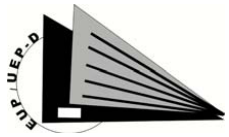
eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

MÁSTER UNIVERSITARIO EN

INGENIERÍA DE SISTEMAS EMPOTRADOS



Escuela Politécnica

Donostia-San Sebastián

IoT con IBM y NI

Alumno: **Jon Larrañaga Fuerte**

Director: **Andoni Arruti Illarramendi**

Instructor en la empresa: **Oskar Berreteaga Castro**

Proyecto Fin de Máster, julio de 2016

RESUMEN

El Internet of Things, abreviado IoT, es el concepto de dar conectividad a la red a cualquier objeto y poder capturar, almacenar y gestionar toda la información emitida por dichos objetos con la finalidad de automatizar procesos, permitiendo además la comunicación entre todos los objetos conectados.

Este proyecto se centra en crear una solución IoT, logrando además la integración de dispositivos de National Instruments con software de IBM. Para ello, se estudia la mejor manera de lograr la conexión de los dispositivos.

Una parte de esta solución IoT se trata de una aplicación web para poder explotar los datos enviados desde los dispositivos conectados, añadiendo a esta aplicación diversas alertas para poder avisar al usuario del estado de los dispositivos. Además se ha realizado la aplicación web pensando en un caso real de monitorización.

Por otra parte, se ha modelado el sistema de conexión de los dispositivos a la plataforma IoT de IBM con el fin de facilitar la conexión de otros dispositivos en el futuro.

Índice

Lista de figuras

Lista de tablas

1. INTRODUCCIÓN

El presente proyecto se centra en la creación de una solución IoT mediante la integración del software de IBM y el hardware de National Instruments, permitiendo la conexión de diversos dispositivos de National Instruments con la plataforma Watson IoT de IBM.

1.1. DESCRIPCIÓN GENERAL

En la iniciativa de Internet of Things (IoT) existen diversas soluciones tecnológicas para dotar a los dispositivos de funcionalidades de conectividad y agregación de datos. IBM propone su solución a través de IBM Watson IoT Platform. Esta plataforma se trata de un servicio alojado en la nube completamente gestionada, diseñada para simplificar y derivar los valores de los dispositivos IoT. En este caso, aprovechando la existencia de dispositivos de National Instruments, se precisa conectar diversos dispositivos de esta casa, como compactRIO-9033 o myRIO, con IBM Watson IoT Platform. De esta forma se logrará la implementación de IoT en estos sistemas empotrados dejando abierta la puerta a la posibilidad de añadir otros dispositivos en el futuro.

1.2. OBJETIVOS DEL PROYECTO

El proyecto se engloba dentro de la estrategia de ULMA Embedded Solutions de integración entre National Instruments e IBM, además de innovar en el mundo del Internet of Things. El objetivo principal del proyecto es crear un proceso de integración de dispositivos de National Instruments con la plataforma IBM Watson IoT. Para lograrlo se debe de estudiar la plataforma y determinar cuál es la mejor manera para crear la aplicación que permita la conexión de los dispositivos con la plataforma IBM Watson IoT. Además, el proyecto debe de incluir el diseño de una aplicación web en IBM Bluemix con el fin de explotar los datos obtenidos desde los dispositivos.

Por último, se debe de incluir el modelado de la conexión a la plataforma mediante IBM Rational Rhapsody para poder facilitar futuras conexión de nuevos dispositivos.

1.3. FASES DEL PROYECTO

1) Analizar el problema y estudiar la plataforma IBM Watson IoT.
2) Diseño, desarrollo y test de la aplicación de conexión a la plataforma.
3) Diseño, desarrollo y test de la aplicación web mediante IBM Bluemix.
4) Ampliar la solución modelando el software en IBM Rational Rhapsody.

1.4. ENMARQUE DEL PROYECTO

Este proyecto fin de máster se ha llevado a cabo en la empresa ULMA Embedded Solutions S.Coop ubicada en Oñati (Gipuzkoa). Esta empresa está situada dentro del grupo ULMA y proporciona servicios de ingeniería en el ámbito de sistemas embebidos.

El presente proyecto se enmarca dentro de la línea de desarrollo de soluciones para la integración entre IBM y National Instruments, así como dentro del desarrollo de soluciones para IoT.

Figura 1 – Logotipo de Ulma Embedded Solutions

1.4.1. SOBRE EL GRUPO ULMA

Hace cincuenta años que se formo el Grupo ULMA, un proyecto nacido de la ilusión así como del carácter emprendedor de unas personas que siguiendo ese sueño cooperativo, decidieron dejar sus anteriores empleos y se embarcaron en aquella aventura.

De aquel esfuerzo y dedicación nacieron diferentes negocios que perduran hasta el día de hoy. En una sucesión de cooperativas que han pasado a formar parte de un importantísimo, por cifra de negocio y número de personas que trabajan en el mismo, grupo empresarial que con ahínco trata de expandirse, internacionalizarse y diversificar su actividad; donde tal vez, esto último conforme su mayor valor o activo. No obstante, sin renunciar al espíritu cooperativo que marca su nacimiento y que todavía está absolutamente presente en el ADN de la entidad, hoy en

día se encuentra dentro de un grupo empresarial internacional mayor, la Corporación Mondragón, que es el mayor grupo empresarial cooperativo del mundo y también, por su singularidad, objeto de estudio por importantes Escuelas de Negocios a la vista de los éxitos obtenidos por este modelo.

1.4.2. Sobre ULMA Embedded Solutions

ULMA Embedded Solutions fue creada en 2009 en la incubadora del Grupo ULMA con el objetivo de ofrecer servicios especializados más allá de los prototipos funcionales de sistemas electrónicos, con una respuesta más acorde a las necesidades de la industria.

Hoy en día ofrece servicios especializados de ingeniería a lo largo de todo el ciclo de vida del producto electrónico, desde la conceptualización hasta la fabricación y mantenimiento, incluyendo la especificación de requisitos y el plan de validación, las fases de diseño, desarrollo y test.

Trabajan principalmente en sectores regulados por normativas y estándares como EN 50155, IEC 62304, EN 50128, ISO 26262, IEC 60730, EN 60601, UL 61010, ISO 13849, IEC 61508, etc.

Colaboran con el cliente y sus partners para desarrollar sistemas innovadores, competitivos y escalables. Entre su red de colaboradores se encuentran: Altium, IBM, LDRA, Mondragón Unibertsitatea, National Instruments, NXP, The Reuse Company, Xilinx y XJTAG.

2. ESTADO DEL ARTE

2.1. ORIGEN DEL INTERNET OF THINGS

Aunque parezca lo contrario, el término Internet of Things es un concepto bastante nuevo, ya que no fue hasta el año 2009 en el que el investigador Kevin Ashton empleó por primera vez este nombre. Lo mencionó en un artículo sobre sensores e identificadores de radio frecuencia y en él empleaba el término IoT como el concepto de conectar todos los dispositivos que nos rodean y saber el estado de cada dispositivo en cada momento. Pero había sido 10 años antes, en 1999, cuando el ingeniero Bill Joy había utilizado por primera vez el término IoT, en círculos privados de investigación, al lograr automatizar y controlar diferentes procesos mediante la comunicación establecida entre dos dispositivos conectados a Internet. Desde entonces el concepto ha ido creciendo y se ha generado gran expectación alrededor de él.

Hoy en día el término IoT es demasiado amplio como para que haya una definición estándar. Una de las definiciones es que el Internet of Things engloba a la dotación de conexión a Internet de todo dispositivo que nos rodea.

Entre los componentes más importantes que componen el IoT se encuentran los siguientes:

- El dispositivo: Se trata del objeto que se conecta a la red. Normalmente suele ser un dispositivo inteligente con bajo consumo de energía.
- La infraestructura de comunicación: Son las tecnologías de red que posibilitan la conexión de los dispositivos a Internet.
- La infraestructura de computación: Se trata de las herramientas que consumirán los datos enviados desde los dispositivos, como son las plataformas IoT y las aplicaciones IoT.

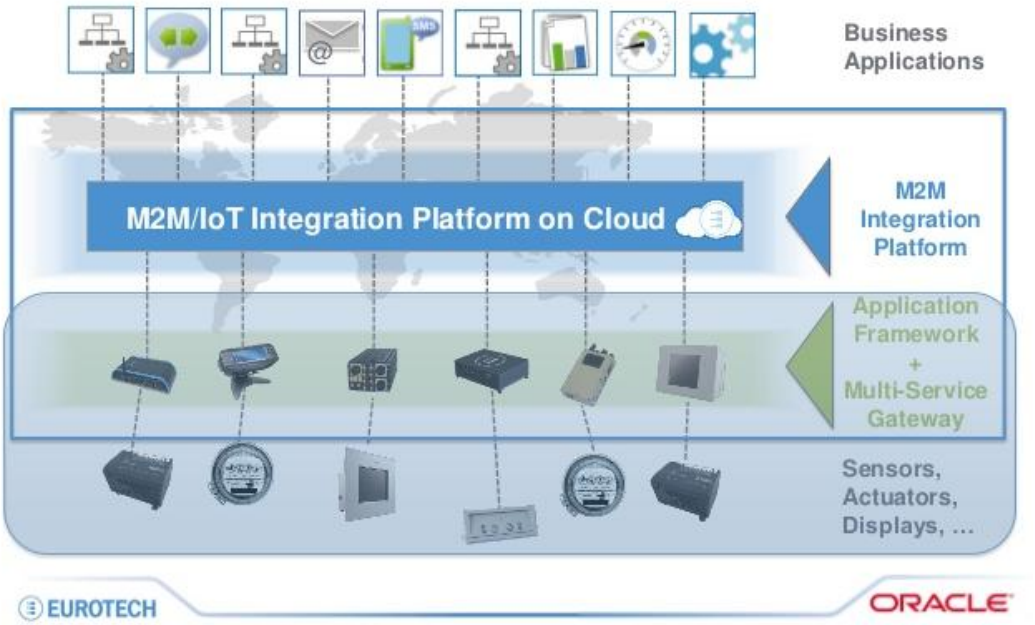


Figura 2 – Arquitectura de Internet of Things. Ej.: Oracle

Por otra parte, la fama que ha ido ganando este término es en gran medida por el crecimiento imparable de la cantidad de dispositivos conectados a Internet. Según Cisco, antes de que se conociese el término IoT existían en el mundo 500 millones de dispositivos por 6,3 mil millones de personas, esto quiere decir 0,08 dispositivos conectados a la red por persona. Pero en el año 2009, cuando se empezó a utilizar el término IoT mundialmente, se sobrepasó por primera vez la barrera de un dispositivo por persona.

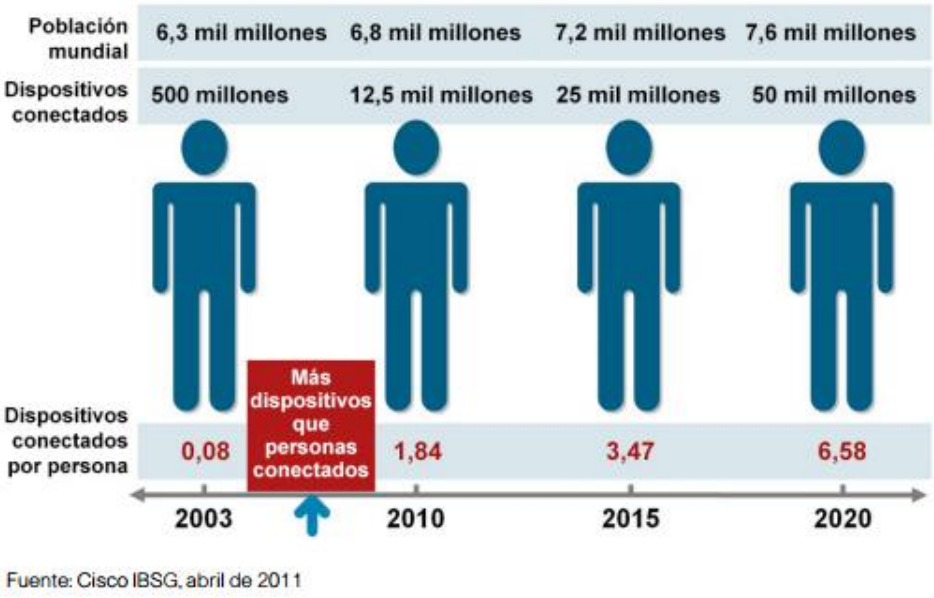


Figura 3 – Comparación de uso de dispositivos

Además del aumento del uso mundial de dispositivos, otros factores también han afectado en el uso masivo del término IoT. Entre los más destacados están:

- Nuevos sensores y dispositivos: Se han creado nuevos sensores con menor consumo de energía y coste, lo que ha supuesto su uso masivo en todo el mundo, además han aparecido nuevas plataformas hardware que se pueden combinar con estos sensores. Es el caso de Raspberry Pi o Arduino,
- La aparición del cloud computing: La creación del concepto de computación en la nube ha supuesto la puesta en marcha de nuevos servicios basados en la red.
- La aparición de los teléfonos inteligentes: Estos productos aprovechan los avances en las comunicaciones (Internet + Wi-Fi) para ofrecer la posibilidad de llevar una dirección IP en el bolsillo.
- Y por último el crecimiento de las redes inalámbricas de las que hablaremos en el siguiente tema.

2.1.1. REDES INALÁMBRICAS

La conexión inalámbrica es una de las culpables del impulso que ha cogido el IoT en los últimos años. Gracias a los progresos de la conectividad inalámbrica el IoT ha evolucionado, de estar al principio enfocada a la tecnología RFID a poder adoptar varias tecnologías para poder conectar diferentes dispositivos.

Para poder entrar más en detalle en los tipos de redes inalámbricos se deben de separar las redes según su rango de comunicación:

- **WAN** (Wide Area Network): Las redes que están dentro de esta categoría son aquellas que recorren grandes distancias, entre diferentes ciudades o países, esto es, se tratan de redes que se extienden sobre un área geográfica extensa. Por ello muchas veces suelen ser redes que utilizan porciones de redes de varias compañías diferentes.
- **MAN** (Metropolitan Area Network): Se tratan de redes que trabajan con mayor alcance que las redes locales dentro de una misma ciudad o entre varias ciudades cercanas.
- **LAN** (Local Area Network): Estas redes de área local son redes de propiedad privada con pocos kilómetros de extensión. Suelen utilizarse en redes dentro de una empresa o en el hogar.
- **PAN** (Personal Area Network): Son redes de corto alcance, de una extensión máxima de unos pocos metros.

Los siguientes tipos de redes inalámbricas son con los que la mayoría de aplicaciones de IoT están relacionadas:

- **Wi-Fi**

  Se trata de una red de datos inalámbrica que utiliza los estándares IEEE 802.11. Se enfoca dentro del tipo LAN y opera sobre las frecuencias de 2,4 GHZ y 5 GHz. Se diferencia de las demás redes por su compatibilidad nativa para redes IP, lo cual es muy importante dentro del ámbito del IoT, pero es una red que consume una cantidad de energía relativamente alta y la distancia de recepción de la señal es limitada, sobre unos 100 metros en espacio abierto y 20 en espacios cerrados,

- **Bluetooth**

Esta red se enfoca dentro del tipo PAN y se creó para la transmisión de servicios multimedia. A partir de esta tecnología se han creado otras y entre estas se encuentra el Bluetooth Low Energy (BLE). Esta nueva tecnología se enfoca en la reducción del consumo y en minimizar la potencia de transmisión. Los dispositivos wearables se basan sobre todo en este estándar.

- **Z-Wave**

Es un protocolo de comunicación dentro del tipo de red LAN que se enfoca en la domótica. Trabaja en la banda de 908.42MHz evitando las emisoras de la banda 2,4GHz. Su principal característica es la poca necesidad de energía y ancho de banda para transmitir.

- **Zigbee**

Es una tecnología que se enfoca dentro del estándar IEE 802.15.4 y está orientado a redes PAN para comunicar dispositivos de bajo coste y velocidad. Se utiliza en el control remoto y su uso se ha generalizado en la domótica, ya que se transmiten cantidades de información pequeñas y proporciona una larga duración de batería.

- **6LoWPAN**

Es un estándar que permite a las redes basadas en el estándar IEEE 802.15.4 el uso de IPv6, logrando que dispositivos que están conectadas a una red inalámbrica puedan comunicarse con dispositivos IP.

- **RFID**

Se trata de un estándar de identificación por frecuencia radial y se utiliza principalmente para identificar objetos a una distancia muy corta, de unos pocos metros, y la detección se hace mediante un lector estacionario que se comunica de manera inalámbrica con pequeñas etiquetas que están pegadas a los objetos. Estas etiquetas serán unas pequeñas baterías transpondedoras.

Entre todas estas redes, las consideradas más importantes en el ámbito del IoT se encuentran las redes Wi-Fi, BLE y los 802.15.4 (Zigbee y 6LoPAN). En la Tabla 1 se observa una comparativa entre estas tres redes.

	Wi-Fi	BLE	Zigbee
Estándar	802.11	802.15.1	802.15.4
Alcance máximo	100 metros	10 metros	75 metros
Duración de la batería	Días	Semanas	Años
Principal característica	La alta velocidad	El bajo coste y potencia	La alta escalabilidad y el bajo coste
Uso	En aplicaciones que requieran la transmisión de datos	Transmisión de contenido multimedia	Domótica y el control remoto.

Tabla 1 – Comparación entre redes en IoT

2.1.2. PROTOCOLOS

En cuanto a la comunicación entre dispositivos en IoT normalmente el mayor reto es evitar la pérdida de mensajes y el consumo excesivo de recursos de red. Para esto se han creado nuevos protocolos exclusivamente pensados para IoT, además de utilizar los protocolos de comunicación tradicionales.

Entre los protocolos más utilizados en el ámbito del IoT se encuentran los siguientes:

- **MQTT**

  Se trata de un protocolo de mensajería tipo publicación/subscripción con un flujo de datos optimizado para permitir reducir el tráfico de red. Está diseñado para redes de comunicación poco fiables y tiene un consumo de energía reducido. Este protocolo trabaja sobre TCP/IP.

- **MQTT-SN** (MQTT for Sensor Networks)

  Similar a MQTT pero con la diferencia de que trabaja sobre UDP por lo que está pensado para redes que no son TCP/IP. Está pensado también para dispositivos donde el bajo consumo de energía es primordial.

- **CoAP**

  Se trata de un protocolo de transferencia pensado para dispositivos con poca capacidad de procesado o memoria. También permite trasladar el modelo HTTP a redes restrictivas.

- **HTTP ReST API**

  Se trata de un protocolo muy utilizado y sencillo de implementar que facilita el desarrollo de servicio en el IoT. Tiene la ventaja que muchas aplicaciones están ya desarrolladas para utilizar esta tecnología.

- **AMQP**

  Protocolo enfocado en no perder mensajes, trabaja sobre TCP y proporciona una conexión punto a punto fiable. Se utiliza sobre todo en funciones de análisis basados en servidores.

2.1.3. CAMPOS DE APLICACIÓN

Una vez explicado el concepto IoT y sus principales redes y protocolos, es necesario explicar los principales usos que se pueden dar a este conjunto de tecnologías.

2.1.3.1. Industria 4.0

En las últimas décadas se está produciendo una revolución tecnológica global y está cambiando la forma de vivir y actuar de la sociedad. Dentro de esta revolución tecnológica han nacido nuevos dispositivos y tecnologías que han cambiado la vida de las personas. Y está revolución también ha afectado a la industria, incorporando estas nuevas tecnologías, como puede ser la automatización y la conectividad.

El concepto Industria 4.0 nació en Alemania a finales del 2011 cuando de la mano de Roberto Bosch GmbH se creó el Grupo de trabajo Industria 4.0 que comenzó a asesorar al gobierno alemán en temas industriales. Definieron que la revolución que se estaba llevando a cabo en la industria se trata de la cuarta gran revolución industrial. Las primeras tres fueron las siguientes:

- La primera revolución industrial fue en la que se empezó a utilizar la energía del vapor para aumentar la producción
- La segunda revolución ocurrió cuando gracias a la energía eléctrica se empezó a producir en masa.
- La tercera revolución fue la que empezó a introducir la automatización industrial gracias al uso de la electrónica.

En la última revolución que se está llevando a cabo, la llamada Industria 4.0, se basa en el uso masivo de sistemas industriales conectados con sensores y actuadores, además de la interconexión entre las industrias y las interfaces abiertas para los servicios. A fin de cuentas esta revolución es la de aplicar el IoT al entorno industrial.

2.1.3.2. Smart City

Este nuevo concepto de ciudad es el llamado ciudad inteligente, en la cual emplea las nuevas tecnologías con el objetivo de mejorar la calidad de vida y accesibilidad de sus habitantes y asegurar el desarrollo de la ciudad de manera sostenible. El IoT aporta a estas ciudades el poder de adquisición de nuevos datos y de poder actuar de forma autónoma en función de estos datos. Dentro de estas ciudades inteligentes se emplean diferentes aplicaciones como la sensórica medioambiental, la eficiencia energética, la iluminación y el tráfico inteligente o la gestión de residuos urbanos de manera inteligente.

Figura 4 – Smart city

2.1.3.3. Wearable

La aparición del IoT ha permitido crear nuevos dispositivos y entre ellos se encuentran estos dispositivos personales móviles, que asisten a las personan que las emplean de la información relativa de su día a día. Estos dispositivos están pensados para llevarlos encima y se enfocan dentro del sector de la electrónica de consumo.

Figura 5 – Wearables

2.1.3.4. Smart Home

Dentro de este concepto se engloba la tecnología, normalmente la domótica, que sea capaz de comunicarse y actuar de manera autónoma. El objetivo normalmente suele ser la de ofrecer la máxima seguridad y comodidad posible al usuario, sin que sea el ahorro de gastos el mayor de los objetivos. Entre los ejemplos más claros se encuentran las aplicaciones para el control autónomo de la temperatura o el riego de plantas de manera autónoma.

Figura 6 – Smart Home

2.2. 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, además de los datos de otras fuentes como pueden ser los datos del tiempo, mapas etc. Por otro 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 SERVICE IOT

Se trata de una plataforma que sirve para conectar diferentes dispositivos a la nube de Amazon Web Service. Además permite conectar los dispositivos con otros dispositivos, trabajar sobre los datos enviados por los dispositivos conectados y crear aplicaciones para interactuar con los dispositivos.

AWS IoT permite la conexión y el envío de mensajes mediante los protocolos MQTT, HTTP y WebSockets. Para la fácil conexión Amazon proporciona un SDK, este está disponible en C y JavaScript.

Por otro lado, esta plataforma utiliza un gateway para dispositivos que permite a los dispositivos comunicarse de manera segura. Además, permite comunicaciones individuales o múltiples. Gracias a esta última comunicación es posible que un dispositivo reparta datos a más de un subscriptor a la vez.

En cuanto a la seguridad, AWS IoT utiliza una autenticación mutua y cifrada para que no ocurra una suplantación de identidad con ningún dispositivo. Para la autenticación se basa en el certificado X.509 y utiliza el método SigV4. Según que protocolo de conexión se utilice se establece un método de autenticación diferente:

- MQTT utiliza la autenticación basada en el certificado X.509

- HTTP utiliza cualquiera de los dos
- Con WebSockets se utiliza el método SigV4

Además, permite registrar cada dispositivo para así lograr que cada dispositivo tenga una identidad única.

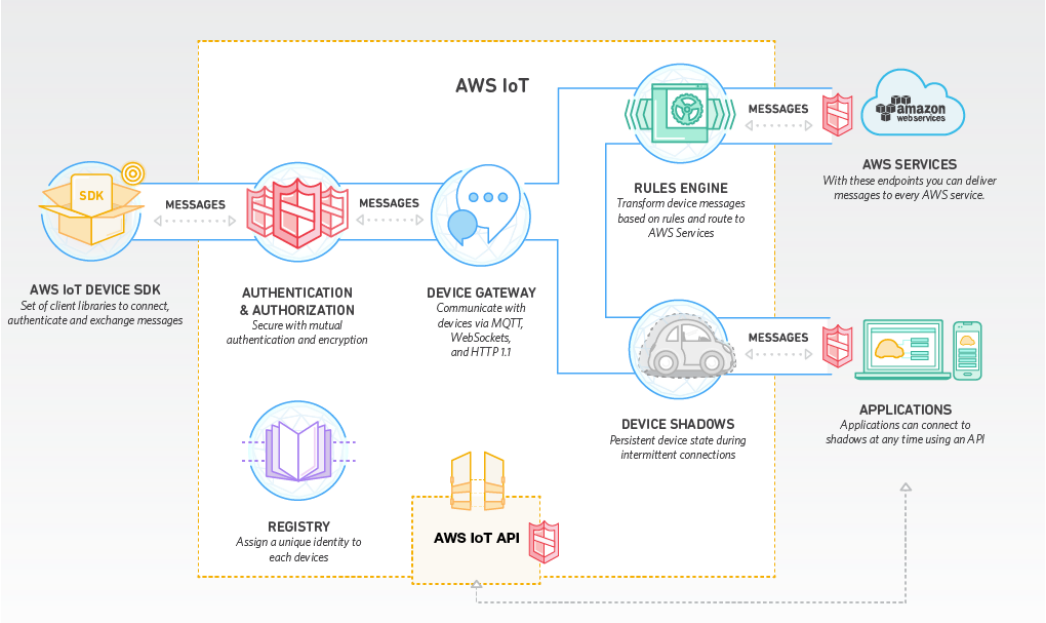


Figura 7 – Arquitectura del AWS IoT

2.2.1. Plataforma de IoT de Oracle

Oracle proporciona un servicio de IoT llamado Oracle Internet of Things Cloud Service. Este proporciona la posibilidad de conectar dispositivos en tiempo-real a la nube. Además permite el análisis de los datos enviados y la integración de los datos con otras aplicaciones. Este servicio se presenta como una plataforma como servicio (PaaS).

La conexión a la nube se puede lograr mediante el uso de librerías de cliente, software de gateway o utilizando directamente ReST API.

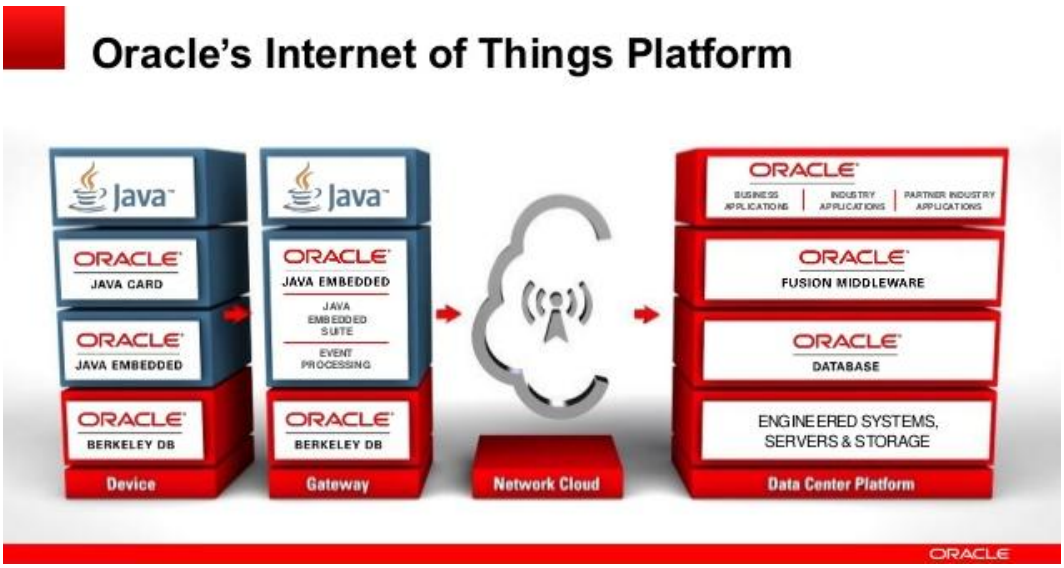


Figura 8 – Arquitectura de la plataforma IoT de Oracle

2.2.2. MICROSOFT AZURE IOT

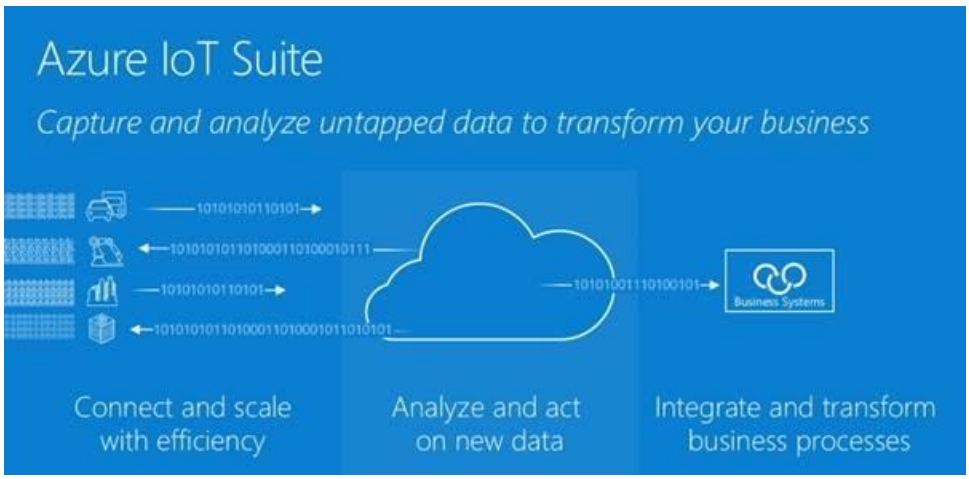


Figura 9 – Plataforma Azure IoT de Microsoft

Microsoft ofrece un conjunto de aplicaciones IoT de Azure que permiten crear escenarios de IoT desde cero o desde soluciones preconfiguradas. Microsoft propone como plataforma IoT a Azure IoT Suite. Esta plataforma basada en Azure, incluye además un servicio llamado Azure IoT Hub, que permite la comunicación fiable y bidireccional entre los dispositivos IoT y la plataforma.

Para la conexión de los dispositivos ofrece diferentes SDK de cliente en distintos lenguajes de programación (C, JavaScript, Java, C# y Python).

Entre las aplicaciones que permite crear dentro de la plataforma se encuentran las siguientes:

- Aplicaciones inteligentes (Cortana Intelligence y servicios cognitivos).
- Aplicaciones móviles (Aplicaciones nativas y multiplataforma, notificaciones push etc.).
- Aplicaciones web.
- Aplicaciones analíticas dentro del IoT (aprendizaje automático, análisis de transmisiones, predicciones analíticas etc.).

Además, ofrece dos soluciones preconfiguradas (monitorización remota y mantenimiento predictivo) que permiten al usuario no empezar desde cero.

2.2.3. OTRAS PLATAFORMAS

A parte de estas plataformas IoT es importante mencionar el proyecto Eclipse IoT. Se trata de un proyecto de código abierto en que se han implementado diferentes estándares, servicios y frameworks que han posibilitado crear un IoT abierto. Entre los estándares se encuentran MQTT, CoAP, LWM2M y OneM2M. En cuanto a los frameworks, algunos ejemplos pueden ser Eclipse NeoSCADA o Kura. Este último, ofrece servicio para IoT gateways basado en JAVA/OSGi que permite la configuración de la red y la comunicación con plataformas IoT.

3. ESTUDIO DE LA TECNOLOGÍA

3.1. ESTUDIO DEL HARDWARE SELECCIONADO

3.1.1. NI COMPACTRIO 9033

Se trata de un controlador embebido con un procesador Real-Time y una FPGA reconfigurable. Combina su procesador dual-core de 1.33 GHz con una FPGA Xilinx Kintex-7 y 4 ranuras para módulos de entrada y salida de la serie C que permiten tener un sistema de control y monitoreo de alto rendimiento. Todo ello dentro de una cubierta robusta que sin tener un ventilador permite sobrevivir en los entornos más severos.

Además, tiene diferentes opciones de conectividad gracias a sus entradas Gigabit Ethernet, dos USB de alta velocidad, un dispositivo USB y dos puertos series.

Gracias a que ejecuta el sistema operativo NI Linux Real-Time proporciona a los desarrolladores acceso al ecosistema de Linux. El sistema operativo NI Linux Real-Time se trata de un sistema operativo en tiempo real (RTOS) basado en Linux. Su mayor característica es que a pesar de mejorar su rendimiento en tiempo real no sacrifica su capacidad de uso.

Figura 10 – NI compactRIO-9033

Como se ha comentado anteriormente el dispositivo contiene dos componentes programables: un microprocesador que ejecutan un sistema operativo en tiempo real (RTOS) y una FPGA. NI compactRIO permite dividir el código a programar entre la FPGA y el RTOS según la necesidad. En la Figura 11 se puede observar la arquitectura de un hardware embebido de National Instruments como puede ser el caso de compactRIO-9033.

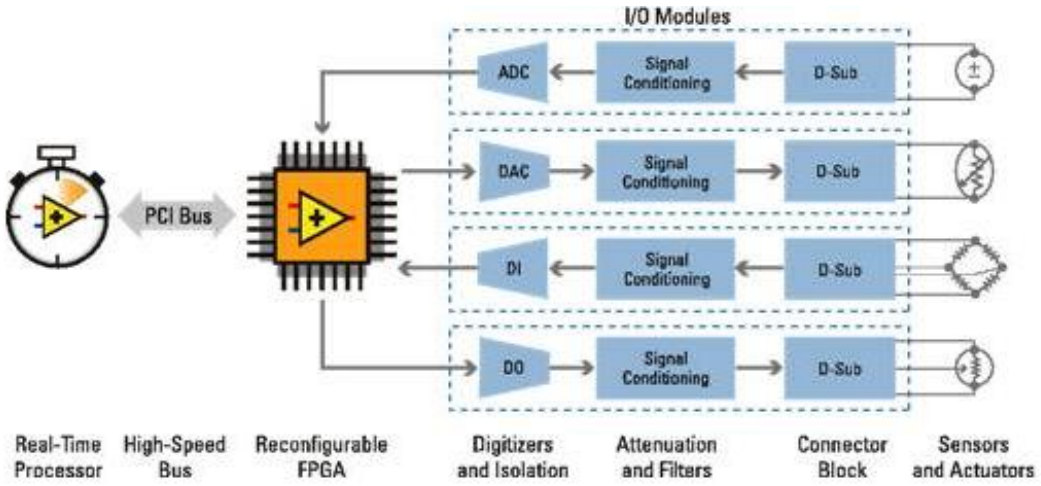


Figura 11 - Arquitectura hardware de un sistema embebido de NI

En cuanto a la programación, el usuario puede programar el RTOS usando LabVIEW Real-Time o C/C++. Mientras que para programar la FPGA se puede utilizar el módulo LabVIEW FPGA.

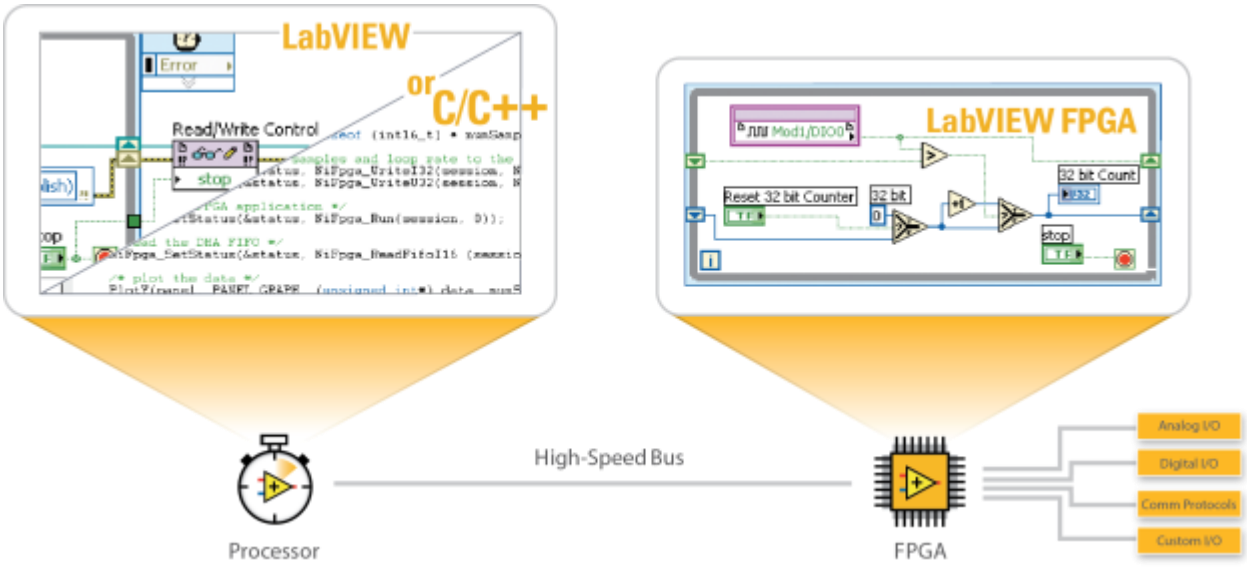


Figura 12 – Arquitectura software de un sistema embebido de NI

Existen cuatro arquitecturas de software para elegir a la hora de crear una aplicación en Tiempo Real:

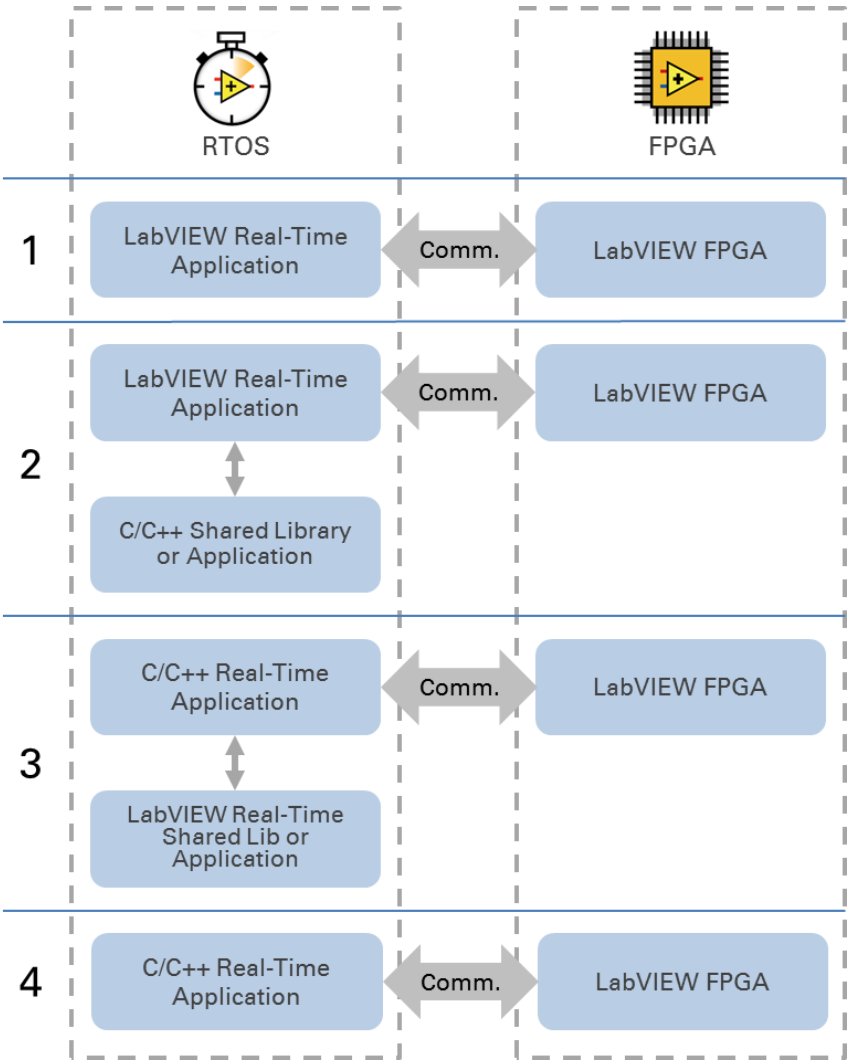


Figura 13 – Arquitecturas de diseño software de un sistema embebido de NI

1. LabVIEW para sistemas de Tiempo Real y FPGA: Permite utilizar el mismo lenguaje y entorno de programación de LabVIEW para la aplicación en tiempo real y el código FPGA gracias a los módulos LabVIEW Real-Time y LabVIEW FPGA.

2. Llamar a aplicaciones o bibliotecas C/C++ desde LabVIEW: Permite reutilizar el código de C/C++ existente al llamar a bibliotecas compartidas de C/C++ desde LabVIEW o ejecutar aplicaciones C/C++ en paralelo al ejecutable creado en LabVIEW Real-Time.

3. Llamar a bibliotecas de LabVIEW desde una aplicación de C/C++ en Tiempo Real: En los casos en los que la aplicación principal en tiempo real esté escrita en C/C++ se permite utilizar funciones de análisis de LabVIEW para procesamiento de señales al desarrollar una biblioteca compartida de LabVIEW y llamar a esta biblioteca desde el código C/C++.

4. Aplicación en Tiempo Real en C/C++: Permite desarrollar, depurar e implementar aplicaciones escritas en C/C++ utilizando cualquier entorno de desarrollo integrado y beneficiarse de LabVIEW para programar la FPGA.

3.1.1.1. Comunicación con el dispositivo

El compactRIO-9033 se puede configurar mediante la aplicación Measurement & Automation Explorer (MAX). Para ello se debe de conectar al PC mediante el puerto USB.

La conexión a la red se produce mediante un cable RJ-45 Gigabit Ethernet conectado a uno de sus dos puertos. El compactRIO inicializa una conexión a la red mediante DHCP la primera vez se conecta mediante Ethernet.

Por otro lado, para acceder al NI Linux Real-Time del compactRIO desde el PC se utiliza el protocolo SSH. Este protocolo permite la conexión segura con máquinas remotas para después ejecutar comandos o transferir archivos mediante otros protocolos asociados, como es por ejemplo, SFTP. La conexión es segura gracias al cifrado asimétrico que se utiliza para autentificar el servidor y al cifrado simétrico que se utiliza para garantizar la confidencialidad.

Para la conexión se ha utilizado el cliente SSH PUTTY que ha permitido lograr el acceso remoto al compactRIO-9033.

3.1.1.2. Módulos

Mediante las 4 ranuras que tiene el compactRIO9033 en su chasis se pueden conectar diferentes módulos de E/S que permitirán conectar a ellos diferentes sensores o dispositivos externos.

3.1.1.3. NI 9233

Se trata del módulo utilizado para conectar con NI DSA Demo Box. Contiene 4 canales de adquisición de señal dinámica para realizar mediciones. Los conectores de E/S son del tipo BNC.

Figura 14 – Módulo NI 9233

3.1.2. NI DSA DEMO BOX

Este dispositivo se trata de una caja que se separa en dos secciones: sección de audio y la sección de vibración.

La sección de audio permite modificar una señal de audio que se obtiene desde una de las entradas. La señal recogida pasa por un amplificador y después esta señal se envía a un altavoz. Además, esta caja permite añadirle distorsión a la señal antes de que se amplifique. Aparte de enviar la señal al altavoz también se puede extraer esta señal desde una de las dos salidas.

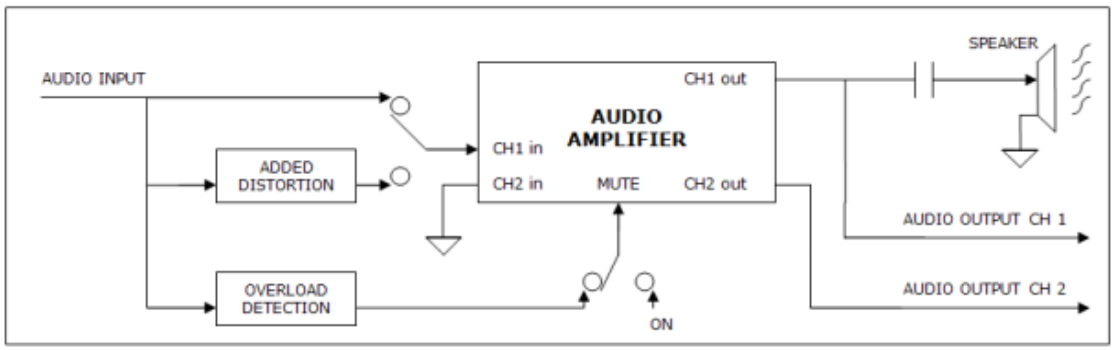


Figura 15 – Arquitectura de sección de audio del NI DSA Demo Box

La sección de vibración consta de dos ventiladores. Los dos ventiladores son exactamente iguales excepto que uno de ellos tiene un aspa rota. Además contiene un acelerómetro que devuelve el voltaje proporcional a la aceleración que detecta en las direcciones X e Y de la caja. Por otra parte, con un interruptor se puede elegir entre que ventilador de los que contiene la caja se quiere elegir para que devuelva sus valores. Por otra parte, el control del voltaje aplicado a los ventiladores se puede hacer mediante una entrada BNC o un dial.

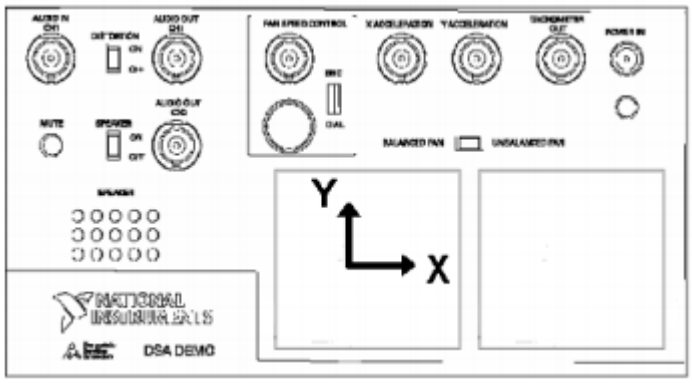


Figura 16 – Dibujo de la cara frontal del NI DSA Demo Box

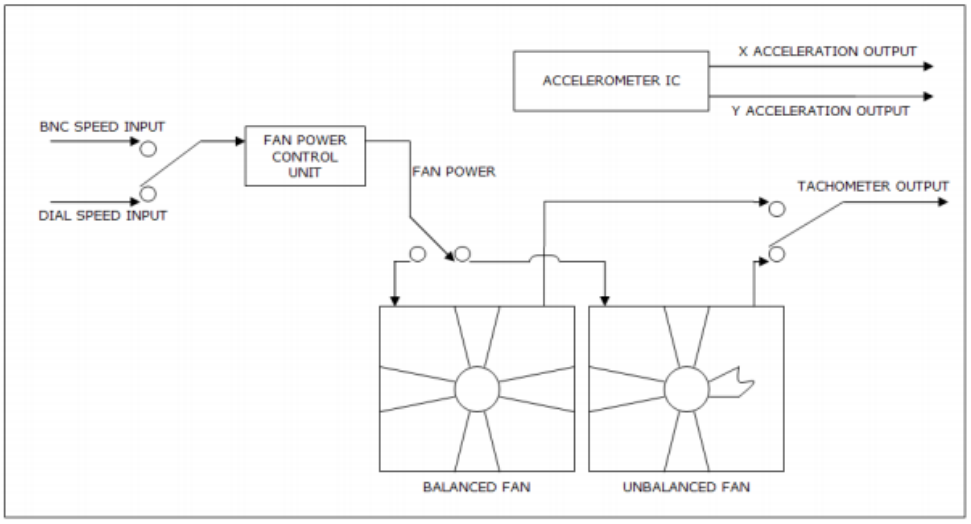


Figura 17 – Arquitectura de la sección de vibración del NI DSA Demo Box

Por lo tanto, esta sección devuelve 3 valores del ventilador que se elija: el valor del acelerómetro X, el valor del acelerómetro Y y la velocidad de giro del ventilador. Para obtener estos valores se deberán de utilizar conectores BNC como se puede observar en la Figura 18.

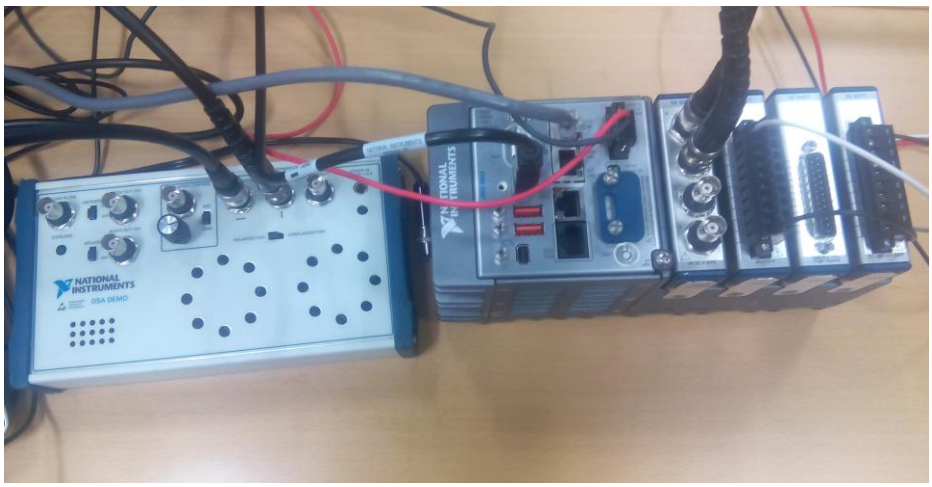


Figura 18 – Conexión entre el compactRIO-9033 y NI DSA Demo Box

3.1.3. NI MyRIO

Se trata de un controlador embebido portable pensado en un principio para la enseñanza pero que también tiene su uso en la industria. Es una herramienta pensada para implementar diferentes conceptos de diseño con un solo dispositivo.

Contiene 3 conectores de E/S reconfigurables, conexión inalámbrica y acelerómetro interno. Además, al igual que la compactRIO-9033, dispone de un procesador ARM Cortex-A9 en tiempo real dual-core y una FPGA Xilinx Z-7010. Por otro lado, también ejecuta un sistema operativo NI Linux Real-Time.

Figura 19 – NI myRIO

3.1.3.1. Comunicación con el dispositivo

NI myRIO se puede configurar mediante Measurement & Automation (MAX). En esta configuración se puede definir la red Wi-Fi a la que se conecta el dispositivo.

Por lo tanto, hay dos maneras de conectarse al dispositivo: mediante el cable USB o mediante la conexión inalámbrica. Y estas dos opciones son las disponibles para acceder al NI Linux Real-Time del myRIO. Además, mediante el protocolo SSH y el cliente SSH PUTTY se ha logrado acceder de manera remota al myRIO.

3.2. Estudio de la plataforma IoT seleccionada: IBM Watson IoT Platform

Al principio de este proyecto se detallaba como requisito el uso de la plataforma IoT de IBM Watson IoT Platform. Dicha plataforma es parte de la solución que propone IBM para dotar a los dispositivos de funcionalidades de conectividad y agregación de datos.

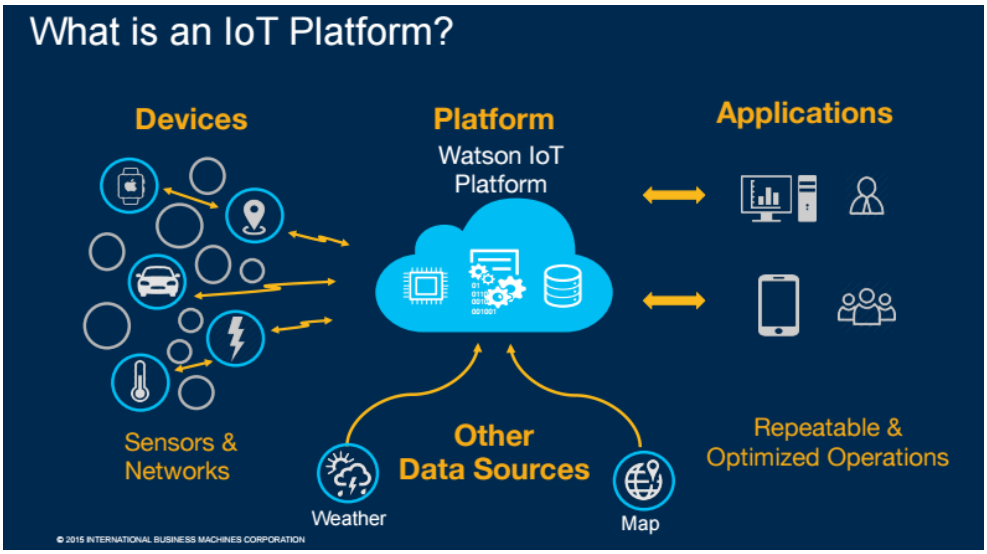


Figura 20 – Idea general del Watson IoT Platform

Watson IoT Platform se trata de un servicio completamente gestionado alojado en la nube, que además está diseñado para simplificar y derivar el valor de los dispositivos IoT. Esto es, trabaja como una centralita de todo lo relacionado a IoT dentro de IBM. Permite configurar y gestionar los dispositivos conectados a él para poder permitir que las aplicaciones creadas y enlazadas a los dispositivos puedan tener acceso a los datos corrientes e históricos. Además ofrece diferentes APIs seguros para vincular las aplicaciones con los datos procedentes de los dispositivos conectados a la plataforma. Para crear estas aplicaciones, IBM ofrece el acceso a su plataforma cloud híbrida llamada IBM Bluemix.

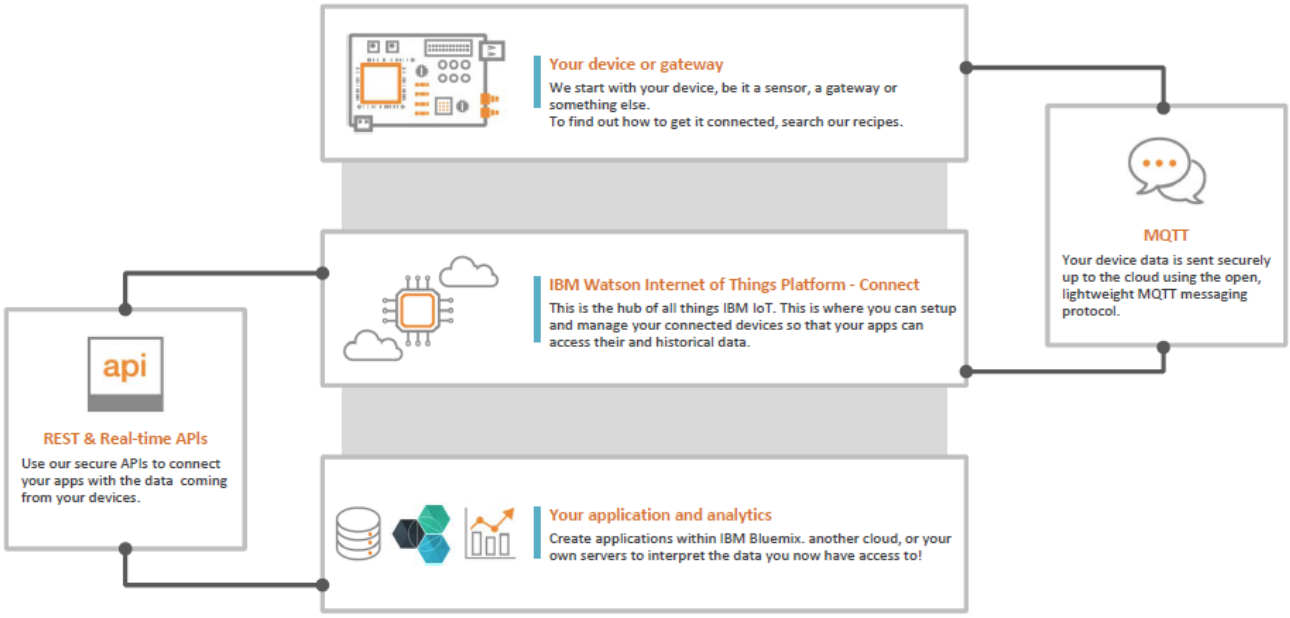


Figura 21 - Arquitectura de la conexión de un dispositivo a Watson IoT Platform

Por otra parte, IBM ha agrupado la plataforma Watson IoT en cuatro grandes bloques y que componen todo lo necesario para crear soluciones IoT:

- IBM Watson IoT Platform Connect

  Este bloque engloba la comunicación entre los dispositivos y la nube mediante los protocolos MQTT y HTTPS, permitiendo la conexión y el manejo de cualquier dispositivo IoT.

- IBM Watson IoT Platform Information Management

  Esto aporta la capacidad de integración de los datos de terceros, como puede ser la información del tiempo, además del almacenamiento de los datos de los dispositivos IoT.

- IBM Watson IoT Analytics

  Provee a la plataforma IoT de la capacidad de hacer análisis predictivo, cognitivo, análisis en tiempo real y análisis contextual.

- IBM Watson IoT Risk Management

  Provee de la seguridad necesaria a las soluciones IoT ante diferentes anomalías en el mundo del IoT

Figura 22 – IBM Watson IoT Platform

Entre los elementos principales de la plataforma Watson IoT de IBM se encuentran el acceso a las herramientas analíticas y la integración con la plataforma IBM Bluemix.

3.2.1. CONCEPTOS GENERALES

Para entender mejor el funcionamiento de la plataforma se detallan a continuación los conceptos generales.

Organizaciones

Cuando se registra un dispositivo en el Watson IoT Platform se le asigna automáticamente una ID de una organización. Se trata de un identificador único de 6 caracteres. Las organizaciones se aseguran de que los datos son solo accesibles desde nuestros dispositivos y aplicaciones. Una vez registrado el dispositivo, este y las claves API están atadas a una sola organización. Cada organización es una instancia de IoT Platform dedicada para nuestro uso.

IBM Watson IoT Platform ofrece una organización especial pública llamada Quickstart que sirve para hacer un test inicial del dispositivo conectado.

Dispositivo

Un dispositivo puede ser cualquier cosa que sea capaz de conectarse a la red y enviar o recibir datos de la nube. Cada dispositivo tendrá un identificador particular para conectarse a Watson IoT Platform, como puede ser una dirección MAC.

El dispositivo tiene que ser registrado en una Organización antes de ser conectado, quitando el caso del "Quickstart" que no es necesario registrarse. Los dispositivos registrados se identificarán en el Watson IoT Platform con un único identificador para el dispositivo y un Identificador de autenticación único que solo lo podrá utilizar ese dispositivo.

Dispositivo gestionado

Los dispositivos gestionados son definidos como dispositivos que contienen un agente gestionado. Este agente es un conjunto de lógicas que permiten al dispositivo conectarse al servicio Device Management de Watson IoT Platform vía protocolo Device Management. Por otro lado, los dispositivos no gestionados son aquellos que no disponen de agentes gestionados. Estos dispositivos pueden seguir conectándose al Watson IoT Platform, pero no podrán recibir ni operaciones ni solicitudes de dispositivos gestionados.

Aplicación

Una aplicación es cualquier cosa que se conecte a internet y que quiere interactuar con el dato enviado desde el dispositivo y/o que quiere controlar el comportamiento de este dispositivo

de alguna manera. Cada aplicación se identifica ante el Watson IoT Platform con una clave API y una ID de aplicación única.

Las aplicaciones no necesitan registrarse antes de que se puedan conectar a Watson IoT Platform pero tienen que tener una clave API válida que tiene que ser previamente registrada.

Dispositivo gateway

Los gateways son un tipo de dispositivo especial. Combinan capacidades de una aplicación y un dispositivo permitiéndose actuar como un punto de acceso. Permiten la conectividad a otros dispositivos que no tienen la habilidad de conectarse directamente a la plataforma.

Los gateways pueden registrar nuevos dispositivos y pueden mandar y recibir datos de parte de los dispositivos conectados a ellos. Los gateways tienen que ser registrados antes de poder ser conectados a un servicio.

Eventos y comandos

Eventos son los mecanismos que cada dispositivo utiliza para publicar los datos en Watson IoT Platform. Son los dispositivos los que controlan el contenido de los eventos y asigna un nombre a cada evento enviado.

Cuando un evento enviado desde un dispositivo es recibido por el Watson IoT Platform las credenciales de la conexión sirven para identificar el dispositivo desde el cual se ha mandado el evento. Así es imposible que un dispositivo se haga pasar por otro.

Los comandos son mecanismos que cada aplicación utiliza para comunicarse con los dispositivos. Solamente las aplicaciones pueden mandar comandos, quienes solo pueden ser mandados a unos específicos dispositivos.

3.2.2. PROTOCOLOS DE COMUNICACIÓN

En cuanto a la publicación de mensajes con la plataforma Watson IoT se dispone de diferentes opciones a la hora de elegir el protocolo de comunicación:

3.2.2.1. MQTT

MQTT es un protocolo de transporte de mensajería tipo Cliente Servidor de publicación y subscripción. Es utilizado por los dispositivos para comunicarse con la plataforma IBM Watson IoT. Además, es utilizado por las aplicaciones para enviar y recibir datos en tiempo real a la

plataforma. Se trata de un protocolo creado especialmente para su uso en el IoT, ya que su flujo de datos está optimizado para la reducción del tráfico de la red.

Trabaja encima de TCP/IP o encima de otros protocolos de comunicación que proveen orden, no pérdidas y conexiones bidireccionales.

Utiliza un patrón de envío y suscripción de mensajes que provee una distribución de mensajes de uno a muchos y una separación de las aplicaciones. Su objetivo es ofrecer un servicio de publicación/suscripción a fuentes de datos de forma muy sencilla y pensada en implementaciones ligeras donde el HTTP y los servicios web resultan muy pesados.

Además de utilizar el protocolo MQTT para conectar con Watson IoT Platform también se puede utilizar MQTT sobre WebSockets para enviar mensajes a Watson IoT Platform.

Este protocolo utiliza dos diferentes puertos: el puerto 1883 para la conexión de un cliente desencriptado y el puerto 8883 para la conexión de un cliente encriptado. Al utilizar el modo desencriptado todos los envíos que se hacen desde los dispositivos se hacen en texto plano, incluyendo las credencias de autenticación del dispositivo.

MQTT define tres niveles de calidad de servicio (QoS). Estos niveles definen el nivel de insistencia que los clientes o brokers de MQTT hacen para asegurarse de que los mensajes han sido recibidos.

Los tres niveles de QoS son:

- QoS 0(Como mucho una vez): El cliente o servidor entrega el mensaje una vez sin esperar a la confirmación.
- QoS 1(Como poco una vez): El cliente o servidor entrega el mensaje al menos una vez requiriendo la confirmación.
- QoS 2(Exactamente una vez): El cliente o servidor entrega el mensaje una vez utilizando un protocolo de 4 pasos.

Por otra parte, la topología de MQTT se basa en un Cliente MQTT y un Servidor MQTT. Para comunicarse entre ellos utilizan los paquetes de control. Estos paquetes se detallan en la Tabla 2.

Paquete de control	Dirección	Descripción
CONNECT	De cliente a servidor	Petición del cliente para conectarse al servidor
CONNACK	De servidor a cliente	Respuesta de la petición de conexión
PUBLISH	De cliente a servidor o de servidor a cliente	Publicación del mensaje de aplicación
PUBACK	De cliente a servidor o de servidor a cliente	Publicación de la respuesta (QoS 1)
PUBREC	De cliente a servidor o de servidor a cliente	Publicación de la respuesta a un mensaje PUBLISH (QoS 2 parte 1)
PUBREL	De cliente a servidor o de servidor a cliente	Publicación de la respuesta a un paquete PUBREC (QoS 2 parte 2)
PUBCOMP	De cliente a servidor o de servidor a cliente	Publicación de la respuesta a un paquete PUBREL (QoS 2 parte 3)
SUBSCRIBE	De cliente a servidor	Solicitud de la petición de subscripción del cliente
SUBACK	De servidor a cliente	Respuesta de la petición de subscripción
UNSUBSCRIBE	De cliente a servidor	Petición de la cancelación de subscripción
UNSUBACK	De servidor a cliente	Respuesta a la petición de cancelación de subscripción
PINGREQ	De cliente a servidor	PING de petición
PINGRESP	De servidor a cliente	PING de respuesta
DISCONNECT	De cliente a servidor	Desconexión del cliente

Tabla 2 – Paquetes de MQTT

Las cabeceras de estos mensajes son lo más pequeñas posibles. Estas cabeceras se reparten en dos partes: cabecera fija y variable. Se diferencian en que todos los mensajes de control tienen cabeceras fijas de 2 bytes pero no cabeceras variables. Estas últimas suelen contener el identificador del paquete. A parte de la cabecera, un mensaje de control también puede contener una carga. Esta carga puede ser de hasta 256 MB. Al ser las cabeceras de pequeño tamaño este protocolo es apropiado para IoT debido a la pequeña cantidad de datos trasmitido a través de redes restringidas.

Por otro lado, cuando el cliente MQTT se conecta a un servidor MQTT puede crear un tema y publicar los mensajes en ese tema al subscribirse a él.

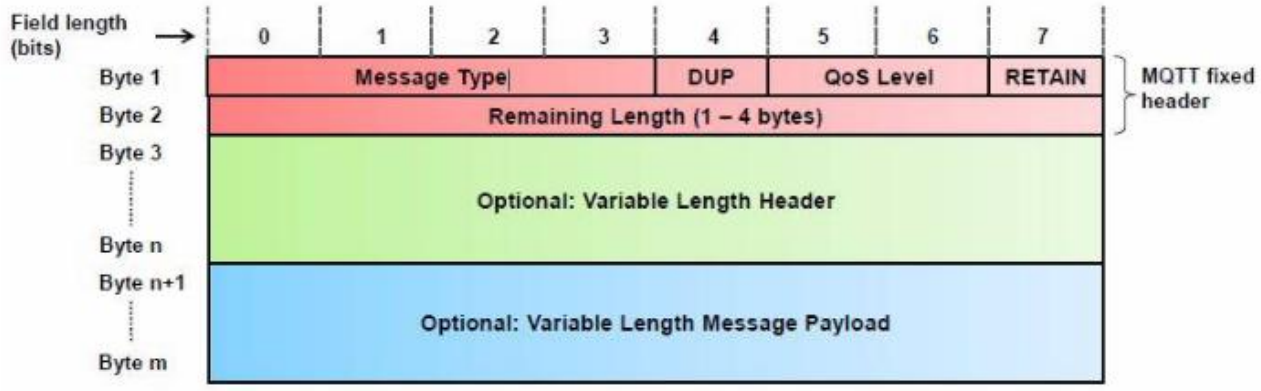


Figura 23 – Mensaje de MQTT

Para efectuar la conexión a la plataforma mediante el protocolo MQTT hay que tener en cuenta los siguientes conceptos:

Conexión del cliente:

Para la conexión mediante MQTT de un dispositivo a una organización dentro de Watson IoT Platform se puede utilizar uno de los tres siguientes puertos:

- Puerto 1883: Se envían los datos de manera no segura. La información del dispositivo y las credenciales de autenticación se envía en texto plano.
- Puerto 8883: Se envían los datos de manera segura. La conexión se encripta con TLS v1.2.
- Puerto 443: Puerto que se utiliza para WebSockets. La conexión es segura gracias a la encriptación de la conexión.

Identificador del cliente MQTT:

Cada dispositivo debe de autenticarse utilizando un ID de cliente en el siguiente formato:

```
d:org_id:type_id:device_id
```

- **d:** Identifica que el cliente es un dispositivo.
- **org_id:** la ID única de la organización. Esta se logra al registrarse por primera vez en la plataforma.
- **type_id:** es el identificador del tipo de dispositivo que se conecta.
- **device_id:** debe de ser un identificador único que diferencia el dispositivo entre todos los dispositivos.

Autenticación en MQTT:

- Nombre del usuario: El servicio solo soporta la autenticación de los dispositivos basado en identificadores.
- Contraseña: se utilizan como contraseña el identificador de autenticación del dispositivo.

Formato de los mensajes:

Todos los mensajes son enviados en formato JSON. Estos mensajes se separan en dos tipos:

- Request:

```
{     "d": {...},
      "reqId": "b53eb43e-401c-453c-b8f5-94b73290c056" }
```

  - **d:** contiene los datos relevantes en la solicitud.
  - **reqId:** es el identificador de la solicitud.

- Response:

```
{     "rc": 0,

      "message": "success",

      "d": {...},

      "reqId": "b53eb43e-401c-453c-b8f5-94b73290c056"}
```

- **rc:** es el código del resultado de la solicitud original.
- **message:** es un elemento opcional que describe el código de la respuesta.
- **d:** es un elemento de datos opcional que acompaña a la respuesta.
- **reqId:** es el identificador de la solicitud original. Será el mismo que el de la petición.

3.2.2.2. HTTP(S)

Es otra opción para poder conectar con Watson IoT Platform aparte del protocolo MQTT. Es ideal en el caso de que nuestro dispositivo esté atrapado detrás de un firewall que solo permite comunicarse mediante el puerto 80.

Los eventos de HTTP son seguros gracias al uso de TLS y el envío mediante HTTPS.

El envío de eventos se hace mediante una HTTP API. Mediante el uso de la API los dispositivos pueden hacer diferentes peticiones pero siempre utilizando cabeceras de autorización en los mensajes. En estas cabeceras deben contener las credenciales que autentifiquen al dispositivo. Además en una petición HTTP se debe de enviar una cabecera que especifique el tipo de contenido del mensaje. En la Tabla 3 se detallan los diferentes tipos de soportados.

Cabecera del tipo de contenido	Formato
Text/plain	Texto
Application/json	JSON
Application/xml	XML
Appication/octet-stream	Binario

Tabla 3 – Tipos de contenido que soporta HTTP(S)

En cuanto a la calidad el servicio, el protocolo HTTP(S) utiliza el tipo de entrega llamado “como mucho una vez” equivalente al QoS 0 del protocolo MQTT.

Pero por otro lado, no se debe de olvidar que este protocolo no está creado especialmente para el IoT, por lo que su consumo de batería al enviar los mensajes y el tamaño de los paquetes es bastante elevado.

3.2.3. LA SEGURIDAD EN IBM WATSON IOT PLATFORM

El IoT ofrece grandes oportunidades para los negocios y para los clientes, especialmente en el área de la sanidad, transporte y logística. Con estas nuevas oportunidades en estas áreas los desarrolladores se han enfrentado al reto de crear aplicaciones IoT lo suficientemente seguras como para no poner en riesgo todos los datos sensibles que se utilizan en estas determinadas áreas. Más aun cuando en los últimos meses se han reportado muchas brechas de seguridad en soluciones IoT.

Por lo tanto, los desarrolladores de soluciones IoT se deben de centrar en diferentes aspectos de seguridad. Para lograr la mayor seguridad posible en las aplicaciones IoT creadas estas deben de:

- Prevenir las posibles brechas del sistema:

  La aplicación debe de implementar de manera preventiva medidas eficaces para mantener a los atacantes alejados.

- Soportar una monitorización continuada:

  Debido a que hasta el sistema más seguro ofrece alguna vulnerabilidad, la aplicación creada debe de soportar una monitorización continuada y una actualización constante para prevenir cualquier ataque.

- Ser resistente:

  La aplicación debe de ser lo suficientemente resistente como para poder ser capaz de minimizar los daños y recuperarse en el caso de que ocurriese algún ataque.

Pero a la hora de desarrollar las aplicaciones IoT se deberán de tener en cuenta las limitaciones que ofrecen la mayoría de las veces los dispositivos IoT. En la mayoría de los casos estos dispositivos tendrán la fuerza de computación limitada, además de tener la capacidad de memoria limitada dificultando la implementación de algoritmo complejos de encriptación. Pero gracias a los nuevos protocolos seguros de comunicación especialmente creados para IoT, como Message Queuing Telemetry Transport (MQTT) y Contrained Application Protocol (CoAP), se ha logrado avanzar en el tema de la seguridad a la hora de la comunicación.

La seguridad en las soluciones IoT se puede dividir en tres niveles:

- Nivel de dispositivo o Gateway

  Se debe de proteger ante supuestos servidores falsos que envían falsos comandos o ante la intrusión de hackers que intentan escuchar los datos enviados desde los sensores privados.

- Nivel de transporte o red

  Se debe de proteger ante falsos dispositivos que envían falsas mediciones para poder corromper los datos que se envían a la aplicación.

- Nivel de aplicación

  Se debe de proteger ante el uso malicioso de los datos enviados a la aplicación y se debe de garantizar la seguridad ante la manipulación de las analíticas hechas sobre los datos.

Este último nivel es el que recibe mayores ataques y por ello la seguridad de la aplicación debe de ser una parte importante del ciclo de vida del desarrollo de software de las aplicaciones IoT.

La Figura 24 recrea los tres niveles de seguridad que utiliza una aplicación típica que utiliza IBM Watson IoT Platform para nivel de transporte y la plataforma cloud IBM Bluemix para el nivel de aplicación:

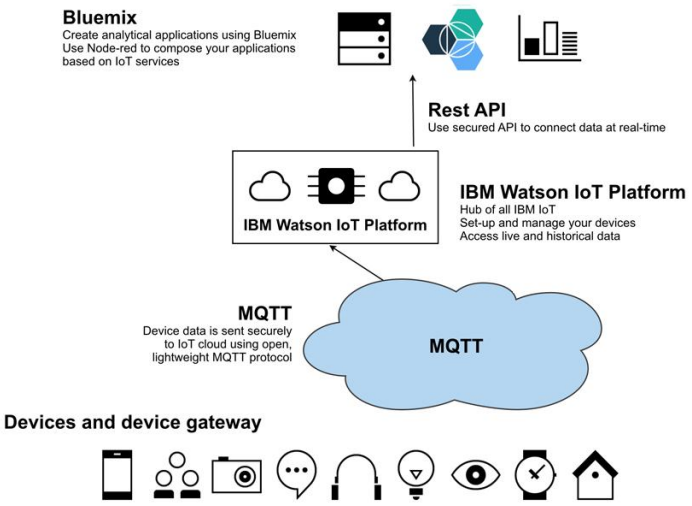


Figura 24 – Niveles de seguridad de una aplicación en la plataforma IoT de IBM

En la Tabla 4 se describe la seguridad utilizada en cada nivel.

Nivel	Descripción	Seguridad
Aplicación	Las aplicaciones IoT son desplegadas en la plataforma Bluemix	• Llamadas API seguras a IBM Watson IoT Platform • Node-Red seguro • Desencriptación de mensajes
Red/Transporte	IBM Watson IoT Platfom provee de una plataforma de mensajería basada en MQTT para las aplicaciones IoT	• Autenticación de dispositivos • Autorización • Seguridad API • Transporte y configuración segura
Dispositivos/Gateways	Los dispositivos envían directamente o a través de gateways los valores de los sensores	• Autenticación • Encriptación de la carga del mensaje • Entrega de certificado y su verificación • Transporte MQTT seguro • Firewall

Tabla 4 – Niveles de seguridad en IBM Watson IoT

Por lo tanto, la seguridad de los dispositivos y gateways se basa en su autenticación con el objetivo de garantizar el envío correcto de datos. Y para ello nació MQTT. Este popular protocolo

de mensajería como se ha comentado anteriormente está enfocado en el mundo IoT y a pesar de utilizar unos pocos mecanismos de seguridad todos ellos se enmarcan dentro de los estándares comunes de seguridad como puede ser el SSL/TLS. Gracias a este mecanismo logra que los datos se envíen encriptados y que su integridad se valide. Además, la mayoría de implementaciones de MQTT utilizan la autorización en el control de acceso al servidor de MQTT. En este caso, el TLS garantiza la autenticación del cliente ante el servidor mediante el certificado del cliente y también garantiza la autenticación del servidor ante el cliente utilizando el certificado del servidor. En cuanto a la autenticación ante una aplicación MQTT utiliza la autenticación mediante usuario y contraseña. La contraseña es conocida como identificador. Al enviar el identificador el servidor de MQTT puede comprobar la validez de la firma para así autenticar al cliente de MQTT. Es precisamente este tipo de autenticación la que utilizan las aplicaciones de IBM Watson IoT Platform. Para ello utilizan la autenticación mediante ID, clave y identificador. Estas dos últimas se logran al registrar el dispositivo en la organización en IBM Watson IoT Platform.

En cuanto a la autorización del dispositivo el mecanismo utilizado se asegura de que no haya ninguna fuga en la comunicación entre dos dispositivos. Para ello se utiliza MQTT, que está basado en la publicación y subscripción de temas. Esto es, cada mensaje enviado se publica en un determinado tema y cada subscripción contiene un filtro de temas.

En IBM Watson IoT Platform después de autenticar los dispositivos solo se les autoriza para que publiquen o se subscriban en un determinado tema. Esto permite que se eviten las suplantaciones de identidad entre dispositivos. Logrando así que solo se pueda suplantar la identidad de un dispositivo obteniendo las credenciales de seguridad (clave e identificador) creadas al registrar el dispositivo.

Por otro lado, IBM Watson IoT Platform permite la validación mediante ID de aplicación, añadiendo así un nivel extra a la seguridad entre la aplicación IoT y el dispositivo. Además con este mecanismo se logra que ninguna falsa aplicación pueda mandar comandos al dispositivo, ya que este guardará el ID de la aplicación IoT y lo validará cuando reciba comandos desde la aplicación.

En la Figura 25 se observa el mecanismo de verificación en IBM Watson IoT Platform.

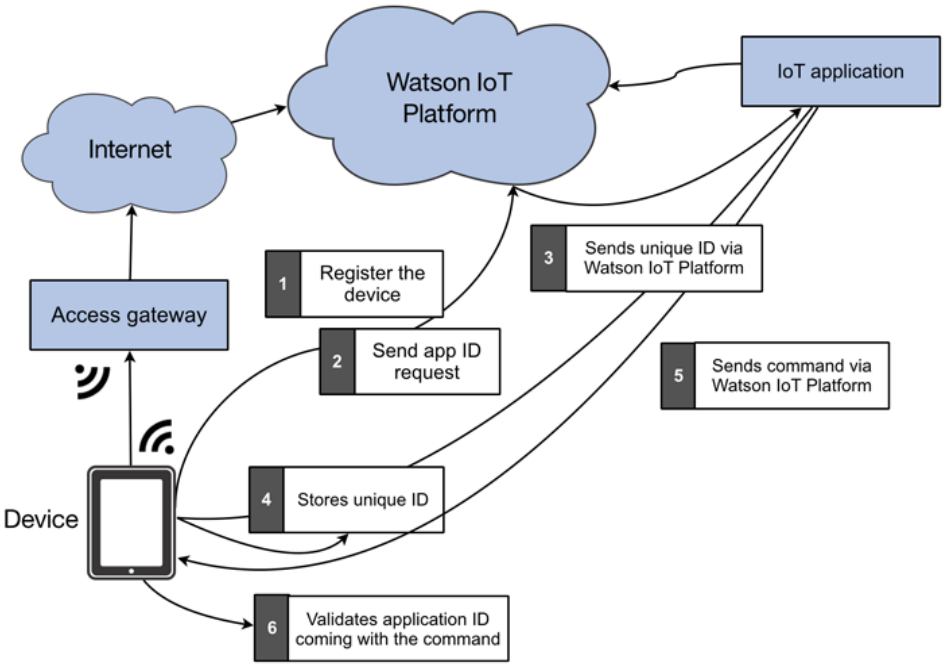


Figura 25 – Mecanismo de verificación en IBM Watson IoT Platform

En cuanto a la seguridad en el transporte IBM Watson IoT Platform soporta TLS 1.2 y utiliza el puerto 8883, logrando así eliminar el riesgo del acceso sin autorización a los mensajes.

MQTT depende de TCP como protocolo de transporte y por defecto su conexión no utiliza una comunicación encriptada. Pero utilizando TLS permite a las aplicaciones el intercambio de datos sensibles de manera segura. MQTT solo encripta la parte de carga de datos del mensaje. Esta carga se trata de los datos recibidos desde los sensores privados.

Por lo tanto MQTT certifica la seguridad utilizando TLS en la capa red y la encriptación de los datos en la capa de aplicación.

Pero la encriptación de los datos también tiene ciertas desventajas como:

- Es posible que no se pueda implementar en dispositivos con muy bajos recursos.
- A pesar de utilizar la encriptación un atacante puede modificar los datos si no se utiliza un canal de comunicación seguro.

Por otro lado, IBM Watson IoT Platform provee diferentes recursos para asegurar la seguridad al acceder a sus REST APIs. Entre ellos se encuentra la autenticación de las llamadas

API mediante una clave API que se genera en Watson IoT Platform o el uso de checksum para proteger los datos.

En la Figura 26 se define el proceso de uso de las APIs, donde el dispositivo encripta la parte de los datos del mensaje y lo manda en forma de elemento JSON. Mientras la aplicación utiliza la API de Watson IoT Platform para recibir los datos, desencripta y descodifica la parte de los datos del mensaje además de validar el checksum o la suma de verificación.

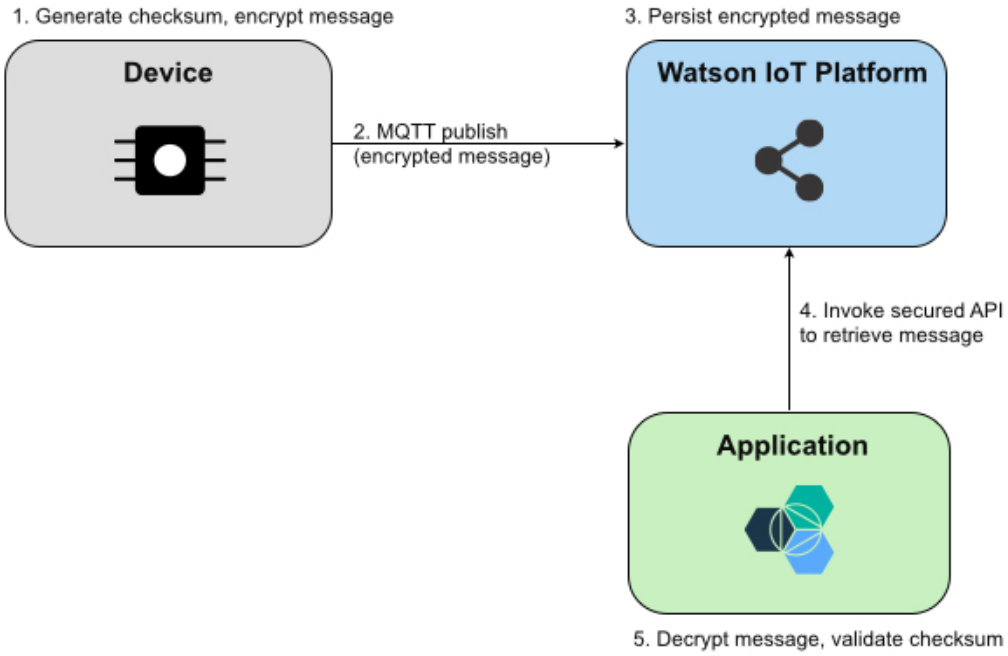


Figura 26 – Uso de una API en IBM Watson IoT

En cuanto a la seguridad en la capa de aplicación, el objetivo de las aplicaciones IoT basadas en la nube es la de asegurarse de que usuarios no autorizados no tengan acceso a los datos sensibles.

En IBM Watson IoT Platform las aplicaciones se crean mediante la plataforma cloud IBM Bluemix. En esta plataforma es posible crear diferente servicios y entre ellos está el servicio de Watson IoT Platform. Al crear el servicio se obtienen las credenciales que garantizan el acceso a IBM Watson IoT Platform.

4. DESARROLLO DEL PROYECTO

4.1. ARQUITECTURA DEL SISTEMA

En la Figura 27 se observa la arquitectura general de la solución IoT creada.

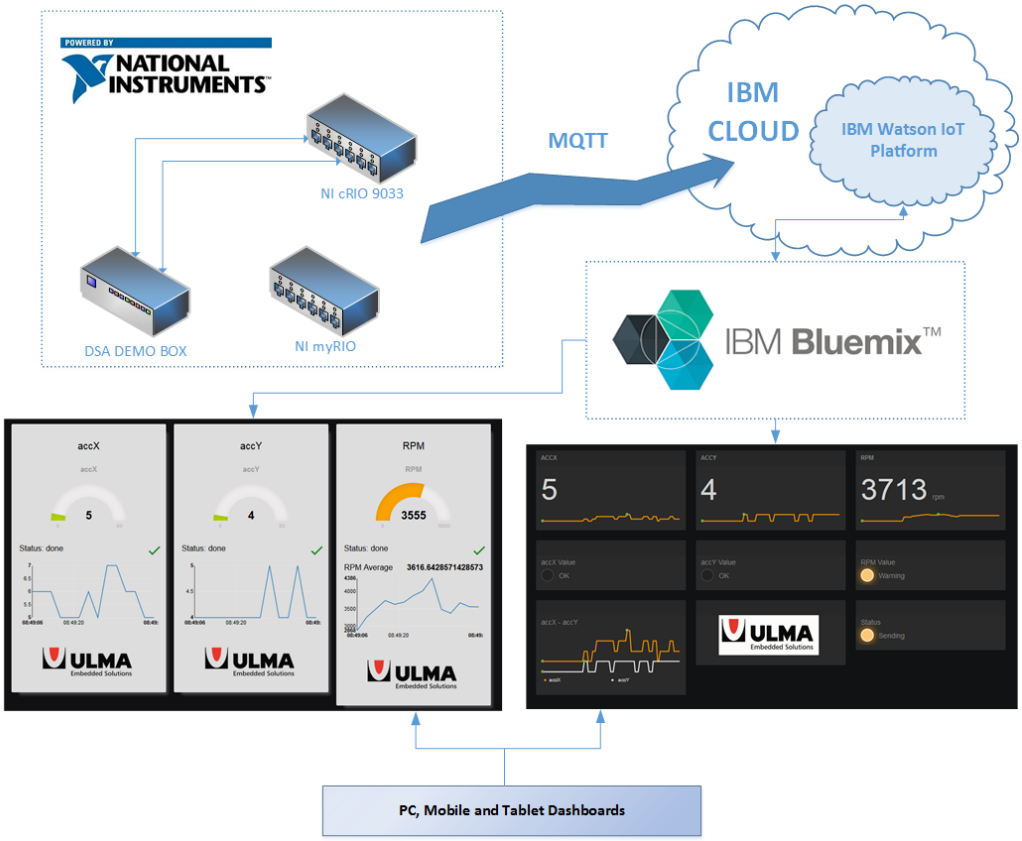


Figura 27 – Arquitectura del sistema creado

4.2. HERRAMIENTAS DE DESARROLLO

A continuación se detallan las herramientas utilizadas para desarrollar el sistema.

4.2.1. ENTORNO DE DESARROLLO SOFTWARE

4.2.1.1. Eclipse C & C++ Development Tools for NI Linux Real-Time 2014

Se trata de una herramienta de desarrollo que junta una cadena de herramientas de C/C++ con el entorno de desarrollo integrado Eclipse para programar el hardware de National Instruments que ejecuta el sistema operativo NI Linux Real-Time.

Esto es, la herramienta C/C++ Development Tools for NI Linux Real-Time Eclipse Edition permite crear aplicaciones NI Linux Real-Time en C/C++ y combinarlos con el módulo de LabVIEW FPGA. Así se logra desarrollar, depurar y desplegar aplicaciones escritas en C/C++ utilizando Eclipse añadiendo los beneficios de programar la FPGA mediante LabVIEW.

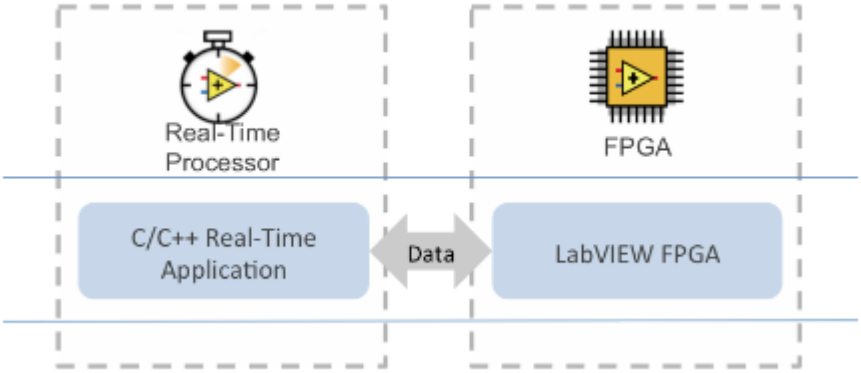


Figura 28 – Herramientas utilizadas para programar sistemas embebidos de NI

4.2.1.2. LabVIEW

Se trata de un entorno de programación para el desarrollo de aplicaciones basado en la programación gráfica. Los programas creados en LabVIEW son basados en diagramas de bloques. Mediante esta herramienta se facilita la creación de programas complejos de la manera más sencilla posible, gracias a sus paneles visuales que lo convierten en una herramienta intuitiva y fácil de aprender.

Los programas en LabVIEW se denominan instrumentos virtuales (VI). Se dividen en dos partes: panel frontal y diagrama de bloques.

- Panel frontal: Se trata de la interfaz del usuario. En ella se definen los controles e indicadores que se utilizan para conseguir observar los resultados obtenidos. Entre los controles se encuentran diferentes botones y pulsadores, mientras que entre los indicadores están los gráficos, las tablas etc. También se utilizan para trasmitir entradas y recibir salidas cuando se llama al programa VI desde un diagrama de bloques diferente.

- Diagrama de bloques: Se trata de la parte en la cual se programan las funciones que se van a ejecutar durante la ejecución del programa. En él se controlan y procesan las entradas y salidas que se crean en panel frontal, además de interconectar diferentes

terminales que representan a los controladores e indicadores. La interconexión se efectúa mediante cables que unen funciones, estructuras o terminales.

Para comunicar la FPGA programada en LabVIEW y la aplicación en tiempo real programado en C/C++ es necesario el uso de la herramienta FPGA Interface C API. Esta herramienta permite generar archivos escritos en C desde programas de LabVIEW de manera automática, por lo que facilita a los desarrolladores la programación de la FPGA mediante la herramienta gráfica de LabVIEW y el sistema operativo en tiempo real mediante entornos de desarrollo en C/C++ como Eclipse o las herramientas GNU Compiler Collection (GCC). Esto es, los programas en C/C++ creados con la herramienta FPGA Interface C API se pueden ejecutar en un procesador en tiempo-real de una compactRIO u otro dispositivo, logrando así la interacción con los Vis que a su vez se ejecutan sobre la FPGA.

4.2.1.3. IBM Bluemix

Esta herramienta es un entorno PaaS (Platform-as-a-Service) desarrollado por IBM. Permite el despliegue de aplicaciones realizados en diferentes lenguajes. Además permite enriquecer estas aplicaciones con diferentes servicios. Se basa en la tecnología Cloud Foundry y se sitúa sobre la infraestructura de SoftLayer. Entre los lenguajes de programación que soporta Bluemix se encuentra Java, PHP, node.JS, y Ruby.

Esta herramienta dispone de un amplio catálogo donde se encuentran los servicios disponibles. Entre los servicios disponibles se encuentran:

- Aplicaciones cognitivas
  - Voz a texto
  - Texto a voz
  - Reconocimiento visual
  - …
- Aplicaciones móviles
  - Notificaciones push
  - Integración con Twilio
  - Manejo de contenido de la aplicación para atraer al usuario con contenido personal
  - …

- Aplicaciones web
    - Posibilidad de crear aplicaciones en diferentes lenguajes:
        - Java
        - Node.JS
        - XPages
        - Go
        - PHP
        - Python
        - Ruby
        - Swift
        - Tomcat
    - Posibilidad de integrar las aplicaciones web con otros servicios que mejoran su uso
    - ...
- Análisis de los datos
    - Integración con diferentes base de datos
        - Cloudant NoSQL DB
        - dashDB
        - MongoDB
        - PostgreSQL
    - Análisis predictivo integrándolo con IBM SPSS Modeler
    - Apache Spark
    - …
- Aplicaciones dentro de Internet de las cosas
    - Watson IoT Platform
    - IoT Real-Time Insights

Como se ha comentado anteriormente, la combinación entre IBM Watson IoT Platform e IBM Bluemix permite crear aplicaciones que se enlazan con los dispositivos conectados. Para ello Bluemix ofrece el servicio “Internet of Things Platform”. Este servicio permite a las aplicaciones comunicarse con dispositivos conectados a Watson IoT Platform y consumir los datos que recopilan. Además ofrece una interfaz de usuario sencilla, donde se pueden añadir y gestionar dispositivos de manera fácil.

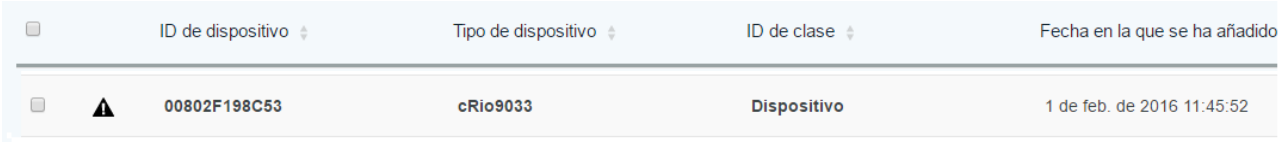


Figura 29 – Gestión de dispositivos en Watson IoT Platform

En la Figura 29 se observa el dispositivo conectado llamado cRio9033 en Watson IoT Platform.

A continuación, se detallarán diferentes servicios que ofrece IBM Bluemix y que se han utilizado en este proyecto.

4.2.1.3.1. Node Red

Se trata de una herramienta desarrollada por IBM que permite el cableado entre dispositivos hardware, APIs y servicios online de diferentes maneras. Se encuentra dentro de IBM Bluemix clasificado como un servicio.

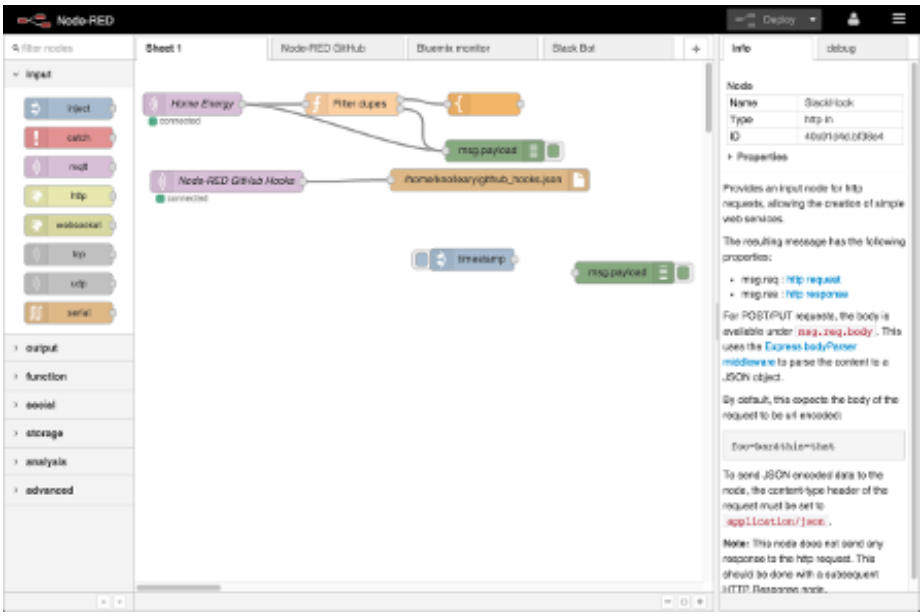


Figura 30 – Vista general de Node-Red

Node-Red provee de un editor de flujo basado en navegador que permite la conexión entre diferentes nodos y el flujo. Además posee un editor de texto que posibilita la creación de nodos con funciones en JavaScript.

Entre los diferentes tipos de nodos que ofrece Node-Red se encuentran:

- Input
  - MQTT
  - webSocket
  - IBM IoT
  - …
- Output
  - Debug
  - HTTP response

  - TCP
  - …
- Nodos de programación
  - Función (para desarrollar en JavaScript)
  - Plantillas (para desarrollar en HTML, CSS etc.)
  - Conversores de objetos a:
    - CSV
    - HTML
    - JSON
    - XML
  - …
- Almacenamiento (envío de datos a diferentes bases de datos)
  - MongoDB
  - CloudantDB
  - DashDB
  - …
- IBM Watson
  - Análisis de lenguaje
  - Transformación de voz a texto
  - Reconocimiento visual
  - …

4.2.1.3.2. IoT Real-Time Insights

Se trata de otro servicio dentro de IBM Bluemix. Se sitúa dentro del IoT y permite realizar analíticas de datos en tiempo real. Para ello se basa en el uso de reglas simples y una infraestructura ampliable que permite aprovechar los datos enviados desde los dispositivos. Además ofrece un enriquecimiento de los datos gracias a que permite detectar el mal

funcionamiento del dispositivo y crear acciones que se ejecutan de manera automática. Permite enlazar las reglas y las acciones. Entre las acciones se encuentran:

- Envío de correos electrónicos: Permite el envío de emails a diferentes destinatarios cuando una regla se ha cumplido.
- Enlace con Node-Red: Con esta acción permite conectarse a una aplicación de Node-Red cuando una regla se ha cumplido.
- IFTT: La acción IFTT permite poner en marcha una fórmula IFTT cuando una regla se ha cumplido.
- Webhook: Permite crear una petición HTTP a una servicio web basado en Webhook cuando una regla se ha cumplido.

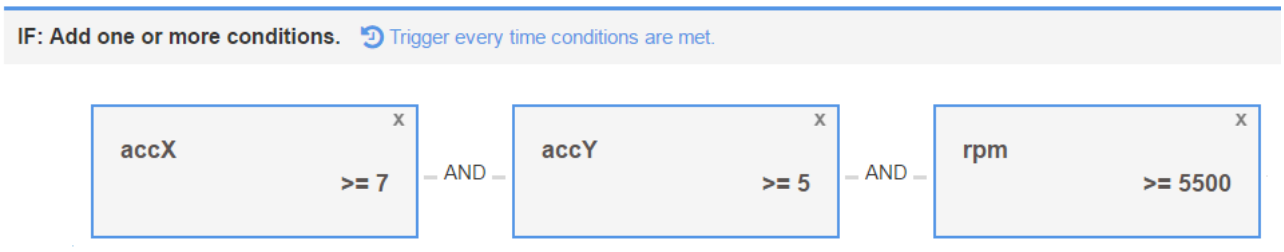


Figura 31 – Ejemplo de creación de una alerta en IoT Real-Time Insights

4.2.1.3.3. IoT Workbench

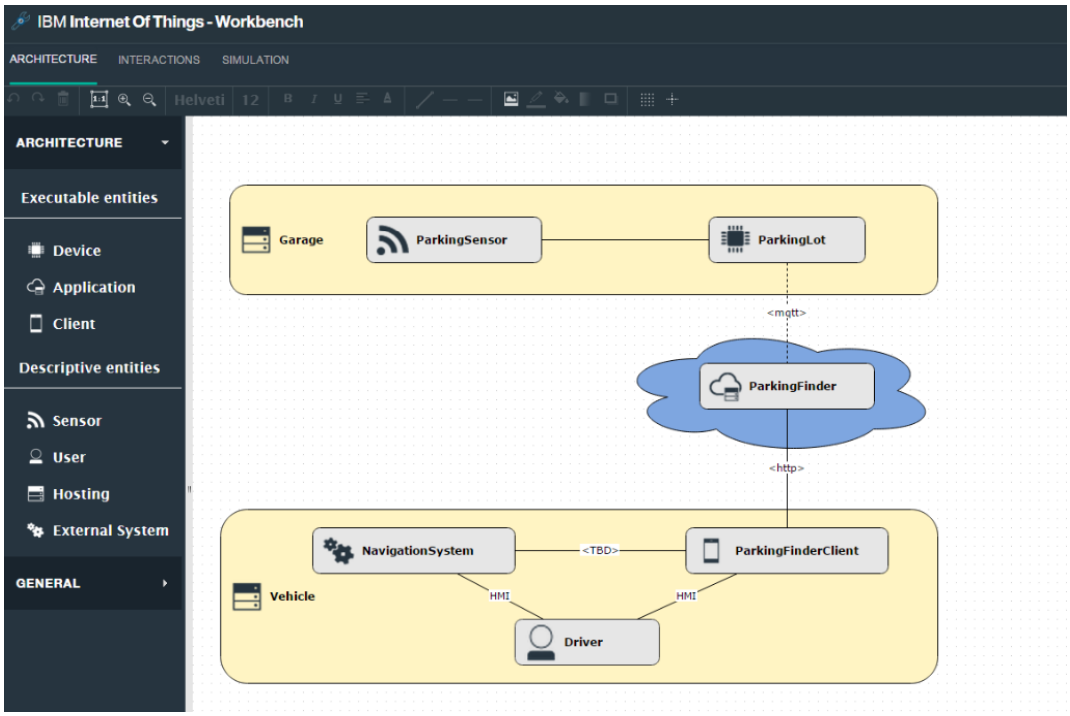


Figura 32 – Ejemplo de un diseño en IoT Workbench

Se trata de servicio experimental situado dentro de IBM Bluemix que permite un diseño y prototipado rápido y visual de un sistema IoT. Además ofrece la posibilidad de simular un sistema añadiéndole dispositivos, servicios cloud y clientes. Por lo que permite:

- Diseñar arquitecturas y escenarios para un sistema IoT.
- Permite simular el comportamiento de dispositivos.
- Generar el esqueleto de aplicaciones como base de un servicio cloud.
- Simular y probar un sistema IoT completo.

Dentro del diseño de la arquitectura de un sistema IoT se pueden añadir diferentes elementos como se observa en la Tabla 5.

Icono	Nombre	Descripción
	Aplicación	Una aplicación IoT que monitoriza y controla dispositivos.
	Cliente	Un cliente mediante el cual el usuario interactúa con la solución.
	Dispositivo	Un dispositivo embebido conectado a la red IoT.
	Sistema externo	Un sistema externo a la solución que interactúa con una o más entidades.
	Almacenamiento	Una entidad que hospeda otras entidades.
	Usuario	Un humano que utiliza los servicio que ofrece la solución

Tabla 5 – Elementos que están accesibles en IoT Workbench

Por otro lado, los dispositivos tienen diferentes atributos que representan su estado. La variación de estos atributos se puede simular para comprobar cómo cambia el estado del dispositivo.

En cuanto a la aplicación, IoT Workbench ofrece dos tipos de aplicaciones: una aplicación desde cero en Node.JS y una aplicación en Node-Red. Este elemento permite acceder a los datos de los dispositivos simulados desde Node-Red donde después se pueden enlazar con otros servicios.

4.2.1.3.4. Cloudant NoSQL DB

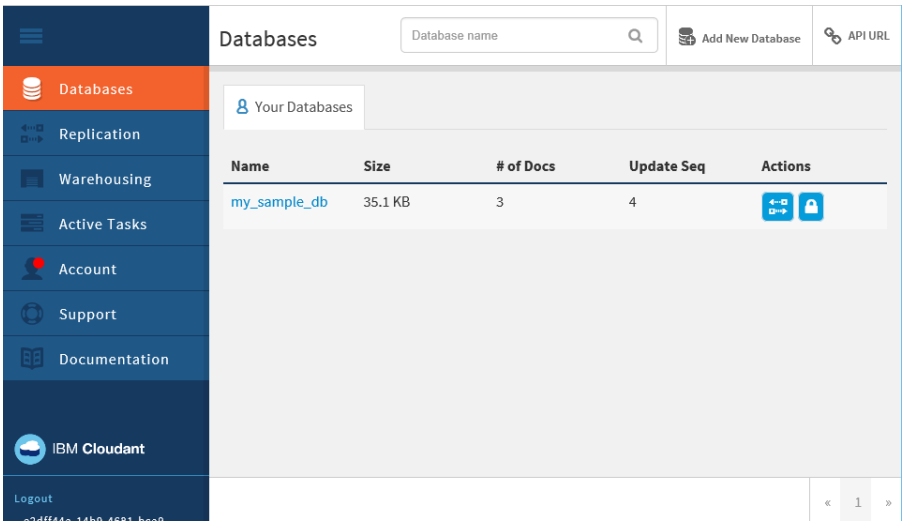


Figura 33 – Vista de panel de control de Cloudant

Se trata de un servicio basado en la nube de IBM. Es una base de datos NoSQL que es compatible con otras bases de datos como CouchDB y se puede acceder a él mediante el uso de una interfaz HTTP para aplicaciones web y móvil.

Mediante una API RESTful se puede tener acceso como JSON a través de una URL a los documentos almacenados en la base de datos. Además, ofrece un sistema robusto de acceso a los datos, permitiendo el acceso a ellos desde dispositivos móviles y aplicaciones del cliente.

IBM Bluemix ofrece un servicio de Cloudant NoSQLDB y se puede acceder a la consola de Cloudant mediante el cuadro de mando de Bluemix. Mediante servicios como Node-Red, se pueden enlazar los datos recibidos en la plataforma Watson IoT a la base de datos Cloudant.

4.3. PLANTEAMIENTO DE LA SOLUCIÓN

4.3.1. APLICACIÓN DE MONITORIZACIÓN

En este apartado se describe la aplicación creada para la monitorización del DSA Demo Box. Esta aplicación se ha desarrollado para lograr la monitorización en diferentes dispositivos mediante un navegador web del comportamiento del dispositivo DSA Demo Box.

4.3.2. FASES DE DESARROLLO DE LA APLICACIÓN

La aplicación se separa en dos grandes bloques: por un lado la fase de la conexión de los dispositivos a la plataforma Watson IoT Platform, y por otro, la fase de desarrollo de la aplicación web para llevar a cabo la monitorización.

Por lo tanto estos han sido las fases del desarrollo:

1. Desarrollo de la aplicación para la conexión de la compactRIO.
2. Modificación de la aplicación para el envío de los valores del DSA Demo Box.
3. Desarrollo de la aplicación web mediante IBM Bluemix y Node-Red.
4. Ampliación de la aplicación para llevar a cabo experimentos de mantenimiento predictivo.

4.3.3. DESCRIPCIÓN GENERAL DE LA APLICACIÓN

Como se ha explicado anteriormente la aplicación se ha separado en dos partes y a continuación se detallan cada una de esas partes.

4.3.3.1. Aplicación de conexión a la plataforma Watson IoT

Esta aplicación es la encargada de enviar los datos de los dispositivos a la nube. Se ejecuta dentro del dispositivo que se quiera conectar con la plataforma, en este caso un compactRIO-9033. La aplicación adquiere los datos del dispositivo DSA Demo Box que se conecta mediante conectores BNC a la compactRIO. Después de adquirir los datos, la aplicación envía los datos a la plataforma Watson IoT.

Implementación

Para crear la aplicación que se encarga de enviar los datos a la plataforma IBM Watson IoT se ha utilizado las herramientas LabVIEW y el entorno de desarrollo Eclipse C & C++ Development Tools for NI Linux Real-Time 2014.

La aplicación se trata de un programa ejecutable que se debe de poner en marcha en el Linux Real-Time del dispositivo que se quiere conectar con la nube.

El desarrollo de esta aplicación se ha repartido en cuatro fases:

- Lectura de datos del dispositivo.
- Creación del proyecto para llevar a cabo la conexión.
- Importación del proyecto, compilación y puesta en marcha.
- Comprobación.

Para llevar a cabo la lectura de los datos enviados desde el DSA Demo Box se ha utilizado el entorno de desarrollo LabVIEW de National Instruments.

LabVIEW permite separar los valores recibidos desde los dispositivos, esto es, los valores de los acelerómetros X e Y y los valores del tacómetro.

Para lograr separar los valores recibidos se ha creado un proyecto de LabVIEW, donde se encuentran las VIs. Para poder acceder a los datos del módulo NI 9233 se ha añadido al proyecto un FPGA target, debido a que es la FPGA la que tiene acceso a los datos. A continuación, se detalla el desarrollo de la aplicación que permite acceder a los datos.

En la Figura 34 se observa la composición del programa para poder acceder a los datos del módulo 1 del compactRIO-9033 en el cual se sitúa el NI-9233.

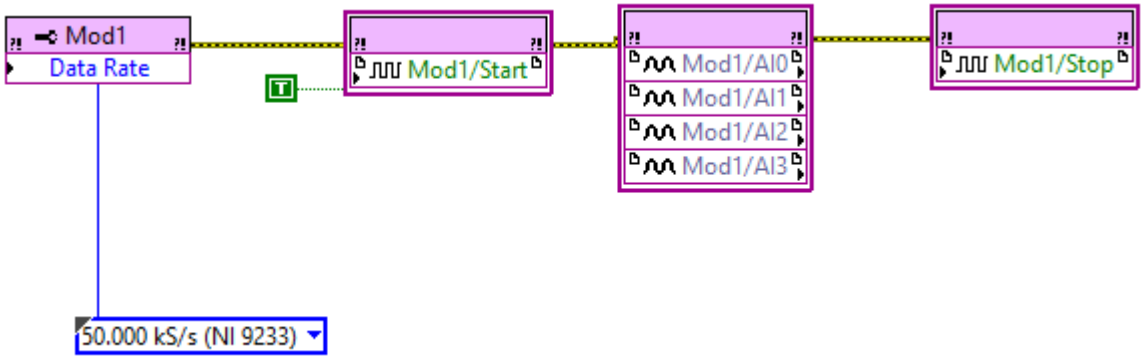


Figura 34 – Acceso a los datos del módulo

En el AI0 se encuentra el conector BNC que está conectado a la salida del acelerómetro X del DSA Demo Box:

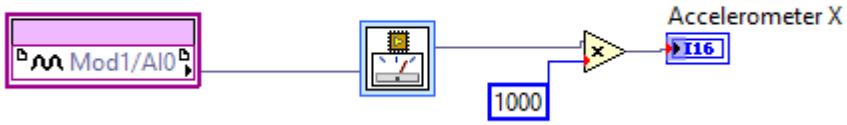


Figura 35 – Acceso a los datos del acelerómetro X

Por otro lado, en el AI1 se encuentra el conector que se comunica con el acelerómetro Y del DSA Demo Box:

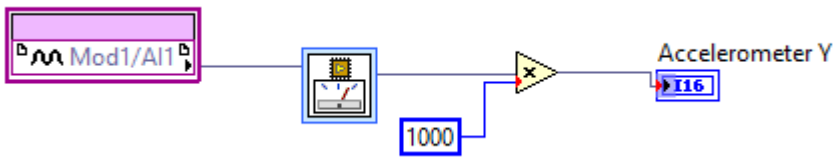


Figura 36 – Acceso a los datos del acelerómetro Y

Por último, el AI2 se conecta a la salida del tacómetro del DSA Demo Box. Pero para trasformar los datos recibidos desde el tacómetro a RPMs y así lograr su velocidad de rotación se debe de utilizar un decodificador de la señal del tacómetro.

El decodificador se reparte en dos subVIs que leen los valores del tacómetro y calculan el tiempo entre cada pulso del tacómetro.

El primer subVI se encarga de calcular el periodo del tacómetro entre dos pulsos, mientras que el segundo subVI es el encargado de verificar si se supera un umbral predefinido, con la intención de evitar tratar como señal válida el ruido en el caso de que este último se detectase. Para ello en este segundo subVI, además de predefinir un umbral, se ha incorporado la histéresis como herramienta para que el algoritmo no considere erróneamente que haya sobrepasado el umbral. El algoritmo comprueba si se ha excedido el umbral y si eso ocurre genera un evento para comprobar el nivel de histéresis.

En la Figura 37 y Figura 38 se detalla el proceso en el que se comprueba el umbral y la histéresis.

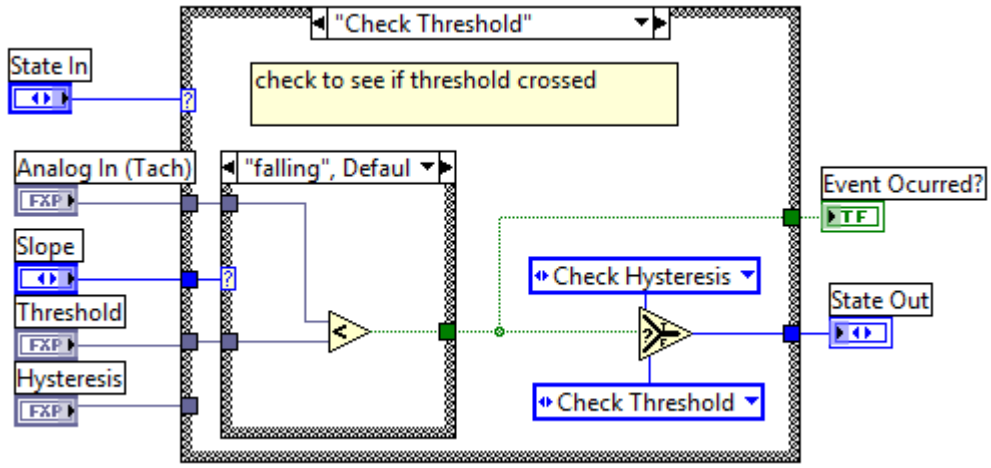


Figura 37 – Control del umbral

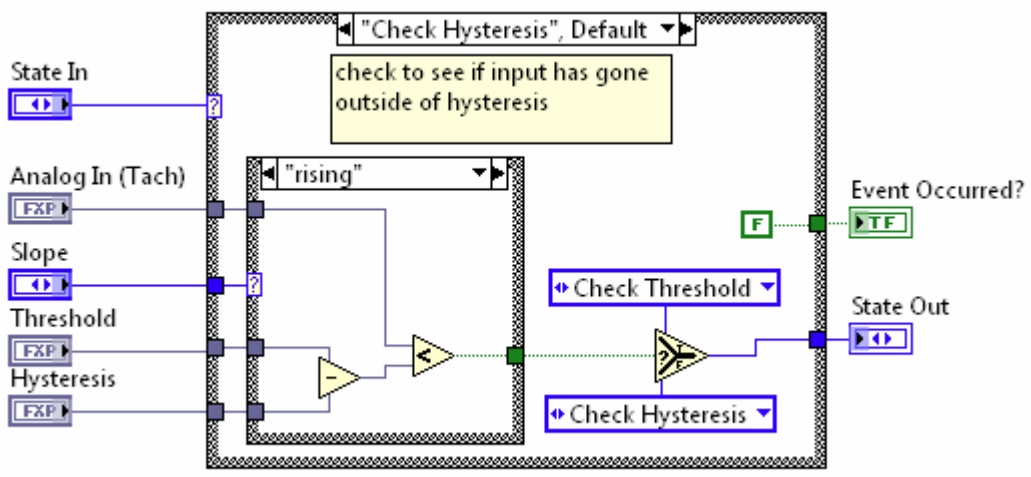


Figura 38 – Control de la histéresis

En la Figura 39 se observa el proceso para calcular el periodo de la señal del tacómetro entre dos pulsos.

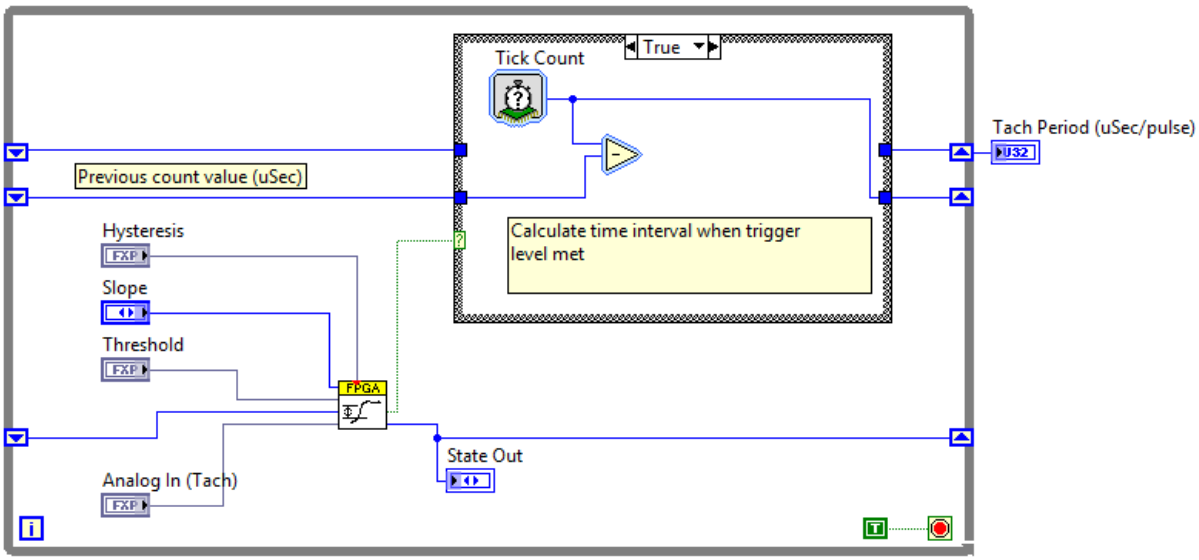


Figura 39 – Cálculo del periodo del tacómetro

Por último, el VI principal se encarga de transformar el periodo del tacómetro a revoluciones por minuto:

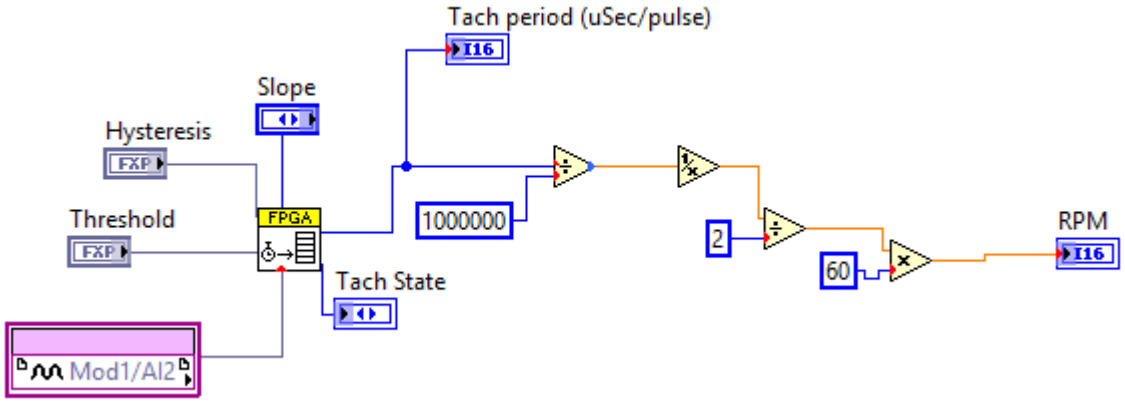


Figura 40 – Cálculo de los RPMs

En cuanto a la visualización de los datos, LabVIEW proporciona un panel frontal que permite añadir diferentes controles con la intención de visualizar los datos de la manera que el usuario quiera:

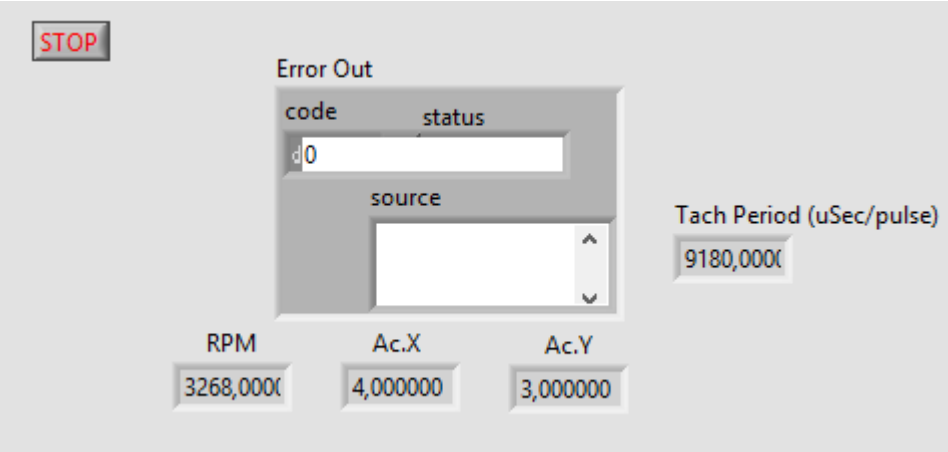


Figura 41 – Panel frontal de la aplicación

Para conseguir los ficheros en C/C++ se ha utilizado la herramienta FPGA Interface C API. Esta herramienta permite la interacción directa entre aplicaciones C/C++ con los VIs compilados de LabVIEW FPGA sin el uso de LabVIEW.

Con los ficheros creados por esta herramienta se ha creado un proyecto dentro de Eclipse C & C++ Development Tools for NI Linux Real-Time 2014. Mediante este entorno de desarrollo se ha modificado los ficheros creados por FPGA Interface C API para lograr solo los datos que se refieren a la velocidad de giro y a los acelerómetros X e Y de los ventiladores del dispositivo DSA Demo Box.

Por otra parte, como se ha comentado anteriormente IBM permite el uso de los protocolos HTTP y MQTT para el envío de datos a su plataforma. En este caso se ha decidido utilizar el protocolo MQTT, ya que se trata de un protocolo específicamente creado para el IoT y su flujo de datos está optimizado para reducir el tráfico de la red.

Para facilitar el uso de este protocolo, IBM ofrece librerías de cliente escritos en diferentes lenguajes de programación. Entre estas librerías se encuentra la librería de cliente escrita en Embedded C, que permite la comunicación con IBM Watson IoT Platform utilizando el protocolo MQTT.

Esta librería se ha importado al Linux Real-Time del compactRIO-9033 y mediante la modificación de dicha librería y la combinación con los ficheros del proyecto creado dentro de Eclipse mediante el cual se obtiene el acceso a los datos del dispositivo DSA Demo Box, se ha obtenido una única librería que al compilarla y ejecutarla se ha logrado acceder a los datos de la FPGA y mandarlos a la plataforma IBM Watson IoT Platform.

En la Figura 42 se observa la recepción en la plataforma IBM Watson IoT Platform de los datos de la velocidad de giro y de los acelerómetros X e Y.

Figura 42 – Visualización de los datos en Watson IoT Platform

4.3.3.2. Aplicación web para la monitorización

La aplicación web se encarga de visualizar los datos recibidos desde los dispositivos. Esta aplicación permite la monitorización de los datos en tiempo real.

La aplicación visualiza los datos actuales, además de los datos históricos y los datos almacenados en la base de datos. Con estos datos históricos la aplicación permite trabajar y efectuar diferentes operaciones sobre estos datos. Además de monitorizar los datos, la aplicación permite la creación de diferentes alertas para el envío de avisos al correo electrónico del usuario.

Por otra parte, este aplicación web es accesible desde cualquier dispositivo que tengo conexión a internet y se amolda automáticamente a cualquier tamaño de pantalla.

Implementación

Para crear una aplicación web que se encargue de la visualización se ha utilizado la herramienta Node-Red que está integrada dentro de IBM Bluemix.

Node-Red permite la interconexión entre los dispositivos que se han conectado a la plataforma y los diferentes nodos que actúan dentro de la aplicación web.

Se ha utilizado el paquete “node-red-contrib-ui” que permite implementar en Node-Red diferentes nodos que se utilizan en las interfaces de usuario.

Para poder recibir los datos desde IBM Internet of Things Platform se ha utilizado en Node-Red el nodo “ibmiot” que permite conectarse con esta plataforma y definir desde que dispositivo se quieren leer los datos. Este nodo recibe los eventos recibidos desde los dispositivos en formato JSON.

Figura 43 – Uso del nodo “ibmiot”

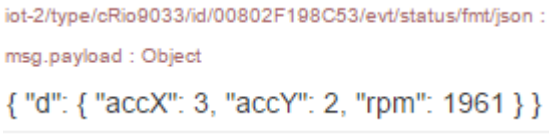


Figura 44 – Datos recibidos desde los dispositivos en Node-Red

Los datos recibidos se deben de separar en diferentes flujos de datos para poder trabajar individualmente sobre cada uno de ellos. En este caso los datos que se recibirán desde IBM Internet of Things Platform serán:

- accX: Valor del acelerómetro X del ventilador del DSA Demo Box.
- accY: Valor del acelerómetro Y del ventilador del DSA Demo Box.
- RPM: Valor en RPMs de la velocidad de giro del ventilador del DSA Demo Box.

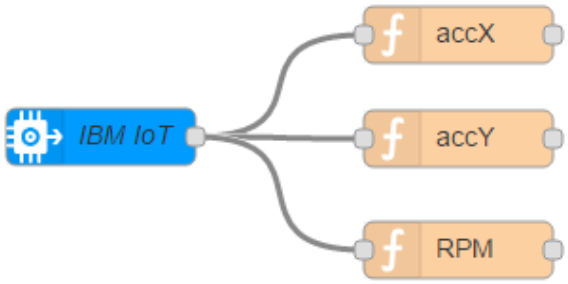


Figura 45 – Separación de los datos recibidos

En cada uno de estos nodos que se observan en la Figura 45 se escriben las funciones para separar los datos.

Todos los datos recibidos desde los dispositivos se almacenan en una base de datos de Cloudant NoSQL DB. Para poder almacenar automáticamente se ha utilizado el nodo “Cloudant” que permite almacenar los datos en una base de datos Cloudant NoSQL DB que anteriormente se haya creado en el mismo espacio de IBM Bluemix. Los datos se almacenan sumándoles la fecha y hora en la que se han generado para facilitar su posterior uso.

Figura 46 – Almacenamiento de los datos recibidos desde los dispositivos

Una vez almacenados los datos se puede trabajar sobre ellos en Node-Red. Para poder tener acceso a los datos se ha utilizado el nodo de “cloudant in” que permite buscar en una base de datos los documentos.

Figura 47 – Uso del nodo “cloudantIn”

Por otra parte, con el objetivo de simular el sistema que se ha llevado a cabo y para poder observar el posible comportamiento del sistema se ha utilizado la herramienta Internet of Things Workbench dentro de IBM Bluemix.

Con esta herramienta se logran simular todos los dispositivos reales que no se disponen físicamente.

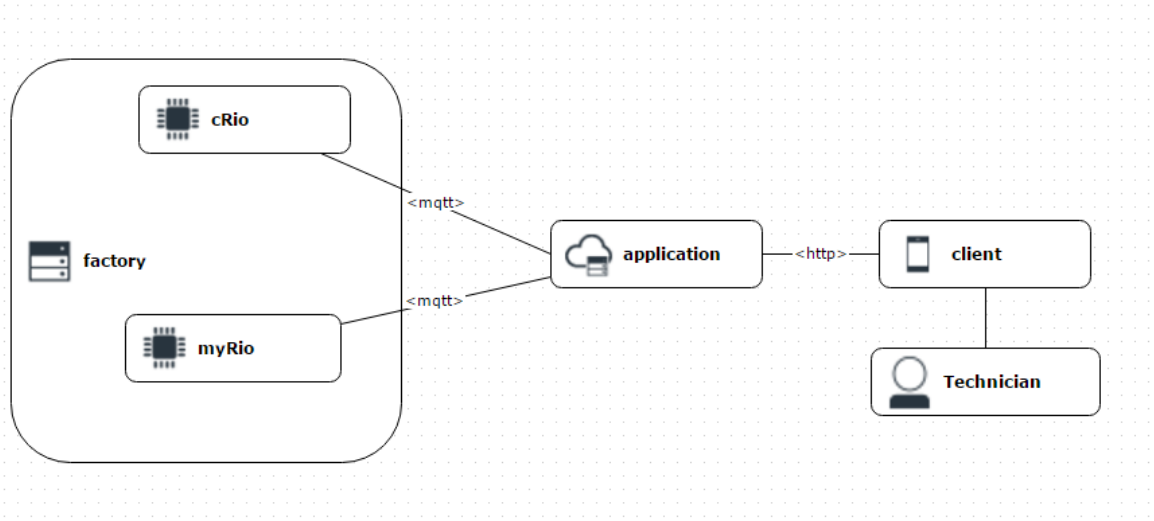


Figura 48 – Simulación del sistema creado en IoT Workbench

Para lograr el envío de alertas al usuario se ha utilizado el servicio IoT Real-Time Insights en Bluemix. Dentro de este, se han creado diferentes reglas que si se cumplen se envía automáticamente un email al usuario alertando de esta incidencia.

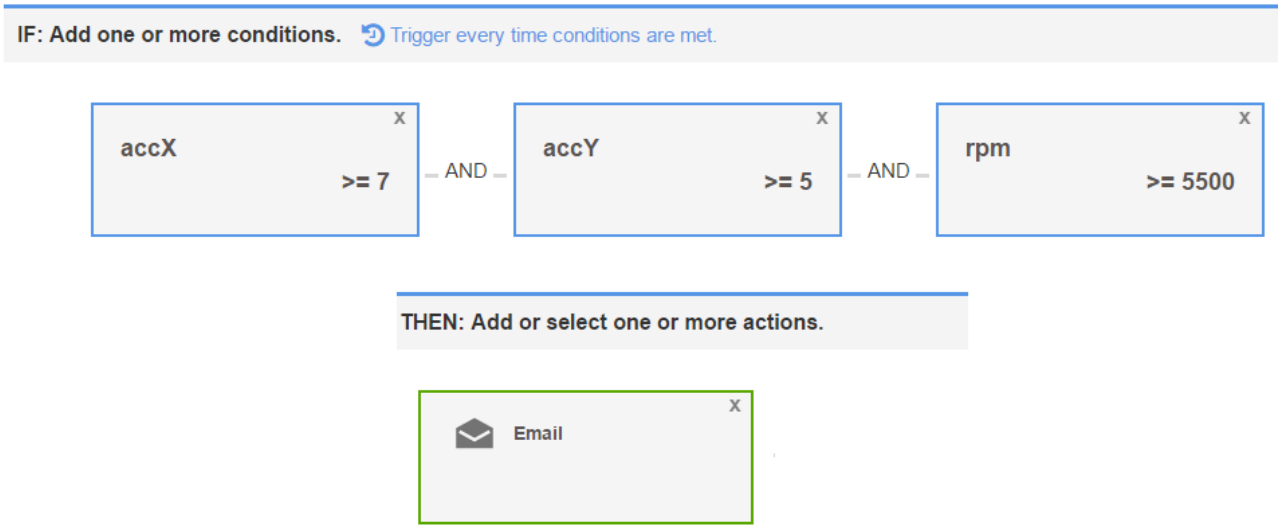


Figura 49 – Regla creada en IoT Real-Time Insights

En la Figura 49 se observa una regla creada y que esta enlazada con la acción de enviar un email al usuario en cuanto se cumpla la regla establecida.

4.3.4. EJEMPLO DE LA APLICACIÓN WEB: MONITORIZACIÓN DE UNA PLANTA EÓLICA

Para buscar un ejemplo real de monitorización para el uso del DSA Demo Box se ha decidido simular una planta eólica donde los ventiladores del NI DSA Demo Box se toman como si fuesen turbinas eólicas.

La aplicación sirve para que los técnicos que tienen que supervisar el estado de las turbinas puedan hacerlo de manera remota, teniendo en todo momento los valores de la turbina accesibles en cualquier dispositivo, ya que, la aplicación está accesible desde cualquier navegador web y se amolda a cualquier tamaño de pantalla de manera automática.

Además de poder monitorizar en todo momento el estado de las turbinas, el técnico recibe alertas en su correo electrónico si alguno de los valores sobrepasa un límite definido anteriormente.

Una vez que el técnico acceda a la aplicación mediante un navegador web, puede elegir que valores desea monitorizar desde un menú inicial.

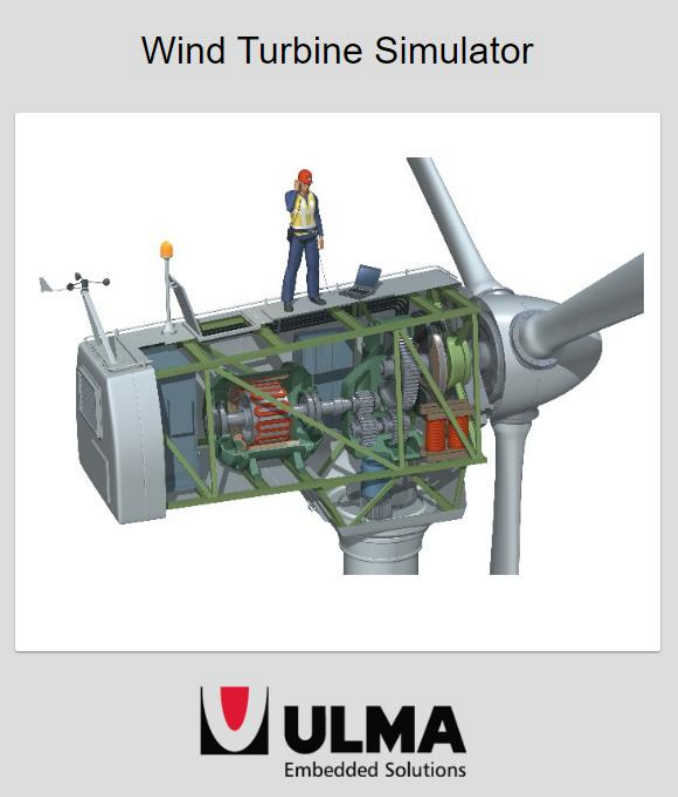


Figura 50 – Pantalla inicial de la aplicación web

En la Figura 51 se observa el menú desde el cual el técnico puede elegir los parámetros a monitorizar:

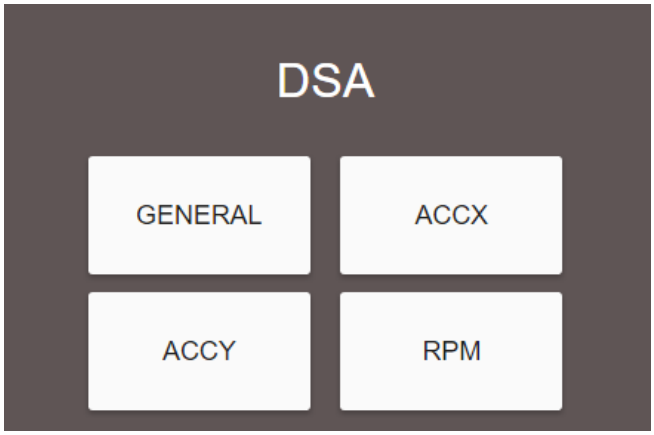


Figura 51 – Menú principal

Una vez elegido el o los parámetros a monitorizar, el técnico puede observar los valores y gráficos de la siguiente manera:

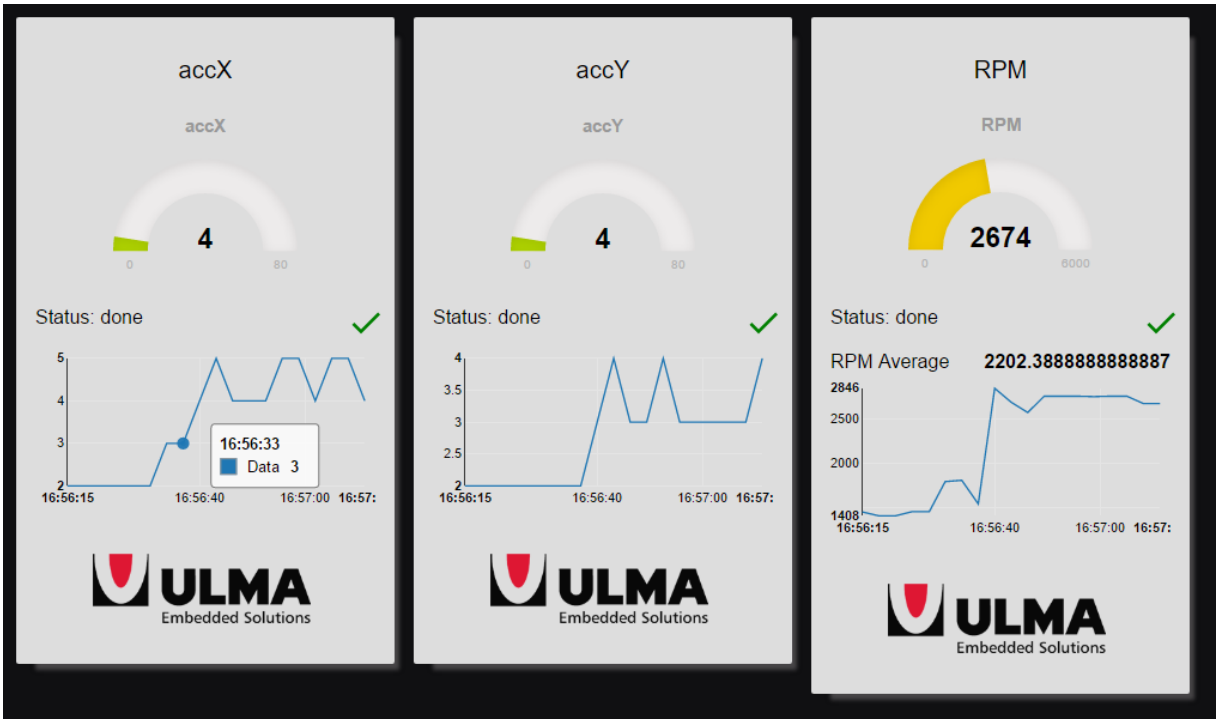


Figura 52 – Visualización de los datos

Para cada valor los campos que se observan son los mismos:

- El valor en ese mismo instante:

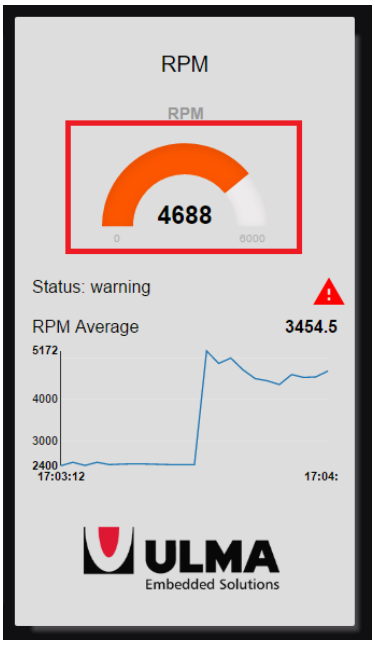


Figura 53 – El valor actual

- El estado del sistema en función del valor del parámetro monitorizado:

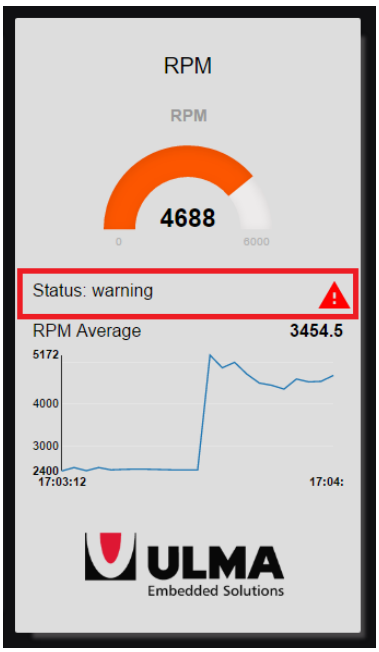


Figura 54 – Estado del sistema

- Un gráfico donde se observa el valor del parámetro a monitorizar en los últimos 3 minutos. Pasando el ratón por encima o pulsando encima se puede observar más en detalle el valor en cada momento:

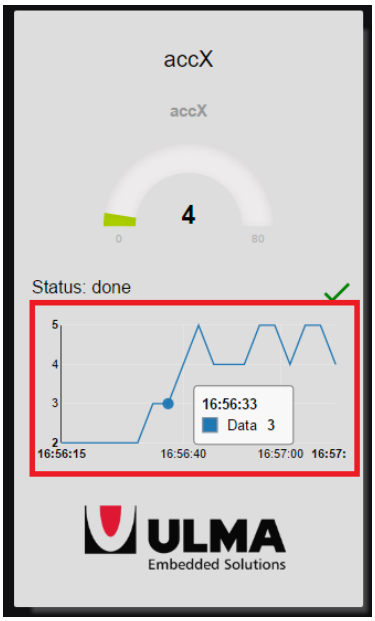


Figura 55 – Gráfico de los datos de los últimos 3 minutos

- En el caso de la monitorización de los RPM, también se añade la media de los últimos 3 minutos:

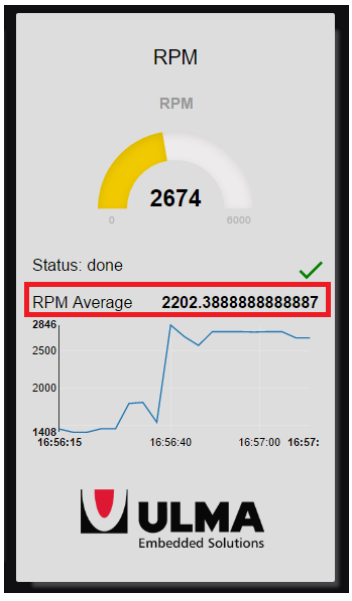


Figura 56 – Media de los valores de los últimos minutos

Como se ha comentado anteriormente, la aplicación está accesible desde cualquier navegador web y además se adecua a cualquier tamaño de pantalla.

Además de observar los datos del DSA Demo Box, en esta aplicación también se puede observar los datos enviados desde la NI myRIO. En este caso, este dispositivo no tiene ningún módulo conectado y envía los datos de su acelerómetro interior. Se trata de una extensión de la aplicación de monitorización aunque no tenga nada que ver con la monitorización de una turbina eólica. Añadiendo los datos de la NI myRIO se ha querido demostrar la facilidad de añadir nuevos dispositivos a la solución.

La aplicación web permite al usuario elegir entre que dispositivo elegir para observar los datos enviados desde dicho dispositivo.

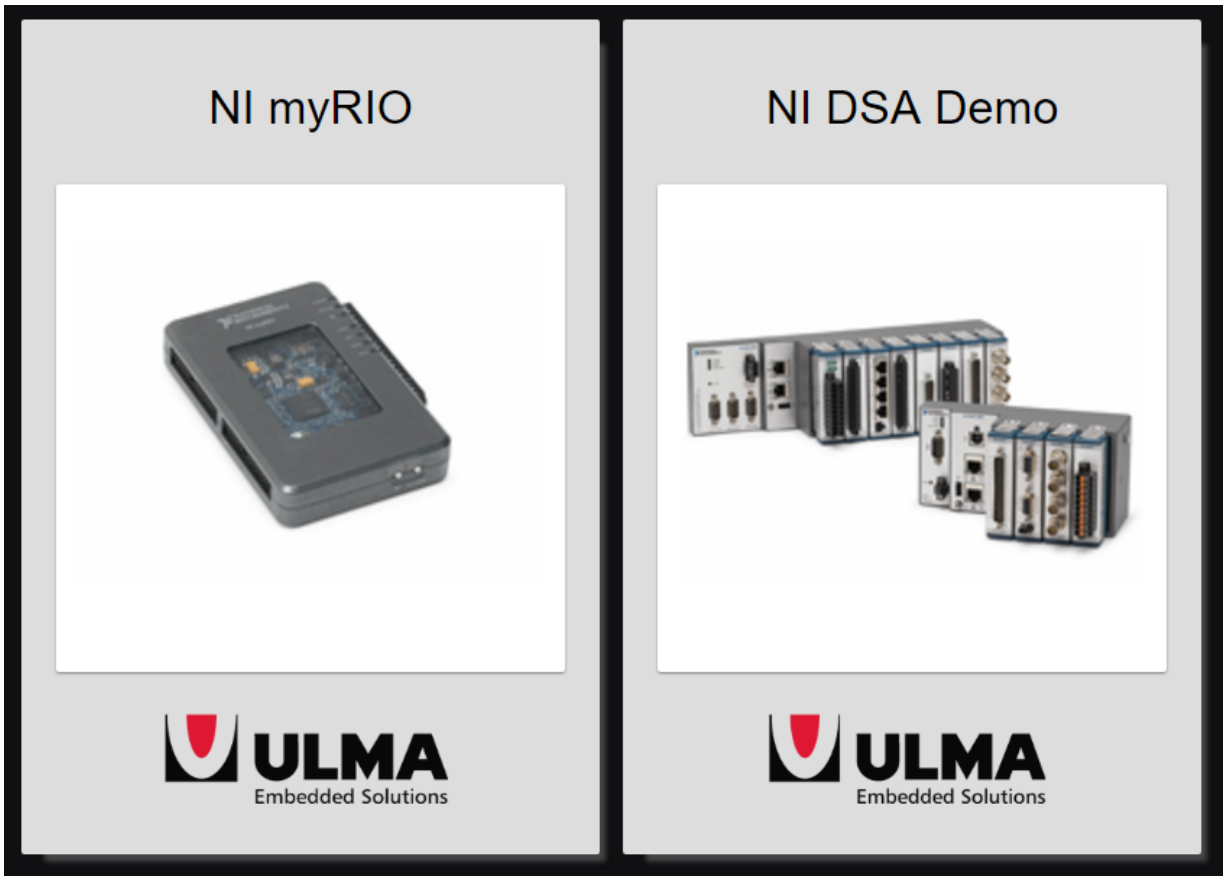


Figura 57 – Ventana para elegir el dispositivo

Al elegir la opción de observar los datos de la NI myRIO, el usuario observa los datos del acelerómetro interno del dispositivo. Este acelerómetro devuelve los valores de sus tres ejes X, Y y Z.

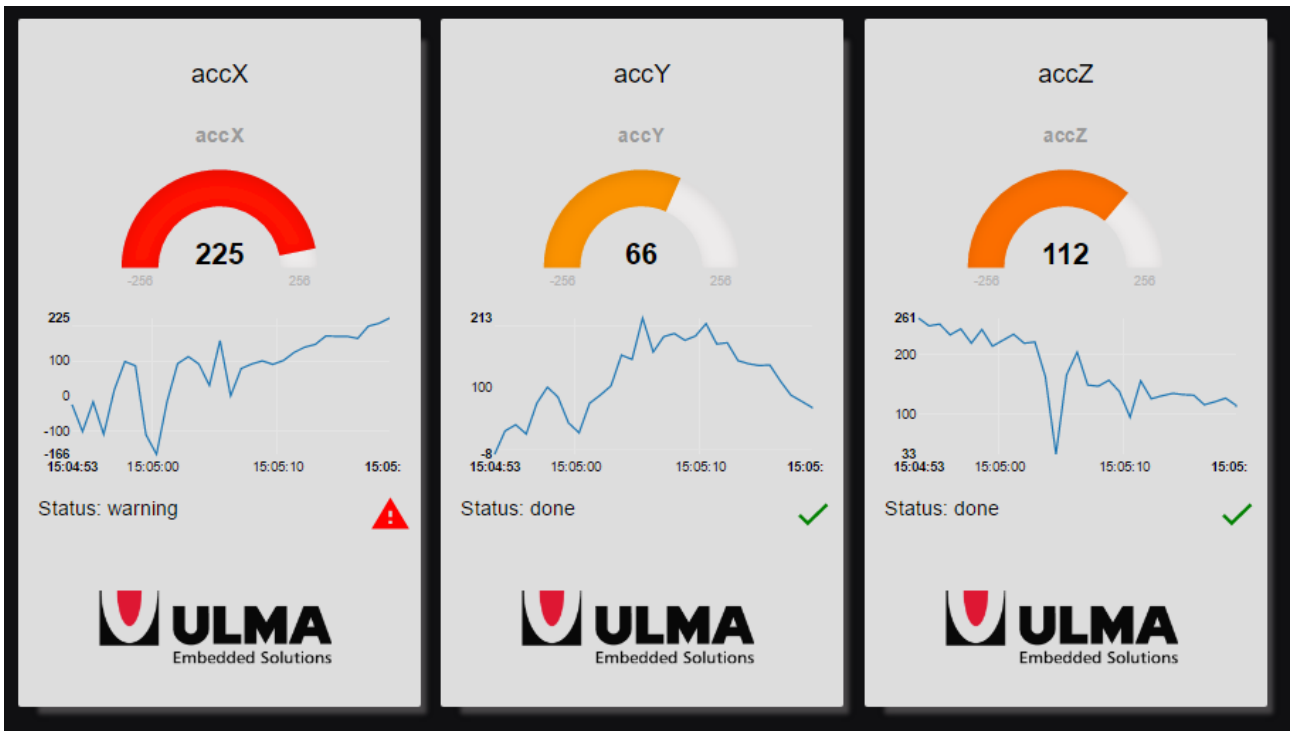


Figura 58 – Visualización de los datos enviados desde myRIO

El proceso seguido para lograr la conexión del NI myRIO ha sido el mismo que el del compactRIO-9033, con la diferencia de la lectura de los datos del DSA Demo Box, que en el caso del myRIO es sustituido por los valores de su acelerómetro interno

4.3.5. ANÁLISIS PREDICTIVO

Debido a la gran cantidad de datos consumidos por la aplicación y aprovechando el almacenamiento de estos datos en un base de datos, se ha decidido llevar a cabo un análisis predictivo del comportamiento de los ventiladores con el objetivo de aplicar estos análisis en un futuro al mantenimiento predictivo de una planta real. Un análisis predictivo es una técnica que se basa en el análisis de datos actuales e históricos con el objetivo de predecir futuros datos.

Dentro del análisis predictivo existen tres tipos de modelos sobre los que se lleva a cabo el análisis:

- Modelo predictivo: Utilizan los datos históricos y actuales para evaluar la probabilidad que detectar factores críticos de riesgos mientras se está llevando a cabo una determinada operación
- Modelo descriptivo: Utilizan los datos y la información obtenida del cliente con el objetivo de detectar vínculos y relaciones entre los datos y servicios. A diferencia del modelo predictivo no predicen el comportamiento de un único cliente.
- Modelo de decisión: Mediante la descripción de relación entre todos los datos utilizados para tomar una decisión tiene el fin de predecir los resultados de las decisiones de muchas variables.

Para llevar a cabo un análisis predictivo del comportamiento de un ventilador se ha decidido utilizar la herramienta de IBM SPSS Modeler que se encuentra dentro del grupo de productos de análisis predictivo de IBM. Esta herramienta nos permite crear y desplegar modelos predictivos con el objetivo de tomar las decisiones correctas.

4.3.5.1. IBM SPSS Modeler

Se trata de un conjunto de herramientas de minería de datos que permite desarrollar modelos predictivos y desplegarlos en diferentes operaciones con el objetivo de tomar las mejores decisiones. Proporciona un rango de algoritmos y técnicas avanzadas que proceden del aprendizaje automático, la inteligencia artificial y el estadístico. Por ejemplo ofrece técnicas de modelado tales como pronósticos, clasificaciones, segmentación y algoritmos de detección.

Las técnicas disponibles en SPSS Modeler son:

- Segmentación: K-medias, Kohonen, Bietápico y Anomalía.

- Asociación: A priori, GRI, CARMA, Análisis de secuencia.
- Clasificación: Factorial, Discriminante, Red Neuronal, C5.0, GLM, Máquinas de Vectores de Soporte, Redes Bayesianas, Modelos de auto aprendizaje, Vecino más próximo, Árboles, Listas de decisión y Selección de características.
- Predicción: Regresión Lineal, Series Temporales, Regresión de Cox y Regresión Logística
- Automáticos: Autonumérico, Autoclasificador, Autoagrupación y Modelizador ARIMA automático.

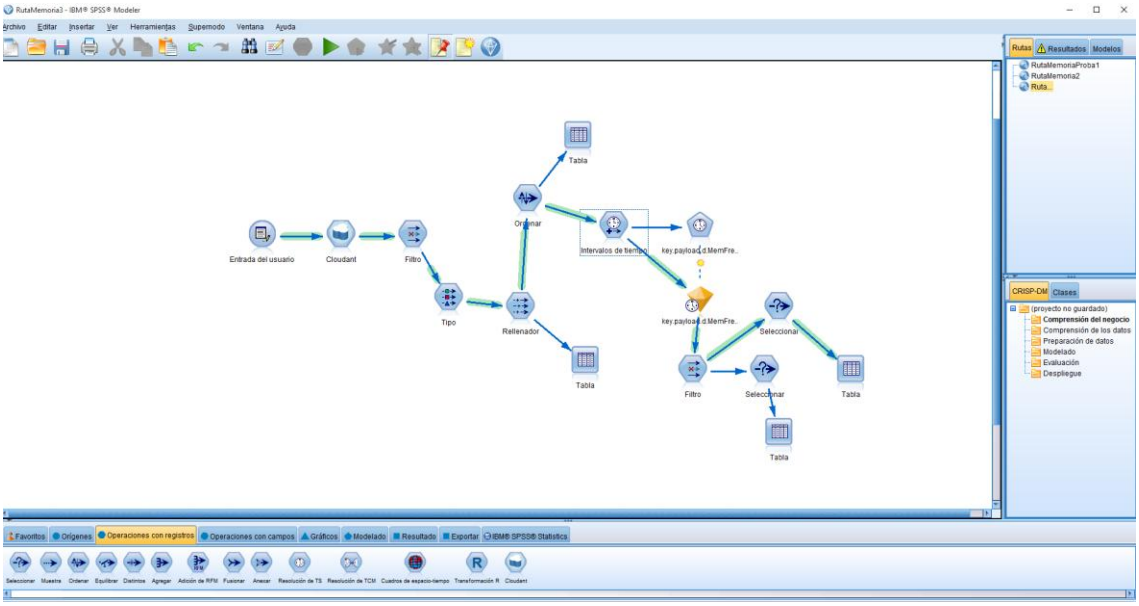


Figura 59 – Panel de IBM SPSS Modeler

En este caso el objetivo ha sido predecir diferentes valores en un determinado tiempo por lo que se ha utilizado la técnica llamada series temporales. Una serie temporal es un conjunto de mediciones ordenadas tomadas en intervalos regulares.

Los métodos de modelado de series temporales suponen que lo ocurrido hasta un cierto punto se vuelve a suceder, no de manera exacta, de una manera lo suficientemente parecida como para que estudiando su pasado se puedan sacar conclusiones de cara al futuro.

Los nodos modelos de serie temporal estiman tres tipos de métodos de previsión:

- **Suavizado exponencial**: Se trata de un método de previsión que utiliza valores ponderados de las observaciones anteriores para predecir los del futuro. Es una técnica útil para hacer previsiones de las series que muestran una tendencia, estacionalidad o ambas.

- **ARIMA** (Modelos autorregresivos integrados en media móvil): Proporcionan métodos más sofisticados que el suavizado exponencial, ya que ofrecen la ventaja de incluir variables independientes en el modelo. Estos modelos utilizan variaciones y regresiones de datos estadísticos anteriores para poder crear patrones con los que predecir el futuro. Son útiles cuando se desean incluir predictores que puedan ayudar a explicar el comportamiento de la serie que se está previendo.

  El modelo ARIMA se suele expresar con tres parámetros (P,D,Q) donde cada número indica el orden de los componentes del modelo:

  - P: componente autorregresivo
  - D: componente integrado
  - Q: componente de media móvil

- **Modelizador experto**: permite ajustar y estimar el modelo ARIMA o suavizado exponencial que más se ajusta a las necesidades, permitiendo no tener que identificar qué modelo es el apropiado mediante tests de prueba y error.

El modelado de series temporales devuelve diferentes valores con el prefijo $TS:

$TS-*nombrecol*	Valor pronosticado por el modelo para cada serie objetivo.
$TSLCI-*nombrecol*	Los intervalos de confianza más bajos para cada serie pronosticada.*
$TSUCI-*nombrecol*	Los intervalos de confianza más altos para cada serie pronosticada.*
$TSNR-*nombrecol*	Valor de residuo de ruido para cada columna de datos del modelo generado.*
$TS-Total	Total de los valores de $TS-*nombrecol* de esta fila.
$TSLCI-Total	Total de los valores de $TSLCI-*nombrecol* de esta fila.*
$TSUCI-Total	Total de los valores de $TSUCI-*nombrecol* de esta fila.*
$TSNR-Total	Total de los valores de $TSNR-*nombrecol* de esta fila.*

Figura 60 – Valores devueltos del modelado de series temporales

4.3.5.2. Desarrollo del problema

Con el objetivo de llevar a cabo diferentes experimentos sobre el análisis predictivo, para que en un futuro cercano se pueda llevar a cabo mantenimiento predictivo en los dispositivos conectados a la plataforma Watson IoT Platform, se ha utilizado la herramienta IBM SPSS Modeler. El mantenimiento predictivo de los dispositivos conectados a la plataforma es un tema muy importante, debido a que puede ayudar a los operarios a adelantarse a posibles fallos o averías en las instalaciones.

En la Figura 61 se puede observar la arquitectura del sistema para lograr hacer la pruebas predictivas.

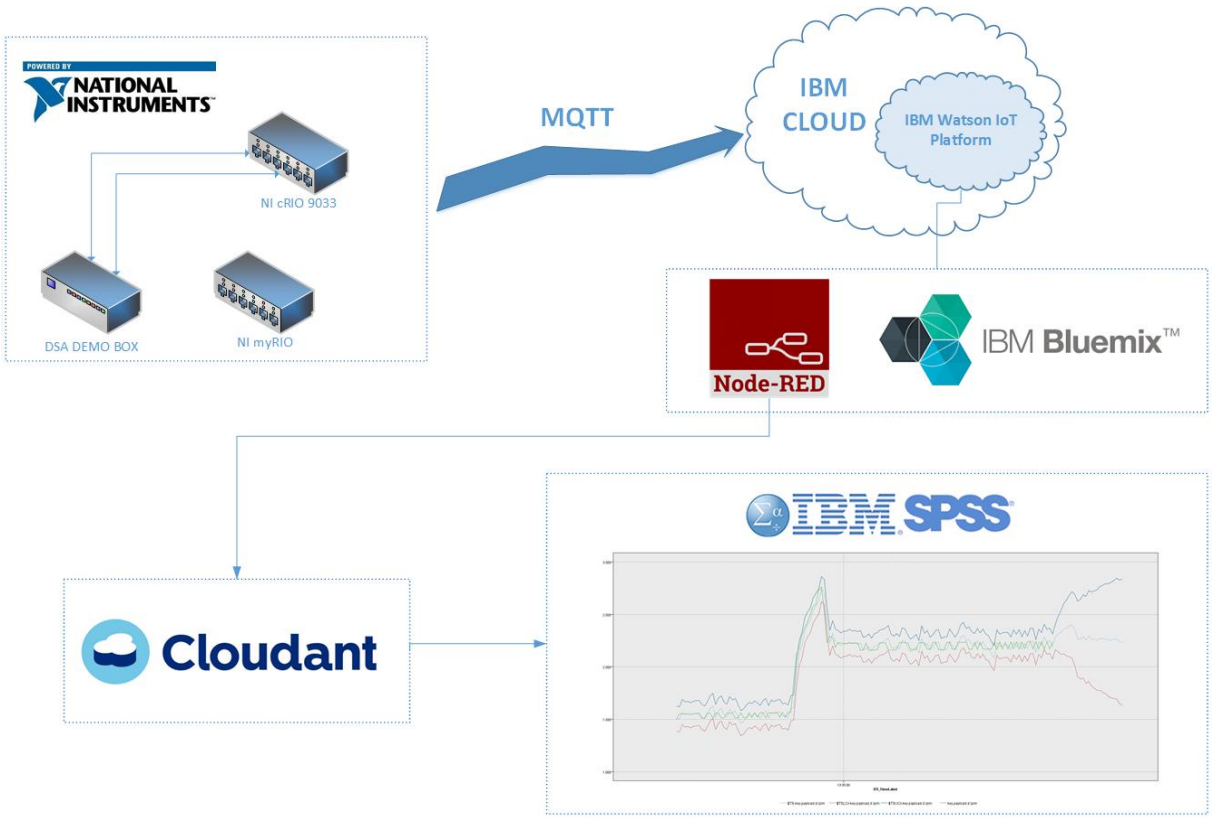


Figura 61 – Arquitectura del sistema para lograr las predicciones

La primera parte del sistema es el mismo de la aplicación de monitorización, donde los dispositivos se conectan al Watson IoT Platform mediante la aplicación de conexión. Además, mediante Node-Red se guía a los datos a una base de datos en CloudantDB para después acceder a estos datos desde IBM SPSS Modeler. Toda esta parte se ejecuta en la nube, pero la herramienta SPSS Modeler se ejecuta de manera local en el host, esto es, de una manera offline.

Para conseguir el acceso a los datos almacenados desde IBM SPSS Modeler se ha utilizado la extensión de Cloudant para SPSS Modeler.

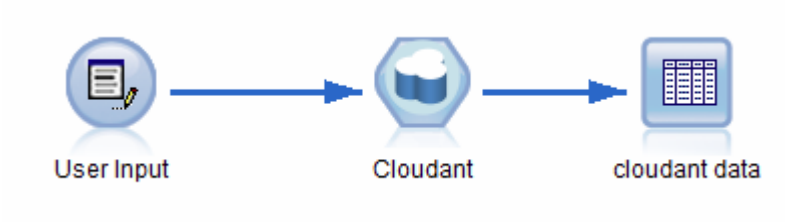


Figura 62 – Extensión de Cloudant en IBM SPSS Modeler

Para poder acceder a los datos de manera correcta se debe de crear dentro de la base de datos Cloudant una vista dentro de un documento de diseño. Las vistas permiten acceder a los datos almacenados dentro de una base de datos. Se tratan de funciones escritas en JavaScript y se definen dentro del documento de diseño.

Mediante el uso de una vista se pueden hacer diferentes consultas a la base de datos. Para hacer estas consultas, dentro de una vista se define una función de mapeo. Esta función produce datos de salida que representa un análisis de los documentos almacenado en la base de datos.

La extensión de Cloudant en SPSS Modeler permite filtrar los datos obtenidos desde la base datos. Ofrece tres opciones:

- Devolver la base de datos entera
- Devolver un documento específico
- Hacer una consulta a una vista y devolver su resultado

En este caso se ha decidido esta última opción con el objetivo de obtener solo datos válidos de la base de datos.

Una vez que se ha tenido acceso a los datos que están almacenados en Cloudant desde SPSS Modeler, se ha tenido que tratar los datos recibidos para poder trabajar y hacer con ellos análisis predictivos de manera correcta. Mediante diferentes nodos para el tratamiento de datos se ha especificado sobre qué datos hacer la predicción y se han amoldado los formatos de los datos con el objetivo de facilitar la predicción.

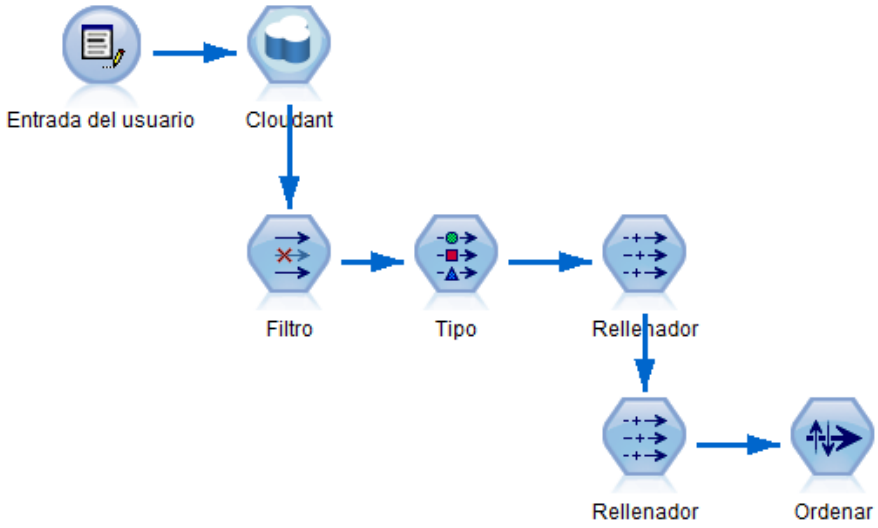


Figura 63 – Tratamiento de los datos recibidos desde Cloudant

Como se ha comentado anteriormente existen diferentes técnicas en SPSS Modeler para hacer predicciones y en este caso se ha elegido la técnica de series temporales, ya que el objetivo es predecir la respuesta del sistema en un futuro cercano.

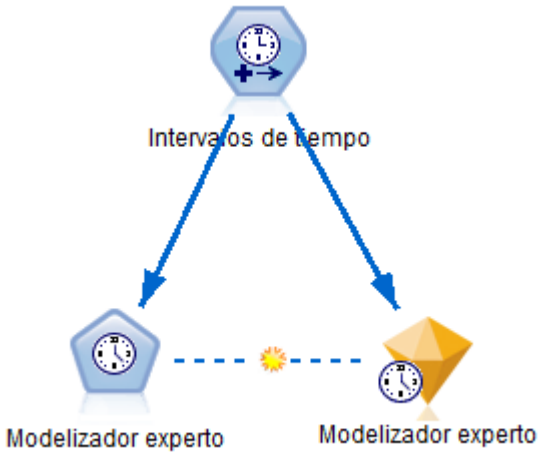


Figura 64 – Uso de los nodos para llevar a cabo la técnica de series temporales

Como resultado de la modelización se han logrado los datos predichos por el modelo. La herramienta permite crear gráficos con los que visualizar estos datos predichos, pero también permite observarlos mediante tablas o matrices.

En la Figura 65 y Figura 66 se observan los gráficos con los datos recibidos desde Cloudant y los datos predichos dependiendo del método utilizado. Los datos predichos son los que están dentro de los rectángulos rojos.

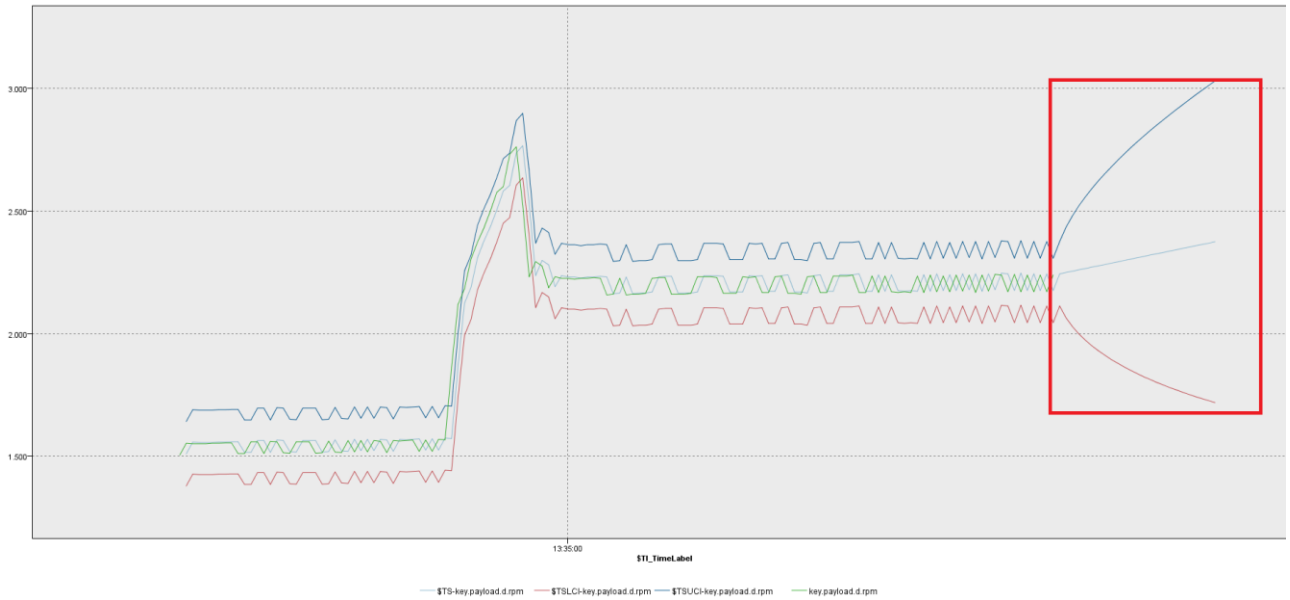


Figura 65 - Gráfico de los datos mediante el uso del método ARIMA

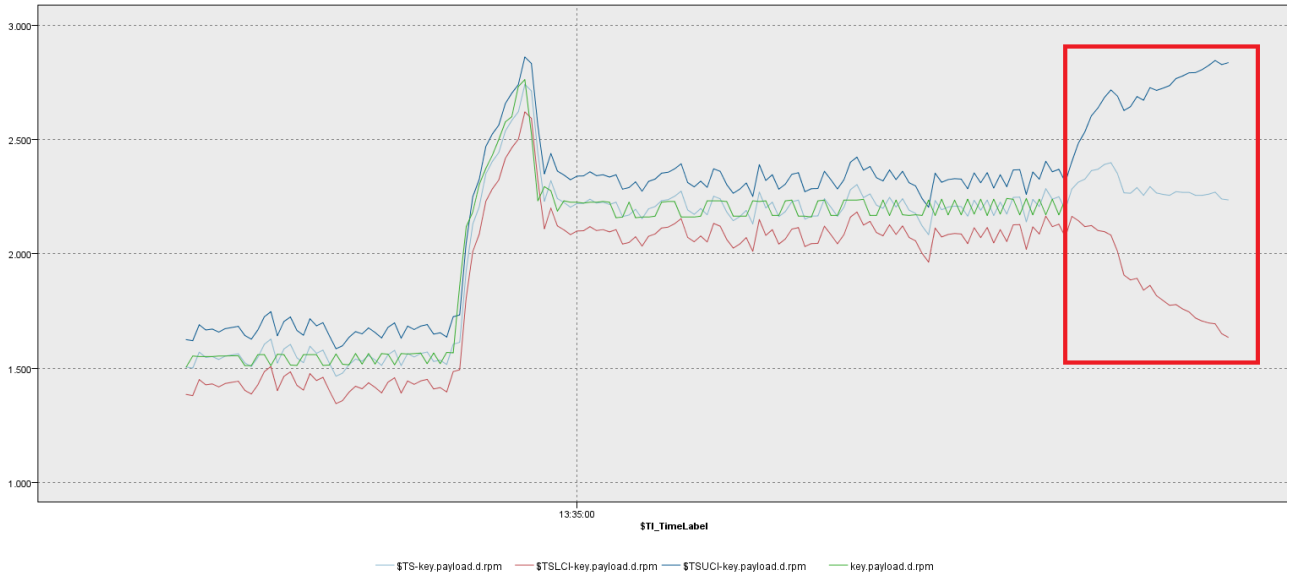


Figura 66 – Gráfico de los datos mediante el uso del método de Suavizado exponencial

En los dos casos se le han pasado al modelizador los datos de la velocidad de giro del tacómetro de uno de los ventiladores del DSA Demo Box, más concretamente se han utilizado los datos de los últimos 4 minutos. Además se ha definido una predicción de un minuto.

En la Tabla 6 y Tabla 7 se expone un fragmento de los resultados obtenidos al hacer la predicción. Se tratan de dos pequeños fragmentos de lo que se observa en la Figura 65 y Figura 66. A continuación se detallan los parámetros que aparecen en dichas tablas:

- Key.payload.d.rpm: El valor real del tacómetro en revoluciones por minuto.
- $TI_TimeLabel: La hora exacta.
- $TI_Futuro: Si este parámetro se encuentra a 1 significa que es un dato predicho.
- $TS-key.payload.d.rpm: Valor pronosticado por el modelo para la velocidad de giro del tacómetro.
- $TSLCI-key.payload.d.rpm: Valor del intervalo de confianza más bajo para la velocidad de giro del tacómetro.
- $TSUCI-key.payload.d.rpm: Valor del intervalo de confianza más alto para la velocidad de giro del tacómetro.

key.payload.d.rpm	$TI_TimeLabel	$TI_Futuro	$TS-key.payload.d.rpm	$TSLCI-key.payload.d.rpm	$TSUCI-key.payload.d.rpm
2171	13:37:20	0	2247	2116	2379
2239	13:37:22	0	2176	2045	2308
2171	13:37:24	0	2244	2113	2376
2239	13:37:26	0	2176	2045	2308
2171	13:37:28	0	2244	2113	2376
2239	13:37:30	0	2176	2045	2308
$null$	13:37:32	1	2244	2113	2376
$null$	13:37:34	1	2250	2064	2435
$null$	13:37:36	1	2255	2028	2482
$null$	13:37:38	1	2261	1998	2523
$null$	13:37:40	1	2266	1973	2559
$null$	13:37:42	1	2272	1950	2593
$null$	13:37:44	1	2277	1930	2624
$null$	13:37:46	1	2282	1912	2653
$null$	13:37:48	1	2288	1894	2681
$null$	13:37:50	1	2293	1879	2708
$null$	13:37:52	1	2299	1864	2734

Tabla 6 – Fragmento de la tabla de resultados mediante el uso del método ARIMA

key.payload.d.rpm	$TI_TimeLabel	$TI_Futuro	$TS-key.payload.d.rpm	$TSLCI-key.payload.d.rpm	$TSUCI-key.payload.d.rpm
2171	13:37:20	0	2238	2118	2358
2239	13:37:22	0	2207	2087	2327
2171	13:37:24	0	2285	2165	2405
2239	13:37:26	0	2239	2119	2359
2171	13:37:28	0	2250	2130	2370
2239	13:37:30	0	2194	2074	2314
$null$	13:37:32	1	2284	2164	2404
$null$	13:37:34	1	2314	2145	2484
$null$	13:37:36	1	2327	2119	2534
$null$	13:37:38	1	2364	2124	2604
$null$	13:37:40	1	2370	2102	2638
$null$	13:37:42	1	2391	2097	2684
$null$	13:37:44	1	2399	2082	2717
$null$	13:37:46	1	2351	2011	2690
$null$	13:37:48	1	2267	1907	2627
$null$	13:37:50	1	2265	1886	2644
$null$	13:37:52	1	2290	1893	2688

Tabla 7 – Fragmento de la tabla de resultados mediante el uso del método de Suavizado exponencial

4.3.6. MODELADO DEL SOFTWARE

En este apartado se describe el modelado del software de conexión a IBM Watson IoT Platform mediante la herramienta IBM Rational Rhapsody.

La modelación de la aplicación tiene el objetivo de facilitar los futuros usos del sistema, reduciendo la cantidad de cambios en los casos que la monitorización no se trate del DSA Demo Box o de que se utilicen diferentes dispositivos para enviar los datos a IBM Watson IoT Platform.

4.3.6.1. IBM Rational Rhapsody

Esta herramienta ofrece ayuda para analizar y validar requisitos, diseñar de forma rápida con prototipos y crear aplicaciones coherentes utilizando SysML(Systems Modeling Language) y UML(Unified Modeling Language).

Existen diferentes versiones del software IBM Rational Rhapsody. Cada una de las versiones está enfocada para cumplir las necesidades de los ingenieros de sistemas y desarrolladores. En este caso se ha utilizado la versión Rational Rhapsody Developer que ayuda a los usuarios a desarrollar y validar aplicaciones, además de añadir la posibilidad de incluir la simulación, la creación de código y la integración de sistemas en tiempo real.

La herramienta IBM Rational Rhapsody Developer se trata de un entorno de desarrollo pensado para ayudar en la productividad durante todo el ciclo de vida de desarrollo del software desde la captura de requisitos a la implementación y desarrollo.

Entre los servicios que ofrece se encuentran los siguientes:

- Generación completa de aplicaciones

  Permite generar código de la aplicación para los lenguajes C, C++, Java y Ada, permitiendo incluir gráficos de estado y diagramas de actividad. Además, ofrece la sincronización entre el diseño y el código y permite la integración con Eclipse logrando un entorno integrado entre el código, el modelo y la depuración.

- Simulación y pruebas basadas en modelos

  Gracias a la animación, Rhapsody ayuda en la depuración del modelo permitiendo eliminar defectos de manera anticipada. Las animaciones hacen el trabajo de un depurador tradicional pero con mayor nivel de abstracción. Además, ofrece la posibilidad de inyectar eventos o llamar a operaciones que permiten recorrer el modelo. Por otra parte, también

ofrece la posibilidad de visualizar los requisitos complejos con UML, SysML o lenguajes DSL.

- Trazabilidad de los requisitos

  Proporciona una trazabilidad completa a los requisitos permitiendo insertar información de los requisitos en el código para ayudar en la trazabilidad de la implementación final, cumpliendo así con los estándares de seguridad. Además incluye análisis estático en la comprobación de los modelos, ayudando así a mejorar la coherencia e integridad de los modelos.

- Colaboración en equipo

  Ayuda a los equipos a colaborar para gestionar la complejidad de desarrollar diseños coherentes en diferentes entornos.

- Soporte durante el ciclo de vida y el software complementario

  Rational Rhapsody Developer se integra con otros productos de IBM Rational como pueden ser IBM Rational DOORS, IBM Rational Systems Architect, IBM Rational Team Concert o IBM Rational ClearCase permitiendo así un desarrollo completo de todo el ciclo de vida de un producto. Por otro lado, entre el software complementario se encuentran el software de herramientas y utilidades de IBM Rational Rhapsody que permite crear paneles gráficos para bocetos de un diseño o el software complementario de generación automática de pruebas de IBM Rational Rhapsody que ayuda a facilitar una mayor cobertura a las pruebas.

4.3.6.2. Desarrollo del problema

Con el objetivo de modelar la conexión de los dispositivos a IBM Watson IoT Platform se ha utilizado la herramienta de desarrollo Eclipse C & C++ Development Tools for NI Linux Real-Time 2014 añadiéndole el plugin Rational Rhapsody Modeling. Este plugin permite desarrollar proyectos de Rational Rhapsody desde dentro de Eclipse. Además de permitir crear modelos en Rational Rhapsody, permite generar automáticamente el código en C/C++ correspondiente a estos modelos.

Figura 67 – Integración de Rational Rhapsody en Eclipse

Después de generar el código, se puede crear un ejecutable al compilar el código e importarlo a la compactRIO mediante la herramienta de Eclipse llamada Remote System Explorer. Una vez importado, el modelo se ejecuta en el dispositivo, logrando visualizar su comportamiento desde el host u ordenador personal.

El modelo creado consta de dos partes:

- Generación de datos:

  Esta parte se encarga de la lectura de datos desde el DSA Demo Box. Esta clase llama a una aplicación desde dentro del NI Linux Real-Time. Esta aplicación es el que lee los valores enviados desde el DSA Demo Box y el encargado de devolver los datos. Una vez ejecutado la aplicación, esta clase envía mediante un puerto los datos a la clase que se encarga del envío de los datos a la plataforma.

- Envío de datos a la plataforma:

  Esta parte se encarga de crear los paquetes con los datos recibidos desde la parte de generación de datos y después envía los datos a la plataforma. Los datos los recibe desde un puerto que está conectado con el puerto de la clase encargada de la generación de datos.

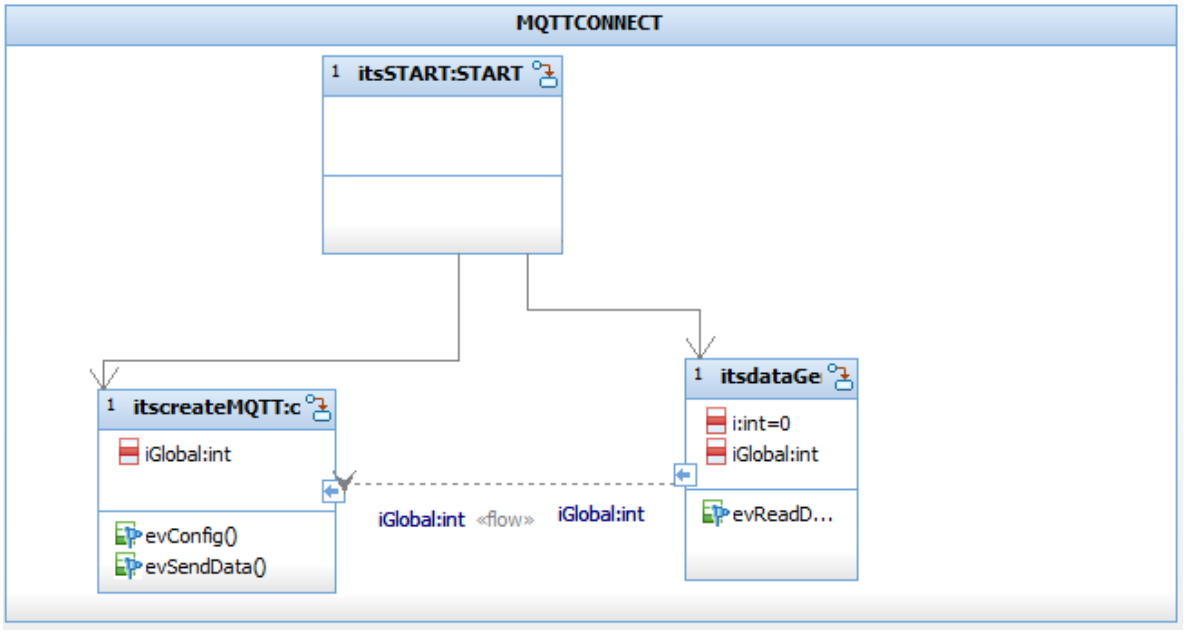


Figura 68 – El diagrama de clases utilizado

Para probar y verificar los modelos creados, Rational Rhapsody Developer ofrece la posibilidad de añadir animaciones al modelo. Así se puede comprobar su funcionamiento cuando se ponga en marcha el modelo, esto es, la animación es el mecanismo que permite validar el modelo. La animación de Rational Rhapsody, además, ayuda a depurar el sistema a nivel de diseño de manera más eficaz que utilizando la depuración a nivel de código.

En este caso se ha modificado la configuración de Eclipse con el fin de poder añadir animaciones al modelo cuando este se ejecute en el dispositivo. Además, se ha añadido la posibilidad de depurar el código aparte de depurar el modelo a nivel de diseño. Estas dos posibilidades nos permiten validar el modelo mediante dos niveles de depuración: a nivel de código y a nivel de diseño.

Al añadir las animaciones a la depuración, el usuario puede ir generando los eventos en Rhapsody mientras depura el código generado. Además, se pueden añadir puntos de interrupción tanto en la animación como en el código permitiendo así un mayor nivel de depuración. Esto es, añadiendo un punto de interrupción en la animación se puede saber en qué punto del código se encuentra el programa cuando salte dicha interrupción. Por el contrario, si se añade un punto de interrupción en el código se puede saber en qué estado se encuentra la animación al saltar la interrupción.

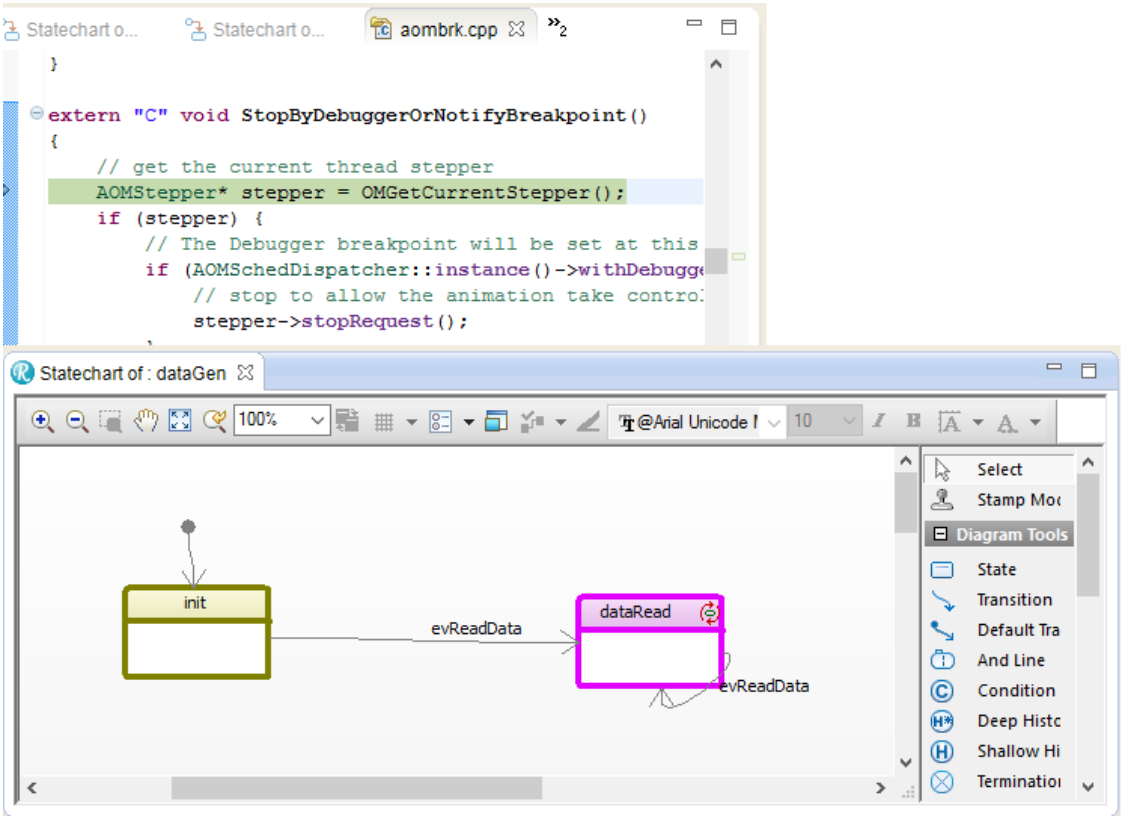


Figura 69 – Ejemplo de la animación del modelo creado y la depuración del código

5. CONCLUSIONES

5.1. OBJETIVOS DEL PROYECTO

Al inicio del proyecto, el objetivo consistía en estudiar e implementar la integración entre IBM Watson IoT Platform y los dispositivos de National Instruments, además de crear una aplicación para explotar los datos. Por último, como extensión, se precisaba la modelización de la solución creada mediante IBM Rational Rhapsody.

Se considera el objetivo cumplido e incluso se ha llegado a añadir al proyecto la funcionalidad de predicción, ya que una vez logrado la integración y creada la aplicación de monitorización se ha extendido el proyecto mediante experimentos de predicción gracias al software de IBM SPSS Modeler.

La aplicación de monitorización creada servirá como referencia para otros proyectos que se enfoquen dentro del IoT. Ya que se ha logrado crear y mantener una aplicación IoT totalmente personalizable permitiendo añadir nuevos dispositivos a la solución de manera sencilla. Además, la integración de Eclipse con Rational Rhapsody y su configuración puede servir como referencia para otros proyectos que precisen la modelización junto a la depuración del código.

Respecto a las fases del proyecto, la mayoría de ellas se han completado de manera satisfactoria. Se ha realizado el estudio de la plataforma IBM Watson IoT y gracias a este estudio se ha determinado qué librerías eran las idóneas para la conexión de los dispositivos. En cuanto al envío de los datos, se ha utilizado LabVIEW y su API llamada FPGA Interface C API para efectuar la comunicación entre la FPGA y la aplicación en tiempo real que se encarga de leer los datos de los dispositivos conectados. Para integrar el código en Eclipse y configurarlo para ejecutar los programas dentro de NI Linux Real-Time ha habido algunas dificultades debido a las diferentes versiones de los compiladores y a la compleja configuración de Eclipse, pero una vez logrado configurar Eclipse de manera correcta esta configuración ha servido para todo el desarrollo del proyecto. Otra de las dificultades que han ocurrido ha sido al compilar los programas dentro del NI Linux Real-Time. Debido a que a la hora de instalar nuevos paquetes National Instruments ofrece un soporte limitado y que NI Linux Real-Time utiliza el mánager de paquetes ligero OPKG, se han debido de instalar paquetes adicionales que en un principio no estaban pensados instalar.

Después de conseguir ejecutar la aplicación en NI Linux Real-Time y de enviar los datos a IBM Watson IoT Platform se ha creado la aplicación en IBM Bluemix utilizando Node-Red. El uso

de los módulos utilizados dentro de esta herramienta, que han servido para crear la aplicación web, ha precisado del conocimiento de desarrollo front-end, pero sobre todo de AngularJS, por lo que se han tenido que aprender diferentes conceptos de este framework de JavaScript.

Por otra parte, se ha podido modelar en Rational Rhapsody el software para conectar los dispositivos y enviar los datos a la plataforma. El modelo creado mediante Rational Rhapsody se ha ejecutado desde el dispositivo enviando los datos a la plataforma. Además, se ha podido verificar su funcionamiento gracias a la integración de las animaciones en Rational Rhapsody y la depuración del código dentro de la herramienta Eclipse. En esta parte del desarrollo también ha habido dificultades, como ha sido la configuración de Eclipse para poder acceder a las librerías necesarias para poder ejecutar el modelo y las animaciones en el dispositivo con NI Linux Real-Time.

Por último, la valoración a nivel general del proyecto y de la estancia en la empresa es positiva. Me ha permitido desenvolverme en un entorno de trabajo real y compartir el día a día con un grupo de profesionales altamente cualificados. Además, he tenido la oportunidad de presentar este proyecto en el foro tecnológico NIdays 2016 organizado por National Instruments en Madrid y también he podido hacer diferentes demostraciones de la solución creada a representantes de IBM y National Instruments.

5.2. LÍNEAS FUTURAS

A parte del trabajo hecho, todavía se pueden añadir al proyecto diferentes mejoras de cara al futuro.

Entre estas mejoras se encuentran la de integrar la herramienta IBM Rational Rhapsody TestConductor en el modelo creado en Rhapsody y ejecutar diferentes pruebas en NI Linux Real-Time. Esta herramienta proporciona la posibilidad de especificar tests o pruebas y ejecutarlas integrándose con el modelo creado mediante UML/SysML. En nuestro caso, se ha intentado integrar esta herramienta con el modelo creado y ejecutar las pruebas desde dentro del NI compactRIO-9033, pero han ocurrido algunos problemas que han impedido llevar a cabo estas pruebas, por lo que sería uno de los trabajos futuros investigar más a fondo esta integración.

Por otro lado, otro de los trabajos futuros es integrar más sensores o dispositivos que envíen datos a la plataforma para observar cómo reaccionaría la aplicación creada. Además, sería interesante el uso de gateways para que gestionen todos estos dispositivos.

Por otra parte, se ha intentado añadir la función de predicción a la aplicación web mediante el uso del servicio IBM Bluemix Predictive Analytics. Este servicio está pensado para permitir ejecutar en la nube los modelos creados en IBM SPSS Modeler, pero no se ha podido llevar a cabo esta integración debido a los diferentes problemas que han ocurrido. La escasa documentación de dicho servicio y la falta de soporte ofrecido por parte de IBM sobre este servicio han dificultado su uso y la posibilidad de añadir las predicciones a la aplicación web. Otra línea a seguir en el futuro es la profundizar más en el uso de este servicio y lograr hacer esas predicciones de manera online. Además, IBM se encuentra trabajando en la integración de las funcionalidades de su herramienta IBM SPSS Modeler dentro de la plataforma IBM Watson Analytics, con la intención de permitir en un futuro cercano que los usuarios puedan utilizar los modelos predictivos en la nube de IBM mediante el uso de dicha plataforma. Esta sería otra manera de lograr hacer las predicciones de manera online. Gracias a esto, se lograría una solución IoT que permitiría hacer una monitorización real de los dispositivos conectados y además lograr un mantenimiento predictivo permitiendo a los usuarios anticiparse a futuros errores o averías.

6. REFERENCIAS

1) Informe sobre la transformación digital de la industria española

http://www6.mityc.es/IndustriaConectada40/informe-industria-conectada40.pdf

2) Las tecnologías IoT dentro de la industria 4.0

https://www.eoi.es/savia/documento/eoi-80491/las-tecnologias-iot-dentro-de-la-industria-conectada-40

3) Comunidad de IBM Bluemix en IBM developerWorks

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/We0d917403ade_46b2_8991_d1eabb8126f6

4) Vibration Analysis with LabVIEW and the Sound and Vibration Measurement Suite. 2013 National Instruments

5) Introduction to NI Linux Real-Time – National Instruments

http://www.ni.com/white-paper/14627/en/

6) C/C++ Embedded System design Tools – National Instruments

http://www.ni.com/white-paper/14623/en/

7) NI cRIO-9033 User Manual – National Instruments

http://www.ni.com/pdf/manuals/376211a_03.pdf

8) NI myRIO-1900 User Guide and Specifications

http://www.ni.com/pdf/manuals/376047a.pdf

9) MQTT Man page:

http://mosquitto.org/man/mqtt-7.html

10) IBM Internet of Things Platform Documentation. David Parker. March 2016

https://media.readthedocs.org/pdf/iotf/latest/iotf.pdf

11) Tutorial de LabVIEW. Asun Zafra Cabeza

http://www.esi2.us.es/~asun/LCPC06/TutorialLabVIEW.pdf

12) Documentación de IBM SPSS Modeler V17.0.0 en IBM Knowledge Center

http://www.ibm.com/support/knowledgecenter/es/SS3RA7_17.0.0/clementine/knowledge_center/product_landing.html

13) Securing IoT devices and gateways

http://www.ibm.com/developerworks/security/library/iot-trs-secure-iot-solutions1/index.html?ca=drs-&ce=ism0070&ct=is&cmp=ibmsocial&cm=h&cr=crossbrand&ccy=us

14) Cloudant online documentation

https://docs.cloudant.com/index.html

15) IBM Rational Rhapsody Developer

http://www-03.ibm.com/software/products/en/ratirhap

16) Essentials of IBM Rational Rhapsody Designer for Systems Engineers V8.1.1. 2011 IBM Corporation

7. ANEXOS

7.1. ANEXO 1: MODELO DE INTEGRACIÓN CREADO EN RHAPSODY

El presente anexo tiene el objetivo de mostrar el modelo creado en IBM Rational Rhapsody para lograr enviar los datos recibidos desde los dispositivos a la plataforma IBM Watson IoT.

Como se ha explicado más arriba el modelo creado se compone de dos clases:

- **dataGEN**: Esta clase se encarga de generar los datos a enviar, esto es, cada vez que se ejecuta lee un dato desde el DSA Demo Box. Este dato puede ser cualquiera de los tres datos enviados desde el DSA Demo Box (accX, accY y RPM).
- **createMQTT:** Esta clase se encarga de enviar los datos a la plataforma IBM Watson IoT Platform. Se le deberá de pasar el dato que se quiera enviar. Este dato se genera en la clase dataGEN y se recibe mediante un Flow Port.

En la siguiente imagen se observa el diagrama principal del modelo:

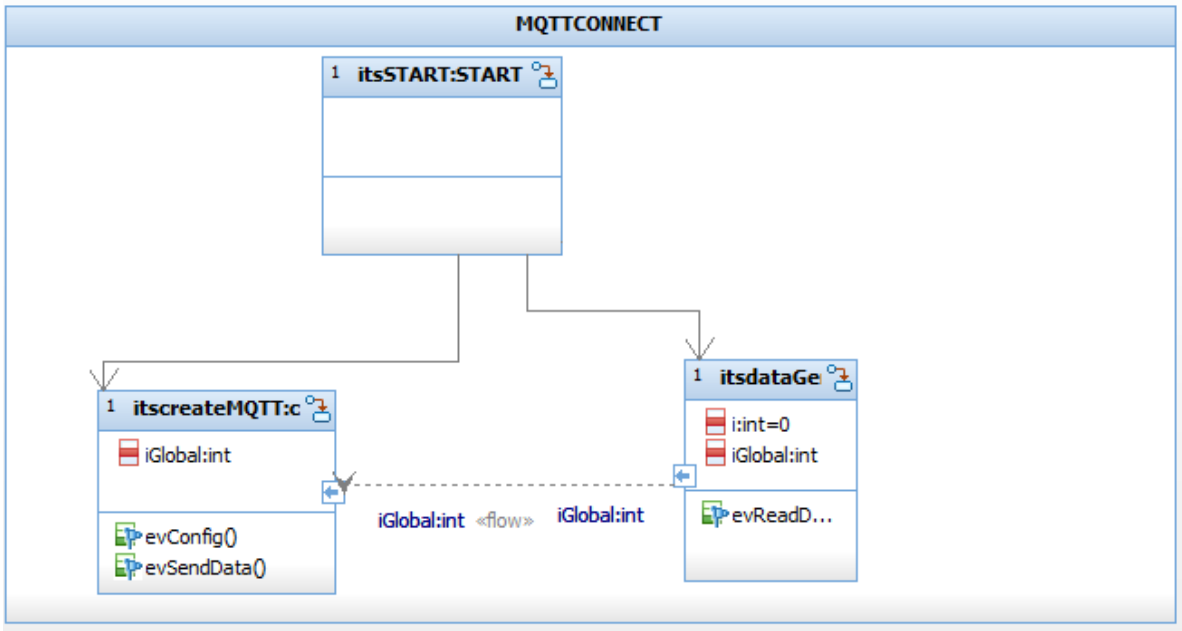


Figura 70 - Diagrama principal del modelo

7.1.1. CLASE DATAGEN

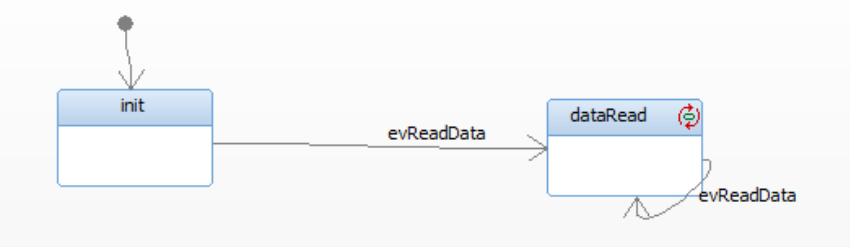


Figura 71 – Diagrama de estados de la clase dataGEN

Esta clase se compone de un diagrama de estados con dos estados:

- **init**: Estado inicial.
- **dataRead**: Estado en el que se ejecuta una acción a su entrada. En dicha acción se ejecuta el ejecutable dataGen dentro del compactRIO. Este ejecutable lee los valores recibidos desde el DSA Demo Box.

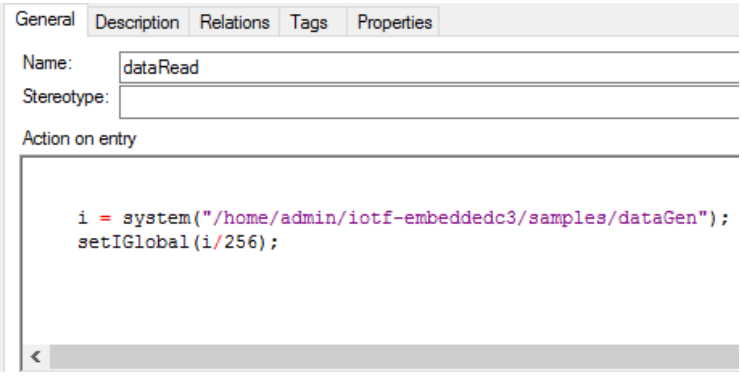


Figura 72 – Acción del estado dataRead

Para el envío de los datos generados a la clase *createMQTT* se ha utilizado un Flow Port. Al utilizar este tipo de puertos y al generar el código del modelo, Rhapsody crea la función *setIGlobal* que se encarga de enviar el parámetro a la otra clase en la que se encuentre el destino del flujo.

```
void dataGen::setIGlobal(int p_iGlobal) {

    if (iGlobal != p_iGlobal)  {
        iGlobal = p_iGlobal;
        FLOW_DATA_SEND(iGlobal, iGlobal_SP, SetValue, x2String);
    }
}
```

En este caso, esta función ha servido para el envío de los datos recibidos desde el DSA Demo Box a la clase createMQTT, que es la clase encargada del envío de los datos a la nube.

7.1.2. CLASE CREATEMQTT

Al igual que la clase anterior, la clase createMQTT también dispone de un diagrama de estados que está compuesto de dos estados.

- **initConfig**: Estado inicial.
- **iotfclient**: Estado en el que se ejecuta una acción a su entrada. En esta acción se llama al ejecutable de compactRIO, que envía los datos recibidos a la plataforma IBM Watson IoT.

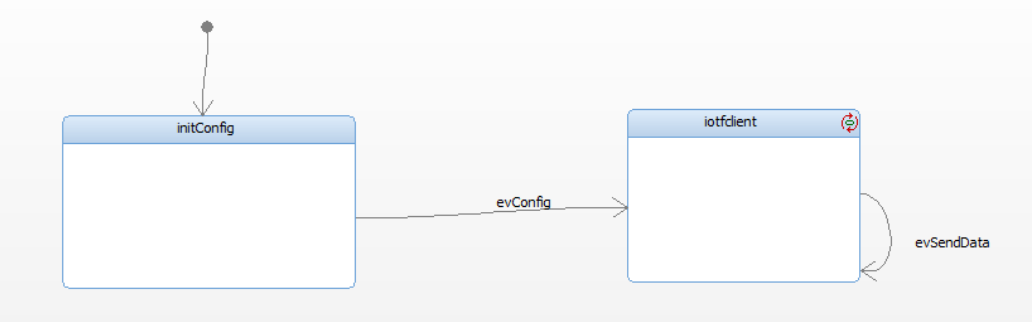


Figura 73 – Diagrama de estados de la clase createMQTT

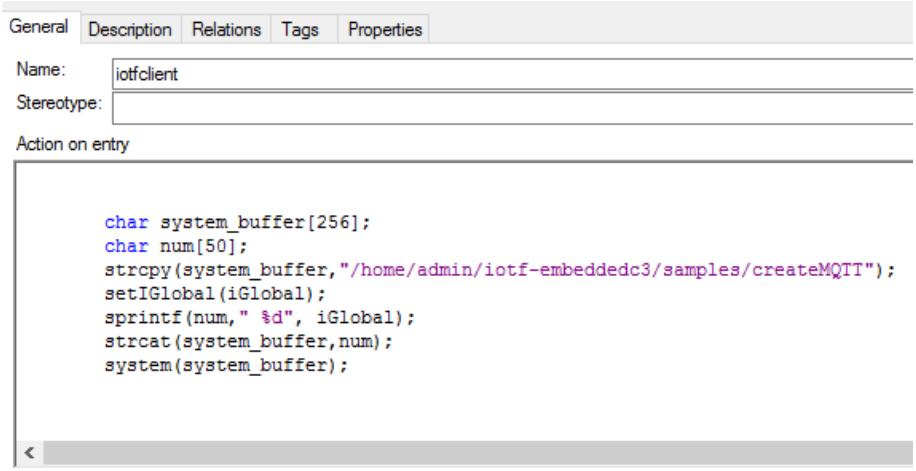


Figura 74 - Acción del estado iotfclient

Para recibir los datos desde el Flow Port se ha utilizado como en la clase *dataGEN* la función *setIGlobal*, pero esta función es diferente al de dicha clase, ya que esta está preparada para recibir un flujo de datos.

```
void createMQTT::setIGlobal(int p_iGlobal) {
    if (iGlobal != p_iGlobal) {
        iGlobal = p_iGlobal;
        FLOW_DATA_RECEIVE("iGlobal", iGlobal, x2String);
    }
}
```

Por lo tanto, esta clase se encarga de recibir un dato desde su Flow Port y mandarlo a la plataforma mediante un ejecutable desde el NI Linux Real-Time.