```

66 // Descarga de muestras desde la nube
67
68
69 // Devuelve tiempo UNIX en segundos
70 function tiempoUNIXs(dia, hora) {
71     let fecha = new Date( dia + " " + hora + ":00" );
72     return Math.floor(fecha.getTime()/1000);
73 }
74
75 // Variables para fichero CSV de medidas
76 const SEP = ","; // separador de columnas
77 const EOL = "\n"; // separador de filas: nueva línea
78 let ficheroCSV_prov=""; // contenido de fichero CSV
79 let comaCSV = true; // números con coma (true) o punto (false) decimal
80 const intervalo_descarga = 3600; // segundos de medidas a partir de la hora indicada de descarga
81
82 // Si ya se ha bajado algún fichero CSV de medidas
83 let hay_csv_temperatura = false;
84 let hay_csv_humedad = false;
85
86
87 // Listeners para descargar medidas desde la nube
88 const cargarTemperaturaForm = document.querySelector("#temperatura_dia_hora");
89 const cargarHumedadForm = document.querySelector("#humedad_dia_hora");
90
91 cargarTemperaturaForm.addEventListener("submit", (e) => {
92     e.preventDefault();
93     traer_desde_nube("temperatura");
94 });
95 cargarHumedadForm.addEventListener("submit", (e) => {
96     e.preventDefault();
97     traer_desde_nube("humedad");
98 });

```

```

101 // baja un fichero de medidas de la nube en formato CSV
102 function traer_desde_nube(medida){
103     // hora y día de la medida
104     let dia = document.querySelector("#dia_" + medida).value;
105     let hora = document.querySelector("#hora_" + medida).value;
106     // tiempo UNIX inicial de medidas
107     tmin = tiempoUNIXs(dia, hora);
108     // tiempo UNIX final de medidas: inicial más 1 hora (3600 s: variable intervalo_descarga)
109     tmax = tmin + intervalo_descarga;
110     //console.log("dia: #" + dia + "#, hora: $" + hora + "$" + "tmin: " + tmin + ", tmax: " + tmax);
111     // Búsqueda en la base de datos de Firebase: medidas comprendidas entre tmin y tmax
112     let valores = db.ref('historicos/sensores/' + medida).orderByChild("tiempo").startAt(tmin).endAt(tmax)
113     valores.once("value", (snap) => {
114         if (snap.val()==null) {
115             alert("No hay registros con los datos temporales " + dia + " " + hora);
116             return;
117         }
118         ficheroCSV_prov = "tUNIX" + SEP + "fecha" + SEP + "hora" + SEP + "medida" + SEP + "unidades" + EOL;
119         let y = [];
120         let x = [];
121         // creación del fichero CSV

```

```

121 // creación del fichero CSV
122 snap.forEach( (childSnapshot)->{
123     var childKey = childSnapshot.key;
124     var childData = childSnapshot.val();
125     console.log("childKey: " + childKey);
126     console.log("childData: " + childData);
127     console.log("tiempo: " + childData.tiempo);
128     console.log("valor:" + childData.valor);
129     console.log("unidades:" + childData.unidades);
130     let tiempo = new Date(childData.tiempo * 1000) // tiempo UNIX en ms
131     let fecha = tiempo.getDate()+"/"+(tiempo.getMonth()+1)+"/"+tiempo.getFullYear(); // (D)D/(M)M/AAAA
132     let instante = tiempo.getHours() + ":" + tiempo.getMinutes() + ":" + tiempo.getSeconds() + "." + tiempo.getMilliseconds(); // (h)h:(m)m:(s)s.sss
133     let unidades = childData.unidades;
134
135     ficheroCSV_prov += childData.tiempo + SEP + fecha + SEP + instante + SEP + childData.valor + SEP + unidades + EOL;
136     y.push(childData.valor);
137     x.push(tiempo);
138     switch(medida) {
139         case "temperatura": hay_csv_temperatura = true; break;
140         case "humedad":     hay_csv_humedad     = true; break;
141     }
142 });
143
144 var data = [{
145     x: x,
146     y: y,
147     mode: 'scatter',
148     line: {color: '#80CAF6'}
149 }];
150 switch(medida) {
151     case "temperatura": data[0].name='Sensor temp. 1 (histórico)'; break;
152     case "humedad":     data[0].name='Sensor humedad 1 (histórico)'; break;
153 }
154 document.querySelector("#divGrafica_" + medida).classList.add("oculto");
155 document.querySelector("#divGrafica_" + medida + "_historico").classList.remove("oculto");
156 Plotly.newPlot("divGrafica_" + medida + "_historico", data, layout2, config2);
157
158 document.querySelector("#descargar_csv_" + medida).classList.remove("deshabilitado");
159 document.querySelector("#punto_coma_decimal_csv_" + medida).classList.remove("deshabilitado");
160
161 //console.log(ficheroCSV);
162 });
163 } // traer_desde_nube(medida)

```

```

167 // Simula clicado sobre un elemento artificialmente creado
168 function descargarCSV(filename, ficheroCSV) {
169     const text = (comaCSV)? ficheroCSV.replaceAll(".", ",") : ficheroCSV;
170     const element = document.createElement('a');
171     element.setAttribute('href', 'data:text/plain;charset=utf-8,' + encodeURIComponent(text));
172     element.setAttribute('download', filename);
173
174     element.style.display = 'none';
175     document.body.appendChild(element);
176
177     element.click();
178
179     document.body.removeChild(element);
180 }
181
182
183 // Descarga de fichero de temperatura "temperatura.csv"
184 let descargar_CSV_temperatura = document.querySelector("#descargar_csv_temperatura");
185 descargar_CSV_temperatura.addEventListener("click", (e) => {
186     if (descargar_CSV_temperatura.classList.contains("deshabilitado")) return;
187     descargarCSV("temperatura.csv", ficheroCSV_prov);
188 });
189
190 // Descarga de fichero de humedad "humedad.csv"
191 let descargar_CSV_humedad = document.querySelector("#descargar_csv_humedad");
192 descargar_CSV_humedad.addEventListener("click", (e) => {
193     if (descargar_CSV_humedad.classList.contains("deshabilitado")) return;
194     descargarCSV("humedad.csv", ficheroCSV_prov);
195 });

```

```

203 // Control de punto o coma decimal en números del fichero CSV
204
205 let punto_coma_decimal_csv_temperatura = document.querySelector("#punto_coma_decimal_csv_temperatura");
206 let punto_coma_decimal_csv_humedad = document.querySelector("#punto_coma_decimal_csv_humedad");
207
208
209 function comaPuntoCSV(medida) {
210     let punto_coma_decimal_csv = document.querySelector("#punto_coma_decimal_csv_" + medida);
211     if (punto_coma_decimal_csv.classList.contains("deshabilitado")) return;
212     if (comaCSV) {
213         punto_coma_decimal_csv_temperatura.innerHTML=".00";
214         punto_coma_decimal_csv_humedad.innerHTML=".00";
215     } else {
216         punto_coma_decimal_csv_temperatura.innerHTML=",00";
217         punto_coma_decimal_csv_humedad.innerHTML=",00";
218     }
219     comaCSV = !comaCSV;
220 }
221
222 punto_coma_decimal_csv_temperatura.addEventListener("click", (e) => {
223     comaPuntoCSV("temperatura")
224 })
225
226 punto_coma_decimal_csv_humedad.addEventListener("click", (e) => {
227     comaPuntoCSV("humedad")
228 })

```

2. comunicacionFB.js

```

4 // Se pierde la autenticacion cuando se cierra la sesión
5 auth.setPersistence(firebase.auth.Auth.Persistence.SESSION)
-
9 // Modal del Sign-up o registro
10 const mySignUpModal = new bootstrap.Modal('#signupModal', {keyboard:true, focus:true});
11 const signupForm = document.querySelector("#signup-form");
12 signupForm.addEventListener("submit", (e)=>{
13     e.preventDefault();
14     const email = document.querySelector("#signup-email").value;
15     const password = document.querySelector("#signup-password").value;
16     console.log("enviando datos")
17     auth.createUserWithEmailAndPassword(email, password)
18     .then(userCredential => {
19         signupForm.reset()
20         mySignUpModal.hide();
21         console.log("OK sign up")
22     })
23
24 })

```

```

26 // Identificador
27 let usuarioEmail;
28
29 // Modal del login o entrada
30 const myLoginModal = new bootstrap.Modal('#loginModal', {keyboard:true, focus:true});
31 const loginForm = document.querySelector("#login-form");
32 loginForm.addEventListener('submit', e=>{
33     e.preventDefault();
34     const email = document.querySelector("#login-email").value;
35     usuarioEmail = email;
36     const password = document.querySelector("#login-password").value;
37     auth.signInWithEmailAndPassword(email, password)
38         .then(userCredential => {
39             loginForm.reset();
40             myLoginModal.hide();
41             document.querySelector("#principal").classList.remove("desaparecido");
42             console.log("OK login")
43             document.querySelector("#usuario_entrada").classList.add("oculto");
44             document.querySelector("#usuario_registro").classList.add("oculto");
45             document.querySelector("#usuario_nombre").classList.remove("oculto");
46             document.querySelector("#usuarioEmail").innerHTML = usuarioEmail;
47         }).catch((e)=>{
48             alert("Error en email/password");
49         })
50 })
51
54 // Logout: se eliminar la autenticación, contacto con FB y recarga ventana
55 const logout = document.querySelector("#logout")
56 logout.addEventListener('click', e=>{
57     e.preventDefault();
58     auth.signOut().then(()=>{
59         console.log("signed out")
60         db.goOffline();
61         window.location.reload();
62     })
63 })
64
69 // cambio en el estatus del usuario
70 auth.onAuthStateChanged(user => {
71     if (user) { // autenticación OK
72         //usuario = user;
73         actualizacionAutomaticaTelemetria(); // se pide actualización de telemetría
74         actualizacionAutomaticaTeleControl(); // se pide actualización de telecontrol
75         console.log("auth: sign in")
76     } else {
77         console.log("auth: sign out")
78     }
79 })

```

```

84 // Actualización automática de telecontrol: cualquier modificación en la base de datos
85 // se verá reflejada localmente, haya sido este usuario el promotor de dicho cambio o no.
86 function actualizacionAutomaticaTeleControl() {
87     // Actualización del actuador 1
88     db.ref('actuadores/actuador1').on('value', (instantanea) => {
89         const valor = instantanea.val();
90         document.querySelector("#actuador1").checked = valor;
91     })
92     // Actualización del actuador 2
93     db.ref('actuadores/actuador2').on('value', (instantanea) => {
94         const valor = instantanea.val();
95         document.querySelector("#actuador2").checked = valor;
96     })
97     // Actualización del regulador 1
98     db.ref('actuadores/regulador1').on('value', (instantanea) => {
99         const valor = instantanea.val();
100        document.querySelector("#regulador1").value = valor;
101        document.querySelector("#val_regulador_1").innerHTML = ("+"+valor+");
102
103    })
104    // Actualización del regulador 2
105    db.ref('actuadores/regulador2').on('value', (instantanea) => {
106        const valor = instantanea.val();
107        document.querySelector("#regulador2").value = valor;
108        document.querySelector("#val_regulador_2").innerHTML = ("+"+valor+");
109    })
110 }

113 // Programación de actualización automática de telemetría
114 function actualizacionAutomaticaTelemetria() {
115
116     // Actualización de temperatura: tanto reloj como gráfica de valores actuales
117     db.ref('sensores/temperatura').on('value', (instantanea) => {
118         const temp = instantanea.val();
119
120         let dato = +(temp.valor).toFixed(2);
121         actualizar_indicador(indicador_temperatura, null, dato, 'divReloj_temperatura', layout1, config1);
122
123         actualizar_grafica('divGrafica_temperatura', new Date(), dato);
124     })
125
126     //db.ref('sensores/presion').on('value', (instantanea) => {
127     // const press = instantanea.val();
128     //})
129
130     // Actualización de humedad: tanto reloj como gráfica de valores actuales
131     db.ref('sensores/humedad').on('value', (instantanea) => {
132         const humed = instantanea.val();
133
134         let dato = +(humed.valor).toFixed(2);
135         actualizar_indicador(indicador_humedad, null, dato, 'divReloj_humedad', layout3, config3);
136
137         actualizar_grafica('divGrafica_humedad', new Date(), dato);
138     })
139 }
140 }

```

```

143 // Accion sobre actuadores: indicación a FireBase
144 function actuacion(checkbox) {
145     if (checkbox.id=="actuador1") {
146         console.log("actuador 1: " + checkbox.checked);
147         db.ref('actuadores/actuador1').set(checkbox.checked)
148     } else {
149         console.log("actuador 2: " + checkbox.checked);
150         db.ref('actuadores/actuador2').set(checkbox.checked)
151     }
152 }
153
154 // Accion sobre reguladores: indicación a FireBase
155 function regulador(reg) {
156     if (reg.id=="regulador1") {
157         db.ref("actuadores/regulador1").set(+reg.value);
158         document.querySelector("#val_regulador_1").innerHTML="+reg.value+";
159     } else if ((reg.id=="regulador2") ){
160         db.ref("actuadores/regulador2").set(+reg.value);
161         document.querySelector("#val_regulador_2").innerHTML="+reg.value+";
162     }
163 }

```

3. relojes_graficas.js

```

6 // Dimensiones de los relojes y las gráficas para plotly
7
8 let width_indicadores = 300;
9 let height_indicadores = 250;
10
11 let width_graficas = 350;
12 let height_graficas = 300;
13
14
15 // Características de los indicadores de tempertura y humedad
16
17
18 ∨ indicador_temperatura = {
19     type: "indicator",
20     value: 50,
21     delta: { reference: 50 },
22     gauge: { axis: { visible: true, range: [-50, 150] } },
23 }
24
25 ∨ indicador_humedad = {
26     type: "indicator",
27     value: 50,
28     delta: { reference: 50 },
29     gauge: { axis: { visible: true, range: [0, 100] } },
30 }

```

```

35 // Layout para plotly del reloj (temp.)
36
37 let layout1 = {
38     width: width_indicadores,
39     height: height_indicadores,
40     template: {
41         data: {
42             indicator: [
43                 {
44                     title: { text: "Temperatura (°C)" },
45                     mode: "number+delta+gauge",
46                     delta: { reference: -50 }
47                 }
48             ]
49         }
50     }
51 }
52
56 // Layout para plotly de la gráfica (temp.)
57
58 let layout2 = {
59     width: width_graficas,
60     height: height_graficas,
61     showlegend: true,
62     legend: {
63         x: 1,
64         xanchor: 'right',
65         y: 1.2
66     },
67     xaxis: {
68         title: {
69             text: 't',
70             font: {
71                 family: 'Courier New, monospace',
72                 size: 14,
73                 color: '#7f7f7f'
74             }
75         }
76     },
77     yaxis: {
78         title: {
79             text: 'Temperatura (°C)',
80             font: {
81                 family: 'Courier New, monospace',
82                 size: 14,
83                 color: '#7f7f7f'
84             }
85         }
86     },
87 }

```

```

90 // Layout para plotly del reloj (humedad)
91
92 let layout3 = {
93   width: width_indicadores,
94   height: height_indicadores,
95   template: {
96     data: {
97       indicator: [
98         {
99           title: { text: "Humedad relativa (%)" },
100          mode: "number+delta+gauge",
101          delta: { reference: -50 }
102        }
103      ]
104    }
105  }
106 }
109 // Layout para plotly de la gráfica (humedad)
110
111 let layout4 = {
112   width: width_graficas,
113   height: height_graficas,
114   showlegend: true,
115   legend: {
116     x: 1,
117     xanchor: 'right',
118     y: 1.2
119   },
120   xaxis: {
121     title: {
122       text: 't',
123       font: {
124         family: 'Courier New, monospace',
125         size: 14,
126         color: '#7f7f7f'
127       }
128     }
129   },
130   yaxis: {
131     title: {
132       text: 'Humedad relativa (%)',
133       font: {
134         family: 'Courier New, monospace',
135         size: 14,
136         color: '#7f7f7f'
137       }
138     }
139   },
140 }

```



```

144 // configuración para reloj de temperatura
145 let config1={
146     responsive: true,
147     scrollZoom: false,
148     toImageButtonOptions: {
149         format: 'png', // one of png, svg, jpeg, webp
150         filename: 'medida.png',
151         height: 500,
152         width: 700,
153         scale: 1 // Multiply title/legend/axis/canvas sizes by this factor
154     },
155     displayModeBar: true
156 };

158 // misma configuración para resto de relojes y gráficas
159 config2 = config1;
160 config3 = config1;
161 config4 = config1;
162
163
164 // Inic. indicador humedad
165 Plotly.newPlot('divReloj_temperatura', [indicador_temperatura], layout1, config1);
166 // Inic. indicador temperatura
167 Plotly.newPlot('divReloj_humedad', [indicador_humedad], layout3, config3);
170 // Inic gráfica de humedad
171 let data2 = [{
172     x: [],
173     y: [],
174     name: 'Sensor temp. 1',
175     mode: 'scatter',
176     line: {color: '#80CAF6'}
177 }]
178 Plotly.newPlot('divGrafica_temperatura', data2, layout2, config2);
179
180
181 // Inic. gráfica de temperatura
182 var data4 = [{
183     x: [],
184     y: [],
185     name: 'Sensor humedad 1',
186     mode: 'scatter',
187     line: {color: '#80CAF6'}
188 }]
189 Plotly.newPlot('divGrafica_humedad', data4, layout4, config4);

```

