Escola Universitària Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA DE CATALUNYA

GRADO EN INGENIERÍA INFORMÁTICA

Comparativa y estudio de plataformas IoT

AUTOR: RODRIGO MARTINEZ JACOBSON
TUTOR: PERE BARBERAN AGUT

PRIMAVERA 2017

TecnoCampus Mataró-Maresme

Dedicatoria

A mi familia, por darme confianza siempre que tenía dudas,

por darme ánimos cuando los necesitaba, y porque gracias

a ellos estoy donde estoy

Resum

Aquest projecte té com a objectiu analitzar i comparar diferents plataformes IoT, per obtenir coneixements sobre el mercat actual i les solucions disponibles.

El projecte analitzarà una selecció de les plataformes més interessants per funcionalitats i enfocament. D'aquesta mostra de plataformes, s'escolliran dues per ser analitzades més a fons, mostrant el procés de connexió d'un dispositiu, la recol·lecció de dades, anàlisi i representació gràfica d'aquests.

Resumen

Este proyecto tiene como objetivo analizar y comparar diferentes plataformas IoT, para obtener conocimientos sobre el mercado actual y las soluciones disponibles.

El proyecto analizará una selección de las plataformas más interesantes por funcionalidades y enfoque. De esa muestra de plataformas, se escogerán dos para ser analizadas más a fondo, mostrando el proceso de conexión de un dispositivo, recolección de datos, análisis y representación gráfica de estos.

Abstract

The objective of this project consists in analysing and comparing different IoT platforms, to obtain some insights about the actual market and the available solutions.

The project will analyse a selection of the most interesting platforms. Two platform from this selection will be analysed in a deeper way, showing the process to connect a device, collect data, analyse and represent that data graphically.

Índex contenido

Índice de tablas

Índice imágenes

1. Objetivos

1.1. Propósito

Realizar un estudio de mercado sobre una selección de plataformas IoT que representen los diferentes nichos de mercado del segmento.

Además, se realizarán pruebas de concepto sobre dos plataformas seleccionadas después del estudio de mercado. Estas pruebas de concepto consistirán en conectar un dispositivo a la plataforma, recopilar datos de un sensor, enviar los datos a la plataforma, analizarlos y/o almacenarlos, y por ultimo mostrar una representación gráfica del resultado del análisis de los datos.

1.2. Finalidad

Obtener un conocimiento tanto teórico como practico sobre las diversas plataformas disponibles tanto comerciales como de código abierto. Además, realizar pruebas con un dispositivo para comprobar cuál es el proceso a seguir para recolectar datos desde un sensor, enviarlos a la plataforma y realizar ciertas transformaciones sobre estos.

1.3. Objeto

Un documento que contenga el análisis de mercado de las plataformas IoT, y las pruebas de concepto realizadas sobre las dos plataformas escogidas, junto con el código necesario para replicar dichas pruebas.

1.4. Alcance

Realizar un análisis de mercado sobre las plataformas IoT y analizar dos en profundidad mediante pruebas de concepto para verificar las soluciones ofrecidas por estas.

Implementar los scripts necesarios para la obtención de datos mediante un sensor conectado a la placa de desarrollo y el envío de estos a la plataforma correspondiente.

Habilitar un panel dentro de las plataformas para visualizar gráficamente los datos recogidos por el dispositivo.

2. Justificación del trabajo

Para realizar la tarea requerida, se aplicarán conocimientos adquiridos durante los estudios. Además, se realizará una investigación sobre las plataformas disponibles en el mercado, tanto comerciales como de código abierto.

La finalidad de este trabajo será la de aportar una perspectiva del estado actual de las soluciones IoT en forma de plataformas, tanto para empresas como para usuarios domésticos.

Para las pruebas de concepto será necesario el uso de una placa de desarrollo con al menos un sensor que será el encargado de recoger los datos que, más adelante, se subirán a las plataformas.

Los lenguajes utilizados para el desarrollo de los scripts que gestionarán la recolección y el envío de datos serán Python y C, ya que son los que disponen de un mayor soporte de los fabricantes de sensores y de las plataformas IoT.

2.1. Arquitectura de una plataforma IoT

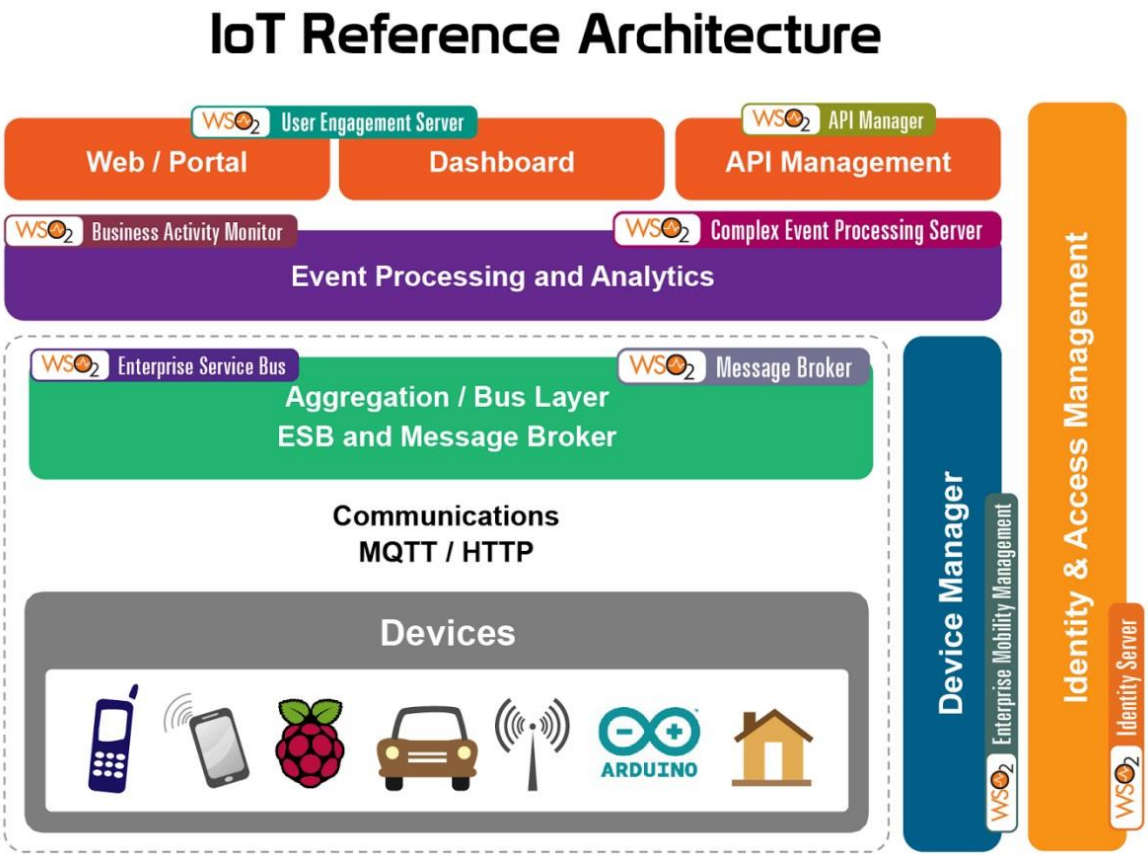


Imagen 1 Arquitectura Plataforma IoT

Para que una plataforma IoT pueda considerarse como opción dentro del desarrollo de un producto de este tipo, esta deberá poder gestionar de manera solvente la información, esto quiere decir:

- Ser capaz de recoger la información enviada por el dispositivo.
- Ser capaz de almacenar y/o analizar la información.
- Ser capaz de representar o exponer la información de forma que el usuario pueda hacer uso de esta.

Además, esta debe garantizar en todo momento la seguridad del sistema, para no exponer los datos a nadie que no haya sido autorizado previamente.

Por lo tanto, una plataforma IoT debería estar constituida, al menos, por los siguientes módulos o bloques:

- Conectividad y normalización: Permitir la conexión mediante protocolos, y la recepción de diferentes formatos de datos en una interfaz que garantice la precisa transmisión de datos y la interacción con los dispositivos.
- Almacenamiento de datos: Los datos deben ser almacenados para un posterior análisis, representación o integración con una herramienta propia o de terceros.
- Procesamiento y gestión de la acción: Los datos deben ser procesados para, según un conjunto de normas reglas o disparadores, ejecutar acciones dependiendo del valor resultante.
- Analítica y Visualización: Los datos deben de poder ser analizados y transformados, para luego poder ser visualizados mediante gráficos o expuestos en APIs para aplicaciones externas a la plataforma.

Por último, no es fundamental para una plataforma IoT contar con los siguientes módulos, pero sí que es recomendable para garantizar una mayor libertad del usuario a la hora de hacer uso de los datos que ha recogido:

- Gestión de dispositivos: La capacidad de gestionar los dispositivos de una forma flexible, mediante agrupaciones por localización, función u otros criterios, facilita la escalabilidad de las soluciones IoT basadas en una plataforma.
- Herramientas adicionales e interfaces externas: Si la empresa que ha desarrollado la plataforma cuenta con otras soluciones de software que podrían cubrir necesidades de sus clientes, o llega a un trato con algún proveedor de ciertos servicios relacionados, el ecosistema de esta plataforma crecerá y ofrecerá un conjunto de herramientas más completas para el usuario. Además, la capacidad de integrarse con sistemas o servicios de terceros permite suplir necesidades del negocio que no están en la plataforma.

2.2. Mercado de plataformas IoT

Al igual que el término IoT, una plataforma IoT es un concepto muy amplio. Puede tratarse de simples plataformas que sirven para almacenar datos y ofrecen interfaces estándares al usuario, hasta sistemas más completos que permiten el uso de herramientas para hacer predicciones, analíticas o para crear interfaces más complejas.

Una plataforma IoT debe de permitir recoger los datos enviados desde los diferentes dispositivos conectados. Por otra parte, debe de facilitar la creación de aplicaciones, tanto móviles como para otros dispositivos, que visualicen de manera clara los datos recibidos de los dispositivos IoT conectados a la plataforma, además de los datos sobre los que se ha trabajado.

Algunas de las plataformas IoT más famosas que se pueden encontrar en el mercado son las siguientes:

2.2.1. Amazon Web Services IoT

2.2.1.1. ¿Qué es?

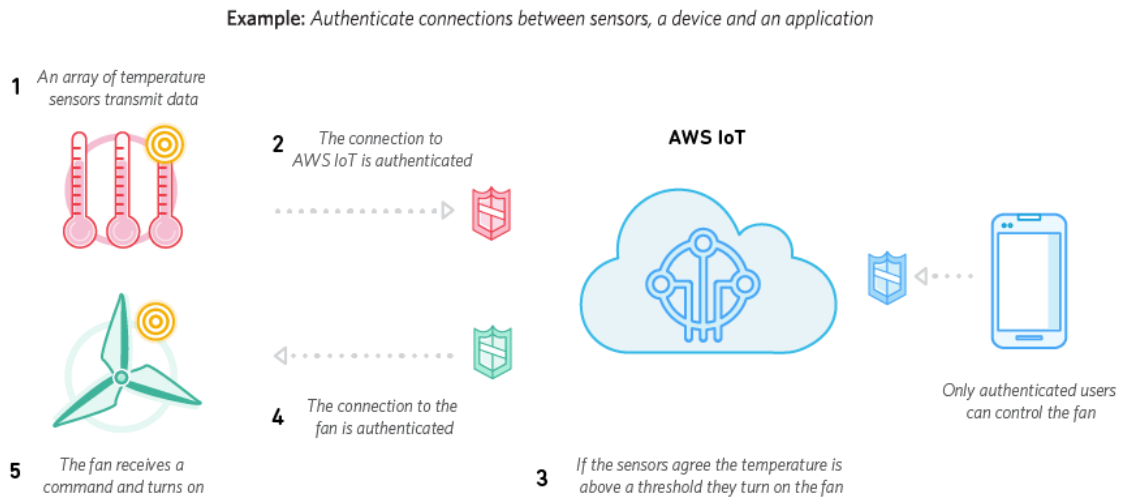


Imagen 2 Pasos de recolección de Datos AWS IoT

Es una plataforma que permite conectar diferentes dispositivos a la nube de Amazon Web Services. Además, desde esta plataforma se pueden crear interacciones entre los diferentes

dispositivos conectados, procesar los datos recibidos y crear aplicaciones para interactuar con los dispositivos.

2.2.1.2. ¿Qué ofrece?

AWS IoT facilita una integración con otros servicios de Amazon Web Services, entre ellos: AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning y Amazon DynamoDB, Amazon CloudWatch, AWS CloudTrail y Amazon Elasticsearch Service. De esta forma, se puede filtrar, transformar y utilizar los datos recibidos desde estos dispositivos para facilitar su uso desde las aplicaciones que los consumen.

Además, AWS ofrece una capa de seguridad con mecanismos de autenticación y cifrado integral en todos los puntos de conexión, con el fin de asegurar la privacidad y la veracidad de los datos.

2.2.1.3. ¿Cómo funciona?

Esta plataforma permite que dispositivos tales como sensores, accionadores y dispositivos incrustados se conecten mediante HTTPS, WebSockets o MQTT. También incluye un mecanismo de comunicación bidireccional seguro y de baja latencia llamado “Gateway para dispositivos”.

El motor de reglas incluido permite procesar de forma continua los datos recibidos. Estas reglas facilitan procesar los datos y almacenarlos o enviarlos a servicios ajenos a AWS mediante Lambda.

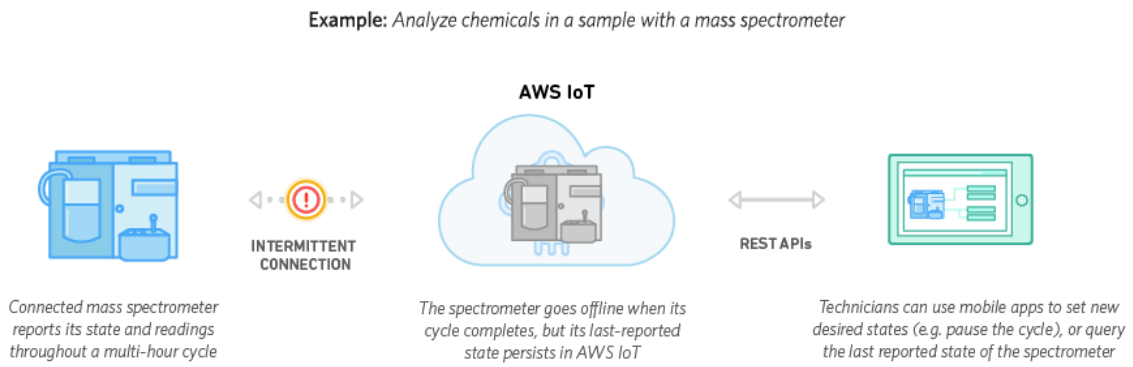


Imagen 3 Mecanismo de sombra de AWS IoT

Por otra parte, cuenta con un registro de dispositivos para registrar y seguir los dispositivos conectados a la plataforma. También proporciona un mecanismo llamado “sombras de los dispositivos”, que maneja la comunicación con los dispositivos y permite a la nube y a las aplicaciones consultar los datos enviados por los dispositivos y enviar comandos a estos.

2.2.1.4. Precios

Precios por región

Región	Precio
EE.UU. Este (Norte de Virginia)	5 USD por millón de mensajes
EE.UU. Este (Ohio)	5 USD por millón de mensajes
EE.UU. Oeste (Oregón)	5 USD por millón de mensajes
UE (Irlanda)	5 USD por millón de mensajes
UE (Frankfurt)	5 USD por millón de mensajes
UE (Londres)	5 USD por millón de mensajes
Asia Pacífico (Sídney)	6 USD por millón de mensajes
Asia Pacífico (Seúl)	6 USD por millón de mensajes
Asia Pacífico (Tokio)	8 USD por millón de mensajes
Asia Pacífico (Singapur)	8 USD por millón de mensajes

Imagen 4 Tabla de precios AWS IoT

Amazon cobra según el número de mensajes recibidos su plataforma AWS IOT y proporciona gratuitamente otros servicios de AWS, tales como: : Amazon S3, Amazon DynamoDB, AWS Lambda, Amazon Kinesis, Amazon SNS y Amazon SQS.

2.2.2. Azure IoT Hub

2.2.2.1. ¿Qué es?

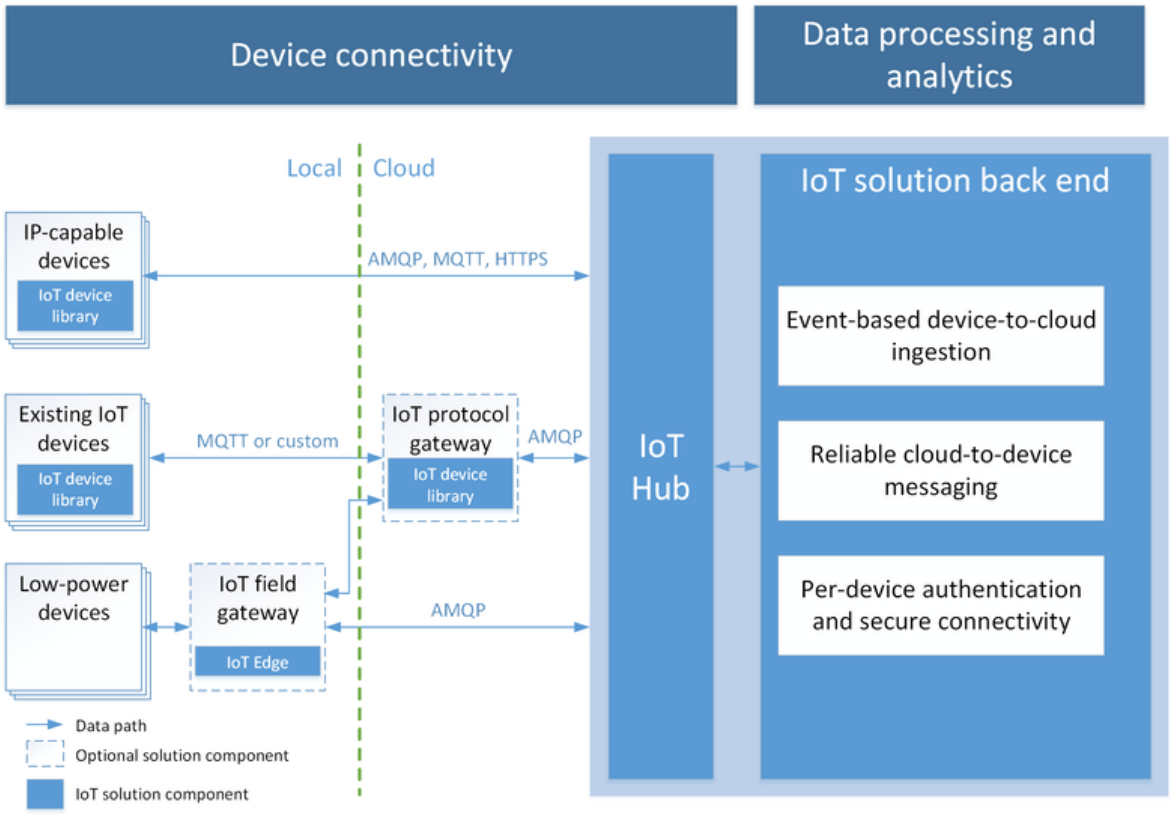


Imagen 5 Arquitectura Azure IoT

Es una plataforma que permite administrar la comunicación bidireccional, fiable y segura entre dispositivos IoT y un back-end de soluciones, tales como:

- IoT Hub (centro de comunicación con los dispositivos).
- Stream Analytics (servicio que permite analizar y hacer un procesado inicial de los datos).
- Event Hub (servicio para configurar y lanzar eventos que desencadenen acciones).
- Web Apps (se encarga de la parte visual o de una API de acceso).
- Bases de Datos (para almacenar datos procesados).
- Blobs de Almacenamiento (para almacenar los datos en crudo).

2.2.2.2. ¿Qué ofrece?

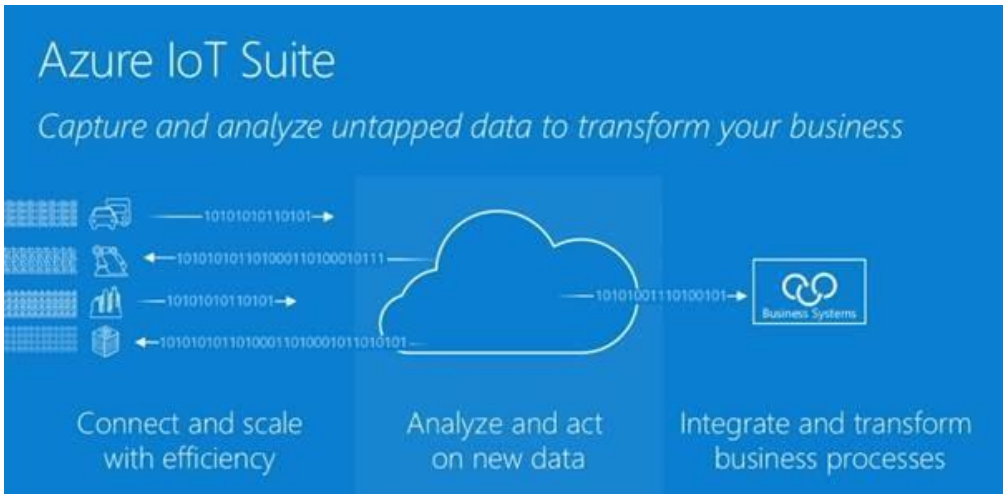


Imagen 6 Recolección de datos con Azure IoT

Microsoft ofrece un conjunto de aplicaciones IoT en Azure para crear escenarios específicos desde cero o desde soluciones pre configuradas.

Tales escenarios pueden ser:

- Almacenaje y sincronización de metadatos e información de estados entre dispositivos mediante “Dispositivos Gemelos”.
- Autenticación por dispositivo y conectividad segura mediante una clave segura única por dispositivo y un registro de identidades en IoT Hub.
- Definición de rutas de mensajes a partir de reglas de enrutamiento, con el fin de controlar desde donde son enviados los mensajes del dispositivo a la nube.
- Compatibilidad con un amplio conjunto de dispositivos mediante los SDK de dispositivo IoT de Azure, los cuales son compatibles con plataformas como Linux, Windows y sistemas operativos en tiempo real, y con lenguajes como C#, Java y Javascript.
- Detección de problemas de conexión con los dispositivos mediante la supervisión de operaciones de conectividad del dispositivo, que cuenta con un registro detallado sobre operaciones de administración de identidad y eventos de conectividad.

2.2.2.3. ¿Cómo funciona?

Azure IoT implementa el modelo de comunicación asistida por servicio para mediar en las interacciones entre los dispositivos y su back-end de soluciones. Esta comunicación establece rutas de acceso de comunicación bidireccional y de confianza entre los dispositivos y un sistema de control.

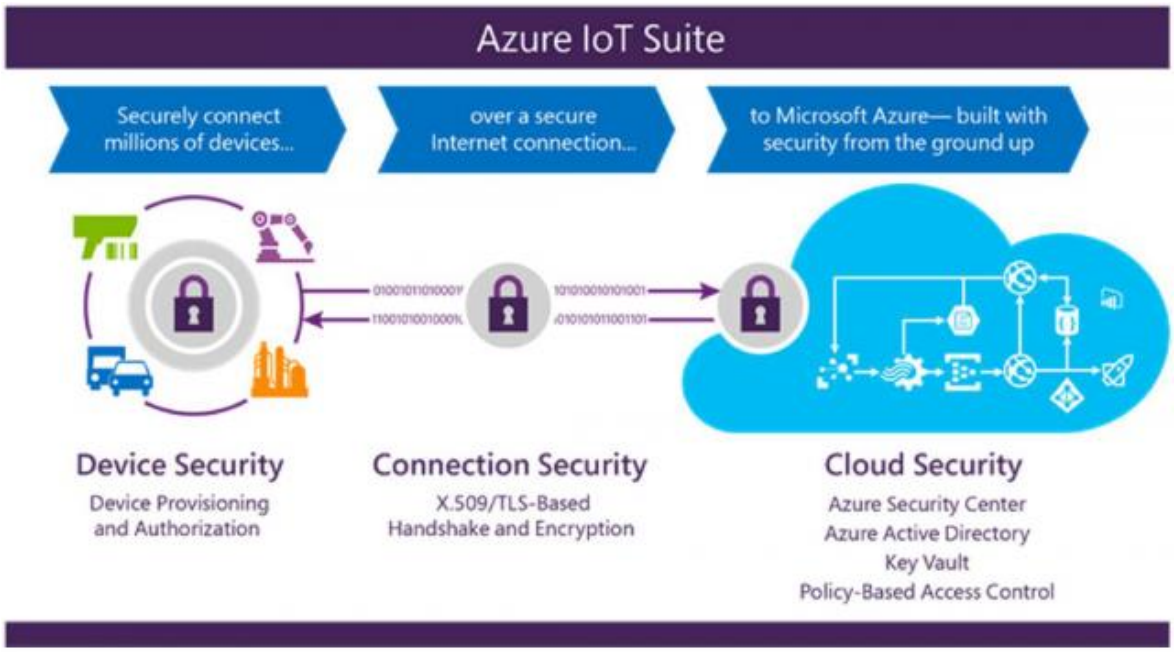


Imagen 7 Mecanismos de seguridad en Azure IoT

La comunicación se realiza mediante los protocoles MQTT v3.1.1, HTTP 1.1 o AMPQ 1.0 de forma nativa. Otros protocolos son admitidos mediante Azure IoT Edge y la personalización de la puerta de enlace de protocolo de IoT Azure.

2.2.2.4. Precios

TIPO DE EDICIÓN	PRECIO POR UNIDAD (AL MES)	NÚMERO TOTAL DE MENSAJES POR DÍA Y POR UNIDAD	TAMAÑO DEL MEDIDOR DE MENSAJES
Gratis	Gratis	8.000	0,5 KB
S1	€42,17	400.000	4 KB
S2	€421,65	6.000.000	4 KB
S3	€4.216,50	300.000.000	4 KB

Imagen 8 Tabla de precios Azure IoT

Microsoft establece cuatro grandes segmentos a la hora de contratar un plan de servicio con Azure IoT. Estos se diferencian por dos factores: número de mensajes totales por día y tamaño medio del medidor de mensajes (los mensajes se fragmentan en bloques del tamaño indicado en la tabla).

Por otra parte, si se utiliza algún otro producto o solución back-end de Azure, se facturará por separado.

2.2.3. Oracle Internet of Things Cloud Service

2.2.3.1. ¿Qué es?

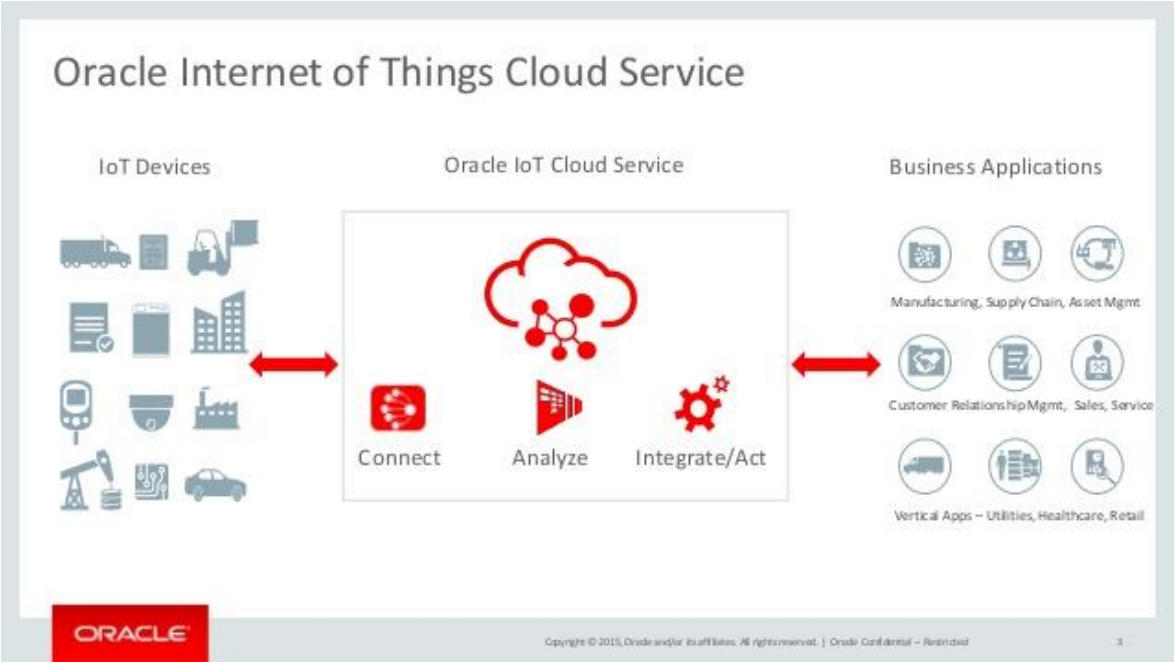


Imagen 9 Plataforma Oracle IoT Cloud

Es una plataforma que proporciona la posibilidad de conectar dispositivos en tiempo real a la nube y de analizar e integrar los datos recibidos con otras aplicaciones. El servicio se presenta como una plataforma como servicio (PaaS).

2.2.3.2. ¿Qué ofrece?

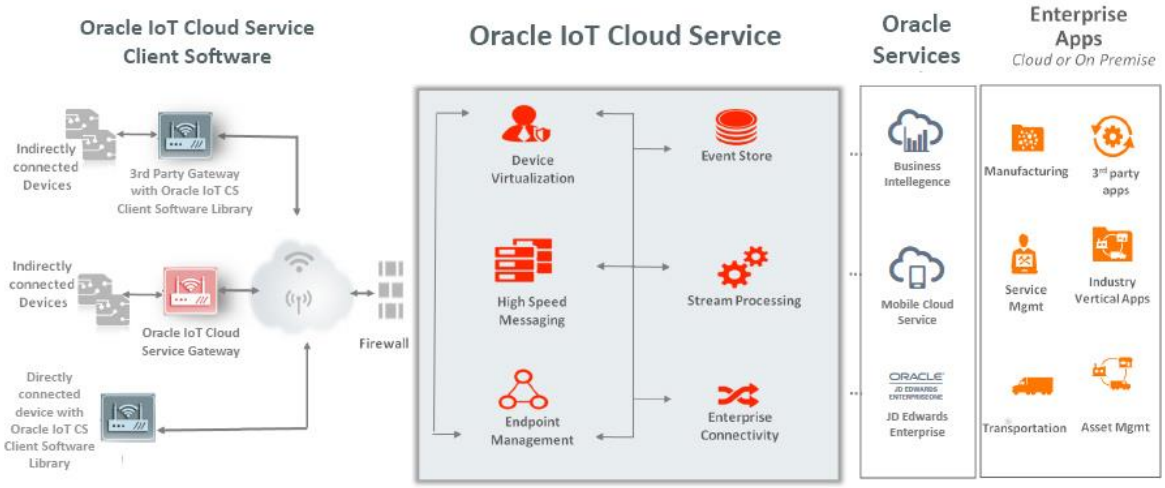


Imagen 10 Arquitectura Oracle IoT Cloud

La oferta de Oracle con su plataforma puede ser resumida en tres grandes puntos:

- Recopilación de datos de una forma segura y fiable desde cualquier dispositivo y mercado.
- Análisis de big data y predictivos en tiempo real con estadísticas del flujo de datos y los eventos IoT.
- Permite utilizar interfaces abiertas e integraciones con ofertas de PaaS y SaaS de Oracle.

Por otra parte, el servicio Oracle IoT incluye aplicaciones incorporadas de IoT tipo SaaS, las cuales facilitan la transformación y el análisis de los datos recogidos por los dispositivos.

2.2.3.3. ¿Cómo funciona?

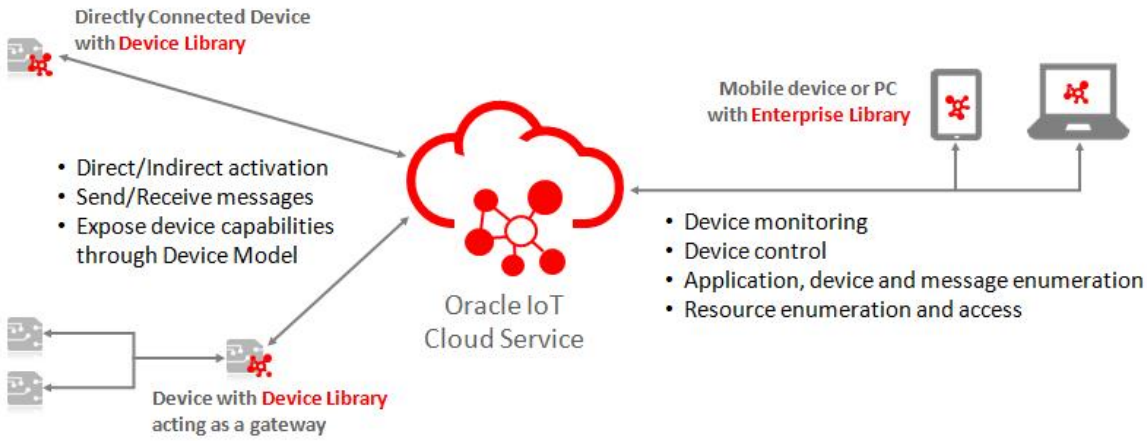


Imagen 11 Funcionamiento Oracle IoT Cloud

La plataforma de Oracle no tiene en cuenta el tipo de dispositivo, ya que estos se virtualizan en Oracle IoT Cloud Service al exponer el dispositivo como un conjunto de servicios, los cuales estarán siempre disponibles para la aplicación descendente. Esto evita que el desarrollador tenga que preocuparse acerca de la conectividad o la disponibilidad del dispositivo.

El dispositivo se puede conectar a la plataforma de las siguientes formas:

- Bibliotecas de clientes de Oracle IoT Cloud Service: Son binarios y código fuente que puede incluir el desarrollador en las aplicaciones con el fin de garantizar la conectividad segura y fiable del dispositivo con la plataforma. Estas bibliotecas están disponibles para una gran variedad de plataformas, entre ellas, C Posix, Windows, mbed, Java, Android, Javascript e iOS.
- Gateway de Oracle IoT Cloud Service: Aplicación Java SE Embedded que despliega dispositivos de Gateway. Esta aplicación proporciona un conjunto de funciones que asegura una comunicación fiable y segura e incluye un marco adaptador de dispositivos, lo cual permite capturar y conectar los dispositivos que no pueden conectarse directamente por el motivo que sea.
- API RESTful: Para dispositivos o desarrollos que no puedan utilizar las bibliotecas o los Gateway, los servicios de Oracle IoT Cloud se muestran como un conjunto de API RESTful.

2.2.3.4. Precios

Metered Services

Product	Price	Metric	Minimum
Oracle Internet of Things Cloud - Enterprise - Metered	€4,341.00	OCPU Per Month	2
Oracle Internet of Things Cloud - Enterprise - Metered	€7.00	OCPU Per Hour	2

Non-Metered Services

Product	Price	Metric	Minimum
Oracle Internet of Things Cloud - Enterprise - Non-metered	€2,171.00	OCPU	2

Imagen 12 Tabla de precios Oracle IoT Cloud

Oracle ofrece dos modalidades:

- Medido: Esta modalidad es la que permite escalar en caso de necesidad. Y en caso de cancelación en mitad de un período de facturación, solo se cobrará hasta el día de la baja. Dentro de esta modalidad hay dos opciones: Mensual o por horas. Además, la métrica utilizada es los ciclos de procesador.
- No Medido: Un precio fijo al mes, sin la posibilidad de escalar de manera reactiva.

Por otra parte, si es necesario el almacenamiento de datos en la nube u otros servicios ofrecidos por Oracle Cloud Service, se tendrán que contratar aparte.

2.2.4. Watson IoT

2.2.4.1. ¿Qué es?

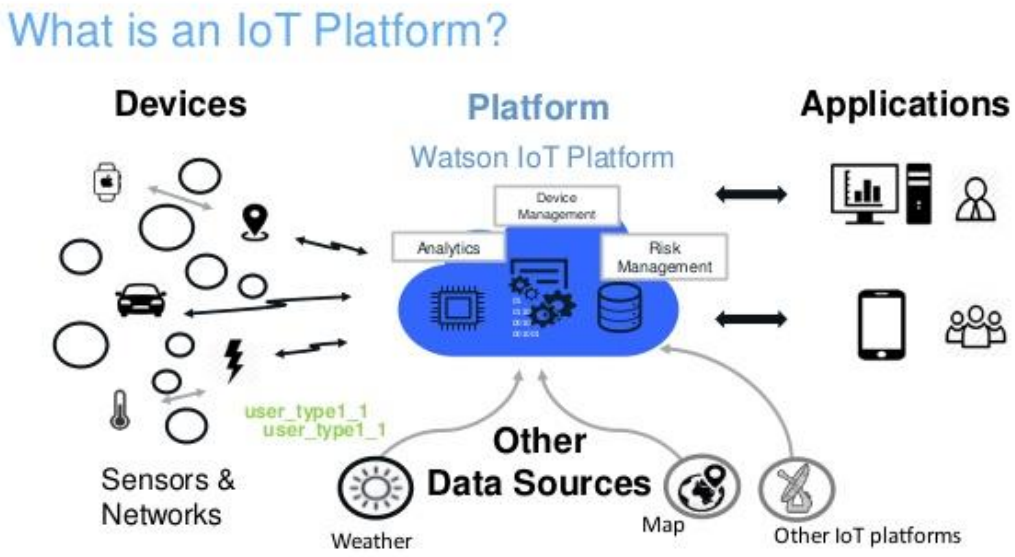


Imagen 13 Plataforma Watson IoT

Es un servicio ofrecido por IBM, completamente gestionado y alojado en la nube. Está diseñado para analizar, gestionar, transformar, almacenar y exponer los datos de los dispositivos IoT.

2.2.4.2. ¿Qué ofrece?

Permite configurar y gestionar los dispositivos conectados al servicio y, de esta forma, permite que las aplicaciones creadas y enlazadas a los dispositivos puedan tener acceso a los datos en tiempo real y al histórico.

Por otro lado, ofrece APIs seguros que vinculan las aplicaciones con los datos procedentes de los dispositivos conectados a la plataforma.

Además, permite el análisis de los datos sin salir de la plataforma mediante Bluemix.

IBM ha agrupado la plataforma en cuatro grandes bloques:

- IBM Watson IoT Platform Connect: Bloque que engloba la comunicación entre los dispositivos y la nube.
- IBM Watson IoT Platform Information Management: Permite la integración con datos de terceros y el almacenamiento de los datos recibidos por los dispositivos.
- IB Watson IoT Analytics: Provee a la plataforma la capacidad de realizar análisis predictivos, cognitivos, en tiempo real y contextuales.
- IBM Watson IoT Risk Management: Permite añadir a la plataforma una capa proactiva de seguridad y protección ante anomalías en el mundo del IoT

2.2.4.3. ¿Cómo funciona?

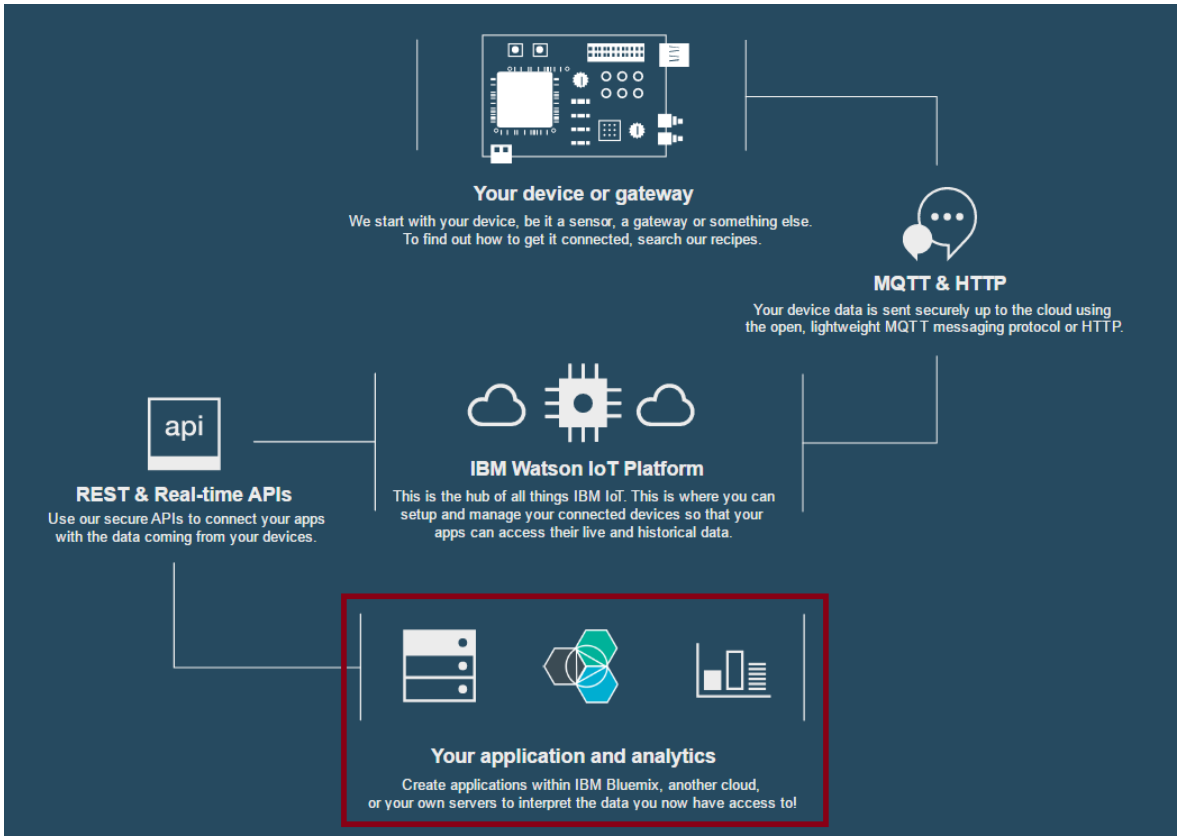


Imagen 14 Esquema de funcionamiento de Watson IoT

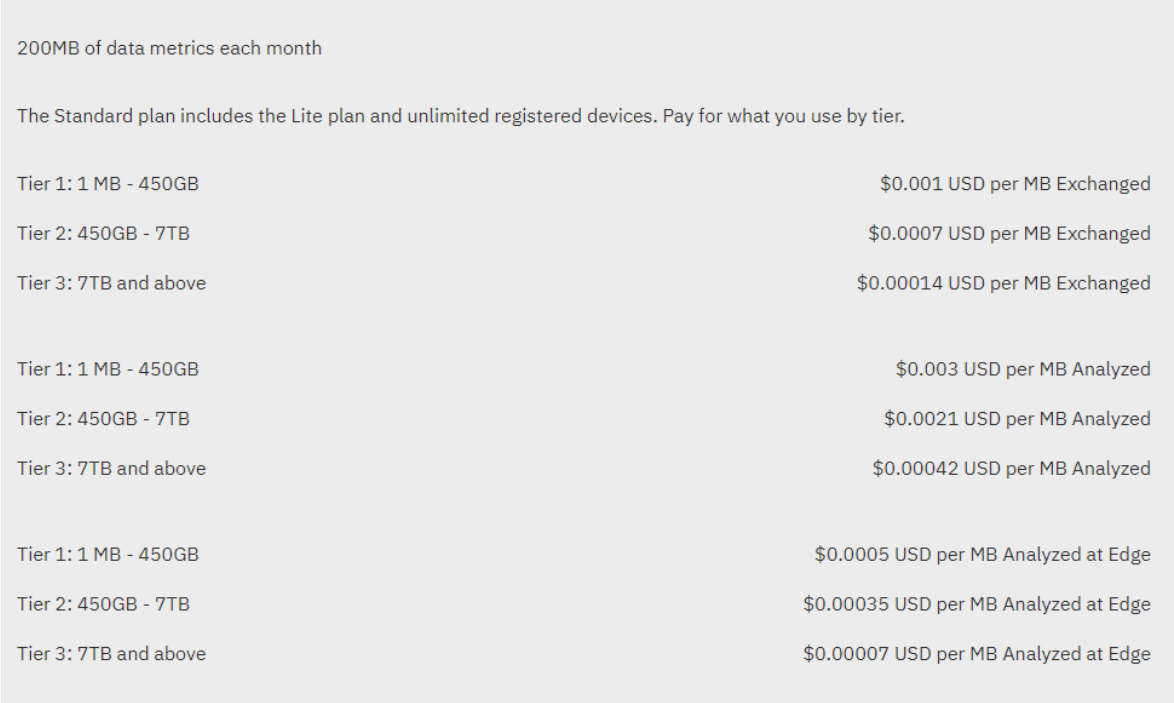
Los dispositivos son asignados a organizaciones, los cuales delimitan el dominio de la información de esos dispositivos. Cada organización representa una instancia de la plataforma.

IBM distingue dos tipos de dispositivos en su plataforma:

- Gestionados: Permiten al dispositivo conectarse al Device Management de la plataforma mediante el protocolo Device Managenement. De esta forma, este dispositivo puede ser gestionado de forma remota.
- No gestionados: Se conectarán a la plataforma mediante otros protocolos como MQTT y HTTPS, pero no podrán recibir ni operaciones ni solicitudes de gestión.

Los dispositivos conectados a la plataforma enviarán los datos y estos podrán ser analizados y transformados, para luego quedar expuestos mediante las APIs REST y en tiempo real de la plataforma.

2.2.4.4. Precios



200MB of data metrics each month

The Standard plan includes the Lite plan and unlimited registered devices. Pay for what you use by tier.

Tier 1: 1 MB - 450GB	$0.001 USD per MB Exchanged
Tier 2: 450GB - 7TB	$0.0007 USD per MB Exchanged
Tier 3: 7TB and above	$0.00014 USD per MB Exchanged
Tier 1: 1 MB - 450GB	$0.003 USD per MB Analyzed
Tier 2: 450GB - 7TB	$0.0021 USD per MB Analyzed
Tier 3: 7TB and above	$0.00042 USD per MB Analyzed
Tier 1: 1 MB - 450GB	$0.0005 USD per MB Analyzed at Edge
Tier 2: 450GB - 7TB	$0.00035 USD per MB Analyzed at Edge
Tier 3: 7TB and above	$0.00007 USD per MB Analyzed at Edge

Imagen 15 Tabla de precios de Watson IoT

La plataforma ofrece un plan gratuito de hasta 500 dispositivos y 200 MB de datos al mes. Para consumos superiores, ofrece tres tipos de planes. Todos los planes son medidos, y están diferenciados por las franjas de consumo.

La unidad utilizada como métrica son la cantidad de MB utilizados en intercambios de datos.

2.2.5. Xively

2.2.5.1. ¿Qué es?

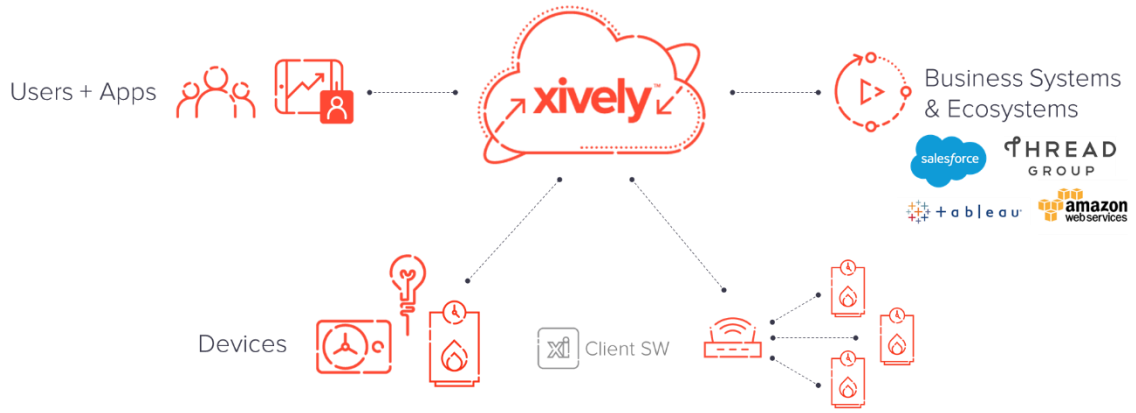


Imagen 16 Plataforma Xively

Es una plataforma IoT que incluye entre otros servicios de directorio, servicios de datos, motor de seguridad y aplicaciones de gestión web.

2.2.5.2. ¿Qué ofrece?

Xively ofrece una solución completa de gestión de productos conectados, la cual provee de las capacidades necesarias para gestionar aplicaciones relacionadas con el internet de las cosas.

Además de los ya mencionados servicios de datos, esta plataforma ofrece otros servicios, tales como:

- Gestión de identidad: Controla identidades y autenticación de forma segura.
- Series de tiempo: Permite guardar y analizar las series de datos históricos.
- Planos y mensajes: Controla cómo son representados los elementos y las identidades.

2.2.5.3. ¿Cómo funciona?

Imagen 17 Esquema de funcionamiento de Xively

Esta plataforma se basa en un sistema de mensajes mediante el protocolo MQTT, aunque su API soporta otros protocolos, tales como: HTTP REST y WebSockets.

Además, Xively proporciona diversas soluciones para que la integración de los dispositivos con la plataforma sea lo más accesible. Entre estas soluciones se encuentran un cliente IOT embebido para las comunicaciones mediante MQTT y un SDK para iOS y Android.

2.2.5.4. Precios

Xively no ofrece una tabla de precios y no facilita una estimación según una métrica definida. Es necesario contactar con el departamento de ventas para que realicen una estimación dependiendo del volumen de datos y la cantidad de dispositivos conectados.

Dispone de una cuenta de prueba, donde se limitará la cantidad de llamadas a la API a 25 por minuto, y los datos serán almacenados durante un máximo de 30 días.

2.2.6. Samsung Artik

2.2.6.1. ¿Qué es?

Imagen 18 Logotipo Artik Cloud

Plataforma de IoT integrada, desarrollada por Samsung a partir de la adquisición de SmartThings. Esta plataforma no sólo proporciona el software necesario, sino que también cuenta con una gama de dispositivos y placas de desarrollo fabricados por Samsung que integran librerías para facilitar la conectividad.

2.2.6.2. ¿Qué ofrece?

Esta plataforma nos permite recolectar datos de cualquier tipo de dispositivo capaz de consumir sus APIs. Además, nos proporciona una extensa selección de aplicaciones e integraciones con otras plataformas, tales como: Amazon Echo, Belkin WeMo y Nest.

El objetivo de Artik es unificar el hardware, el software, la nube, la seguridad y el ecosistema de servicios en una sola oferta integrada, a diferencia de otras plataformas como AWS IoT y Azure. Además, en todo momento se intenta ocultar la complejidad del internet de las cosas mediante las APIs, SDK y herramientas que proporciona.

Por otra parte, también proporciona los apartados básicos en este tipo de plataformas:

- Conexión segura y gestión de dispositivos.
- Manipulación, almacenamiento y visualización de datos.

Imagen 19 Placas de desarrollo Artik

En el apartado de hardware, además de permitir todo tipo de dispositivos, Samsung ofrece la gama Artik de módulos integrados, los cuales según la gama están enfocados a automatización, procesamiento de datos y geolocalización.

2.2.6.3. ¿Cómo funciona?

La plataforma requiere que cada dispositivo que vaya a interactuar con sus servicios sea dado de alta, mediante la creación de un manifiesto que describe los posibles estados y acciones del dispositivo.

Una vez configurado el dispositivo, el envío de los datos se produce mediante las llamadas a las APIs, utilizando protocolos de comunicación tales como REST / HTTP, WebSockets, MQTT y CoAP.

Además, mediante un framework se proporcionan unos conectores para acceder a servicios de terceros que exponen una API.

En el apartado de visualización, Samsung proporciona una herramienta de gráficos y análisis de datos propia.

2.2.6.4. Precios

En el apartado de precios Samsung establece tres segmentos:

- Plan gratuito: Hasta 100 mil mensajes al mes y los datos son almacenados un máximo de 30 días.
- Plan pequeños negocios: 15$ por cada 1 millón de mensajes, los datos son almacenados un máximo de 30 días.
- Plan hecho a medida: Dependiendo de las necesidades del cliente, con una cuota de mensajes y un período de retención de datos personalizado. Además, ofrece un SLA como garantía para grandes clientes.

2.2.7. Carriots

2.2.7.1. ¿Qué es?

Carriots es una plataforma como servicio (PaaS) diseñada para proyectos de internet de las cosas y de máquina a máquina. Esta plataforma nace en 2011 y su nombre es un acrónimo de "CARRying the Internet of ThingS".

Desde el primer momento, la plataforma fue pensada por y para la nube. No está contemplada una versión para arquitectura tradicional.

2.2.7.2. ¿Qué ofrece?

Esta plataforma permite recopilar y almacenar todo tipo de datos provenientes de los dispositivos conectados. Además, la plataforma incluye lo siguiente:

- Gestión simplificada de múltiples proyectos.
- Amplia variedad de APIs y un potente SDK que utiliza Groovy como lenguaje de desarrollo.
- Fácilmente escalable.

2.2.7.3. ¿Cómo funciona?

Los dispositivos envían datos a la API de la plataforma mediante mensajes HTTP REST / MQTT, con su contenido en formato XML o JSON. Esta información puede ser de cualquier tipo, ya que será almacenada en una base de datos NoSQL donde luego será procesada por las aplicaciones creadas dentro de la plataforma.

Las aplicaciones van desde un acercamiento básico con expresiones simples y un IF-THEN-ELSE, hasta un acercamiento más avanzado con una lógica más compleja mediante scripts GROOVY.

Una vez procesada, esta información puede ser descargada, publicada mediante una API REST proporcionada por la misma plataforma o enviada a otro sistema/plataforma mediante eventos.

2.2.7.4. Precios

La plataforma ofrece un plan gratuito de hasta 2 dispositivos, con 4 claves para conectar con su API. Además, la interacción con su API está limitada a 500 envíos de datos al día y 10 por minuto. Por último, también se limita el tamaño de los mensajes a un máximo de 5KB y el total de datos enviados a 5000KB al día. La retención de datos de este plan es de 3 meses.

Por otro lado, la plataforma dispone de otros dos planes más:

- Corporate: Enfocado a un uso mediante pocos dispositivos que envían gran cantidad de mensajes y de hasta 1MB. Este plan cuesta 2€ por dispositivo (con un mínimo de 11 dispositivos).
- Lite: Pensado para utilizarse con muchos dispositivos enviando no demasiados mensajes y de un tamaño muy pequeño. Este plan cuesta 0.50€ por cada dispositivo (con un mínimo de 100 dispositivos).

También ofrecen un plan a medida, donde el precio depende de las necesidades del cliente. Esta opción contempla la instalación de la plataforma en una nube privada.

2.2.8. Adafruit.IO

2.2.8.1. ¿Qué es?

Imagen 20 Logotipo de Adafruit.IO

Es una solución para la construcción de aplicaciones IoT creada por Adafruit, comercializadora de hardware open-source. Actualmente está en beta.

Está construida en Ruby on Rails y Node.js y está enfocada al usuario doméstico.

2.2.8.2. ¿Qué ofrece?

Esta plataforma permite recolectar datos y analizarlos mediante conexiones de datos simples y muy poco código.

Incluye librerías para clientes que envuelven sus APIs REST y MQTT. Estas librerías están disponibles para Arduino, Ruby, Python y Node.js.

2.2.8.3. ¿Cómo funciona?

Su funcionalidad es bastante básica, ya que no contempla la gestión de dispositivos ni de usuarios.

En esta plataforma no se puede gestionar cómo se almacena la información, ya que pasa directamente del dispositivo (mediante las librerías que facilitan las peticiones a su API rest) al panel donde se puede visualizar los datos. No se permite descargar o exponer estos datos a ninguna otra plataforma.

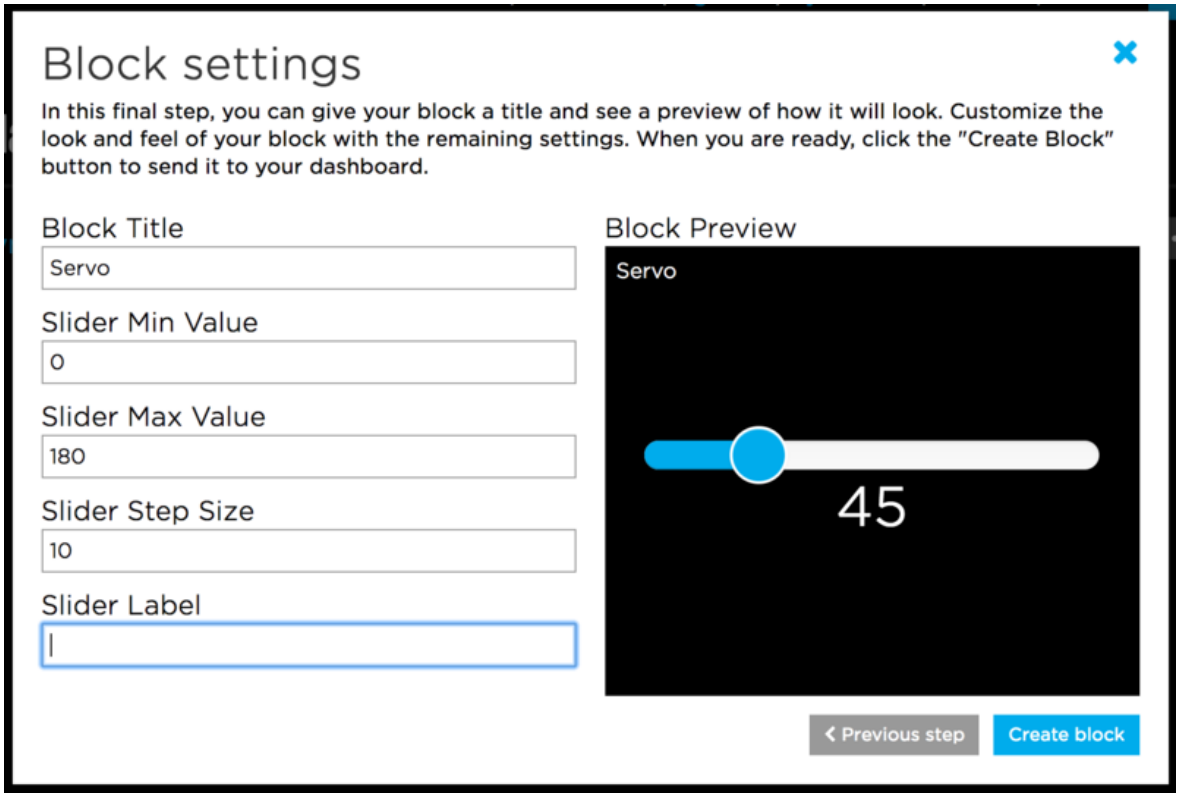


Imagen 21 Configuración de un componente

En el apartado de visualización, los paneles se crean mediante un editor bastante sencillo y directo que limita a ciertos tipos de componentes y donde el título que se asigna al componente a su vez será el nombre de la variable referenciada desde el código.

Más de un dispositivo puede enviar datos al mismo panel.

2.2.8.4. Precios

Esta plataforma es completamente gratuita y sólo tiene restricciones en la cantidad de datos que puede recibir:

- Máximo de 50000 datos a la vez. Es FIFO, esto quiere decir que a medida que entran datos nuevos se borran los antiguos.
- Máximo de 10 dispositivos por cuenta, y 5 paneles de visualización de datos.
- Máximo de 125 mensajes por minuto.
- La información se almacena un máximo de 30 días.

2.2.9. Ubidots

2.2.9.1. ¿Qué es?

Es una plataforma IoT en la nube que permite almacenar e interpretar información de sensores en tiempo real.

Ubidots hace posible la creación de aplicaciones para el internet de las cosas de una manera rápida, fácil y divertida.

2.2.9.2. ¿Qué ofrece?

Esta plataforma nos permite enviar datos y configurar eventos para que la API reaccione en tiempo real según el valor de los datos.

Mediante estos eventos, la plataforma permite el envío de alertas mediante Email/SMS.

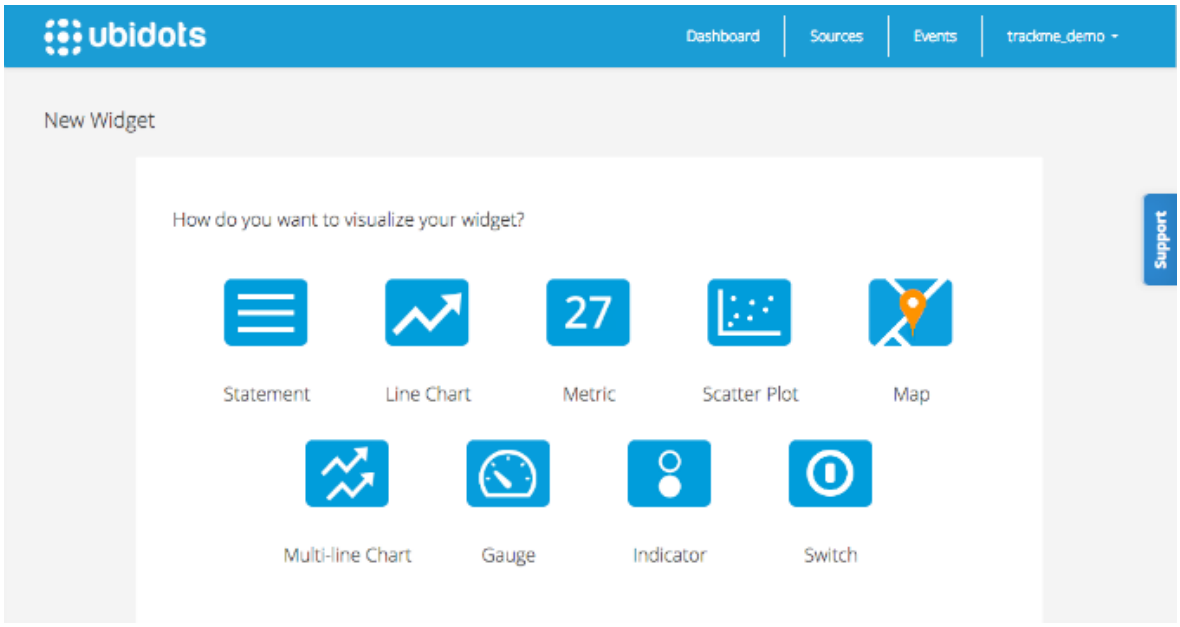


Imagen 22 Creación de un panel con Ubidots

En el apartado de visualización, permite crear paneles mediante un editor gráfico y sencillo.

Es compatible con un gran número de dispositivos, tales como: Arduino, Raspberry Pi, Android, Spark.io, Tessel y cualquier dispositivo capaz de enviar peticiones HTTP.

Proporciona una completa colección de librerías para envolver las llamadas a la API. Los lenguajes disponibles para estas librerías son: Python, Java, C, PHP, Node, Ruby.

Por último, dispone de una documentación en varios idiomas muy completa, con toda la información detallada y ejemplos.

2.2.9.3. ¿Cómo funciona?

La plataforma no permite gestionar dispositivos, pero sí que permite identificarlos. Estos dispositivos son referenciados como “Data Source” en los paneles donde se muestra la información de forma gráfica.

Su API permite enviar, leer, editar y borrar la información almacenada mediante los métodos estándares de HTTP: GET, POST, PUT, DELETE.

Los mensajes se envían desde el dispositivo mediante peticiones REST utilizando el protocolo HTTP. Estas peticiones se pueden construir con los clientes disponibles en varios lenguajes o manualmente desde un script propio.

XML	JSON

<item>
<value>1000</value> <context>
<estado>encendido</estado> </context>
 	
</item>
	{
	“value”: 1000, “context”: { “estado”:
	“encendido” }
	
	}

Imagen 23 Formatos disponibles para enviar información

Los formatos de datos admitidos en estos mensajes son XML y JSON.

2.2.9.4. Precios

En el apartado de precios Ubidots solo contempla dos opciones:

- Ubidots para la educación: Este plan es gratuito, permite hasta 20 dispositivos y 60 transacciones por minuto con su API. La retención de datos es de 3 meses.
- Ubidots para empresas: Plan hecho a medida dependiendo de las necesidades del cliente.

2.2.10. myDevices Cayenne

2.2.10.1. ¿Qué es?

Plataforma de IoT orientada a simplificar la creación de soluciones para el mundo conectado. Cayenne es la primera plataforma en conseguir una solución basada en arrastrar y soltar dentro del mundo del internet de las cosas.

2.2.10.2. ¿Qué ofrece?

Esta plataforma destaca por la sencillez a la hora de conectar el dispositivo, analizar los datos y representarlos gráficamente. Estos son algunos de sus puntos fuertes:

- Aplicación móvil, la cual permite configurar, monitorizar y controlar los sensores conectados a los dispositivos.
- La capacidad de configurar un dispositivo con sus sensores y actuadores en cuestión de minutos.
- Un motor de reglas que posibilita el desencadenar acciones a través de dispositivos.
- Panel con widgets para la visualización de los datos recogidos por los dispositivos.
- Control directo de los pines GPIO de la placa de desarrollo.

2.2.10.3. ¿Cómo funciona?

La plataforma utiliza una API MQTT para gestionar la conexión y envío de datos. Para utilizar esta API son necesarias las librerías y los controladores de sensores proporcionados por myDevices.

Al ser necesario el uso de las librerías, la compatibilidad de hardware es bastante reducida por el momento. Los dispositivos compatibles actualmente son:

- Raspberry Pi (1,2,3 y Zero).
- Arduino y compatibles.
- Dispositivos LoRa: dispositivos que utilizan una tecnología inalámbrica especialmente pensada para IoT.

Por otro lado, disponen de una API REST mediante el protocolo HTTP y con una capa de seguridad mediante OAuth2. Esta API permite gestionar y listar ciertos recursos de la cuenta de desarrollo, pero no permite acciones relacionadas con la configuración del dispositivo ni con el envío de datos desde este.

Una vez configurado el dispositivo compatible, desde la aplicación móvil o el navegador se puede realizar la conexión con la plataforma. Cuando esta conexión sea establecida correctamente, Cayenne redirigirá al usuario al panel del proyecto, donde se podrán crear los widgets necesarios para representar la información recogida por el dispositivo.

2.2.10.4. Precios

Actualmente la plataforma es gratuita, ya que no dispone de un plan de precios ni de unas limitaciones según métricas.

En un futuro próximo, myDevices tiene pensado lanzar planes a medida para fabricantes y empresas, pero su intención es que la plataforma permanezca como gratuita para los desarrolladores y miembros de la comunidad.

2.2.11. Macchina.IO

2.2.11.1. ¿Qué es?

Macchina.IO es un conjunto de herramientas para el desarrollo de aplicaciones de sistemas embebidos del internet de las cosas.

Es Open-Source y proporciona todo lo necesario para permitir que aplicaciones se comuniquen con diversos sensores, dispositivos y servicios en la nube.

2.2.11.2. ¿Qué ofrece?

Imagen 24 Arquitectura de Macchina.IO

Este conjunto de herramientas ofrece todo lo necesario para el desarrollo de aplicaciones orientadas al internet de las cosas. Estas herramientas están agrupadas de la siguiente forma:

- Plataforma: Dentro de este grupo se encuentran las herramientas necesarias para la recolección y análisis de datos.
- Componentes IoT: En este grupo se encuentra todo lo relacionado con la interacción directa con los dispositivos. Para ello, se facilitan cosas como los controladores de los sensores, los protocolos de comunicación con la plataforma y los servicios de eventos.

El conjunto de herramientas incluye una solución para las notificaciones mediante Email y SMS.

No se proporciona una solución de hosting, por lo tanto, la plataforma siempre se ejecutará en una nube contratada por el usuario o bien en una privada.

2.2.11.3. ¿Cómo funciona?

Mediante los componentes IoT facilitados por Macchina.IO se definen los dispositivos y los sensores a través de interfaces genéricas. Estas interfaces se encargarán de mapear la distribución de los puertos GPIO y los sensores conectados a los dispositivos.

La plataforma se comunica con los dispositivos de internet de las cosas mediante el protocolo MQTT.

Una vez establecida la comunicación, la plataforma recogerá la información y la guardará en la base de datos para más adelante exponer la información o analizarla.

Con los servicios proporcionados, Macchina.io permite que los datos sean consumidos por otra plataforma o bien, analizados y mostrados mediante gráficos desde la interfaz web.

2.2.11.4. Precios

El conjunto de herramientas de Macchina.IO es completamente gratuito, ya que es Open-Source. El gasto que conlleva esta solución depende del hosting escogido para el despliegue de la plataforma, sea en una nube privada o en otras tales como Amazon Web Services, Azure o Google Cloud.

2.2.12. ThingSpeak

2.2.12.1. ¿Qué es?

Plataforma que permite la recolección y almacenamiento de datos recolectados por dispositivos IoT, a la vez que el desarrollo de aplicaciones orientadas a aprovechar esos datos y actuar dependiendo de unos parámetros pre configurados.

Esta plataforma es abierta y de código libre. Su código está disponible en GitHub y dispone de una gran comunidad de desarrolladores dando soporte a una gran variedad de dispositivos.

2.2.12.2. ¿Qué ofrece?

Esta plataforma ofrece tanto un hosting en la nube gestionado por la desarrolladora (MathWorks), como el código de su plataforma para modificarlo y ejecutarlo en servidores propios o en la nube.

Ofrece compatibilidad oficial con los siguientes dispositivos:

- Arduino.
- Los modulos Photon y Electron de Particle.
- Modulo Wifi ESP8266.
- Raspberry Pi.

Por otro lado, cualquier dispositivo capaz de comunicarse con las APIs RESTful proporcionadas es capaz de comunicarse con la plataforma.

Esta solución dispone de una amplia lista de aplicaciones y plataformas con las cuales integrarse, tales como Twitter y Twilio, para permitir las notificaciones a través de diversos canales.

Por último, el análisis de datos y la visualización de estos es mediante MATLAB, desarrollado por la misma empresa matriz que la plataforma.

2.2.12.3. ¿Cómo funciona?

La plataforma trabaja sobre un concepto base denominado “Canal ThingSpeak”. Un canal almacena la información que envían los dispositivos, y se compone de las siguientes partes:

- Ocho campos para almacenar datos de todo tipo: Este tipo de campos pueden ser utilizados para almacenar datos de un sensor o un dispositivo embebido.
- Tres campos para almacenar información: Se utilizan para guardar latitud, longitud y elevación.
- Un campo denominado estado, el cual se utiliza para describir la información almacenada en el canal.

Los datos viajan desde los dispositivos al API mediante los protocolos HTTP y MQTT.

En esta plataforma los dispositivos no se gestionan, ya que todo gira en torno al concepto de Canal.

Una vez creado el canal, los dispositivos pueden enviar mensajes con los atributos definidos en el canal. Si el mensaje tiene el formato correcto, la plataforma lo almacenará y lo analizará.

Más adelante, con la integración de MATLAB se puede analizar la información y presentarla de manera gráfica. En caso de querer utilizar otra solución de terceros, es posible exponiendo la información mediante otra API RESTful.

Por último, se pueden configurar eventos para enviar notificaciones o accionar respuestas de los dispositivos, como, por ejemplo, encender un led como mensaje de error.

2.2.12.4. Precios

En lo referente a precios, Thingspeak ofrece los siguientes planes para alojar la plataforma en su nube:

Plan	Gratuito	Estudiante	Hogar	Académico	Estándar
Uso	Para pequeños proyectos no comerciales y para evaluar el servicio.	Para estudiantes de instituciones educativas.	Para uso personal.	Para uso del personal académico de instituciones educativas.	Para uso comercial, gubernamental y actividades que generan beneficios.
Escalable	No	Sí	Sí	Sí	Sí
Número de mensajes	3M / Año	33M / Año	33M / Año	33M / Año	33M / Año
Limitación de tiempo entre mensajes	1 cada 15 segundos	1 cada segundo	1 cada segundo	1 cada segundo	1 cada segundo
Tiempo máximo de computo	20 segundos	20 segundos	20 segundos	60 segundos	60 segundos

Tabla 1 Precios ThingSpeak

Los precios de los planes son los siguientes:

- Estudiante: 79$ anuales.
- Hogar: 95$ anuales.
- Académico: 250$ anuales.
- Estándar: 650$ anuales.

En caso de querer alojar la plataforma en una nube privada o de terceros, no habría limitaciones más allá del plan de alojamiento contratado.

2.3. Tabla Comparativa Plataformas IoT

Al igual que el término IoT, una plataforma IoT es un concepto muy amplio. Puede tratarse de simples plataformas que sirven para almacenar datos y ofrecen interfaces estándares al usuario, hasta sistemas más completos que permiten el uso de herramientas para hacer predicciones, analíticas o para crear interfaces más complejas.

Plataformas	SDK/ Lenguajes soportados	Protocolos soportados	Ventajas	Desventajas
AWS IoT	C, JavaScript, Java, Python, iOS, Android, C++	MQTT (Incluye soporte para WebSocket) Http	Plataforma líder, con casi infinitas posibilidades mediante sus diferentes servicios. Sus tarifas son muy baratas.	Requiere mucha investigación para conectar los diferentes servicios entre sí. La plataforma cambia constantemente y requiere bastante formación para mantenerse al día.
Azure IoT	C, Python, Node.js, Java, .NET	MQTT AMQP HTTP	Plataforma bastante completa, con multitud de servicios y una arquitectura por capas muy bien definida. Posee de un sistema de interacción con el dispositivo muy completo.	Los mensajes entre servicios no se incluyen dentro de la tarifa base. Los precios se engloban en 4 categorías poco flexibles y hay que tener mucho cuidado con el consumo para

				no acabar en una categoría sobredimensionada
Oracle IoT	Android, C, iOS, Java SE, JavaScript	MQTT HTTP	Facilita la conexión a los dispositivos mediante clientes y puertas de enlace que se encargan de gestionar todo el proceso. Herramientas de análisis de datos muy completas.	Plataforma de reciente creación con un número limitado de servicios en comparación con sus competidores directos. Categorías de precios muy difíciles de entender.
Watson IoT	Node.js, Java, Python, C#, C, C++	HTTP MQTT	Plataforma muy completa a la hora de analizar los datos para machine learning y minería de datos.	Carece de una solución interna al nivel de otras plataformas para la representación de los datos. La cantidad disponible de servicios es más reducida en comparación a sus competidores.
Xively	iOS Android	MQTT WebSocket HTTP	Muy sencilla de gestionar y poner en marcha. Gestión de dispositivos muy	Plataforma que solo provee de los servicios básicos para la recolección, análisis y

			completa, mediante agrupaciones por localización o función.	representación de los datos. No disponen de una tabla de precios con estimaciones, hay que contactar con su departamento de ventas.
Samsung Artik	C C++ Node.js	HTTP WebSockets MQTT CoAP	Placas de desarrollo especificas del fabricante y muy sencillas de conectar a la plataforma para facilitar el desarrollo de soluciones IoT. Muchas integraciones con herramientas de terceros configuradas automáticamente.	Intenta ocultar tanto el proceso de configuración y las opciones, que puede resultar más cerrada a desarrollos con necesidades específicas. Es una plataforma bastante más cara y menos flexible que sus competidores.
Carriots	Groovy	MQTT HTTP	Ofrece la opción de generar la lógica de las aplicaciones básicas mediante una herramienta visual con bloques. Utiliza un lenguaje funcional y moderno como Groovy para el desarrollo de aplicaciones.	Su sistema de reglas y alertas es bastante más simple que el ofrecido por otras soluciones dentro del mercado. No es posible elegir como almacenar la información dentro de la plataforma una vez definido el formato de datos, por

				lo tanto es necesario exportar los datos a una base de datos externa para realizar según qué tipo de operaciones.
Adafruit.IO	Arduino, Ruby, Python, Node.js	MQTT HTTP	Posee de una amplia sección de tutoriales. Provee de la mayoría de los controladores necesarios para los sensores. Los paneles que representan la información gráficamente son muy fáciles de crear.	No es posible exportar la información fuera de la plataforma. No dispone de integraciones con terceros. No es posible configurar alertas o reglas para actuar según los valores recogidos por los sensores.
Ubidots	Python, Java, C, PHP, Node.js, Ruby	MQTT HTTP	Dispone de numerosas librerías, tanto para placas de desarrollo específicas como para lenguajes de programación. Su herramienta de representación gráfica de los datos es muy potente.	No permite la gestión de dispositivos por grupos, ya que cada uno tiene un panel individual. Carece de ofertas flexibles por uso, ya que requiere la negociación de un plan a medida con su equipo de ventas.
myDevices Cayenne	Arduino	MQTT	Plataforma capaz de gestionar directamente	Limitado soporte para placas de

	Raspberry Pi		los conectores de las placas de desarrollo, para evitar el tener que programar la interacción del sistema con los sensores. Documentación muy extensa y con muchos ejemplos.	desarrollo y sensores, ya que tienen que ser compatibles con el software proporcionado por la plataforma. Se basa excesivamente en el uso de su aplicación móvil para la configuración y gestión de los dispositivos IoT.
Macchina.io	JavasScript	MQTT COAP ModBus	La plataforma es de código abierto. Dispone de un sistema de alertas con reglas.	Utiliza una base de datos SQLite muy limitada para la mayoría de escenarios industriales. La herramienta de representación gráfica es demasiado limitada y requiere la implementación de los componentes JavaScript por parte del usuario.
ThingSpeak	MATLAB	MQTT HTTP	La plataforma es de código abierto.	Su sistema de alertas se limita a una integración con Twitter. Para otro

			Gran oferta de integraciones por parte del desarrollador. Permite agrupar dispositivos IoT por función o localización mediante sus canales. Fácil de utilizar.	tipo de alertas es necesario utilizar un servicio de terceros. Su sistema de reglas es poco intuitivo y difícil de configurar. La instalación de la plataforma en una nube privada es complicada y tiende a fallar. Poca documentación.

Tabla 2 Comparativa Plataformas IoT

3. Hardware

Para realizar las pruebas de concepto en las dos plataformas escogidas, el hardware seleccionado ha sido una placa Raspberry Pi 3 junto a un sensor de humedad y temperatura DHT11.

3.1. Placa de desarrollo

La Raspberry Pi 3 es una de las placas de desarrollo más utilizadas por los usuarios domésticos del internet de las cosas.

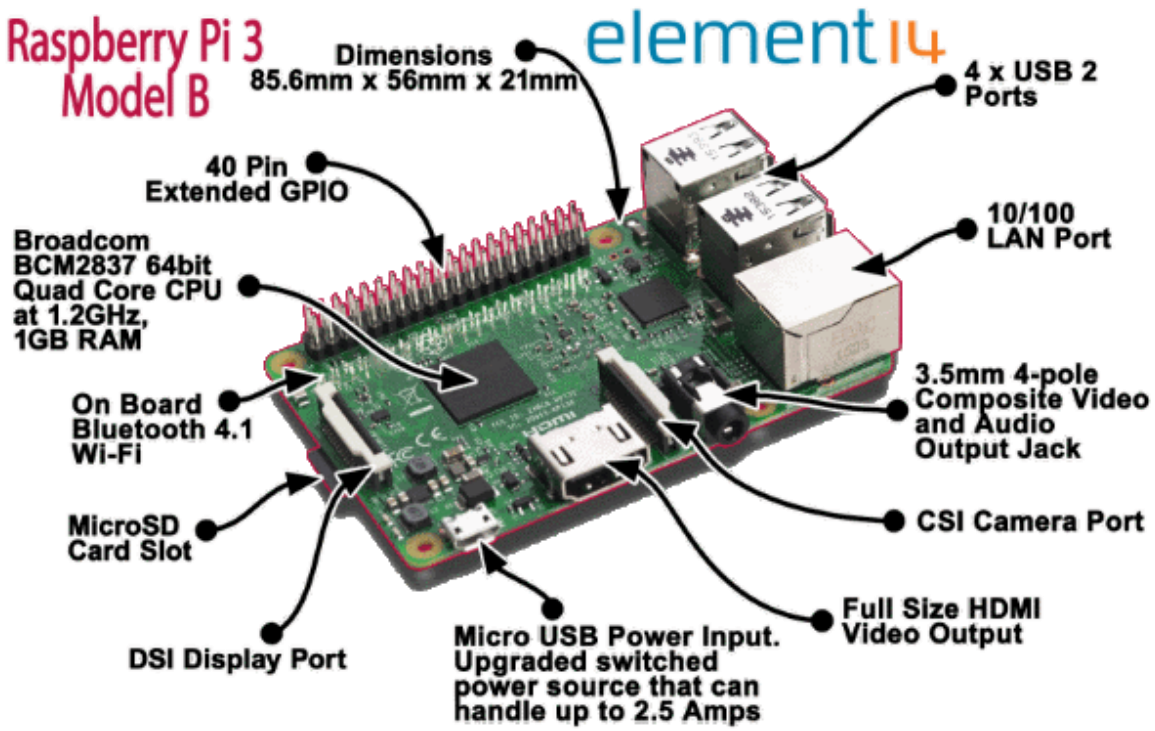


Imagen 25 Raspberry Pi 3 y sus conectores

Sus especificaciones la hacen ideal para el desarrollo de prototipos, ya que dispone de todo tipo de conectividad de serie, y soporta multitud de distribuciones Linux. Además, son muchas las plataformas que incluyen opciones de conexión hechas a medida para esta placa.

Sus especificaciones técnicas son las siguientes:

- Procesador:

  - Chipset Broadcom BCM2387.
  - 1,2 GHz de cuatro núcleos ARM Cortex-A53.
- GPU
  - Dual Core VideoCore IV ® Multimedia Co-procesador. Proporciona Open GL ES 2.0, OpenVG acelerado por hardware, y 1080p30 H.264 de alto perfil de decodificación.
  - Capaz de 1 Gpixel / s, 1.5Gtexel / s o 24 GFLOPs con el filtrado de texturas y la infraestructura DMA.
- RAM: 1GB LPDDR2.
- Conectividad
  - Ethernet socket Ethernet 10/100 BaseT.
  - 802.11 b / g / n LAN inalámbrica y Bluetooth 4.1 (Classic Bluetooth y LE).
  - Salida de vídeo
    - HDMI rev 1.3 y 1.4.
    - RCA compuesto (PAL y NTSC).
  - Salida de audio
    - Jack de 3,5 mm de salida de audio, HDMI.
  - USB 4 x Conector USB 2.0
  - Conector GPIO
    - 40-clavijas de 2,54 mm (100 milésimas de pulgada) de expansión: 2x20 tira.
    - Proporcionar 27 pines GPIO, así como 3,3 V, +5 V y GND líneas de suministro.
  - Conector de la cámara de 15 pines cámara MIPI interfaz en serie (CSI-2).
- Ranura de tarjeta de memoria Empuje / tire Micro SDIO.

Esta placa dispone de un hardware bastante completo, que contempla la ampliación mediante los conectores GPIO.

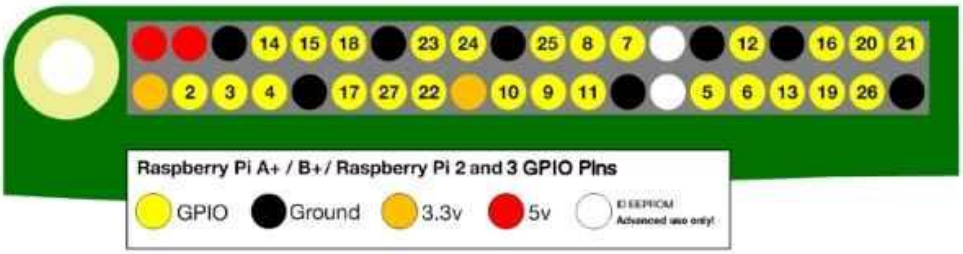


Imagen 26 Conectores GPIO Raspberry

Por último, en caso de querer utilizar una solución más cómoda a los conectores GPIO, la empresa Dexter Industries ha desarrollado una placa de expansión llamada GrovePi. Esta placa permite conectar multitud de sensores de una forma fácil y cómoda.

Imagen 27 Placa GrovePi y sensores compatibles

3.2. Sensores

Para las pruebas se utilizará el sensor DHT11, ya que es el más común de su tipo y los controladores necesarios para recoger datos con él están disponibles en multitud de lenguajes de programación.

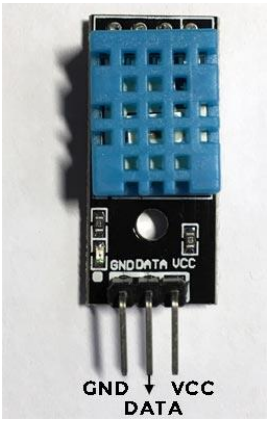


Imagen 28 Sensor DHT11 con resistencia integrada en PCB

Además, para una mayor comodidad, se utilizará una versión con PCB, que ya cuenta con la resistencia necesaria para hacer funcionar el sensor.

Las especificaciones del sensor son las siguientes:

- Alimentación: 3Vdc ≤ Vcc ≤ 5Vdc.
- Rango de medición de temperatura: 0 a 50 °C.
- Precisión de medición de temperatura: ±2.0 °C.
- Resolución Temperatura: 0.1°C.
- Rango de medición de humedad: 20% a 90% RH.
- Precisión de medición de humedad: 4% RH.
- Resolución Humedad: 1% RH.
- Tiempo entre medición: 1 seg.

Es un sensor básico, con unos márgenes de error bastante correctos. En caso de buscar mayor precisión, se puede utilizar el modelo superior, DHT22.

3.3. Configuración

El primer paso para configurar el dispositivo es la instalación del sistema operativo. El sistema operativo de la Raspberry Pi 3 ha de ser instalado en una memoria microSD.

Para facilitar la instalación del sistema operativo, los fabricantes de la placa proporcionan un instalador de sistemas operativos al que han llamado Noobs. Este instalador contiene Raspbian de forma local, y un sinfín de sistemas operativos compatibles de forma remota, los cuales hay que descargar.

Para las pruebas de concepto se utilizará Raspbian, ya que es la mejor distribución para instalar el software y los controladores necesarios.

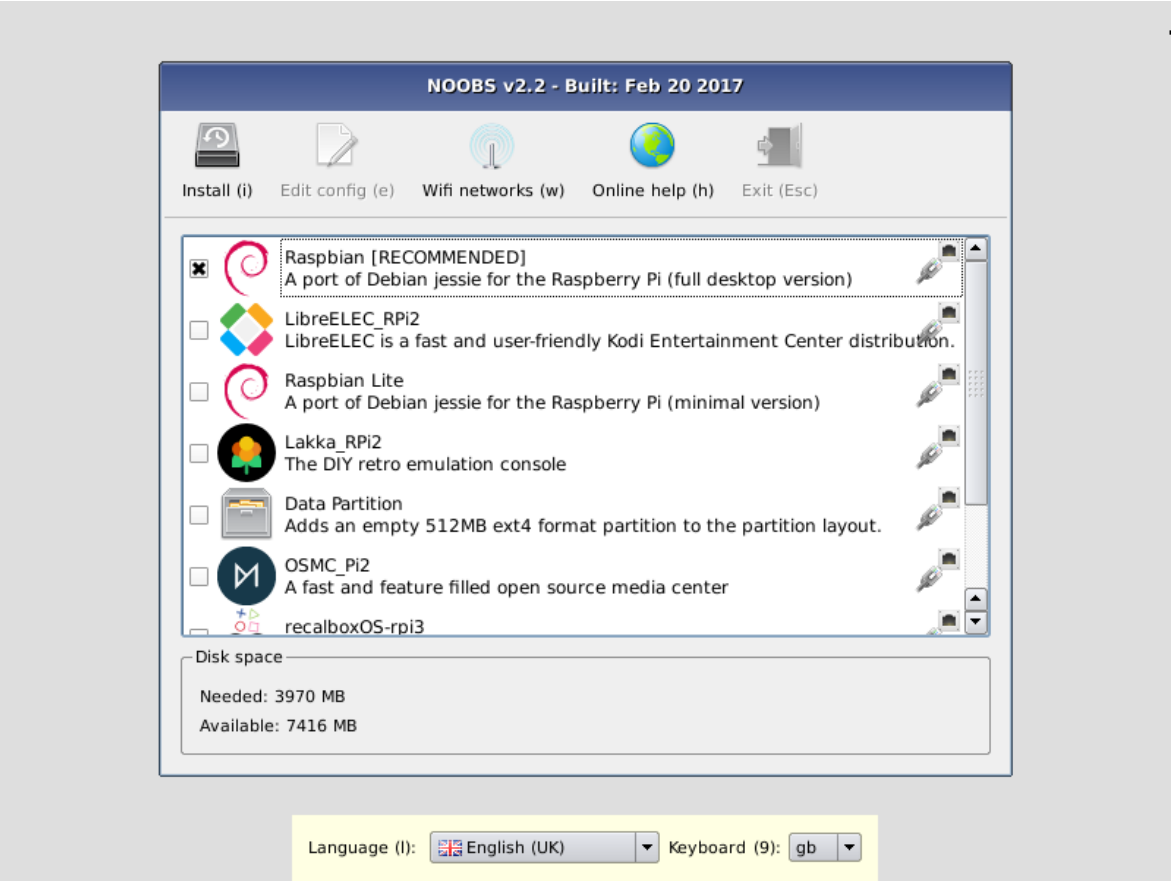


Imagen 29 Distribuciones disponibles mediante NOOBS

Una vez instalado el sistema operativo, el primer paso será acceder a la configuración de la Raspberry Pi y habilitar la comunicación remota (SSH, VNC) y las lecturas desde lectores de tipo I2C y SP:

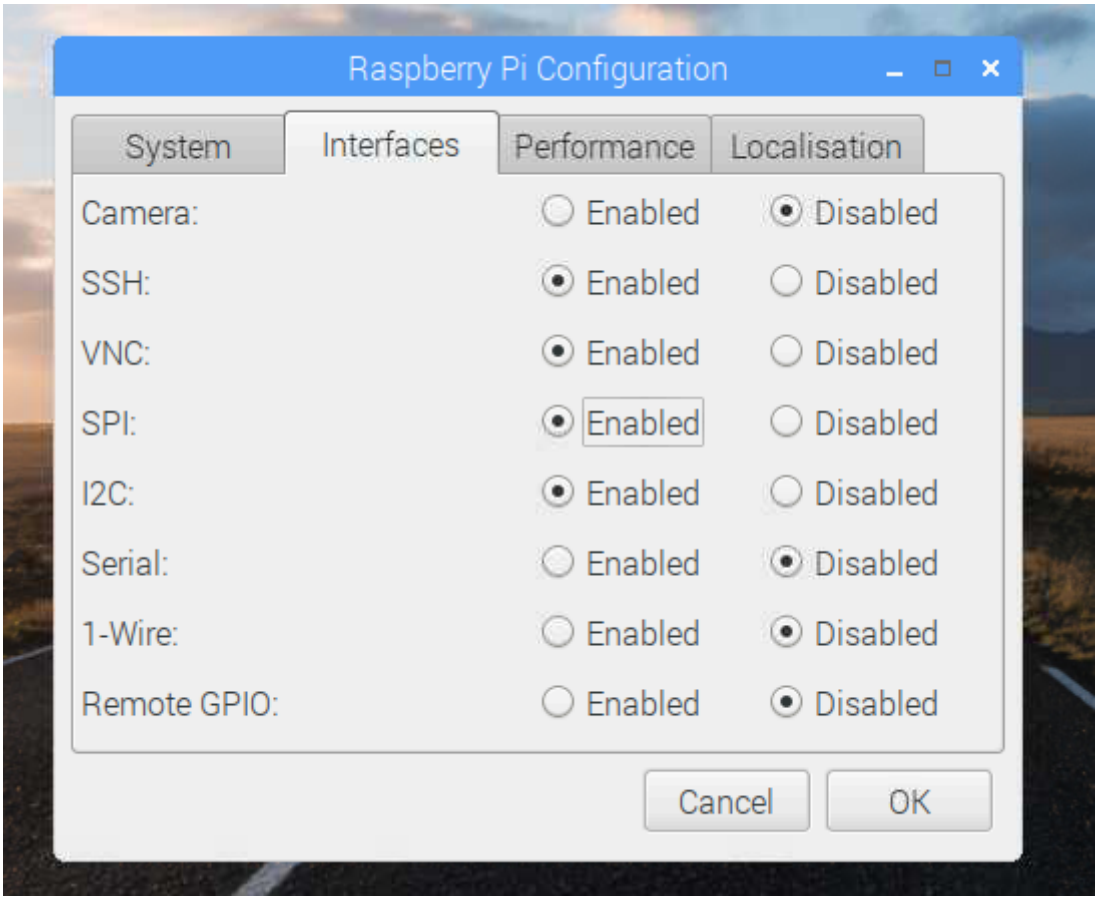


Imagen 30 Configuración Interfaces Raspberry

Además, se instalarán mediante la consola de comandos el servidor ftp para subir los certificados a la Raspberry de forma remota y el cliente MQTT PAHO de Eclipse:

```
sudo apt-get install ftp
pip install paho-mqtt
```

Por último, será necesario clonar e instalar el controlador de Adafruit para el sensor DHT11. Para ello, antes hay que comprobar que el sistema está actualizado y dispone del software necesario para la instalación de este controlador mediante los siguientes comandos:

```
sudo apt-get update
sudo apt-get install build-essential python-dev
```

Lo siguiente es clonar el repositorio de Github donde se encuentran los binarios y ejecutar el script de instalación:

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT/
sudo python setup.py install
```

Con esto, la instalación y configuración del dispositivo ya está completa, lo que resta es conectar el sensor DHT11 a la Raspberry. Para ello se utilizarán 3 conectores GPIO.

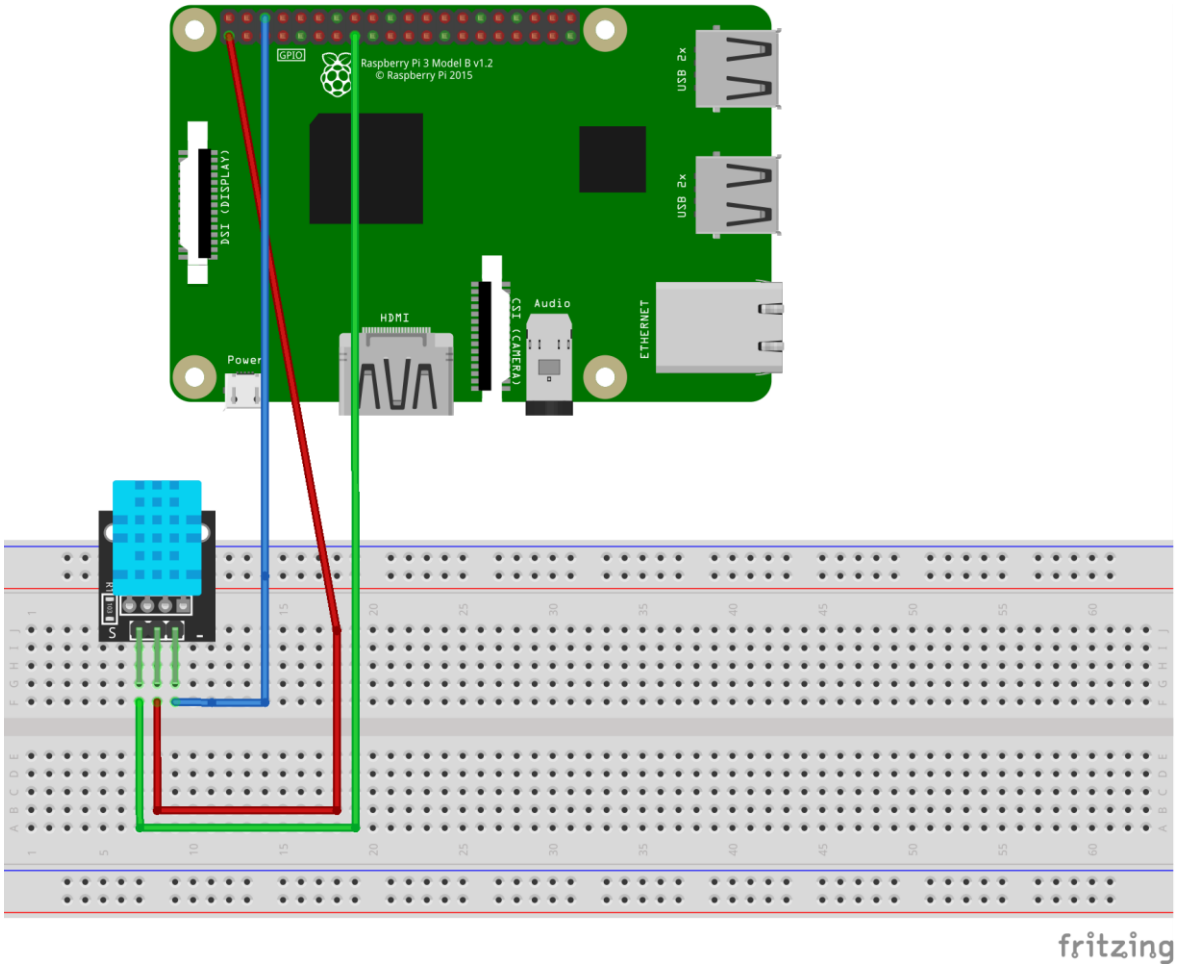


Imagen 31 Diagrama de la conexión de las Raspberry con el sensor DHT11

El cable rojo del diagrama es el que proporciona energía al sensor. En este caso, se proporcionan 3.3V.

El cable azul del diagrama está conectado al GPIO de masa/tierra y se conecta a la pata derecha del sensor, la cual es el negativo.

Por último, el cable verde está conectado al GPIO 22 que recoge los datos, el pin digital.

4. Plataformas escogidas

4.1. Amazon Web Service IoT

Esta plataforma ha sido escogida por ser la que ofrece más servicios y opciones de integración de todas las analizadas. Su punto fuerte está en la ingente cantidad de interconexiones entre los diferentes productos de la plataforma, de forma que los datos pueden ser almacenados, analizados y transformados en múltiples formas.

4.1.1. Panel AWS IoT

Una vez creada la cuenta de Amazon Web Services, la primera pantalla a la que se podrá acceder es a la consola de administración.

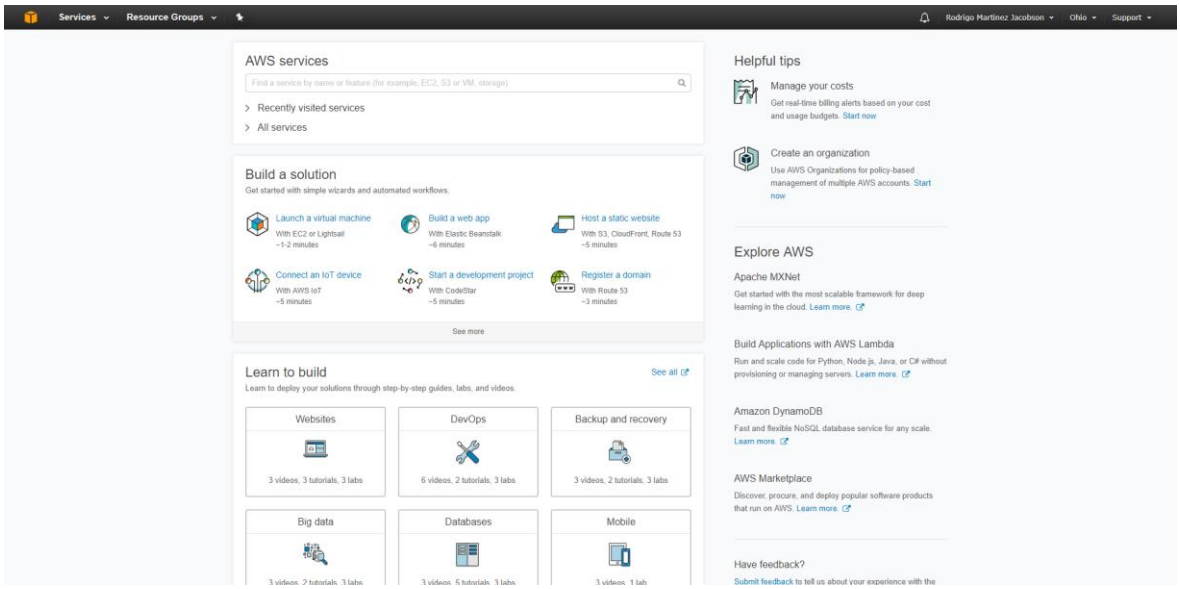


Imagen 32 Consola de Administración de AWS

En esta página se muestran diferentes opciones, desde buscar entre los diferentes servicios que están englobados dentro de AWS a construir una solución o seguir un tutorial paso a paso para aprender a utilizar una herramienta en concreto.

Utilizando la barra de búsqueda, se accederá al panel AWS IoT:

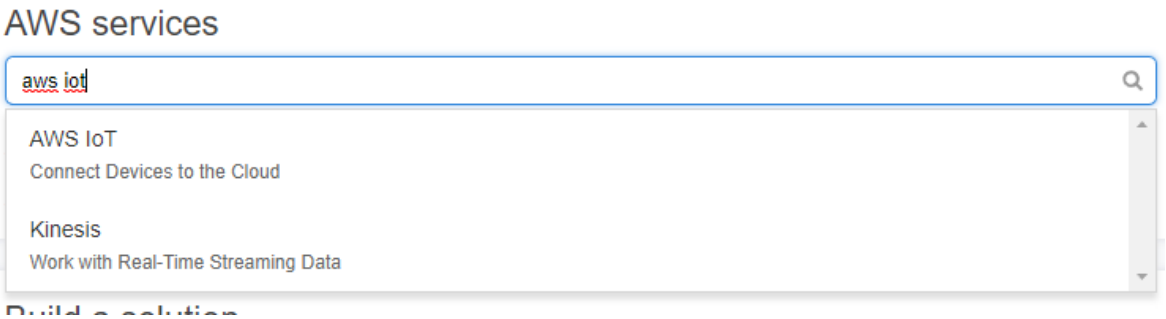


Imagen 33 Buscador de servicios de AWS

El apartado que muestra por defecto AWS IoT al entrar es un panel lleno de gráficos sobre peticiones recibidas o enviadas, de qué tipo y el protocolo utilizado. Además, en caso de tener reglas configuradas, también muestra gráficos sobre cuántas reglas han sido lanzadas.

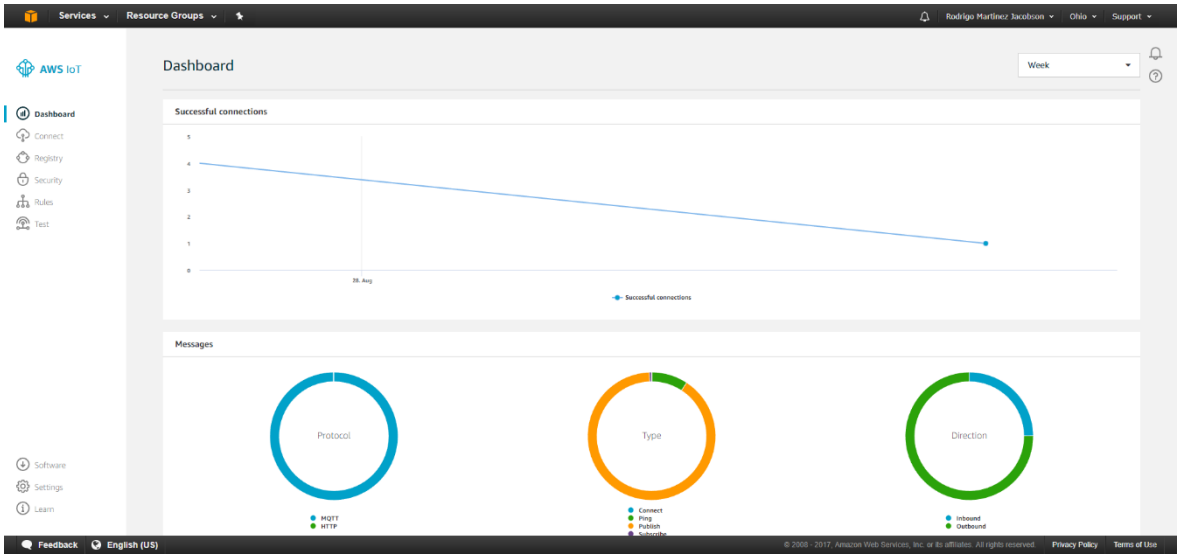


Imagen 34 Panel principal de AWS IoT

En el menú lateral se encuentran los diferentes apartados en los que se divide el servicio:

- Dashboard: Panel inicial donde se muestran estadísticas de uso.
- Connect: Asistentes para configurar paso a paso un dispositivo y escoger el SDK más apropiado para la conexión entre este y la plataforma.
- Registry: Apartado para conectar un dispositivo a la plataforma. A diferencia del apartado anterior, aquí se realiza una configuración rápida y totalmente gestionada por el usuario. Además, en este apartado se pueden configurar tipos para agrupar los dispositivos.

- Security: Apartado para crear y gestionar los certificados y políticas que permiten a los dispositivos interactuar con la plataforma, enviando y recibiendo datos.
- Rules: Permite configurar reglas, las cuales se utilizan para responder a cierto filtro sobre la información recibida, transformarla y enviar el resultado a otro servicio de la plataforma.
- Test: Apartado que permite simular un dispositivo capaz de enviar todo tipo de mensajes a la plataforma para probar las reglas y los servicios asociados a estas.

4.1.2. Configuración del dispositivo y los certificados

Para iniciar la configuración del dispositivo, se puede empezar por el apartado Connect en caso de tener dudas sobre el SDK a elegir y el sistema operativo del dispositivo. En caso contrario, se puede acceder a la configuración manual por el apartado Registry.

En el caso de esta prueba de concepto, se utilizará el apartado Registry, ya que tanto el sistema operativo del dispositivo como el lenguaje de programación ya han sido escogidos previamente.

Se le asignará un nombre al dispositivo, y no se realizará ninguna configuración opcional.

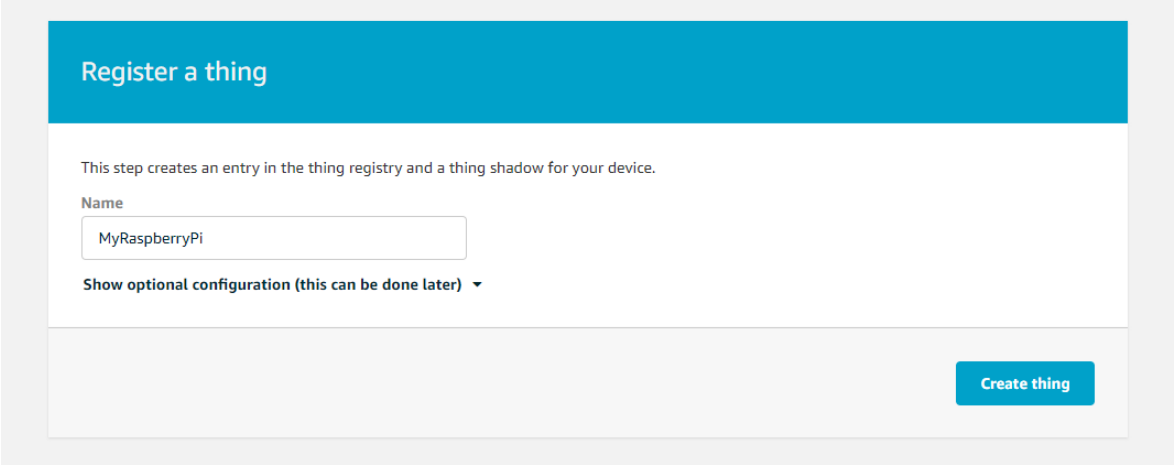


Imagen 35 Registro de dispositivo en AWS IoT

Una vez creado el dispositivo, en la próxima página, la plataforma indicará la dirección con la que se identifica el dispositivo dentro de esta:

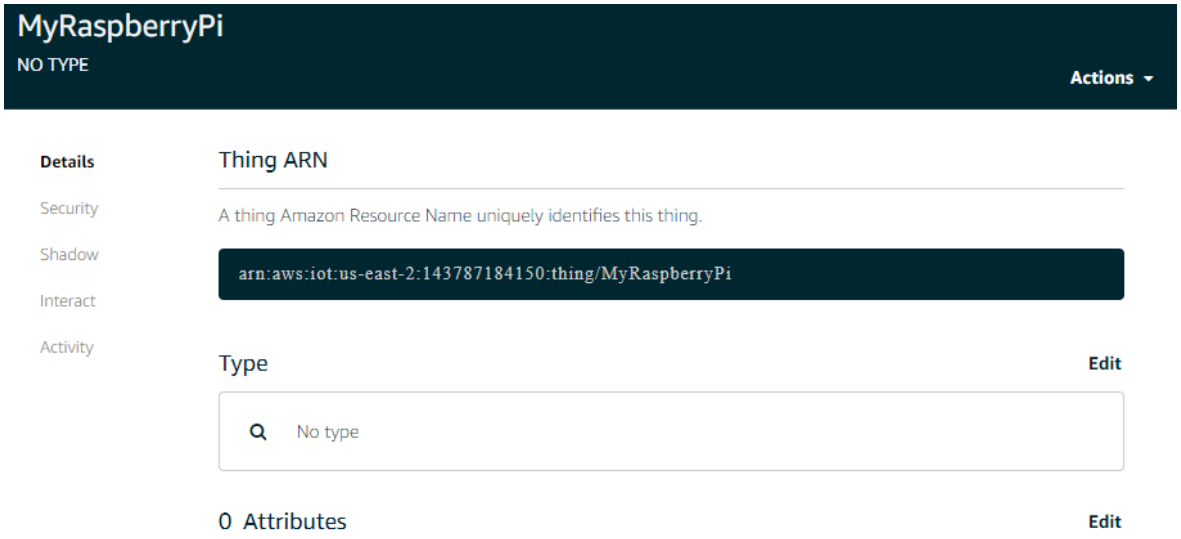


Imagen 36 AWS IoT: Detalles del dispositivo

Dentro de esta página, seleccionando el apartado Interact, se indicará la dirección a la que el dispositivo habrá de conectarse para enviar y recibir información. Además, se mostrará un listado de los endpoints disponibles para interactuar con la plataforma y la información enviada por el dispositivo.

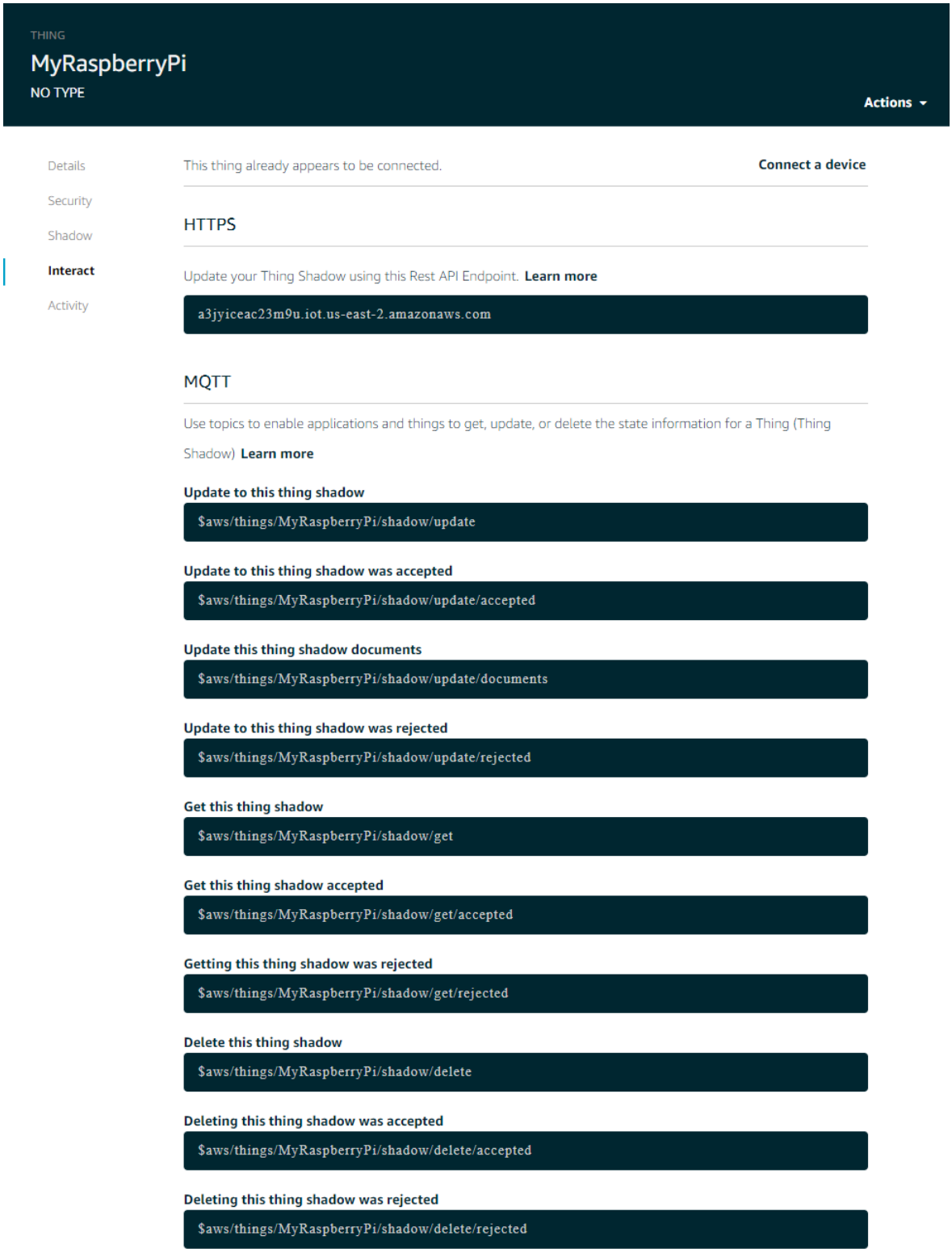


Imagen 37 URIs para consumir recursos de un dispositivo en AWS IoT

Otro dato interesante de la plataforma, es que el denominado sistema shadow, se encarga de responder automáticamente con el último estado proporcionado por el dispositivo, aunque este no esté disponible en el momento de la petición.

El último apartado donde es necesario realizar alguna configuración es en Security. En este apartado, seleccionando Create Certificate, se crearán los certificados necesarios para conectar con la plataforma asegurando que la identidad del dispositivo es correcta.

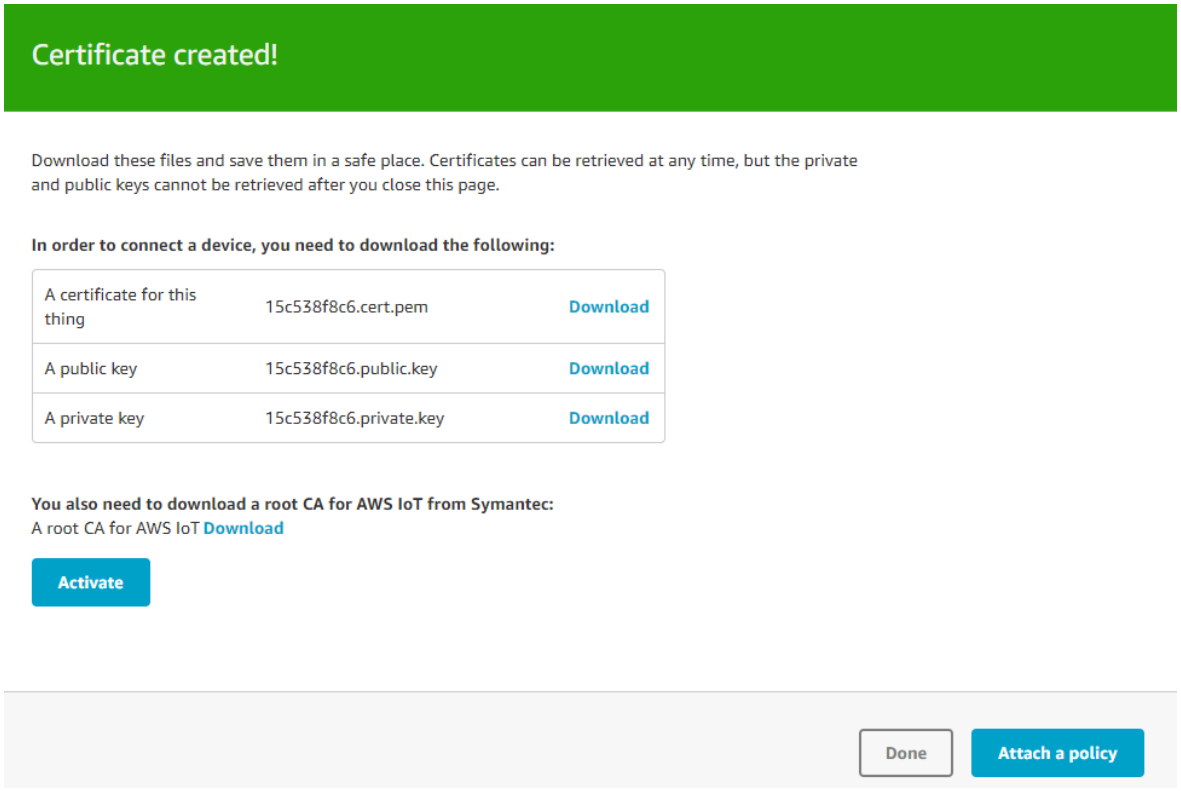


Imagen 38 Creación de certificados en AWS IoT

Es muy importante descargar los certificados en este paso y almacenarlos en el dispositivo en cuestión, ya que son imprescindibles para interactuar con la plataforma.

Además, también será necesario descargar el certificado raíz, que identifica al emisor de los certificados y que también será necesario para conectar con la plataforma.

Una vez generados los certificados, solo falta asignar una política de seguridad al certificado mediante el botón “Attach policy”. Dentro de este apartado, es necesario crear una nueva política de seguridad con la siguiente configuración:

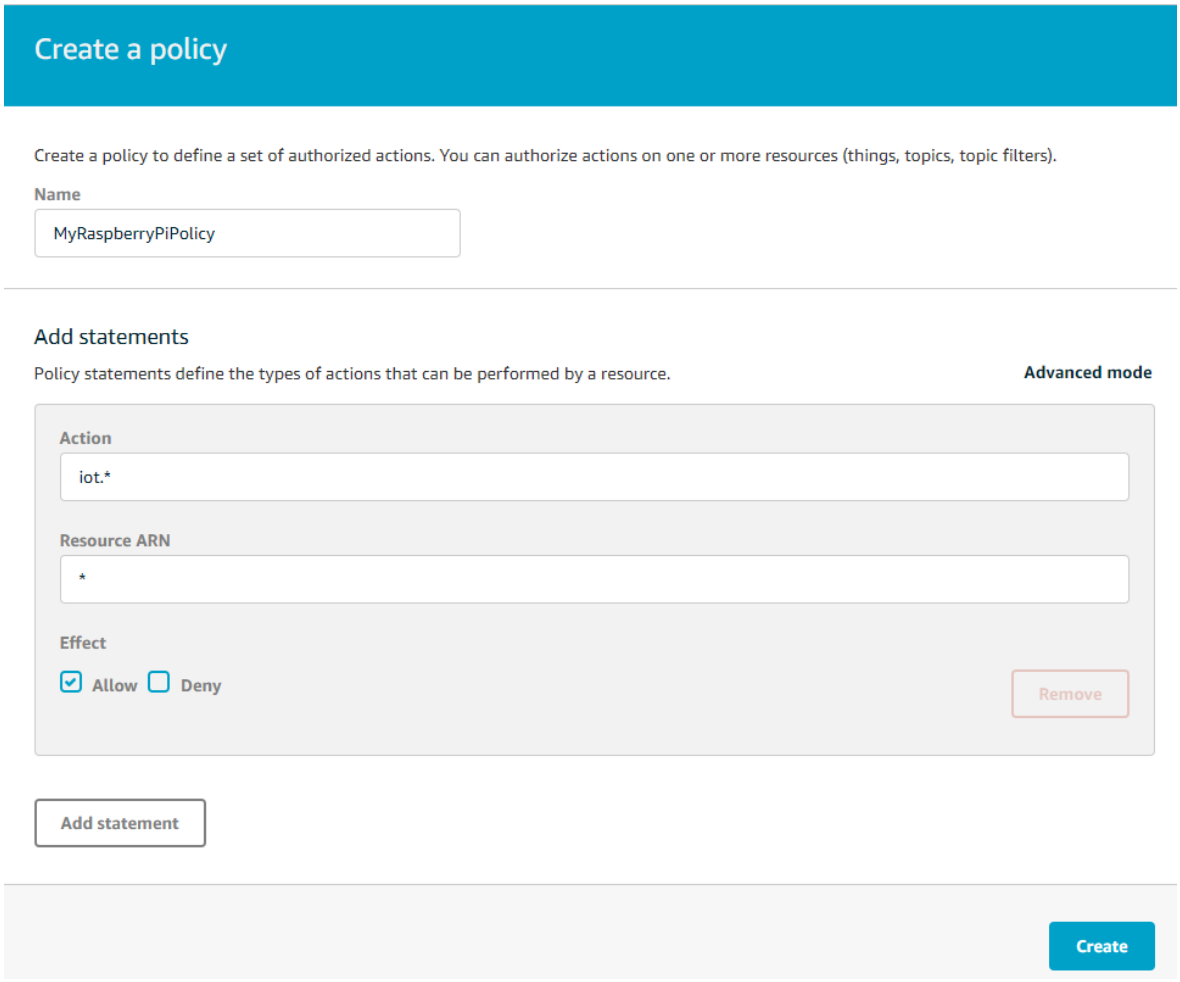


Imagen 39 Creación de una política de seguridad en AWS IoT

Esta política permitirá al dispositivo publicar y recibir mensajes.

Una vez creada la política de seguridad, la plataforma llevará automáticamente al usuario al panel principal. En este, hay que dirigirse al apartado de seguridad y al sub-apartado de Certificados.

Seleccionando el certificado creado anteriormente, y desplegando el menú contextual, se seleccionará la opción “Attach Policy” y aparecerá la siguiente ventana:

Imagen 40 Adjuntar una política a un certificado en AWS IoT

Seleccionando la política de seguridad creada previamente y haciendo clic en “Attach”, el certificado ya estará configurado para ser utilizado.

El último paso es indicar al certificado cuál es el dispositivo que lo va a utilizar. Para ello, sin salir del panel Certificates y desplegando el menú contextual del certificado, se seleccionará la opción “Attach thing” y se seleccionará el dispositivo:

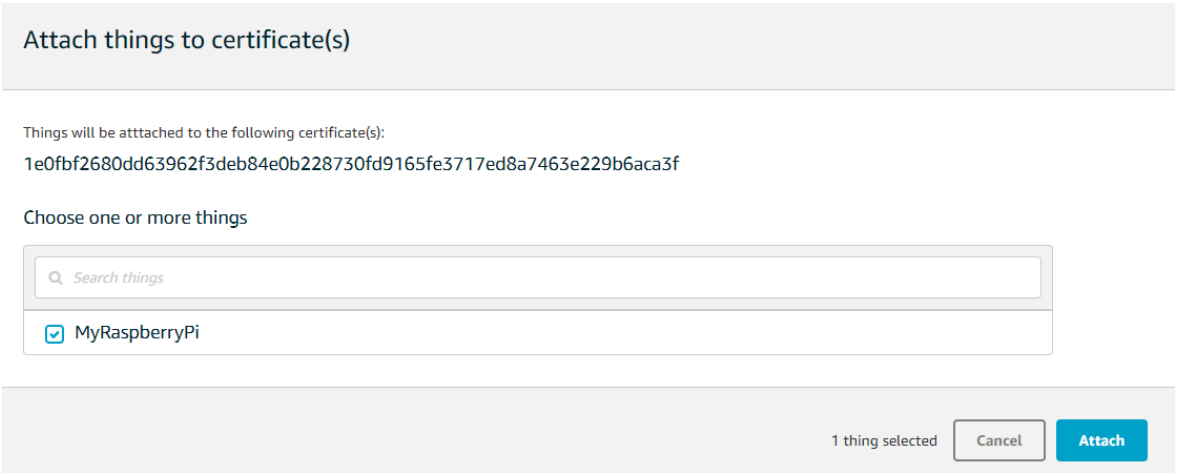


Imagen 41 Adjuntar dispositivo a una política en AWS IoT

De esta forma, el dispositivo ya dispone de unos certificados válidos para conectar con la plataforma.

4.1.3. Envío de datos a la plataforma

En este apartado se explicará cómo proceder para enviar datos a la plataforma.

El primer paso consiste en crear un script en Python para poder leer la temperatura y la humedad, y para enviar esos datos a Amazon Web Service. Este script utilizará el controlador proporcionado por Adafruit para el sensor DHT11 y el cliente MQTT Paho de Eclipse.

Para la lectura de los datos del sensor, se utilizará el script proporcionado por Adafruit dentro del paquete del controlador a modo de ejemplo. Este script con nombre "simpletest.py", localizado dentro de la carpeta examples, proporciona todo lo necesario para leer y mostrar por pantalla los valores de humedad y temperatura.

```
import Adafruit_DHT

# Sensor should be set to Adafruit_DHT.DHT11,
# Adafruit_DHT.DHT22, or Adafruit_DHT.AM2302.
sensor = Adafruit_DHT.DHT11

# Example using a Beaglebone Black with DHT sensor
# connected to pin P8_11.
#pin = 'P8_11'

# Example using a Raspberry Pi with DHT sensor
# connected to GPIO22.
pin = 22

# Try to grab a sensor reading.  Use the read_retry method which will retry up
# to 15 times to get a sensor reading (waiting 2 seconds between each retry).
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

# Note that sometimes you won't get a reading and
# the results will be null (because Linux can't
# guarantee the timing of calls to read the sensor).
# If this happens try again!
if humidity is not None and temperature is not None:
    print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
else:
    print('Failed to get reading. Try again!')
```

Imagen 42 Código para la lectura de datos del sensor

El script es compatible tanto con BeagleBoard como con Raspberry Pi. Para esta prueba de concepto, se ha comentado la línea que indica el pin en la placa BeagleBoard y se ha indicado que el pin utilizado por el sensor es el GPIO22 de la Raspberry Pi.

Una vez actualizado el script de ejemplo, con una simple ejecución mostrará los valores de humedad y temperatura de la siguiente forma:

```
pi@raspberrypi:~/Adafruit_Python_DHT/examples $ python simpletest.py
Temp=28.0*C  Humidity=29.0%
```

Imagen 43 Resultado de la ejecución del script para leer datos del sensor

El siguiente paso es hacer que la medición del sensor se repita tras un intervalo fijo de tiempo y se envíe a la plataforma de Amazon.

Para ello, se modificará el script, haciendo uso del cliente MQTT desarrollado por eclipse, y se generará una conexión segura con Amazon mediante los certificados.

Primero, es necesario importar unas cuantas librerías más, necesarias para establecer conexión con la plataforma, dar forma al mensaje y enviarlo. Además, se crearán variables para indicar los nombres de los ficheros de los certificados necesarios para establecer la conexión.

```
#!/usr/bin/env python

# Send sensor data continually to AWS IoT.

import time
import Adafruit_DHT
import datetime
import ssl
import json
import paho.mqtt.client as mqtt

# Name of our Raspberry Pi, also known as our "Thing Name"
deviceName = "MyRaspberryPi"
# Public certificate of our Raspberry Pi, as provided by AWS IoT.
deviceCertificate = "1e0fbf2680-certificate.pem.crt"
# Private key of our Raspberry Pi, as provided by AWS IoT.
devicePrivateKey = "1e0fbf2680-private.pem.key"
# Root certificate to authenticate AWS IoT when we connect to their server.
awsCert = "ca.pem.crt"

# Sensor should be set to Adafruit_DHT.DHT11,
# Adafruit_DHT.DHT22, or Adafruit_DHT.AM2302.
sensor = Adafruit_DHT.DHT11

# Example using a Beaglebone Black with DHT sensor
# connected to pin P8_11.
#pin = 'P8_11'

# Example using a Raspberry Pi with DHT sensor
# connected to GPIO22.
pin = 22
```

Imagen 44 Primera parte del script para enviar datos a AWS IoT

Ahora, utilizando la librería del cliente MQTT, se crearán los métodos necesarios para interactuar con la plataforma y mostrar la respuesta por pantalla. Además, se establecerá la conexión a la dirección asignada al dispositivo desde la plataforma.

Como se puede observar, se utilizan los tres certificados, además de la librería SSL para realizar la conexión.

```
isConnected = False


# This is called when we are connected to AWS IoT via MQTT.
def on_connect(client2, userdata, flags, rc):
    print("Connected to AWS IoT...")
    # Subscribe to our MQTT topic so that we will receive notifications of updates.
    client2.subscribe("$aws/things/" + deviceName + "/shadow/update")
    global isConnected
    isConnected = True


# This is called when we receive a subscription notification from AWS IoT.
def on_message(client2, userdata, msg):
    print(msg.topic + " " + str(msg.payload))


# Print out log messages for tracing.
def on_log(client2, userdata, level, buf):
    print("Log: " + buf)


# Create an MQTT client for connecting to AWS IoT via MQTT.
client = mqtt.Client("awsiot")
client.on_connect = on_connect
client.on_message = on_message
client.on_log = on_log

# Set the certificates and private key for connecting to AWS IoT.
client.tls_set(awsCert, deviceCertificate, devicePrivateKey, ssl.CERT_REQUIRED, ssl.PROTOCOL_TLSv1_2)

# Connect to AWS IoT server.
print("Connecting to AWS IoT...")
client.connect("a3jyiceac23m9u.iot.us-east-2.amazonaws.com", 8883, 60)

# Start a background thread to process the MQTT network commands concurrently, including auto-reconnection.
client.loop_start()
```

Imagen 45 Conexión con AWS IoT desde el script

Por último, se leerán los datos del sensor como en el ejemplo de Adafruit y se generará un JSON con el formato apropiado para que la plataforma lo acepte. Una vez hecho esto, se podrá enviar el mensaje con los datos, indicando que es una actualización del estado del dispositivo.

```
# Loop forever.
while True:
    try:
        # If we are not connected yet to AWS IoT, wait 1 second and try again.
        if not isConnected:
            time.sleep(1)
            continue
        # Try to grab a sensor reading.  Use the read_retry method which will retry up
        # to 15 times to get a sensor reading (waiting 2 seconds between each retry).
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

        # Sometimes Linux can't guarantee the timing of calls to read the sensor).
        # If this happens try again!
        if humidity is not None and temperature is not None:
            print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
            # Read sensor values. Prepare our sensor data in JSON format.
            payload = json.dumps({
                "state": {
                    "reported": {
                        "temperature": temperature,
                        "humidity": humidity,
                        "timestamp": datetime.datetime.now().isoformat()
                    }
                }
            })
            print("Sending sensor data to AWS IoT: ", payload)

            # Publish our sensor data to AWS IoT via the MQTT topic, also known as updating our "Thing Shadow".
            client.publish("$aws/things/" + deviceName + "/shadow/update", payload)
            print("Sent to AWS IoT")

            # Wait 10 seconds before sending the next set of sensor data.
            time.sleep(10)
        else:
            print('Failed to get reading. Try again!')

    except KeyboardInterrupt:
        break
    except IOError:
        print("Error")
```

Imagen 46 Lectura y envío de la información a AWS IoT

El formato del mensaje enviado, incluye los datos de temperatura, humedad, y la marca temporal de la lectura de datos.

Cuando el script se ejecuta, la salida por consola es la siguiente:

```
pi@raspberrypi:~/certs $ python uploadDHT11.py
Connecting to AWS IoT...
Log: Sending CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=awsiot
Log: Received CONNACK (0, 0)
Connected to AWS IoT...
Log: Received SUBACK
Temp=26.0*C  Humidity=30.0%
('Sending sensor data to AWS IoT: ', '{"state": {"reported": {"timestamp": "2017
-09-05T21:49:59.969996", "temperature": 26.0, "humidity": 30.0}}}')
Log: Sending PUBLISH (d0, q0, r0, m2), '$aws/things/MyRaspberryPi/shadow/update'
, ... (107 bytes)
Sent to AWS IoT
Log: Received PUBLISH (d0, q0, r0, m0), '$aws/things/MyRaspberryPi/shadow/update
', ...  (107 bytes)
$aws/things/MyRaspberryPi/shadow/update {"state": {"reported": {"timestamp": "20
17-09-05T21:49:59.969996", "temperature": 26.0, "humidity": 30.0}}}
```

Imagen 47 Resultado de la ejecución del script

Estas líneas indican que el mensaje fue aceptado por la plataforma.

En cuanto al resultado en el panel de AWS IoT, se verá reflejado la recepción del mensaje de la siguiente forma:

Imagen 48 Prueba de que la plataforma recibe los mensajes

4.1.3. Almacenamiento de datos en DynamoDB

El objetivo de este apartado es almacenar la información enviada por el sensor en una base de datos Clave – Valor proporcionada por Amazon en su plataforma, en este caso, DynamoDB.

La idea es almacenar el cuerpo del mensaje, el cual contiene los valores de temperatura, humedad y marca temporal.

Lo primero es configurar una regla, en el panel de AWS IoT. Para ello, en el apartado rules, se creará una regla nueva con la siguiente configuración:

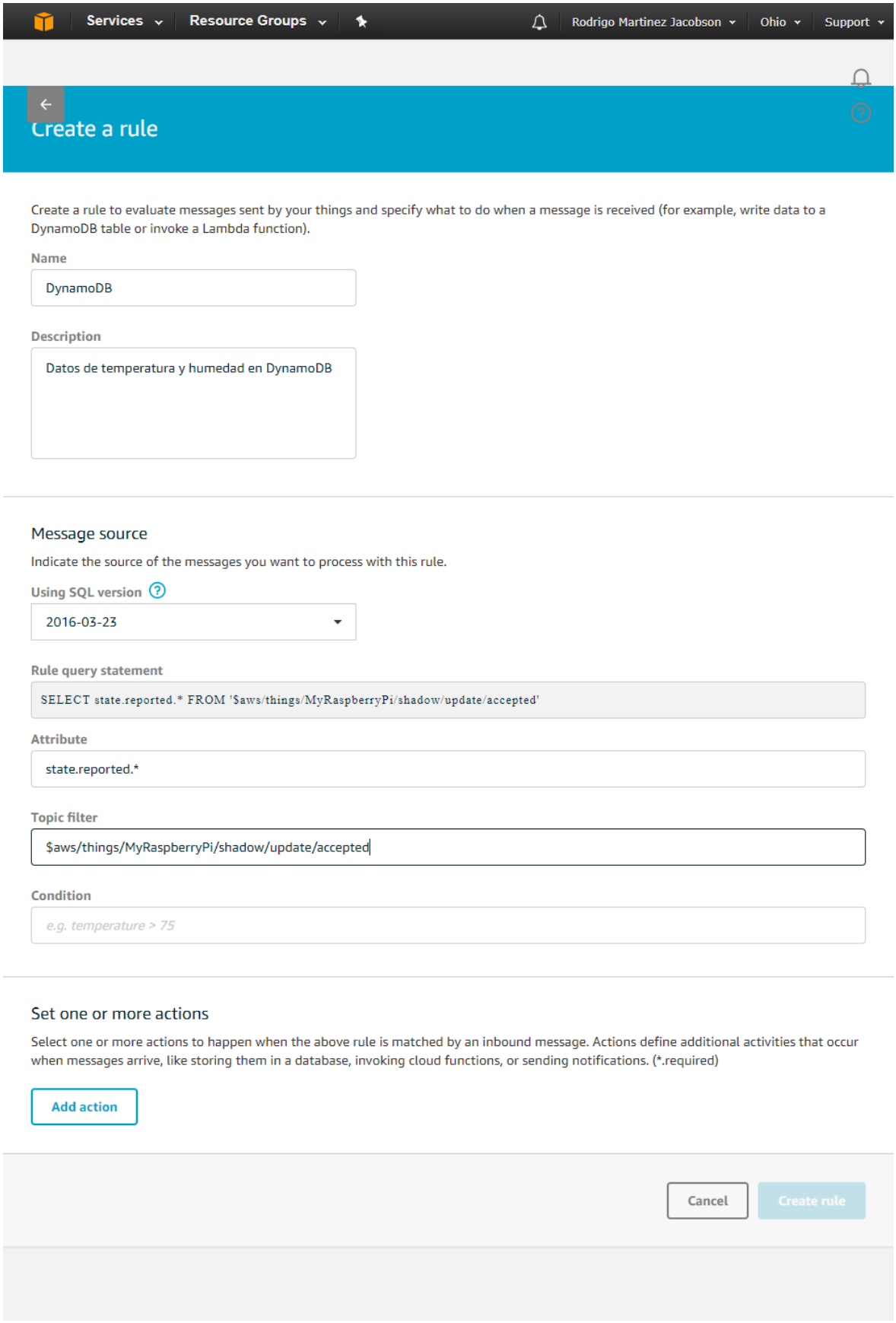


Imagen 49 Creación de una regla en AWS IoT

Los primeros apartados son simplemente para dar un nombre y una descripción a la regla, para facilitar el reconocer que hace con un simple vistazo.

La segunda parte de la configuración consiste en indicar el origen del mensaje. Esto es el dispositivo y/o tema por el que deseamos filtrar y el atributo que deseamos recoger. En este caso, se filtra por el dispositivo MyRaspberryPi y el tema es el último mensaje enviado por el dispositivo y aceptado por la plataforma.

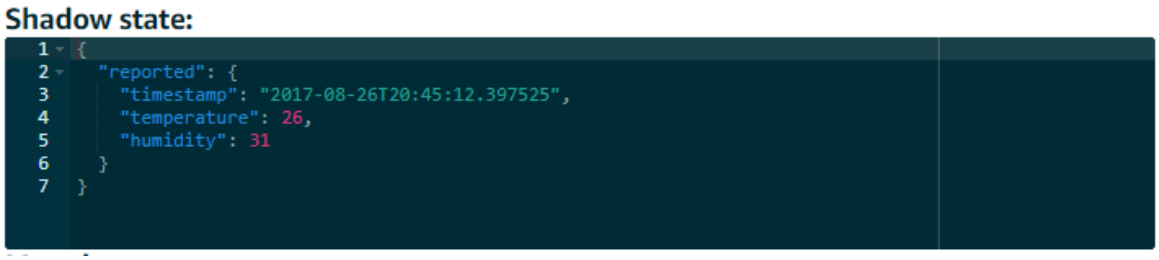


Imagen 50 Ultimo mensaje del dispositivo recibido por la plataforma

Para seleccionar el cuerpo del mensaje como atributo, es necesario seleccionar todo el contenido del objeto reported, dentro del objeto state que devuelve la plataforma.

El siguiente paso en la creación de la regla es seleccionar la acción que se debe tomar. En este caso, la opción seleccionada será la de insertar un mensaje en una tabla de DynamoDB.

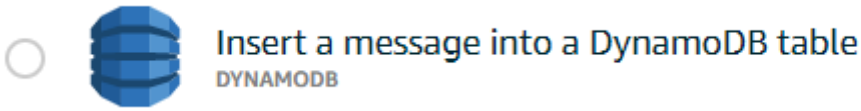


Imagen 51 Opción DynamoDB como acción

Al seleccionar esa acción y asignarla, la plataforma mostrará otra pantalla de configuración donde hay que seleccionar la tabla donde se quieren enviar los datos, y la clave con la que se han de emparejar.

Lo primero es crear la tabla, indicando como clave de partición en este caso será la marca temporal:

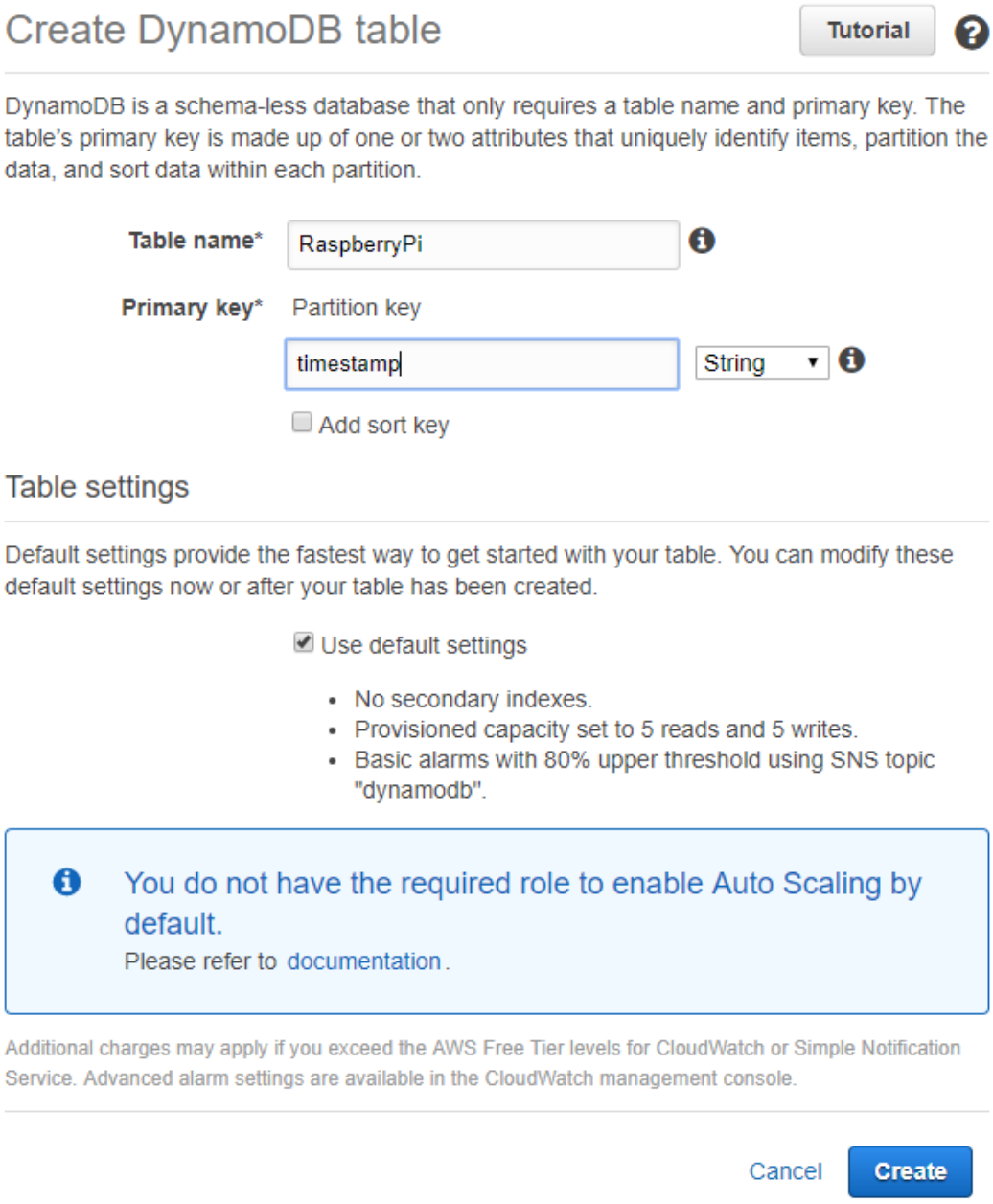


Imagen 52 Creación de una tabla en DynamoDB

Esta clave de partición hará las funciones de clave primaria y, en caso de escalar la base de datos en más de un equipo, ayudará a repartir los datos entre las diferentes estancias de la base de datos.

Una vez configurada la tabla, hay que indicar cómo debe guardar los datos en la regla. Para ello, se ha de indicar la parte del mensaje que será la clave principal, su formato y su valor. Para el valor, con indicarlo mediante la sintaxis de una variable es suficiente para que la plataforma lo asigne automáticamente.

La configuración de la acción quedaría de la siguiente forma:

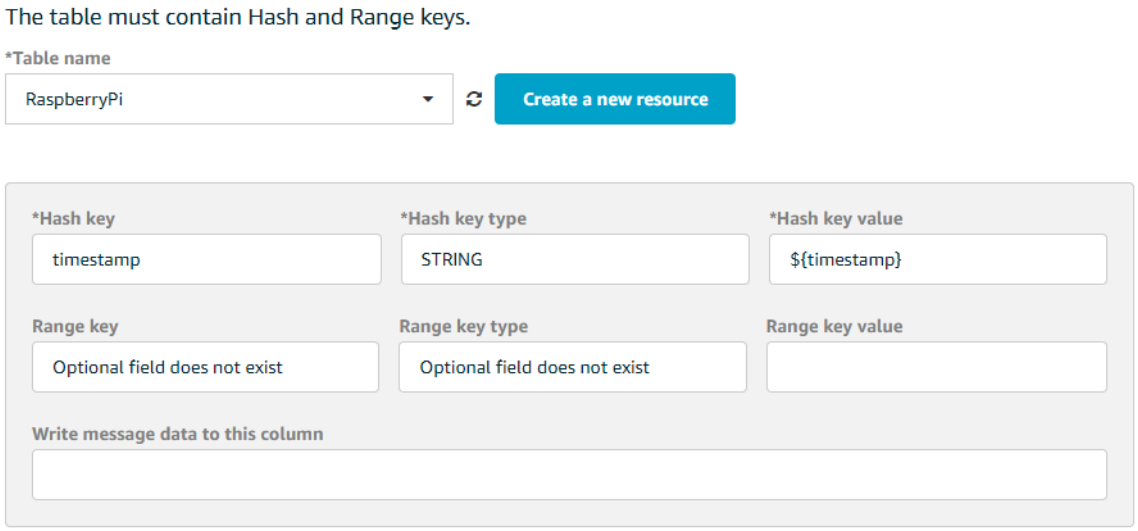


Imagen 53 Datos a guardar en DynamoDB

Y de esta forma, cuando la plataforma envíe un mensaje, será recogida por la tabla. Los mensajes almacenados en la tabla se pueden inspeccionar desde el visor de objetos desde el apartado de DynamoDB y pueden ser consultados por otras aplicaciones configurando la exposición al exterior de la plataforma.

timestamp	payload
1503768258	{ "humidity" : { "N" : "30" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T17:24:18.338176" }}
1503770816	{ "humidity" : { "N" : "30" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T18:06:56.708192" }}
1503765589	{ "humidity" : { "N" : "29" }, "temperature" : { "N" : "27" }, "timestamp" : { "S" : "2017-08-26T16:39:49.842364" }}
1503768542	{ "humidity" : { "N" : "30" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T17:29:02.410753" }}
1503776167	{ "humidity" : { "N" : "30" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T19:36:07.318758" }}
1503768563	{ "humidity" : { "N" : "30" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T17:29:23.485821" }}
1503769985	{ "humidity" : { "N" : "31" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T17:53:05.146134" }}
1503779541	{ "humidity" : { "N" : "30" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T20:32:21.016293" }}
1503774280	{ "humidity" : { "N" : "30" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T19:04:40.082248" }}
1503768616	{ "humidity" : { "N" : "31" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T17:30:16.173072" }}
1503770424	{ "humidity" : { "N" : "30" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T18:00:24.724511" }}
1503769116	{ "humidity" : { "N" : "30" }, "temperature" : { "N" : "26" }, "timestamp" : { "S" : "2017-08-26T17:38:36.483809" }}

Imagen 54 Visualización de los datos guardados en DynamoDB

4.1.4. Alertas por mensajería mediante SNS

Amazon ofrece una opción para enviar notificaciones mediante diversos canales (SMS, Email, notificaciones Push, Twitter, etc) a través de su servicio Simple Notification Service (SNS).

En el caso de AWS IoT, esto permite combinar ambos servicios y generar alertas según los valores recogidos por la Rasbperry Pi.

Para configurar la regla, la configuración es bastante parecida al apartado de DynamoDB, solo que necesita una condición para decidir cuándo es interesante notificar al usuario.

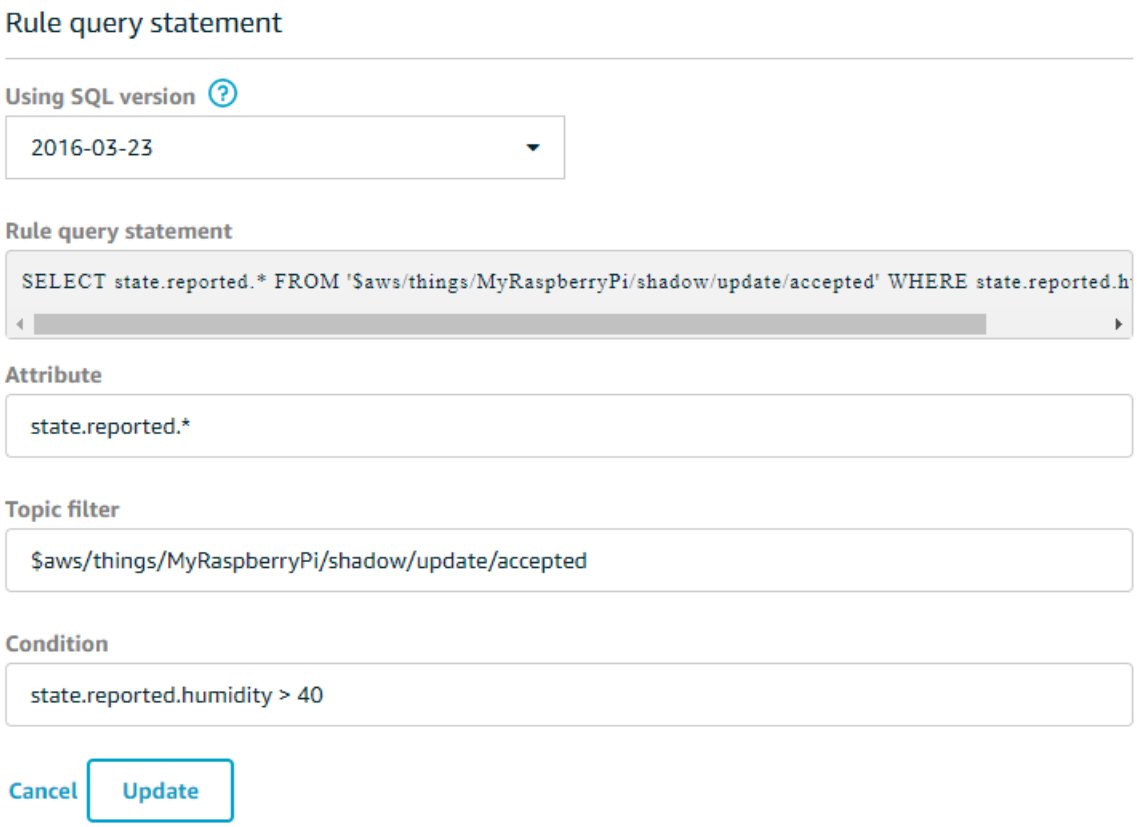


Imagen 55 Configuración de una regla con condición

En este caso, interesa enviar al usuario el mensaje completo (temperatura, humedad y marca temporal) en caso de que la humedad supere el 40%. Como en la regla de DynamoDB, se recoge el cuerpo del último mensaje aceptado por la plataforma y se analiza el campo humidity para decidir si se envía la notificación.

Para la acción, se selecciona la única disponible con SNS:

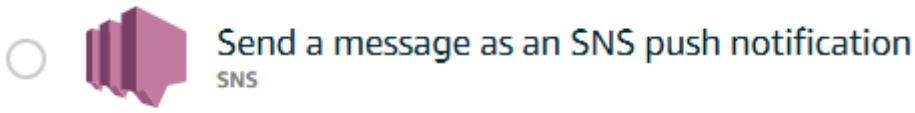


Imagen 56 Opción de SNS como acción

En la pantalla de configuración de la acción, será necesaria la creación de un nuevo recurso. En este caso, el recurso será un tema.

Un tema es una de las formas que proporciona la plataforma de Amazon para agrupar los mensajes. En el panel de temas, por ejemplo, aparecerá la tabla de DynamoDB creada en el paso anterior.

Para gestionar las notificaciones, se creará un tema nuevo, al cual se le asignará un nombre identificativo y el nombre que se mostrará como remitente en el caso de los SMS.

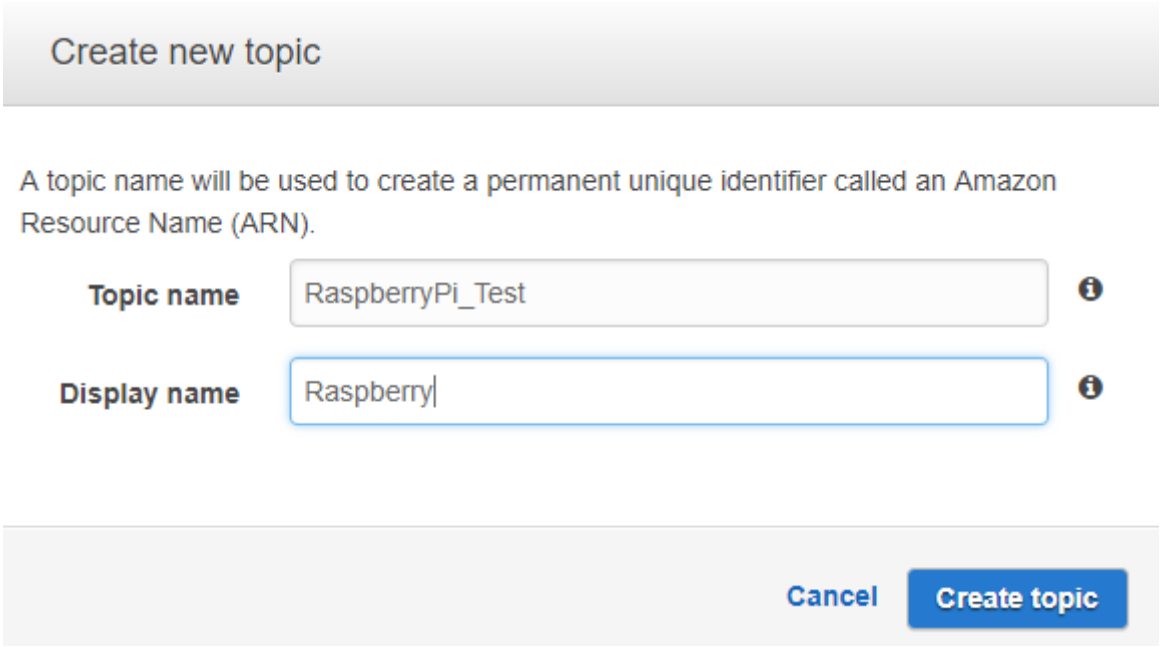


Imagen 57 Creación de un nuevo tema de difusión

Una vez creado el tema, se puede acceder al panel de control con la uri interna de Amazon, a la cual denominan ARN. Dentro del panel de control será necesario crear una nueva suscripción, la cual sirve para enviar la notificación por un canal y a un usuario.

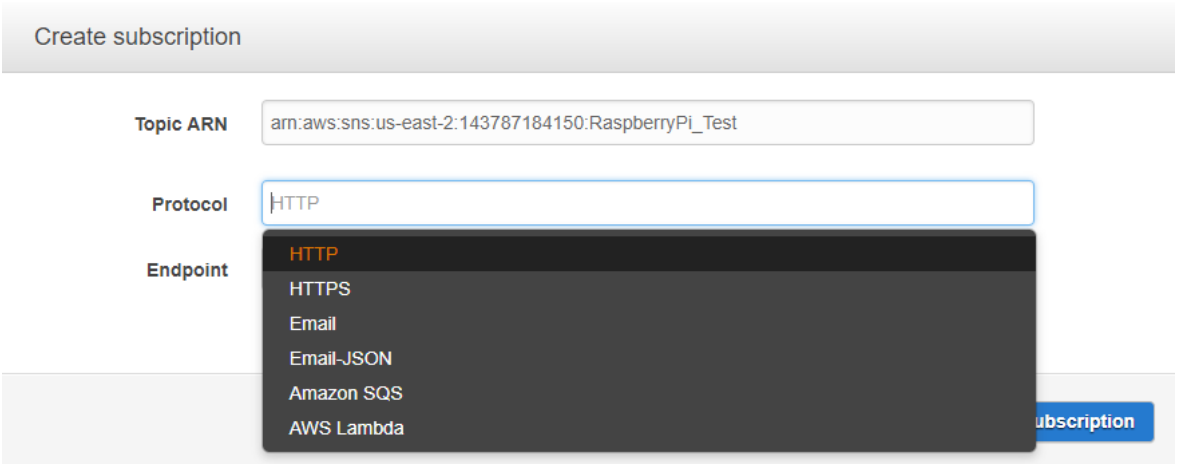


Imagen 58 Creación de una subscripción a un tema

En este caso, la prueba consistirá en un email que contiene el JSON del mensaje. Es un simulacro de los mensajes que suelen recibir los técnicos de mantenimiento de equipos con mediciones en tiempo real.

Para la prueba, se utilizará una dirección de correo electrónico y el protocolo seleccionado será Email.

Imagen 59 Subscripción creada y a la espera de confirmación

Las suscripciones tienen un estado inicial pendiente de confirmación para los emails. Esto es así por una cuestión de seguridad, ya que el receptor de estas notificaciones tiene que confirmar su dirección de correo electrónico.

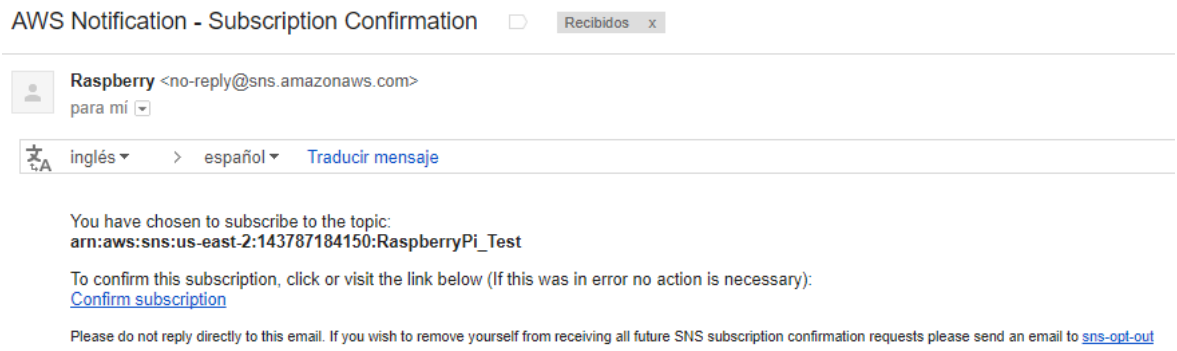


Imagen 60 Mensaje de confirmación a una subscripción

Ahora, para recibir la notificación, hay que hacer que el sensor recoja más de 40% en el ambiente. En cuanto se dé el escenario configurado, llegará la siguiente notificación mediante correo electrónico:

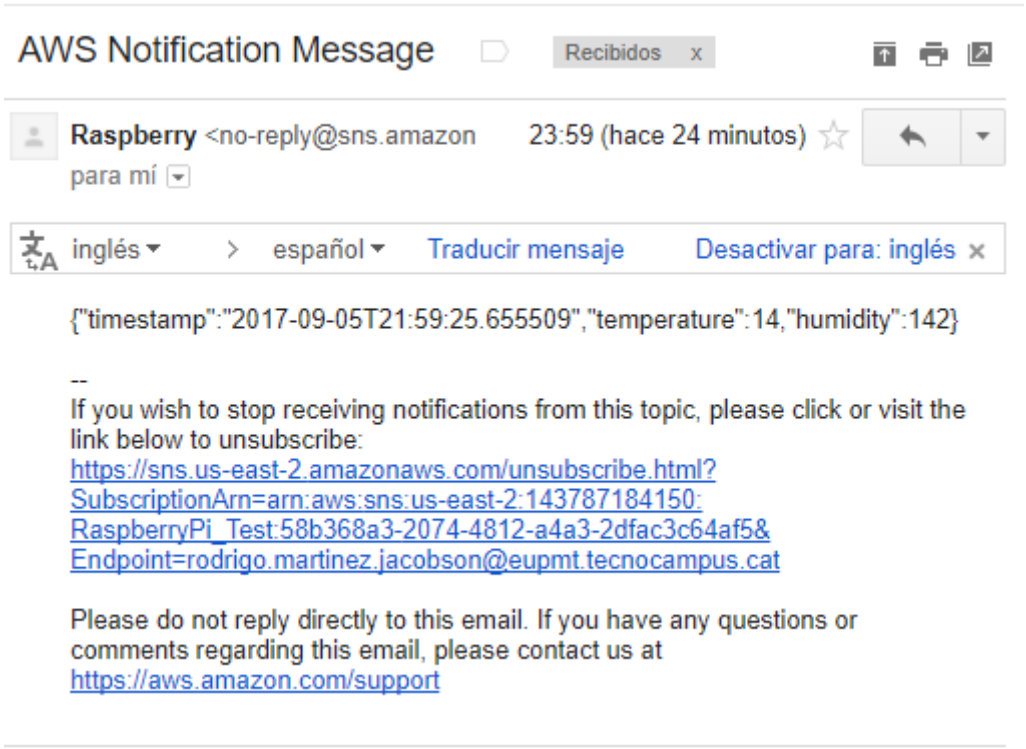


Imagen 61 Ejemplo de notificación mediante un correo electrónico

4.1.5. Representación gráfica en Kibana

Para representar los datos, la plataforma de Amazon ofrece una solución interna basada en ElasticSearch con Kibana.

ElasticSearch es un motor de búsqueda basado en Lucene, y se utiliza principalmente en entornos Enterprise para el almacenamiento y análisis de logs. Sobre esta herramienta se ha creado un panel llamado Kibana para mostrar de forma gráfica el resultado de los análisis de los datos o documentos almacenados en un dominio de ElasticSearch.

El primer paso es configurar una regla que acciona una respuesta en ElasticSearch, seleccionando la siguiente acción en el listado ofrecido por el asistente:

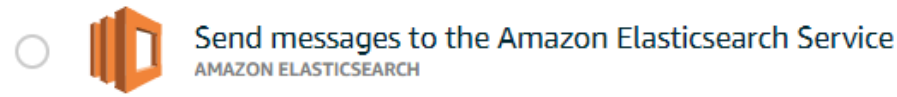


Imagen 62 Opción de ElasticSearch como acción

Una vez seleccionada esa acción, será necesario crear un nuevo dominio de ElasticSearch. En el asistente para configurar el dominio, hay que dar un nombre al dominio, seleccionar la capacidad del servidor, la hora cuando se realiza la copia de seguridad, etc. Todos estos ajustes vienen con las opciones más básicas seleccionadas de forma predeterminada, así que sólo hará falta asignar un nombre.

Una vez hecha la configuración básica, Amazon solicitará el perfil de seguridad para este dominio. Para esta prueba de concepto es suficiente con asignar el perfil de acceso abierto para todos:

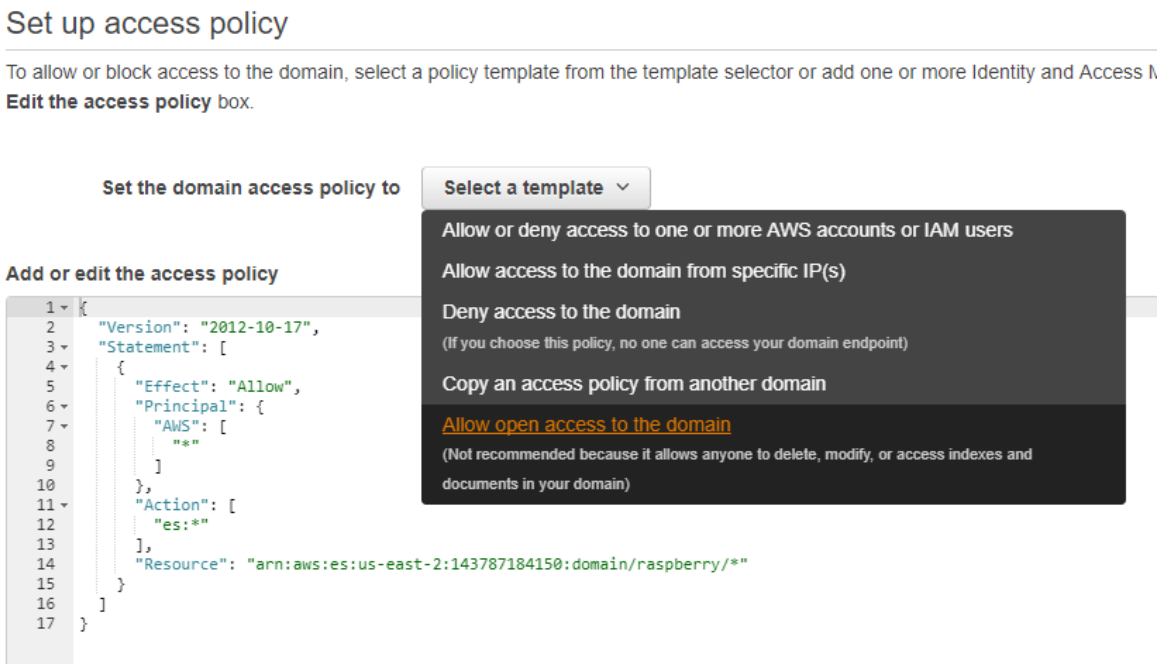


Imagen 63 Configuración de una política de acceso

En la siguiente ventana, la plataforma solicita confirmar los datos del dominio y, una vez aceptados, procede al despliegue de la estancia EC2 en la plataforma.

A continuación, una vez creado el dominio de ElasticSearch, se procederá a crear la regla, la cual enviará un objeto al dominio y permitirá configurar el panel de Kibana. Para crear la regla, se seleccionará el dominio creado, se asignará un identificador universal único autogenerado por la plataforma, y el objeto que se desea enviar con la información. En este

caso, el objeto será llamado dht ya que contiene la información de la humedad y temperatura recogida por el sensor.

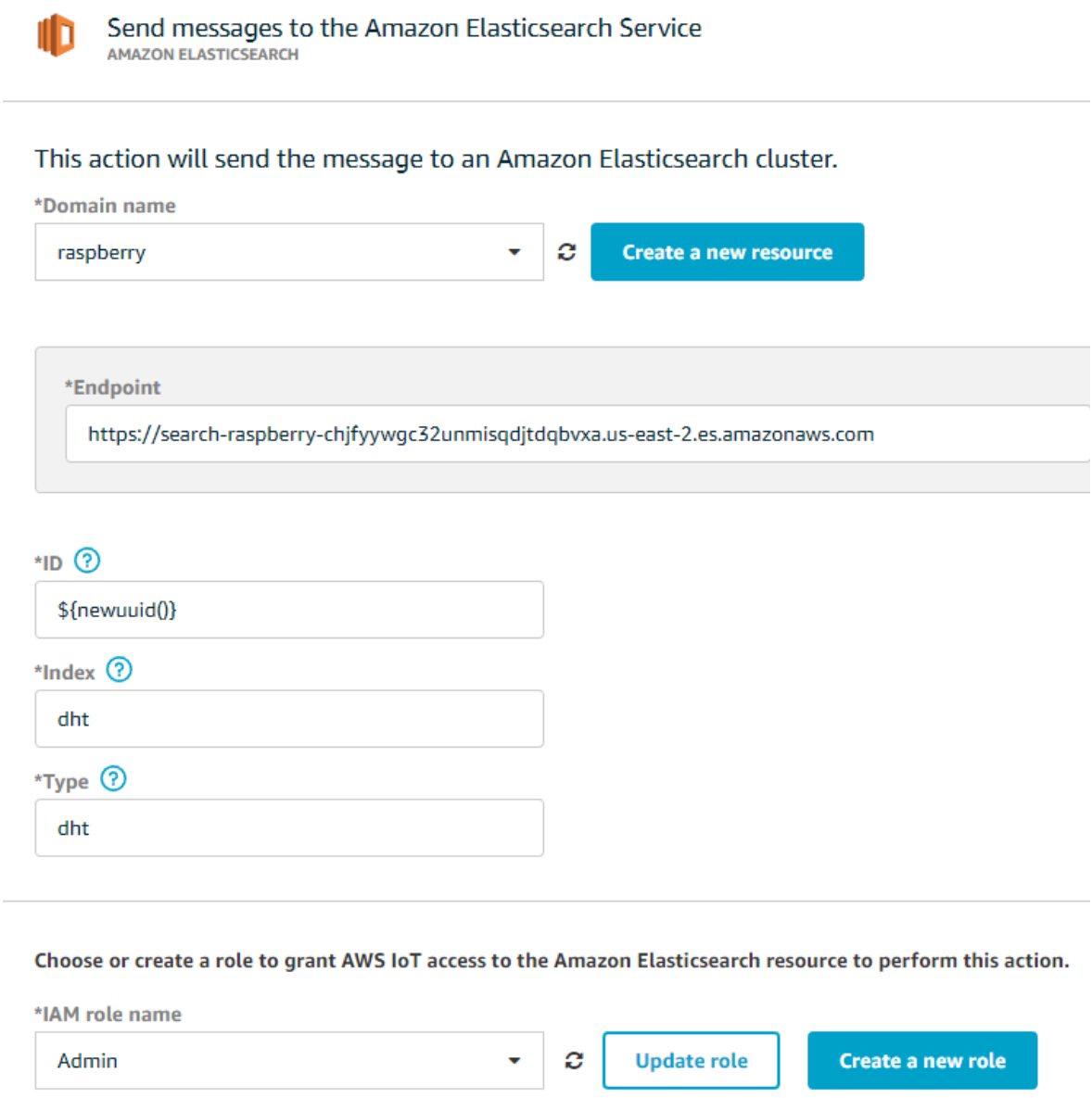


Imagen 64 Datos a enviar al panel de ElasticSearch desde la regla

El siguiente paso, una vez configurado el dominio y creada la regla, es acceder al panel de Kibana enlazado a este. Dentro de este panel, será necesario indicar el formato de los datos a analizar. Para este panel, se utilizará de índice el objeto dht y como índice temporal se utilizará la marca temporal que este contiene.

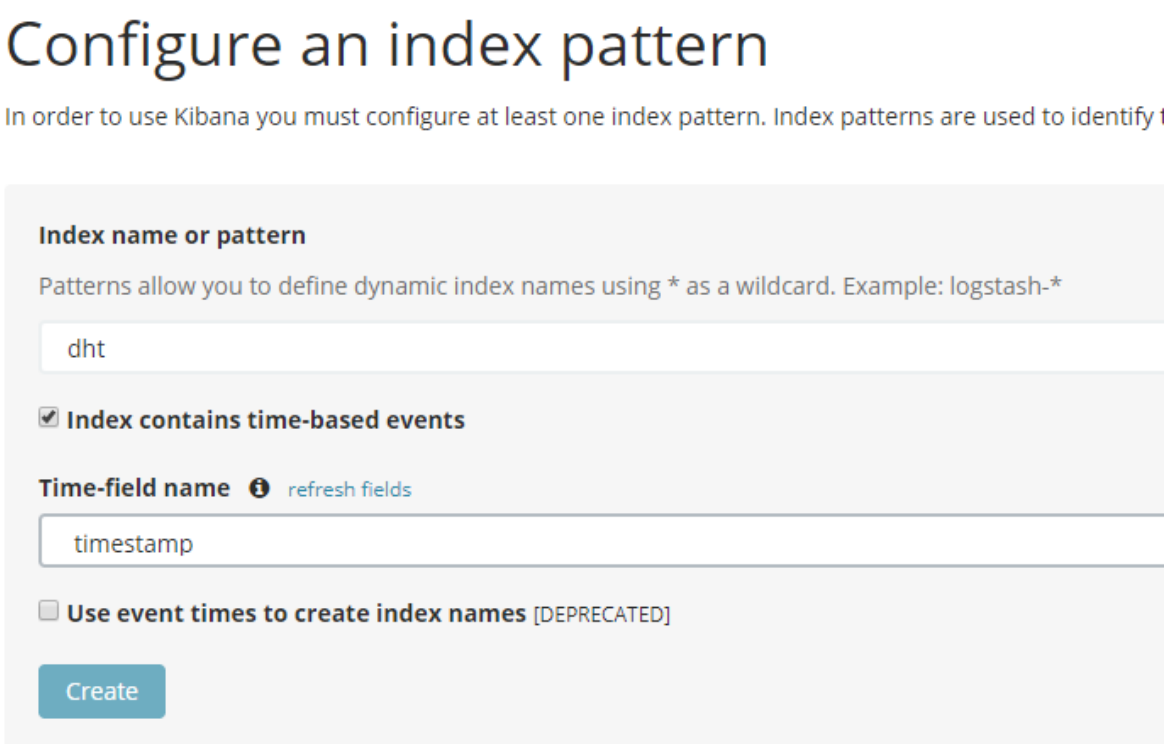


Imagen 65 Configuración inicial de Kibana

Una vez configurado todo, desde el apartado discover del panel de Kibana, se puede ver lo datos recibidos:

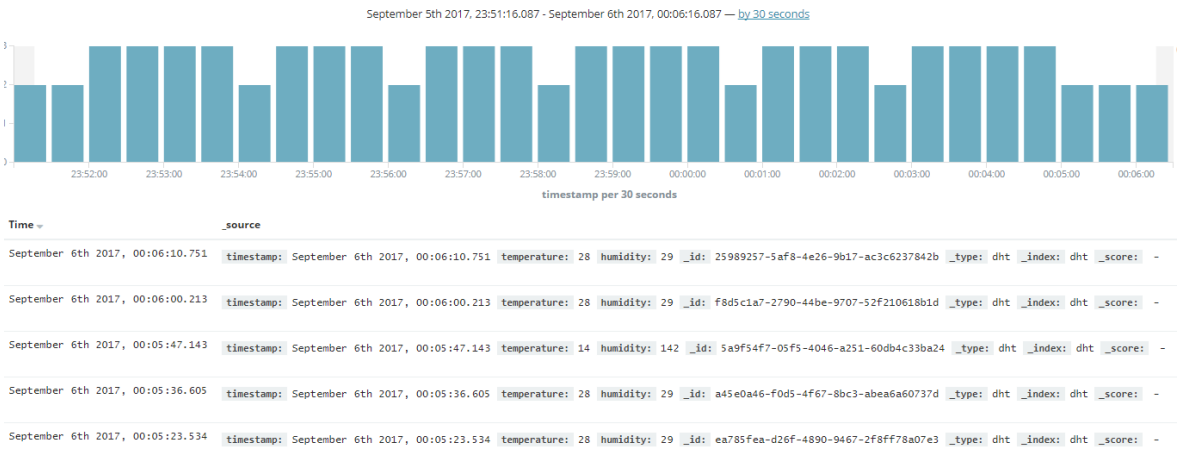


Imagen 66 Visualización por defecto de los datos en Kibana

Para crear un panel donde se muestren de forma independiente la humedad y la temperatura, primero se tendrán que crear los componentes en el apartado visualize. Dentro de este apartado, se puede crear un gráfico a partir de la información transmitida por el dispositivo:

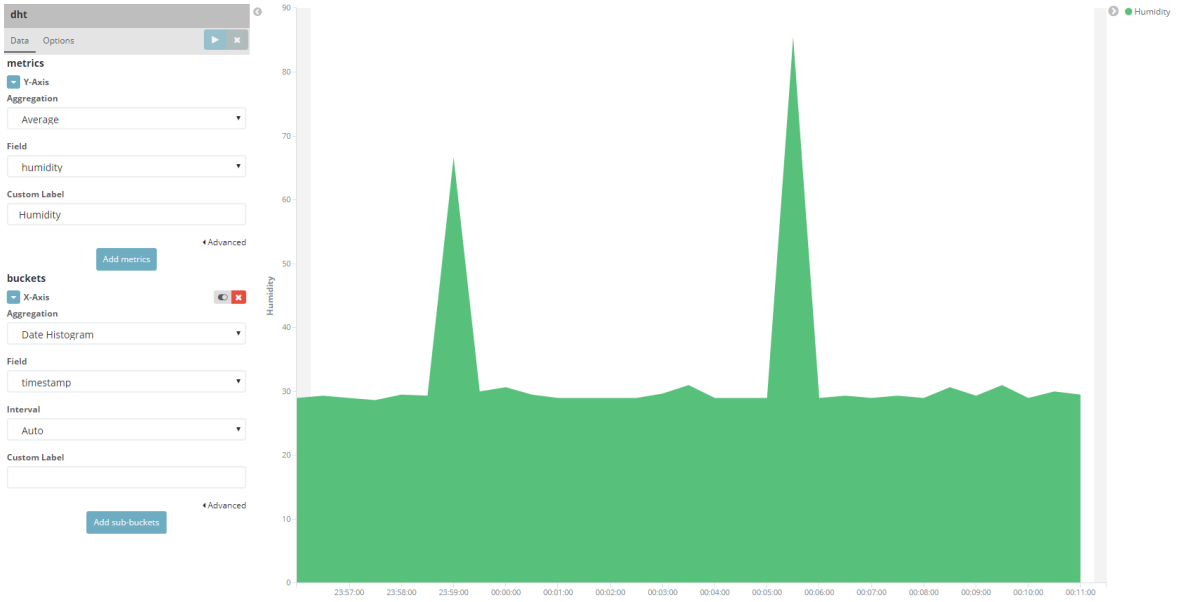


Imagen 67 Creación de un componente en Kibana

Una vez creados los componentes, desde el apartado dashboard se puede crear un panel donde añadir varios componentes, quedando algo así:

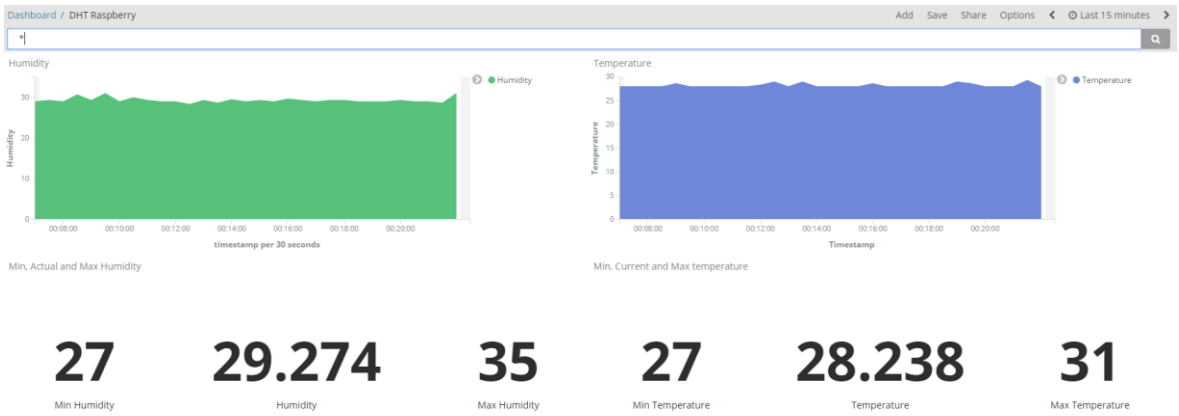


Imagen 68 Panel compuesto por múltiples componentes

4.2. Thingspeak

Esta plataforma ha sido escogida por ser una de las plataformas de código abierto más completas y con mayor comunidad. La cantidad de posibilidades que ofrece la plataforma sigue todavía muy lejos de los exponentes del sector tales como AWS IoT, Azure IoT y Watson IoT, pero ha sido capaz de suplir las carencias integrándose con servicios y plataformas de terceros. Además, pese a ser de código abierto, está respaldada por una empresa como MathWorks, con mucha solera en ofrecer herramientas y lenguajes para el análisis estadístico de datos.

Por otro lado, la sencillez de la que hace gala la plataforma y el poder instalarla en un servidor privado local o en la nube, hace que sea la elección de muchas organizaciones privadas, públicas y centros educativos para la implementación de proyectos del internet de las cosas.

4.2.1. Panel ThingSpeak

La plataforma creada por MathLabs denomina a su panel principal como Canal. Estos canales pueden contener hasta 8 tipos campos o variables distintos.

Para esta prueba de concepto, se creará un canal con el nombre Raspberry DHT y dos campos que contendrán la humedad relativa y la temperatura. Es muy importante recordar el orden asignado a los campos, ya que en las peticiones se referencia por número de campo y no por nombre asignado.

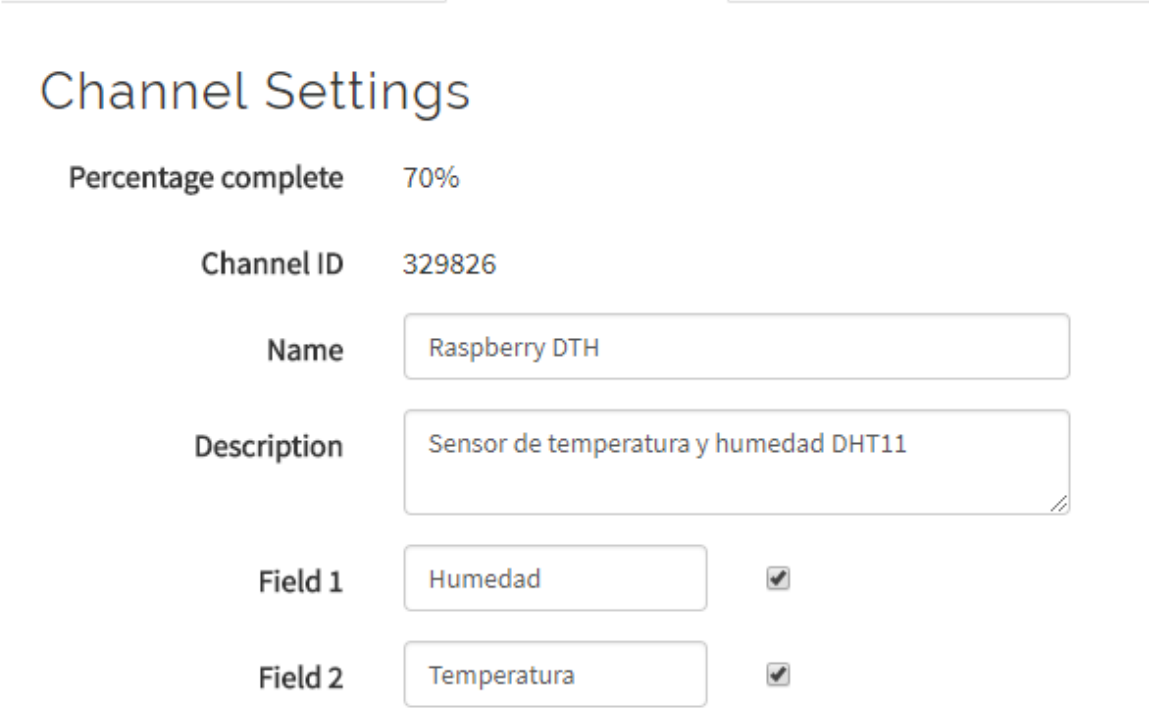


Imagen 69 Configuración de un canal en ThingSpeak

La configuración del panel es tan simple como asignar un nombre, una descripción y al menos 1 campo. Una vez hecho esto, se proporcionará el ID del canal y la clave de la API para escribir en este.

Por defecto, el canal es privado y toda interacción con este mediante la API deberá realizarse con las claves de escritura y lectura. Para cambiar las opciones de seguridad se debe acceder a la pestaña "Sharing". En cambio, si lo que se desea es consultar las claves de escritura/lectura, se debe acceder a la pestaña "API Keys".

Para esta prueba de concepto se configurará el canal como público, para que sea más sencilla la interacción con las herramientas de MATLAB.

Por último, la plataforma ofrece ciertas integraciones con servicios de terceros, tales como la API de Twitter o un equivalente a las reglas de AWS IoT, a las cuales llaman "React".

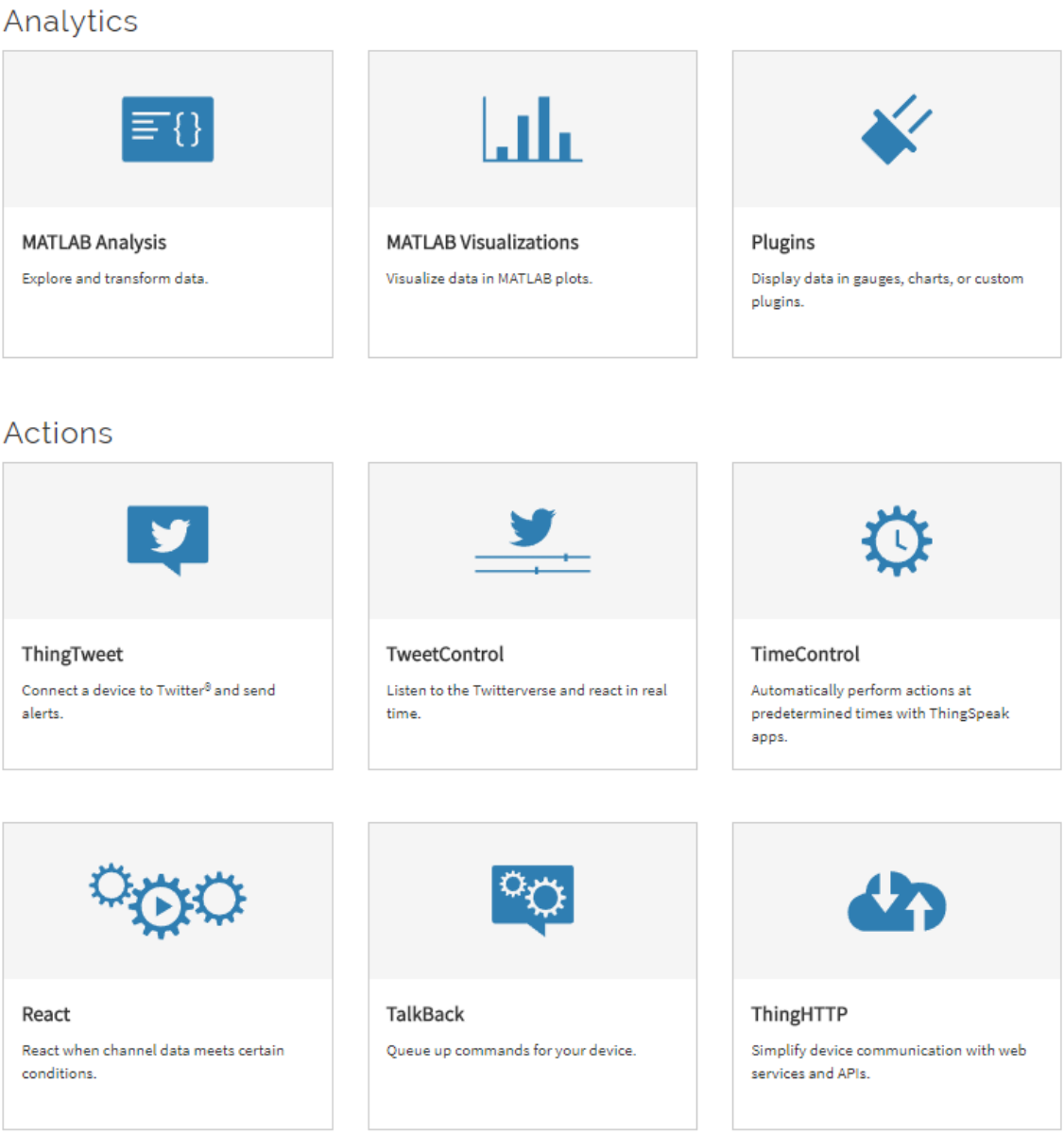


Imagen 70 Opciones de la plataforma ThingSpeak

Estas opciones de la plataforma serán utilizadas más adelante para enviar alertas cuando algún parámetro supere cierto límite.

4.2.2. Envío de datos a la plataforma

Para esta prueba de concepto, la configuración del dispositivo es bastante similar a la que se realizó con AWS IoT, ya que comparten gran parte del proceso de recolección y envío de datos.

El software necesario para el desarrollo del script en Python es el siguiente:

- Python 2.7.
- Cliente MQTT Eclipse Paho.
- Librería del controlador de Adafruit para el sensor DHT11.

El script necesario para obtener los datos de temperatura y humedad del sensor será exactamente el mismo que se utilizó en la prueba de AWS IoT.

En cambio, para la parte de transmisión de datos, el script variará. En esta ocasión, como el sistema de seguridad de ThingSpeak se basa en claves de seguridad y no en certificados, con una conexión sencilla (Single) mediante la librería Publish de Paho será suficiente para enviar los datos.

Para escribir los datos, hay que indicar en una uri el canal y la clave de escritura. De esta forma, ThingSpeak simplifica el paso de enviar datos a la plataforma, permitiendo que múltiples dispositivos se conecten a un mismo canal bajo la misma clave.

El primer paso para desarrollar el script es parametrizar el canal y la clave para obtener un script que se puede reutilizar para diferentes canales.

```
#  ThingSpeak Channel Settings

# The ThingSpeak Channel ID
channelID = "329826"

# The Write API Key for the channel
apiKey = "5MINCKRH4TIQRWZ9"

# The Hostname of the ThinSpeak MQTT service
mqttHost = "mqtt.thingspeak.com"

# Connection properties
tTransport = "tcp"
tPort = 1883
tTLS = None
```

Imagen 71 Datos de conexión a ThingSpeak en el script

De esta forma, se le indica al script el identificador único del canal, la clave de escritura, la dirección base del servicio MQTT de ThingSpeak y las propiedades de la conexión. Para

esta prueba de concepto, la conexión será simple, aunque se podrían utilizar WebSockets (seguros o inseguros) si fuese necesario.

El último apartado del código es muy parecido al utilizado con AWS IoT. En este caso, se genera la uri del canal, al cual llamamos Topic, y el mensaje que contiene la humedad y la temperatura.

```
# Create the topic string
topic = "channels/" + channelID + "/publish/" + apiKey
print(topic)
while True:
  # build the payload string
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
    tPayload = "field1=" + str(humidity) + "&field2=" + str(temperature)
    print(tPayload)
    print('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
    # attempt to publish this data to the topic
    try:
        mqtt.single(topic, payload=tPayload, hostname=mqttHost, port=tPort, tls=tTLS, transport=tTransport)
        print("Sent to ThingSpeak")
        time.sleep(20)
    except (KeyboardInterrupt):
        break

    except:
        print ("There was an error while publishing the data.")
```

Imagen 72 Código que recopila los datos del sensor y los envia a ThingSpeak

A diferencia de Amazon, ThingSpeak no envía los datos mediante un mensaje JSON, sino que los envía como parámetros individuales.

Se realizará un intento de enviar la información mediante la función Single de la librería Publish de Paho. Esta función establece una conexión con el servidor, publica un único mensaje y cierra la conexión. A la función se la pasa por parámetros el mensaje a enviar y los parámetros de conexión.

Si la información ha sido publicada correctamente, el script se detiene durante 20 segundos, para no tener problemas con la limitación del plan gratuito de ThingSpeak (1 mensaje cada 15 segundos).

Una vez ejecutado el script, se mostrará por pantalla el mensaje a enviar a la plataforma, los valores recogidos por el sensor y un mensaje indicando si la publicación se ha realizado correctamente o se ha producido un error:

```
field1=29.0&field2=26.0
Temp=26.0*C  Humidity=29.0%
Sent to ThingSpeak
```

Imagen 73 Resultado de la ejecución del script

4.2.3. Representación gráfica con MATLAB

La plataforma ofrece por defecto una gráfica con las series de tiempo de cada campo configurado en el canal. Esto hace que la plataforma sea ideal tanto para usuario con proyectos muy sencillos como para usuarios avanzados.

Imagen 74 Representación gráfica de los datos en MATLAB

En caso de querer añadir más representaciones de los datos, ThingSpeak ofrece las "Visualizaciones" mediante MATLAB:

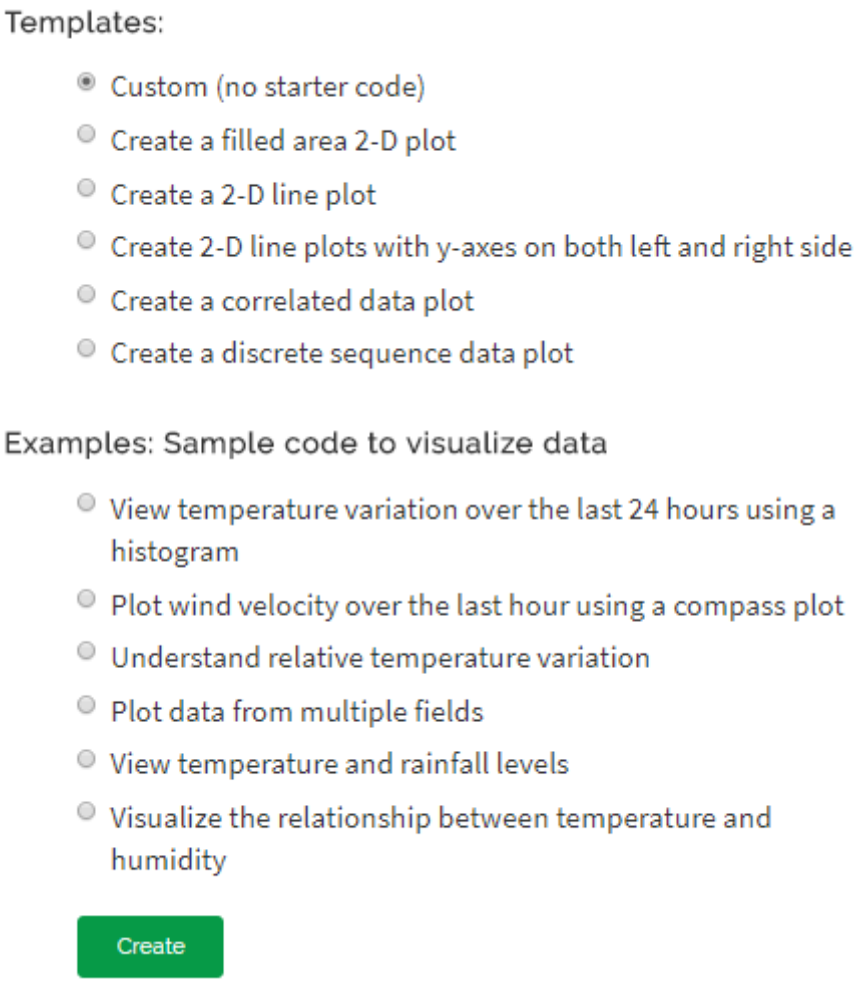


Imagen 75 Opciones de creación de nuevos componentes para el panel

Estas visualizaciones son programadas en M, lenguaje propietario de la herramienta. La plataforma ofrece distintas plantillas para ayudar a crear nuevas visualizaciones. Además, dispone de ciertos ejemplos para analizar y mostrar los datos en diferentes formas.

Para esta prueba de concepto se escogerá el ejemplo que muestra la variación de la temperatura cada 24 horas.

Una vez seleccionado el ejemplo, se creará automáticamente el código señalando un canal público dentro de la plataforma.

Para que los datos mostrados sean los que muestra el canal que se ha configurado previamente, se modificarán las variables “readChannelId” y “TemperatureFieldID”. Los datos necesarios sobre el canal se encuentran a la derecha en un cómodo panel.

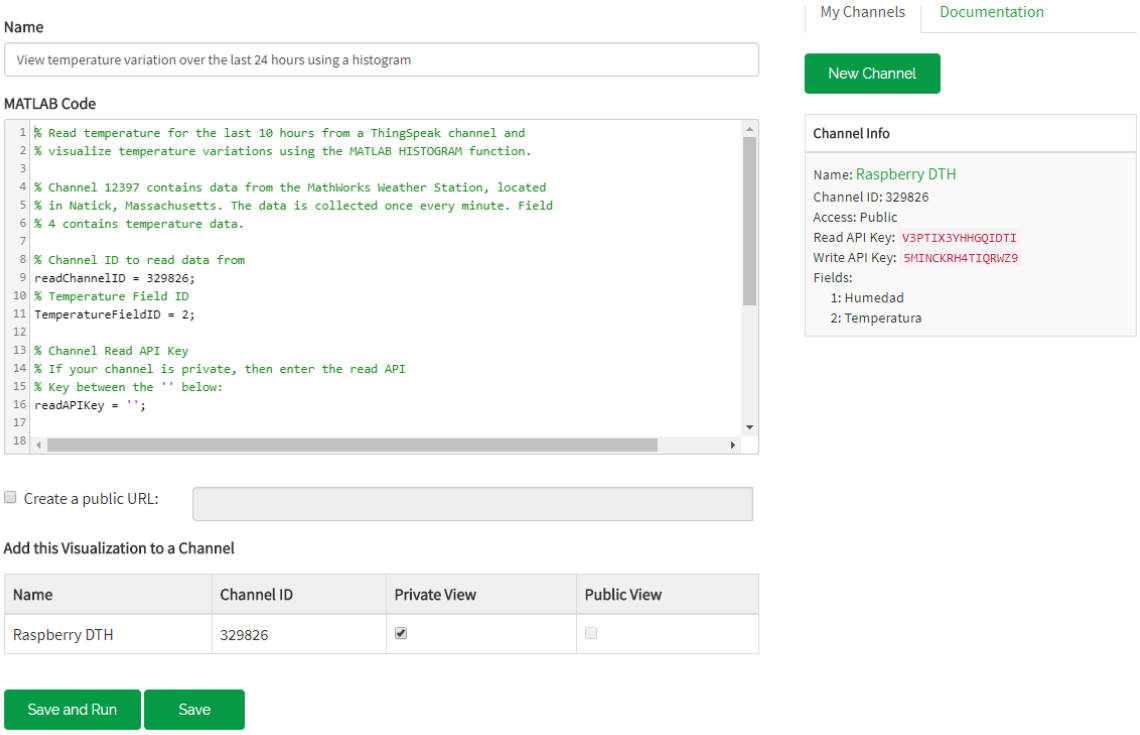


Imagen 76 Creación de un nuevo componente

Una vez modificadas esas variables, se indicará que se desea añadir la visualización a la vista privada y se hará clic en el botón “Save and Run” para poder ver el resultado.

El resultado final quedaría así:

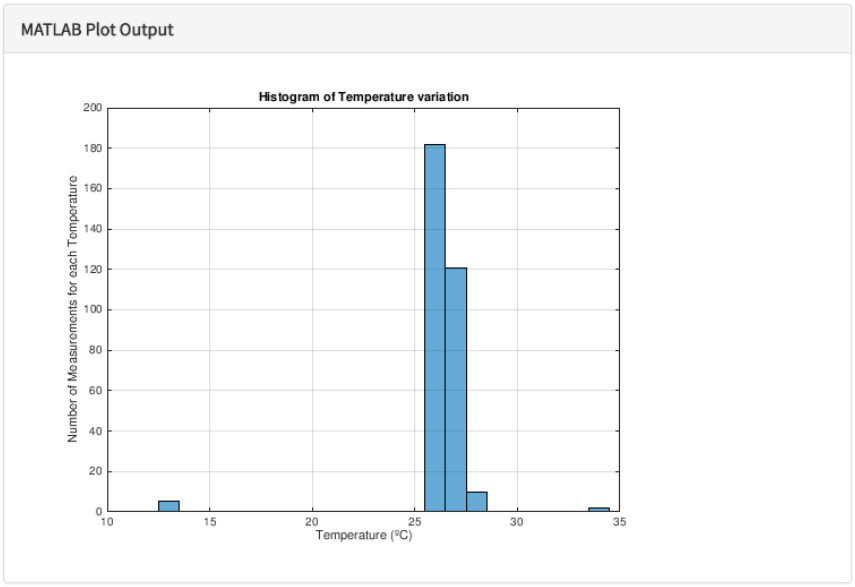


Imagen 77 Aspecto del componente una vez creado

Y si se consulta la vista privada, se podrá comprobar como la visualización ha sido añadida a esta:

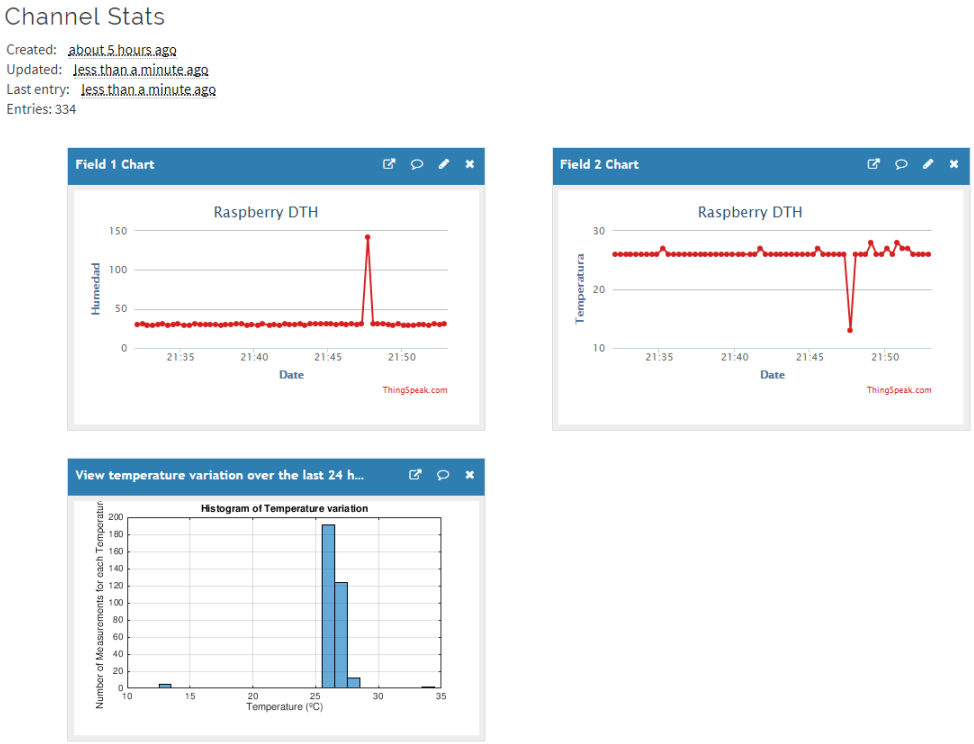


Imagen 78 Panel del canal con el nuevo componente

Las posibilidades de análisis y representación de los datos son las mismas que en la versión de escritorio de MATLab, por lo tanto, la plataforma ofrece unas posibilidades sólo limitadas por el conocimiento del usuario sobre el lenguaje de programación M.

4.2.4. Implementación de notificaciones vía email

La plataforma da la opción para enviar avisos o notificaciones mediante una integración con Twitter de forma nativa, pero, a la hora de enviar otro tipo de notificaciones, estas quedan en manos de los desarrolladores o de servicios de terceros.

Para poder simular la misma funcionalidad que ofrece Amazon mediante SNS, se utilizara la acción ThingHTTP, que permite enviar peticiones a las APIs de terceros y que estas gestionen el envío del correo electrónico como alerta.

El primer paso es configurar el servicio que enviará el correo electrónico. Para esta prueba de concepto, se utilizará el servicio recomendado por la comunidad de ThingSpeak, PushingBox.

Una vez creada una cuenta en este servicio, en la pestaña “My Services”, se encuentra el apartado donde crear la acción. En este caso se creará un servicio de enviar un correo electrónico a la dirección indicada:

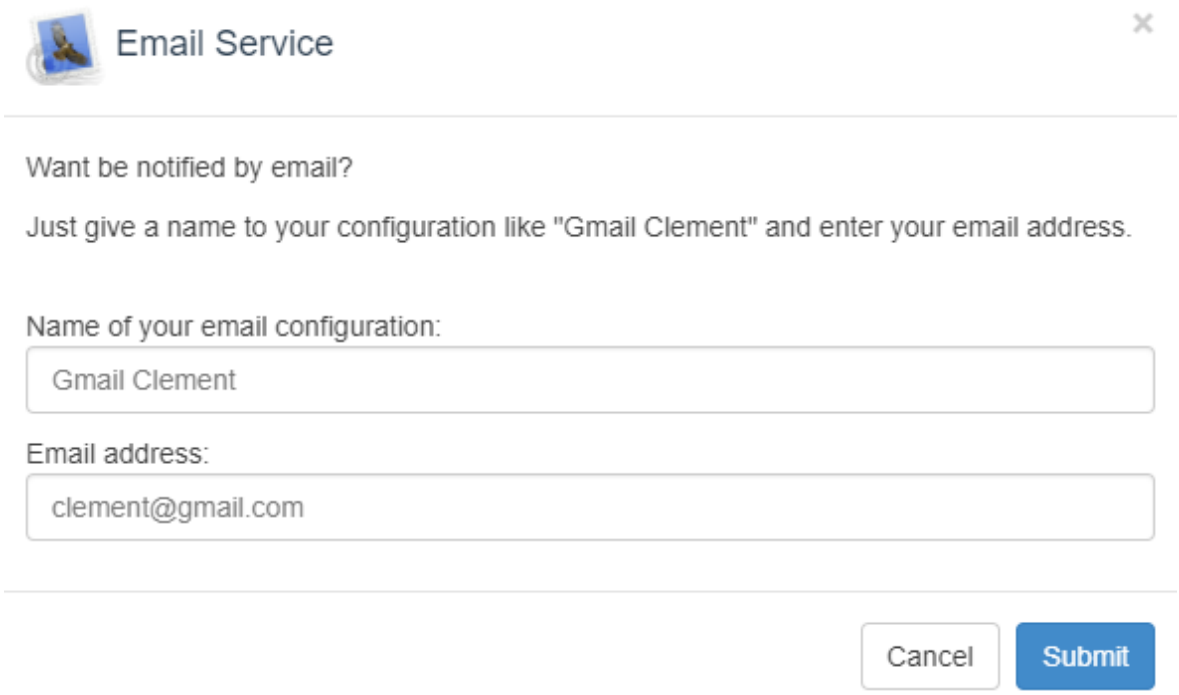


Imagen 79 Creación de un servicio en PushingBox

Con el servicio ya configurado a la dirección donde se desea recibir notificaciones, es necesario crear un “Scenario”, el cual es el que enlaza la acción con el servicio. Se le asignará el nombre “Send Email” a este escenario y se seleccionará el servicio creado previamente.

Una vez creado el escenario y enlazado el servicio, sólo queda configurar la acción para que envíe un correo electrónico. El mensaje de este correo electrónico será la temperatura y la humedad que había en la sala en el momento que se cumplió la condición configurada en ThingSpeak.

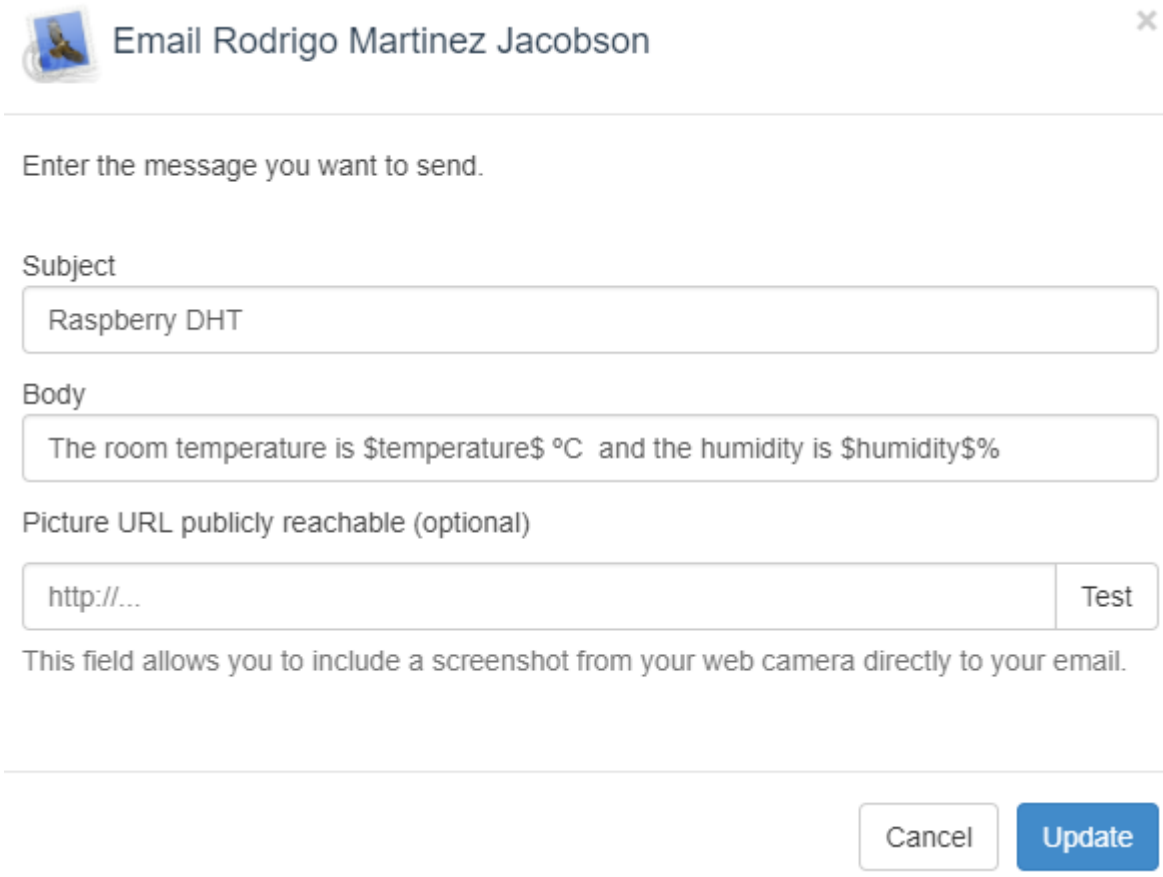


Imagen 80 Configuración del mensaje de alerta

La forma de introducir variables en el mensaje es mediante la siguiente sintaxis:

$NombreVariable$

Con la configuración del envío del correo electrónico mediante PushingBox, el siguiente paso es configurar el disparador para esta alerta, mediante la herramienta React de la plataforma. Esta herramienta permite lanzar una acción cuando se cumpla una condición.

Para poder crear el disparador e invocar la acción, primero es necesario tener creada la acción. En la sección “Apps”, dentro del conjunto “Actions”, se encuentra la categoría ThingHTTP. Se creará una acción de este tipo con la siguiente configuración:

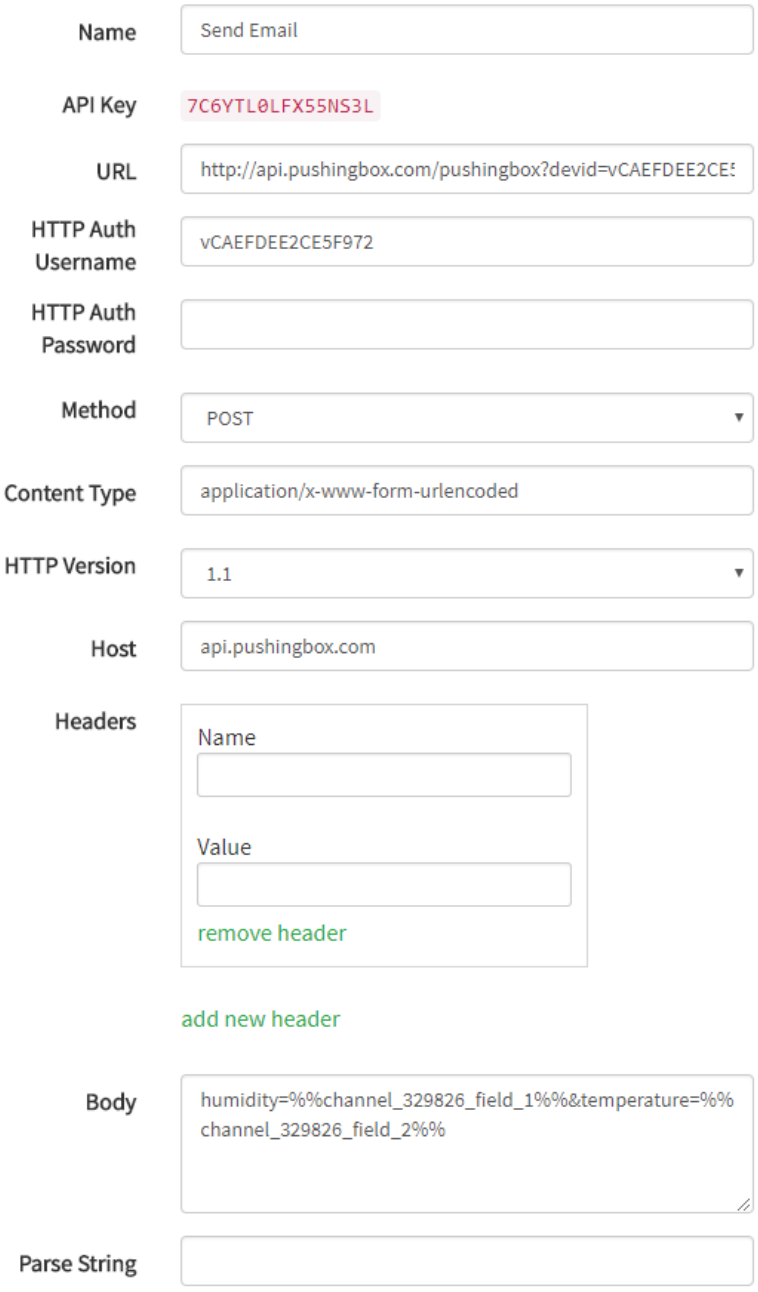


Imagen 81 Configuración de la acción en ThingSpeak

La configuración básicamente consiste en llamar a la API de PushingBox, enviando como parámetro de la uri el DeviceId proporcionado al crear el escenario.

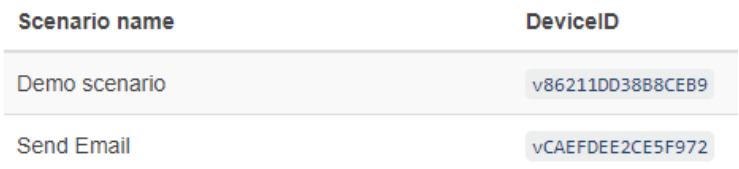


Imagen 82 Escenarios disponibles en ShoutingBox

Además, el usuario también será el DeviceId. En el apartado del cuerpo del mensaje, se enviarán los datos. En este caso los datos son la humedad y la temperatura. Como ThingSpeak no les asigna un nombre a los campos, se utiliza la sintaxis por defecto. Para acceder a estos campos, se debe indicar el canal y el número del campo de la siguiente forma:

%%channel_idCanal_field_idCampo%%

Finalmente, la configuración del disparador para las notificaciones mediante correo electrónico será la siguiente:

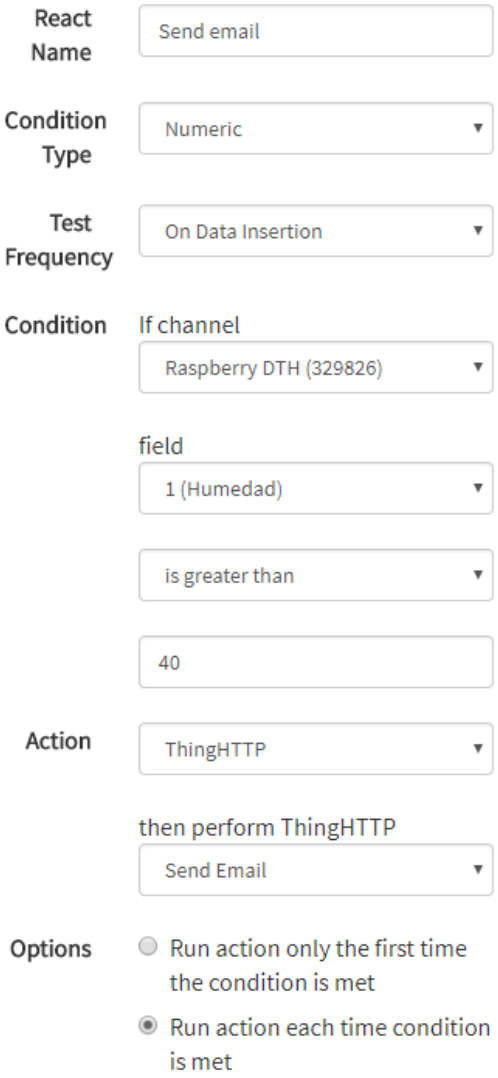


Imagen 83 Configuración del disparador en ThingSpeak

Una vez hecha toda la configuración, aumentando la humedad de la habitación por encima del 40% se recibirán notificaciones como esta:

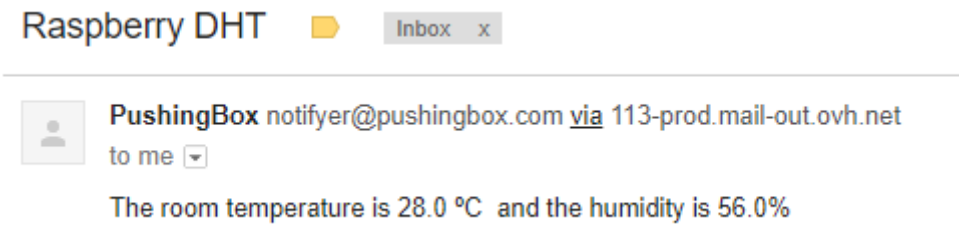


Imagen 84 Ejemplo de la notificación recibida mediante ShoutingBox

5. Conclusiones

El propósito inicial de este proyecto era estudiar las soluciones disponibles en el mercado de las plataformas de IoT y profundizar en el funcionamiento de las más destacadas y diferenciadas.

Analizar el mercado fue una tarea difícil, ya que filtrar las propuestas serias de las propuestas incompletas, abandonadas o con escaso soporte requería mucha investigación y análisis.

El mercado de las plataformas IoT está en auge debido a que la cantidad de dispositivos que requieren de una solución a la necesidad de gestionar la ingente cantidad de información recopilada crece a pasos agigantados. El problema es que, a diferencia de la gran innovación en estos dispositivos IoT para mejorar sus prestaciones y facilidad de uso cada día, la gran mayoría de las plataformas IoT se han estancado en una arquitectura incapaz de asimilar los cambios que se producen en el mercado a un ritmo vertiginoso.

Exceptuando algunos casos como las grandes dominadoras del sector (AWS IoT, Azure IoT, Watson IoT), las cuales están respaldadas por gigantes del software, los demás actores de este mercado están orientando sus productos a nichos específicos.

Por otro lado, las propuestas de código abierto han demostrado que, pese a ser bastante solventes a la hora de competir con la mayoría de las soluciones privativas, también carecen de argumentos a la hora de plantar cara a las grandes soluciones orientadas a la nube.

Este análisis de las diferentes plataformas me ha permitido entender que hay un nicho con cada vez más usuarios que había sido bastante descuidado hasta ahora, ya que la mayoría de las soluciones del mercado están orientadas al usuario profesional y/o académico. Pero, plataformas como Adafruit.io, han sabido responder a la necesidad de la comunidad de usuarios domésticos, los cuales no harán más que crecer con el auge de tendencias como el de las impresoras 3D domésticas y la domótica low-cost.

Para llevar a cabo este proyecto, fueron necesarios conocimientos de diferentes disciplinas de la carrera, adquiridos a lo largo de los últimos 4 años del Grado en Ingeniería Informática. Y es que, para entender un producto IoT y la plataforma con la que comunica, no sólo se necesitan conocimientos de programación para generar el software necesario para la

recogida y envío de la información. Desde la conexión de los sensores, la encapsulación de la información para ser enviada a través de la red y la representación gráfica de los datos, todo esto requirió conocimientos de redes, sistemas operativos, programación y estadística.

Finalmente, como objetivo para un futuro próximo, mi intención es seguir profundizando en la plataforma de Amazon, ya que me da una flexibilidad absoluta a la hora de decidir cómo implementar mis pruebas de conceptos y prototipos, y ofrece un precio muy competitivo para la cantidad de recursos utilizados.

6. Bibliografía

[1] Arquitectura de una plataforma IoT

https://about.sofia2.com/2016/11/11/arquitectura-de-referencia-de-una-plataforma-iot

[2] Documentación AWS IoT. *https://aws.amazon.com/es/documentation/iot/*

[3] DHT11 Python driver. *https://github.com/adafruit/Adafruit_Python_DHT*

[4] Eclipse Paho MQTT client. *https://github.com/eclipse/paho.mqtt.python*

[5] Documentación ThingSpeak. *https://es.mathworks.com/help/thingspeak/*

[6] Documentación Raspberry Pi. *https://www.raspberrypi.org/documentation/*