Plataforma base IoT de código abierto para el despliegue de aplicaciones.

Juan Ibero Bilbao
Grado en Ingeniería Informática
Administración de redes y sistemas operativos

J. Ramon Esteban Grifoll
Jordi Serra Ruiz

06/2021

Agradecimientos

A mis padres Ángel y Enilse que siempre han estado ahí en el “*Backend*” apoyándome y creyendo en mí de manera incondicional.

A mis hermanos Fran y José, que hace mucho tiempo plantaron la semilla de este mundo en mi sin saber que esto se convertiría en algo más que en una partida al ordenador o a la “*Family*”.

A Salas por escucharme largo y tendido explicándole todo el proyecto y apoyarme en este último esfuerzo.

A Carlos que ha tenido que aguantarme muchos años normalizándolo todo.

A todos los que me habéis apoyado y aguantado durante este duro pero gratificante camino el cual volvería a recorrer una y 100 veces.

¡Muchas gracias!

FICHA DEL TRABAJO FINAL

Título del trabajo:	Plataforma base IoT de código abierto para el despliegue de aplicaciones.
Nombre del autor:	Juan Ibero Bilbao
Nombre del consultor/a:	J. Ramon Esteban Grifoll
Nombre del PRA:	Jordi Serra Ruiz
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	Grado Ingeniería Informática
Área del Trabajo Final:	Administración de redes y sistemas operativos
Idioma del trabajo:	Castellano
Palabras clave	Internet de las cosas, sistema de control *IoT* Internet of the things, control *IoT* system

Resumen del Trabajo (máximo 250 palabras): Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.

Los sistemas conectados IoT representan una clara ventaja competitiva para las organizaciones tanto a nivel de proceso de datos como en valor añadido del propio sistema (actuadores, inteligencia de negocio, control remoto, visibilidad, etc.).

Existen numerosas trabas de complejidad e implementación de estos sistemas para que sean seguros, escalables y sobre todo accesibles para todas las capas del tejido industrial, ya que la creciente brecha tecnología aleja el IoT de las empresas menos especializadas.

Tratando de dar respuesta a estos sectores, se plantea el estudio y desarrollo de un sistema complete de IoT genérico que pueda utilizarse de base para construir otros sistemas concretos de una manera simple, accesible y segura para aquellas organizaciones que quieran aprovechar las ventajas del IoT.

El sistema se basa en un flujo de IoT completo, desde los sensores o actuadores hasta la gestión de los dispositivos por parte del mantenedor. Esto correrá a cargo de una aplicación WEB dedicada.

Aparte del completo sistema (desde las Things hasta el cliente), se plantean 2 aplicaciones de ejemplo apoyadas en el sistema IoT que simulen 2 empresas concretas con sus propias necesidades, medición de cuadros eléctricos, y medición de ambiente en CPD`s remotos.

Abstract (in English, 250 words or less):

Connected IoT systems represent a competitive advantage for organizations both in data processing and in the added value of the system itself (actuators, business intelligence, remote control, visibility, etc.).

There are numerous obstacles of complexity and implementation of these systems so that they are safe, scalable and above all accessible to all layers of the industrial fabric since the growing technological gap moves the IoT away from less specialized companies.

Trying to respond to these sectors, the study and development of a complete generic IoT system is proposed that can serve as a basis for building other specific systems in a simple, accessible and secure way for those organizations that want to take advantage of the advantages. of the IoT.

The system is based on a complete flow of IoT, from the sensors or actuators to the management of the devices by the maintainer. This will be done through a dedicated WEB application.

In addition to the complete system (from Things to the client), a control web portal is proposed with 2 example applications supported by the IoT system that simulate 2 specific companies with their own needs, measurement of electrical panels and remote measurement of the environment in CPD `s.

Índice

Lista de figuras

Lista de Tablas

1. Introducción

1.1. Contexto y justificación del Trabajo

El valor de los datos y la monitorización remota son una realidad tangible de nuestra era, tanto en los hogares como en la industria. Supone un activo muchas veces infravalorado que puede aportar valor a prácticamente cualquier servicio/producto/instalación o compañía. La explotación de estos es una tendencia en auge y un trampolín a la trasformación digital de prácticamente cualquier infraestructura (doméstica o industrial) que podamos imaginar.

Durante años la industria se ha modernizado y la tendencia del "4.0" en las compañías ha supuesto una revolución en productividad y optimización de los recursos. Desde el punto de vista del *IoT* (del inglés Internet of things) el crecimiento ha evolucionado desde el uso doméstico para cada vez más trasladar el foco al sector industrial (*IIoT*).

Ya en 2020 existían 20.400 millones de dispositivos inteligentes (un crecimiento del 219% desde 2017) (García Munguía et al., 2020) Según el sector se prevé un crecimiento medio anual del 7,4% entre 2020 y 2025 lo que supondría elevar el volumen de negocio a 110.600 millones de dólares.

Aprovechar las ventajas que puede proporcionar el *IoT* es un aliciente que está revolucionando todos los mercados, descubrir cómo y cuándo se pueden utilizar estos subsistemas quizás no esté tan claro para la mayoría del tejido industrial con menos recursos.

Las palabras *IIoT* resuenan en los estudios y en las grandes empresas, pero el acceso de la mayoría de pequeñas y medianas organizaciones (en gran parte gracias al bajo grado de transformación digital del mercado) no está en el foco de los grandes planes de implantación o modernización *IoT*.

Es aquí donde el mercado es exponencial y surgen multitud de oportunidades para nuevos modelos que pueden aportar valor o una visión distinta.

El valor de la gestión remota y la optimización de los datos está fuera de toda duda, sin embargo, en la actualidad la complejidad de los entornos y la falta de accesibilidad para las pequeñas y medianas organizaciones es una barrera que de momento no acaba de atravesar el mercado del *IIoT*. El mundo se mueve hacia un ecosistema conectado, todo está "en línea" y esta realidad se ve día a día en la vida de la gente (smartphones, Smart TV, bombillas inteligentes, gadgets y wareables varios…). Aunque es cierto que en el tejido empresarial español (más del 99% PYMES) no se acaba de integrar del todo con las posibilidades de *IIoT*.

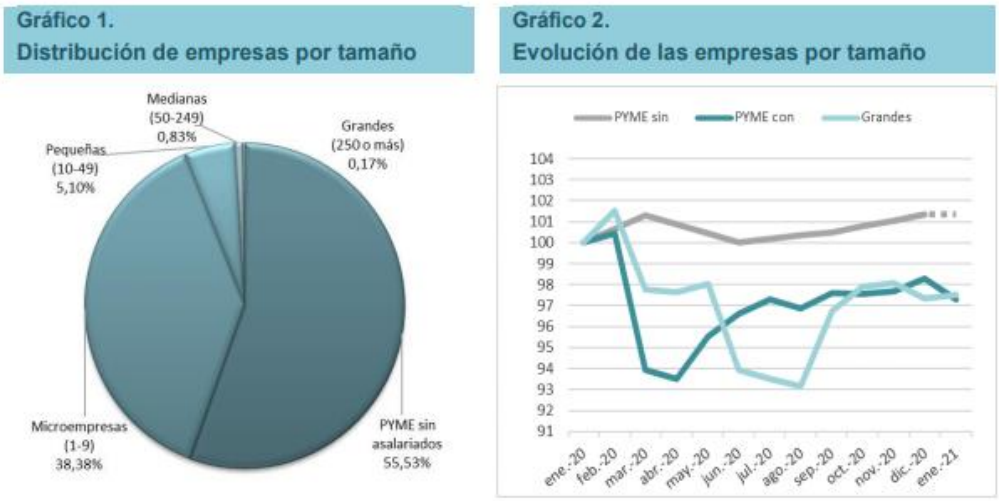


Imagen 1 – Cifras PYME. Datos enero 2021. Ministerio de Industria (Dirección General de Industria y de la PYME, 2020)

La cantidad de datos suministrados por en *IoT* se alinea con los ecosistemas conectados y se prevé que hasta 2025 la tasa de crecimiento anual compuesta se vea incrementada un 28,7%, un total de 41.600 millones de dispositivos (*IoT* y *IIoT*) conectados generando la nada despreciable cantidad de 79,4 zettabytes (ZB) de datos (*Crecimiento Dispositivos Conectados 2025 @ Signalsiot.Com*, n.d.).

Existen soluciones de mercado que pueden ofrecer estos servicios (Google, Amazon, Efor, sumologic, Microsoft...), pero la conclusión que se puede obtener haciendo una búsqueda preliminar es que se ofrecen grandes soluciones, complejas de implementar, y orientadas a grandes compañías, cuyo fin último excluye a la mayoría de pequeñas y medianas empresas que pueden beneficiarse del *IIoT* sin llegar a necesitar complejos modelos de predicción de datos y deslocalización de sistemas a nivel mundial.

Este TFG trata de estudiar el diseño e implementación de un sistema de IIoT orientado al control más recurrente de sensores y entradas/salidas, con un producto que pueda ser accesible además con un coste contenido para pequeñas y medianas empresas, aportando también valor y control de los datos que las pequeñas compañías generan, sin necesitar complejas implementaciones.

1.2. Objetivos del Trabajo

Se plantean los siguientes objetivos a cumplir dados los plazos y las limitaciones de infraestructura:

- Disponer de una plataforma base de gestión centralizada de IoT basada en Open Source donde apoyar aplicaciones de servicio a empresas.

- Garantizar el fácil acceso a la plataforma por internet y la gestión de las things.

- Disponer de una aplicación basada en la plataforma IoT para monitorizar el entorno en centros de procesamiento de datos e intensidad en cuadros eléctricos.

1.3. Enfoque y método seguido

Hoy en día es difícil adentrarse en un campo que no esté ya explotado por la industria. En el ámbito de las tecnologías de la información, surgen a diario oportunidades dada la elevada velocidad con la que se producen cambios en la industria. El *IoT*, es relativamente reciente en términos absolutos, aunque en términos de tecnología, lleva tiempo suficiente como para que el mercado se haya asentado para tener una oferta suficientemente grande.

Por todo esto, se usarán herramientas de análisis de mercado de soluciones ya existentes. Esto dará una perspectiva global del mercado y de cómo se llevan a cabo las tareas en la industria.

Una vez se tenga una visión general, se hará un análisis de posibles tecnologías a aplicar en el proyecto, teniendo el foco en los objetivos descritos. De este análisis surgirá un planteamiento de arquitectura del sistema, abarcando todos los aspectos de la planificación y los objetivos, desde los clientes hasta los sensores finales.

Teniendo este mapa de ruta tecnológico y una visión global del funcionamiento de este tipo de sistemas, se procederá a realizar un desarrollo de prototipo focalizando los esfuerzos en conseguir cumplir los objetivos de mayor impacto, teniendo en cuenta al cliente final y el valor añadido que se pueda asociar con el producto.

Por tanto, el desarrollo se centrará en los siguientes grandes bloques:

1. Análisis de Mercado y Cliente
2. Estudio y análisis de Infraestructura
3. Construcción del Core del prototipado
4. Diseño e implementación de herramientas de cliente

En el ámbito del desarrollo de software, en lo que se refiere al Core y las presentaciones de clientes, seguiremos una metodología clásica en el que se unirá un desarrollo en cascada con un enfoque de prototipado, pudiendo hacer una planificación vertical centrada en los objetivos (cascada) pero recurriendo a pruebas básicas del prototipo para poder ajustar el producto. A esta metodología se le conoce como Espiral y aporta gran valor a este tipo de proyectos.

Se ejecutarán ciclos de PLANIFICACION-DESARROLLO-ANÁLISIS consiguiendo en cada iteración una versión del producto más avanzada y funcional.

La estrategia se basa en la adquisición de conocimientos mediante el estudio de soluciones, tecnologías y mercado, pudiendo tener una visión clara del camino seguido por la industria y evitando cometer errores de implementación o selección de arquitectura. Este planteamiento aporta unas bases sólidas de conocimiento, una planificación adecuada y un crecimiento en capacidades transversales escalonado, dando pie a un producto apoyado en el conocimiento y la planificación, pudiendo obtener un prototipo de calidad que satisfaga los objetivos planteados.

1.4. Planificación del Trabajo

En la programación se ha tenido en cuenta todos los días naturales hasta la entrega de este TFG, los días festivos no se han tenido en cuenta dada la naturaleza de la redacción de este TFG que se harán fuera del horario laboral habitual.

Por cada día se ha estimado un trabajo constante de 2 a 2,5 horas exceptuando momentos específicos donde se solapan redacción de PEC`s con trabajos propios del desarrollo del prototipo donde se estima un trabajo diario de 2,5 a 5 horas.

No obstante, dada la priorización del objetivo 1 y 2, si surgieran retrasos o inconvenientes graves en el desarrollo del prototipo se puede reducir el desarrollo del objetivo 3 a una de las organizaciones.

1.4.1. Tabla de trabajo

En la siguiente tabla se detalla una programación preliminar del TFG:

Nº	Nombre de tarea	Duración	Comienzo	Fin
1	**Trabajo Especifico TFG**	**96 días**	**mié 17/02/21**	**dom 23/05/21**
1.1	**Tarea 1 - Análisis del mercado de IoT y sus tecnologías**	**10 días**	**mié 17/02/21**	**vie 26/02/21**
1.1.1	Recabar documentación general sobre IoT	3 días	mié 17/02/21	vie 19/02/21
1.1.2	Recabar documentación técnica y de sistemas IoT	3 días	sáb 20/02/21	lun 22/02/21
1.1.3	Recabar información y documentación sobre otros productos o servicios	3 días	mar 23/02/21	jue 25/02/21
1.1.4	Presentar Propuesta de trabajo	1 día	vie 26/02/21	vie 26/02/21
1.2	**Tarea 2 - Análisis de cliente y mercado**	**8 días**	**sáb 27/02/21**	**sáb 06/03/21**
1.2.1	Definir un perfil de los clientes propuestos	2 días	sáb 27/02/21	dom 28/02/21
1.2.2	Definir los requisitos de cada Cliente propuesto	2 días	lun 01/03/21	mar 02/03/21
1.2.3	Definir los objetivos de la implementación de los clientes	3 días	mié 03/03/21	vie 05/03/21
1.2.4	Documento de Check list de OBJETIVOS TFG	1 día	sáb 06/03/21	sáb 06/03/21
1.3	**Tarea 3 - Estudio y análisis de infraestructura**	**29 días**	**dom 07/03/21**	**dom 04/04/21**
1.3.1	Estudio y análisis de los protocolos IoT	2 días	dom 07/03/21	lun 08/03/21
1.3.2	Estudio y análisis de comunicaciones	2 días	mar 09/03/21	mié 10/03/21
1.3.3	Estudio y análisis de sistemas Operativos	3 días	jue 11/03/21	sáb 13/03/21
1.3.4	Estudio y análisis de Bases de datos	4 días	dom 14/03/21	mié 17/03/21
1.3.5	Estudio y análisis de electrónica low cost	4 días	jue 18/03/21	dom 21/03/21
1.3.6	Estudio y análisis de tecnologías de infraestructura de software	5 días	lun 22/03/21	vie 26/03/21

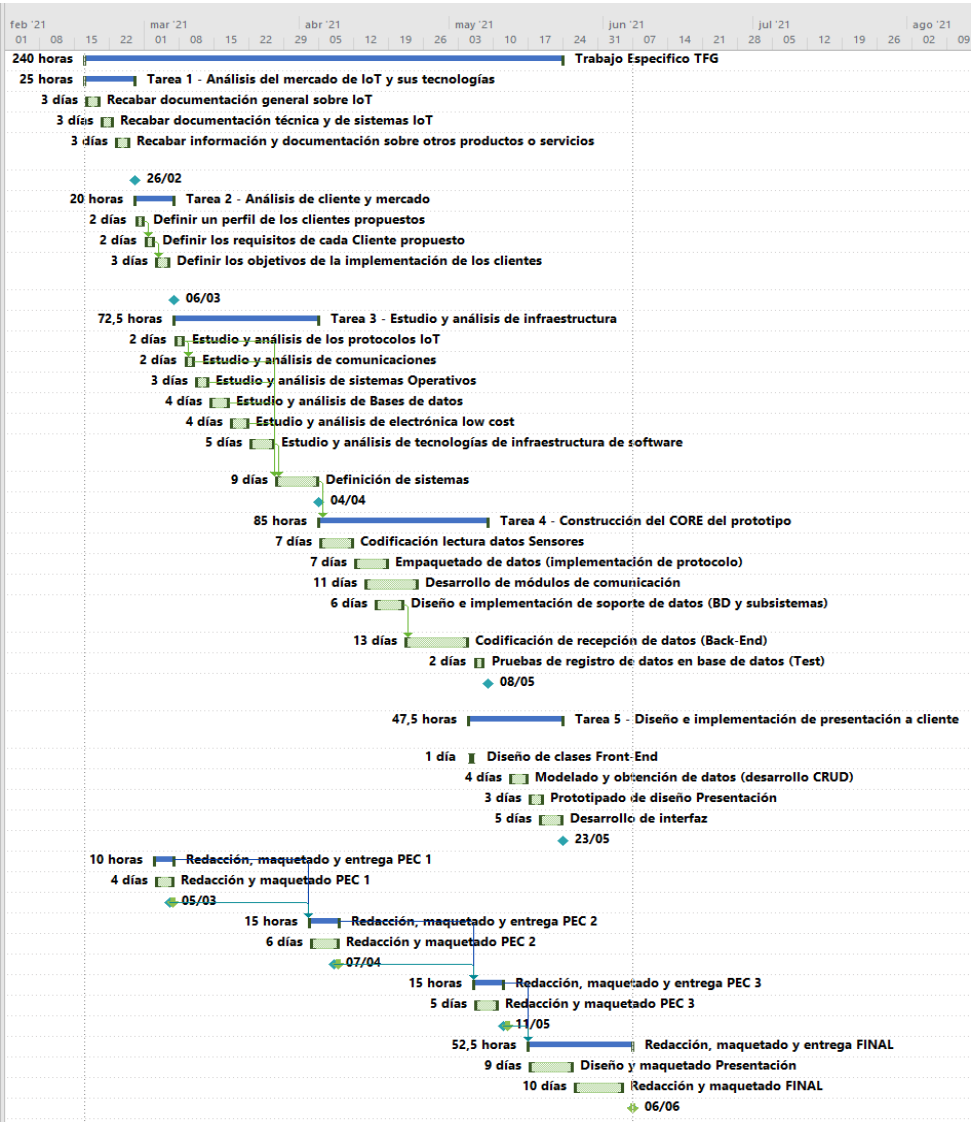
Nº	Nombre de tarea	Duración	Comienzo	Fin
1.3.7	Definición de sistemas	9 días	sáb 27/03/21	dom 04/04/21
1.3.8	Documento de especificación de Arquitectura	1 día	dom 04/04/21	dom 04/04/21
1.4	**Tarea 4 - Construcción del CORE del prototipo**	**34 días**	**lun 05/04/21**	**sáb 08/05/21**
1.4.1	Codificación lectura datos Sensores	7 días	lun 05/04/21	dom 11/04/21
1.4.2	Empaquetado de datos (implementación de protocolo)	7 días	lun 12/04/21	lun 19/04/21
1.4.3	Desarrollo de módulos de comunicación	11 días	mié 14/04/21	sáb 24/04/21
1.4.4	Diseño e implementación de soporte de datos (BD y subsistemas)	6 días	vie 16/04/21	mié 21/04/21
1.4.5	Codificación de recepción de datos (Back-End)	13 días	jue 22/04/21	mar 04/05/21
1.4.6	Pruebas de registro de datos en base de datos (Test)	2 días	jue 06/05/21	vie 07/05/21
1.4.7	Se registran los datos de los sensores Correctamente en el sistema	1 día	sáb 08/05/21	sáb 08/05/21
1.5	**Tarea 5 - Diseño e implementación de presentación a cliente**	**19 días**	**mié 05/05/21**	**dom 23/05/21**
1.5.1	Diseño de clases Front-End	1 día	mié 05/05/21	mié 05/05/21
1.5.2	Modelado y obtención de datos (desarrollo CRUD)	4 días	jue 13/05/21	dom 16/05/21
1.5.3	Prototipado de diseño Presentación	3 días	lun 17/05/21	mié 19/05/21
1.5.4	Desarrollo de interfaz	5 días	mié 19/05/21	dom 23/05/21
1.5.5	Pruebas y test de funcionamiento	1 día	dom 23/05/21	dom 23/05/21
2	**Redacción, maquetado y entrega PEC 1**	**4 días**	**mié 03/03/21**	**sáb 06/03/21**
2.1	Redacción y maquetado PEC 1	4 días	mié 03/03/21	sáb 06/03/21
2.2	Entrega PEC1	1 día	vie 05/03/21	vie 05/03/21
3	**Redacción, maquetado y entrega PEC 2**	**6 días**	**sáb 03/04/21**	**jue 08/04/21**
3.1	Redacción y maquetado PEC 2	6 días	sáb 03/04/21	jue 08/04/21

Nº	Nombre de tarea	Duración	Comienzo	Fin
3.2	Entrega PEC2	1 día	mié 07/04/21	mié 07/04/21
4	**Redacción, maquetado y entrega PEC 3**	**6 días**	**jue 06/05/21**	**mar 11/05/21**
4.1	Redacción y maquetado PEC 3	5 días	jue 06/05/21	lun 10/05/21
4.2	Entrega PEC3	1 día	mar 11/05/21	mar 11/05/21
5	**Redacción, maquetado y entrega FINAL**	**20 días**	**vie 17/05/21**	**mar 06/06/21**
5.1	Diseño y maquetado Presentación	9 días	lun 17/05/21	mar 25/05/21
5.2	Redacción y maquetado FINAL	10 días	mié 26/05/21	vie 04/06/21
5.3	Entrega Final	1 día	dom 06/06/21	dom 06/06/21

Tabla 1 - Planificación

1.4.2. Diagrama de Gantt

	Núm de esquema	Nombre de tarea	Duración	Comienzo	Fin
1	**1**	**Trabajo Especifico TFG**	**240 horas**	**mié 17/02/21**	**dom 23/05/21**
2	**1.1**	**Tarea 1 - Análisis del mercado de IoT y sus tecnologías**	**25 horas**	**mié 17/02/21**	**vie 26/02/21**
3	1.1.1	Recabar documentación general sobre IoT	3 días	mié 17/02/21	vie 19/02/21
4	1.1.2	Recabar documentación técnica y de sistemas IoT	3 días	sáb 20/02/21	lun 22/02/21
5	1.1.3	Recabar información y documentación sobre otros productos o servicios	3 días	mar 23/02/21	jue 25/02/21
6	1.1.4	Presentar Propuesta de trabajo	1 día	vie 26/02/21	vie 26/02/21
7	**1.2**	**Tarea 2 - Análisis de cliente y mercado**	**20 horas**	**sáb 27/02/21**	**sáb 06/03/21**
8	1.2.1	Definir un perfil de los clientes propuestos	2 días	sáb 27/02/21	dom 28/02/21
9	1.2.2	Definir los requisitos de cada Cliente propuesto	2 días	lun 01/03/21	mar 02/03/21
10	1.2.3	Definir los objetivos de la implementación de los clientes	3 días	mié 03/03/21	vie 05/03/21
11	1.2.4	Documento de Check list de OBJETIVOS TFG	1 día	sáb 06/03/21	sáb 06/03/21
12	**1.3**	**Tarea 3 - Estudio y análisis de infraestructura**	**72,5 horas**	**dom 07/03/21**	**dom 04/04/21**
13	1.3.1	Estudio y análisis de los protocolos IoT	2 días	dom 07/03/21	lun 08/03/21
14	1.3.2	Estudio y análisis de comunicaciones	2 días	mar 09/03/21	mié 10/03/21
15	1.3.3	Estudio y análisis de sistemas Operativos	3 días	jue 11/03/21	sáb 13/03/21
16	1.3.4	Estudio y análisis de Bases de datos	4 días	dom 14/03/21	mié 17/03/21
17	1.3.5	Estudio y análisis de electrónica low cost	4 días	jue 18/03/21	dom 21/03/21
18	1.3.6	Estudio y análisis de tecnologías de infraestructura de software	5 días	lun 22/03/21	vie 26/03/21
19	1.3.7	Definición de sistemas	9 días	sáb 27/03/21	dom 04/04/21
20	1.3.8	Documento de especificación de Arquitectura	1 día	dom 04/04/21	dom 04/04/21
21	**1.4**	**Tarea 4 - Construcción del CORE del prototipo**	**85 horas**	**lun 05/04/21**	**sáb 08/05/21**
22	1.4.1	Codificación lectura datos Sensores	7 días	lun 05/04/21	dom 11/04/21
23	1.4.2	Empaquetado de datos (implementación de protocolo)	7 días	lun 12/04/21	dom 18/04/21
24	1.4.3	Desarrollo de módulos de comunicación	11 días	mié 14/04/21	sáb 24/04/21
25	1.4.4	Diseño e implementación de soporte de datos (BD y subsistemas)	6 días	vie 16/04/21	mié 21/04/21
26	1.4.5	Codificación de recepción de datos (Back-End)	13 días	jue 22/04/21	mar 04/05/21
27	1.4.6	Pruebas de registro de datos en base de datos (Test)	2 días	jue 06/05/21	vie 07/05/21
28	1.4.7	Se registran los datos de los sensores Correctamente en el sistema	1 día	sáb 08/05/21	sáb 08/05/21
29	**1.5**	**Tarea 5 - Diseño e implementación de presentación a cliente**	**47,5 horas**	**mié 05/05/21**	**dom 23/05/21**
30	1.5.1	Diseño de clases Front-End	1 día	mié 05/05/21	mié 05/05/21
31	1.5.2	Modelado y obtención de datos (desarrollo CRUD)	4 días	jue 13/05/21	dom 16/05/21
32	1.5.3	Prototipado de diseño Presentación	3 días	lun 17/05/21	mié 19/05/21
33	1.5.4	Desarrollo de interfaz	5 días	mié 19/05/21	dom 23/05/21
34	1.5.5	Pruebas y test de funcionamiento	1 día	dom 23/05/21	dom 23/05/21
35	**2**	**Redacción, maquetado y entrega PEC 1**	**10 horas**	**mié 03/03/21**	**sáb 06/03/21**
36	2.1	Redacción y maquetado PEC 1	4 días	mié 03/03/21	sáb 06/03/21
37	2.2	Entrega PEC1	1 día	vie 05/03/21	vie 05/03/21
38	**3**	**Redacción, maquetado y entrega PEC 2**	**15 horas**	**sáb 03/04/21**	**jue 08/04/21**
39	3.1	Redacción y maquetado PEC 2	6 días	sáb 03/04/21	jue 08/04/21
40	3.2	Entrega PEC2	1 día	mié 07/04/21	mié 07/04/21
41	**4**	**Redacción, maquetado y entrega PEC 3**	**15 horas**	**jue 06/05/21**	**mar 11/05/21**
42	4.1	Redacción y maquetado PEC 3	5 días	jue 06/05/21	lun 10/05/21
43	4.2	Entrega PEC3	1 día	mar 11/05/21	mar 11/05/21
44	**5**	**Redacción, maquetado y entrega FINAL**	**52,5 horas**	**lun 17/05/21**	**dom 06/06/21**
45	5.1	Diseño y maquetado Presentación	9 días	lun 17/05/21	mar 25/05/21
46	5.2	Redacción y maquetado FINAL	10 días	mié 26/05/21	vie 04/06/21
47	5.3	Entrega Final	0 días	dom 06/06/21	dom 06/06/21



1.5. Breve sumario de productos obtenidos

El diseño se plantea como un completo sistema de *IoT* capaz de dar respuesta a interacciones simples de sensores desplegados en diferentes lugares para convertir estos datos en valiosos para la industria.

El diseño de Core será el encargado de gestionar de forma genérica los sensores, pudiendo crear e implementar estos de manera sencilla mediante una interfaz WEB.

En segundo lugar, se dispondrá de aplicativos de empresas para explotar los datos obtenidos:

- Un panel que pueda monitorizar a distancia los diferentes CPD deslocalizados.

- Un panel capaz de monitorizar cuadros eléctricos.

El sistema es modular y permite basar aplicativos variopintos y de distinta índole sobre el *core*, pudiendo dar servicio a diferentes clientes.

1.6. Breve descripción de los otros capítulos de la memoria

A continuación, se procede a hacer una descripción breve de los capítulos de este TFG:

- *Capítulo 2 Análisis del entorno IoT - Posibilidades*

  En este capítulo se describe y analiza el entorno del *IoT*, opciones de mercado, historia, arquitectura, visión etc.

  Con este capítulo se sientan las bases de conocimiento para las siguientes fases del trabajo, es clave para comprender el contexto y la teoría asociada.

- *Capítulo 3 Modelo de negocio valor y propuesta*

  En este capítulo se analizan las ideas de negocio y las estrategias para concluir en una propuesta de valor para el proyecto.

  Las conclusiones y visión que se obtienen de este capítulo centraran los objetivos además de dar foco a los siguientes apartados, dado que el mundo del *IoT* es muy extenso y heterogéneo.

- *Capítulo 4 Diseño del Sistema – Arquitectura*

En esta fase nos centraremos primero en la toma de decisiones técnicas sobre la arquitectura, posteriormente en la implementación y desarrollo del core, para concluir en la implementación de aplicaciones de cliente.

- *Capítulo 5 Puesta en marcha y configuración*

En la quinta fase el trabajo se centra en la instalación y configuración más particular de todas las piezas de la arquitectura.

- *Capítulo 6 Costes estimados del proyecto*

Fase de estimación de costes, se tendrá en cuenta el desarrollo, el despliegue (local y cloud) y el trabajo necesario para cada uno de estos.

- *Capítulo 7 Conclusiones TFG*

En este capítulo se analizará el desarrollo del trabajo y se extraerán conclusiones de este, haciendo hincapié en el aprendizaje y el análisis.

2. Análisis del entono IoT - Posibilidades

2.1. Introducción

En el presente capítulo se pretende dar una idea general de lo que es y ha sido el *IoT*, sus ventajas e inconvenientes, así como los sensores sistemas y protocolos más utilizados actualmente.

El situarse en el marco tanto histórico como actual de los sistemas da perspectiva e información para poder entender los planteamientos realizados.

2.2. Historia

El termino *IoT* o Internet de las cosas tiene un origen más bien reciente (1999, Kevin Ashton), en términos de tecnología 22 años suponen una etapa enorme, sin embargo, en términos históricos es un periodo relativamente corto.

En 1999 Kevin Ashton acuño el termino, aunque este se utilizaba en círculos internos (concretamente en el Instituto Tecnológico de Massachussets), no fue hasta 2009 cuando el propio Ashton utilizo el termino de forma pública.

El origen reciente del término no implica que este no existiese ya mucho antes de ponerle un nombre. Realmente el origen de los objetos conectados como concepto se remonta a 1874 cuando unos científicos franceses instalaron unos sensores de telemetría meteorológica en la cima del Mont Blanc. Los datos de estos sensores eran transmitidos a París donde se procesaban de manera mucho más cómoda que en la cima de la montaña.

Se empezaba a vislumbrar un futuro prometedor para los sistemas interconectados, las posibilidades eran infinitas y visionarios de la época como Nikola Tesla (1926) o Alan Turing (1950) plasmaron visiones al respecto en sus escritos científicos.

La interconexión de las cosas, impulsada por el avance de la tecnología, desemboco entre los años 60 y 80 con el desarrollo Arpanet (en 1969 se envió el primer mensaje) y los primeros protocolos de comunicación que son la base del internet moderno, en concreto el modelo TCP/IP que fue desarrollado en 1973, integrándose en el corazón de ARPANET en 1983 remplazando los demás protocolos.

En 1982 en el Departamento de *Computer Science* del Carnegie Mellon se produjo la interconexión de una máquina de Coca Cola a uno de los servidores de la compañía, desde el propio ordenador de los empleados

se podía comprobar si la maquina tenía bebidas o si estas estaban a una temperatura adecuada, una primera versión sin duda de lo que hoy conocemos como “Objetos conectados”.

Sin embargo, no fue hasta los años 90 cuando nació lo que hoy conocemos como Internet, el auténtico impulsor de las redes *IoT*. En este periodo (1990-1999) varios proyectos de objetos conectados surgieron, aunque sin mucho éxito, sin embargo, la industria ya estaba cambiando, cada vez se volvía más necesario estar conectado y empezaban a surgir multitud de tecnologías que pretendían revolucionar el mapa de las empresas.

Una de estas tecnologías surge a comienzos del siglo XXI. Esta permite interconectar sistemas de manera inalámbrica y se populariza como Wifi. Aquí es donde se produce la primera gran expansión de los objetos conectados, el proporcionar un estándar de comunicación inalámbrica supuso un antes y un después para el *IoT*.

Detrás quedan los sensores colocados en el Mont Blanc, en 2008 ya había más objetos conectados que personas, el *IoT* ya era una realidad tangible y las previsiones eran increíblemente prometedoras.

Como hemos comentado antes, en 2009 Ashton publica *That “Internet of Things”Thing* (ASHTON, n.d.), en el artículo Ashton habla del término *IoT* y de su futuro con frases clarificadoras como “*The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so.”.*

Durante esta década, el *IoT* ha irrumpido con fuerza tanto en la vida de las personas como en la industria, y ha venido para quedarse, aunque no seamos conscientes, los objetos conectados están por todas partes allí donde vallamos y son parte del día a día del mundo.

La tendencia hoy es de crecimiento, los datos que estos sistemas manejan se cuentan por zettabytes (ZB) y manejar bien esta tecnología en la industria supone una importante ventaja competitiva no solo para las grandes compañías, sino también para pequeñas y medianas empresas donde el termino *IoT* aun suena como algo lejano.

2.3. Sectores de implementación *IoT*

Los sectores de negocio de un producto requieren especial atención en el diseño de modelos que pretenden añadir valor no como producto, sino como servicio. En el mundo *IoT*, generalmente hay que tener muy presente los diferentes sectores de actuación ya que las implementaciones cambian radicalmente tanto en arquitectura, en modelos de negocio, en valor añadido, etc.

Los sectores donde el *IoT* está teniendo especial relevancia se pueden resumirlos en 4 grandes grupos:

1. **Smart City**

   Conocidas como ciudades inteligentes, tiene un prometedor futuro (y en algunos casos presente) en el mundo de las implementaciones de sistemas conectados.
   Pueden utilizarse como ejemplo para:

   - Servicios a la ciudadanía como aparcamiento, alquiler de vehículos (bicicletas, coches…)
   - Edificios inteligentes (eficiencia en la gestión energética)
   - Gestión inteligente del tráfico

2. **Industria de la Salud**

   En mundo relacionado con la salud siempre ha sido un nicho de negocio selecto y bastante cotizado, el hecho de tener un mercado que nunca se agota, la movilidad necesaria para la monitorización de pacientes, y el continuo avance del sector hace que el *IoT* sea necesario para permitir que los datos valiosos lleguen de los pacientes a los investigadores/médicos/farmacéuticas etc.…

   Se pueden obtener aplicaciones como:

   - Gestión de inventario de medicamentos para su disponibilidad (la reciente pandemia ha destapado una fehaciente necesidad en este aspecto)
   - Análisis masivo de enfermedades por parte de los investigadores

     1. Datos en tiempo real
     2. *Machine learning*
     3. Decisiones mucho más certeras

   - Monitorización de los pacientes en tiempo real

3. **Industria del automóvil y del hogar**

   Los hogares inteligentes ya son una realidad, es un hecho que el *IoT* ha irrumpido con fuerza en los hogares de todo el mundo, aunque es cierto que el paso a viviendas conectadas aún está por llegar. Se puede vislumbrar un futuro en que se tenga en cuenta el *IoT* en las viviendas de nueva construcción y que la auténtica Smart Home o casa conectada tenga un hueco importante en el mercado.

   Los automóviles son otro gran reto para el sector, todos los vehículos ya cuentan con sistemas que almacenan electrónicamente valores en tiempo

real en sus centrales de datos, solo queda el paso de que estos datos sean conectados con *IoT*.

Los productos previstos más prometedores son:

- Hogares inteligentes, eficientes y seguros con *IoT*
- Mantenimientos predictivos en los vehículos
- Datos de uso para seguros tanto del hogar como de los automóviles
- Conectividad total con las infraestructuras de transporte por parte de las personas y los vehículos

4. Industria 4.0

El sector industrial es por excelencia (y volumen de negocio) el grupo más prometedor para estas tecnologías, el hecho de poder interconectar todos los elementos de una fábrica/industria/edificio y obtener patrones, acciones y datos valiosos y poder darles un valor de retorno en procesos productivos más eficientes, fabricas "más inteligentes", empleados más eficientes etc., da a la industria 4.0 un valor para el sector muy superior a la implementación de este tipo de sistemas.
Es en este sector donde centraremos este TFG.

2.4. Visión actual de los sistemas *IIoT*

En la actualidad, prácticamente cualquier entorno es propenso de la utilización de sistemas de *IoT*, el estado actual de las tecnologías de comunicación y el abaratamiento de la electrónica ha llevado al *IoT* a un "punto dulce" de madurez.

Como hemos visto en el apartado anterior la oferta es amplia y la competencia feroz, las grandes compañías ofertan sus servicios intentando diferenciarse de la competencia cada vez más igualada, la potencia de computación en la nube hace posible manejar una cantidad ingente de datos, obtener complejos modelos de inteligencia artificial y plasmar complicadas secuencias de datos en simples paneles de inteligencia de negocio.

Es una realidad que el día a día está plagado de dispositivos conectados, aunque también es cierto que los sistemas no domésticos albergan complejidad en su implementación, y las ventajas competitivas se despegan del complejo tejido empresarial donde las pequeñas y medianas empresas no disponen de conocimientos para implementar modelos de *IIoT* que puedan ayudarles.

Son numerosas las opciones y complicadas las soluciones, el objeto de este TFG es estudiar, valorar y construir una plataforma *IoT* que sea viable y de bajo coste para las empresas que no necesiten compiladas soluciones de análisis de datos. El hacer accesible el *IoT* al grueso de la

industria es un reto a gran escala donde todavía queda mucho trabajo por hacer.

La reutilización de código del sistema y la integración sencilla a un coste contenido son claves para un proyecto de este tipo.

2.5. Comparativa de sistemas existentes

Se estima que en 2019 las plataformas conocidas de *IoT* eran de 620 (*Number of Publicly Known Internet of Things (IoT) Platforms Worldwide from 2015 to 2019 - Www.Statista.Com*, n.d.) (el doble que en 2018), las grandes empresas muestran su interés por esta industria que mueve miles de millones al año y como hemos analizado, tiene un futuro prometedor.

A continuación, se van a analizar las plataformas de *IoT* más sólidas del mercado y que aportan ventajas con la competencia.

2.5.1. Amazon Web Services (AWS) IoT

Amazon es líder en sistemas conectados y en servicios Cloud, es uno de los sistemas más elegidos por las empresas y como es de esperar tiene una plataforma de *IoT* "AWS *IoT*". Una de las ventajas competitivas de la plataforma de AWS es la oferta de servicios separados para la gestión de datos y monitorización. Dispone también de potentes servicios de análisis de datos y una escalabilidad simple y extremadamente potente.

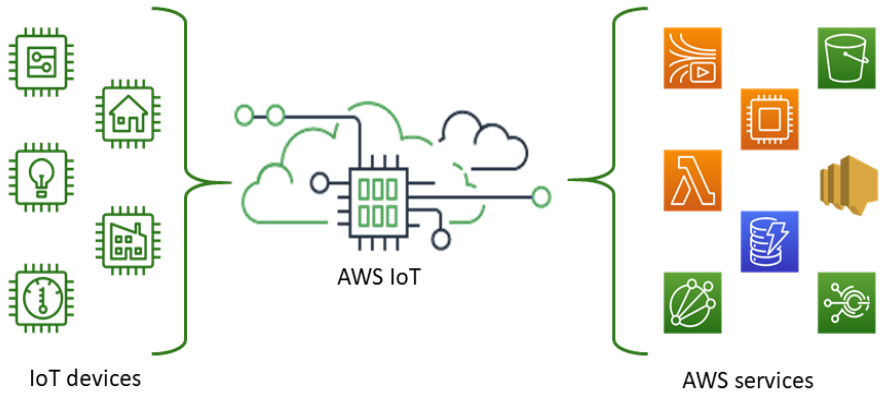


Imagen 2 – Core AWS IoT - Fuente aws.amazon.com (*Amazon AWS IoT - Aws.Amazon.Com*, n.d.)

Los servicios más populares que se ofertan son los siguientes:

- AWS *IoT* Core

Permite que los sistemas interactúen de manera sencilla

- AWS *IoT* Device Defender

Auditoria de seguridad para *IoT*

- AWS *IoT* Device Management

Este servicio Facilita el registro y la organización de los datos

- AWS *IoT* Analytics, SiteWise, Events, ThingsGraph

Son servicios orientados al análisis de volúmenes masivos de datos, diseñados para facilitar el control a los desarrolladores.

2.5.2. Soluciones del Internet de las cosas de Google Cloud

Otro de los grandes comercializadores de servicios en internet, como era de esperar, Google también dispone de servicios Cloud para *IoT* llamada Google Cloud *IoT*. El servicio ofrece herramientas para procesar, analizar y conectar sistemas, aunque de una manera bastante compleja y sin muchas opciones de personalización, lo que puede llegar a ser un problema para la gran mayoría de empresas.

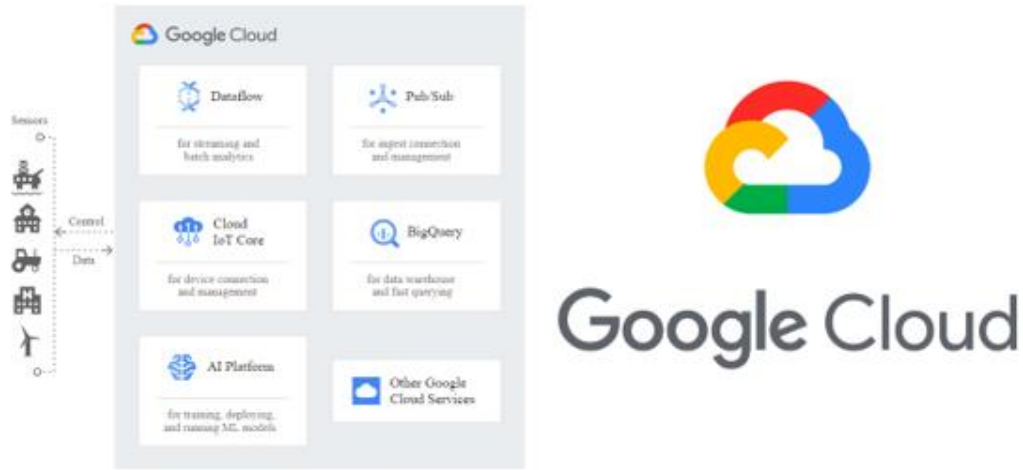


Imagen 3 – Core Google IoT – Fuente cloud.google.com (*IOT Google - Cloud.Google.Com*, n.d.)

Entre las aplicaciones más populares se encuentran:

- Edificios y ciudades inteligentes
- Mantenimiento predictivo mediante análisis de datos
- Seguimiento de activos en tiempo real (*IIoT*)
- Gestión logística y de la cadena de suministro
- Análisis de datos Avanzado

2.5.3. Microsoft Azure *IoT*

Microsoft no podía quedarse fuera del negocio del *IoT*, ni tampoco podía desperdiciar el potencial de la que posiblemente sea una de las mejores plataformas de la nube en la actualidad, la suite Azure.

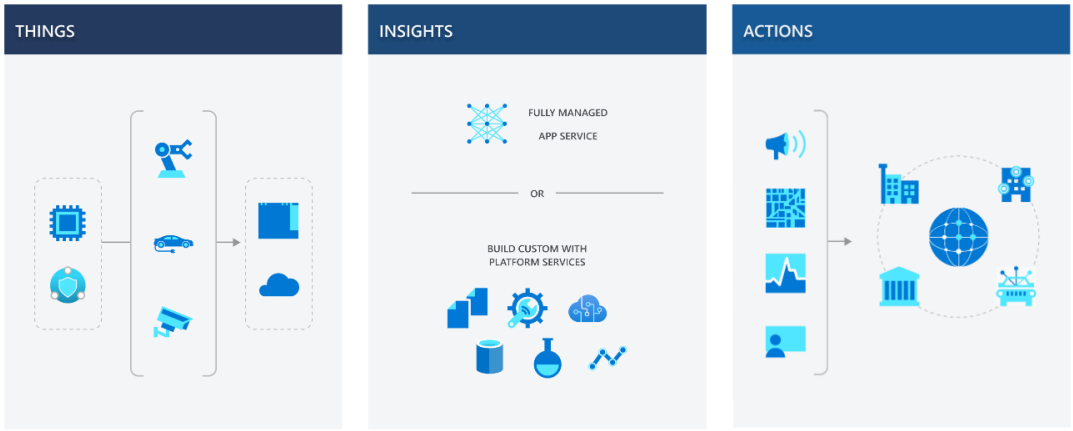


Imagen 4 – Implementación Azure IoT – Fuente azure.microsoft.com(*Microsoft, Explore the Benefits of Azure IoT - Azure.Microsoft.Com*, n.d.)

Los servicios de Azure aportan un perímetro de actuación de los sistemas de *IoT* que interconectan los sensores con la plataforma para dar valor a los datos y poder tomar acciones en consecuencia. Definitivamente un planteamiento acertado similar a la competencia.

No obstante, es cierto que el acceso a estos servicios no es sencillo, se necesita conocimientos en la plataforma e ingenieros especializados para poder realizar proyectos de envergadura. A cambio se ofrece potencia de cálculo y escalabilidad a niveles difícilmente alcanzables por empresas más pequeñas.

Imagen 5 - Azure IoT Core - Fuente azure.microsoft.com (*Microsoft, Explore the Benefits of Azure IoT - Azure.Microsoft.Com*, n.d.)

A continuación, se listan una serie de casos de uso populares en el entorno de *IoT* de Microsoft:

- Mantenimiento preventivo Industrial
- Edificios seguros
- Energía
- Fabricación discreta
- Cuidado de la salud
- Fabricación de procesos
- Monitoreo de condiciones para *IoT* industrial
- Transporte y logística
- Análisis intensivo de datos

2.5.4. PTC ThingWorx

La relación de ThingWorx con el *IoT* está orientada sobre todo a entornos industriales o *IIoT*. Las soluciones que se ofertan son muy completas para una actuación industrial, además, son capaces de gestionar una industria 4.0 de manera muy eficiente, lo que los convierte en líderes mundiales en entornos industriales.

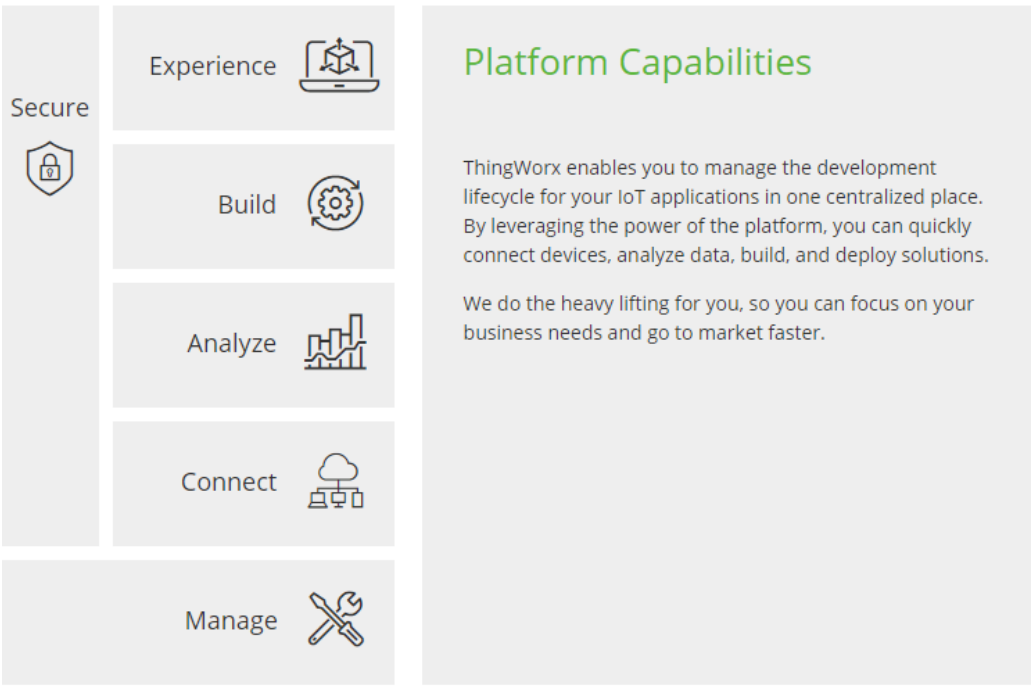


Imagen 6 - PTC ThingWorx Core – Fuente developer.thingworx.com(*ThingWorx Industrial IoT Platform - Developer.Thingworx.Com*, n.d.)

La plataforma ofrece numerosos servicios de conectividad, API´s de comunicación, interfaces de desarrollo y mucha experiencia en el sector.

Se pueden diferenciar varias aplicaciones y casos de uso de su plataforma:

- Monitorización remota de condiciones
- Visibilidad en tiempo real de procesos industriales
- Producto como servicio
- Instrucciones de trabajo digitales y retroalimentación de procesos
- Benchmarking de plantas y análisis intensivo de productividad
- Control de sistemas remotos

Existen también numerosas webs y servicios para aficionados que suministran entornos funcionales, pero tiene numerosas carencias profesionales tanto en escalabilidad, redundancia y configuraciones.

2.5.5. Comparativa de soluciones

En el siguiente cuadro, se resume la información útil obtenida de la documentación oficial de las 3 plataformas líderes del sector. Los campos estimamos que son los más útiles para el planteamiento y desarrollo de este TFG.

Servicio	Protocolos	Plataformas Certificadas	Docum.	Precio
Amazon	HTTP, MQTT	Broadcom, Marvell, Microchip, Intel, Mediatek, Qualcomm, Seeed, Texas Instruments,…	Excelente	Conectividad, Mensajes, Operaciones
Google	HTTP, MQTT	Raspberry PI, Beaglebone, Arduino, Intel, Microchip, NXP, Sierra Wireless….	Muy Buena	Volumen de datos
Azure	HTTP, AMQP, MQTT	Intel, Raspberry PI, Freescale, Texas Instruments, MinnowBoard, BeagleBoard, Seeed, Resin.io,….	Excelente	Precio por dispositivo y mensajes de estos

Tabla 2 - Comparativa sistemas IoT

Como se puede apreciar, hay diferentes solución y plataformas preparadas para el mercado, esto nos ayuda a discernir patrones y estándares de funcionamiento de estas para implementar este TFG.

2.6. Necesidades detectadas en los entornos

Analizando la realidad actual de los sistemas de *IoT* y de las empresas, queda claro que los sistemas actualmente no son accesibles a muchas industrias propensas a implementarlos. Hay una gran brecha tecnológica entre las empresas y la implementación de sistemas competentes que puedan dar ventajas competitivas.

Las grandes industrias y entornos de gran capacidad económica pueden acercarse ya sea por su infraestructura, por su conocimiento colectivo o por sus fuertes medios económicos a los sistemas punteros, mediante grandes desarrollos tecnológicos y procesos de transformación digital profundos.

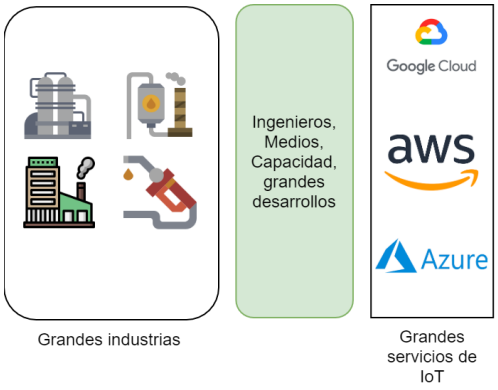


Imagen 7 - Grandes Industrias e IoT

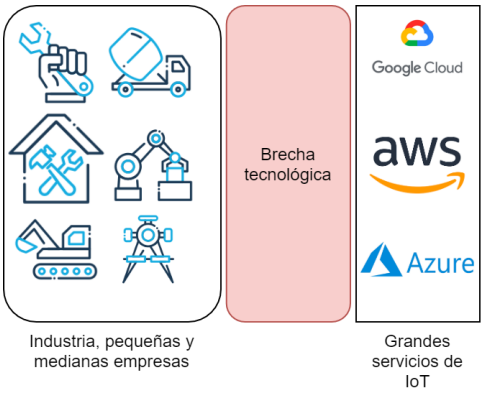


Imagen 8 - Brecha tecnológica en el IoT

Sin embargo, quedan fuera de estas implementaciones las pequeñas industrias con potencial de aprovechamiento. Estas pretenden conseguir una ventaja competitiva, aunque su realidad es que carecen de los medios exigidos para implementar dichos sistemas.

Esta situación provoca una brecha tecnológica de accesibilidad, los sistemas y la tecnología está ahí, sin embargo, son poco accesible y complicados de implementar.

El hueco existente entre la realidad de la industria y la comunicación de las cosas para obtener valor de negocio es donde este TFG pretende actuar y analizar las posibilidades de sistemas más sencillos y al alcance de la gran mayoría de las empresas, sacrificando potencia de cálculo, escalabilidad y complejos análisis, por accesibilidad y economía.

2.7. Sensores ¿Qué se puede medir?

En *IoT*, los sensores y actuadores están en la vanguardia del sistema, son los encargados de medir o actuar sobre las “Things” y aportan toda esta información a las capas superiores.

Existe un amplio abanico de posibilidades, la electrónica y el avance imparable del *IoT* ha hecho posible medir prácticamente cualquier valor, a continuación, se presenta una pequeña parte de estos posibles valores para tener en cuenta, aunque teniendo presente que es posible ampliar estos a prácticamente cualquier cosa.

- Sensores de Proximidad, presencia, posición o velocidad

Estos sensores son capaces de determinar mediante tecnologías variadas si hay algún objeto en una posición concreta, si este objeto está cerca o lejos, o incluso su posición relativa. Son muy útiles en aplicación de seguridad, aplicativos de análisis de personas etc.

- Sensores de Temperatura y humedad

El valor de temperatura o humedad de cualquier situación es un valor clásico para infinidad de situaciones tanto domesticas como industriales.

- Sensores ópticos.

Otro tipo sensor clásico de la industria, son ampliamente utilizados para detectar la cantidad de iluminación de un ambiente.

- Sensores de sonido o vibración.

En función de la presión sonora se pueden enviar señales a las plataformas para tomar decisiones acordes a los parámetros, por ejemplo, en situación de alarmas de sonido, vibración en estructuras críticas, etc.

- Sensores de detección de corrientes eléctricas o magnéticas.

Muy utilizados en el sector electico, permiten por ejemplo medir la cantidad de corriente eléctrica (Amperios) que está recorriendo un conductor.

- Sensores de detección de gas o fluidos químicos.

Estos sensores permiten detectar ciertos componentes químicos para los que han sido configurados, conforme se detecta presencia de estos, el sistema podría por ejemplo avisar a un operador para que pueda actuar en consecuencia.

- Sensores de detección de flujo.

Estos sensores se utilizan para medir flujo de ciertas sustancias como agua, aire, combustibles, etc.

- Medidores de Presión o Fuerza.

Utilizados en la industria en multitud de maquinarias. Estos sensores permiten determinar el trabajo o la fuerza a la que es sometida un elemento.

- Medidores de Nivel.

  Sensores capaces de detectar el nivel de un determinado fluido. Se utilizan en aplicaciones hidrográficas o em medidores de contenedores.

  Un uso clásico de estos sensores son los depósitos de materias primas de las fábricas, estos avisan de niveles bajos/altos de material.

Como se puede apreciar las magnitudes son muy variadas, se puede desarrollar sensores para medir muchos parámetros y situaciones, no obstante, en el desarrollo de un sistema genérico, los más adecuado es centrarse en aquellos que sean más populares y ampliamente utilizados (temperatura, entradas/salidas, tensión electica, etc..).

2.8. Comunicación en *IoT*

En *IoT* el entorno de operaciones de área local y área extendida es una elección importante, hay que tener en cuenta cómo van a conectarse las diferentes tecnologías y las necesidades de los sistemas.

Para conseguir una comunicación satisfactoria con los servidores de gestión *IoT*, las *Things* deben contar con la infraestructura necesaria para poder desarrollar su cometido, a continuación, se detallan las tecnologías más utilizadas en el mercado y sus respectivas características con el afán de dar una visión panorámica del entorno.

2.8.1. Tecnologías de área local LAN

La conexión de las things hacia internet comienza por el área local, la evolución de la tecnología ha permitido conectarnos de forma masiva y el *IoT* ha aprovechado este aspecto para crecer con fuerza.

Hay que tener muchos aspectos en cuenta a la hora de seleccionar una u otra tecnología, no obstante, se van a describir los aspectos más importantes y su impacto en un posible sistema final:

- Banda de frecuencia:

  Las conexiones inalámbricas que conocemos hoy se transmiten por el aire median ondas que se emiten en una determinada banda de frecuencia. Los rangos en los que actúa cada conexión determinan el nivel de penetración de la conexión inalámbrica, la distancia de alcance y el consumo eléctrico que la conexión exige, por tanto, es un parámetro muy relevante.

Tiene un impacto directo en el coste, la capacidad de flujo de datos, el consumo de los equipos y la tolerancia a obstáculos.

- Ancho de banda o Throughput:

Este valor indica la capacidad de transmisión de datos en relación con el tiempo que la conexión puede alcanzar.

Hay que considerar este parámetro en conjunto con los demás valores críticos, ya que puede ser determinante en equipos de envío masivo de datos.

- Alcance:

El alcance es determinante a la hora de poder desplegar diferentes equipos alrededor de un único punto de acceso a una WAN. Un gran alcance puede determinar menor coste en conexiones y mejor rendimiento energético (bajas frecuencias) pero menos ancho de banda.

- Consumo:

Dependiendo de la potencia de las antenas de os dispositivos y el rendimiento de estas, los equipos pueden tener una enorme variación en la potencia necesaria para su funcionamiento.

Las tecnologías de comunicación LAN más utilizadas son WIFI y Bluetooth, aunque en IoT se usan otras menos conocidas como Zigbee y LoRa.

Zigbee se basa en el estándar de la IEEE 802.15.4 y es una red inalámbrica muy utilizada en *IoT* ya que posee capacidades de conexión en estrella y *mesh*. Los nodos pueden interconectase entre si formando una gran red y solo uno de estos nodos (como mínimo) debe estar en contacto con una salida a internet.

Esta tecnología opera en la banda 868 MHz – 2,4 GHz y puede alcanzar una velocidad de 250 kbps en buenas condiciones, todo ello manteniendo un consumo muy reducido en comparación con WIFI (similar a Bluetooth).

Una de sus principales desventajas es que es débil a la hora de atravesar obstáculos, ya que estos reducen enormemente los 100 metros de alcance teórico que tiene.

En cuanto a LoRa, es una red conocida como *Low-Power Wide Area Network* (LPWAN) y pude utilizarse para WAN o LAN según las necesidades ya que el alcance de un entorno LoRa suele ser de unos 15 kilómetros.

La red Lora emplea un tipo de modulación en radiofrecuencia patentado por la empresa *Semtech*. Se denomina *Chirp Spread Spectrum* (o CSS) y se emplea en comunicaciones militares y espaciales desde hace mucho tiempo. Opera en 3 bandas (169MHz, 433 MHz y 868 MHz) y el consumo es extremadamente bajo (unas 800 veces más bajo que WIFI) aunque en contrapartida la velocidad obtenida es también muy baja (10kbps) aunque suficiente para enviar datos de sensores.

Tecnología	Ancho de banda	Consumo	Alcance
Bluetooth	0,100 - 2 Mbps	Bajo	10 m
WIFI	11-1300 Mbps	Alto/Moderado	30 m
Zigbee	15 - 260 kbps	Bajo	100 m
LoRa	10 kbps	Muy bajo	15 km

Tabla 3 - Comparativa Tecnologías LAN

2.8.2. Tecnologías WAN

La conexión a las áreas extendidas proporciona a los entornos *IoT* una puerta de acceso para publicar datos hacia los servidores capaces de darles valor. Hoy existen multitud de tecnologías en el mercado que se pueden seleccionar y conviene hacer un repaso de estas para conocer sus características.

GSM es el primer escalón de las tecnologías de comunicación móviles modernas, es una versión antigua de comunicación inalámbrica, pero plantea una ventaja, su bajo consumo frente a otras, aunque su ancho de banda es bastante reducido (20kbps). Emite en 900/1800 MHz y es bastante utilizada en *IoT* en aplicativos que no necesitan excesiva velocidad de conexión.

El 3G es la evolución de GSM, no es muy utilizada hoy en día ya que el consumo es muy elevado comparado con GSM, no aporta grandes ventajas de comunicación ya que las nuevas generaciones (4G, 5G…) son más eficientes en todos los aspectos. No obstante, sí que existe un gran parte de dispositivos que utilizan aun 3G y es una opción económica, aunque sus 2 Mbps de ancho de banda limitan su uso.

Por otro lado, el 4G es el presente de las comunicaciones móviles, ampliamente utilizada en todos los lugares del mundo y con una velocidad muy superior a 3G, alcanzando los 100 Mbps. El hándicap del 4G sigue siendo el consumo eléctrico, ya que es bastante elevado para *IoT*, aunque hoy es la solución óptima en la ratio *througput*/consumo (sin contar con el 5G).

Lora WAN es utilizada también en área extendida y las características son las mismas a las expuestas en el apartado 2.9.1.4. Hay que tener en cuenta este tipo de conexiones en comunicación WAN.

Finalmente, la tecnología más avanzada y nueva, el 5G. Es el futuro de las conexiones móviles inalámbricas, la tecnología está en expansión y es otro de los revulsivos para el *IoT*, no solo por su ancho de banda de 1 Gbps, sino también por su bajísima latencia de menos de 1ms, lo que le confiere las propiedades perfectas para las comunicaciones en tiempo real de manera inalámbrica y deslocalizada. El consumo también es muy reducido en comparación a las otras tecnologías.

Por nombrar una desventaja, la red aún no está muy extendida y se focaliza en grandes ciudades, además su precio es mayor que el de una conexión 4G. No obstante, en cuanto el despliegue sea masivo el 5G cambiará el panorama del *IoT*.

Tecnología	Ancho de banda	Consumo	Precio
GSM	20 kbps	Bajo	Bajo
3G	2mbps	Alto	Bajo
4G o LTE	100 Mbps	Medio	Medio
5G	1 Gbps	Bajo	Alto

Tabla 4 - Comparativa comunicaciones

Las opciones son muchas y el ámbito muy variado, como se ha visto, la elección de las comunicaciones es un punto para tener en cuenta en *IoT*, aunque cada caso es distinto, centrarse en una buena selección de comunicaciones puede suponer el éxito o el fracaso de un proyecto.

2.9. Protocolos de comunicación

Para poder comunicar dos entidades del sistema de una manera bidireccional, es necesario que los sistemas y la plataforma puedan entenderse de manera óptima y estructurada. Es aquí donde entran en juego los protocolos de comunicación.

Los protocolos nos permiten establecer a nivel de aplicación una comunicación donde los diferentes elementos puedan enviar y recibir datos, actúan como intermediarios entre el software de *Backend* y el de las things, todo a su vez apoyado sobre la capa de comunicación física que se ha comentado en el apartado anterior.

Hay que destacar que en *IoT* el *multicasting* es importante, ya que la situación de multidifusión de mensajes es muy habitual en los entornos conectados.

Existen 3 grandes protocolos que se utilizan en *IoT*, en el punto 2.5 de este documento se han nombrado ya que son soportados por todas ellas.

2.9.1. HTTP

Es muy utilizado en internet y la utilización de este para *IoT* es heredado de su propia popularidad, aunque no es la mejor opción para la comunicación ya que no está específicamente diseñado para los entornos actuales de internet de las cosas.

Se basa en petición GET a un servidor especifico y en la respuesta de este, es decir, sigue una arquitectura cliente servidor.

Los sistemas de *IoT* que funcionan con HTTP se aprovechan de la madurez y robustez del protocolo ya que es ampliamente utilizado hoy, sin embargo, tiene muchas carencias de configuración como calidad de servicio, escalabilidad, y plantea otras complicaciones heredadas de la adaptación de los entornos para este.

2.9.2. MQTT

Protocolo extremadamente ligero de comunicación basado en un modelo de suscripción/publicación. A diferencia de HTTP, MQTT si está pensado para *IoT* y es muy eficiente a la hora de gestionar tanto el ancho de banda como los eventos asociados.

Necesita un servidor *(bróker)* que es quien gestiona las publicaciones y suscripciones a los *topics* o temas. Las things generalmente publican información y los clientes o las propias things se subscriben a estos. El protocolo se basa en TCP, por tanto, es muy seguro a la hora de la recepción de la información.

Está pensado para funcionar en redes con una fiabilidad baja y un ancho de banda reducido, por tanto, es ideal para sistemas de *IoT* distribuidos, además implementa calidad de servicio con prioridad en los mensajes.

Es un estándar en la industria, lo cual lo ha hecho extremadamente popular y hay mucha información además de librerías que ayudan a los desarrolladores a implementarlo en los desarrollos.

Desde 2014 MQTT se ha convertido en un estándar OASIS, cabe destacar la importancia de este hecho ya que permite que aquellos proyectos basados en esta tecnología puedan desarrollarse con ciclos de vida correctos y tengan continuidad.

Además de ser un estándar, MQTT es de código abierto y su código es público a la vez que gratuito.

2.9.3. Funcionamiento MQTT

Se basa en un servicio de mensajería *push* (publicación/suscripción) donde un equipo principal o bróker intercambia mensajes y datos con los subscriptores y publicadores.

Las colas de mansajes en MQTT se gestionan mediante temas o *topics*, los clientes del sistema pueden publicar información en un determinado tema o simplemente subscribirse a cualquiera de estos y ser informado de cada publicación.

Normalmente los sensores solo publican en los temas que necesitan información, mientras que los actuadores/servicios o clientes juegan en ambos roles.

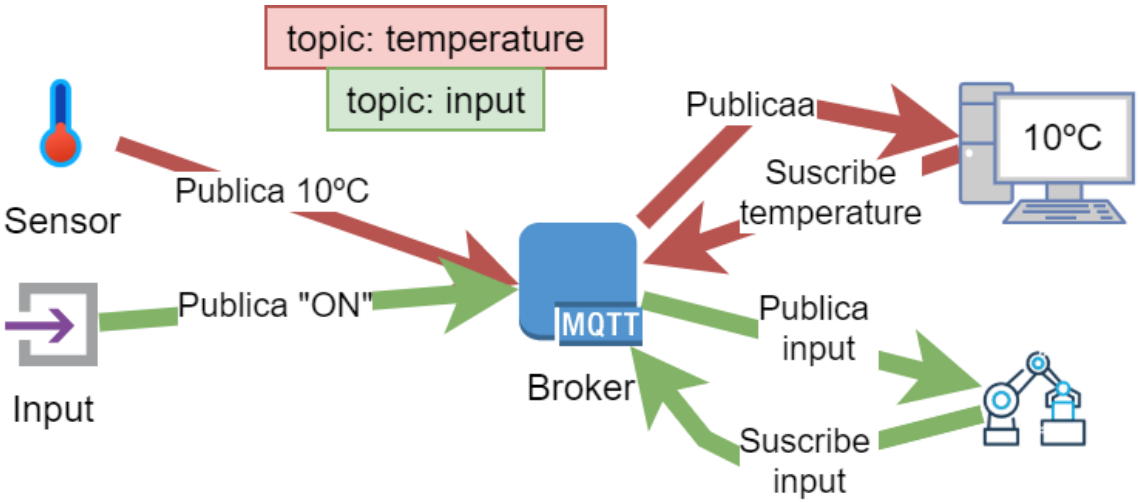


Imagen 9 - Funcionamiento MQTT

El *Broker* es el elemento central, gestiona todos los clientes que están publicando/suscribiéndose a los *topics*. Este mantiene una lista de los equipos que tienen una conexión abierta, gestionada por mensajes CONNECT y CONNACK que contienen la información necesaria para establecerla y mantenerla.

En este tipo de arquitectura MQTT el emisor no sabe a quién va dirigido el mensaje que envía, sino que solo el bróker lo sabe, este mantiene en memoria a los subscriptores de cada uno de los *topics*.

2.9.4. Temas o *topics*

El tema es la relación del mensaje con su contenido, es decir, describe de manera estructurada que es lo que se está publicando. Se organiza por niveles y los inferiores son subniveles de los principales de forma jerárquica.

Utiliza una sintaxis de separación por el carácter “/” el cual marca el inicio y el fin de uno de los niveles.

Se puede describir como ejemplo un edificio con dos plantas y diferentes sensores de temperatura por cada planta:

- El tema principal sería “edificio/”
- Cada una de las plantas:
- “edificio/panta1”
- “edificio/panta2”
- Los sensores como:
- “edificio/panta1/sensor1”
- “edificio/panta1/sensor2”
- “edificio/panta2/sensor1”
- “edificio/panta2/sensor2”.

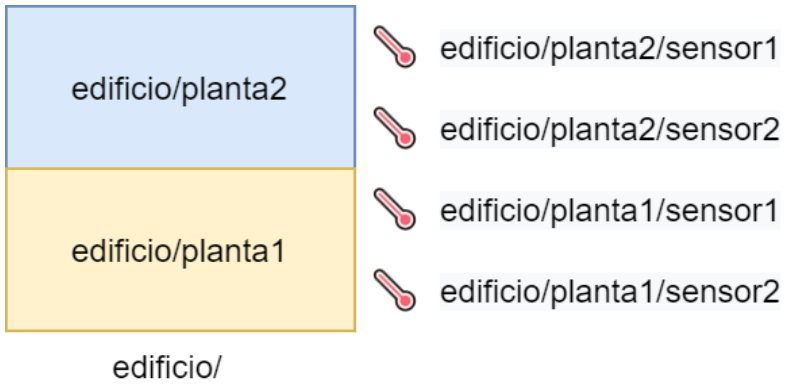


Imagen 10 - Ejemplo distribución Topics

De esta manera se obtiene una estructura por niveles con la cual se puede interactuar mediante el bróker.

En MQTT también existen comodines que se pueden utilizar para subscribirse a niveles completos o a todos los temas de un nivel:

- El símbolo + se sustituye por cualquier nivel. (edificio/+/sensor1 por ejemplo para todos los sensores 1 de todas las plantas)
- El símbolo # es un comodín. Este símbolo cubre varios niveles aguas abajo.

2.9.5. Estructura de un mensaje MQTT

Según la documentación oficial de MQTT (MQTT, n.d.) , los mensajes se componen de 3 partes, un encabezado (obligatorio) de 5 Bytes como máximo, otro encabezado variable (no obligatorio) variable en tamaño, y finalmente el mensaje o *playload* que puede tener como máximo 256 Mb aunque lo habitual es implementar pequeños mensajes de 2 kb.

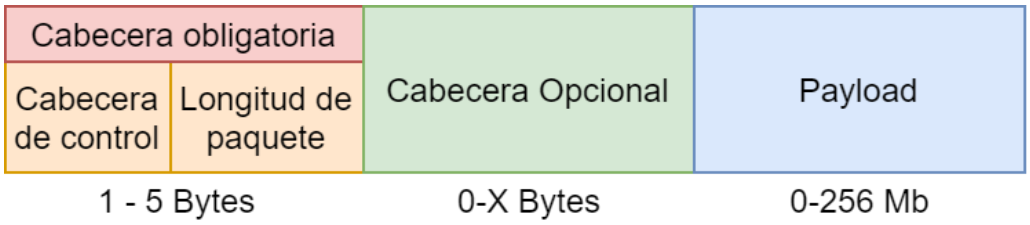


Imagen 11 - Mensaje MQTT

Esta estructura simple, permite enviar mensajes de manera muy eficiente y con una carga de trabajo tanto computacional como de red relativamente baja.

2.9.6. QOS – Calidad de servicio en MQTT

Los mecanismos de calidad de servicio que gestiona MQTT son muy importantes a la hora de desarrollar software que interactúe con el *Broker*.

En *IoT* las conexiones a menudo fallan, y los entornos muchas veces no son óptimos.

QoS da a los mensajes un tratamiento distinto según la importancia de este, pudiendo garantizar la entrega de estos a costa de incrementar el tráfico para una interacción.

Existen 3 niveles de calidad de servicio(OASIS, 2014):

1. *QoS 0 unacknowledged:*

   El mensaje es enviado una única vez al bróker, si la entrega falla, es posible que el mensaje se pierda. Se utiliza cuando el consumo de datos debe ser mínimo y el envío es masivo o carece de importancia estratégica.

2. *QoS 1 acknowledged:*

   En este nivel de QoS se garantiza la entrega, el tráfico aumenta, pero se asegura la recepción de los mensajes.

3. *QoS 2 assured:*

   Este es el nivel superior de calidad y es poco utilizado, este nivel garantiza que el mensaje es entregado al subscriptor una única vez. Requiere gran ancho de banda y consume recursos tanto del Broker como del subscriptor.

Como se aprecia, conforme aumenta la calidad, aumenta el número de mensajes necesarios para realizar la comunicación y por ende el sistema requiere más recursos tanto de red como computacionales.

2.9.7. AMQP

Es similar a MQTT con la diferencia que implementa una gestión más eficiente de las colas de mensajes. El protocolo TCP en el que se basa garantiza la entrega optima de paquetes con control de errores, además está bastante extendido en la industria.

Por el contrario, es algo más costoso de implementar ya que es más complejo que MQTT, además consume más recursos computacionales en la gestión.

3. Modelo de negocio valor y propuesta

3.1. Ideas de negocio basadas en *IoT*

Se han analizado muchos sectores donde se puede realizar una gran aportación en *IoT*, el sector está en pleno auge y hay multitud de nichos que cubrir.

¿Cómo se aporta valor? Para responder a esta pregunta es necesario realizar más preguntas, la reflexión sobre que se puede medir, el cómo se va a hacerlo y para que se quiere hacer es seguramente el proceso más complejo de llevar a cabo.

Los sistemas, la tecnología y el entorno es propicio para que las ideas se desarrollen sin barreras tecnológicas considerables, por tanto, trabajar en estas 3 preguntas es un trabajo necesario para conseguir un producto apropiado además de viable.

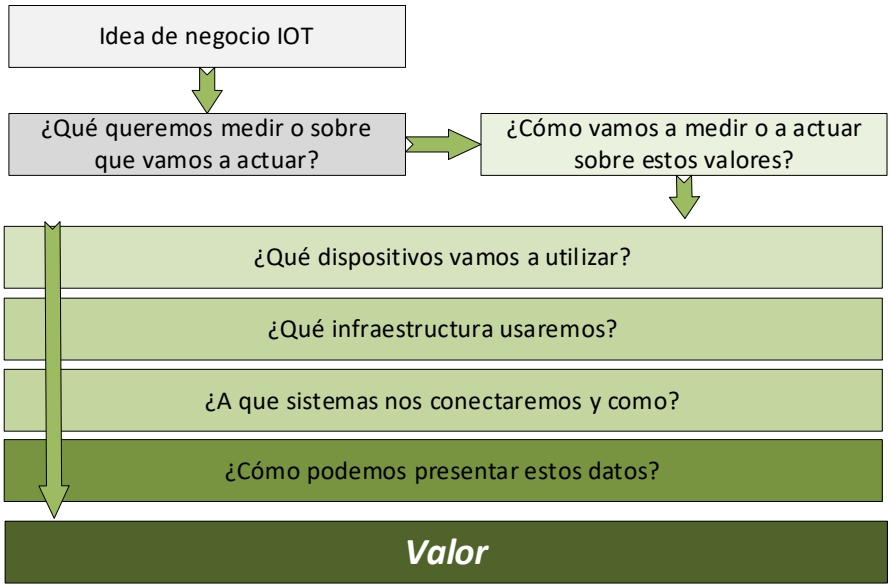


Imagen 12 - Modelado de idea de negocio

3.2. Valor añadido de la propuesta

Son muchas las consideraciones para tener en cuenta en el diseño de un sistema *IoT*, la casuística es infinita y los modelos que aportan valor pueden enfocarse de muchas maneras diferentes.

En el caso de este TFG se centrará el diseño en una plataforma genérica de *IoT*, utilizando la información para construir una arquitectura que facilite el acceso de las empresas a una gestión amigable, sin perder la versatilidad que nos pueden dar las diferentes tecnologías.

La idea es que el sistema sea muy modular y adaptable, pudiendo aprovechar desarrollos que puedan pedirnos para ofertarlos a más empresas pudiendo reaprovechar los desarrollos con estrategias de sinergia empresarial.

¿Cómo se aporta valor?

- ¿Qué se va a medir o donde se va a actuar?

El fin último es proporcionar una plataforma integral de IoT, que abarque los entornos más utilizados y sencillos sobre los que poder actuar, aunque sin cerrar la puerta a futuros sensores a los que integrar en el sistema con forma los clientes lo soliciten.

- Actuar sobre salidas

  Poder encender un simple relé en un sistema es algo básico en cualquier entorno, ya que esto es la manera de conectar cualquier plataforma con el mundo físico.

  Hay infinidad de ejemplos, desde bombillas hasta cualquier sistema que funcione con electricidad. Todos son propensos a interactuar mediante salidas.

- Leer entradas

  Otro de los imprescindibles en *IoT*, la mayoría de los sistemas industriales cuentan con salidas libres de potencial que indican a los sistemas diferentes situación del entorno. El estado de estas salidas vuelve al entorno inteligente y le da información para tomar decisiones sobre otras salidas/sensores o enviar avisos.

- Leer temperaturas/Humedad

  El hecho de conocer la temperatura o la humedad de un establecimiento o un entorno y poder actuar en consecuencia es también otro de los grandes nichos de mercado de *IoT*.

  Hoy, en un mundo deslocalizado y con cada vez más sistemas automatizados, el poder tomar decisiones conforme a estos parámetros es algo imprescindible.

- Leer flujo eléctrico (Amperios)

  La monitorización eléctrica y el estado del consumo de las instalaciones de las empresas es un valor para tener en cuenta para dar consistencia a nuestro entorno y tener una oferta variada, más aún desde la reciente regulación de las eléctricas y productoras que obliga a tener digitalizados sus sistemas.

- Demas sensores

  Muchos de los sensores que se nos puedan ocurrir a monitorizar (gases, humos, alarmas etc.) son simples salidas que se pueden medir mediante el punto 2, por tanto, para el prototipo será suficiente para validar el sistema.

- ¿Cómo se va a medir o a actuar sobre estos valores?

En el caso presente de este TFG se pretende conseguir un sistema genérico y adaptable por tanto habrá que tener en cuenta que las situaciones pueden ser muy cambiantes, por ende, se intentará generalizar en el diseño dejando abierta la puerta a particularizar el sistema si se requiere, pero sin perder de vista el objetivo principal que es llegar a la gran mayoría de empresas no especializadas.

- ¿Qué dispositivos se van a utilizar?

  El dispositivo debe ser versátil, hoy la microelectrónica no supone un gasto tan elevado, no obstante, intentar utilizar dispositivos que no se queden limitados de potencia en un corto plazo puede ser una opción bastante acertada.

- ¿Qué infraestructura de red se usará?

  Al igual que en la anterior pregunta se tiene que ser generalistas, en apartados anteriores hemos visto los tipos de conexiones *IoT* tanto LAN como WAN, muchas de ellas muy eficiente y potentes, pero bastante poco comunes.

  Por tanto, una elección de tecnologías WIFI y conexiones 4G o de fibra óptica habituales en todas las empresas/hogares, sea una idea acertada para el desarrollo del prototipo, expandir la funcionalidad del sistema a conexiones Zigbee o LoRA queda fuera del alcance de este TFG.

- ¿A qué sistemas se conecta y cómo?

  Los sistemas comerciales hoy en día son una realidad en *IoT*, se pueden almacenar o tratar datos en muchas plataformas variopintas y hacer una gestión de estos muy buena en cualquiera de ellas, no obstante, el control y versatilidad de utilizar nuestra propia plataforma con la que tomar nuestras propias decisiones de diseño y desarrollo es una de las principales metas de este TFG, por tanto, la elección de los sistemas y el desarrollo del *Backend* de todo el sistema forman parte de este documento.

Aun teniendo nuestro propio sistema de gestión *IoT*, siempre se puede conectar con las grandes plataformas para aprovechar la inconmensurable potencia de cálculo de estas si alguno de los desarrollos lo necesitara.

- ¿Como se pueden presentar estos datos?

  Presentar los datos forma parte del proceso y del ciclo de creación de un sistema *IoT*, el poder ver en tiempo real los valores, o poder actuar sobre según que sistemas será clave en el éxito de prácticamente todas las aplicaciones que utilicen el sistema.

  En un entorno multi-cliente, donde se necesite agilidad de desarrollo y presentación deslocalizadas además de acceder desde cualquier sitio, un entorno web es la mejor solución posible. Son entornos muy robustos y maduros donde se puede presentar cualquier tipo de dato o modelo de una manera organizada.

En el siguiente esquema se plantea la propuesta:

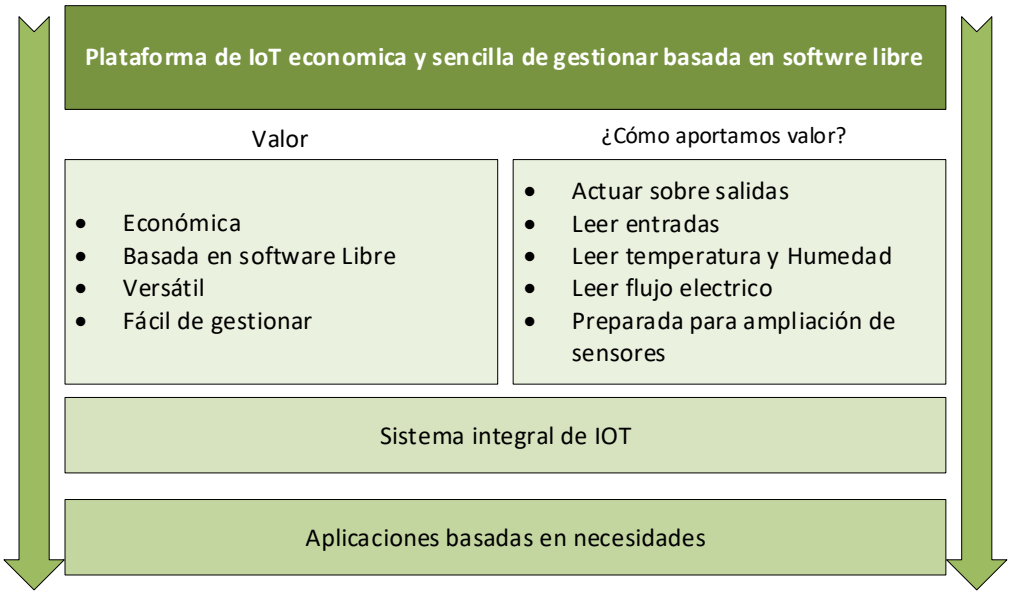


Imagen 13 - Propuesta de negocio

La parte principal de este TFG es crear una plataforma completa de IoT con la que se puedan gestionar las things y tratar los datos de estas para aportar valor. Sobre la plataforma es donde se construyen las aplicaciones concretas que aportan funcionalidades extra, en este caso, una aplicación de control de ambiente en CPD´s y otra para monitorizar datos de intensidad en cuadros eléctricos, pudiendo instalarse y reutilizarse estas para ofertarlas como servicio de la plataforma.

4. Diseño del sistema – Arquitectura

4.1. Arquitectura *IoT* general

Los sistemas de internet de las cosas abarcan un gran abanico de posibilidades tecnologías tanto de infraestructura como de implementación. Se ha analizado el mercado y definido objetivos, modelando a alto nivel el prototipo que permita cumplir con las necesidades registradas.

Las arquitecturas *IoT* siguen un patrón de diseño por capas donde cada una de estas da soporte a la siguiente. Todas estas juntas forman un sistema completo como el que se propone en este TFG.

- En una primera capa están los dispositivos y sus respectivos sensores (things)
- Justo por encima de esta, la conectividad, tanto LAN como WAN
- Posteriormente está la plataforma de IoT
- Y encima de esta, los paneles de cliente

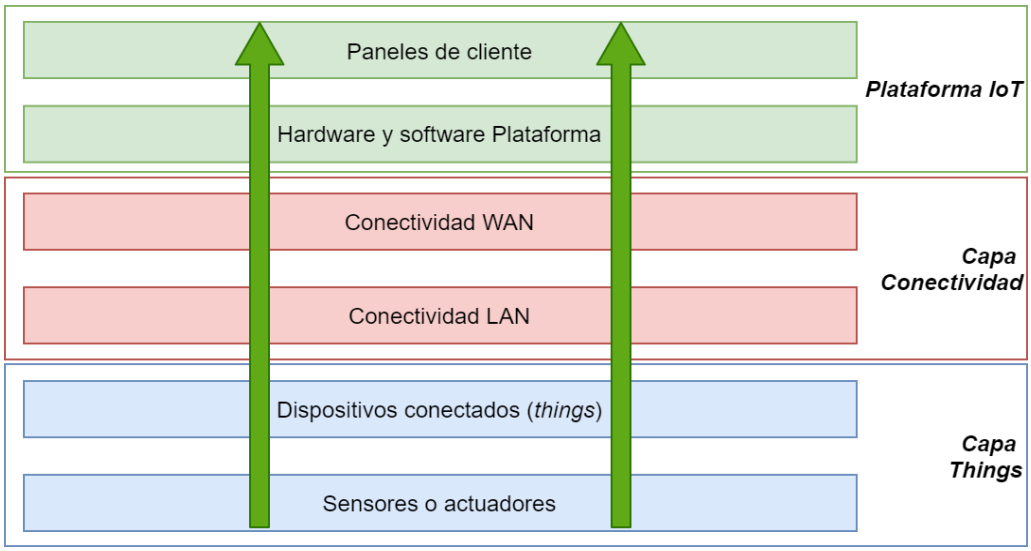


Imagen 14 - Arquitectura básica IoT - estrutura de capas.

Si se analiza las plataformas comerciales, se obtiene un patrón lógico estructurado de los sistemas basado en niveles como el que se expone, estos niveles van desde los sensores que detectan eventos o actúan con el entorno, a finalmente las plataformas de *IoT* donde son o bien presentados a los clientes, o también analizados por el propio sistema que toma decisiones conforme a los datos (reglas).

A continuación, detallamos estos niveles y su interconexión:

- Sensores o actuadores (1):

En este primer nivel se encuentran los sensores o actuadores que recogen datos o interactúan con el entorno, son la clave de todo esto, donde se colocan, que es lo que miden o sobre qué elementos actúan son la clave del valor del sistema, y la decisión más importante a tomar en cada uno de los entornos.

- Dispositivos conectados (*things*) (2):

El nivel de las *things* se complementa con los sensores (podríamos agruparlos en un todo uno) y se compone del hardware y software capaz de interpretar los sensores e interactuar con los niveles superiores de conectividad para enviar datos que tengan sentido y puedan ser interpretados.

- Conectividad local o conectividad LAN (3):

Lo normal en *IoT* es que los dispositivos desplegados sobre una infraestructura se conecten con un elemento que pueda dar conectividad hacia internet (puerta de enlace).

Esta capa proporciona estanqueidad frente a Internet y permite abaratar las conexiones dado que se pueden reducir las líneas de conexión necesarias que el sistema puede necesitar.

- Conectividad a internet o conectividad WAN (3 + 4):

Es el escalafón superior de la conectividad y proporciona acceso al sistema desde cualquier lugar del mundo.

Existen numerosos protocolos que pueden utilizarse, aunque normalmente esta capa es transparente y nos la proporciona un operador, aunque sí que es importante tener en cuenta el tipo de conectividad (fibra óptica, 3G, 4G, 5G, satélite, etc.) por ancho de banda, velocidad, coste, etc.

- Sistema *IoT* – plataforma (5):

En la parte superior de la jerarquía nos encontramos la plataforma de *IoT*, esta engloba toda la arquitectura (software y hardware) que dota al sistema de la capacidad para realizar la gestión de datos, dispositivos, paneles de control, análisis etc.

Este apartado puede englobarse en el apartado 4, sin embargo, tiene la suficiente importancia como para mencionarlo en el nivel superior ya que es la puerta de acceso al valor de los datos.

Estos paneles permiten tanto la gestión como el análisis de las mediciones o actuaciones de las things.

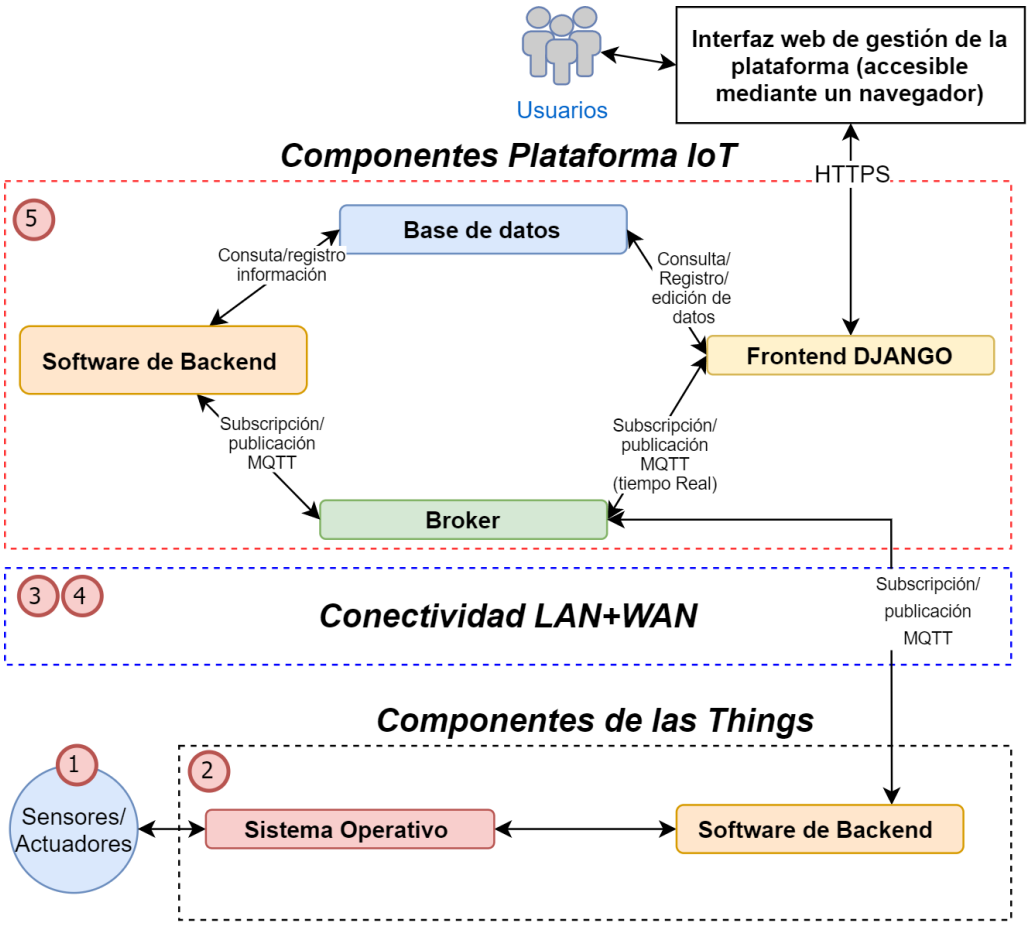


Imagen 15 - Interconexión entre capas (funcional)

La interconexión entre las diferentes partes del sistema se ilustra en la imagen 15 y la imagen 16 desde diferentes perspectivas, una funcional y otra de infraestructura. Cabe destacar que la virtud de una plataforma de este tipo es la versatilidad para añadir nuevas funcionalidades (o apps) que necesiten explotar nuevos datos, sensores o accione y hacer esto de manera ágil y escalable (nuevas apps, nuevas things, etc..).

Una parte importante para tener en cuenta en este tipo de plataformas (y que dan mala fama al IoT) es la seguridad de las comunicaciones, estas son deslocalizadas y remotas, por tanto, están sujetas a numerosos tipos de ataques e intentos de intrusión que deben contemplarse.

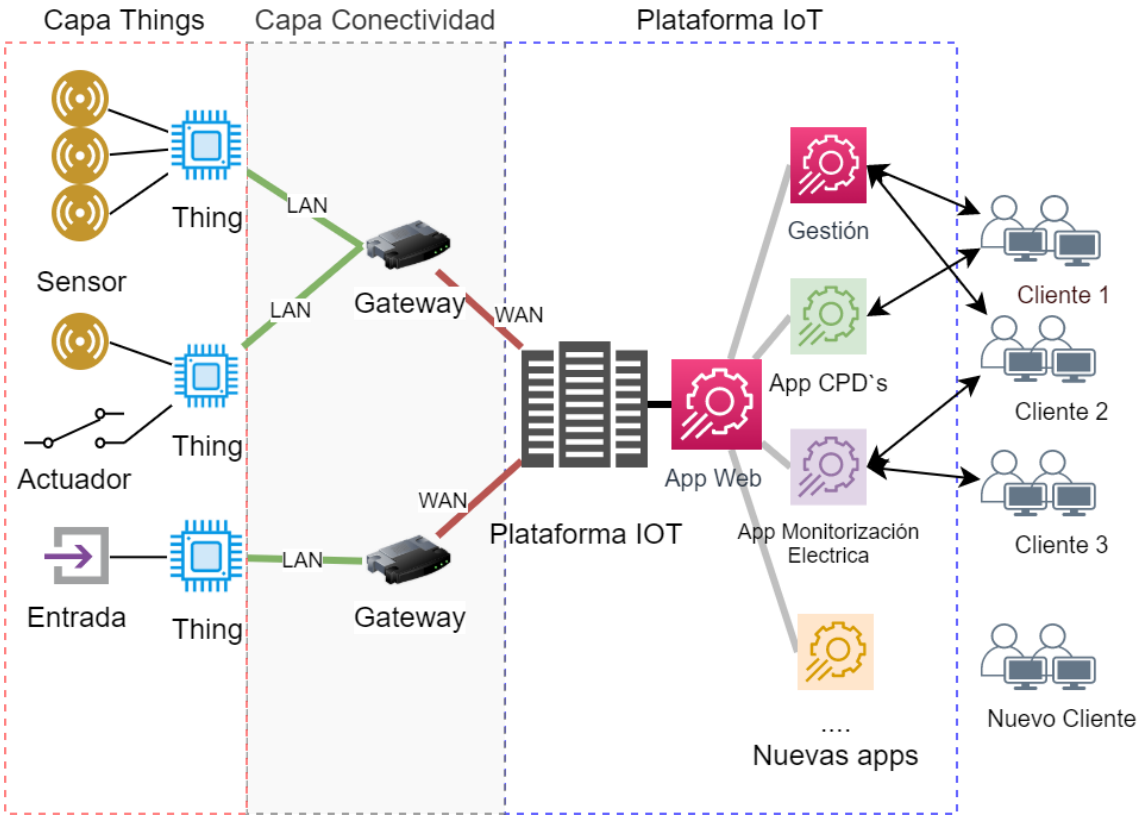


Imagen 16 – Interconexión entre Capas (infraestructura)

4.2. Capa Things

En esta sección la toma de decisiones se divide apartados, los sensores y las propias placas electrónicas de control.

Un sistema de *IoT* debe admitir multitud de modelos de placas y sensores, estos deben seleccionarse según el tipo de proyecto y el alcance de este. En el caso de este TFG seleccionaremos un equipo lo más versátil posible que permita realizar todas las labores de prueba sin requerir modificaciones del hardware. En futuras actualización del sistema se pueden incluir multitud de placas electrónicas con diferentes características que den solución a situaciones más concretas.

Para abarcar el máximo de situación con un coste contenido, necesitamos un equipo que pueda gestionar todo tipo de librerías, que pueda manejar entras, salidas, conexiones serie, wifi y LAN, y que además tenga suficiente potencia de cálculo para afrontar situaciones de entornos complejos.

4.2.1. Sensores y actuadores

En la propuesta de valor se han planteado una serie datos sobre los que se pretende construir el prototipo funcional. Estas entradas y salidas deben concretarse en forma de modelos de sensores comerciales que podamos utilizar en el prototipo.

- Temperatura y Humedad:

Los sensores de temperatura y humedad son muy comunes en el mercado y su precio es muy reducido.

Imagen 17 - Sensor TMP36 (fuente www.aliexpress.com)

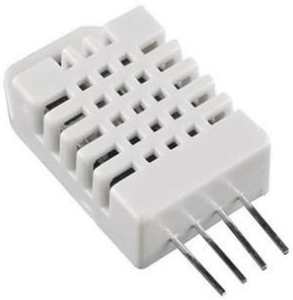


Imagen 18 - Sensor DHT22 (fuente www.aliexpress.com)

- Sensor temperatura TMP36

Este sensor acepta un rango de temperaturas de -40ºC a 150ºC y tiene un coste de 2,5€ aproximadamente. Tiene una precisión suficiente (+-1ºC) y viene calibrado de fabrica para su uso.

Este sensor nos devuelve un valor de voltaje basado en un valor analógico con una equivalencia en ºC. Presenta un factor de escala lineal de 10 mV/ºC que se deberá utilizar para hacer la conversión.

- Sensor temperatura DHT22

Es un sensor algo más caro que el anterior (unos 4,5€), pero tiene una ventaja clara con respecto al TMP36, puede dar valores de humedad. También viene calibrado de fábrica y es muy fiable.

Tiene un rango de temperatura de -40ºC a 80ºC y en mucho más preciso ya que su rango oscila entre +-0.5ºC.

El funcionamiento del sensor se basa en un sensor capacitivo de humedad y un pequeño termistor para medir el aire. La ventaja de este sensor es que es fácil de implementar dado a la inmensa

comunidad que lo usa, existiendo multitud de librerías en varios lenguajes de programación.

Ambos sensores estarán incluidos en el proyecto. El TMP36 por su reducido coste y el DHT22 por su versatilidad y facilidad de uso.

- Salidas de potencia

Para ejemplificar en el prototipo salidas de potencia reales que puedan controlar sistemas u objetos utilizaremos relés electromecánicos comunes.

Estos relés están formados por una bobina que al paso de una corriente crea un campo magnético capaz de atraer una pieza metálica que forma parte del relé, provocando el corte de electricidad.

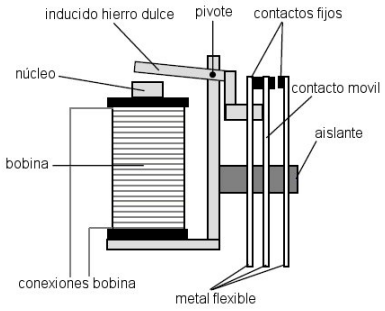


Imagen 19 - Relé - Fuente Wikipedia(*Relé, Derscripcion y Funcionamiento - Es.Wikipedia.Org*, n.d.)

Cuando la corriente para, cesa también el campo magnético, la pieza vuelve a su sitio y la corriente se reestablece.

El precio de estos dispositivos ronda los 5 euros y son muy versátiles en conectividad.

- Lectura Amperaje eléctrico y voltaje

Para realizar la medición de tensión y amperaje por parte de las things recurriremos a las placas RPICT3V1, son circuitos electrónicos preparados para adaptarlos en los proyectos y proporcionan información mediante puerto serie.

Imagen 20 - RPICT3V1 (lechacal.com) (*Data of RPICT3V1 - Lechacal.Com*, n.d.)

Este modelo en concreto consta de 3 entradas para sensores de medición electromagnética de corriente y una entrada de medición de voltaje (V). Toda la gestión se realiza mediante un microcontrolador (Attiny84) y desde la página oficial del fabricante se puede acceder a las instrucciones para leer los valores o reconfigurar la placa. (*How to Program an Attiny85 or Attiny84 - Lechacal.Com*, n.d.)

Las entradas de la RPICT3V1 aceptan para la medición de intensidad sensores SCT-013. Este tipo de sensores no invasivos funcionan como transformadores, la tensión que se desea medir actúa como devanado primario e internamente, el sensor, tiene un devanado secundario que nos proporciona el valor a medir.

La selección de un sistema ya funcional además de sencillo de implementar nos proporciona con un precio contenido (13 €) todas las ventajas de una placa ya probada y funcional, ahorrando mucho tiempo de desarrollo.

4.2.2. Tipos de plataformas Prototyping

Los módulos de desarrollo para las things conllevan en sí mismas una serie de consideraciones de diseño que deben adaptarse al proyecto concreto en el que se está trabajando. Hay que tener en cuenta restricciones de diseño en tamaño, coste, memoria, funcionalidades, consumo eléctrico, etc. Además, hay que analizar con detalle la conectividad, sensores, interfaces de entradas y salidas, potencia de cálculo etc.

Se han considerado 4 grandes familias de dispositivos *IoT* para seleccionar dado su extendido uso en este tipo de proyectos, placas tipo Arduino, Raspberry Pi, procesadores ARM Cortex M (no es una plataforma en sí) y por ultimo los ASICS o dispositivos a medida.

- Placas tipo Arduino, NodeMCU etc.

Estas placas embebidas son muy comunes en proyectos de *IoT*, utilizan varios modelos de procesadores (ATMEGA 328, ATmega2560, …) y carecen de sistema operativo como tal.

Se usan en aplicaciones sencillas y generalmente solo controlan actuadores, pero no se usan para aplicaciones que requieran acciones en tiempo real dado su bajo rendimiento.

En contrapartida son bastante usados y por tanto tienen una comunidad muy extensa, aparte de ser muy compatible con los diferentes sensores que hay en el mercado.

- Rasberry Pi

La Raspberry Pi es un *Single Board Computer* o Pc de placa única (SBC), consta de 40 pines de E/S de uso variado y se puede encontrar por menos de 37 €. Fue desarrollada en Reino Unido por la Fundación Raspberry Pi y en un principio se pensó para la enseñanza.

Consta de procesadores de 16/32 bits y una memoria RAM que va de los 512 MB hasta los 4 GB en modelos muy avanzados, más que de sobra para cualquier proyecto de *IoT*.

En contrapartida no es sencilla de integrar en productos finales dado su tamaño y su consumo es elevado, aunque las últimas versiones de Raspberry Pi Zero y Raspberry Pico solucionan en gran medida estos inconvenientes.

- ARM Cortex M

Estas CPU pueden considerarse un estándar de los sistemas empotrados. Son CPU de 32 bits muy utilizadas y adaptables. Esta adaptabilidad permite afinar en la ratio potencia de cálculo/consumo y conseguir productos muy eficientes.

En contrapartida necesitan un desarrollo electrónico y más trabajo de prototipado.

- Circuito Integrado para aplicaciones específicas (ASIC)

Las ASIC`S son dispositivos diseñados a medida para cada aplicación, no hay límites ni barreras más allá de las propias de los materiales.

Son dispositivos muy específicos además de caros dada la fase de diseño y fabricación. Por tanto, solo se utilizan en desarrollos de *IoT* que requieran un número de unidades muy elevado para compensar los gastos de diseño.

En conclusión, dejaremos fuera del análisis el ARM Cortex M y los sistemas a medida, por tanto, seleccionaremos la plataforma Raspberry por su versatilidad y altas prestaciones a un precio contenido además de su madurez en el diseño, características avanzadas (wifi, LAN, serial, multitud de sensores, potencia de cálculo, etc..).

	Arduino	Raspberry Pi	ARM Cortex M	ASICS
Sistema Operativo	No	Si	Si	Si
Capacidad de proceso	Baja	Media/Alta	Alta	-
Almacenamiento	Bajo	Alto	Depende	Depende
Precio	20-40€	15-50€	Depende	Depende
Conectividad	Media	Alta	Depende	Depende

Tabla 5 - Comparativa Things

No obstante, sería interesante en futuros desarrollos contar en el stock del sistema con SBC´s como NodeMCU o Arduino para despliegues de sensores simples.

Características Raspberry PI:

El equipo Raspberry Pi (*Model* 3b+) consta con una CPU + GPU Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz y 1GB de memoria SDRAM LPDDR2, en cuanto a conectividad consta con Wi-Fi + Bluetooth 2.4GHz y 5GHz IEEE 802.11. b/g/n/ac, Bluetooth 4.2, BLE además de una conexión Gigabit Ethernet sobre USB 2.0 (300 Mbps).

No solo tiene unas características muy buenas para *IoT*, sino que además consta de salida HDMI y 4 USB para conectar periféricos varios.

Por el contrario, el coste es algo más elevado que otros SBC y el tamaño dificulta su integración con sistemas muy reducidos. El consumo tampoco es discreto, pero si es razonable.

Imagen 21 - Raspberry Pi model 3 B+

4.2.3. *Backend* Things

Una vez definido el modelo de las *Things (Raspberry pi),* se puede utilizar el sistema operativo Debian (preparado por el fabricante) que les confiere toda la potencia de un pequeño servidor y permite la ejecución de código pyhton para conectar con el bróker, actuar sobre los sensores y añadir cierta inteligencia al sistema.

Gracias a los 40 pines de entradas salidas y la versatilidad de Python+ Debian se pueden conectar todo tipo de sensores o actuadores en la placa sin necesidad añadir sistemas extra, utilizando el software de *Backend* desarrollado para este TFG compuesto de varios programas separados que se encargan de una función en concreto:

- `iot_dht22_background.py:`
  Lectura regular de sensores DHT22 configurados.

- `iot_input_background.py:`
  Lectura de entradas libres de potencial configuradas.

- `iot_rpict3v1_background.py:`
  Lectura regular de los datos proporcionados mediante puerto serie por la placa de medición RPICT3V1.

- `iot_temperature_background.py:`
  Lectura regular de sensores de temperatura TMP36 configurados.

- `System.py:`
  Script que monitoriza todas las ordenes dirigidas a la thing, y que actúa en consecuencia (encender salidas, lectura forzada de datos, etc.)

Componentes de las Things

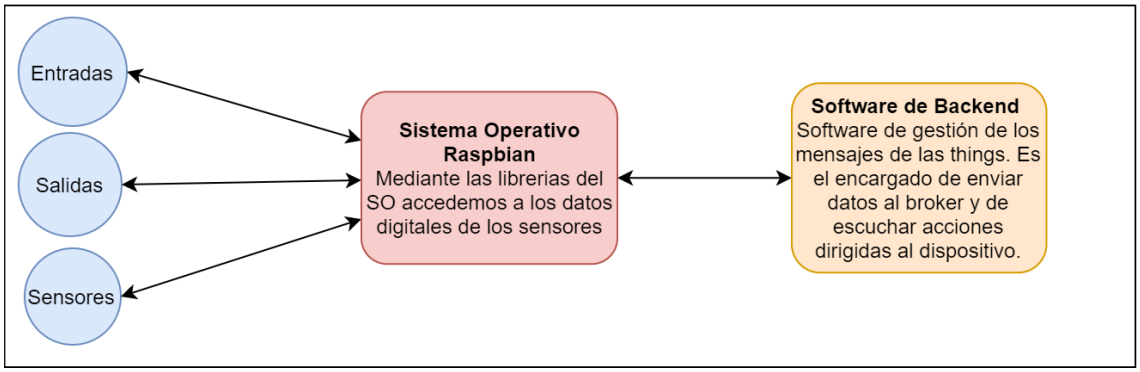


Imagen 22 - Componentes de las Things

4.3. Capa de Conectividad

La conexión LAN en *IoT* es un factor determinante de precio y consumo de energía que puede decantar un proyecto como exitoso o como fracaso. Hemos analizado las diferentes tecnologías y los diferentes modos de conexión para cada situación.

Para el prototipo de conexión del sistema y dadas las características del SBC seleccionado, optaremos por una conexión estándar WIFI o LAN mediante una puerta de acceso a internet.

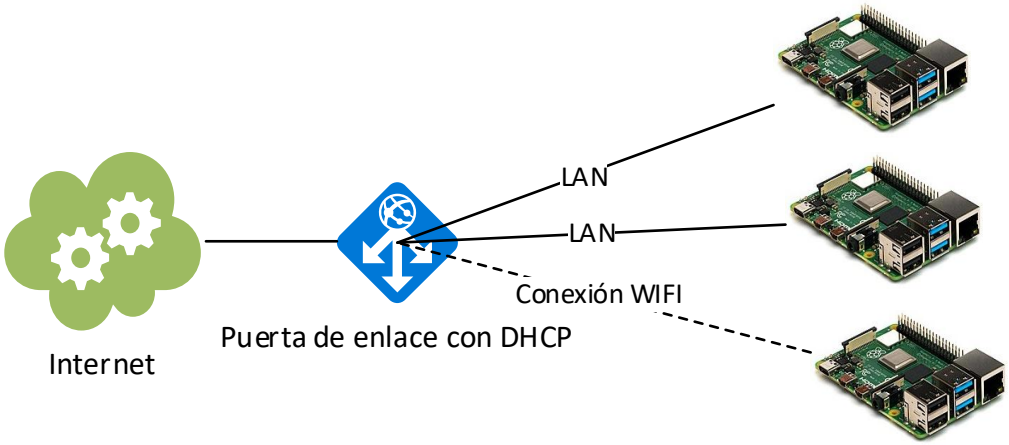


Imagen 23 - Configuración LAN

Esta arquitectura es la más básica posible, pero sin embargo es la más utilizada en la industria o el hogar, los proyectos más deslocalizados de *IoT* necesitan de desarrollos de conectividad aparte para adaptarlos a redes LoRa o Zigbee.

No obstante, debe ser uno de los siguientes pasos del prototipo, pero no es competencia de este TFG.

Con respecto a la conectividad WAN, se utiliza para el prototipo una conexión estándar de pruebas, el objetivo del prototipo es validar el funcionamiento general del sistema, por tanto, una conexión a internet de fibra óptica de 100 Mbps simétrica, bastante sobredimensionada será la elección para este TFG.

El estudio del despliegue de los dispositivos debe adaptarse y estudiarse en cada caso particular, no es lo igual sensorizar una fábrica con accesos WIFI que un área rural de 50 kilómetros de extensión.

Al igual que en el apartado LAN, como siguientes pasos al prototipo se deberían estudiar diferentes modelos de *routers* 4G y 5G con configuraciones estándar para así tener el sistema preparado.

4.3.1. Seguridad e infraestructura

La seguridad es un punto importante para tener en cuenta en una plataforma de IoT, el configurar y contar con estas premisas desde la base del diseño de la plataforma proporciona una sólida base de seguridad.

Protocolo y comunicaciones:

Como se ha descrito anteriormente, el entorno utiliza MQTT para la comunicación entre los clientes, Broker, servidor etc. MQTT es un protocolo ligero, aunque en sí mismo ya implementa estándares de seguridad como SSL/TLS para el cifrado. El Broker es el encargado junto

a las things de cifrar la información mediante certificados y asegurar una comunicación en el transporte de manera segura.

Además, el protocolo implementa a nivel de aplicación una identificación única de los clientes y una validación de estos mediante usuario y contraseña para identificar los clientes de manera inequívoca a este nivel (mediante el Broker utilizado).

Se puede implementar la seguridad mediante conexiones cifradas de redes virtuales privadas (VPN) aunque esto incrementa enormemente la complejidad y los requisitos del hardware, por tanto, escapa a la filosofía de este TFG. No obstante, es interesante en infraestructuras críticas de sistemas de relevancia.

Validación a nivel de Backend:

El sistema propuesto acopla un segundo nivel de validación determinado por la información del propio core del aplicativo, son los datos almacenados en el sistema que una vez pasado el corte del Broker validan los códigos de ID y equipos conectados, dando más robustez a la seguridad del sistema (a costa de ciertos peligros de sobrecarga o ataques de denegación de servicio (DDOS)).

Seguridad a nivel de sistema operativo:

Generalmente la mayoría de los ataques a sistemas IoT provienen de agujeros de seguridad en los propios sistemas operativos de las things o el servidor. Mantener una buena política de actualizaciones mitiga en gran medida ataques de *botnets* o fallos conocidos de seguridad ya parcheados por los desarrolladores del sistema operativo.

Infraestructura de acceso a la plataforma (Firewall):

Como norma general, en el acceso al sistema que gestiona las conexiones de la plataforma se encuentra un elemento de control de acceso conocido como cortafuegos o firewall, este es el encargado de gestionar las conexiones entrantes de internet hacia los servicios de los servidores (Broker, BBDD, Backend, Frontend etc.)

Se debe tener en cuenta esta infraestructura par a conseguir una seguridad global del sistema, y aunque no es el objetivo de este TFG si lo es dar una serie de configuraciones base que contemplar en el despliegue.

Se debe mantener una filosofía de "*usar solo y solo lo necesario*" es muy importante contemplar y monitorizar todos los puertos exteriores de acceso a la infraestructura interna de los sistemas. También es muy importante no utilizar los puertos estándar ya que estos pueden proporcionar pistas a los atacantes sobre la infraestructura.

A continuación, se describen los puertos necesarios a contemplar para una configuración de firewalling básica):

- Puertos a nivel de Broker:
  - mqtt:ssl (conexiones cifradas): 8883
  - mqtt:tcp (conexiones tcp sin cifrar): 1883
  - mqtt:ws (web socket de acceso a Broker): 8083
  - mqtt:wss (web socket de acceso a Broker cifrado): 8084
- Puertos a nivel de base de datos:
  - Tcp 3306
- Si las things están también bajo algún firewall (con control de salida)
  - mqtt:ssl (conexiones cifradas): 8883
  - mqtt:tcp (conexiones tcp sin cifrar): 1883
  - mqtt:ws (web socket de acceso a Broker): 8083
  - mqtt:wss (web socket de acceso a Broker cifrado): 8084
- Servidor WEB
  - Puerto estándar HTTPS: 443
  - Puerto estándar HTTP: 80

Se insiste que esta es una configuración estándar y puede (debe si se requiere un nivel elevado de seguridad) cambiarse por otra totalmente personalizada.

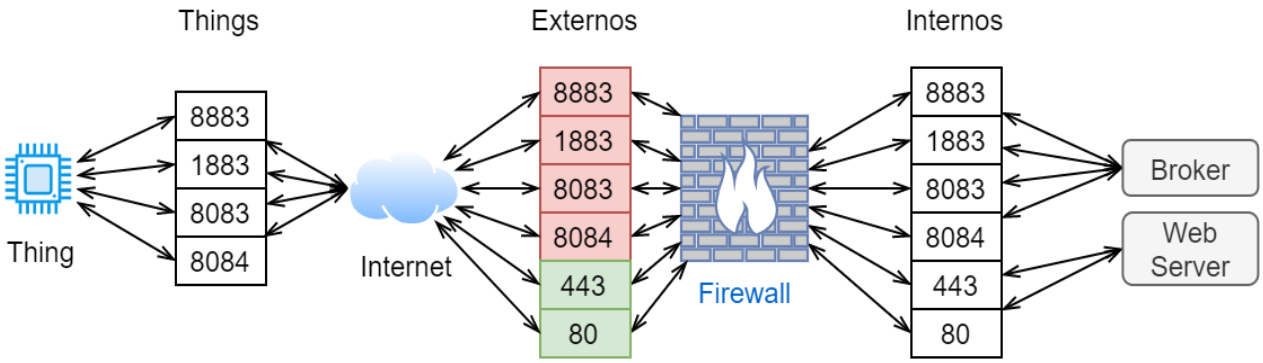


Imagen 24 - Estructura de comunicaciones a nivel de puertos.

En la imagen 24 se ve la estructura de comunicaciones a nivel de puertos, en rojo están marcados los puertos externos del *firewall* que convendría modificar en un despliegue de producción. A nivel de things, cada infraestructura es diferente, pero sería necesario tener abiertos, como mínimo estos 4 puertos para que el sistema sea operativo.

4.4. Plataforma *IoT*

Una vez está definida la estructura de comunicación y la arquitectura de hardware de las things, es momento de decidir qué sistema utilizaremos para centralizar todas las acciones de estas.

Existen multitud de soluciones y multitud de escenarios, el desarrollo se centra en protocolos estándar y soluciones *open Source* para dar al sistema robustez y continuidad sin incurrir en un coste extra.

La plataforma se compone de varios componentes que se ejecutan en el lado del servidor (o servidores) del sistema. Cada parte tiene una función específica y se interconecta con las demás para aportar funcionalidad.

Todos los elementos se apoyan en un sistema operativo que es el encargado de gestionar los recursos de hardware para que el sistema sea estable y funcional.

En el prototipo todo está sustentado por Debian 10 que proporciona la basa para los demás aplicativos.

Componentes del Servidor

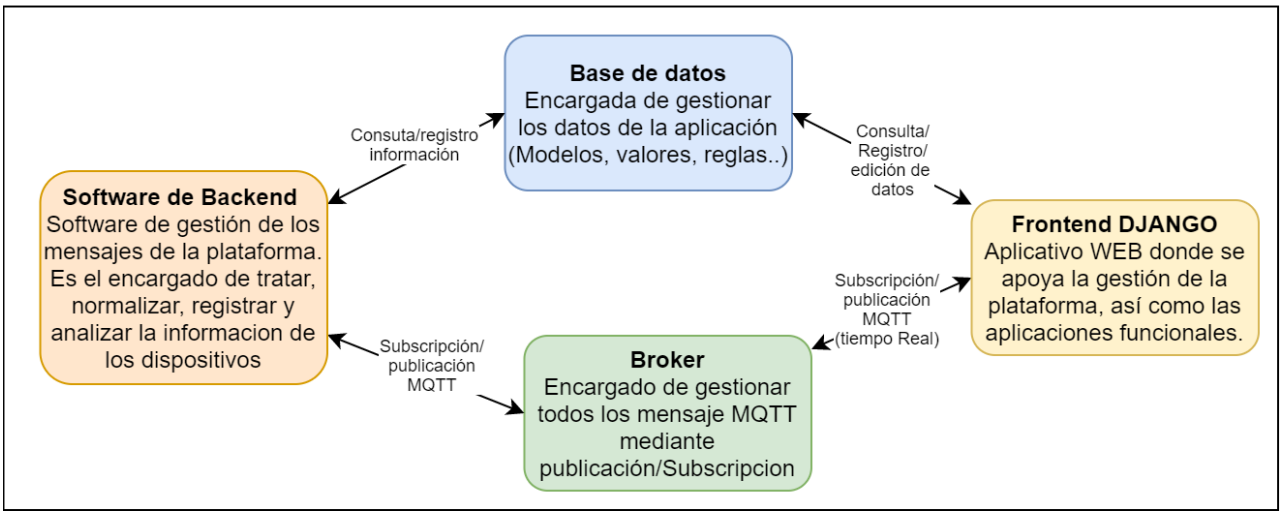


Imagen 25 - Lado del Servidor

Los componentes son los siguientes:

- Broker

El *Broker* se encuentra alojado en este servidor y se comunica con todas las partes del sistema a excepción de la base de datos. Desde este se accede a los mensajes publicados por las *things* y los clientes/actuadores, además, se actúa en consecuencia según los mensajes y los *topics* (almacenar en base de datos, mensajes de estado, etc.).

- Frontend DJANGO (Aplicación WEB)

La aplicación Web Django también se sitúa en este lado, y es la encargada de mostrar los datos de la base de datos a los clientes mediante el navegador, pero también de actualizar los datos en tiempo real mediante las subscripciones al *Broker*.

- Software de *Backend* (Python)

Encontramos también el Back-end funcionando en el lado del servidor, es quien gestiona todas las publicaciones definidas y de interés y las almacena en la base de datos. Aparte es el encargado de normalizar los datos y evitar registros erróneos o inseguros (validación de datos y conexiones).

El *Backend* se compone de un programa construido en Python que está permanentemente a la escucha de los mensajes del Broker, los interpreta al recibirlos y si es necesario los almacena en la base de datos.

- Software de Base de Datos

Este software (María DB) es el responsable de almacenar los datos. En el diseño se ha utilizado una estructura de modelo mediante una filosofía de *Code First,* es decir, desde el modelo de DJANGO se establecen los parámetros y la estructura de los datos complejos, después mediante migraciones se actualiza la base de datos.

4.4.1. Sistema operativo para servidores

Siguiendo la línea de selección Open Source de los sistemas, la base donde se ejecutarán tanto el *Backend* como el *Broker* EMQ debe serlo también.

Las distribuciones basadas en un Kernel Linux son una buena opción con estas características (Open Source), son sistemas gratuitos, de código abierto y extremadamente estables.

Los puntos clave para elegir un software Linux son los siguientes:

- Estabilidad y seguridad
- Versiones de servidores
- 100% Open Source
- Versiones LTS
- Comunidad extensa
- Soporte extenso de hardware

Todas estas características y muchas más son el sello de identidad de Debian, una de las distribuciones de Linux más antiguas y estables en el mundo del código abierto. También sirve como base para Ubuntu, la versión más popular de Linux de escritorio.

Debian es 100% Open Source, es decir, no incluye ningún paquete que no cumpla con sus estrictos estándares de calidad y licenciamiento. Tiene una característica muy útil cuando el sistema puede instalarse en la nube o en soluciones *On Premiss*, es compatible con hardware muy antiguo, lo cual es una ventaja cuando los despliegues en las instalaciones de los clientes son muy heterogéneos.

En concreto utilizaremos la versión de Debian “Buster” o Debian 10, la versión con mantenimiento extendido más reciente en el momento de redactar este TFG.

4.4.2. Broker MQTT

La selección del *Broker* es un paso importante ya que este será la pieza central del sistema. Este debe ser de código abierto para no incurrir en gasto de licencias, pero debe tener detrás soporte por si el sistema crece.

Si se estudian las comparativas de estos sistemas (acceso a la comparativa: *Comparison of MQTT implementations* (*Comparison of MQTT Implementations - En.Wikipedia.Org*, n.d.)) se observa que existen multitud de servicios, todos con sus pros y contras, en este TFG nos centraremos en soluciones Open Source.

Dada la gran variedad que pueden dar funcionalidad, y teniendo en cuenta la modularidad del sistema (independencia del Broker), existen estudios de rendimiento cómo “*A comparison of mqtt brokers for distributed iot edge computing*”(Koziolek et al., 2020) que pueden aproximarnos a una elección acertada dando un rendimiento superior al Broker EMQ X sobre otras soluciones.

EMQ X(*Broker EMQX - Www.Emqx.Io*, n.d.) aparte, es uno que más funcionalidades nos aporta dentro de la licencia Open Source (*Apache License Versión* 2.0) incluidos los mensajes MQTT cifrados por TLS, *QoS* de nivel 2 y funcionamiento a nivel de clúster.

La documentación de EMQ es muy extensa y completa además de soportar de forma nativa MQTT, está diseñado para funcionar como un clúster (de manera nativa) lo que ofrece escalabilidad rápida además estable.

EMQ X pude ser desplegado en la mayoría de los sistemas operativos que el mercado ofrece actualmente, lo cual es una ventaja a la hora de migrar la solución o seleccionar un sistema que no suponga un coste extra en licencias.

4.4.3. *Backend - Python* como base de código

En base a las anteriores consideraciones y en el amplio abanico de posibilidades que los sistemas ofrecen, seleccionar un lenguaje de programación para acometer todas las fases del desarrollo sería una medida acertada dada la extensión del proyecto y la heterogeneidad de los sistemas.

Python es un lenguaje interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y un código legible además de claro. Es de código abierto, por tanto, puede utilizarse para cualquier fin sin tener en cuenta licencias ni costes adicionales.

Otra de sus principales ventajas es que es multiplataforma, por tanto, ofrece mucha versatilidad a la hora de seleccionar opciones para el sistema, ya que pueden migrarse los desarrollos a diferentes entornos de producción sin remodelar el código.

La ventaja de utilizar Python es que se puede utilizar las mismas librerías y código similar en la parte del *Back-end* y en la parte de las *Things* lo cual aporta mucha agilidad de desarrollo reduciendo costes en los sistemas (uno de los objetivos del proyecto).

En cuanto al software en sí, es el encargado de interpretar toda la información proveniente del Broker, la potencia de las librerías de Python permite crear clientes MQTT de manera dinámica, dando gran capacidad de filtrado y eficiencia.

Desde las subscripciones en los clientes del *Backend* se recibe la información procedente de las things, se procesa, se valida y si procede se registra en la memoria del sistema para ser explotada posteriormente, además, el software es capaz de enviar información tanto a las things como a los clientes (mediante publicaciones MQTT) por ejemplo, cuando una thing necesita algún dato almacenado en el sistema.

4.4.4. DJANGO en el Frontend

El frontend y la gestión de dispositivos debe tener una respuesta rápida y dado que Python es la base del *Backend* y de las *Things,* se utiliza Django (Python) para el desarrollo web.

Django es un *freamework* de alto nivel que permite que los sitios web sean desarrollados y desplegados de manera óptima, eficiente y con garantías de seguridad además de escalabilidad. Es un *freamework* completo, es decir, dispone de serie de todas las herramientas para desplegar una web, y además proporciona un panel de gestión de administradores con el cual se pueden gestionar usuarios y grupos de acción.

Utiliza un patrón de programación basado en el modelo vista controlador (modelo vista templete en el caso de Django), lo que asegura un código ordenado y eficiente.

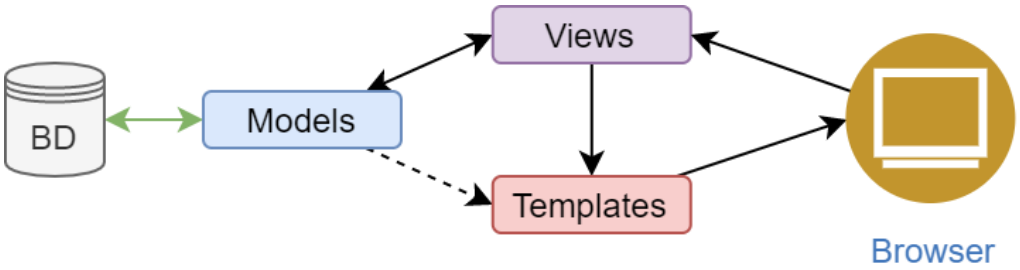


Imagen 26 - Modelo MVT Django

Las peticiones del navegador se pasan a la vista en Django, esta recupera los datos del modelo definido en la base de datos y se renderizan los datos en los *Templates* (HTML + CSS + Javascript + Python).

4.4.5. Almacenamiento de datos

La persistencia de datos es una característica necesaria en el sistema propuesto, el almacenarlos y explotarlos es necesario para conseguir un funcionamiento correcto del sistema, por tanto, es necesario la implementación de un sistema de base de datos que soporte esta tarea.

Siguiendo un criterio Open Source y dada la extensión en la comunidad que esta tiene, utilizaremos MariaDB como solución de almacenamiento del sistema.

MariaDB 100% Open Source y libre, por tanto, no requiere ningún tipo de licenciamiento, ofrece todas las funciones necesarias para operar y tiene una comunidad muy extendida.

Admite más de 200.000 conexiones simultaneas es muy simple de instalar, gestionar y migrar además de soportar numerosos motores como: Aria, SphinxSE, TokuDB, FederatedX, ScaleDB, Spider, etc.

4.5. Definición de la arquitectura

Finalmente, si se unen todas las piezas, se obtiene un sistema de *IoT*, basado en software libre, económico, potente, escalable y además de versátil, ya que en los diferentes escenarios lo requieren.

El sistema está pensado para ser modular y basado en estándares, es totalmente susceptible de adoptar diferentes placas en el lado de las Things, de utilizar cualquier tipo de almacenamiento o incluso de migrar el tipo de *Broker*.

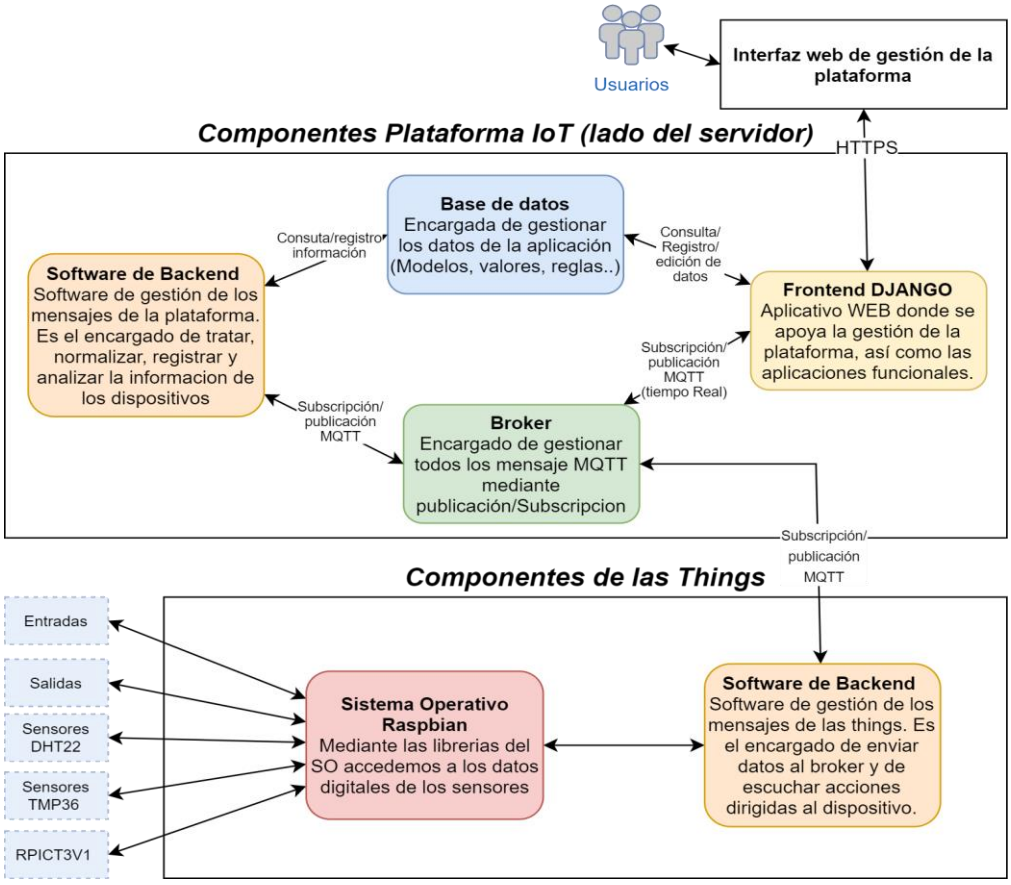


Imagen 27 - Arquitectura completa IoT

Este puede ser desplegado tanto en las propias instalaciones de los clientes como en servicios *cloud* como AWS o Azure ya que todos los componentes son soportados en ambas.

4.6. Topics y *Backend*

Los temas de suscripción/publicación son la vía de comunicación del sistema con las things, ambos tienen un software conectado al Broker que se suscribe a información y publica contenido según las situaciones.

El software en ambos casos (things/sistemaIoT) sigue un esquema de topics específico en el que se detalla primero el ID del sistema que emite el mensaje, seguido de una acción que determina como se debe actuar y finalmente un dato relacionado con el pin de actuación del equipo donde está conectado el sensor:

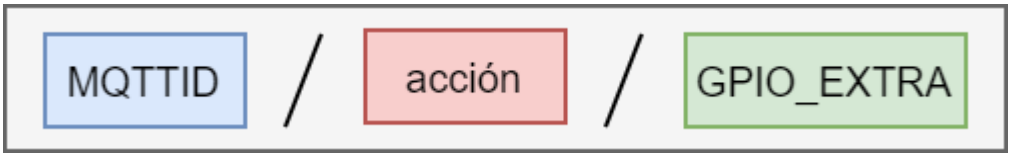


Imagen 28 - Estructura de TOPICS

Mediante esta estructura se puede identificar quien envía los mensajes (o a quien son enviados), que acción se pretende realizar, y en que elemento.

Además del topic, se envía un payload de trabajo donde va codificado el mensaje en cuestión. Este también estará estructurado para las acciones propias en las things o las publicaciones de datos relevantes.

Los equipos remotos o things, gracias a la estructura descrita, pueden subscribirse a todos los mensajes que se emitan en su Código MQTTID, mediante un comodín multinivel (#). De esta manera se puede establecer una comunicación directa entre el usuario que desea realizar una acción y el elemento remoto, que estará escuchando estas mediante una suscripción al Broker.

Por ejemplo, para leer un sensor DHT22 en el pin 16 del equipo "123456789" se haría una publicación como la siguiente:

- Topic: `123456789/action/16`
- Payload**:** `READDHT22`

El equipo 123456789 tiene que estar subscrito a todos los topics que van dirigidos a él (suscripción mediante comodín: "123456789/#") y al recibir el mensaje de action. El software del equipo con este topic y este payload realizara la siguiente interpretación:

- Idmqtt: 123456789 (es correcta)
- Accion a realizar: action (actuar sobre algun pin)
- Pin sobre el que actuar: 16
- Acción concreta que realizar: READDHT22 (leer temperatura y humedad)
- El equipo publica un mensaje con la información requerida

Una vez mapeados todos los datos mediante la suscripción, el equipo realizara una publicación con la información solicitada: leer la temperatura y la humedad mediante el sensor DHT22 conectado al pin 16 del equipo 123456789.

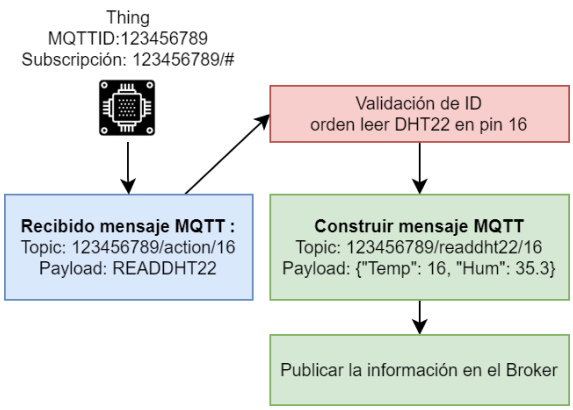


Imagen 29 - Ejemplo de mensaje MQTT

Las things, aparte de escuchar y actuar en consecuencia, también deben ser capaces de enviar información de forma recurrente y automática (temperaturas, tensiones etc..) por esto, el software de gestión tiene varias instancias automáticas que envían información al Broker según se determine en su fichero de configuración:

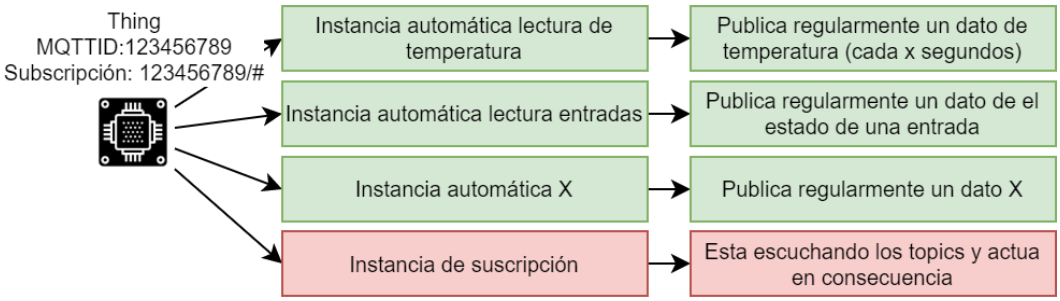


Imagen 30 - Instancias Things

Para el prototipo se han definido 5 acciones con las que interactuar en el sistema, conforme se añadan sensores y actuadores este número debe ir aumentando para tratar todos los casos.

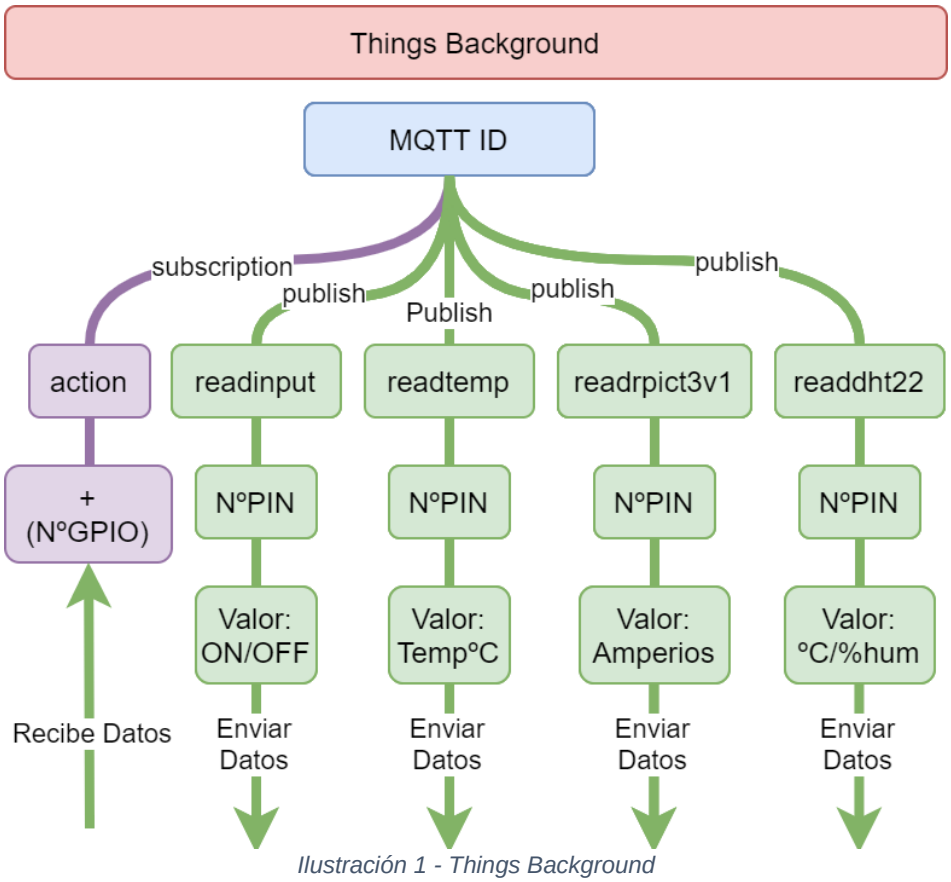


Ilustración 1 - Things Background

En cuanto a los *topics* de acción, soportan varios PAYLOADS válidos para interactuar que se describen en la siguiente imagen:

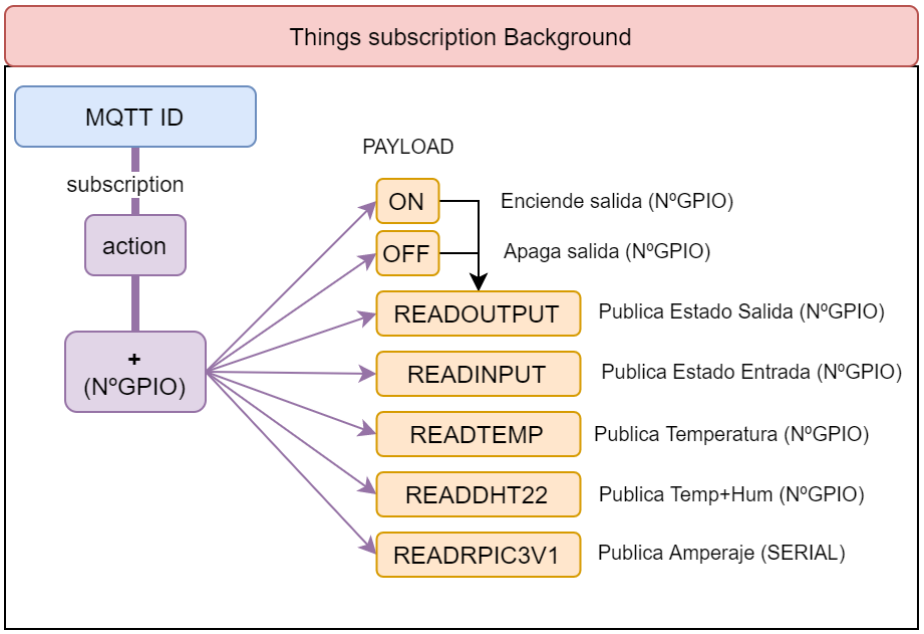


Imagen 31 - Acciones definidas

Por otra parte, se tiene el software de gestión del servidor, el cual también se nutre de la estructura de los topics, pero en este caso solo está a la escucha de mensajes para normalizarlos y grabarlos en la base de datos si es preciso.

La instancia de escucha del servidor está subscrita a todos los topics que tengan una acción valida definida, por ejemplo:

- “+/readtemp/#”
- “+/readinput/#”
- “+/readdht22/#”
- “+/action/#”
- “+/statusin/#”
- ...

Gracias a los comodines el sistema está a la escucha de todos los mensajes de interés para procesarlos y almacenarlos si procede en una base de datos.

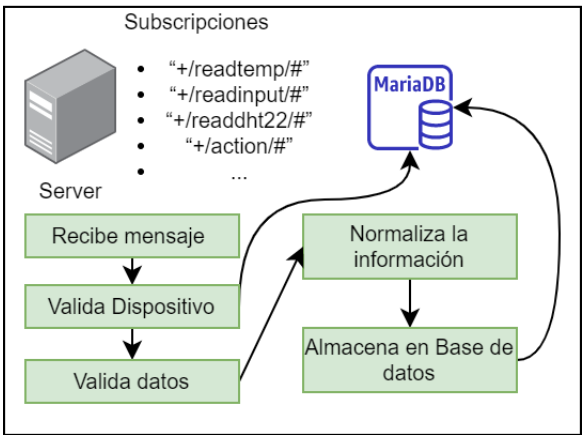


Imagen 32 - Software del servidor

Dado que el lenguaje de programación utilizado en el desarrollo es Python tanto en el lado de las Things como en el lado del servidor, se utilizará la librería Paho para la comunicación MQTT con el *Broker*.

La biblioteca de Python de Paho es una librería de MQTT disponible para Python, Android, C++ y etc. Es compatible con Python 2.7 y 3.x. Se enmarca en las licencias *Open Source de Eclipse Foundation* (*Paho Eclipse Foundation - Www.Eclipse.Org*, n.d.).

La biblioteca basa el código en el funcionamiento de MQTT y funciona mediante la asignación de eventos a funciones. Utiliza un objeto cliente para gestionar las conexiones, publicaciones y subscripciones.

Mediante *callbacks* el cliente gestiona mensajes entrantes para tratarlos, enviarlos y todos los servicios disponibles, todos los detalles de la librería de Python se encuentran en el enlace de pypi.org (*PAHO Mqtt 1.5.1 - Descripcion Funcionamiento @ Pypi.Org*, n.d.).

La polivalencia de la combinación del Broker EMQX, la librería paho y la codificación del servidor de *Backend* da al sistema la capacidad de desarrollar un motor de reglas propietario según el tipo de mensajes que queramos tratar, pudiendo disparar eventos según la información proporcionada por las things y los parámetros definidos por el usuario. Por ejemplo, si una temperatura es mayor a una regla de alerta por temperatura, el sistema puede enviar un Email a un usuario predefinido.

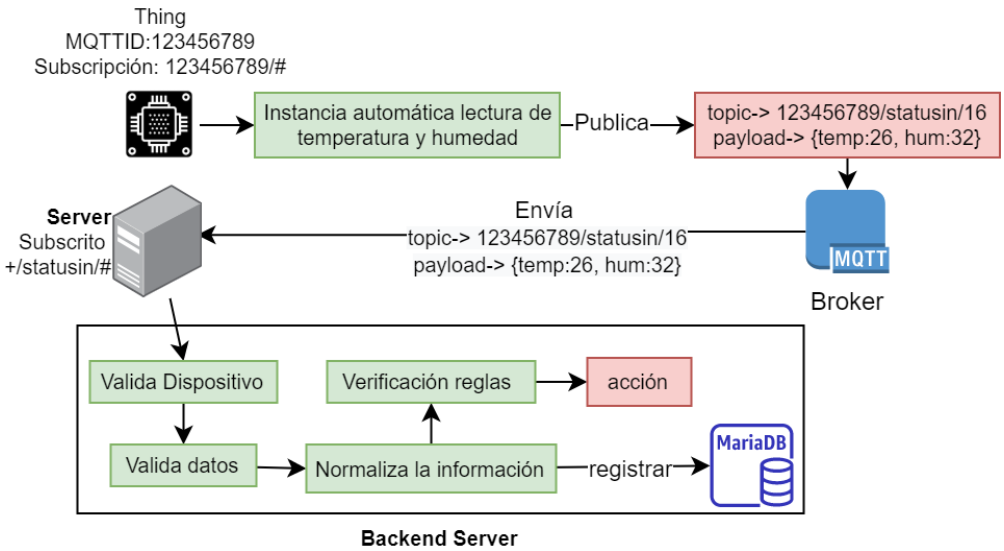


Imagen 33 - Flujo datos MQTT de things a servidor

En la figura 33 se detalla uno flujo de un mensaje MQTT de lectura automática por el sistema, se aprecia como mediante la definición del topic, se puede determinar el origen del mensaje, validarlo y gestionarlo para que este sea posteriormente presentado a un cliente o se produzca una acción configurada en alguna de las reglas del sistema.

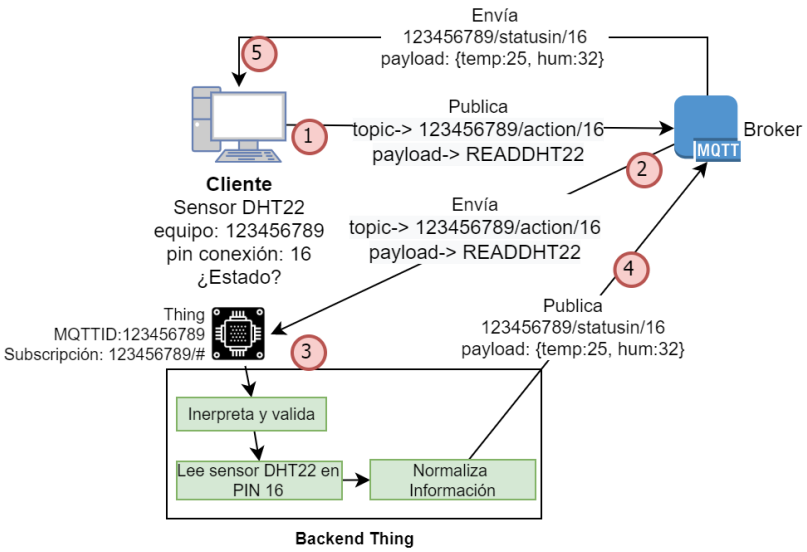


Imagen 34- Flujo mensaje MQTT backed things.

Sin embargo, en el lado del *Backend* de las things, se puede obtener información directamente de los dispositivos mediante la web del cliente, en la imagen 34 puede verse como el propio cliente (1) en tiempo real solicita información al dispositivo, enviando este una actualización inmediata (4) del estado actual (mediante el bróker), esta información se liga mediante un cliente MQTT en el navegador y se presenta al usuario en tiempo real (5).

4.7. Sistema Front-End

En fase de diseño se ha decidido utilizar un *framework* de programación web que facilite el diseño y despliegue, Django. Como con todas las aplicaciones (web, de escritorio, móviles, etc.) el diseño lógico es parte fundamental del ciclo de vida del aplicativo, este apartado pretende, no obstante, describir el diseño del modelo y la estructura funcional de la aplicación, así como la estructura de escalabilidad de aplicaciones.

4.7.1. Estructura aplicación

Las aplicaciones de Django se basan en el acceso a los datos mediante el modelo, este se encuentra programado en forma de clases de objetos Python que se relacionan entre sí. Las clases aportan versatilidad al diseño web y proporcionan toda la potencia del mapeo de objeto relacional (ORM) de Django para acceder, modificar, editar y borrar datos de una base de datos mediante las clases (modelo CRUD).

Las aplicaciones Django por encima de los modelos se estructuran en “Apps” que organizan el código (mediante anidamiento) y las convierten en diseños escalables. (Django Software Foundation, 2020)

En el caso de este TFG se ha partido de una aplicación base de Core llamada “iot_django” donde se sitúan las clases de modelo más genéricas y abstractas. A partir de este proyecto se van añadiendo nuevas Apps según el diseño de la aplicación va creciendo, es decir, todas las aplicaciones o paneles añadido se basan en los modelos definidos en iot_django. Para el prototipo actual serán 2 nuevas “Apps”, “cpds_monitor” y “energy_monitor”.

Se describe la estructura base con los modelos y las dos aplicaciones para el prototipo a continuación:

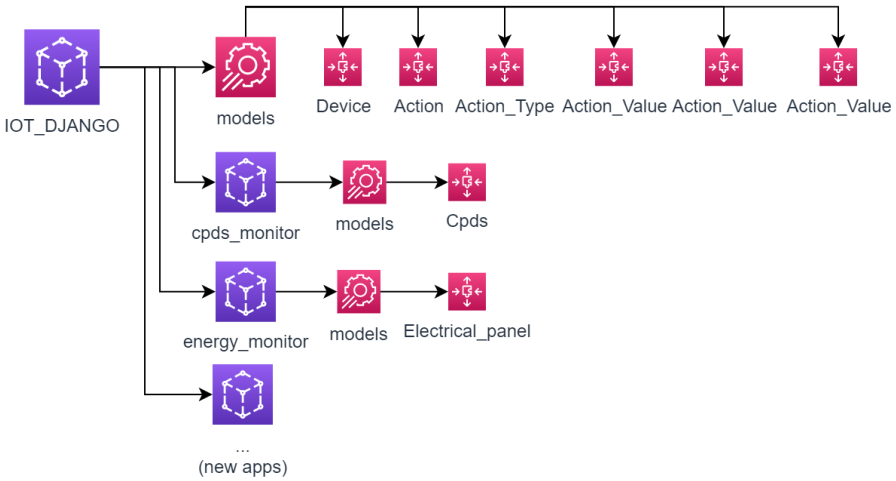


Imagen 35 - Django modelos

Es necesario incluir también los modelos proporcionados por Django para la gestión de usuarios y grupos de usuarios. Esto facilita enormemente la gestión y la seguridad del sitio web.

La estructura de Django es ideal para la gestión de un numero incierto de aplicativos desde un único sitio web.

En la siguiente imagen se describen además de los modelos, las relaciones entre estos:

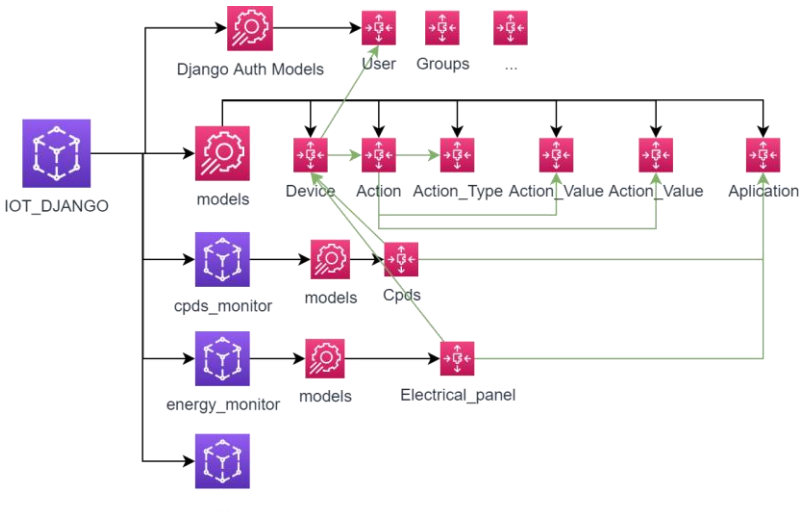


Imagen 36 - Django modelos y relaciones

4.7.2. Modelo de datos

Definiendo con la filosofía *Code First* los modelos en DJANGO, generamos mediante migraciones la estructura de datos que se presenta a continuación:

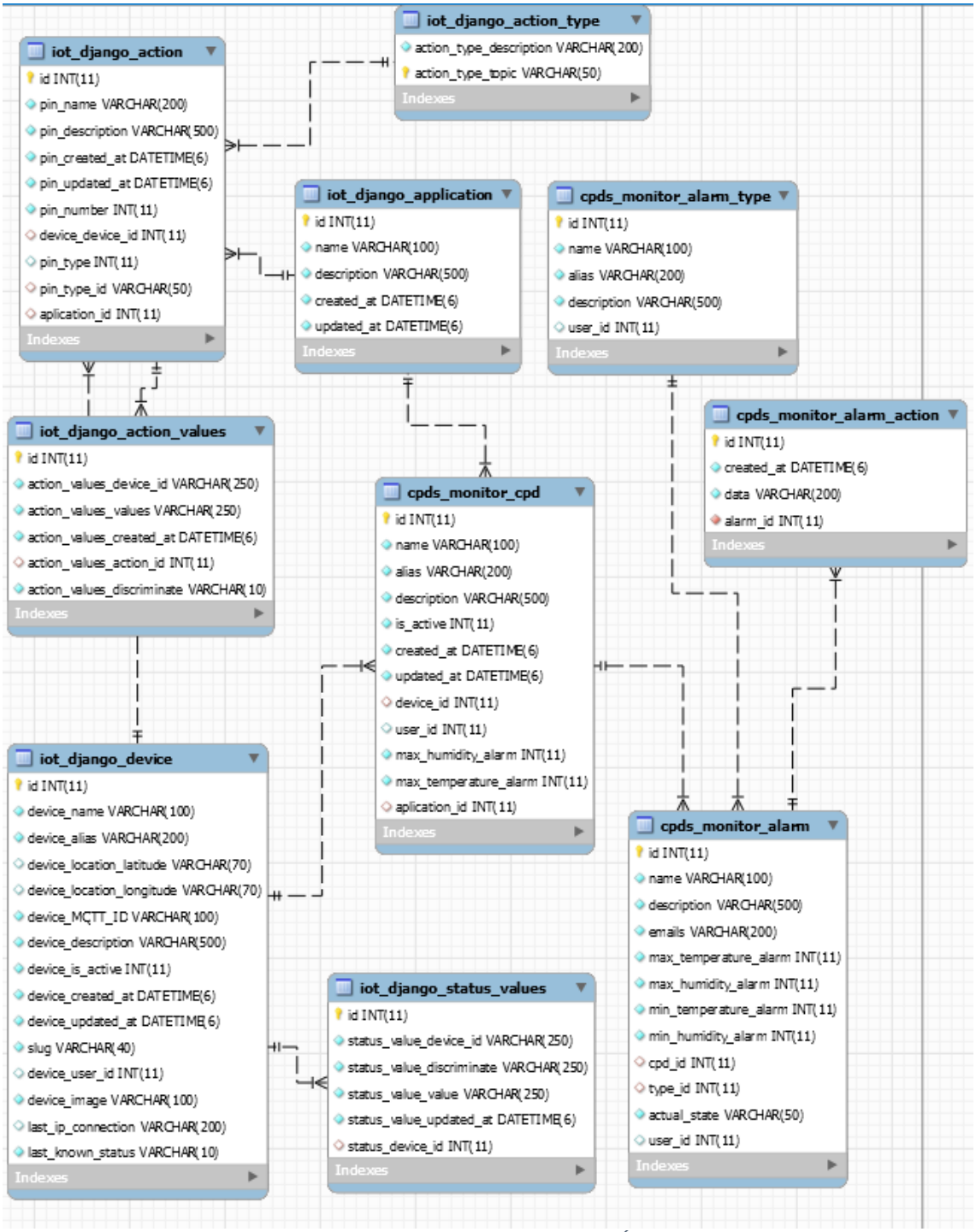


Imagen 37 - Modelo de datos APLICACIÓN

Este modelo es el producido en este TFG mediante código, aparte existe toda la infraestructura de modelo generada por DJANGO para la

autenticación y la necesaria para integrar el Broker EMQX con MySQL para la validación a nivel de aplicación.

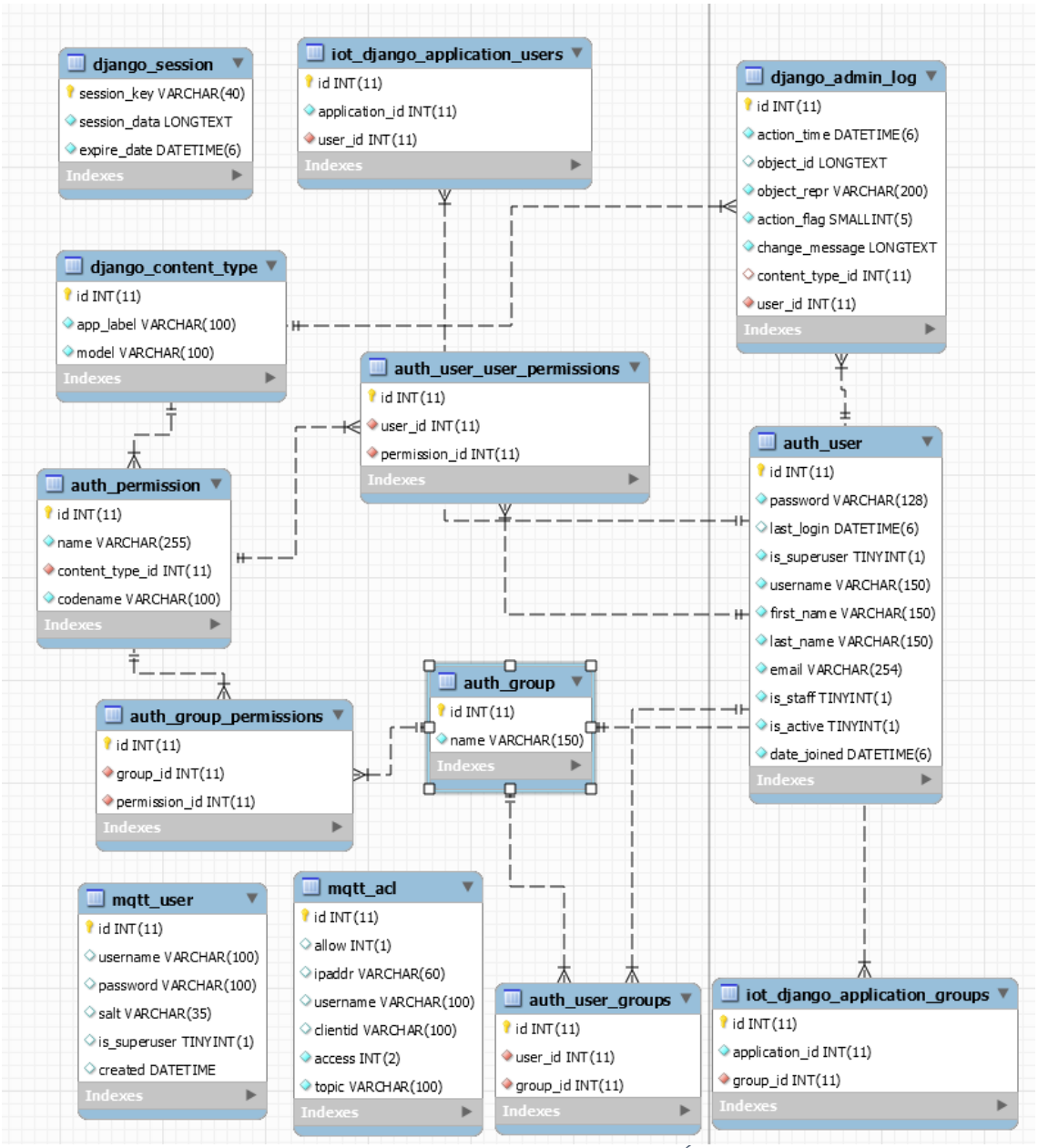


Imagen 38 - Modelo de datos AUTENTICACIÓN

4.7.3. Permisos y usuarios

En el diseño de permisos se ha utilizado el estándar aportado en Django, este sistema determina si un usuario es quien dice ser, y si lo es, que es lo que puede hacer este.

El sistema proporciona las siguientes características avanzadas (*Django Auth Documentation (Docs.Djangoproject.Com)*, n.d.):

- Usuarios
- Permisos: banderas binarias (sí / no) que designan si un usuario puede realizar una determinada tarea.
- Grupos: una forma genérica de aplicar etiquetas y permisos a más de un usuario.
- Un sistema de hash de contraseña configurable.
- Formularios y herramientas de visualización para iniciar sesión en usuarios o restringir contenido
- Un sistema *Backend* conectable.

Siguiendo la lógica de negocio explicada en fase de estudio, el sistema está orientado a aportar servicio a organizaciones con múltiples usuarios, aprovechando por tanto la potencia de los grupos para convertir estos en organizaciones, y poder filtrar los permisos de las things como de los paneles.

De esta manera, aunque un usuario sea el dueño de una de las things, todos los miembros de su organización puedan interactuar con esta.

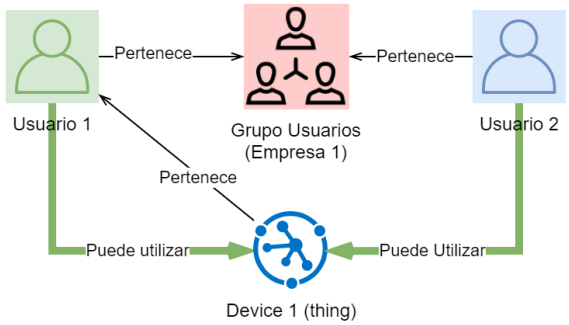


Imagen 39 - Permisos sobre las Things

Además, las aplicaciones están diseñadas para relacionarse con los grupos de usuarios (Empresas en nuestro caso), utilizando estas para dar rápidamente permisos sobre una aplicación a los grupos de usuarios (organizaciones).

Se observa por ejemplo en la siguiente imagen, como los empleados de la empresa 1 no pueden acceder mediante los permisos de las aplicaciones a “Monitor cuadros eléctricos”, mientras que los de la empresa 2 si tiene acceso a ambas aplicaciones.

Imagen 40 - Permisos de aplicaciones Django

4.7.4. Esquema WEB

En un principio el prototipo parte de un diseño simple con las funcionalidades básicas de gestión de dispositivos. Estos pueden crearse, editarse, borrarse y consultarse mediante la aplicación de Django. Además de los dispositivos, las acciones también pueden ser gestionadas mediante la plataforma, dando total libertad a los usuarios de interactuar con los dispositivos que sean de su propiedad.

Se divide por tanto el sitio web en 4 grupos para la gestión:

- Home:

Es la página de inicio y dará al usuario información de los dispositivos de su organización, así como acceso rápido a ciertas gestiones. En un futuro pueden presentarse ofertas y nuevos servicios relacionados. En caso de no tener una sesión válida el usuario será redirigido a una página de inicio de sesión.

- Equipos:

En este apartado se presentan en forma de lista los diferentes dispositivos asociados a la cuenta, permite la gestión y da acceso a las acciones (lectura de temperaturas, entradas, salidas…)

- Acciones:

Se accede desde la vista de equipos y las acciones van asociadas a un dispositivo. Desde esta vista se pueden ver los detalles de cada una de estas y añadir, borrar editar etc.

- Aplicativos:

Desde esta vista se presenta a los usuarios los servicios que tengan dados de alta, en el caso de este TFG, la monitorización d ellos cuadros eléctricos y los CPD`s.
La vista está concebida para ofrecer nuevos servicios y poder gestionarlos desde este mismo panel. Estos aplicativos son generales y pueden ir ampliándose conforme los clientes los soliciten. Al añadir nuevos servicios pedidos por un cliente, estos pueden ser ofertados a los demás usuarios.

En la siguiente imagen se da una descripción del sitio web del prototipo:

Imagen 41 - Mapa del sitio WEB

Las flechas verdes son accesibles para usuarios sin identificar, mientras que las rojas solo pueden ser recorridas por usuarios validados e identificados.

Una de las peculiaridades de DJANGO es que ya cuenta con un potente sitio de administracion por defecto, este panel administrativo será accesible para los miembros del staff que pueden dar permisos a los usuarios y organizaciones para acceder a los diferentes aplicativos.

4.7.5. Prototipado

En cuanto al prototipo esquemático del sitio, su usa un diseño simplista y limpio, acorde a las tendencias actuales de los diseños web. El sitio está concebido con un menú superior de navegación donde se acceden a todas las funciones, y una zona inferior para desplegar las vistas.

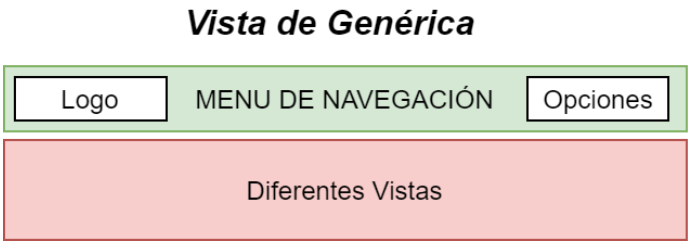


Imagen 42 - Vista genérica WEB (prototipado)

Partiendo de esta base, las vistas de equipos y de acciones parten de la base anterior, con un submenú de herramientas una tabla de datos (dispositivos o acciones) y una serie de tarjetas con información útil (conectividad, Id, fecha de actualización etc..).

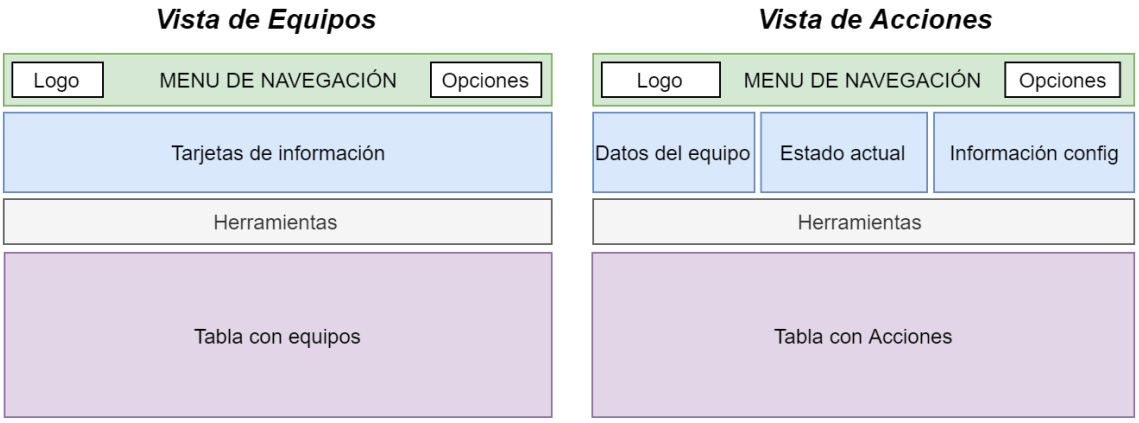


Imagen 43 - Prototipado Equipos y Acciones

Las diferentes vistas de detalle simplemente mostrarán los datos en forma de gráfica y de tabla de datos.

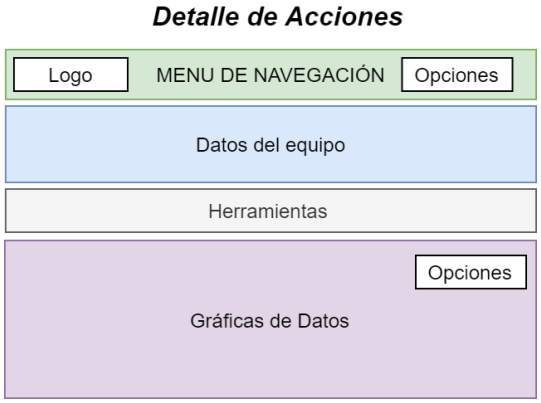


Imagen 44 - Prototipado de detalle

5. Producto, puesta en marcha y configuración

En este apartado se describe el sistema así como los pasos necesarios para dejar el entorno de Back-end (Server+Things) operativo además de la instalación de DJANGO, queda fuera de este TFG el despliegue real de toda la infraestructura. Los pasos son orientativos y para los sistemas que hemos seleccionado en las versiones concretas de operación.

Cabe destacar que el detalle de las configuraciones e implementaciones quedan fuera del ámbito de estudio de este TFG, por tanto, se dan instrucciones de guía y no un completo manual de configuración e instalación, para este se hace referencia a la documentación oficial de cada sistema.

5.1. Solución y producto

5.1.1. Software de *Backend* Things

El software de *Backend* de las things está presente en los equipos remotos en forma de programas agrupados en scripts. Estos son ejecutados y monitorizados mediante PM2, un gestor de procesos de producción con balanceo de carga y supervisión.

Este gestor de procesos *(GNU Affero General Public License 3.0 (AGPL 3.0))* permite mantener la actividad en los procesos de manera continua, nos da funciones de recarga automática de scripts en caso de errores y facilita las labores administrativas (logs, inicio, recarga, publicación…).

Aunque una de las mayores ventajas (sobre todo para un entorno IoT) es que proporciona modos de funcionamiento individuales (ejecución en un hilo de procesamiento o fork) o modo de funcionamiento en clúster, donde el software se encarga de balancear la carga y ejecutar múltiples hilos de un programa para aprovechar el multithreading del hardware anfitrión (más orientado a la parte del server que también funciona con PM2).

```
root@SENSORX:~# pm2 status
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
4	iot_dht22_background	default	N/A	fork	21411	5m	3779	online	0%	13.4mb	root	enabled
3	iot_input_background	default	N/A	fork	510	29h	16	online	0%	13.4mb	root	enabled
5	iot_rpict3v1_background	default	N/A	fork	480	29h	8	online	0%	13.5mb	root	enabled
2	iot_temperature_background	default	N/A	fork	22108	9s	483…	online	0%	12.9mb	root	enabled
1	system	default	N/A	fork	20789	5D	45	online	0%	13.5mb	root	disabled

Module

id	module	version	pid	status	↺	cpu	mem	user
0	pm2-logrotate	2.7.0	8482	online	0	0%	74.9mb	root

Imagen 45 - Procesos Backend Things PM2

Volviendo a los programas del Backend, cada uno tiene una tarea concreta y dispone de todo lo necesario para funcionar de manera independiente.

El equipo se configura desde un fichero de configuración alojado en memoria, desde este fichero se puede gestionar que tareas debe realizar el equipo, en que PINS debe hacerlo y la frecuencia de los procesos automáticos (tiempo de envío de temperaturas, por ejemplo).

config_iot_sys.ini

```
[mqtt]
mqtt_server = 192.168.1.21
mqtt_port = 1883
device_id = 123456789a
register_time_publish = 30
topics_suscribe = action

[sensor]
read_temperature = 1
gpio_sensors = 4
time_publish = 120

[sensorDHT22]
read_temperature = 1
read_humidity = 1
gpio_sensors = 22
time_publish = 2

[input]
read_input = 1
input_gpio = 13,16
```

Imagen 46 - Fichero de configuración Things Backend

5.1.2. Software de *Backend* Server

Al igual que las things, el software de *Backend* utiliza PM2 para ejecutar su instancia, aunque a diferencia del software de estas, solo dispone de un programa de escucha dividido en múltiples subscripciones que tratan los mensajes del bróker.

```
root@SRVDEBIAN:~# pm2 status
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	iot_basic_background	default	N/A	fork	71934	6D	6574	online	0%	23.7mb	root	enabled

Imagen 47 - Instancia Backend Servidor

El software utiliza dos paquetes secundarios creados para apoyar la gestión de los datos, `db_util.py` para la gestión con la base de datos y `message_util.py` para el tratamiento y normalización de los mensajes.

Para la gestión de objetos también se ha creado un fichero que contiene un modelo de clases de las things para facilitar la comprensión y programación del software (`model.py`).

5.1.3. Plataforma de gestión.

La plataforma de gestión de los dispositivos es la base del proyecto, desde sus vistas se puede interactuar con las things y asociarlas tanto a las aplicaciones como a las acciones con las que queremos actuar.

Esta alojada en el propio servidor y corre bajo el servidor de *testing* que proporciona DJANGO:

El primer contacto con la plataforma se realiza con el formulario de inicio de sesión, donde se un usuario deberá dar unas credenciales válidas para interactuar con la plataforma (Imagen 45).

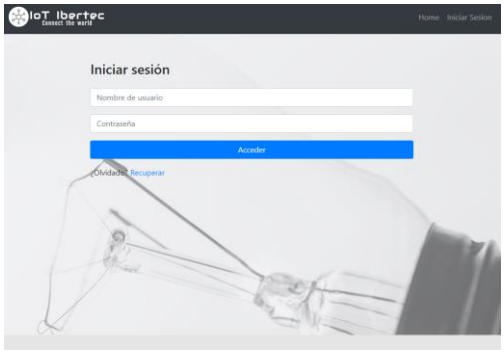


Imagen 48 - Inicio de sesión Plataforma

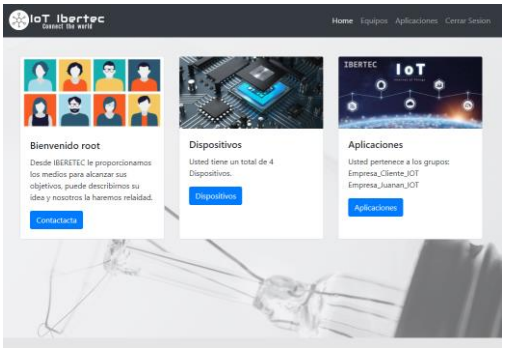


Imagen 49 - Vista de Home

Una vez estemos autorizados con un usuario y contraseña, el sistema redirige al usuario a la vista de home, donde se presenta el acceso a un formulario de contacto, a la gestión de los dispositivos y a la gestión de las aplicaciones.

La presentación de la gestión se hace mediante una página con todas las acciones a realizar (listado de things, detalles de estas, edición, borrado.)

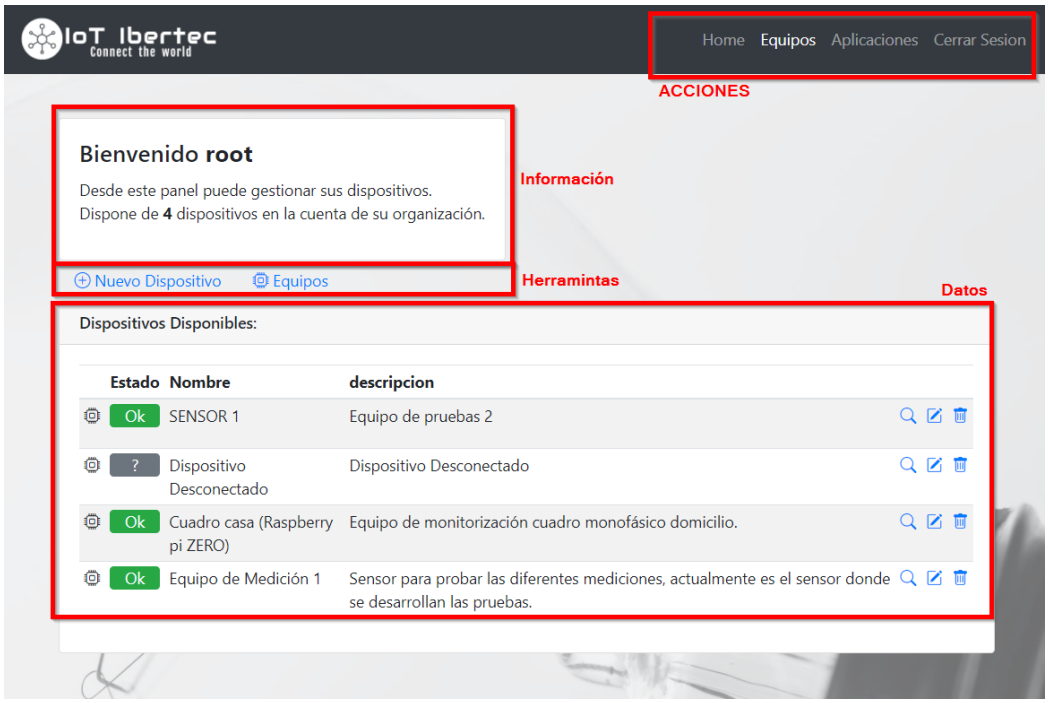


Imagen 50 - Gestión de Equipos interfaz WEB

Las things al crearse se asocian con el usuario del cliente (el que este validado creando el dispositivo), esto se hace de manera automática y deja ligado el dispositivo tanto al usuario como a los usuarios del mismo grupo operativo que el creador (la misma empresa).

Desde esta parte principal es posible ver el estado de los dispositivos y los datos más destacados, además, se puede acceder a todos los equipos asociados y a su gestión, como por ejemplo la edición de los datos de las things (descripción, alias, dirección MQTT etc..), se pueden borrar los equipos del cliente, se pueden crear nuevos dispositivos (barra de herramientas) o se puede acceder al detalle del dispositivo para interactuar con las acciones de los sensores de este.

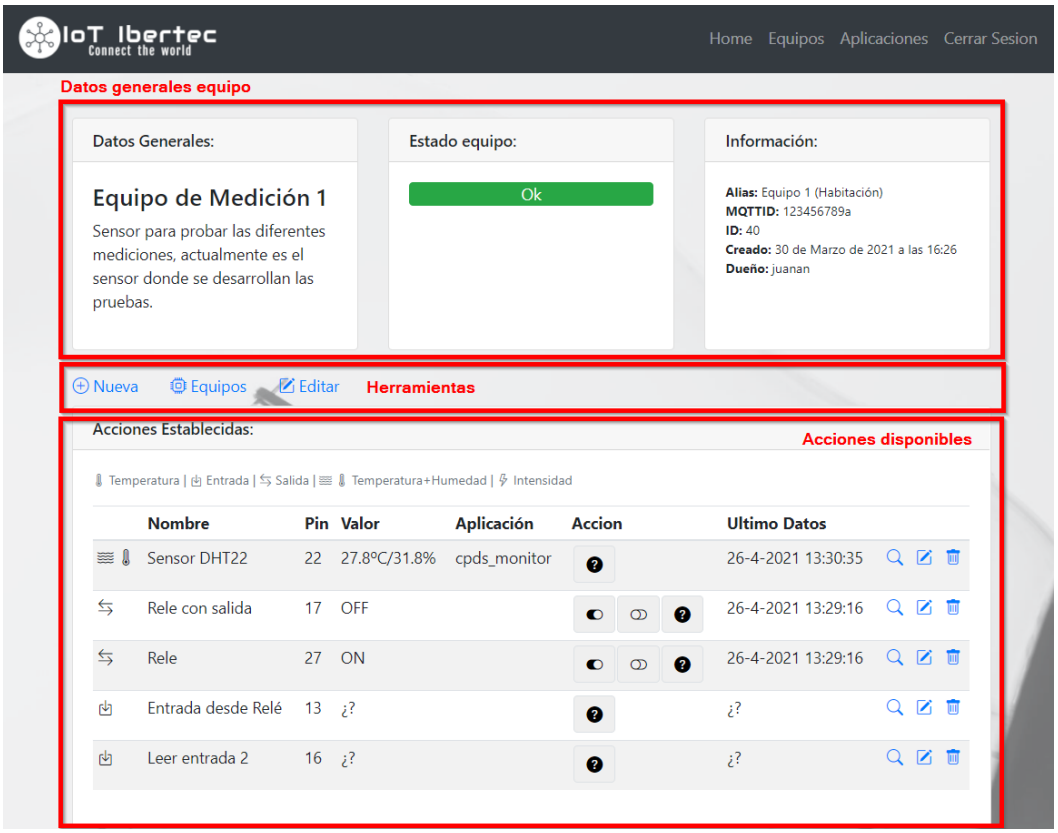


Imagen 51 - Detalle equipo - Gestión de acciones

Desde el detalle de cada equipo, el usuario puede crear nuevas acciones (una nueva salida, un nuevo sensor de temperatura…) y asociarlas a las aplicaciones a las que tenga acceso, además puede interactuar en tiempo real con estas (encender un contacto, leer un sensor, …). La gestión es completa y no está limitada, el usuario puede crear, editar y eliminar cualquier tipo de sensor que este dado de alta, además de acceder directamente a los datos registrados por este sensor en la base de datos:

Imagen 52 - Detalle de salida

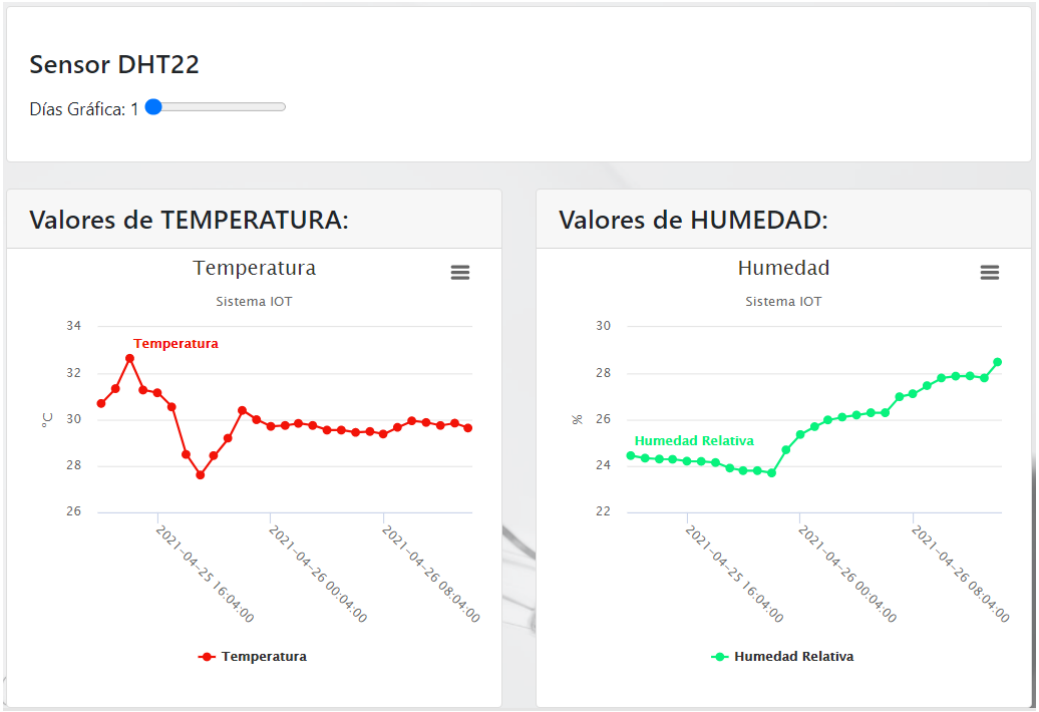


Imagen 53 - Detalle de sensor de temperatura DHT22

Por otra parte, la gestión del Framework nos proporciona una completa interfaz de administracion de los modelos que sean configurados. En el caso que nos ocupa, la gestión de grupos, usuarios y tablas maestras se realiza mediante esta interfaz.

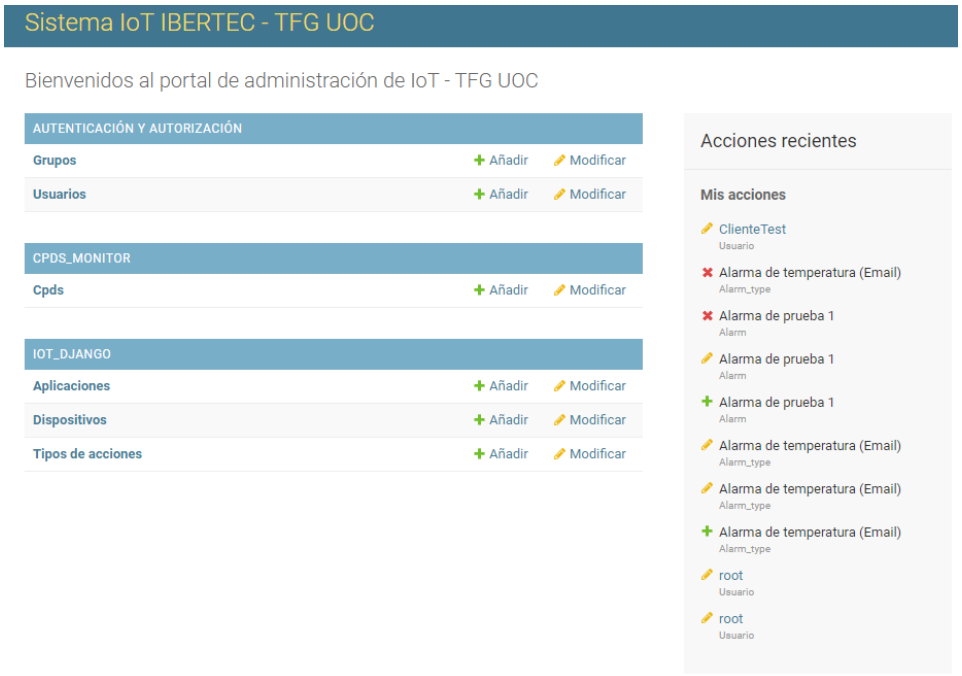


Imagen 54 - Interfaz de administracion de DJANGO

La parte más importante de la gestión administrativa de Django es la posibilidad de tratar los permisos de todos los usuarios de la plataforma, tanto por grupos, como individualmente. Estos permisos se conceden por elemento del modelo, es decir, que se pueden acotar todos los permisos de una manera muy meticulosa desde DJANGO, dando una doble seguridad a los aplicativos (impedir el acceso mediante la programación de la web y mediante el panel administrativo de DJANGO).

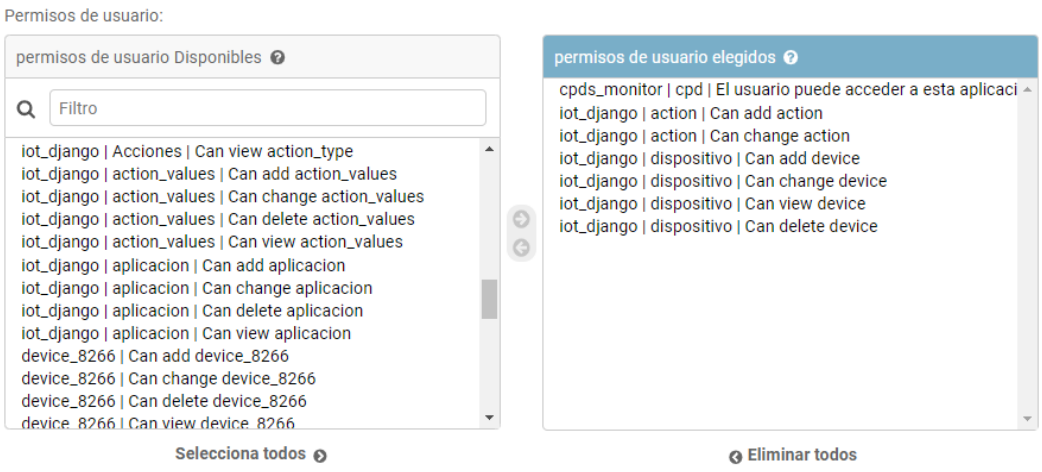


Imagen 55 - Permisos de usuario DJANGO

Para testear el modularidad de la plataforma y construir aplicaciones sobre las cuales aportar funcionalidades extra al sistema se describe en los dos siguientes apartados 2 aplicaciones creadas sobre la plataforma IoT, una para medir la temperatura y la humedad sobre los CPD´s de una

empresa, y otra para medir la intensidad en las fases de un cuadro eléctrico.

5.1.4. Medición de CPD`S y cuadros eléctricos

La aplicación muestra la temperatura y humedad de los centros de procesamiento de datos de una organización, utilizando las things de la plataforma IOT, los sensores DHT22 y la propia plataforma para visualizar los valores.

Esta se ha creado dentro de la aplicación principal de Django, guardando sus propios modelos, URL`s y templates HTML.

Para dar de alta un CPD simplemente hay que crearlo desde la consola de administracion de Django y asignarle un dispositivo con un sensor DHT22 (creado previamente en la plataforma).

Imagen 56 - Crear CPD en Django

En la plataforma en la zona de aplicaciones se aprecia el botón (si se tienen los permisos adecuados) que nos permite acceder a la visualización de los datos:

Una vez dentro se puede ver una vista de los centros de datos dados de alta, así como los datos de los sensores asociados a estas.

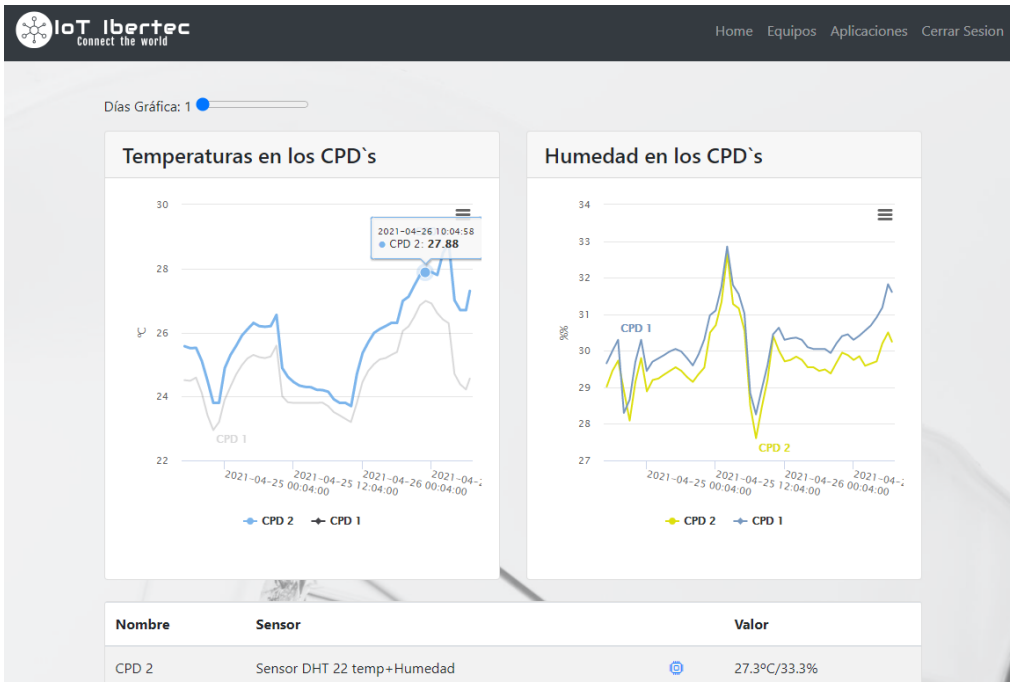


Imagen 57 - Datos de CPD`s

El mismo patrón se utiliza para la medición de cuadros eléctricos.

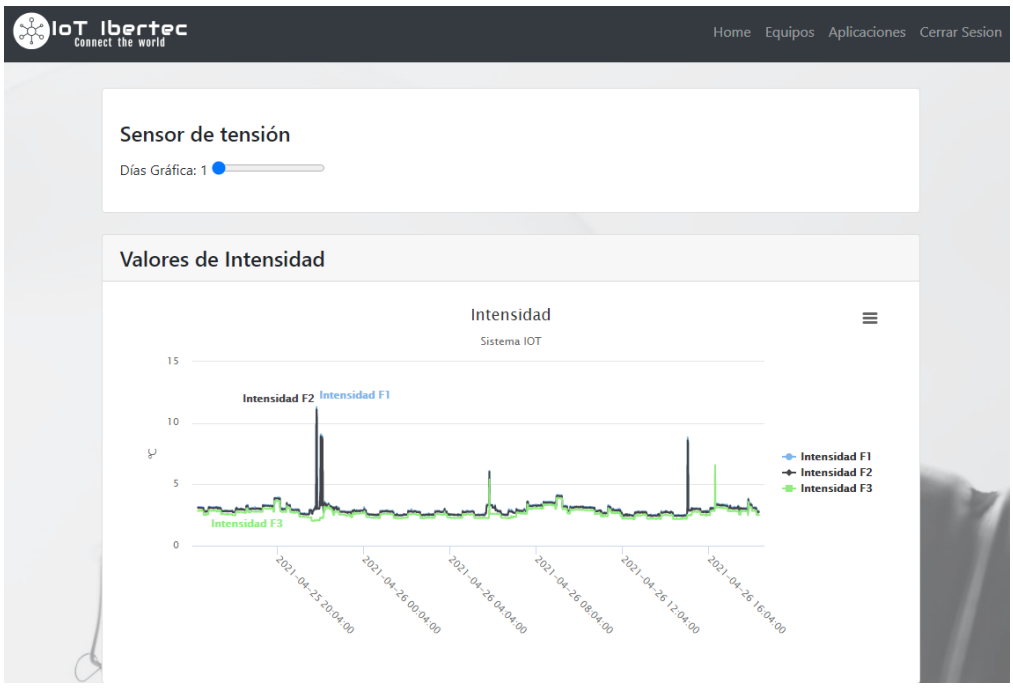


Imagen 58 - Datos monitor eléctrico

En donde se muestra el valor de las 3 fases monitorizadas (en realidad se está monitorizando la misma fase en diferentes magnetotérmicos de un hogar, aunque está diseñado para cuadros trifásicos) en vez de las temperaturas.

Cabe destacar que estos datos son mostrados en forma de gráficas, pero también se muestran en forma de tabla de datos y son descargables en varios formatos.

5.2. Puesta en marcha Servidor

El primer paso para poner en funcionamiento el sistema es instalar el sistema operativo sobre un hardware, en este caso un servidor local, aunque en un modelo de producción es conveniente desplegarlo en un VPS (Azure o AWS EC) para obtener las ventajas de la computación en la nube.

Para instalar Debian simplemente hay que proceder a la descarga el sistema (*Obtener Debias 10 BUSTER - Www.Debian.Org*, n.d.) y seguir los pasos de la documentación oficial y su detallado manual de instalación (*Debian 10 Install Manual (Www.Debian.Org)*, n.d.).

Una vez se tiene soporte y el entorno operativo (red de datos, acceso a internet etc..), instalaremos el software del Broker.

La documentación de EMQX es muy minuciosa y detallada, seguir las instrucciones del fabricante es muy simple:

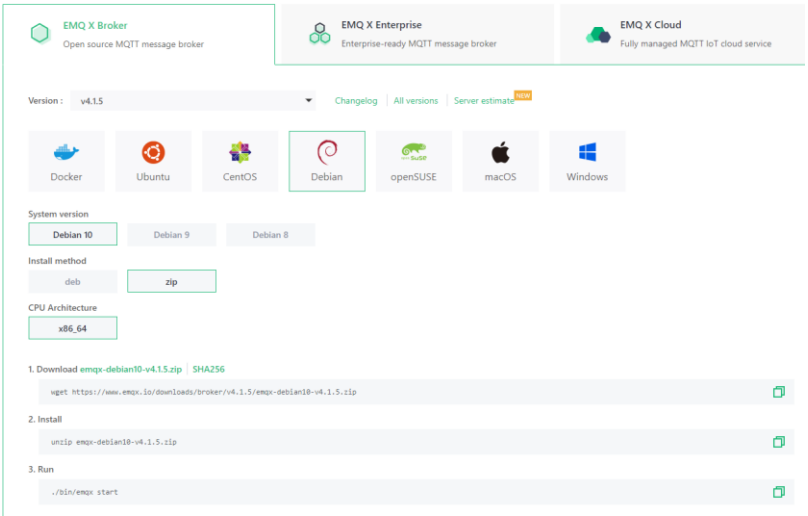


Imagen 59 - Instrucciones instalación EMQX (emqx.com/downloads) (*Download EMQX Manual - Www.Emqx.Io*, n.d.)

Una vez instalado y configurado se puede comprobar el funcionamiento mediante la interfaz web del propio Broker. El propio EMQ X tiene un cliente que utiliza web socket para testear el servidor (en el puerto 18083):

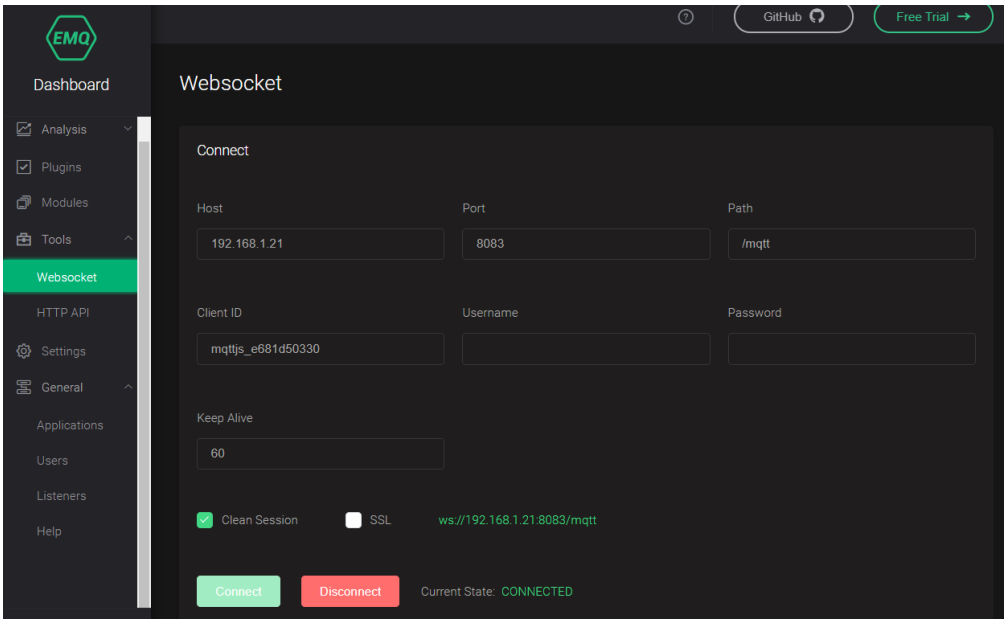


Imagen 60 - EMQ X web dashboard (puerto 18083)

Para activar la seguridad de Broker y proceder con los siguientes pasos debemos instalar y configurar la base de datos, en el caso de este TFG MARIA DB.

Al igual que en los anteriores casos, la documentación oficial es muy completa y detallada.

```
1. sudo apt update
2. sudo apt install mariadb-server
3. sudo mysql_secure_installation
```

Una configuración importante en EMQX es activar el plugin de autenticación de EMQX, para MYSQL (compatible con María DB). Una vez más, la documentación es fundamental (*Auth-Mysql in EMQX Broker - Docs.Emqx.Io*, n.d.).

Abrir si es necesario los puertos hacia el servidor EMQ, en el caso de la configuración de este TFG:

- MQTT:SSL → puerto 8883
- MQTT:TCP → puerto 1883
- MQTT:Web Socket → puerto 8083
- MQTT:Secure Web Socket → 8084

Si las things van a estar detrás de un router/firewall hay rdirigir los puertos hacia el servidor donde está EMQ X.

El siguiente paso es dar la base para ejecutar el software de *Backend*, PM2.

Desde la documentación oficial del sistema se dan las instrucciones necesarias (*PM2 Quick Start (Pm2.Keymetrics.Io)*, n.d.).

```
1. sudo apt install nodejs npm
2. npm install pm2 -g
```

El software está alojado en un repositorio de GitHub para mantener un control de versiones adecuado, y poder desplegarlo con relativa facilidad, además hay que instalar alguna librería complementaria.

```
1. git clone https://github.com/jiberob/IOT_SYS_BACKGROUND.git
2. pip3 install paho-mqtt
3. pip3 install mysqldb
```

Inicializar el servidor con PM2 y guardamos la configuración para que se inicialice al reiniciar el servidor.

```
1. pm2 start  iot_basic_background.py --interpreter python3 –watch
2. pm2 save
```

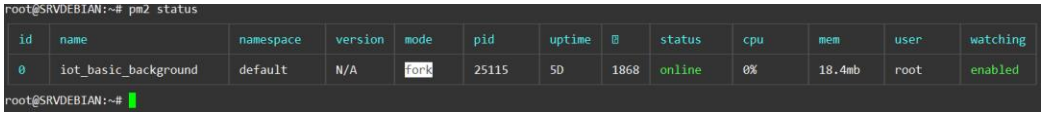


Imagen 61 - PM2 Status (salida)

Se puede testear el bróker con cualquier cliente de publicación/suscripción.

El proceso para la aplicación web se describe con claridad en la documentación de Django (*Install DJANGO Manuals and Guides (Docs.Djangoproject.Com)*, n.d.) (*Writing Your First Django App (Docs.Djangoproject.Com)*, n.d.), al iniciar las aplicaciones simplemente es necesario clonar el repositorio:

```
3. django-admin startproject iot_django
4. cd iot_django
5. python3 manage.py startapp cpds_monitor
6. python3 manage.py startapp electric_monitor
7. cd ..
8. git clone https://github.com/jiberob/django_apps.git
9. python3 manage.py runserver 0.0.0.0:8000
```

5.3. Puesta en marcha Things Raspberry Pi

Para la Raspberry Pi utilizaremos la imagen oficial del fabricante para el clonado y ejecución de Raspbian Lite un SO basado en Debian.

Seguir las instrucciones del fabricante para descargar e instalar Raspbian Lite (también pude utilizarse el propio Debian o cualquier derivado) (*Raspberry PI, Installing Operating System Images - Www.Raspberrypi.Org*, n.d.)

Al igual que en el *Backend*, es necesario instalar PM2 para ejecutar el software.

```
10.   sudo apt install nodejs npm
11.   npm install pm2 -g
```

Posteriormente solo hay que clonar el repositorio e el software en PM2 (las instrucciones están en el repositorio, ya que es necesario instalar librerías extra)

```
1. git clone https://github.com/jiberob/IOT_SYS.git
2. pm2 start system.py --interpreter python3 --watch
3. pm2 start iot_temperature_background.py --interpreter python3 --
   watch
4. pm2 start iot_input_background.py --interpreter python3 --watch
5. pm2 start iot_dht22_background.py --interpreter python3 --watch
6. pm2 start iot_rpict3v1_background.py --interpreter python3 --
   watch
7. pm2 startup
8. pm2 save
```

Seguir las instrucciones detalladas del readme.txt de la carpeta clonada. Dentro del software se detallan los pasos de instalación para los servicios de MQTT.

Modificar el fichero de configuración con los parámetros requeridos.

Uno de los objetivos de este proyecto es conseguir un sistema sencillo tanto de instalar como de configurar, como hemos visto el proceso no es muy complejo, pero aún necesita ciertas cualidades técnicas para su puesta en marcha. Para el testeo del prototipo se considera suficientemente sencillo para desplegar, no obstante, en una versión más comercial den trabajarse aspectos de despliegue en masa de una manera más trivial desde la propia plataforma.

En cuanto al conexionado electrónico, este depende del prototipado del proyecto en concreto, la plataforma permite dar de alta las GPIO`s que se necesiten (de manera dinámica) y asignarles una función (por ejemplo, al GPIO 6 asignarle una salida), no obstante, se describe el conexionado tipo de cada sensor para implementarlo en las soluciones:

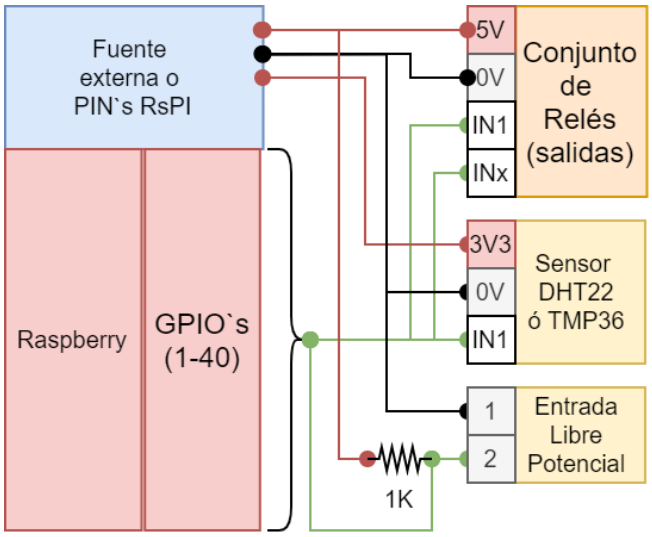


Imagen 62 - Conexionado estándar sensores.

La placa de medición de intensidades (lechacal) está fabricada para conectarse directamente al diseño del SBC de la raspberry pi, por tanto, no es necesario realizar ningún cableado/diseño.

6. Costes Estimados del proyecto

La estimación de costes del prototipo es importante, y aunque puede variar en gran medida para cada proyecto, la idea es aproximar un precio tipo de prototipo y estimar la viabilidad económica del proyecto.

Dado que todo el software utilizado es Open Source, el coste en licenciamiento es de cero euros.

Presupuesto Desarrollo sistema			
Recurso	***Descripción***	***Número de unidades***	***Precio***
Hardware	Hardware desarrollo	1	820,00 €
	Servidor ProLiant	1	800,00 €
	Raspberry pi (M3)	2	60,00 €
	Raspberry pi (Zero)	1	15,00 €
	Placa RPIV3	1	15,00 €
	Sensores (Varios)	1	30,00 €
Servidores Cloud	AMAZON EC2 (prueba)	1	- €
Personal	Diseño de arquitectura y programación	320 horas (66 eu/Hora)	21.780,00 €
TOTAL			23.520,00 €

Tabla 6 - Costes Estimado desarrollo

La plataforma una vez funcional puede ser explotada por un precio mínimo (solo costeando el mantenimiento básico y la plataforma de VPS). El coste de desarrollo y explotación de nuevas apps también debería ser bastante reducido ya que ser apoyaría en la infraestructura ya funcional.

Por otra parte, hay que contemplar dos escenarios en la implantación, uno donde el cliente solicite un despliegue en sus propias instalaciones (*On Premise*) y otro donde el cliente utilice la propia infraestructura cloud del sistema:

Presupuesto instalación On Premise sistema			
Recurso	***Descripción***	***Número de unidades***	***Precio***
Hardware	Servidor ProLiant	1	800,00 €
Personal	Puesta en marcha	30 horas (66 eu/Hora)	1.980,00 €
TOTAL			2.780,00 €
+			
Precio Unitario conjunto de THING+ SENSOR	Raspberry Pi + 4 salidas + 2 entradas + sensor DHT22	1	110,00 €
Sensores	Intensidad	1	15,00 €
	Salidas (4)	1	10,00 €
	Entradas	1	5,00 €
	DHT22	1	5,00 €

Tabla 7 - Presupuesto instalación On Premise

Con estas cantidades, una vez instalado el sistema, el coste por unidad determina el precio final del sistema, así como añadir nuevos sensores a estas.

El precio en un despliegue cloud es inferior en inicio, aunque requiere un licenciamiento mensual para costear la infraestructura (Software as a service SaaS). Este modelo está en auge actualmente ya que supone una comodidad para las organizaciones, y los costes son estables, mientras que el despliegue en infraestructura (a parte que el desembolso inicial es mayor) conlleva mantenimiento y personal especializado.

Presupuesto instalación Cloud			
Recurso	***Descripción***	***Número de unidades***	***Precio***
Hardware	Licencia (mensual)	1	60 €
Personal	Puesta en marcha	5 horas (66 eu/Hora)	300,00 €
TOTAL			360,00 €
+			
Precio Unitario conjunto de THING+ SENSOR	Raspberry Pi + 4 salidas + 2 entradas + sensor DHT22	1	110,00 €
Sensores	Intensidad	1	15,00 €
	Salidas (4)	1	10,00 €
	Entradas	1	5,00 €
	DHT22	1	5,00 €

Tabla 8 - Presupuesto instalación Cloud

7. Conclusiones

El desarrollo y puesta en marcha de un completo sistema de IoT ha sido un reto enriquecedor, la cantidad de tecnologías que conviven funcionando al unísono que hacen que todo parezca una sola pieza es lo que más me ha fascinado de este proyecto.

Es evidente que la extensión no permite abordar un alcance como me hubiese gustado, pero aun así considero que un sistema como este puede ser una buena forma de modelo de negocio.

El proyecto ha sido todo un reto, aunque he disfrutado este cuatrimestre aplicando todo lo aprendido estos años, mi experiencia en desarrollo web era muy limitada al comenzar y creo que he descubierto una nueva rama en la que tengo que invertir más esfuerzo, lo cual es algo muy alentador y que me llena de ganas de continuar más allá de la finalización de los estudios reglados.

En cuanto al trabajo, en líneas generales los objetivos se han cumplido de manera satisfactoria, la plataforma es funcional, versátil y permite alojas las aplicaciones propuestas y otras nuevas si son requeridas, esta es accesible desde cualquier parte si es alojada en internet y es relativamente sencilla de utilizar.

Los objetivos proponían la construcción de una plataforma de IoT donde apoyar dos aplicaciones concretas, un monitor de intensidad para cuadros eléctricos, y un monitor de entorno para CPD´s. Estos sistemas debían ser accesibles en un despliegue desde internet.

Los objetivos se han cumplido en un 90%, la plataforma es funcional y está construida para que los propios usuarios y sus organizaciones puedan gestionar tanto sus Things como sus aplicativos. Esta es totalmente accesible por internet ya que la arquitectura está diseñada para ser desplegada en cualquier servidor (VPS EC2, Azure…) o incluso en las propias dependencias de una empresa.

Uno de los objetivos que no se ha cumplido es la fácil configuración de las Things, la plataforma debía ser capaz de gestionar este trabajo desde la interfaz web, sin embargo, se pueden configurar los elementos, pero no es posible la configuración remota de los equipos, hay que acceder a un fichero de configuración interno y editarlo, lo cual puede ser confuso y poco amigable para cierto tipo de usuarios.

Este objetivo dada la infraestructura del sistema es perfectamente viable, bastaría con desarrollar los módulos en el *Backend* de las things que tengan la funcionalidad mediante un mensaje MQTT cambiar el fichero de configuración, sin embargo, este desarrollo es complejo a tanto a nivel de lógica como a nivel de seguridad, por tanto, se decidió dejarlo para futuras mejoras del sistema.

En conclusión, el trabajo ha sido largo pero una buena planificación y estudio de las posibilidades ha dado lugar a un prototipo funcional, profesional, seguro y con proyección. En siguientes pasos este prototipo puede dar lugar a un modelo viable de negocio que pueda facilitar a muchas empresas la gestión de su día a día en un mundo cada vez más deslocalizado y conectado.

7.1. Fortalezas y defectos

El trabajo ha centrado el desarrollo en cumplir los objetivos propuestos de manera ágil, ya que la extensión de un sistema de este calado puede llevar el proyecto al fracaso dada la vasta extensión de los sistemas y opciones.

La principal fortaleza del proyecto es la claridad de la arquitectura y su modularidad, el sistema está pensado para ser versátil, es decir, la arquitectura puede adaptarse con relativa facilidad nuevos cambios en cualquiera de las capas de implementación. La incorporación de nuevos módulos (aplicaciones, sensores…) es trivial y no supone esfuerzo a nivel de implementación lo cual aporta ventaja en un mundo en el que hay que los cambios y novedades son algo muy frecuente.

El licenciamiento también puede considerarse una fortaleza, es 100% Open Source, lo cual reduce en gran medida el coste del proyecto y de futuras implementaciones.

Por otro lado, las things, aunque a costa de un mayor precio, son también versátiles a la hora de exigirles trabajo extra, esto asegura que el sistema puede ser lo suficientemente genérico para abordar el grueso de las situaciones sin necesitar demasiado esfuerzo técnico ni material.

En contrapartida, y dada la condición del prototipo, la configuración de nuevas things quizás es algo más compleja de lo que este TFG pretendía, el desarrollo de módulos de configuración de las things tanto en la plataforma de gestión como en el *Backend* del servidor y las things habría sido lo ideal, sin embargo, esto requiere una segunda fase de desarrollo y un tiempo que se ha invertido en mejorar la usabilidad de la plataforma WEB.

7.2. líneas de trabajo futuro

En el prototipo funcional de IoT que se ha presentado se han concluido la mayoría de las ideas que se habían planteado en un principio. No obstante, muchas otras ideas o carencias han ido surgiendo en el desarrollo que deberían ser abordadas en futuras implementaciones.

Mejoras de la configuración remota: El configurar de manera ágil una thing tiene que ser una de las primeras cosas a abordar en futuros Sprint de desarrollo. La idea es que la plataforma sea accesible, y esto es clave para conseguirlo.

Mejoras en la seguridad: Una de las sobras del IoT, sin embargo, con los conocimiento adecuados y los suficientes medios se puede construir un sistema tan seguro como cualquiera (o más) de los que se ven actualmente en cualquier organización. Se debe trabajar especialmente en los siguiente apartados:

- Seguridad en MQTT
- Encriptado de los datos mediante certificados
- Control más estricto en el BackOffice
- Configuración exhaustiva del Broker

Inclusión de nuevas placas a la plataforma: La inclusión de nuevas placas de prototipado como Arduino u otras daría a la plataforma muchas más posibilidades de ajustar el sistema a soluciones concretas.

Mejora en el despliegue del lado del servidor: Estudiar e implementar una mejor manera de despliegue del sistema de gestión del lado del servidor puede ser una línea de mejora interesante para robustecer el sistema y facilitar su despliegue.

Optimización de los datos: El alcance de este TFG no compete la mejora y optimización de la base de datos, no obstante, en IoT la gestión de estos es importante, por tanto, tener foco en la mejora de la estructuras de datos para la visualización y explotación es importante.

Nuevos medios de comunicación: La capa de comunicaciones de n sistema IoT pude variar mucho según la situación del proyecto. El tener contemplado en el sistema diferentes tecnologías como LoRa o Zigbee puede ser interesante para aportar valor.

8. Glosario

Backend: Parte de la arquitectura que interactúa con el sistema en la parte del servidor. Este software es el encargado de procesar los datos y tomar decisiones basada en la programación y/o configuración.

Things: En referencia al mundo del IoT, las Things (cosas en inglés) son los sistemas encargados de interactuar mediante hardware y software con los sensores que miden o actúan en los entornos finales de los sistemas.

Templates: Combinación de archivos de código que componen la parte visual de un sitio web. Se utilizan para lograr sitios web uniformes y estructurados con una lógica de programación concreta.

Broker: En arquitecturas publicación/subscripción, el Broker es el gestor que se encarga de procesar los datos e identificar subscriptores y mensajes. Además, es responsable de identificar clientes y temas de interés para reenviar los mensajes a los interesados según su configuración.

Mesh: en tecnologías de comunicación, mesh hace referencia a un conjunto de dispositivos cualesquiera que utilizan un lenguaje en común para comunicarse, actuando como repetidores de señal para los siguientes dispositivos formando una “malla”.

QoS: En inglés *quality of service,* hace referencia a la calidad de un servicio de comunicaciones visto por un usuario.

Framework: Conjunto de herramientas y librerías de desarrollo que se utilizan para facilitar la creación de software haciendo hincapié en la resolución de los problemas más comunes

HTML: Del inglés *HyperText Markup Language,* se utiliza como lenguaje de marcado para elaborar páginas web principalmente. Es un estándar a cargo de *World Wide Web Consortium (W3C)* o Consorcio WWW.

9. Bibliografía

Amazon AWS IoT - aws.amazon.com. (n.d.). https://aws.amazon.com/es/iot/?nc=sn&loc=0

ASHTON, K. (n.d.). *That 'Internet of Things' Thing*. https://www.rfidjournal.com/that-internet-of-things-thing

Auth-mysql in EMQX Broker - docs.emqx.io. (n.d.). https://docs.emqx.io/en/broker/v4.3/advanced/auth-mysql.html#mysql-connection-information

Broker EMQX - www.emqx.io. (n.d.). https://www.emqx.io/products/broker

Comparison of MQTT implementations - en.wikipedia.org. (n.d.). https://en.wikipedia.org/wiki/Comparison_of_MQTT_implementations

Crecimiento Dispositivos conectados 2025 @ signalsiot.com. (n.d.). https://signalsiot.com/el-crecimiento-de-dispositivos-conectados-iot-generara-794-zb-de-datos-en-2025/

Data of RPICT3V1 - lechacal.com. (n.d.). http://lechacal.com/wiki/index.php?title=RPICT3V1

Debian 10 Install Manual (www.debian.org). (n.d.). https://www.debian.org/releases/stable/installmanual

Dirección General de Industria y de la PYME. (2020). Cifras PYME. *Portal PYME*, 1–3. http://www.ipyme.org/es-ES/publicaciones/Paginas/estadisticaspyme.aspx

Django Auth documentation (docs.djangoproject.com). (n.d.). https://docs.djangoproject.com/es/3.2/topics/auth/

Django Software Foundation. (2020). Writing your first Django app. In *Django Documentation* (pp. 13–65). https://media.readthedocs.org/pdf/django/3.0.x/django.pdf

Download EMQX Manual - www.emqx.io. (n.d.). https://www.emqx.io/downloads#broker

García Munguía, M., Molina Ruíz, H. D., Cornejo Velázquez, M., Moreno Gutiérrez, S. S., & Alvarado Reséndiz, J. L. (2020). Internet de las cosas. *TEPEXI Boletín Científico de La Escuela Superior Tepeji Del Río*, *7*(14), 46–51. https://doi.org/10.29057/estr.v7i14.5698

How to program an Attiny85 or Attiny84 - lechacal.com. (n.d.). http://lechacal.com/wiki/index.php/How_to_program_an_Attiny85_or_Attiny84

Install DJANGO Manuals and guides (docs.djangoproject.com). (n.d.). https://docs.djangoproject.com/en/3.2/intro/install/

IOT Google - Cloud.Google.Com. (n.d.). https://cloud.google.com/solutions/iot?hl=es-419

Koziolek, H., Grüner, S., & Rückert, J. (2020). A comparison of mqtt brokers for distributed iot edge computing. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12292 LNCS*, 352–368. https://doi.org/10.1007/978-3-030-58923-3_23

Microsoft, Explore the benefits of Azure IoT - azure.microsoft.com. (n.d.). https://azure.microsoft.com/en-us/overview/iot/

MQTT. (n.d.). *MQTT: The Standard for IoT Messaging - Mqtt.Org*. http://mqtt.org/

Number of publicly known Internet of Things (IoT) platforms worldwide from 2015 to 2019 - www.statista.com. (n.d.). https://www.statista.com/statistics/1101483/global-number-iot-platform/

OASIS. (2014). MQTT Version 3.1.1. *OASIS Standard*, *October*, 81. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

Obtener Debias 10 BUSTER - www.debian.org. (n.d.). https://www.debian.org/distrib/index.es.html

Paho Eclipse foundation - www.eclipse.org. (n.d.). https://www.eclipse.org/paho/

PAHO mqtt 1.5.1 - descripcion Funcionamiento @ pypi.org. (n.d.). https://pypi.org/project/paho-mqtt/#known-limitations

PM2 Quick Start (pm2.keymetrics.io). (n.d.). https://pm2.keymetrics.io/docs/usage/quick-start/

Raspberry PI, Installing operating system images - www.raspberrypi.org. (n.d.). https://www.raspberrypi.org/documentation/installation/installing-images/

Relé, Derscripcion y funcionamiento - es.wikipedia.org. (n.d.). https://es.wikipedia.org/wiki/Relé

ThingWorx Industrial IoT Platform - developer.thingworx.com. (n.d.). https://developer.thingworx.com/en/platform

Writing your first Django app (docs.djangoproject.com). (n.d.). Retrieved April 29, 2021, from https://docs.djangoproject.com/en/3.2/intro/tutorial01/

10. Anexos

ANEXO 1: Características técnicas Raspberry Pi (*pinout*)

ANEXO 2: Características Sensor DHT22

ANEXO 3: Placa RPICT3V, características e instrucciones

ANEXO 4: Características Sensor TMP36