COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER EN INGENIERÍA INDUSTRIAL Y MÁSTER DE INDUSTRIA CONECTADA

TRABAJO FIN DE MÁSTER

DESARROLLO DE UN SISTEMA BASADO EN TECNOLOGÍAS IOT Y DLT PARA GARANTIZAR LA CALIDAD A LO LARGO DE LA CADENA DE SUMINISTRO APLICADO A TRANSPORTE REFRIGERADO

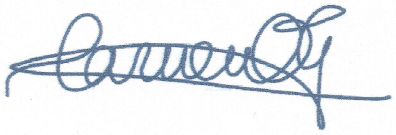
Autora: Carmen Olleros Guerrero

Directores:
Gregorio López López
Alonso Alfredo Rodríguez Rodríguez

Madrid

Julio de 2020

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título Desarrollo de un sistema basado en tecnologías IoT y DLT para garantizar la calidad a lo largo de la cadena de suministro aplicado a transporte refrigerado en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2019/2020 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.



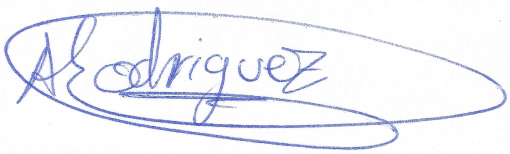
Fdo.: Carmen Olleros Guerrero Fecha: 13/ 07/ 2020

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Gregorio López López Fecha: 18/ 07/ 2020

Fdo.: Alonso Alfredo Rodriguez Rodriguez Fecha: 18/07/2020



COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER EN INGENIERÍA INDUSTRIAL Y MÁSTER DE INDUSTRIA CONECTADA

TRABAJO FIN DE MÁSTER

DESARROLLO DE UN SISTEMA BASADO EN TECNOLOGÍAS IOT Y DLT PARA GARANTIZAR LA CALIDAD A LO LARGO DE LA CADENA DE SUMINISTRO APLICADO A TRANSPORTE REFRIGERADO

Autora: Carmen Olleros Guerrero

Directores:
Gregorio López López
Alonso Alfredo Rodríguez Rodríguez

Madrid

Julio de 2020

A mi familia, por ayudarme y animarme ahora y a lo largo de toda la carrera.

Al Instituto Católico de Artes e Industrias por enseñarme la satisfacción que proporciona el esfuerzo y el trabajo duro.

Índice general

Índice de figuras

Siglas

am ante meridiem

AWS Amazon Web Services

BaaS Blockchain as a service

CO2 Dióxido de carbono

CoAP Constrained Application Protocol

DLT Distributed Ledger Technologies

etc etcétera

HAN Home Area Network

HP Hewlett Packard

HPE Hewlett Packard Enterprise

HTTP Hypertext Transfer Protocol

IIoT Industrial Internet of Things

IoT Internet of things

IP Internet Protocol

ISM Industrial Scientific and Medical

KPI Key Performance Indicator

LAN Local Area Network

LoRaWAN Long Range Wide Area Network

LPPAN Low Power Personal Area Network

LPWAN Low Power Wide Area Network

LTE Long Term Evolution

MaaS Messaging as a Service

MAN Metropolitan Area Network

MKR Maker

MNO Mobile Network Operator

MQTT Message Queueing Telemetry Transport

NBIoT Narrow Band Internet of Things

ODS Objetivos de Desarrollo Sostenible

PaaS Pay as a service

PAN Personal Area Network

PIC Programmable Integrated Circuit

pm post meridiem

QoS Quality of Service

SDK Software Development Kit

SIM Subscriber Identity Module

TCP Transport Control Protocol

TFM Trabajo de fin de máster

UDP User Datagram Protocol

URL Uniform Resource Locator

USB Universal Serial Bus

VoIP Voice over IP

WAN Wide Area Network

WLAN Wireless Local Area Network

Memoria

La aparición de la cuarta revolución industrial [1], definida por la unión del mundo físico, con el digital y el biológico, ha generado un modelo de negocio en el que las industrias se adaptan a las necesidades del cliente haciendo cambios en la cadena de valor de manera eficiente. El valor de este nuevo modelo reside en las comunicaciones centradas en las redes entre productores, clientes y sistemas; en la digitalización de información de distintos puntos de la cadena de valor así como la comunicación entre diferentes eslabones, y en la granularidad, flexibilidad e inteligencia en los procesos productivos, ajustables en tiempo real para adecuarse a las especificaciones del cliente. La personalización de la producción ya es una realidad.

Este contexto industrial muestra una evolución y modernización de las industrias de manera rápida e imprevisible, gracias a nuevas combinaciones de las numerosas tecnologías disponibles [2]. Este trabajo se centra en el sector industrial, especialmente en la cadena de suministro: la aplicación trata la garantía de la cadena de frío en transporte refrigerado, analizando los beneficios de una combinación concreta de tecnologías, IoT y *blockchain*. A continuación se explican brevemente estas dos tecnologías:

- Internet de las Cosas: Comúnmente conocido por las siglas en inglés IoT (*Internet of things*) hace referencia a la interconexión a través de internet de objetos cotidianos. Existe además una rama específica de esta tecnología dedicada a la conectividad de objetos industriales llamada *Industrial IoT* o *IIoT*. Algunas de las ventajas de IoT son la automatización, la obtención de datos y KPIs (*Key Performance Indicators*), el mantenimiento prevdictivo, el aumento de la productividad.
- Las tecnologías de registro distribuido son una clase de base de datos descentralizadas accesible desde diferentes nodos, es decir, hay varias copias de la información distribuidas entre los diferentes participantes. Cuando se realiza una operación la información se actualiza de manera sincronizada en todos los nodos. Son una clase de base de datos que no puede modificarse, semejante a un libro de cuentas. Es frecuente encontrarlo también por las siglas DLT (*Distributed Ledger Technology*). Lo que diferencia a las tecnologías de registro distribuido de las bases de datos generales es que cuando se introduce información nueva, en una DLT esta información debe ser validada antes de ser escrita.

Este trabajo de fin de máster trata de desarrollar una aplicación que surge de la unión de estas dos tecnologías. La aplicación consiste en garantizar la calidad a lo largo de la cadena de suministro en servicios de transporte refrigerado. La calidad se mide con la cadena de frío, que no debe romperse en ningún momento, manteniendo el producto en unos rangos de frío y humedad determinados.

Palabras clave: Internet de las Cosas, Blockchain, Tecnologías de Registro Distribuido, cadena de suministro, Arduino, SigFox, Azure

0.1. Introducción

Con esta base sobre las tecnologías a utilizar en el desarrollo del trabajo, se va a exponer a continuación el ámbito de la aplicación. Se ha decido desarrollar una aplicación enfocada a cadenas de suministro. Por cadena de suministro se entiende el proceso que involucra desde el actor que explota la materia prima hasta el cliente final. Este sector ha cambiado mucho en los últimos años y continúa evolucionando, de manera que aplicando la pareja de tecnologías establecidas a este mercado se espera dar ejemplo de cómo ayudar en la mejora de un proceso. Profundizando un poco más en la cadena de suministro se pueden encontrar diferentes eslabones y participantes. En el proceso se puede diferenciar entre la materia prima, los proveedores de materia prima, los productores, los distribuidores, los comerciantes (si aplica) y el cliente final. Este trabajo se centra en la parte logística que envuelve a la cadena de suministro, es decir, en el movimiento del producto. Se desarrolla una aplicación enfocada al transporte de mercancías refrigeradas, imponiendo unas limitaciones de temperatura y humedad en el proceso.

Derivado de todo lo anterior, la razón por la que se decide hacer el trabajo de fin de máster sobre el tema de Internet de las cosas y tecnologías de registro distribuido se debe a la creciente inclinación de diversos sectores por buscar soluciones o mejoras innovadoras a través de la implementación de estas tecnologías en sus procesos. Se decide hacer el caso de uso de estas tecnologías en el mundo de la logística y cadena de suministro, más específicamente de transporte refrigerado, debido a que es un servicio que está creciendo y adaptándose rápidamente a las necesidades de consumo actuales. Este sector llevaba años sin cambiar su modelo de negocio, pero desde hace unos años existe una mayor conectividad entre las propias empresas y entre empresas y consumidores. La logística ahora juega un papel fundamental. Además este sector se ha ido modernizando permitiendo una mayor trazabilidad de la mercancía y visibilidad para el cliente a lo largo de los envíos. Pero en el transporte de productos en cámara frigorífica debe asegurarse no solo su entrega si no que ésta se haga en buen estado. Y por tanto las soluciones que aportan visibilidad y trazabilidad en los envíos son buenas, pero se quedan un poco limitadas. En este trabajo se va a desarrollar una aplicación que permita trazar y ver el estado del pedido así como la temperatura y humedad a la que se ha realizado el mismo. De esta manera se puede demostrar que el transporte se ha realizado sin poner en riesgo la cadena de frío de la mercancía.

0.2. Definición del proyecto

El objetivo de este trabajo es desarrollar un caso de uso de la combinación de dos tecnologías: Internet de las Cosas y *blockchain* a través de contratos inteligentes. Desarrollar esta aplicación de manera práctica es una manera de dejar plasmado que es posible conectar ambas tecnologías y que el resultado es provechoso. En este caso se va a aplicar a la parte logística de la cadena de suministro de productos que requieran cadena de frío. Esto permitirá añadir al servicio trazabilidad y visibilidad.

Para conseguir desarrollar la aplicación correctamente y facilitar la comprensión de las tecnologías a utilizar se realiza primero un ejercicio de investigación y documentación. Se analizan las posibilidades IoT: placas *hardware* y sensores; los protocolos y tecnologías de comunicación y las principales plataformas *blockchain* del mercado.

0.3. Descripción del sistema

La aplicación que se quiere desarrollar se puede descomponer en cuatro bloques principales: el *hardware* que va a utilizar, la tecnología de comunicación que va a utilizar dicho *hardware* [3], la nube que va a recibir y procesar los datos y la plataforma *blockchain* que ingiere los datos. Para cada uno de estos bloques se ha hecho un estudio de las posibilidades existentes en el mercado, expuesto a continuación.

Respecto a las plataformas *hardware*, se han analizado las principales placas que permiten desarrollar proyectos de IoT, que implica una gran cantidad de dispositivos (por tanto se busca un precio asequible), bajo consumo de energía y que sea *open source*. La Figura 2.1 muestra las principales plataformas analizadas, dentro de las cuales las que más se adaptan a los requisitos de la aplicación son Arduino [4] y Particle.

	MICROCONTROLADOR Y PLACA				SOLO MICROCONTROLADOR	
Plataformas *Hardware*	ARDUINO		SONOFF®		PIC	ESPRESSIF
Modularidad	✓	✓	✓	✓	✗	✗
Módulo de conectividad *cloud*	✓	✓	✓	✓	✗	✗
Escalabilidad y versatilidad	Media alta	Alta	Baja	Alta	Media	Baja
Consumo	Bajo	Alto	Bajo	Bajo	Bajo	Bajo
Precio	€€	€€€	€€	€€	€	€

Figura 1. Tabla comparativa entre plataformas *hardware*.

Analizando el detalle de la placa de Particle más indicada para el proyecto; Particle IoT *starter kit*, se observa que cuenta con conectividad Wi-Fi y Bluetooth, no tiene conectividad a la red móvil. Esto puede suponer un problema ya que en el caso de uso de este proyecto el conjunto sensor más placa se instalaría en el contenedor refrigerado de un camión que se moverá por lo que es interesante que la tecnología proporcione cobertura metropolitana.

Por otra parte, de todas las placas de Arduino, se escoge la placa Arduino MKR Fox 1200. Esta placa tiene un módulo integrado de conectividad a la red SigFox que hace que esa placa sea una gran opción para desarrollar el caso de uso. Estas redes pertenecen a las redes LPWAN (*Low power wide area network*). Por todas las ventajas que ofrecen las placas de Arduino y lo conveniente que es la conectividad que ofrece la placa MKR Fox 1200, se decide trabajar con esta plataforma *hardware*. Además esta placa viene con un año de subscripción gratis a las redes de SigFox.

Para la aplicación de este trabajo y por lo general en proyectos de IoT se desea medir alguna variable de interés en la placa que se va a colocar, en este caso temperatura y humedad. Estas variables pueden obtenerse de dos maneras, con un sensor para cada variable o con un sensor que mida ambas variables. Se estudian los sensores más utilizados tanto para medir por separado como para medir ambas variables a la vez. En la Figura 2.3 se puede ver la comparación.

Se desea que se midan temperaturas negativas y positivas y un amplio rango de humedad relativa, preferiblemente el 100 %. Además se buscar que la variación de lectura de la humedad relativa esté dentro del rango del 5 % respecto a la medida real, y que la variación de la medida de temperatura no supere el grado centígrado respecto a la temperatura real. El sensor que mejor cumple con estas características de todos los vistos en la Figura 2.3 es el DHT22 [5].

	TEMPERATURA			HUMEDAD	TEMPERATURA Y HUMEDAD			
SENSORES	LM35	TMP36	TC74	SHT7x	DHT11		DHT22	
Rango medida	2 a 150°C	-40 a 150°C	-40 a 125°C	0 a 100%	0 a 50°C	20 a 80%	-40 a 150°C	0 a 100%
Precisión	± 0,5°C	± 2°C	± 2°C	± 3%	± 2°C	± 5%	± 0,5°C	± 2%
Precio	0,60 €	1,5 €	3 €	+20 €	5 a 10€		15 a 20€	

Figura 2. Tabla comparativa entre sensores.

A continuación, en la Figura 3.4 se aprecia el cableado de la placa y el sensor.

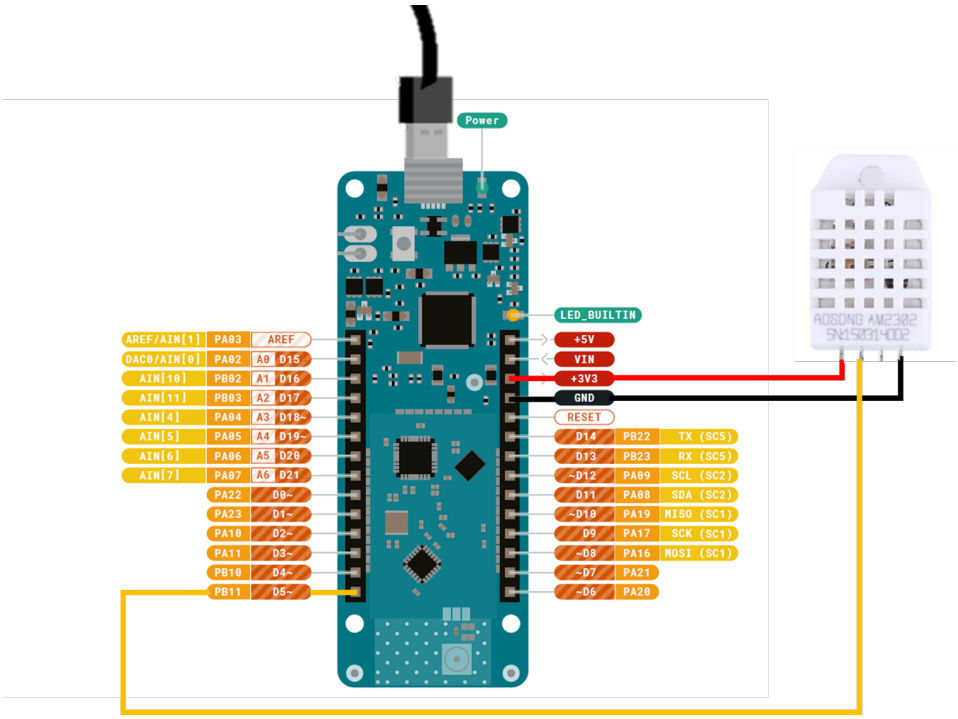


Figura 3. Cableado del sensor en la placa de Arduino.

El siguiente bloque de la aplicación es la tecnología que va a usar la placa para comunicarse con la nube, en este trabajo: SigFox. Para crear una aplicación que pueda ser usada en un ámbito geográfico grande (por ejemplo, que cubra distancias entre ciudades, incluso países) las redes más apropiadas son las redes WAN: *Wide Area Network*. Son de especial interés las redes LPWAN, *Low Power Wide Area Network*, que son redes que se caracterizan por proporcionar largo alcance, bajo consumo y coste y baja tasa de transmisión de datos, lo que los hace muy apropiados para sistemas IoT. Los requisitos para usar redes LPWAN se pueden resumir en: multitud de dispositivos conectados, operados por batería y repartidos por grandes áreas de superficie, requisitos que están perfectamente alineados con el caso de uso.

Dentro de las redes LPWAN se pueden distinguir dos grupos claramente diferenciados. Las redes que operan en las bandas licenciadas operadas por operadores de telefonía [6]o MNO (*Mobile Network Operator*y las redes que no son operadas por MNO. Dentro de las redes MNO destacan LTE-M y NBIoT y dentro de las no MNO destacan LoRaWAN y SigFox. A continuación se exponen las características principales de cada una de ellas:

- LTE-M: Definido en los *releases* 13 y 14 de LTE del 3GPP. Las redes que usan este estándar tienen una conexión muy eficiente con gran cobertura en zonas interiores y subterráneas. Proporcionan baja latencia permitiendo desarrollar aplicaciones que reciban datos en tiempo real, y soporta mobilidad. Algunos ejemplos de aplicaciones que usan este tipo de

comunicaciones son comunicaciones vehiculares, sistemas de emergencia, monitores de pacientes, etiquetas de logística, etc.

- NB-IoT: Este tipo de redes son similares a las LTE-M. Una sola célula de una red NB-IoT puede conectar hasta 50000 dispositivos IoT. En estas redes los datos se agrupan antes de ser enviados por la red (*batch messaging*), aumentando la latencia de la red y los dispositivos conectados a ellas deben de ser estáticos, no admiten movimiento. Por tanto este tipo de redes no es el más adecuado para el proyecto de transporte refrigerado de este trabajo.

- LoRaWan viene de *Longe Range Wide Area Network*, en español red WAN de largo alcance (>20km). LoRaWan es responsable de unificar diferentes dispositivos LoRa para administrar sus canales y parámetros de conexión: canal, ancho de banda, cifrado de datos, etc para así enviar y recibir información. Para integrar LoRa, no es necesaria una tarjeta SIM pero si es necesario tener un punto de acceso específico de LoRa. Plataformas como Arduino o Raspberri Pi cuentan con *shields* o módulos que se pueden acoplar fácilmente a una placa.

- SigFox es una empresa privada líder en soluciones IoT. La tecnología se caracteriza por un muy bajo consumo, muy bajo precio y muy baja tasa de transmisión. Esta empresa cobra alrededor de un euro por dispositivo conectado a su red al año. Ofrece una serie de funcionalidades de control gratis, como son la seguridad del servicio, el mantenimiento de las redes, etc... Además el propietario del dispositivo tiene acceso a una plataforma: SigFox Backend, desde la cual puede manejar sus dispositivos, ver los mensajes enviados, configurar respuestas y acciones, localizar los dispositivos en el mapa, y más operaciones, que funciona las 24 horas del día, 7 días a la semana. Además, SigFox tiene una nube en la que se almacenan temporalmente todos los datos que llegan de los sensores, pero la idea es que el usuario configure una respuesta para enviar esos datos a un lugar en el que les saque provecho.

	Redes LPWAN	
	LTE-M	SigFox
Propietario	Libre	Privado
Espectro	Requiere licencia	No requiere licencia
Alcance	2-5km urbanos	3-10km urbanos
Consumo	Medio	Bajo
Precio	Medio	Bajo

Figura 4. Tabla comparativa entre LTE-M y SigFox.

De las redes MNO únicamente sería viable realizar el proyecto con LTE-M ya que NB-IoT requiere que el dispositivo este estático, pero para trabajar con una LTE-M hay que pagar tanto la tarjeta SIM como la subscripción a la red. Para utilizar LoRa se podría utilizar la placa Arduino MKR 1300, pero debido a que es una tecnologñia menos madura para el desarrollo de aplicaciones industriales se decide no usar. Lo más lógico es utilizar las redes de SigFox ya que la placa escogida está preparada y la subscripción del primer año es gratuita. En la Figura 14Antes de poder enviar datos hay que registrar el dispositivo en el *backend* de SigFox. El *frontend* permite gestionar los dispositivos. Cada dispositivo es único y en cuanto se registra se activa la subscripción inicial gratuita. El dispositivo registrado para el desarrollo es el 1CF5D8. Una vez registrado se puede proceder a enviar datos.

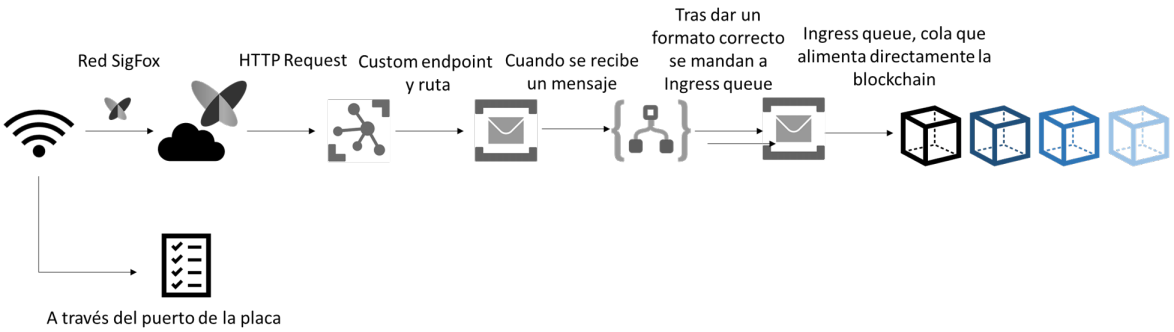


Figura 5. Trayectoria de los datos a lo largo de la aplicación.

Para tener una idea más visual del proceso que se está llevando a cabo, en la Figura 3.5, se pueden ver las principales conexiones que se deben hacer para enviar los datos desde el sensor físico hasta la *blockchain*. La primera conexión ya se ha realizado que es entre el sensor y la nube de SigFox. La rama inferior del esquema hace referencia a los datos que se piden a través del Monitor Serie para comprobar que son iguales a los recibidos en el *backend*.

La tercera parte del análisis de la aplicación se corresponde con la plataforma *cloud* a la que se van a mandar los datos. Interesa utilizar una plataforma *cloud* debido a la accesibilidad, rentabilidad y seguridad que aporta. Existen varias soluciones comerciales que se podrían utilizar, algunas de las más frecuentes para trabajar con dispositivos IoT son las siguientes: Google Cloud Platform, Particle, IBM Watson IoT. Además de estas, destacan las dos plataformas con las que SigFox tiene preparada una conexión facilmente configurable:

- AWS IoT : Una ventaja de AWS es la simplicidad de sus despliegues dentro del amplio abanico de soluciones que ofrece. Además permite realizar proyectos de *software* en múltiples lenguajes: Java, Python, NodeJS, etc. Este servicio de AWS permite “conectar, recopilar, almacenar y analizar datos de dispositivos IoT”. Ofrece dos *software* para los dispositivos IoT: FreeRTOS y AWS IoT Greengrass, que permiten comunicar, administrar y gestionar dispositivos de manera segura. Además ofrece también servicios de control y conectividad y servicios de análisis [7].

- Azure Iot Hub: Una ventaja determinante de Azure es que se integra con el software de Microsoft [8]. Por tanto muchas empresas e instituciones que utilizan Microsoft para trabajar (outlook, office 365, ...) deciden utilizar Azure directamente por las ventajas de capacidad, reducción de costes y mayor rendimiento que supone. Este servicio esta pensado para poder conectarse casi con cualquier dispositivo. Desde esta plataforma en la nube se pueden administrar y configurar infinidad de dispositivos [9]. Permite además una integración sencilla con otros módulos de Azure.

Ambas son soluciones muy potentes que permiten llevar a cabo el caso de uso, pero se decide trabajar con IoT Hub por tratarse de una solución líder, con la que ya se tenía experiencia..

Antes de proceder a configurar la redirección de los mensajes recibidos en SigFox, hay que preparar la plataforma que los va a recibir. en este caso Azure IoT Hub [9].

Se debe tener un directorio en Azure con una subscripción activa. Dentro de ese directorio se crea un grupo de recursos para albergar específicamente el IoT Hub de la aplicación. El módulo Iot Hub desplegado tiene el nombre de IotHub-TFM. Una vez desplegado, se puede realizar la conexión con SigFox. Esto se hace a través de la llave: *Connection string—primary key*.

En SigFox, se prepara un *callback* que es una acción que se toma cuando se reciben datos. En la configuración de este *callback* se introduce la llave de conexión de IoT Hub (*Connection string—primary key*) y se describe el codigo JSON que se desea enviar a Azure. Con tan solo estas dos configuraciones, la conexión y el código queda configurado el envío de datos a IoT Hub.

A continuación se incluye el código JSON que se manda a IoT Hub. En él se pide que se envíen el identificador del dispositivo, los datos del usuario (en hexadecimal), el sello temporal (en formato UNIX), el número de secuencia del mensaje, el identificador del tipo de dispositivo y las variables de interés: temperatura y humedad recibidas del sensor.

```
{
"DeviceID" : "{device}",
"data" : "{data}",
"time" : "{time}",
"seqNumber" : "{seqNumber}",
"deviceTypeId" : "{deviceTypeId}",
"humidity" : "{customData#hum}",
"temperature" : "{customData#temp}"
}
```

Por último, el cuarto bloque del análisis se corresponde con la plataforma en la que se va a desarrollar la *blockchain* en la nube, también conocido como *Blockchain as a Service* (Baas) [10]. Este tipo de plataformas ofrecen la infraestructura y soporte para garantizar el correcto funcionamiento de la *blockchain*. Es una manera fácil de generar aplicaciones *blockchain*, cada vez más extendida, que a la larga va a permitir que las tecnologías DLT se vayan integrando y normalizando. A pesar de que existen muchas plataformas en las que se pueden desplegar *blockchains* enfocadas a IoT (como pueden ser SAP *Cloud-Based Blockchain Service Platform*, IMB *Blockchain Platform*, *Blockchain on* AWS u Oracle *Blockchain Cloud Service*) se va a trabajar con Azure *Blockchain Workbench* debido a la facilidad de integrar este servicio con el resto de la aplicación. Este módulo de Azure se integra fácilmente con muchos otros recursos de Azure (como Azure IoT Hub), favoreciendo la unión de diferentes tecnologías. Además Azure y por tanto Azure *Blockchain Workbench* está pensado para no tener que programar, es decir con las nociones básicas de las diferentes tecnologías se pueden construir aplicaciones de bastante dificultad técnica. La propia plataforma se encarga de realizar los tecnicismos por debajo de la interfaz del usuario.

En resumen, la Figura 3.1 muestra los cuatro bloques con las tecnologías principales que se van a usar en el desarrollo de la aplicación.

Aunque las cuatro tecnologías principales son las mostradas en la Figura 3.1 desde el IoT Hub hasta Azure *Blockchain Workbench* es necesario atravesar por diferentes módulos de Azure. Este proceso permite el transporte de datos y su modificación para adecuarlos al formato que es capaz de leer la *blockchain*. Este proceso se hace tomando de referencia el repositorio de Github "*Delivering Data from IoT Hub to Azure Blockchain Workbench*"[11]. Volviendo a la Figura 3.5 se pueden apreciar los módulos por los que pasan los datos antes de llegar a la aplicación *blockchain*.

Se crea un *Service Bus* llamado servicebusTFM. Este módulo es una plataforma de mensajería en la nube también conocido como mensajería como servicio (MaaS). Permite enviar los mensajes recibidos en IoT Hub a una aplicación lógica que cambia el formato del mensaje

HARDWARE:	Arduino MKR Fox 1200 + DHT22
RED DE COMUNICACIÓN:	SigFox Backend
SOLUCIÓN CLOUD:	Azure IoT Hub
PLATAFORMA BLOCKCHAIN:	Azure Blockchain Workbench

Figura 6. Tecnologías elegidas para desarrollar la aplicación del trabajo.

recibido. Para poder realizar este envío de datos se debe crear una cola dentro de servicebusTFM, y generar una ruta en IoTHub-TFM que conecte los datos recibidos con dicha cola. La cola en cuestión se ha llamado "*myqueue*".

Se despliega un módulo llamado *Logic App* que es el que va a tratar los datos y prepararlos para adecuarlos al formato de *blockchain*. Este servicio permite captar datos de multitud de aplicaciones, pueden ser fuentes ajenas a Azure y de una manera dinámica, sin necesidad de escribir códigos complejos, editar y crear flujos de trabajo [12]. Para la aplicación, el servicio de *Logic App* va a comenzar con un detonante que una vez cada minuto revisa si hay mensajes nuevos en la cola *myqueue*. Previo a continuar se debe desplegar Azure *Blockchain Workbench*. Este servicio realmente está formado por un cúmulo de módulos de Azure, ya preparados para funcionar. Entre todos los módulos desplegados, se encuentra un servidor SQL: db-cjzdec-tfm. Siguiendo los pasos del repositorio de GitHub mencionado más arriba, se debe lanzar una *query* o consulta para guardar una serie de procedimientos que deben quedarse almacenados en la base de datos SQL. El procedimiento de interés es GetContractInfoForDeviceID, que para cada dispositivo va a obtener la información del *smart contract*. Pero para que el servicio de [12] pueda acceder a esta base de datos, debe existir una conexión local al servidor SQL. Es decir hay que establecer una puerta de enlace al servidor SQL en el ordenador desde el que se vaya trabajar el servicio. Para ello existe una aplicación llamada *On-premises data gateway*, que permite configurar dicho enlace rápidamente siguiendo las instrucciones de Azure.

Con Azure *Blockchain Workbench* desplegado y el *gateway* preparado se puede proceder a desarrollar el flujo que van a seguir los datos en *Logic App*.

- Comprobar cada minuto si hay algún mensaje nuevo en la cola *myqueue*, si lo hay, dicho mensaje se saca de la cola y entra en la *Logic App* para ser editado.
- Parsear los datos para poder leer el mensaje recibido.
- Obtener en función del ID del dispositivo, la información del Smart Contract activo. Esto se hace con un módulo de SQL que busca dentro de los procedimientos almacenados en la base de datos. Aquí entra en juego el *gateway* instalado.
- Se vuelven a parsear los datos, se preparan para ser introducidos en *blockchain*.
- Se inicializa una variable, llamada *RequestID*, con ella se genera un id para poder trackear más adelante el proceso.
- Se prepara el sello temporal en varios pasos para dejarlo en tiempo UNIX.

- Ahora el mensaje está preparado, únicamente falta meterlo en la cola de *blockchain*. Este envío es automático, la manera en la que se despliega Azure *Blockchain Workbench* hace que todo mensaje que entra en la *Ingress Queue* sea ingerido por la *blockchain*. Esta cola está dentro del *Service Bus* que se crea al desplegar Azure *Blockchain Workbench* y por tanto las conexiones están hechas. Todo mensaje que llega a esta cola debería ser cogido por el contrato activo pertinente. El módulo final es un módulo de envío a *Service Bus*.

El código del mensaje que se envía se rellena con las variables trabajadas, tal y como se muestra en la Figura 3.15.

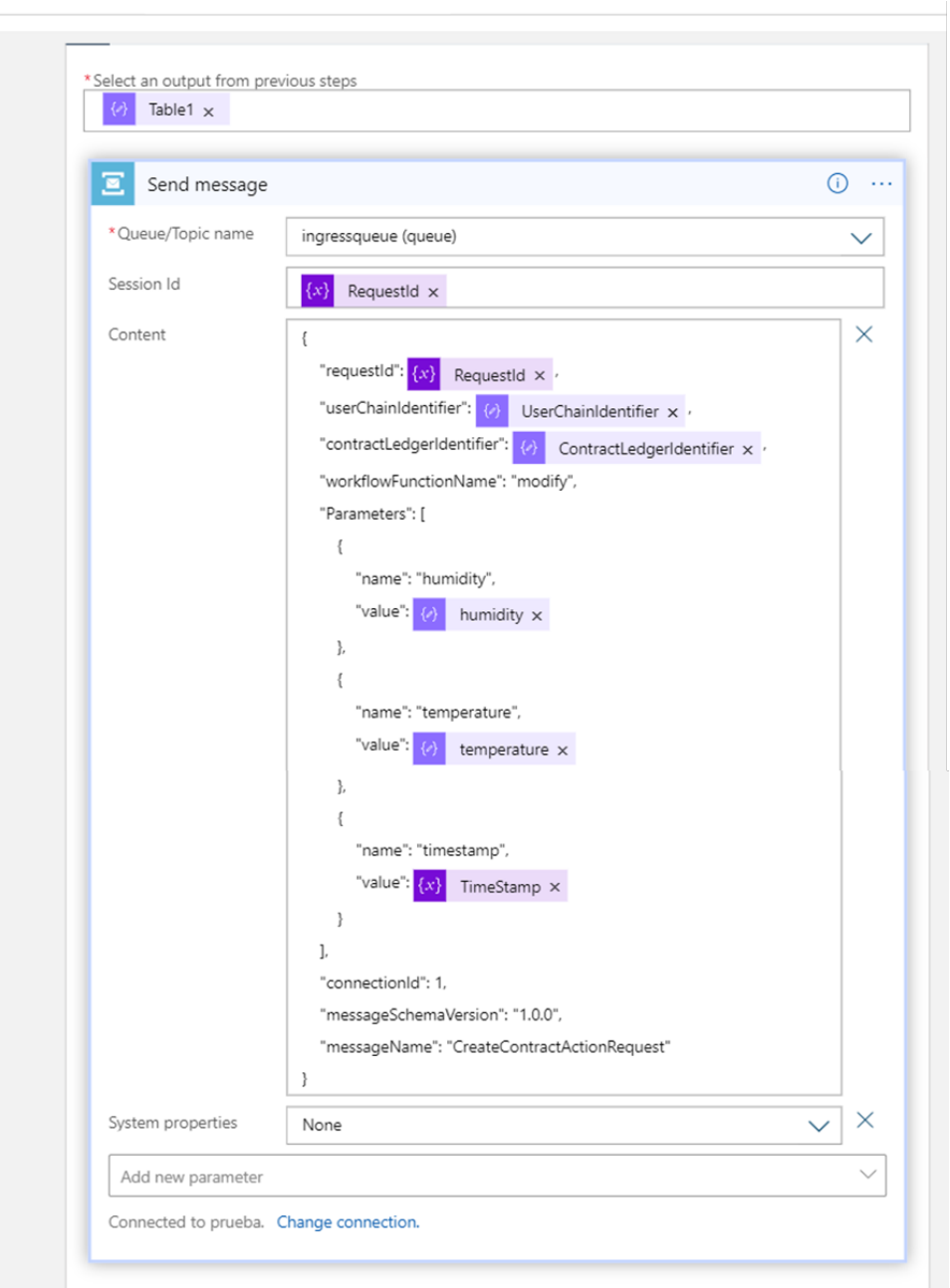


Figura 7. Módulos de envío a *blockchain*.

En *Visual Studio Code* se prepara el contrato que se va a desplegar en la aplicación *blockchain*. Antes de mostrar el contrato se va a exponer qué se quiere conseguir y qué participantes involucra. La máquina de estados del contrato se muestra en la Figura 3.17. En ella se muestran los posibles cambios de estado del contrato.

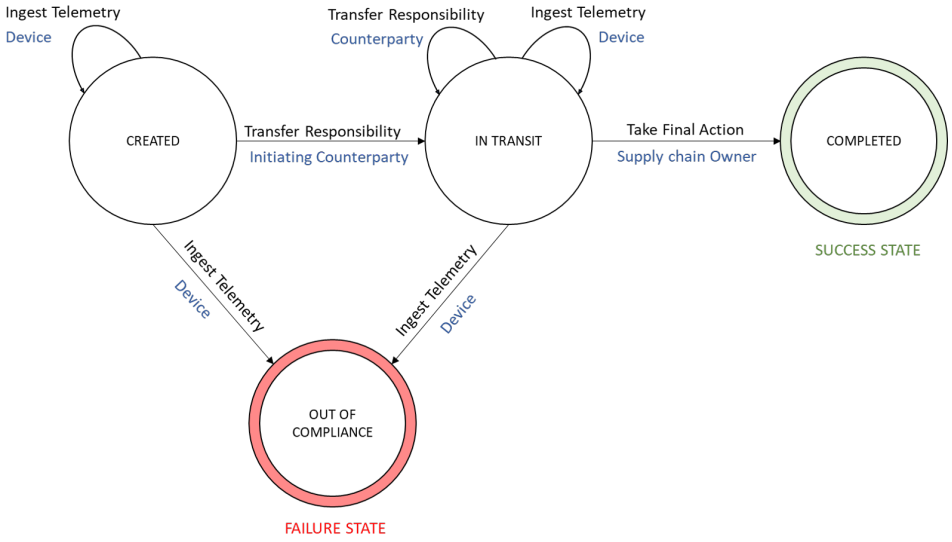


Figura 8. Máquina de estados del contrato.

Los actores involucrados en el proceso son:

- El usuario principal, ya visto más arriba EmpresaTFM@carmenolleroshotmail.onmicrosoft.com.
- El usuario que se corresponde con el dispositivo device@carmenolleroshotmail.onmicrosoft.com.
- Un usuario que actúa como si fuese el transportista transport@carmenolleroshotmail.onmicrosoft.com
- El correo personal carmen.olleros@hotmail.com que se utiliza como observador y auditor del proceso.

Para el caso de uso de este trabajo interesa guardar los datos de telemetría. Se busca que estos datos se vayan guardando de tal manera que estén asociados a cada contenedor, es decir en función del sensor que los manda en un histórico que se pueda ver después almacenado en la base de datos asociada a la *blockchain*. Este histórico de datos es el que va a permitir tener una trazabilidad y una mayor transparencia de lo que ocurre a lo largo del proceso. Con estos datos también es con lo que se puede demostrar que se ha cumplido la cadena de frío. Esto interesa tanto a la empresa encargada del transporte, como al proveedor del producto refrigerado, al cliente final y a cualquier empresa auditora, de seguros o de regulación del producto transportado.

0.4. Resultados

Tras configurar el *hardware*, los mensajes se reciben correctamente en SigFox. Esto verifica que la placa funciona correctamente, está bien alimentada, y que el cableado del sensor se ha hecho correctamente. Los mensajes que llegan a SigFox contienen la siguiente información: el sello temporal (momento en que es recibido el mensaje), los datos de temperatura y humedad enviados, la calidad de la señal, el estado del *callback* y la ubicación desde la cual se ha enviado el mensaje.

En SigFox también se configura el siguiente paso (la conexión con IoT Hub), y se vuelven a mandar datos. Dichos datos enviados desde la placa, ahora llegan a SigFox, pero también son redirigidos a Azure IoT Hub. Esta conexión también es exitosa ya que en IoT Hub, el sensor aparece listado. Del IoT Hub los mensajes pasan a una cola desde la cual toma los datos la *Logic App*, aplicación que modifica los datos para darles el formato apropiado para *blockchain*.

A continuación, en la figura Figura 4.5 se muestra cómo el flujo creado en el servicio *Logic App* funciona correctamente. Se visualiza como cada bloque del flujo tiene una señal de éxito en la esquina superior derecha. En cada uno de ellos se puede pinchar y ver los detalles del proceso. Son de especial interés el primer módulo del flujo, en el que se toman los mensajes de la cola, el módulo que obtiene en función del identificador del dispositivo la información del contrato y el último módulo que ingresa los datos en la cola de la aplicación *blockchain*. En el detalle del flujo de *Logic App* de la Figura 4.5 se pueden ver los tres módulos indicados y como todos ellos se han realizado exitosamente.

Figura 9. Detalle de un flujo de trabajo.

En el primer módulo de la *Logic App* se toma correctamente el mensaje de la cola *myqueue*. Este paso confirma que IoT Hub manda correctamente los datos a la cola y que el enlace entre la cola y este servicio de flujos de trabajo funciona correctamente también.

El siguiente módulo que es interesante entender corresponde al módulo que obtiene en función del ID del dispositivo la información del contrato activo. Es decir es capaz de sacar satisfactoriamente de *blockchain* qué contrato está activo y todas sus características.

El último módulo que prepara el mensaje y lo manda a la cola de ingreso de *blockchain* también es muy importante, ya que hace el enlace entre Azure *Logic App* y la *blockchain* Este módulo al igual que el resto funciona correctamente, el envío parece que se hace bien y si se miran las métricas de la cola de salida, se ve cómo se reciben mensajes.

Realmente, en la cola de salida, se puede ver como entran mensajes pero no salen, en el momento en el que mejor funcionó fue a las 3:41am del 1 de julio que tras enviar datos a la cola se ve cómo hay un amago de envío, pero por lo general los mensajes mandados no parece que salgan de esta cola. En la Figura 4.9 se puede visualizar a la derecha el envío correcto de mensajes a las 3:41am y en la derecha una gráfica que muestra el *Service Bus* de *blockchain* con los mensajes de entrada en azul y los de salida en naranja. A las 3:41am se puede ver que la cola de salida emite 7 mensajes.

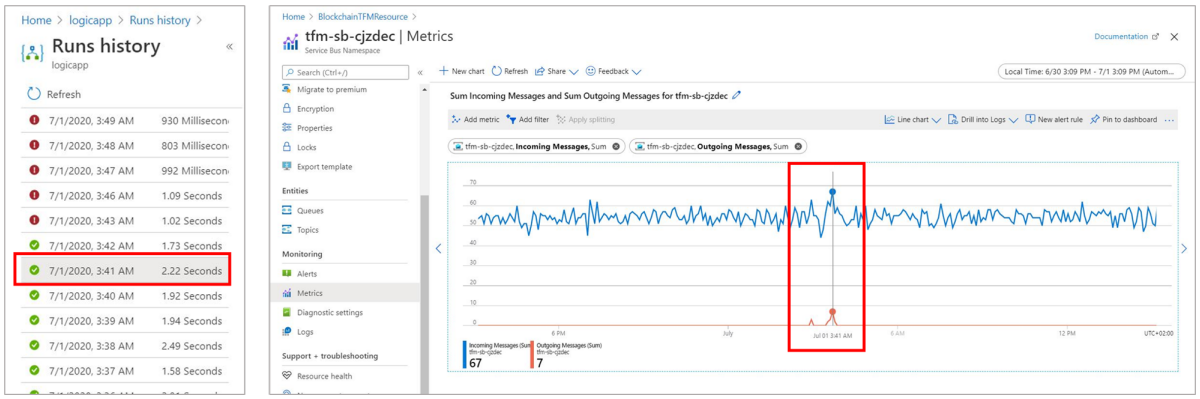


Figura 10. Obtención de la información del *smart contract*.

Por último para poder trabajar el *smart contract* se busca una plataforma alternativa a la aplicación web de Azure *Blockchian Workbench* (debido a que la subscripción con la que desarrolla el trabajo se acaba y deja de poder probarse). Por recomendación de los directores se utiliza Remix una plataforma en la que se pueden probar contratos de manera rápida. Gracias a esta plataforma se puede continuar probando el contrato hasta comprobar que el funcionamiento es el deseado.

0.5. Conclusiones

El objetivo de este trabajo era llevar a cabo una prueba de concepto de una aplicación que monitorice la cadena de frío de transporte refrigerado para asegurar su calidad utilizando IoT, *cloud* y DLT Este objetivo se ha cumplido.

Comprobando las diferentes metas para llegar al objetivo principal, se tiene el estudio de las tecnologías IoT y *blockchain*. Se estudian las tecnologías y protocolos elegidos en el desarrollo de la aplicación en profundidad, especialmente enfocado a su aplicación en la parte logística de la cadena de suministro de productos que requieran cadena de frío. Los bloques analizados son: las plataformas *hardware*, las tecnologías y protocolos de comunicación, la plataformas en la nube y las principales plataformas *blockchain*.

Haciendo especial foco en el mundo logístico, estas tecnologías pueden suponer un cambio en el modelo de negocio importante. Por un lado, *blockchain* permite eliminar intermediarios, ya que no se necesita una tercera parte que verifique los datos. Por otro, IoT aporta en las operaciones y movimientos visibilidad y trazabilidad, quedando registradas automáticamente en la cadena de bloques. Además en el caso desarrollado en este trabajo, las diferentes partes involucradas en la cadena de suministro pueden visualizar lo que está ocurriendo en tiempo real, eliminando complejidad, fomentando la buena praxis con mayor rigor en el control de calidad.

Respecto al desarrollo de la aplicación, se demuestra que es posible combinar una placa de Arduino, conectividad SigFox, y diferentes módulos de Azure hasta llegar a la plataforma de *blockchain*. Debido a que Arduino es una plataforma con la que se ha trabajado más a lo largo de la carrera y máster, fue rápido configurarlo. La placa se debe de alimentar correctamente (en este caso se usa como fuente el ordenador), el sensor debe de conectarse con el cableado correcto y se debe preparar el código que permite enviar los datos a SigFox.

SigFox y Azure han supuesto el mayor reto a la hora de desarrollar este TFM pero con ayuda de los directores y siguiendo las instrucciones de diferentes repositorios se ha llegado muy

lejos. En el momento de realizar la conexión con SigFox, me he podido apoyar en una práctica del curso *IoT-Cloud Communications*, donde se hacía una práctica similar, donde también se mandaban datos de Arduino a Azure a través de SigFox. Se registra la placa de Arduino con la subscripción y se configura el *callback* a Azure IoT Hub.

En Azure se utilizan módulos diferentes a los de la práctica del curso ya que se deben de preparar los datos para que estos puedan ser reconocidos por la aplicación *blockchain*. Destaco lo que he aprendido respecto a parsear datos. Me parece muy interesante la facilidad que da Azure *Logic App* para realizar estos cambios.

Se ha conseguido la conexión de todas las plataformas y módulos. No se ha conseguido recibir los datos del sensor en *blockchain*, el último enlace entre Azure *Logic App*, la cola de entrada de *blockchain* y la publicación de los datos ha dado algún problema a la hora de entrar en el *Smart Contract*. No se ha podido depurar y solucionar este problema, pero sí se ha validado el *smart contract* usando Remix.

Hacer un TFM de desarrollo ha supuesto un reto personal, ya que en varias ocasiones me he encontrado frente a un problema que no sabía resolver pero con dedicación y esfuerzo, encontraba el problema, lo conseguía resolver y avanzaba con el desarrollo.

Personalmente, haber realizado este trabajo me ha permitido tener un conocimiento más profundo de los protocolos y tecnologías que rodean a Internet de las cosas y a *blockchain*. Me ha permitido mejorar mis conocimientos de Python y sobre todo he tenido la oportunidad de aprender Solidity, el lenguaje de los contratos inteligente. He aprendido a no rendirme ante un problema que no entiendo y he mejorado mis capacidades de investigación.

0.6. Trabajos futuros

Sería interesante incluir la ubicación del dispositivo desde el que se envían los datos. Como se ha explicado anteriormente, no es posible hacerlo todo en el mismo *callback* de SigFox pero sí que podría hacerse desde otro separado, y en Azure unir la información obtenida de ambos mensajes (los datos y la ubicación). Para ello, es importante encontrar lo que está impidiendo que la aplicación funcione correctamente.

Otra mejora de la que se beneficiaría mucho la aplicación es una mejora en el contrato inteligente. Se debería añadir al histórico de los datos no sólo de qué camión provienen si no quien está conduciendo este camión en ese momento, por ejemplo, para poder avisar directamente a esa persona en caso de salirse de rangos la temperatura o humedad. También es interesante añadir la persona que lleva la mercancía como métrica que pueda ser estudiada por la empresa que contrata el servicio.

Una vez que la aplicación funciona con estas mejoras, realmente podría llegar a implementarse en el contenedor de un camión, ya que hasta el momento todo se hace con un sensor estático. Sería muy interesante comprobar el funcionamiento de la aplicación en movimiento. Para llegar al punto de colocar el sensor en un camión habría que preparar un encapsulado para el conjunto sensor mas placa que fuese fácil de colocar dentro del contenedor frigorífico.

Tras realizar la prueba con un único sensor, el objetivo siguiente sería organizar un estudio económico de la solución para escalar el proyecto y desplegarlo en una flota de camiones. Con el estudio preparado se podría llegar a un acuerdo o vender la aplicación a una empresa, y comercializar la solución desarrollada en el trabajo.

Abstract

The fourth industrial revolution [1], defined by the union of the physical world, with the digital and biological worlds, has generated a business model in which industries adapt to customer needs by making changes in the value chain efficiently. The value of this new model lies in network-centred communications between producers, customers and systems; in the digitalisation of information from different points in the value chain as well as communication between different links, and in the granularity, flexibility and intelligence of the production processes, which can be adjusted in real time to suit customer specifications. Production customization is already a reality.

This industrial context shows an evolution and modernization of the industries in a fast and unpredictable way, thanks to new combinations of the numerous available technologies [2]. This work focuses on the industrial sector, especially on the supply chain: the application deals with the guarantee of the cold chain in refrigerated transport, analysing the benefits of a specific combination of technologies, IoT and blockchain. These two technologies are briefly explained below:

- Internet of Things: refers to the interconnection of everyday objects through the Internet. There is also a specific branch of this technology dedicated to the connectivity of industrial objects called Industrial IoT or IIoT. Some of the advantages of IoT are automation, data acquisition and KPIs (Key Performance Indicators), predictive maintenance, increased productivity.

- Distributed Ledger Technologies are a kind of decentralized database accessible from different nodes, that is, there are several copies of the information distributed among the different participants. When an operation is performed, the information is updated in a synchronized manner in all the nodes. They are a type of database that cannot be changed, similar to a ledger. It is also often found by the acronym DLT. What differentiates distributed logging technologies from general databases is that when new information is introduced, in a DLT this information must be validated before being written.

This Master Thesis seeks to develop an application that arises from the union of these two technologies. The application consists of guaranteeing quality throughout the supply chain in refrigerated transport services. Quality is measured by the cold chain, which must not be broken at any time, keeping the product in certain cold and humidity ranges.

Key Words: Internet of Things, Blockchain, Distributed Ledger Technologies, Supply Chain, Arduino, SigFox, Azure

---

0.7. Introduction

On this basis of the technologies to be used in the development of the work, the scope of application will be set out below. It has been decided to develop an application focused on supply chains. Supply chain can be understood as the process that involves from the actor who exploits the raw material to the final client. This sector has changed a lot in recent years and continues to evolve, so by applying the technology mix chosen to this market it is hoped to set an example of how to help improve a process. Furthermore, in supply chain different links and participants. In the process one can differentiate between the raw material, the raw material suppliers, the producers, the distributors, the traders (if applicable) and the final customer. This thesis focuses on the logistic part that involves the supply chain. An application is developed focused on the transport of refrigerated goods, imposing temperature and humidity limitations in the process. From all the above, the reason why it was decided to do the final work of the master's degree on the subject of the Internet of things and distributed ledger technologies is due to the growing inclination of various sectors to seek innovative solutions or improvements through the implementation of these technologies in their processes. It was decided to make the case for the use of these technologies in the world of logistics and supply chain, more specifically refrigerated transport, because it is a service that is growing and adapting rapidly to current consumer needs. This sector had not changed its business model for years, but for a few years now there has been greater connectivity between the companies themselves and between companies and consumers. Logistics now plays a fundamental role. In addition, this sector has been modernised, allowing greater traceability of goods and visibility for the customer throughout the shipments. However, when transporting products in cold storage, it is necessary to ensure not only that they are delivered but also that they are in good condition. And therefore the solutions that provide visibility and traceability in shipments are good, but remain a little limited. In this work we are going to develop an application that will allow us to trace and see the status of the order as well as the temperature and humidity at which it has been made. In this way it can be demonstrated that the transport has been carried out without putting the cold chain of the goods at risk.

0.8. Project Definition

The aim of this thesis is to develop a case of the use of the combination of two technologies: Internet of Things and blockchain through smart contracts. Developing this application in a practical way is a way to show that it is possible to connect both technologies and that the result is profitable. In this case it will be applied to the logistics part of the supply chain of products that require a cold chain. This will add traceability and visibility to the service.

In order to develop the application correctly and facilitate the understanding of the technologies to be used, a research and documentation exercise is first carried out. The IoT possibilities are analyzed: hardware boards and sensors; communication protocols and technologies and the main blockchain platform in the market.

0.9. System description

The application to be developed can be broken down into four main blocks: the hardware platform, the communication technology, the cloud that will receive and process the data and the platform that will ingest the data. For each of these blocks, a study has been made of the existing possibilities on the market, as set out below.

Regarding the hardware platforms, the main boards that allow the development of IoT projects have been analysed. This involves a large number of devices (its important that the price is affordable), low energy consumption and open source. The Figure 2.1 shows the main platforms analyzed, within which the ones that are most adapted to the requirements of the application are Arduino [4] and Particle.

	MICROCONTROLADOR Y PLACA				SOLO MICROCONTROLADOR	
Plataformas *Hardware*	ARDUINO		SONOFF®		PIC	ESPRESSIF
Modularidad	✓	✓	✓	✓	✗	✗
Módulo de conectividad *cloud*	✓	✓	✓	✓	✗	✗
Escalabilidad y versatilidad	Media alta	Alta	Baja	Alta	Media	Baja
Consumo	Bajo	Alto	Bajo	Bajo	Bajo	Bajo
Precio	€€	€€€	€€	€€	€	€

Figure 11. Hardware platforms comparison.

Analyzing the detail of the Particle board most suitable for the project; Particle IoT starter kit, it is observed that it has Wi-Fi and Bluetooth connectivity, it does not have connectivity to the mobile network. This can be a problem because in the application of this project the sensor and board would be installed in the refrigerated container of a truck that will move so it is interesting that the technology provides metropolitan coverage.

On the other hand, from all the Arduino boards, the chosen one is the Arduino MKR Fox 1200 board. This board has an integrated SigFox network connectivity module that makes this board a great option for developing the use case. These networks belong to the LPWAN (Low power wide area network). Because of all the advantages offered by Arduino's boards and how convenient is the connectivity offered by the MKR Fox 1200 board, it was decided to work with this hardware platform. This board also comes with a one year free subscription to SigFox networks.

For the application of this thesis and usually in IoT projects, we want to measure some variable of interest in the board to be placed, in this case temperature and humidity. These variables can be obtained in two ways, with a sensor for each variable or with a sensor that measures both variables. The most used sensors are studied both to measure separately and to measure both variables at the same time. The comparison can be seen in the Figure 2.3.

	TEMPERATURA			HUMEDAD	TEMPERATURA Y HUMEDAD			
SENSORES	LM35	TMP36	TC74	SHT7x	DHT11		DHT22	
Rango medida	2 a 150°C	-40 a 150°C	-40 a 125°C	0 a 100%	0 a 50°C	20 a 80%	-40 a 150°C	0 a 100%
Precisión	± 0,5°C	± 2°C	± 2°C	± 3%	± 2°C	± 5%	± 0,5°C	± 2%
Precio	0,60 €	1,5 €	3 €	+20 €	5 a 10€		15 a 20€	

Figure 12. Sensor comparison.

The range of measurement should be negative and positive temperatures and a wide range of humidity are measured, preferably 100 %. In addition, the variation of the reading of the relative humidity should be within the range of 5 % with respect to the real measurement, and the variation of the measured temperature should not exceed one degree Celsius with respect to the real temperature. The best sensor that meets these characteristics of all those seen in Figure 2.3 is the DHT22 [5].

The wiring of the board and the sensor is shown in the fig:cableado below.

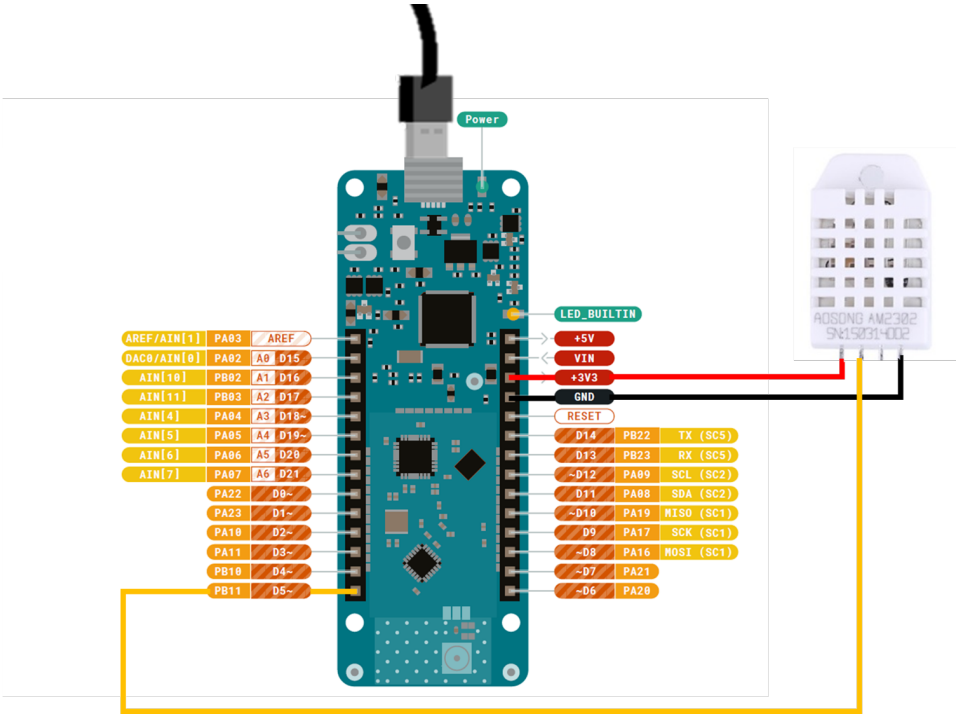


Figure 13. Wiring of the sensor and board.

The next block of the application is the technology that the board will use to communicate with the cloud, in this work: SigFox. To create an application that can be used in a large geographical area (for example, covering distances between cities, even countries) the most appropriate networks are WANs: Wide Area Network. Of special interest are LPWANs, Low Power Wide Area Network, which are networks that are characterized by long range, low power consumption and cost and low data transmission rate, making them very suitable for IoT systems. The requirements to use LPWANs can be summarized as: a multitude of connected devices, battery operated and spread over large surface areas, requirements that are perfectly aligned with the use case.

Within LPWANs, two clearly differentiated groups can be distinguished. Networks in the licensed bands operated by MNOs (Mobile Network Operator) and networks that are not operated by MNOs. Among the MNO networks, LTE-M and NBIoT stand out, and among the non-MNO ones, LoRaWAN and SigFox. The main characteristics of each of these networks are set out below:

- LTE-M: Defined in 3GPP LTE releases 13 and 14. Networks using this standard have a very efficient connection with great coverage in indoor and underground areas. They provide low latency allowing the development of applications that receive data in real time, and support mobility. Some examples of applications that use this type of communications are vehicular communications, emergency systems, patient monitors, logistics tags, etc.

- NB-IoT: This type of network is similar to LTE-M. A single cell of an NB-IoT network can connect up to 50,000 IoT devices. In these networks the data are grouped before being sent by the network (text messaging), increasing the latency of the network and the devices connected to them must be static, not admitting movement. Therefore this type of network is not the most suitable for the refrigerated transport project of this work.

- LoRaWan comes from Long Range Wide Area Network (>20km). LoRaWan is responsible for unifying different LoRa devices to manage their channels and connection parameters: channel, bandwidth, data encryption, etc. in order to send and receive information. To integrate LoRa, a SIM card is not necessary but a specific LoRa access point is required. Platforms such as Arduino or Raspberri Pi have shields or modules that can be easily attached to a board.

- SigFox is a leading private company in IoT solutions. The technology is characterized by a very low consumption, very low price and very low transmission rate. This company charges around one euro per device connected to its network per year. It offers a series of free control functionalities, such as service security, network maintenance, etc. In addition, the owner of the device has access to a platform: SigFox Backend, from which he can manage his devices, see the messages sent, configure responses and actions, locate the devices on the map, and more operations, which works 24 hours a day, 7 days a week. In addition, SigFox has a cloud in which all the data coming from the sensors is temporarily stored, but the idea is that the user configures a response to send that data to a place where he can take advantage of it.

	Redes LPWAN	
	LTE-M	SigFox
Propietario	Libre	Privado
Espectro	Requiere licencia	No requiere licencia
Alcance	2-5km urbanos	3-10km urbanos
Consumo	Medio	Bajo
Precio	Medio	Bajo

Figure 14. LTE-M and SigFox coomparison table.

Of the MNO networks it would only be feasible to carry out the project with LTE-M since NB-IoT requires the device to be static, but to work with a LTE-M you have to pay both the SIM card and the network subscription. To use LoRa, the Arduino MKR 1300 board could be used, but because it is a less mature technology for the development of industrial applications, it is decided not to use it. The most logical thing to do is to use SigFox networks since the chosen board is ready and the first year subscription is free. Before you can send data you have to register the device in the SigFox back-end. The text frontend allows you to manage the devices. Each device is unique and as soon as it is registered the initial free subscription is activated. The device registered for development is the 1CF5D8. Once registered you can proceed to send data.

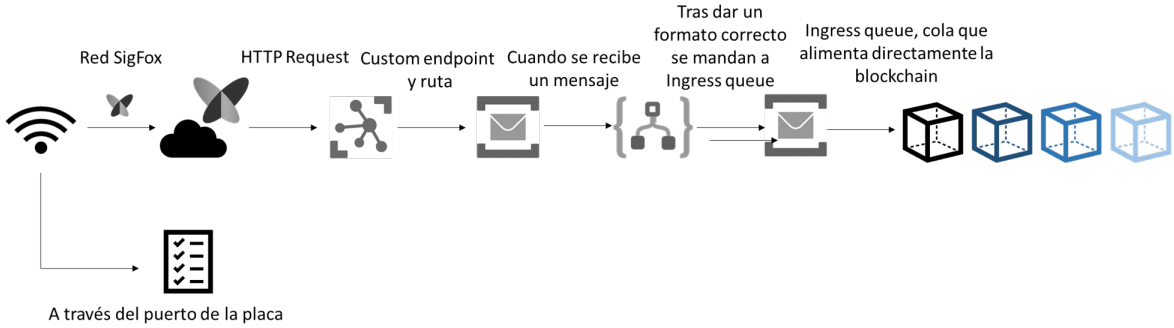


Figure 15. Data path through the application.

To have a more visual idea of the process that is being carried out, in the Figure 3.5, you can see the main connections that must be made to send the data from the physical sensor to the text blockchain. The first connection has already been made which is between the sensor and the SigFox cloud. The lower branch of the diagram refers to the data requested through the Serial Monitor to check that they are the same as those received in the backend text.

The third part of the analysis of the application corresponds to the 'text cloud' platform to which the data will be sent. It is interesting to use a cloud platform due to the accessibility, profitability and security it provides. There are several commercial solutions that could be used, some of the most frequent to work with IoT devices are the following: Google Cloud Platform, Particle, IBM Watson IoT. In addition to these, there are two platforms with which SigFox has prepared an easily configurable connection:

- AWS IoT : One advantage of AWS is the simplicity of its deployments within the wide range of solutions it offers. In addition, it allows you to carry out text software projects in multiple languages: Java, Python, NodeJS, etc. This AWS service allows "connecting, collecting, storing and analysing data from IoT devices"'. It offers two softwares for IoT devices: FreeRTOS and AWS IoT Greengrass, which allow to communicate, manage and administer devices in a safe way. It also offers control and connectivity services and analysis services.

- Azure Iot Hub: A key advantage of Azure is that it integrates with Microsoft's software [8]. Therefore many companies and institutions that use Microsoft to work (outlook, office 365, ...) decide to use Azure directly because of the capacity advantages, cost reduction and higher performance that it provides. This service is designed to be able to connect to almost any device. From this cloud platform, an infinite number of devices can be managed and configured [9]. It also allows easy integration with other Azure modules.

Both are very powerful solutions that allow the use case to be carried out, but it was decided to work with IoT Hub because it is a leading solution, with which we already had experience.

Before configuring the redirection of the messages received at SigFox, the platform that will receive them must be prepared. In this case Azure IoT Hub [9].

You must have a directory in Azure with an active subscription. Within that directory, a group of resources is created to specifically host the application's IoT Hub. The deployed Iot Hub module has the name of IotHub-TFM. Once deployed, the connection to SigFox can be made. This is done through the key: Connection string-primary key.

In SigFox, a callback is prepared which is an action taken when data is received. In the configuration of this text back, the IoT Hub connection key (text string-primary key) is introduced and the JSON code to be sent to Azure is described. With only these two configurations, the connection and the code are set to send data to IoT Hub.

Here is the JSON code to be sent out to IoT Hub. It requests the sending of the device identifier, the user data (in hexadecimal), the time stamp (in UNIX format), the message sequence number, the device type identifier and the variables of interest: temperature and humidity received from the sensor.

```
1    {
2    "DeviceID" : "{device}",
```

```
3     "data" : "{data}",
4     "time" : "{time}",
5     "seqNumber" : "{seqNumber}",
6     "deviceTypeId" : "{deviceTypeId}",
7     "humidity" : "{customData#hum}",
8     "temperature" : "{customData#temp}"
9     }
```

Finally, the fourth block of the analysis corresponds to the platform on which the cloud text blockchain will be developed, also known as text blockchain as a service (Baas). This type of platform provides the infrastructure and support to ensure the proper functioning of the text blockchain. It is an easy way to generate increasingly widespread text blockchain applications that will eventually allow DLT technologies to be integrated and standardised. Although there are many platforms where you can deploy text blockchains focused on IoT (such as SAP blockchain service platform, IMB text blockchain platform, etc), it is important to note that there are many different platforms that can be used to create a blockchain. AWS or Oracle Blockchain Cloud Service will work with Azure Blockchain Workbench due to the ease of integrating this service with the rest of the application. This Azure module is easily integrated with many other Azure resources (such as Azure IoT Hub), favoring the union of different technologies. In addition, Azure and therefore Azure Blockchain Workbench is designed so that you do not have to program, that is, with the basic notions of the different technologies you can build applications of considerable technical difficulty. The platform itself takes care of the technicalities below the user interface.

In summary, the Figure 3.1 shows the four blocks with the main technologies that are going to be used in the development of the application.

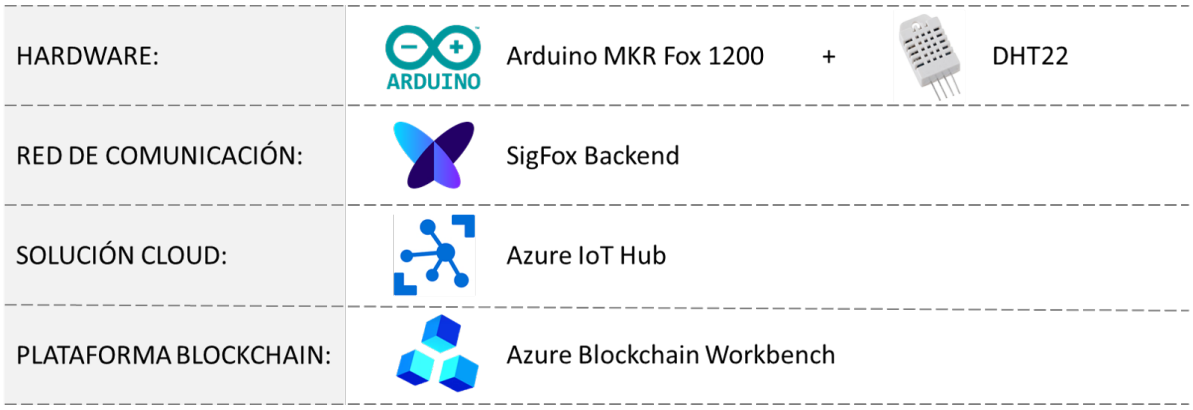


Figure 16. Chosen tecnology to develop the application.

Although the four main technologies are those shown in the **??** from the IoT Hub to Azure Blockchain Workbench it is necessary to go through different Azure modules. This process allows the transport of data and its modification to adapt it to the format that the text blockchain is capable of reading. This process is done taking as a reference the Github repository "Delivering Data from IoT Hub to Azure Blockchain Workbench" [11]. Returning to the autoref:schematics you can see the modules through which the data passes before arriving at the application blockchain.

A text bus is created called servicebusTFM. This module is a cloud-based messaging platform also known as messaging as a service (MaaS). It allows to send the messages received in IoT

Hub to a logical application that changes the format of the received message. To be able to send this data, a queue must be created inside servicebusTFM and a path must be generated on IoTHub-TFM connecting the received data with this queue. The queue in question has been called "myqueue".

A module called Logic App is displayed which will process the data and prepare it to adapt it to the blockchain format. This service allows you to capture data from a multitude of applications, which can be sources outside of Azure and in a dynamic way, without the need to write complex code, edit and create workflows [12]. For the application, the Logic App service will start with a trigger that once every minute checks if there are new messages in the queue. Before continuing, Azure Textchain Workbench must be deployed. This service actually consists of a set of Azure modules, already prepared to work. Among all the modules deployed, there is a SQL server: db-cjzdec-tfm. Following the steps of the GitHub repository mentioned above, you must launch a query or query to save a series of procedures that must remain stored in the SQL database. The procedure of interest is GetContractInfoForDeviceID, which for each device will get the information of the smart contract. But for the Logicapp service to be able to access this database, there must be a local connection to the SQL server. That is to say, a gateway to the SQL server must be established in the computer from which the service is going to work. To do this, there is an application called On-premises data gateway, which allows this link to be configured quickly following the instructions of Azure.

With the Azure Blockchain Workbench deployed and the text gateway ready, you can proceed to develop the flow that the data will follow in the logic app.

- Check every minute if there are any new messages in the queue, if there are, the message is taken out of the queue and enters the Logic App to be edited.
- Parse the data to be able to read the received message.
- Obtain, based on the device ID, the information of the active Smart Contract. This is done with a SQL module that searches within the procedures stored in the database. This is where the installed text gateway comes into play. data is parsed again and prepared for input into the text blockchain.
- A variable, called RequestID is initialized, with it an id is generated to be able to trace the process later.
- The time stamp is prepared in several steps to leave it in UNIX time.
- Now the message is ready, all that remains is to put it in the text blockchain. This sending is automatic, the way the Azure Text Blockchain Workbench is displayed makes every message that enters the text entry queue be swallowed by the text blockchain. This queue is inside the text titled service bus that is created by deploying the Azure Text Blockchain Workbench and therefore connections are made. Any message that arrives at this queue should be picked up by the relevant active contract. The final module is a module for sending to the texttit service bus.

The code of the message that is sent is filled in with the variables worked on, as shown in the fig:enviofinal autoref.

In Visual Studio Code the contract to be deployed in the blockchain application is prepared. Before displaying the contract, it will explain what you want to achieve and which participants

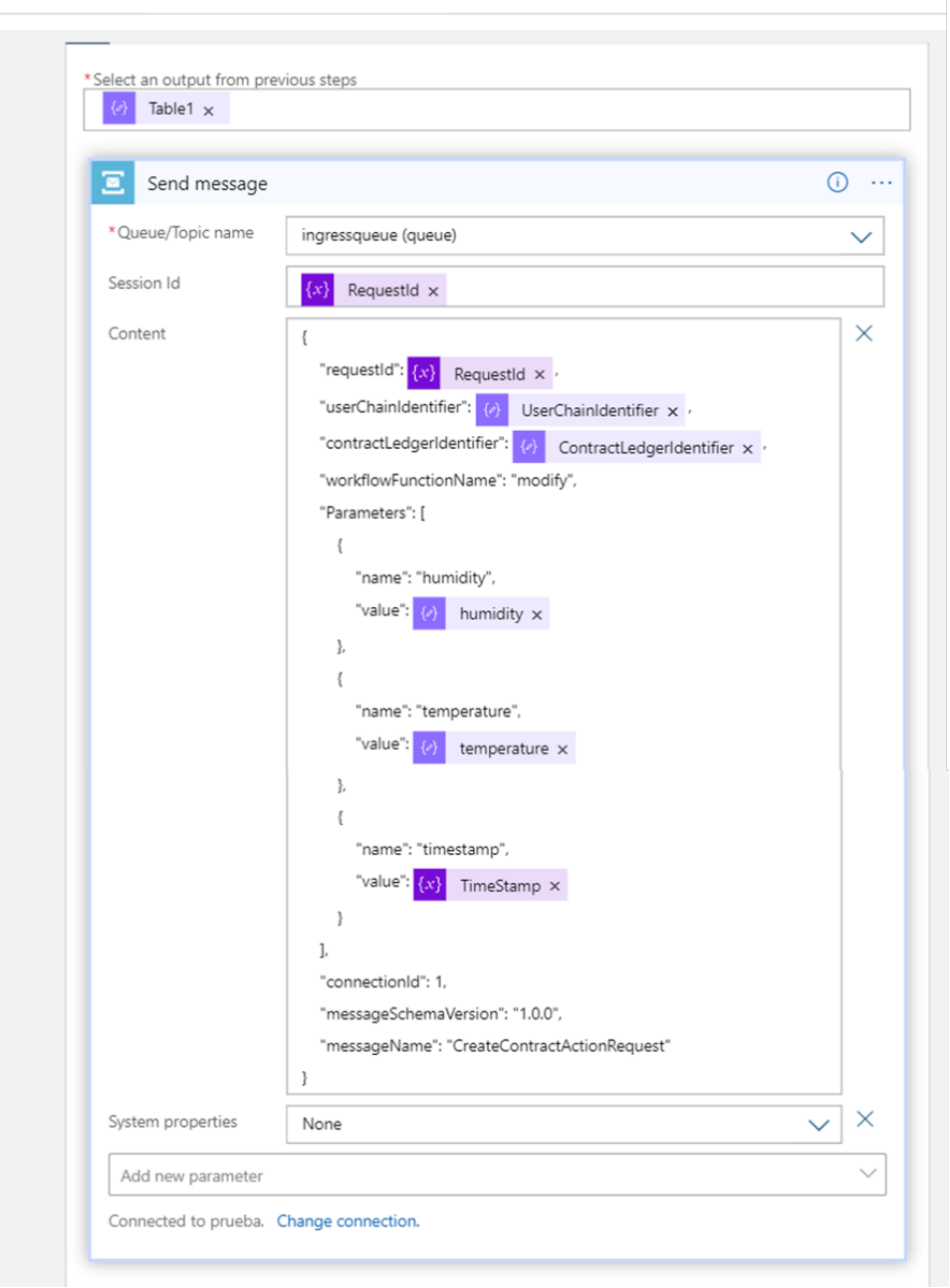


Figure 17. Blockchain ingest modules.

are involved. The contract status machine is shown in the Figure 3.17. This shows the possible changes in the status of the contract.

The actors involved in the process are:

- The main user, already seen above EmpresaTFM@carmenolleroshotmail.onmicrosoft.com.
- The user who corresponds to the device device@carmenolleroshotmail.onmicrosoft.com.
- A user acting as if he were the carrier transport@carmenolleroshotmail.onmicrosoft.com
- The personal email carmen.olleros@hotmail.com which is used as an observer and auditor of the process.

For the case of use of this work it is interesting to save the telemetry data. The aim is for these data to be saved in such a way that they are associated with each container, i.e. in

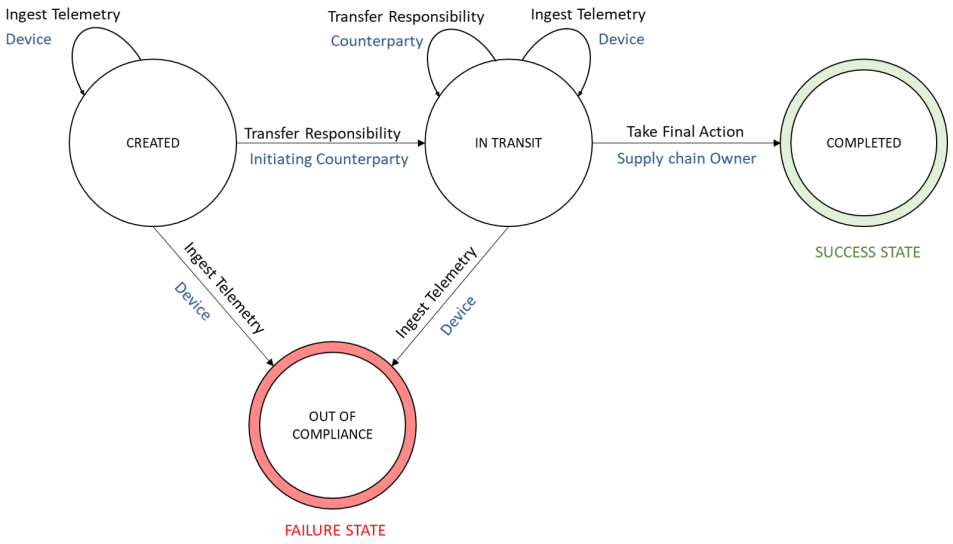


Figure 18. State maquine of the smart contract.

accordance with the sensor that sends them in a history that can later be seen stored in the database associated with the blockchain. This data history is the one that will allow to have a traceability and a greater transparency of what happens along the process. It is also with this data that it is possible to demonstrate that the cold chain has been fulfilled. This is of interest to the company in charge of transport, the supplier of the refrigerated product, the end customer and any company that audits, insures or regulates the transported product.

0.10. Results

After configuring the hardware, the messages are received correctly in SigFox. This verifies that the board works properly, is well powered, and that the sensor wiring has been done correctly. The messages that arrive at SigFox contain the following information: the time stamp (when the message is received), the temperature and humidity data sent, the quality of the signal, the status of the callback and the location from which the message has been sent.

In SigFox the next step (the connection with IoT Hub) is also configured, and data is sent again. Such data sent from the board, now arrives to SigFox, but it is also redirected to Azure IoT Hub. This connection is also successful because on IoT Hub, the sensor is listed. From IoT Hub, the messages go to a queue from which the data is taken by the Logic App, an application that modifies the data to give it the appropriate format for text blockchain.

Next, the figure Figure 4.5 shows how the flow created in the Logic App service works correctly. It is displayed as each block of the flow has a success signal in the upper right corner. In each one of them you can click and see the details of the process. Of special interest are the first module of the flow, in which the messages are taken from the queue, the module that obtains the contract information according to the device identifier and the last module that enters the data in the queue of the blockchain application. In the detail of the flow of the Logic App of the fig:modulos you can see the three modules indicated and how all of them have been carried out successfully.

In the first module of the Logic App, the message is correctly taken from the queue myqueue. This step confirms that IoT Hub sends correctly the data to the queue and that the link between the queue and this workflow service works correctly too.

Figure 19. Work flow detail.

The next module that is interesting to understand corresponds to the module that obtains the information of the active contract depending on the device ID. That is to say, it is able to get out satisfactorily from blockchain which contract is active and all its characteristics.

The last module that prepares the message and sends it to the text blockchain input queue is also very important, since it makes the link between Azure's Logic App and the text blockchain. This module, like the rest, works well. Sending seems to be done well and if you look at the outbound queue metrics, you can see how messages are received.

Actually, in the out queue, you can see how messages come in but they don't go out. The moment that worked better was at 3:41am of July 1st, when after sending data to the queue you can see how there is a sending threat, but usually the sent messages don't seem to go out of this queue. In the **??** you can see on the right the correct sending of messages at 3:41am and on the right a graph showing the Service Bus of blockchain with the incoming messages in blue and the outgoing messages in orange. At 3:41am you can see that the outgoing queue emits 7 messages.

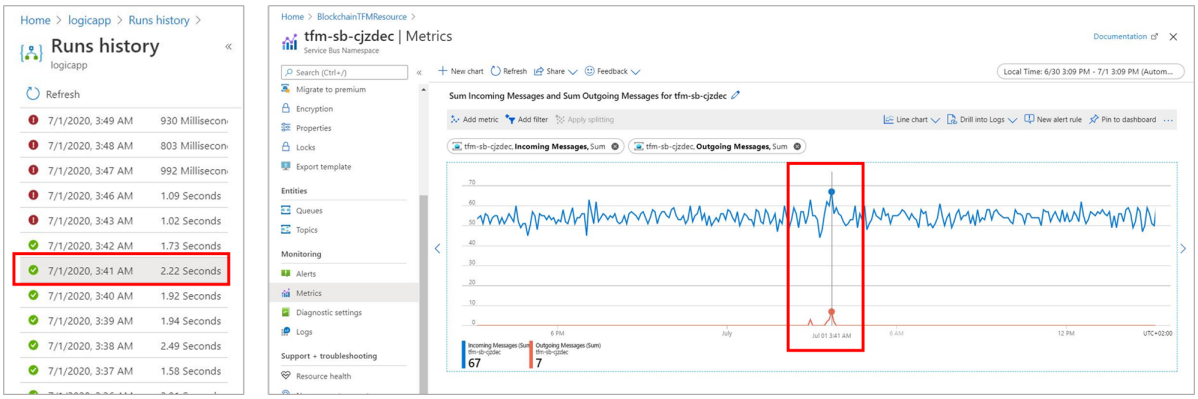


Figure 20. Active smart contract information.

Finally, to be able to work on the smart contract, an alternative platform to the Azure Blockchian Workbench web application is sought (because the subscription with which it develops the work ends and it can no longer be tested). On the recommendation of the directors, Remix is used as a platform where contracts can be tested quickly. Thanks to this platform you can continue to test the contract until you are sure that it works as you want it to.

0.11. Conclusions

The aim of this work was to carry out a proof of concept for an application that monitors the refrigerated transport cold chain for quality assurance using IoT, cloud and DLT. This objective has been achieved.

Checking the different goals to reach the main objective, we have the study of IoT and blockchain technologies. The technologies and protocols chosen in the development of the application are studied in depth, especially focused on their application in the logistics part of the supply chain of products that require a cold chain. The blocks analysed are: the hardware platforms, the communication technologies and protocols, the cloud platforms and the main blockchain platforms.

With a special focus on the logistics world, these technologies can bring about a significant change in the business model. On the one hand, blockchain allows intermediaries to be eliminated, as no third party is needed to verify the data. On the other hand, IoT provides visibility and traceability in the operations and movements, being automatically registered in the block chain. In addition, in the case developed in this work, the different parties involved in the supply chain can visualize what is happening in real time, eliminating complexity, promoting good practice with greater rigor in quality control.

Regarding the development of the application, it is demonstrated that it is possible to combine an Arduino board, SigFox connectivity, and different Azure modules until reaching the blockchain platform. Because Arduino is a platform that has been worked with more throughout the career and master, it was quick to set it up. The board must be powered correctly (in this case the computer is used as the source), the sensor must be connected with the correct wiring and the code must be prepared to send the data to SigFox.

SigFox and Azure have been the biggest challenge when developing this TFM but with the help of the directors and following the instructions from different repositories we have reached a lot. At the time of making the connection with SigFox, I was able to rely on a practice of the course IoT-Cloud Communications, where a similar practice was made, where data from Arduino to Azure was also sent through SigFox. The Arduino board is registered with the subscription and the text titled callback is configured to Azure IoT Hub.

In Azure, different modules are used than in the course because the data must be prepared so that it can be recognized by the text blockback application. I highlight what I have learned about parsing data. I find very interesting the ease with which Azure Logic App makes these changes.

All platforms and modules have been connected. It has not been possible to receive the sensor data in text blockchain, the last link between Azure Text Logic App, the text blockchain input queue and the publication of the data has given some problems when entering the text

blockchain. It has not been possible to debug and solve this problem, but the text smart contract has been validated using Remix.

Making a TFM of development has meant a personal challenge, since in several occasions I have found myself in front of a problem that I did not know how to solve but with dedication and effort, I found the problem, I managed to solve it and I advanced with the development.

Personally, having done this work has allowed me to have a deeper knowledge of the protocols and technologies that surround the Internet of things and blockchain. It has allowed me to improve my knowledge of Python and above all I have had the opportunity to learn Solidity, the language of intelligent contracts. I have learned not to give up on a problem I do not understand and I have improved my research skills.

0.12. Future work

It would be interesting to include the location of the device from which the data is sent. As explained above, it is not possible to do everything in the same SigFox text file but it could be done from a separate one, and in Azure to join the information obtained from both messages (the data and the location). To do this, it is important to find out what is preventing the application from working properly.

Another improvement from which the application would benefit greatly is an improvement in the smart contract. It should add to the data history not only what truck it comes from but who is driving this truck at the time, for example, so that it can directly warn that person if the temperature or humidity is out of range. It is also interesting to add the person driving the goods as a metric that can be studied by the company hiring the service.

Once the application works with these improvements, it could really be implemented in a truck container, since until now everything is done with a static sensor. It would be very interesting to check the operation of the application in movement. To get to the point of placing the sensor in a truck, it would be necessary to prepare a package for the sensor assembly plus a plate that would be easy to place inside the refrigerated container.

After testing with a single sensor, the next objective would be to organize an economic study of the solution to scale the project and deploy it in a fleet of trucks. With the study prepared, an agreement could be reached or the application could be sold to a company, and the solution developed on the job could be marketed.

1

Introducción

Combina y conquista
Accenture (2017)

Este capítulo introduce el tema del trabajo de fin de máster y los principales objetivos.

1.1. Motivación

Este trabajo de fin de máster trata sobre el desarrollo de una aplicación que surge de la unión de dos tecnologías diferentes: el Internet de las Cosas y la tecnología de registro distribuido. La aplicación consiste en garantizar la calidad a lo largo de la cadena de suministro en servicios de transporte refrigerado. La calidad se mide en cuanto a que no se rompa la cadena de frío, manteniendo el producto en unos rangos de frío y humedad determinados.

Para entender la motivación que lleva a realizar este trabajo se van a revisar los acontecimientos más relevantes en el sector industrial: la primera, segunda y tercera revolución industrial. Estas tres etapas de la historia han ido formando el contexto industrial que ha dado pie a la aparición de la cuarta revolución industrial, que es la etapa actual en la que han aparecido las tecnologías que se van a analizar en este trabajo. Además se exponen las principales razones por las cuales esta aplicación es relevante en la cadena de suministro.

La primera revolución industrial comenzó con la aparición de la máquina de vapor, inventada por James Wayy [13], que dio pie a grandes cambios económicos, tecnológicos y sociales. Se extendió el capitalismo, se incrementó la productividad y se redujeron los costes de producción y la población dejó de ser tan rural, pasando a vivir en las ciudades. Con respecto al medio ambiente esta revolución supuso un cambio muy fuerte, ya que las maquinas de vapor consumen combustibles fósiles, principalmente el carbón, aumentando la emisión de gases de efecto invernadero, tendencia que continuará en las siguientes revoluciones industriales. En ese momento las concentración de CO2 en la atmósfera era de aproximadamente 250 ppm [14].

Le siguió la segunda revolución industrial aproximadamente un siglo más tarde. Esta revolución introdujo la producción en masa y la aparición de nuevas fuentes de energía como la

electricidad y el petróleo. En esta etapa aparecieron nuevos modos de transporte como el avión, el automóvil o el ferrocarril; y nuevos métodos de comunicación como la radio y el teléfono [15]. Estas dos primeras revoluciones permitieron una modernización de la sociedad a través de mejoras en la eficiencia y la productividad de las actividades físicas de la tierra, la fuerza de trabajo y el capital. La aparición del petroleo provocó un aumento aún mayor de los gases de efecto invernadero. En 1927 las emisiones anuales de CO2 derivadas del uso de combustibles fósiles industriales ascienden a mil millones de toneladas [16] y en la década de los 60, las partes por millón de CO2 se acercaban a las 300.

La tercera revolución industrial aparece de la mano de nuevas tecnologías de la comunicación y la información. Los cambios de esta revolución permitieron grandes avances en el mundo digital. Se caracteriza principalmente por la conectividad entre equipos, la formación de redes, la aparición de internet y el comienzo de la digitalización de información.

Por último, en 2016 en el Foro Económico Mundial, se utilizó el termino cuarta revolución industrial [1]. Esta revolución, que se está viviendo en la actualidad, puede definirse como la revolución producida por la unión del mundo físico, con el digital y el biológico. Es decir, es la unión de los avances industriales obtenidos en la primera y segunda revolución industrial con las tecnologías desarrolladas en la tercera revolución industrial. Respecto a la huella de carbono actual, se están registrando máximos históricos, en Mauna Loa, Hawai se han registrado 415,26 ppm. A pesar de estar en máximos, la cuarta revoliución industrial es la primera que tiene en cuenta el medio ambiente y persigue un desarrollo sostenible.

Otra manera de ver esta revolución es como la transversalización de la revolución digital al expandirse horizontalmente a otros sectores [17]. La consultora tecnológica Accenture titula se refiere a esta revolución con el lema: combina y conquista [18]. Esta filosofía se resume en tomar las tecnologías existentes, principalmente derivadas de la tercera revolución industrial y combinarlas adecuadamente en diferentes sectores, surgidos de las dos primeras revoluciones, para crear nuevos y exitosos modelos de negocio.

Es especialmente significativo el impacto que tiene esta revolución en el mundo industrial. En esta cuarta revolución industrial aparece el concepto de Industria 4.0 también conocido como Smart Industry. Esto es la industria en la que personas, dispositivos, objetos y redes auto-organizadas de producción se combinan. Esta combinación resulta principalmente en: sistemas ciberfísicos, internet de las cosas, inteligencia artificial y *big data*. [19]. A continuación se desarrollan estos conceptos:

- Sistemas ciberfísicos: En ingles *Cyberphysical Systems* (CPS), son sistemas que integran capacidad computacional al mundo físico, con la finalidad de mejorar la seguridad o fiabilidad y el rendimiento.

- Internet de las cosas: Comúnmente conocido por las siglas en inglés IoT (*Internet of things*) hace referencia a la interconexión a través de internet de objetos cotidianos. Existe además una rama específica de esta tecnología dedicada a la conectividad de objetos industriales llamada *Industrial IoT* o *IIoT*. Algunas de las ventajas de IoT son la automatización, la obtención de datos y KPIs, el mantenimiento preventivo, el aumento de la productividad.

- Inteligencia artificial: Entendido como la inteligencia de las máquinas. Una máquina es inteligente si a parte de llevar a cabo instrucciones es capaz de aprender, adaptarse y resolver problemas. Un ejemplo de inteligencia artificial son los coches autónomos.

- *Big data*: En español datos masivos, hace referencia a la monumental cantidad de datos que se producen de diversas fuentes y para cuyo análisis ya no valen las herramientas de procesamiento habituales. Se ha creado una nueva ciencia alrededor del estudio de datos de este tipo.

Este nuevo modelo de industria se caracteriza por tener un alto grado de flexibilidad en términos de producción, volumen y eficiencia. Producción tanto en especificaciones, como calidad o diseño. La variabilidad en el volumen a producir y en términos de eficiencia y coste de los recursos a usar.

Esto favorece que las industrias se adapten a las necesidades del cliente haciendo cambios en en la cadena de valor de manera eficiente. El valor de este nuevo modelo reside en las comunicaciones centradas en las redes entre productores, clientes y sistemas; en la digitalización de información de distintos puntos de la cadena de valor así como la comunicación entre diferentes eslabones y en la granularidad, flexibilidad e inteligencia en los procesos productivos, ajustables en tiempo real para adecuarse a las especificaciones del cliente. La personalización de la producción ya es una realidad. La conclusión a extraer es que esta cuarta revolución industrial está generando un impacto socio económico dramático, al que empresas de diferentes sectores deben adecuarse si no quieren desaparecer [2]. Aspectos que son de vital importancia respecto a la cuarta revolución industrial y en específico en la Industria 4.0 son el aspecto social y la seguridad que rodea a los sistemas. El social ya que los perfiles demandados han cambiado y con respecto a los cambios en seguridad se deben adecuar a los nuevos riesgos industriales, se cubrirá más el tema de la seguridad según se vayan explicando las tecnologías más adelante.

Este contexto industrial muestra una evolución y modernización de las industrias de manera rápida e imprevisible, gracias a nuevas combinaciones de las numerosas tecnologías disponibles. Este trabajo se centra en el sector industrial, especialmente en la cadena de suministro: la aplicación trata la garantía de la cadena de frío en transporte refrigerado, analizando los beneficios de una combinación concreta de tecnologías: IoT y textitblockchain. Ya queda demostrada la relevancia del internet de las cosas, ahora se va a exponer qué es la tecnología de registro distribuido y *blockchain*, su comienzo y sus usos. A continuación se presentan las definiciones de estos términos, que se explicarán con mayor detalle.

Las tecnologías de registro distribuido son una clase de base de datos descentralizadas accesible desde diferentes nodos, es decir, hay varias copias de la información distribuidas entre los diferentes participantes. Cuando se realiza una operación la información se actualiza de manera sincronizada en todos los nodos. En ingles esta tecnología se llama *Distributed Ledger Technolgy* que puede traducirse como tecnología de libro de cuentas distribuido, ya que verdaderamente es una base de datos que no puede modificarse y por tanto se asemeja a un libro de cuentas. Es frecuente encontrarlo también por las siglas DLT (*Distributed Ledger Technology*). Lo que diferencia a las tecnologías de registro distribuido de las bases de datos generales es que cuando se introduce información nueva, en una DLT esta debe ser validada antes de ser escrita en la base de datos y para ello existen diferentes métodos de validación. En la Figure 1.1 se muestra un esquema del funcionamiento de una DLT.

Por otra parte, una *blockchain* es un tipo de tecnología de registro distribuido, con la peculiaridad de que las operaciones se registran en bloques que una vez completos se unen conformando cadenas. Los bloques de una *blockchain* tienen un encabezado y un final de bloque que lo ata con el bloque siguiente y con el anterior, esta característica hace prácticamente

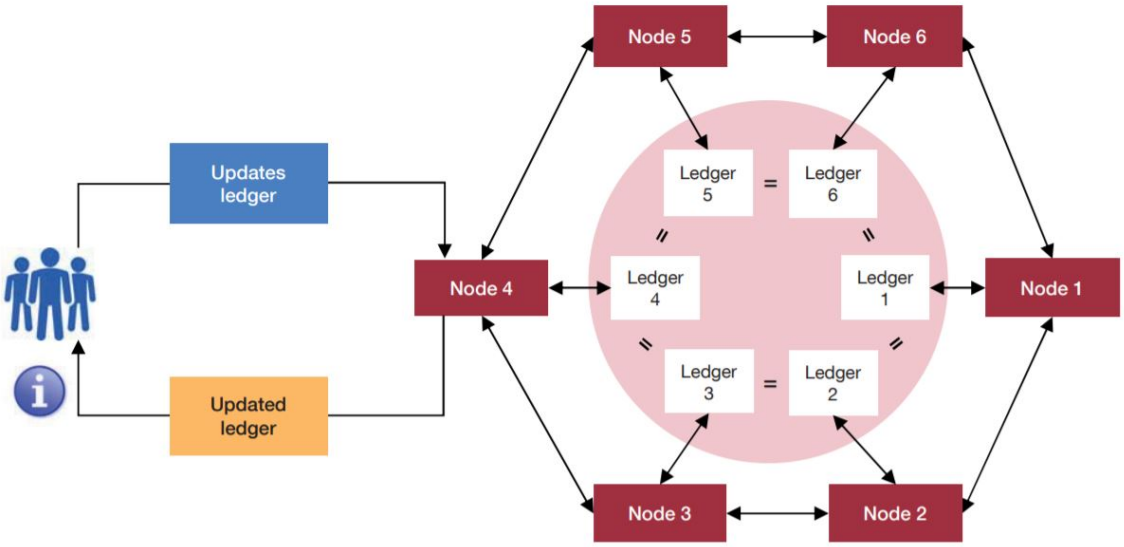


Figure 1.1. Esquema del funcionamiento de una DLT. Fuente: Banco de España [20].

imposible alterar la información guardada a través de esta tecnología. El término *blockchain* se ha hecho muy popular a raíz de Bitcoin, la primera moneda digital descentralizada que apareció en 2006 [21]. A pesar de que los primeros usos de *blockchain* se centraban como Bitcoin en criptomonedas, se ha ido desarrollando su uso en diferentes sectores.Esta tecnología que ha aparecido fuera del entorno industrial esta cada vez más presente en dicho entorno, en este trabajo se va a ver su uso específicamente para la cadena de suministro.

Una característica particular de esta tecnología es la posibilidad de implementación de *smart contracts*, que son códigos o protocolos informáticos que ejecutan los términos de un contrato [22]. Las clausulas contractuales se traducen en código que integra en él alguna propiedad ya sea de *software* o *hardware*, eliminando intermediarios. El uso de *smart contracts* está ayudando en la expansión de *blockchain* y las DLTs a otros sectores [23].

Entre las ventajas de este tipo de registros destaca la inmutabilidad, la transparencia, trazabilidad y seguridad. Inmutabilidad debido a que los registros una vez aceptados no pueden modificarse, transparencia y trazabilidad por que todo cambio de un registro queda plasmado, y dependiendo del tipo de red DLT o *blockchain* visible para los participantes de la misma. Por último seguridad por la dificultad en alterar las cadenas de bloques sin modificar todos los eslabones posteriores al modificado.

Finalmente se puede establecer que la combinación de IoT y de DLTs puede ser muy beneficiosa. Generando una combinación de las ventajas industriales de IoT (la automatización, la obtención de datos y KPIs, el mantenimiento preventivo, el aumento de la productividad) con las ventajas de DLTs.

Con esta base sobre las tecnologías a utilizar en el desarrollo de el trabajo, se va a exponer a continuación el ámbito de la aplicación. Se ha decido desarrollar una aplicación enfocada a cadenas de suministro. Por cadena de suministro se entiende el proceso que involucra desde el actor que explota la materia prima hasta el cliente final. Este sector ha cambiado mucho en los últimos años y continúa evolucionando, de manera que aplicando la pareja de tecnologías establecidas a este mercado se espera dar ejemplo de cómo ayudar en la mejora de un proceso. Profundizando un poco más en la cadena de suministro se pueden encontrar diferentes eslabones

y participantes. En el proceso se puede diferenciar entre la materia prima, los proveedores de materia prima, los productores, los distribuidores, los comerciantes (si aplica) y el cliente final. Este trabajo se centra en la parte logística que envuelve a la cadena de suministro, es decir, en el movimiento del producto. Se desarrolla una aplicación enfocada al transporte de mercancías refrigeradas, imponiendo unas limitaciones de temperatura y humedad en el proceso.

Derivado de todo lo anterior, la razón por la que se decide hacer el trabajo de fin de máster sobre el tema de Internet de las cosas y tecnologías de registro distribuido se debe a la creciente inclinación de diversos sectores por buscar soluciones o mejoras innovadoras a través de la implementación de estas tecnologías en sus procesos. En la Figure 1.2 se puede apreciar el incremento constante de la búsqueda de el término industria 4.0 en todo el mundo durante los últimos 5 años.

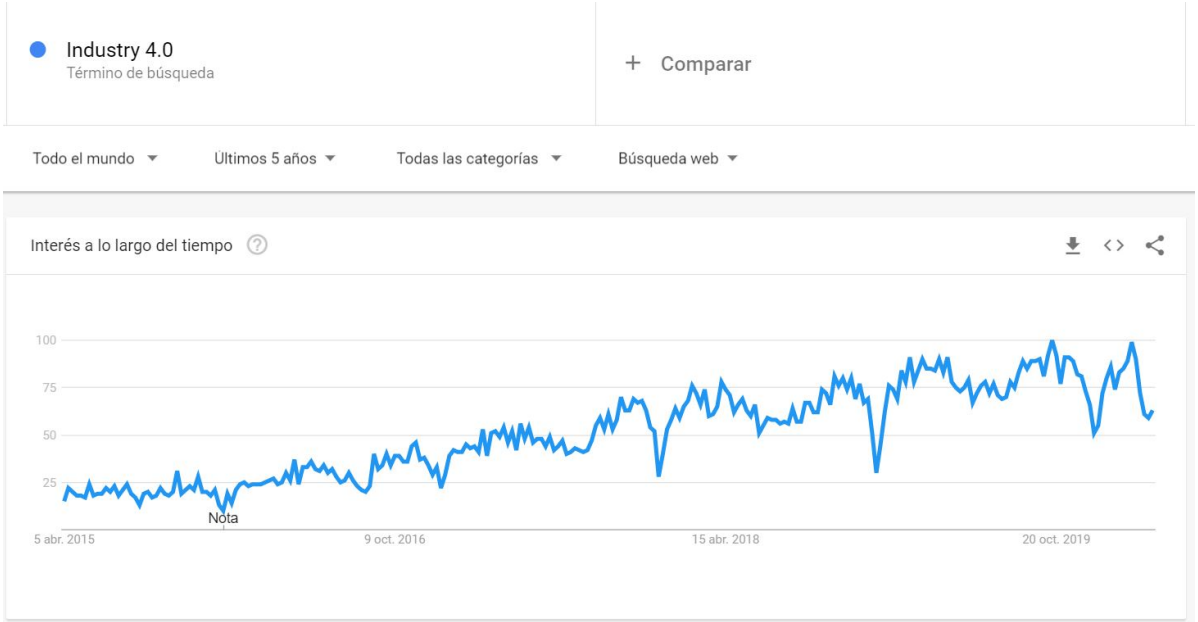


Figure 1.2. Busquedas mundiales en Google del término Industry 4.0 durante los últimos 5 años.

A pesar de que ya existen diversas aplicaciones de *IoT* y DLTs, la tendencia es que continúen apareciendo más aplicaciones a cada cual más innovadora, por eso este trabajo pretende ser una guía para entender las bases de estas tecnologías a través de el desarrollo de un caso de uso y poder así tener conocimiento suficiente como para decidir en qué y cómo se podrían usar estas soluciones.

Se decide hacer el caso de uso de estas tecnologías en el mundo de la logística y cadena de suministro, más específicamente de transporte refrigerado, debido a que es un servicio que está creciendo y adaptándose rápidamente a las necesidades de consumo actuales. Este sector llevaba años sin cambiar su modelo de negocio, pero desde hace unos años existe una mayor conectividad entre las propias empresas y entre empresas y consumidores. La logística ahora juega un papel fundamental. Además este sector se ha ido modernizando permitiendo una mayor trazabilidad de la mercancía y visibilidad para el cliente a lo largo de los envíos. Pero en el transporte de productos en cámara frigorífica debe asegurarse no sólo su entrega si no que ésta se haga en buen estado. Y por tanto las soluciones que aportan visibilidad y trazabilidad en los envíos son buenas pero se quedan un poco limitadas. En este trabajo se va a desarrollar una aplicación que permita trazar y ver el estatus del pedido así como la temperatura y humedad a la que se ha realizado el mismo. De esta manera se puede demostrar que el transporte se ha realizado sin poner en riesgo la cadena de frío de la mercancía.

Aunque este trabajo pretende ser genérico, para que el lector pueda aplicarlo a cualquier cadena de frío, se expone a continuación un ejemplo de aplicación para que sea más fácil comprender la taplicación. Una empresa familiar dedicada a la fabricación de nata montada de gran calidad, tiene que hacer frente al reto de mantener estables las condiciones del producto especialmente en el transporte. La nata montada es un producto que debe mantenerse estrictamente en unas condiciones de frío a lo largo de la cadena de producción y distribución ya que si no se cumplen el producto perece rápidamente. Además buscan dar mayor transparencia y trazabilidad al origen de su materia prima. Por tanto para mejorar su servicio, esta empresa ha decidido apostar por la aplicación de este trabajo.

Para la aplicación se va a usar un sensor *IoT* que envíe durante los trayectos la fecha y hora, la localización, la temperatura y la humedad. A continuación estos parámetros se enviarán a una red de registro distribuido, específicamente una red *blockchain* donde quedarán registrados. Con esta información guardada de manera inmutable y la implementación de Smart Contracts la empresa logística puede garantizar que cumple con los requisitos adecuados en cada uno de los eslabones de la cadena de suministro, dando un servicio de calidad fácilmente demostrable.

A continuación, en la Figure 1.3 se puede ver una ilustración de cómo se monitoriza el proceso, en el caso de la empresa de nata montada. Por un lado, implementan sensores en los vehículos logísticos de leche y por otro en la flota que distribuye la nata montada a los restaurantes.

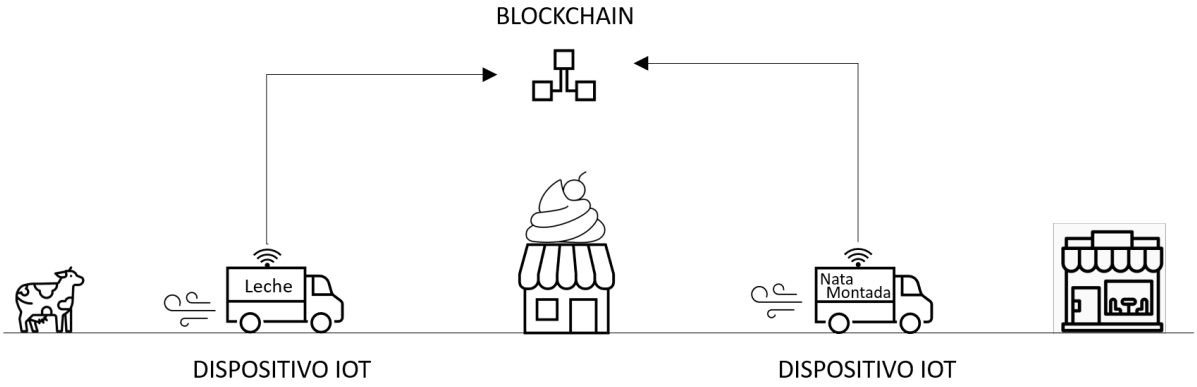


Figure 1.3. Ejemplo despliege de la aplicación para una empresa de nata montada.

1.2. Objetivos del proyecto

El objetivo de este trabajo es desarrollar un caso de uso de la combinación de dos tecnologías: Internet de las Cosas y *blockchain* a través de contratos inteligentes. Desarrollar esta aplicación de manera práctica es una manera de dejar plasmado que es posible conectar ambas tecnologías y que el resultado es provechoso. En este caso se va a aplicar a la parte logística de la cadena de suministro de productos que requieran cadena de frío. Esto permitirá añadir al servicio trazabilidad y visibilidad.

Para poder cumplir con este objetivo final se realiza un trabajo de investigación y documentación para facilitar la comprensión de las tecnologías a utilizar, entender sus fundamentos y los beneficios que pueden ofrecer.

Se realiza un estudio de las posibilidades IoT para desarrollar la aplicación analizando qué tecnología se va a usar para cada una de las capas del modelo de una red IoT, desde los sensores

de temperatura y humedad, pasando por las plataformas IoT disponibles a la nube que recibe los datos. Esto aporta una visión de las alternativas existentes actualmente en el mercado, se defiende y explica por qué se decide utilizar el sensor DHT22, Arduino y SigFox.

Para elegir qué plataforma DLT se utiliza, en concreto con qué plataforma *blockchain* se va a implementar la aplicación práctica también se hace un estudio de las alternativas existentes, mostrando sus ventajas y desventajas.

Con el desarrollo de esta aplicación no sólo se demuestra que es una combinación posible de ambas tecnologías sino que se busca demostrar que es relativamente fácil implementar una solución similar, y que este trabajo puede extrapolarse a muchos otros campos de aplicación. En el Chapter 2 se exponen algunos ejemplos de aplicación de estas tecnologías en otros sectores.

Se realiza un estudio económico del caso de uso, para aportar una visión financiera de la tecnología expuesta en el trabajo. Aquí se analizan los costes del proyecto y su efecto en el beneficio de la empresa. Por último, se añade la relación y cumplimiento del trabajo con los Objetivos de Desarrollo Sostenibles (ODS) de la ONU, se busca realizar un proyecto alineado con uno o varios de los 17 objetivos que se establecieron en 2015 con objetivo de que se cumplan en todo el mundo en un plazo de 15 años [24].

1.3. Metodología de trabajo y planificación

En este trabajo se va a hacer un estudio de qué posibilidades existen hoy en día en el mercado para desarrollar una aplicación de *IoT* junto con tecnologías de registro distribuido en una cadena de suministro, y para ello se analizarán todas las capas, desde la capa física hasta la capa de aplicación.

Para ilustrar mejor la metodología seguida en el trabajo para el desarrollo de la parte de IoT, en la Figure 1.4 se puede apreciar un gráfico en cuyo eje de ordenadas se pueden apreciar las diferentes capas del modelo de una red *IoT* y que contiene diferentes estándares y protocolos que se pueden utilizar para estas comunicaciones [3].

Tras este análisis se desarrolla el caso práctico donde se implantarán las soluciones escogidas. Se ilustrarán los beneficios de combinar estas tecnologías en el caso concreto de garantizar la cadena de suministro en transporte refrigerado. Con esto se pretende que empresas que mueven mercancía de frío en camiones con nevera frigorífica puedan demostrar tanto a la empresa a la que sirve el producto como a la receptora que su producto ha sido trasladado de manera que la cadena de frió no se ha roto en ningún momento. Además la propia empresa logística puede utilizar los datos obtenidos para optimizar su cadena de frío.

Este trabajo se va a realizar a través de un trabajo continuo con reuniones de seguimiento periódicas con los directores del trabajo. Se dedicará tiempo a la investigación y lectura sobre la tecnología a tratar a través de noticias, artículos científicos, informes, nuevas empresas, libros, etc. para documentar el trabajo y apoyar las decisiones que se tomen. Se dedicará tiempo a trabajar en el desarrollo de la aplicación, y a lo largo de todo el proceso se va documentando todo en la memoria.

Como preparación al caso práctico se lleva a cabo un curso de Python de Codeacademy que permite conocer mejor cómo funciona *blockchain*. También se completan los primeros capítulos del curso CryptoZombies que permite aprender a programar *blockchain DApps*, aplicaciones distribuidas que usan *blockchain*, así como a programar en *Solidity*.

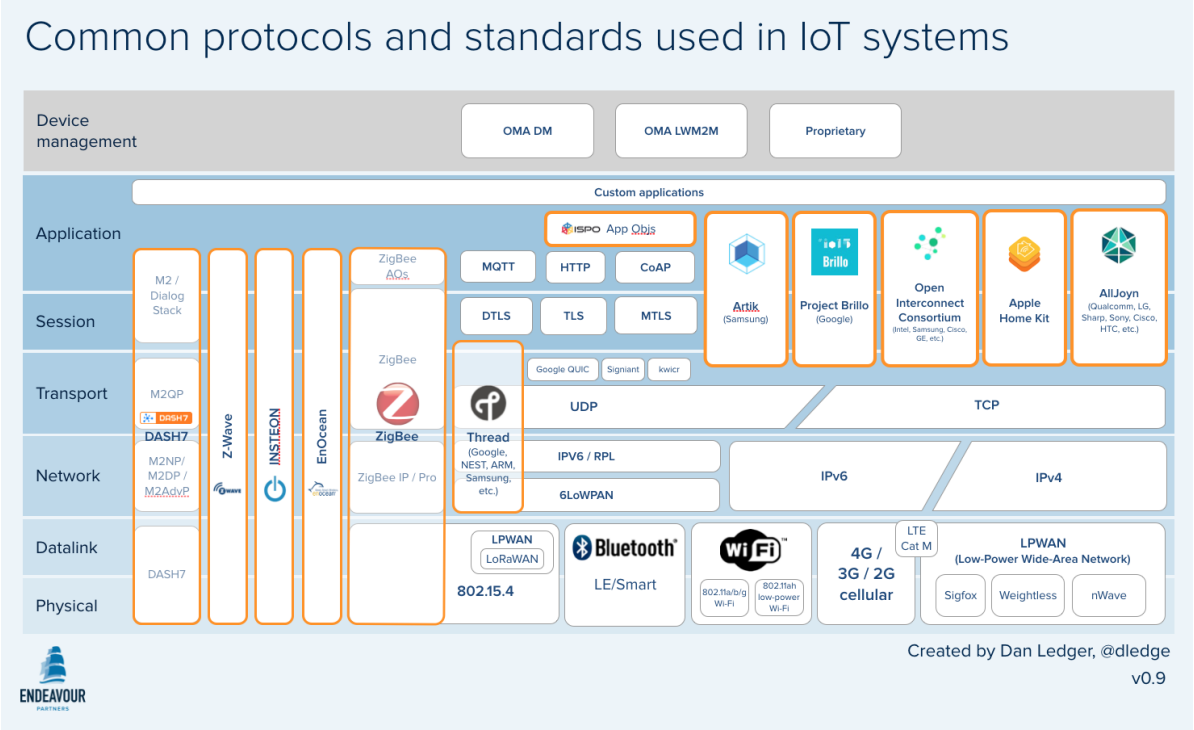


Figure 1.4. Estándares y protocolos ordenados según las diferentes capas de comunicación existentes para redes *IoT*.

La distribución de las horas de trabajo a lo largo del proyecto se encuentra en la siguiente lista. De manera visual en la Figure 1.5 se encuentra el cronograma del proyecto.

- Primer semestre: 4 horas durante las semanas de octubre y noviembre (aproximadamente se han dedicado 32 horas)
- Segundo semestre: 4 horas los sábados de enero y febrero (aproximadamente se han dedicado 32 horas)
- Marzo, Abril y Mayo: 4 horas los viernes y los sábados (aproximadamente se han dedicado 96 horas)
- Junio y Julio: Trabajo diario de 4 horas. (aproximadamente se han dedicado 160 horas)

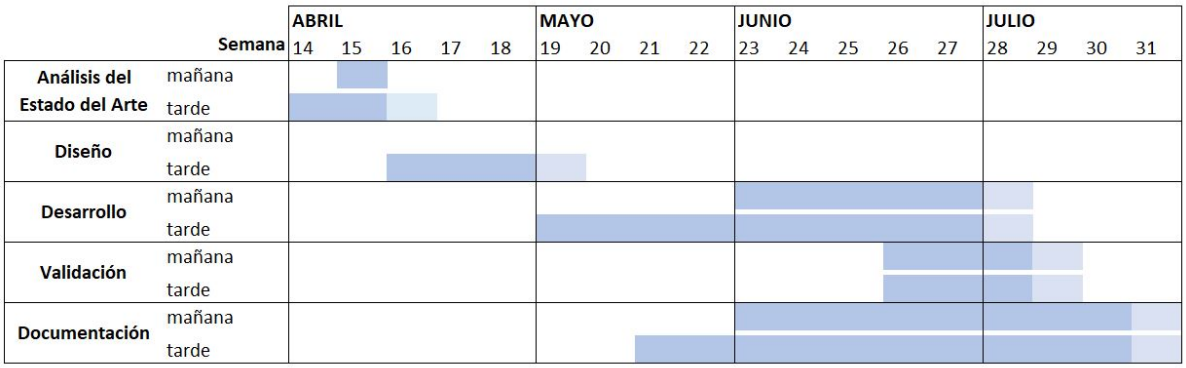


Figure 1.5. Diagrama de Gantt del proyecto.

A pesar de que se distribuye el trabajo a lo largo del año se puede ver cómo la carga de trabajo en los últimos meses es mucho mayor. Esto se debe a que, por la finalización de las clases se pueden dedicar muchas más horas al desarrollo del trabajo que durante el resto del

año. Como se puede ver en la Figure 1.5 en los últimos meses del desarrollo del trabajo se duplica la carga de trabajo a por las mañanas y por las tardes, y a trabajar más días de cada semana.

Se van a emplear diferentes recursos a lo largo del desarrollo del trabajo. Más abajo se encuentran listados en dos bloques: por un lado los recursos utilizados para desarrollar la documentación y redacción del trabajo y a continuación los recursos empleados para desarrollar la parte práctica.

Recursos y herramientas empleados para la documentación y escritura del trabajo:

- Paquete de Office365 de Comillas: OneNote, Correo, OneDrive, Teams, etc. Estas herramientas o plataformas, que permiten mantener reuniones, compartir notas y archivos han sido de gran ayuda debido al Corona Virus.
- Overleaf: editor de LaTex online en el que se desarrolla la memoria del proyecto. Esta plataforma al ser online permite que más de una persona pueda ver el archivo de esta manera los directories pueden monitorizar los avances en la memoria en cualquier momento.
- Herramientas de búsqueda: Explorador avanzado Google Scholar, Google Trends, IEEE, revista ECONOMIST, revista Medium, etc. Estas herramientas permiten realizar busquedas determinadas para encontrar articulos cientificos o tésis que puedan ser relevantes para este trabajo.
- Material de las clases del Máster de Industria Conectada: Tras haber cursado varias asignaturas relacionadas con el tema de estas tecnologías, en el material preparado para las clases se puede encontrar información relevante para el proyecto.

Recursos y herramientas para el desarrollo del trabajo:

- Laboratorios y recursos de la universidad: Debido al Covid 19 no se ha podido hacer tanto uso de los laboratorios de la universidad para trabajar como hubiera sido deseado pero la ventja es que este trabajo puede realizarse prácticamente en cualquier lugar, y por tanto ha sido posible llevarlo a cabo en estas circunstancias.
- Material: Además de las herramientas de software de Comillas, hay otros recursos como el Arduino o la placa base en la que se coloca el sensor que son recursos prestados por la universidad. Por otra parte el sensor fue comprado en una tienda de electrónica.
- Plataformas como SigFox y Microsoft Azure (dentro del cual destaca el uso de Azure IoT Hub y Azure Blockchain Workbench). Estas plataformas permiten el desarrollo del proyecto y se explican en mas detalle en el Chapter 2.

1.4. Organización de la memoria

Este primer capítulo introduce el trabajo, en él se expone la aplicación que se va a desarrollar, cuales son los objetivos principales de este trabajo, y la metodología y planificación que se lleva a cabo.

En el Estado del Arte se investigará sobre las diferentes soluciones *IoT* existentes en el mercado y se mostrarán las ventajas e inconvenientes de diferentes plataformas. Se decidirá también qué tecnología se va a usar para desarrollar el sensor *IoT* del caso práctico. Por

otra parte se verán también las plataformas *blockchain* que hay en el mercado, comparando soluciones tanto públicas como privadas. Una vez decidida la plataforma *blockchain* a usar en el caso práctico se hará una búsqueda de similitudes con otras redes, para cubrir el caso en el que el potencial cliente (o lector del trabajo) solicitase o quisiese almacenar sus datos en otra red.

A continuación se presentará el capítulo donde se desarrollará el caso de uso. El objetivo del caso práctico es garantizar la calidad a lo largo de la cadena de suministro en servicios de transporte refrigerado. Se va a utilizar un sensor IoT que, a través de una red LPWAN (*Low Power Wide Area Networks*), envíe datos para registrarlos mediante tecnologías de registro distribuido. El sensor permitirá transmitir información de temperatura y humedad de la cámara frigorífica de los camiones y esta información será registrada de manera inmutable utilizando tecnología DLT (p.ej., *blockchain*) para que clientes y proveedores puedan comprobar si la mercancía se transportó dentro de los rangos establecidos.

El último capítulo cubre los resultados y los pasos futuros del trabajo. También se incluyen en este trabajo varios anexos. El Anexo A que trata la relación del proyecto con los objetivos de desarrollo sostenible, en el anexo B se encuentra un estudio económico de lo que costaría implementar esta prueba de concepto, y el anexo C recoge los diferentes códigos del desarrollo. Tras estos anexos se pueden encontrar la hoja de características de sensor DHT22 y la hoja de características de la placa Arduino MKR Fox 1200.

2

Estado del Arte

"The IoT is going mainstream"
Microsoft survey, 30 julio 2019 [25]

En este capítulo se da una visión general de las diferentes opciones existentes para llevar a cabo una solución IoT junto con *blockchain*, con especial foco en aquellas que se van a utilizar en el caso de uso de este trabajo.

2.1. Plataformas *hardware*

Una solución IoT requiere una gran cantidad de dispositivos con capacidad computacional y con capacidad de comunicación. Para elegir la plataforma adecuada es importante analizar qué posibles *hardwares* existen. Es importante que la plataforma *hardware* que se elija sea *open source* o de código abierto, flexible, y de bajo coste debido al gran número de dispositivos que suelen requerir las aplicaciones de IoT. En este caso el volumen es un factor importante ya que si el caso de uso de este trabajo se llevase a cabo, se pondría un sensor en cada camión de una flota de camiones refrigerados por lo tanto se busca que no sea caro. Bajo estos criterios podemos encontrar soluciones que implementan microcontrolador y placa base, y soluciones que solo ofrecen el microcontrolador. El estudio se realiza de las opciones que ofrecen placa con el microcontrolador, ya que interesa tener una placa en la que colocar el sensor por la mayor comodidad a la hora de trabajar. Como por el momento no se va a implantar a gran escala prevalece la facilidad para desarrollar.

Dentro de las posibles empresas que ofrecen una plataforma *hardware* con microcontrolador y placa base destacan Arduino, Raspberry Pi, Sonoff, Particle, PIC y Espressif. Dentro de todas estas empresas de *hardware* Arduino y Raspberry son muy conocidas. Arduino es una empresa que ofrece placas muy completas aptas para principiantes además tiene multitud de soluciones diferentes en función de lo que se quiera desarrollar. La principal diferencia entre Arduino y Raspberry Pi es que mientras que las placas de Arduino llevan un microcontrolador, las de Raspberry son como un ordenador en miniatura, tienen sistema operativo y pueden ejecutar varios códigos a la vez. Se utilizan por tanto para proyectos diferentes ya que para un proyecto que requiera ejecutar un código sencillo, una Raspberry Pi tiene demasiada capacidad

y viceversa. Un proyecto muy complejo no puede realizarse con un Arduino. Por otra parte Sonoff ofrece una serie de soluciones para convertir una casa o una oficina en inteligente, a continuación se estudia si alguna de sus soluciones podría ser viable para este trabajo. Particle es una empresa que fabrica placas especificamente enfocadas para proyectos de IoT, permite desplegar una solucion IoT usando redes conectadas confiables, ya sean Wi-Fi o móvil. Pic es una familia de microcontroladores de la empresa Microchip Technologies y el nombre de este tipo de microcontroladores viene del acrónimo: *Programmable Integrated Circuited*, circuito integrado programable, y permiten realizar y controlar una tarea. Por último Estressif, esta empresa se dedica al desarrollo de soluciones IoT de baja potencia con capacidades Wi-Fi y Bluetooth, busca ofrecer soluciones seguras, confiables y que ahorren energía.

	MICROCONTROLADOR Y PLACA				SOLO MICROCONTROLADOR	
Plataformas *Hardware*	ARDUINO		SONOFF		PIC	ESPRESSIF
Modularidad	✓	✓	✓	✓	✗	✗
Módulo de conectividad *cloud*	✓	✓	✓	✓	✗	✗
Escalabilidad y versatilidad	Media alta	Alta	Baja	Alta	Media	Baja
Precio	€€	€€€	€€	€€	€	€

Figure 2.1. Tabla comparativa entre plataformas *hardware*.

A continuación en la Figure 2.1 se puede ver una primera comparación en función de la modularidad, la conectividad a la nube, la versatilidad/escalabilidad y el precio [26]. Estos parámetros dan una idea general de cúanto se aproximan a las necesidades de un proyecto de IoT. El resultado es que con este primer análisis se pueden descartar algunas de las plataformas para el desarrollo del trabajo ya que para monitorizar la cadena de suministro y conseguir mandar datos a la nube es preferible que la plataforma con la que se vaya a trabajar tenga un modulo de conectividad *cloud*, que además no sea excesivamente cara y sea fácilmente escalable. Por tanto quedan descartados PIC y Espressif.

	DESGLOSE CONECTIVIDAD			
Plataformas *Hardware*	ARDUINO		SONOFF	
Wi-Fi	✓	✓	✓	✓
Cellular	✓	✓	✗	✗
Bluetooth	✓	✓	✗	✓
NBIoT	✓	✗	✗	✗
Otros	SigFox	-	-	Mesh

Figure 2.2. Tabla comparativa entre las plataformas *hardware* seleccionadas.

Para hacer una comparación más en detalle de la conectividad entre las cuatro plataformas restantes se estudia la gama con chip ESP de todas las plataformas. Se compara entre: Arduino MKR series, RBPi 3B+, Sonoff Basic R2 y Particle Argon Iot *Starter Kit*. Los resultados se pueden ver en la Figure 2.2. De estas cuatro plataformas Sonoff queda descartado para el caso de uso de este proyecto debido a sus pocas opciones de conectividad. Se elimina a Raspberry Pi de la lista debido a que a pesar de que tiene buenas opciones de conectividad, su precio es elevado y eso no interesa. A continuación se van a analizar más en detalle Arduino y Particle.

2.1.1. Particle

Las plataformas de *hardware* de Particle están especializadas en soluciones IoT que integran desde la recepción física de los datos hasta la visualización de los mismos en la nube. Esta empresa ofrece soluciones de *hardware* potentes, tecnología de conexión confiable: utiliza una amplia gama de redes de operadores, proporcionando cobertura móvil LTE en más de 150 países. Otra característica importante es que también ofrece software para administrar la implementación de los proyectos IoT, Junto con su *hardware*, ofrece la posibilidad de trabajr con un *dashboard* donde se proporcionan funciones intuitivas y potentes, fáciles de usar y así monitorizar ya sea 1 o miles de dispositivos. Proporciona una interfaz en la nube o Rest API para administrar y monitorizar sus dispositivos.

De las diferentes placas que tiene disponibles Particles, interesan aquellas enfocadas y adaptables a proyectos IoT. Estas soluciones estarían dentro del grupo de Kits de Desarrollo que ofrecen, (aquí se encuentra entre otros la familia de placas Particle Argon vista en la comparación de la Figure 2.2). Dentro de la familia Argon de Particle exiten 3 placas: Argon IoT *starter kit*, Argon *starter kit* y Argon *Leak Detection Kit*. La que está más alineada con las necesidades del proyecto sería la placa Argon IoT *starter kit*, debido a que está enfocada a trabajos de Iot, descartando el uso de las otras dos.

2.1.2. Arduino

Arduino tiene una plataforma electrónica de código abierto basada en hardware y software hecho de manera que sea fácil de usar. Las placas de desarrollo de Arduino pueden leer diferentes entradas (la luz de un sensor, la presión de un pulsador o un nuevo mensaje de Twitter) y convertirlas en una salida: encender un motor, encender un LED y publicar contenido en línea. Para ello se envían un conjunto de instrucciones al microcontrolador en la placa, estas instrucciones se proporcionan a través de un lenguaje de programación propio de Arduino y el software Arduino IDE [27]. Estas características hacen que esta plataforma sea ideal para la aplicación que se quiere desarrollar en este trabajo.

Dentro de las diferentes soluciones que ofrece Arduino, se elige la placa que mejor se adapta a las necesidades de este proyecto. Se busca una placa de coste medio bajo, enfocada a proyectos IoT, que consuma poca energía (para que sea una aplicación duradera), y que ofrezca algún tipo de solución de conectividad que sea fácil de implementar. Tras investigas las gamas de placas, la familia de placas Arduino MKR permite a los desarrolladores cambiar fácilmente entre protocolos de comunicación inalámbrica de una manera sencilla con pequeñas modificaciones de software. Es una gama de Arduino especialmente enfocada para proyectos de IoT [4]. Esta gama de placas ofrece soluciones muy versátiles en cuanto a la conectividad: para Wi-Fi el Arduino MKR 1010, para conexión móvil (*cellular*) el Arduino MKR 1400, para *bluetooth* el Arduino MKR 1010, para NBIoT que hace referencia a *Narrow band* IoT (se explicará con más detalle en la Section 2.2: Arduino MKR 1500. El Arduino MKR 1300 tiene un módulo de conectividad Lora y por último para comunicaciones a través de las redes de SigFox: Arduino MKR FOX 1200.

2.1.3. Sensores y Actuadores

En las aplicaciones de IoT se parte casi siempre de leer alguna variable y para poder leer dichas variables se utilizan sensores. Hay muchas clases de sensores pero para hacer un análisis más focalizado a las necesidades de la aplicación a desarrollar: se va a centrar el análisis en

sensores que leen temperatura y humedad. Para establecer un criterio general se va a establecer una calidad de medida mínima. De manera general, se buscar que la variación de lectura de la humedad relativa esté dentro del rango del 5% respecto a la medida real, y que la variación de la medida de temperatura no supere el grado centígrado respecto a la temperatura real. Además se desea que se midan temperaturas negativas y positivas y un amplio rango de humedad relativa, preferiblemente el 100%. El precio también es un aspecto a valorar, se busca que no sea muy elevado debido a la cantidad de sensores que se utilizan en las soluciones de IoT.

Para obtener las dos variables de interés hay dos soluciones: buscar un sensor para la temperatura y otro para la humedad o buscar un sensor que integre la medición de temperatura y de humedad. Si se decide utilizar dos sensores, uno para la temperatura y otro para la humedad, hay que hacer un estudio de las opciones disponibles en el mercado.A continuación se encuentra el listado de los diferentes sensores estudiados:

Sensores de temperatura hay muchos, y su precio varía en función del rango de temperaturas que pueden medir y de la precisión con la que miden. algunos de los más utilizados son:

- Sensor de temperatura LM35: Visto en cursos anteriores de ICAI, es un sensor muy común de temperatura, barato (alrededor de 0,60€). Su salida es analógica y la calibración se hace directamente en grados Celsius. El rango de temperaturas que soporta es de -55 a 150ºC, y la precisión ronda el 1ºC.

- Sensor de temperatura TMP36: Sensor muy similar al LM35, también es analógico, la principal diferencia respecto al LM35 es que con el TMP36 se pueden medir temperaturas bajo cero sin proporcionar un voltaje negativo. Su precio es mayor, aproximadamente 1,5€. El rango de temperaturas que es capaz de medir se encuentra entre los -40ºC y los 150ºC aunque a partir de 125ºC se empieza a comportar de manera no lineal.

- Sensor de temperatura TC74: A diferencia de los dos anteriores este proporciona una salida digital. Es más caro, su precio ronda los 3€. Se recibe la medida en 8 bits, a través de una comunicación I2C. En relación al rango de temperaturas, puede medir de -40ºC hasta 125ºC, pero la precisión tampoco es muy buena oscila en entre los $\pm$ 2 y los $\pm$ 3ºC.

Respecto a los sensores de humedad, se buscan sensores de humedad relativa del aire. No hay sensores comerciales que midan únicamente humedad relativa, suelen ir de la mano de medidas de temperatura, pero siempre se podría usar un sensor que aunque midiese ambas variables, para el proyecto solo se utilizase la lectura de la humedad relativa. Destacan los sensores de la empresa Sensirion:

- Sensor de humedad relativa en el aire SHT7x: Es un sensor digital, menos intuitivo de usar que los descritos en el listado anterior. La humedad se mide mediante un sensor capacitivo y se da como un porcentaje entre 0 y 100%. Son sensores cuya medida es de alta calidad (desviación del 3% sobre la medida real), tiempo de respuesta rápido y que evitan interferencias externas. El precio de este sensor es bastante elevado, superando los 20€.

La otra opción es utilizar un único sensor para medir ambas variables. Para ello buscamos un sensor que mida en el rango adecuado y con la precisión adecuada ambas variables. Aquí encontramos la gama de sensores DHT, dentro de la cual encontramos el sensor DHT11 y el DHT22.

- Sensor de temperatura y humedad DHT11: Sensor digital, muy utilizado en proyectos de electrónica, ya que ofrece un rango de temperaturas y humedad lo suficientemente amplio para abarcar una gran cantidad de proyectos diferentes. Puede medir temperaturas entre 0 y 50ºC y humedades relativas en el rango 20 a 80%. Suele ser un sensor usado para proyectos que requieran poca precisión dado que en temperatura las medidas puede llegar a oscilar ±2ºC y un ±5% las medidas de humedad. Es un sensor asequible cuyo precio oscila entre los 5€ y los 10€.

- Sensor de temperatura y humedad DHT22: Este sensor es el hermano mayor del DHT11, ya que ofrece un rango de medición mayor, tanto en temperatura como en humedad además con mayor precisión. Puede medir entre -40 y 150ºC y todo el espectro de humedad relativa (de 0 a 100%). La precisión de las medidas de temperatura varía en ±0,5ºC sobre la medida real y las de humedad alrededor de ±2%. Este sensor es más caro que el DHT11, rondando los 15-20€.

	TEMPERATURA			HUMEDAD	TEMPERATURA Y HUMEDAD			
SENSORES	LM35	TMP36	TC74	SHT7x	DHT11		DHT22	
Rango medida	2 a 150°C	-40 a 150°C	-40 a 125°C	0 a 100%	0 a 50°C	20 a 80%	-40 a 150°C	0 a 100%
Precisión	± 0,5°C	± 2°C	± 2°C	± 3%	± 2°C	± 5%	± 0,5°C	± 2%
Precio	0,60 €	1,5 €	3 €	+20 €	5 a 10€		15 a 20€	

Figure 2.3. Tabla comparativa entre sensores.

En la Figure 2.3 se puede ver un resumen de toda la información anterior, quedan marcados en rojo los parámetros que no cumplen con las especificaciones deseadas.

2.2. Tecnologías y protocolos de comunicación

2.2.1. Tecnologías de comunicación

Para determinar el método de comunicación hay que tener en cuenta la arquitectura de la solución y la tecnología que se va a usar. Una manera de determinar qué tecnología va a ser necesaria puede ser el alcance de la misma. Por otra parte, hay redes cableadas y redes inalámbricas, para el caso de uso que se desarrolla en este trabajo, se necesita una red inalámbrica debido a la localización del sensor y a la independencia que necesita tener debido al movimiento de los camiones.

A continuación se describen los principales grupos de redes inalámbricas por su alcance geográfico:

- PAN: Las siglas son *Personal Area Network*, este tipo de redes denominadas de área personal recoge todas aquellas cuyo alcance es de unos metros alrededor de una persona o punto central. Un buen ejemplo de una tecnología con este alcance es Bluetooth.

- HAN: Significa *Home Area Network*, traducido es red de alcance doméstico, este tipo de red abarca decenas de metros. Aquí encontramos las tecnologías con un alcance domestico (también se puede usar en otros entornos como oficinas), en estos tipos de redes todos los ordenadores y equipos electrónicos de una casa están conectados una misma red. Un ejemplo de una tecnología que utiliza estas redes es Zigbee, protocolo definido por Zigbee Alliance que cuando es usado por varios dispositivos permite crear casas inteligentes,

monitorizar fábricas, etc. Zigbee pertenece a la capa de aplicación IEEE 802.14.5 que es LPPAN *Low Power Personal Area Network*.

- LAN y WLAN: Estos dos grupos hacen referencia a redes locales, *Local Area Networks*, con la diferencia de que las redes WLAN son inalámbricas, wireless Local Area Network. Aquí se encuentra Ethernet (no interesa por ser una tecnología cableada) y Wi-Fi, que sí que es de interés para el proyecto.

- MAN: *Metropolitan Area Network*, son redes que llegan a abarcar el área de una ciudad completa. Realmente suelen estar formadas por un conjunto de redes LAN (o WLAN). Un ejemplo sería la red Wi-Fi pública de una ciudad. Las redes celulares (2G, 3G y 4G) también se catalogan como MAN.

- WAN: Por último encontramos las redes WAN: *Wide Area Network*, las de mayor alcance, como pueden ser las redes de fibra óptica, WiMAX, 3G y 4G.

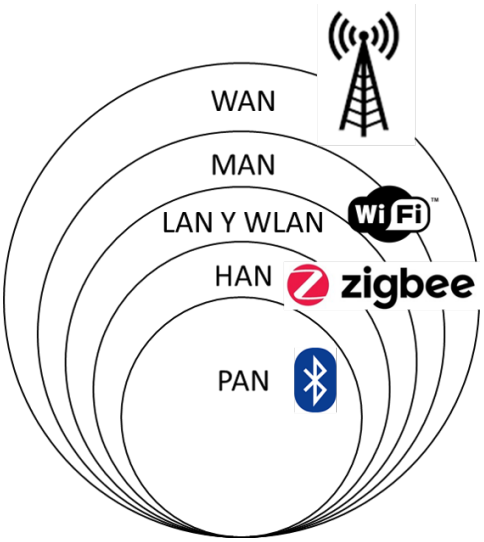


Figure 2.4. Organización de las redes según su alcance.

En la Figure 2.4 se pueden ver las redes anteriormente descritas ordenadas visualmente según su alcance.

Para crear una aplicación que pueda ser usada en un ámbito geográfico más grande que el que ocupa una ciudad (por ejemplo para que pueda ser usado para comprobar la cadena de frío de un producto transportado entre ciudades, o incluso regiones y países), se va a analizar más en profundidad qué opciones existen dentro de las redes WAN. Son de especial interés las redes LPWAN, *Low Power Wide Area Network*, que son redes que permiten la comunicación a largas distancias pero a un *bit-rate* muy bajo [28]. Es decir el consumo por comunicación es muy bajo, siendo entonces un tipo de WAN muy apropiado para sistemas IoT. Los requisitos para usar redes LPWAN se pueden resumir en: multitud de dispositivos conectados, operados por batería y repartidos por grandes áreas de superficie, requisitos que cumple perfectamente el caso de uso.

Dentro de las redes LPWAN se pueden distinguir dos grupos claramente diferenciados. Las redes que operan en las bandas de telefonía móvil y las que no. Los operadores de telefonía móvil (en inglés *mobiles network operators*: MNO) deben tener una licencia para poder funcionar, mientras que las redes que operan sin licencia comparten la banda ISM (banda que no requiere

licencia especifica para su uso industrial, científico y médico (ISM - Industrial, Scientifical and Medical).

A continuación se van a analizar. En el grupo de las redes pertenecientes a MNOs destacan las redes LTE-M y las redes NB-IoT, dos redes que fueron definidas en las Releases 13 y 14 de LTE del 3GPP. Estas dos tecnologías tienen características diferentes principalmente respecto a la latencia y la velocidad [6], aunque para utilizar cualquiera de las dos, es importante tener en cuenta que es necesaria una tarjeta SIM y una tarifa o subscripción a la MNO que se elija. A continuación se comparan ambas redes:

- LTE-M: Definido en los *releases* 13 y 14 de LTE del 3GPP. Las redes que usan este estándar tienen una conexión muy eficiente con gran cobertura en zonas interiores y subterráneas. Proporcionan baja latencia permitiendo desarrollar aplicaciones que reciban datos en tiempo real, y soporta mobilidad. Algunos ejemplos de aplicaciones que usan este tipo de comunicaciones son comunicaciones vehiculares, sistemas de emergencia, monitores de pacientes, etiquetas de logística, etc.

- NB-IoT: Este tipo de redes son similares a las LTE-M. Una sola célula de una red NB-IoT puede conectar hasta 50000 dispositivos IoT. En estas redes los datos se agrupan antes de ser enviados por la red (*batch messaging*), aumentando la latencia de la red y los dispositivos conectados a ellas deben de ser estáticos, no admiten movimiento. Por tanto este tipo de redes no es el más adecuado para el proyecto de transporte refrigerado de este trabajo.

Por otra parte dentro de las redes que operan en la banda ISM y no necesitan pagar licencia destacan las redes de SigFox y de LoRaWAN, en Europa ambas suelen trabajar en la banda 898MHz.

LoRaWan viene de *Longe Range Wide Area Network*, en español red WAN de largo alcance (>20km). LoRaWan es responsable de unificar diferentes dispositivos LoRa para administrar sus canales y parámetros de conexión: canal, ancho de banda, cifrado de datos, etc para así enviar y recibir información. Para integrar LoRa, no es necesaria una tarjeta SIM pero si es necesario tener un punto de acceso específico de LoRa. Plataformas como Arduino o Raspberri Pi cuentan con *shields* o módulos que se pueden acoplar fácilmente a una placa.

SigFox es una empresa privada líder en soluciones IoT. La tecnología se caracteriza por un muy bajo consumo, muy bajo precio y muy baja tasa de transmisión. Esta empresa cobra alrededor de un euro por dispositivo conectado a su red al año. Ofrece una serie de funcionalidades de control gratis, como son la seguridad del servicio, el mantenimiento de las redes, etc... Además el propietario del dispositivo tiene acceso a una plataforma: SigFox Backend, desde la cual puede manejar sus dispositivos, ver los mensajes enviados, configurar respuestas y acciones, localizar los dispositivos en el mapa, y más operaciones, que funciona las 24 horas del día, 7 días a la semana. Además, SigFox tiene una nube en la que se almacenan temporalmente todos los datos que llegan de los sensores, pero la idea es que el usuario configure una respuesta para enviar esos datos a un lugar en el que les saque provecho.

La infraestructura de Sigfox funciona las 24 horas del día, 7 días a la semana, y el usuario no necesita preocuparse de realizar ningún tipo de mantenimiento, además como se comentaba más arriba siempre se puede verificar el estado del dispositivo en SigFox Backend. Además, SigFox tiene una nube en la que se almacenan temporalmente todos los datos que llegan de los sensores, pero la idea es que el usuario configure una respuesta para enviar esos datos a

un lugar en el que les saque provecho. SigFox tiene preparado un sistema de respuesta con las siguientes 4 plataformas:

- AWS IoT
- AWS Kinesis
- Microsoft Azure Event Hub
- Microsoft Azure IoT Hub

	Redes LPWAN	
	LTE-M	SigFox
Red	Mobile Network Operator	No Mobile Network Operator
Especificación	Requiere licencia	No requiere licencia
Costes	Tarifa + tarjeta SIM	Tarifa

Figure 2.5. Comparación de redes LPWAN para proyectos de IoT.

Finalmente, en la Figure 2.5 se comparan las tecnologías LTE-M y SigFox, que son las dos que más se asemejan a las necesidades del proyecto, debido a que permiten movilidad del dispositivo y son de fácil implementación

2.2.2. Protocolos de Transporte

Hay dos protocolos básicos, muy usados en todo tipo de aplicaciones, hay que conocer sus características para saber cual de los dos es más apropiado para cada situación. Los dos protocolos son UDP y TCP:

Empezando por TCP, las siglas son *Transport Control Protocol*. TCP se usa para comunicar dos ordenadores o dispositivos que tengan una IP asociada. Se utiliza cuando la comunicación entre los dos puntos debe ser fiable, ya que este protocolo garantiza la recepción de los datos en orden en el punto de llegada. Ejemplos de uso de este protocolo pueden ser una página web, la descarga de un documento o la recepción de un email. Con este método se asegura que se reciben todos los paquetes de datos y que no se pierde información y por tanto la pagina web se verá correctamente, con los textos e imágenes ordenados, la descarga se completará de manera exitosa y el email se recibirá con todo su contenido. Este protocolo asegura no sólo la recepción de todos los paquetes de datos, sino que además se reciben en el orden correcto.

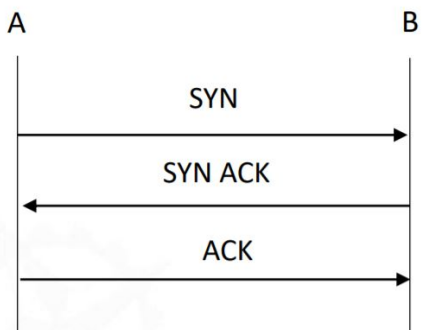


Figure 2.6. Conexión de una comunicación TCP.

Una característica importante de este protocolo es que esta orientado a conexión. Antes de mandar los datos, se establece una conexión entre el punto emisor y el receptor. Para ello se usa un reconocimiento con 3 partes. El punto emisor manda al receptor un primer saludo: SYN: el receptor si lo recibe asiente dicha recepción (segundo saludo): SYN ACK y por último el emisor manda al receptor un tercer saludo en el que avisa que ha recibido el SYN ACK: ACK RECEIVED. Una vez se ha establecido la sesión, se comienza a enviar información, de paquete en paquete, y si alguno no llega se vuelve a enviar. En la Figure 2.6 se puede ver de manera visual la manera en la que se establece la conexión.

El otro protocolo UDP, en inglés *User Datagram Protocol*, es similar a TCP en que también se utiliza para enviar y recibir datos, pero por el contrario carece de protocolo de conexión. No establece una sesión y no garantiza que los datos enviados lleguen al receptor. En otras palabras al nodo emisor no le importa si el nodo receptor recibe el paquete de datos o no, solo se preocupa de mandarlo. Este protocolo es conocido también como *fire and forget*, (dispara y olvida) debido a que no se preocupa de la recepción de los datos. Por otra parte este protocolo es mucho más eficiente que TCP debido a que los paquetes de datos tienen un encabezado mucho más pequeño, incluyendo menos información que deriva también en que no exista una garantía de que lleguen todos los paquetes. Un ejemplo de uso de este protocolo son las aplicaciones con registros en tiempo real como VoIP.

	TCP	UDP
Nombre	Transport Control Protocol	User Datagram Protocol
Protocolo	Orientado a la conexión	No establece conexión
Funcionamiento	Revisa si hay errores y corrige	Revisa si hay errores pero no corrige
Encabezado	Largo: 20 bytes	Corto: 8 bytes

Figure 2.7. Comparativa protocolos de transporte.

En la Figure 2.7 se puede ver de forma resumida, una comparación de ambos protocolos de transporte.

2.2.3. Protocolos de Aplicación IoT

Otro aspecto que es importante conocer son los principales protocolos IoT. Conocer estos protocolos permite poder entender cómo se esta produciendo la comunicación entre dos puntos. Existen muchos protocolos de comunicación, pero este apartado se centra en los más comunes en proyectos de Internet de las Cosas. se van a exponer a continuación las diferencias y similitudes entre MQTT y CoAp, que son los dos protocolos más usados en el entorno IoT.

MQTT o *Message Queueing Telemetry Transport* es un protocolo que utiliza el protocolo TCP de transporte, es decir se establece una comunicación entre el punto emisor y el receptor. La estructura de este protocolo es que existe un *broker* que actúa como punto central, y multitud de nodos asociados a dicho *broker*. No existe comunicación entre los nodos, todas las comunicaciones pasan por el *broker*.

Otra característica importante de este protocolo es que no modifica la información, tan solo envía y recibe. La manera en la que se envía y recibe información es a través de *topics*, que son como temas en los que los diferentes nodos pueden publicar o estar subscritos. Si publican la información entra en el tema y si están subscritos, les llega información cuando es publicada en el tema de interés. De esta manera no todos los nodos tienen que recibir toda la información,

tan solo los nodos subscritos a un tema. Además MQTT tiene los conceptos de *Quality of Service* (QoS) y Store and Forward incluidos dentro del protocolo. Gracias a la comunicación TCP, se conoce que se asegura la recepción de los datos aportando por tanto calidad al servicio.

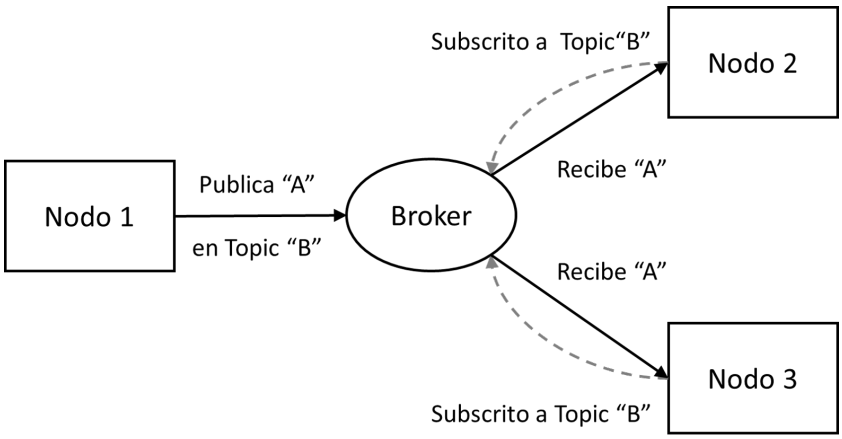


Figure 2.8. Ejemplo de arquitectura de protocolo MQTT.

Este protocolo también es capaz de trabajar con nodos que estén total o parcialmente caídos, esto es muy relevante con respecto a las aplicaciones IoT ya que interesa que los dispositivos consuman poca energía y puedan estar en modo de espera o fuera de línea mientras no tengan que transmitir datos. Con MQTT la posibilidad de conectar con el *broker* solo en momentos determinados es una opción posible. En definitiva, este protocolo es muy potente a la hora de transferir información desde y hacia los dispositivos a través de conexiones inestables. En la Figure 2.8 se puede ver un ejemplo de cómo es una configuración MQTT. Ejemplos de uso de este protocolo son las industrias que pertenecen al mundo 4.0, que automatizan los procesos: las diferentes máquitas están subscritas a ciertos topics y en función de la nformación que reciben realizan unas acciones u otras. Otro ejemplo son los vehículos inteligentes, los componentes del coche pueden comunicarse a través del broker. Aquí la función de nodos parcialmente fuera de uso o dormidos es importante ya que así el coche puede ahorrar energía evitando tener todos los componentes activos contínnuamente.

El otro protocolo de aplicación típico de soluciones IoT es CoAP, que significa *Constrained Aplication protocol*. Este protocolo comparte las bases del protocolo HTTP de las páginas web (*Hypertext Transfer Protocol*), pero está adaptado a las necesidades de los dispositivos IoT. Utiliza el concepto de RESTful API, que es un estilo de arquitectura de software que define un conjunto de restricciones utilizado principalmente para crear servicios web. La metodología utilizada se basa en los cuatro comandos:

- Get: para recuperar un recurso.
- Put: para cambiar el estado o actualizar un recurso, que puede ser un objeto, un archivo o un bloque.
- Post: para crear ese recurso.
- Delete: para eliminarlo.

Lo que aporta CoAP es que es un protocolo que puede hacer estas llamadas de manera *RESTfull* a través de redes específicas de IoT. Al contrario que MQTT utiliza el protocolo UDP de transporte, por lo que se puede utilizar para dispositivos con poca memoria y procesadores lentos. La arquitectura de los protocolos CoAP se basa en la filosofía *client-server*, completamente diferente

a la de MQTT. Aquí no es necesario un intermediario (en el caso de MQTT es intermediario es el *broker*) y un nodo puede comunicar a muchos servidores. Se puede entender como que el nodo pasa a ser el centro de la arquitectura. Esta manera de comunicar se llama *Request-Response*. En este caso las conexiones entre los diferentes puntos deben de ser conexiones estables, no funciona con nodos caídos o dormidos, pero en contraste con MQTT posee la cualidad de ejecutar comandos sobre la información que circula (gracias a los comando Get, Put, Post y Delete vistos más arriba). Por último, en relación a la seguridad, CoAP, al igual que HTTP tiene un protocolo hermano de seguridad: CoAPS (como HTTPS). Un ejemplo de aplicación de este protocolo son los edificios inteligentes, los diferentes nodos pueden pedir información de otros nodos, y actuar en función de dicha información.

	MQTT	CoAP
Comunicación	Publish & Subscribe	Request Response
RESTful	sí	No
Arquitectura	Broker es el centro	El nodo es el centro
Ejemplo	Industria 4.0	Edificio inteligente
Transporte	TCP	UDP

Figure 2.9. Comparación de protocolos de aplicación.

En resumen, ambos protocolos son apropiados para redes de bajo volumen y dispositivos de bajo consumo, y por tanto son muy usados para proyectos de IoT, la Figure 2.9 recoge un resumen de ambos protocolos.

2.3. Plataformas *Cloud* para IoT

Ahora que se conocen las especificaciones de los protocolos de red, de transporte y de aplicación hay que determinar a qué plataforma *cloud* se van a mandar los datos mandados por las redes de SigFox. Las características principales de las plataformas *cloud* son las siguientes:

- Accesibilidad: Se puede acceder a los datos desde cualquier punto y en cualquier momento, si se utilizase un programa *off-line* esto no sería posible
- Rentabilidad: Las plataformas *cloud* están preparadas para manejar grandes cantidades de datos de manera ágil y escalable. Además ofrecen multitud de servicios pero por lo general solo se debe pagar por los usados, esto se conoce como *Pay as a Service* o PaaS.
- Seguridad: Las plataformas se encargan de la seguridad de los datos y aplicaciones desarrolladas en ellas, evitando así una preocupación al usuario.

Como se ha visto en la Section 2.2, SigFox es capaz de conectarse con 4 plataformas, dos de ellas de Amazon *Web Services* y otras dos de Azure, un portal de Microsoft. Antes de indagar en estas 2 plataformas se van a exponer algunas de las plataformas *cloud* más comunes para conectar con dispositivos IoT [29].

1. Google Cloud Platform: Esta plataforma se utiliza para organizar, gestionar y compartir documentos. Particularmen para IoT ofrece soluciones de *smart cities and buildings* (ciudades y edificios inteligentes) y y seguimiento de activos en tiempo real. Es una plataforma preparada para la gestión de dispositivos IoT. La manera de cobrar por sus servicios es en función del volumen de datos usado.

2. Thinger.io: Es una plataforma para proyectos de IoT, que permite trabajar con todo tipo de *hardware*. Surge de una *start up* española. Ofrece un servicio freemium, es decir es gratis desarrollar el proyecto, se paga en el momento que sea necesario escalar el proyecto, también es de código abierto como las dos anteriores. Thinger.io además de SDK también ofrece un servicio *cloud* y una consola desde la cual monitorizar todo, razón por la que trabajan con una política freemium y se vuelve de pago cuando se escala.

3. IRI Voracity: Esta plataforma está más enfocada a *big data*, pero en función de l proyecto IoT puede ser una buena opción. Es una plataforma rápida y asequible para el descubrimiento, integración, migración y análisis de datos, que puede convertir, informar y anonimizar flujos de datos de dispositivos a través de Kafka o MQTT. Por ejemplo, en grandes archivos de registro o tablas de bases de datos. Básicamente agrega todos los pasos de la gestión de datos masivos. A diferencia de Google, su política de precios varía en función de la cantidad de dispositivos transformando o reportando datos.

4. Particle: Esta plataforma se ha estudiado en la Section 2.1.1, debido a las soluciones *hardware* que ofrece, pero Particle tiene también soluciones de conectividad y de *cloud*, es decir es una plataforma muy preparada para la gestión de dispositivos, especialmente los suyos propios. Los precios de uso de sus plataformas dependen de la conectividad de los dispositivos, ofreciendo por tanto diferentes tarifas en función de si es conectividad Wi-Fi, móvil o Mesh.

5. Salesforce IoT Cloud: Esta plataforma *cloud* está preparada para tratar con datos provenientes de sus clientes, asociados y de dispositivos y sensores de todo tipo. Está enfocada a conectar el mundo de IoT con los procesos corporativos, para mejorar las empresas.

6. ThingWorx Industrial IoT Platform: Esta plataforma esta preparada para aglomerar todos los pasos necesarios de una solución IoT: el control y monitorización de dispositivios, el análisis de los datos, y el desarrollo e implementación de una solucion web o móvil. Además ofrecen un servicio de atención al cliente para ayudar al cliente a lo largo de todo el proceso

7. IBM Watson IoT: Esta potente plataforma de IBM se centra dar servicio para todo tipo de soliciones IoT, desde la conectividad, al análisis e incluso servicio de plataforma *blockchain*. Gracias a todas las opciones que ofrece Watson el cliente puede encontrar la que mejor se adapta a sus necesidades. La plataforma permite optimizar operaciones y recursos. Está preparada para gestionar en la nube multitud de dispositivos. Los precios de IBM se basan en el volumen de datos tanto recibidos como analizados.

Todas las plataformas expuestas anteriormente, son buenas opciones para desarrollar una solución IoT, pero como se ha decidido que para el caso de uso de transporte refrigerado de este trabajo se va a utilizar la infraestructura de SigFox, a continuación se analizan las opciones *cloud* que enlazan directamente con dicha infraestructura: AWS IoT, AWS Kinesis, Azure *Event Hub* y Azure IoT Hub.

Entre las características que estas cuatro plataformas *cloud* comparten están el mantenimiento, del que se ocupan cada una internamente y el pago, que se realiza únicamente por los servicios utilizados [8].

- Una ventaja de AWS es la simplicidad de sus despliegues dentro del amplio abanico de soluciones que ofrece. Además permite realizar proyectos se *software* en múltiples lenguajes: Java, Python, NodeJS, etc.

    - AWS Iot: este servicio de AWS permite "conectar, recopilar, almacenar y analizar datos de dispositivos IoT". Ofrece dos *software* para los dispositivos IoT: FreeRTOS y AWS IoT Greengrass, ofrece también servicios de control y conectividad y servicios de análisis [7].

    - AWS Kinesis: Este servicio está pensado para recopilar, procesar y analizar transmisiones de datos y vídeos en tiempo real. No es un servicio especifico de Internet de las Cosas. Está pensado para adaptar las acciones en función de los datos recibidos, adecuando la solución a los datos [30].

- Por otra parte, una ventaja determinante de Azure es que se integra con el software de Microsoft, evitando la necesidad de un servidor adicional. Por tanto muchas empresas e instituciones que utilizan Microsoft para trabajar (outlook, office 365, ...) deciden utilizar Azure directamente por las ventajas de capacidad, reducción de costes y mayor rendimiento que supone.

    - Azure Iot Hub: Este servicio esta pensado para poder conectarse casi con cualquier dispositivo. Desde esta plataforma en la nube se pueden administrar y configurar infinidad de dispositivos. Permite además una integración sencilla con otros módulos de Azure [9].

    - Azure Event Hub: Al igual que AWS Kinesis este servicio no es específico de IoT, si no que está pensado para recibir datos en tiempo real de manera sencilla, segura y escalable. Al igual que Iot Hub este servicio se integra facilmente con otros módulos de Azure [31].

Llegado este punto, ya se han estudiado las diferentes tecnologías necesarias para implementar una solución IoT. A continuación se van a estudiar las plataformas existentes en el mercado para desplegar una plataforma *blockchain*.

2.4. Plataformas DLT

En esta sección se van a estudiar qué plataformas existen en el mercado para desarrollar una *blockchain* en la nube, también conocidas como *Blockchain as a Service* (Baas) [10]. Este tipo de plataformas ofrecen la infraestructura y soporte para garantizar el correcto funcionamiento de una *blockchain*. Es una manera fácil de generar aplicaciones *blockchain*, cada vez más extendida, que a la larga va a permitir que las tecnologías DLT se vayan integrando y normalizando [32].

1. Azure Blockchain Workbench: Se lanzó en 2006, y forma parte de la plataforma de Azure. En ella el usuario puede generar, probar, y desplegar aplicaciones *blockchain* de manera sencilla [33]. Este módulo de Azure se integra fácilmente con muchos otros recursos de Azure, favoreciendo la unión de diferentes tecnologías. Además Azure y por tanto Azure Blockchain Workbench está pensado para no tener que programar, es decir con las nociones básicas de lo las diferentes tecnologías se pueden construir aplicaciones de bastante dificultad técnica. La propia plataforma se encarga de realizar los tecnicismos por debajo de la interfaz del usuario.

2. SAP Cloud-Based Blockchain Service Platform: Esta plataforma salió al mercado en 2017, y su objetivo es aumentar las capacidades de una empresa más allá de los servicios tradicionales ofrecidos por sAP, permitiendo que las empresas puedan incluir la tecnología de registro distribuido en sus metodologías [34].

3. IMB Blockchain Platform: Apareció en 2017, es una de las plataformas más extendidas y potentes del mercado y grandes bancos y empresas la utilizan. Ofrece una solución DLT muy completa que permite a los usuarios desarrollar, operar y vigilar ecosistemas *blockchain* de manera rápida, y económica en la nube.

4. Blockchain on AWS: Aparece en 2018, esta plataforma BaaS permite construir *blockchains* y DLTs de manera intuitiva y fácilmente escalable. Esta pensada, al igual que Azure Blockchain Workbench, para integrarse facilmente con otros de los servicios de AWS.

5. Oracle Blockchain Cloud Service: Se lanza en 2017. Su foco esta en ofrecen un servicio DLT espcefícicamente enfocado al mundo empresarial, para ayudar a las empresas a aumentar la confianza y la agilidad de los movimientos y adaptarlas a esta nueva tecnología. Al igual que Azure y AWS tambien está creada para interactuar con el resto de soluciones que ofrece Oracle, como Oracle Supply Chain Management, Oracle ERP u Oracle Cloud Solutions.

6. HPE Mission Critical Blockchain: Por último se analiza la plataforma lanzada por Hewlett Packard Enterprise en 2017. Plataforma diseñada para entornos que necesitan una tolerancia a fallo absoluta sobretodo enfocado a sistemas críticos. Está pensada para ser escalada fácilmente y adaptada a sistemas y protocolos informáticos que se están quedando obsoletos. Esta plataforma se desarrollo de manera conjunta con la empresa de *blockchain software* R3.

2.5. Casos de uso relacionados

A continuación se van a exponer algunos casos de uso, agrupados por sectores, en los que también se utilizan las tecnologías de Internet de las Cosas junto con *blockchain* [35] [36] [37].

1. **Cadena de suministro y logística**: Las cadenas de suministro actuales pueden ser tan largas que dan la vuelta al mundo, y en ellas participan multitud de interesados, proveedores de materia prima, intermediarios, transportistas, etc y por tanto llevar un seguimiento tanto del bien como de las facturas o cobros puede ser un trabajo complicado y tedioso que puede durar meses. Por esta razón muchas empresas están adaptando sensores IoT en los diferentes eslabones de la cadena, mejorando la visibilidad y trazabilidad. Pero si además los sensores IoT se integran en redes *blockchain* entonces se aporta transparencia y fiabilidad. Gracias a los contratos inteligentes las diferente partes pueden tener información de la mercancía en tiempo real y que en manos de quien se encuentra en dicho momento.

   Un ejemplo por tanto de estas tecnologías en el ámbito de la cadena de suministro puede ser un productor de ropa y calzado que fabrica y distribuye a miles de tiendas por todo el mundo, y busca que sus productos lleguen a las tiendas con el nivel de alta calidad con el que han sido producidos, por tanto para esta empresa implementar una solución IoT que abarque desde los contenedores en los que circula la ropa en los barcos o aviones

cargueros, a en qué almacenes se encuentran los productos a las entregan en paquetes más pequeños que se hace a las tiendas individuales.

2. **Industria Automovilística**: La digitalización hoy en día es una ventaja competitiva, y en el mundo del automóvil se están produciendo ya vehículos completamente sensorizados y automatizados, es decir la parte de la tecnología de internet de las cosas está ya muy integrada en la industria. Por tanto, aquí las tecnologías DLT especialmente *blockchain* van o estan generando casos de uso muy interesantes como el pago automatizado de combustible, parking inteligente o el control del tráfico dinámico en zonas de congestión.

   Estudiando el caso de una ciudad con aparcamiento inteligente, la manera en la que se llevaría a cabo es la siguiente: se deben sensorizar todas las plazas de aparcamiento de la ciudad, para que haya visibilidad de los sitios que están ocupados y los que están libres. Por otra parte los usuarios que quieran aparcar deben de generarse una cartera digital desde la cual se realizarán los cobros al aparcar. Además de esta manera la agencia reguladora del estacionamiento puede ver quién, cuándo y dónde se está produciendo una infracción y avisar o en el caso de que no se arregle la situación multar a la persona indicada.

3. **Hogares inteligentes**: Los dispositivos IoT cada vez están más presentes en la vida cotidiana, y por tanto poco a poco se integran en las casas, un ejemplo muy común son los contadores inteligentes, o los sistemas de vigilancia. Pero, ¿por qué dejar en manos de un tercero la seguridad del hogar? Con la combinación de IoT y *blockchain* en donde se pueden implementar *smart contracts* que actúan acorde a unas normas impuestas por el usuario la seguridad de las casas puede gestionarse desde el teléfono personal del propietario.

   Se podría desarrollar una *blockchain* en la que se implemente un smart contract que vigile la casa, avise de anomalías, que además integrase funciones como persianas automáticas, regulación de temperatura, y otras métricas que pudieran ser interesantes. Para aportar segurida se pueden incluir medidas biométricas, de manera que solo las personas adecuadas puedan hacer modificaciones en los parámetros de la aplicación.

4. **Economía Compartida**: Este concepto es cada vez más popular, siendo adoptado por todo el planeta. La unión de IoT y *blockchain* en este sector, lo que puede conllevar es la eliminación de intermediarios a la hora de realizar un pago/cobro. Esto aumentara la confianza en el sistema, reduce los costes de transacción y acelera el número de transacciones.

   Un caso de uso de este tipo de sistemas podría ser una plataforma que aprobase contratos de manera automática gracias a la firma electrónica. De esta manera se eliminaría la figura notarial, que reduciría considerablemente los costes. Este tipo de aplicaciones ya existem, por lo general de manera interna en las empresas, pero si este sistema se extendiese a lo largo de las industrias y sectores, la manera en que se realizan los negocios cambiaría considerablemente.

5. **Industria farmacéutica**: El caso de uso anterior podría aplicarse al fundo farmaceutico, ya que exiten problemas de falsificación de medicamentos, si se desarrollase un método para trazar los cambios legales de mercancía de medicamentos, probablemente este problema se viese reducido. Otra aplicación para el mundo farmaceutico es la del *stock* de medicinas en farmacia. En españa una farmacia tiene que aventurarse a

pedir los medicamentos sin tener una certeza de que se vayan a vender todos. Esto es especialmente relevante para las medicinas que san más caras ya que si no se venden la farmacia debe hacerse cargo del coste. Una aplicacion interesante es sensorizar el almacén de las farmacias y que de manera automática a través de smart contracts se vayan haciendo pedidos o modificando las cantidades a pedir según se vayan consumiendo los medicamentos.

6. **Agricultura**: El mundo agrario es muy sacrificado ya que la demanda de alimentos es cada vez mayor, pero la competencia y competitividad del mercado es altísima y el consumidor final busca consumir un producto con baja huella de carbono y que garantice la transparencia en su cadena de suministro. Ya existen soluciones que sensorizan los diferentes eslabones de la cadena de suministro, similares a casos explicados más arriba.

   Una solución enfocada a proteger a los agricultores frente a las inclemencias del tiempo, es asegurar sus terrenos con una póliza en *blockchain*. Para ello habría que sensorizar el campo, poniendo sensores de temperatura, humedad, etc.. que reporten a la cadena de bloques para que en caso de tormenta, huracán, lluvias torrenciales, fuegos o cualquier otro desastre medioambiental quede certeza de cuando ha ocurrido, dónde y bajo qué circunstancias. Así el seguro no puede negarle al agricultor una indemnización por no poder demostrar los hechos.

3

Diseño y Desarrollo de la aplicación

"..."
...

En este capítulo se expone el diseño y el desarrollo de la aplicación del trabajo, que consiste en garantizar la calidad a lo largo de la cadena de suministro en servicios de transporte refrigerado. Se va a revisar la selección de tecnologías y a desgranar los pasos del desarrollo.

3.1. Arquitectura y selección de tecnologías

En el capítulo anterior se exponían las tecnologías más utilizadas para aplicaciones de IoT al igual que las más usadas en relación a *blockchain*. En esta sección se expone por qué se escoge cada tecnología, esto se puede ver de manera rápida y visual en la Figure 3.1.

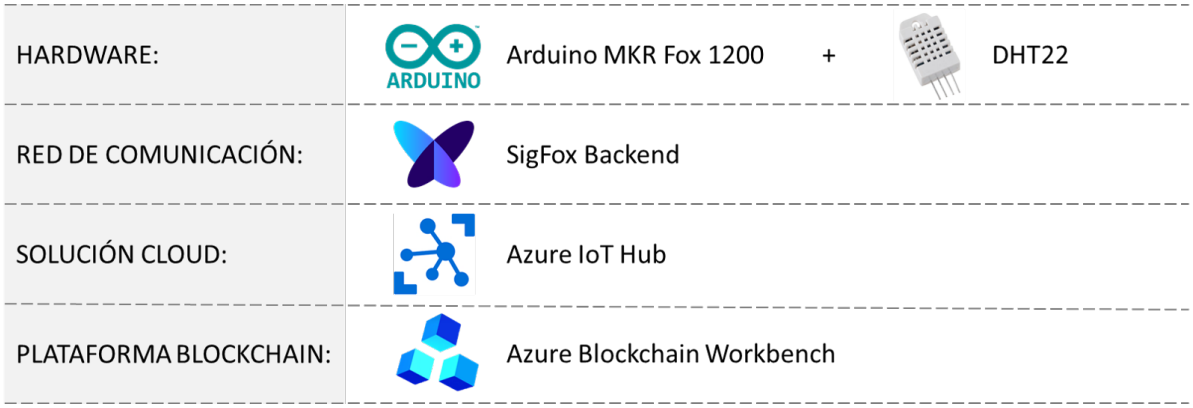


Figure 3.1. Tecnologías elegidas para desarrollar la aplicación del trabajo.

Empezando por las plataformas *hardware*, ya se había acotado la selección a dos plataformas concretas: Particle IoT *starter kit* y la familia de placas Arduino MKR. Analizando más en detalle la placa Particle IoT *starter kit* se observa que cuenta con conectividad Wi-Fi y Bluetooth, no

tiene conectividad a la red móvil. Esto puede suponer un problema ya que en el caso de uso de este proyecto el conjunto sensor más placa se instalaría en la parte de atrás de un camión lugar desde el cual emitir una señal Wi-Fi no es muy potente ya que la emisión de datos es de relativo corto alcance mediante Wi-Fi. Este tema se trata con mayor profundidad en la Section 2.2.

Por otra parte, de todas las placas de la familia Arduino MKR, se escoge la placa Arduino MKR Fox 1200. Esta placa tiene un módulo integrado de conectividad a la red europea de SigFox hace que esa placa sea una gran opción para desarrollar el caso de uso. En la Section 2.2 se explica con más profundidad qué es SigFox y qué beneficios ofrece. Estas redes pertenecen a las redes LPWAN (*Low power wide area network* sin pertenecer a las redes de las empresas de telefonía móvil, y por tanto no necesita licencia, pero tienen alcance mundial. Por todas las ventajas que ofrecen las placas de Arduino y lo conveniente que es la conectividad que ofrece la placa MKR Fox 1200. Además esta placa viene con un año de subscripción gratis a las redes de SigFox.

Lo siguiente que se debe de elegir es el sensor que se va a utilizar, para ellos se toma de base los estudiados en el análisis de la Section 2.1.3 y bajo los criterios necesarios para la aplicación: se desea que se midan temperaturas negativas y positivas y un amplio rango de humedad relativa, preferiblemente el 100%. Además se buscar que la variación de lectura de la humedad relativa esté dentro del rango del 5% respecto a la medida real, y que la variación de la medida de temperatura no supere el grado centígrado respecto a la temperatura real.

Respecto a los sensores de temperatura

- Sensor de temperatura LM35: Aunque es un sensor muy común y fácil de usar,podría ser una opción aunque es preferible seleccionar un sensor con mayor precisión.
- Sensor de temperatura TMP36: Este sensor podría ser válido para el proyecto, pero la precisión del mismo es de $\pm$ 2ºC, que es demasiada variación y por tanto no es admisible para este proyecto.
- Sensor de temperatura TC74: Este sensor digital da una salida bastante buena respecto al rango pero la precisión es baja, puede variar 2 o incluso 3ºC por tanto tampoco es el más indicado.

El sensor de humedad estudiado, el sensor SHT7x, es complejo de usar y además bastante caro.

Estas opciones resultarían en utilizar dos sensores uno para temperatura y otro para humedad. Para simplificar y crear un dispositivo IoT compacto se opta por utilizar un único sensor.Aquí se encuentran los sensores DHT11 y DHT22. El DHT11 a pesar de ser muy común, no mide temperaturas negativas. Se elige por tanto el sensor DHT22 por permitir lecturas de temperaturas negativas, todo el rango de humedad relativa y precisión en las medidas dentro de los rangos deseados. Tras los anexos y las referencias del trabajo se encuentra la hoja de características del sensor. Tras los anexos y las referencias del trabajo se encuentra la hoja de características del sensor.

A continuación se debe elegir que *software development kit* se va a utilizar. De todas las estudiadas: Node-RED, Flutter, Thinger.io y Arduino se decide utilizar esta última. Entre las razones principales se encuentra que la plataforma *hardware* escogida es de esta misma casa

y esto hace que la compatibilidad sea del 100%. Además el conocimiento previo de esta herramienta favorece su uso sobre otras con las que no se ha trabajado nunca.

El siguiente eslabón es la red que va a utilizar el caso de uso. Analizando las redes por su alcance podemos observar lo siguiente:

- PAN: Este alcance de tan solo unos metros se queda corto para el caso de transporte refrigerado.
- HAN: Estas redes que alcanzan decenas de metros se quedan cortas para la flota de camiones que llevarían el dispositivo.
- WLAN: Las redes WLAN tienen el alcance de un edificio más o menos, por tanto aunque abarcan más espacio físico que las redes personales o las domésticas, no son suficientes para cubrir el alcance necesario para el proyecto.
- MAN: Este tipo de redes podrían servir para el proyecto, ya que dependiendo del servicio, para el caso de transporte refrigerado puede darse el caso de que el almacén desde el que se reparte y los puntos de entrega del producto estén en la misma ciudad.
- WAN: Al ser redes que cubren todo el planeta, claramente, son la mejor solución para el proyecto.

Como ya se ha visto en la Section 2.2 dentro de las redes WAN existen unas redes que son muy usadas para desarrollos de proyectos IoT llamadas LPWAN. Dentro de esta categoría se diferencian dos grandes grupos las redes de operadores de telefonía móvil (MNO) y las redes que no pertenecen a operadores de telefonía móvil. Las redes MNO necesitan licencia y por lo tanto generan soluciones más caras, ademas a cada dispositivo se le debe poner una tarjeta SIM. Por tanto dentro de las redes que no MNO, se encuentran LoRaWan y SigFox, como la placa *hardware* ya tiene un módulo a las redes de SigFox lo más lógico es desarrollar una aplicación que las utilice.

De las redes MNO únicamente sería viable realizar el proyecto con LTE ya que NB-IoT requiere que el dispositivo este estático pero para trabajar con una LTE hay que pagar tanto la tarjeta SIM como la subscripción a la red. Para poder utilizar LoRaWAN, además de la placa de Arduino escogida sería necesario comprar el *shield* de LoRa, que supone un gasto adicional. Lo más lógico es utilizar las redes de SigFox ya que la placa escogida está preparada y la subscripción del primer año es gratuita.

Conocidos los protocolos de transporte se puede evaluar qué protocolo se utiliza en el caso de uso en la parte de la red que comunica la placa con la nube de SigFox. Pero como SigFox es una empresa privada ha decidido crear su propio protocolo para sus redes, poniendo mucho foco en la seguridad de las soluciones que ofrece. Ellos mismos comunican que no utilizan ni TCP ni UDP.

Respecto a la plataforma *cloud* a utilizar en el trabajo, se debate entre Amazon *web services* y Azure, ya que son las que tienen enlace directo desde SigFox. La Universidad Pontificia Comillas funciona con Microsoft y por tanto es más fácil implementar una solución con Azure, que tambien es de Microsoft que AWS. Ambas son soluciones muy potentes que permiten llevar a cabo el caso de uso, pero se decide trabajar con la plataforma con la que se tienen más facilidades para trabajar, en este caso, Azure.

Por último de las plataformas para desplegar una *blockchain*, a pesar de que todas las opciones estudiadas en la Section 2.4 son muy buenas opciones, exceptuando quizás la plataforma de HP por ser específica para sistemas críticos. Para el desarrollo del caso de uso de este trabajo se va a implementar Azure Blockchain Workbench por la facilidad de enlace a otros módulos de Azure.

3.2. Desarrollo

En está sección se va a exponer el desarrollo paso a paso del caso de uso. Se separa en subsecciones para tratar independientemente las diferentes partes del proceso. El orden del desarrollo es el siguiente: la implementación *hardware*, la conexión con las redes de SigFox, de SigFox a Azure, la preparación de la plataforma de *blockchain* y la recepción de datos.

3.2.1. Implementación *Hardware*

La placa a utilizar se compone de dos elementos, la propia placa y la antena que permite la comunicación a la red.

En la Figure 3.2 se pueden ver recuadrados en rojo los dos elementos, anclados en una placa base. Se incluye la hoja de características de la placa, que permite conocer sus entradas y cómo se debe alimentar. En este caso la placa se alimenta desde el ordenador a través de la entrada micro-USB que da una tensión de 5V.

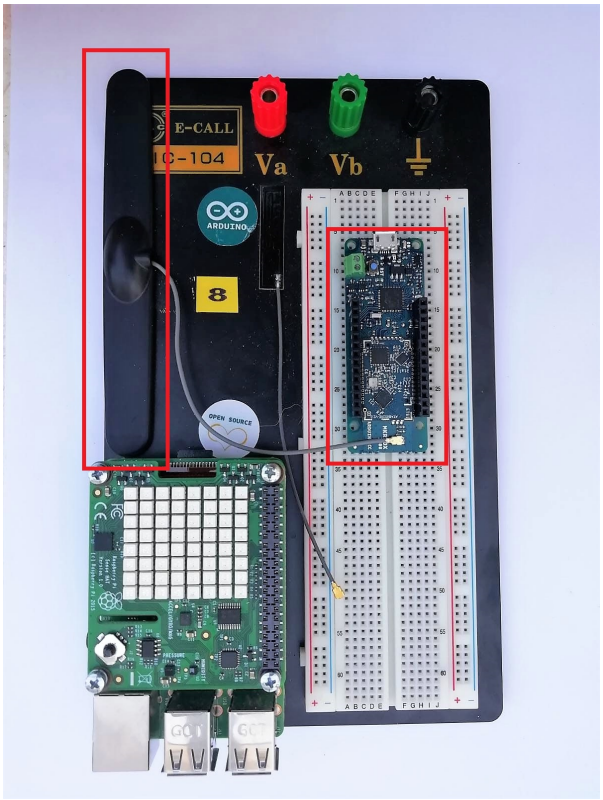


Figure 3.2. Plataforma *hardware*.

El sensor por su parte se puede ver en la Figure 3.3. Se puede ver que incorpora tres cables,gracias a la hoja de características del sensor, se conoce cómo se debe cablear el sensor en la placa.

- Alimentación o Vdd corresponde al cable rojo. Debe estar en el rango 3,3V a 5,5V DC. En este caso la placa se va a conectar a +3,3V en la placa.
- Salida digital de datos el cable amarillo. Debe por tanto conectarse a una entrada digital de la placa, se escoge la entrada D5.
- Tierra corresponde al cable negro. Se acopla a la entrada de tierra GND, contigua a la alimentación de 3,3V.

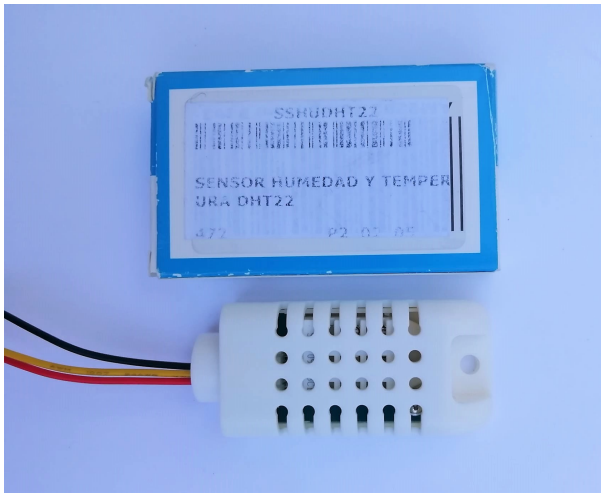


Figure 3.3. Sensor DHT22.

En la Figure 3.4 se puede ver el conjunto de ambos elementos *hardware*. La alimentación de la placa al ordenador y la conexión del sensor.

3.2.2. Conexión con SigFox

La SDK de Arduino, llamada Arduino IDE, es una aplicación de escritorio que permite generar códigos y subirlos a la placa a través de la entrada USB. En el caso de la aplicación de transporte refrigerado lo que se busca es obtener del sensor la temperatura y humedad relativa del ambiente, se desea registrar el momento en que se hace la lectura y la localización. Del *hardware* se van a obtener los dos primero parámetros (temperatura y humedad) y el sello temporal y localización se van a obtener posteriormente de la plataforma de SigFox.

En el Appendix C se muestra el código que se sube a la placa [38] [39]. Una vez está subido no hace falta editarlo, siempre que la placa esté enchufada a la alimentación va a proceder y envíar los datos de temperatura y humedad cada minuto (tiempo que se establece en el código, se podría variar si se quiere establecer otro tiempo). Para el caso de uso como se quiere realizar una aplicación eficaz, se piden lecturas cada minuto, así si se sale de rango se puede detectar el fallo rápidamente y actuar de inmediato.

El código parte por parte se compone primero de un conjunto de librerías que definen el funcionamiento de los sensores Adafruit, especialmente el DHT, la conexión con SigFox y el método para sacar JSON desde Arduino IDE, aunque finalmente esta librería realmente no se utiliza.

A continuación se definen las constantes: el pin al que se conecta la salida de los datos del sensor, el pin D5, cómo se explicaba más arriba, y parámetros del sensor. Se definen también las variables de temperatura y humedad como *floats*, para permitir captar los decimales de las lecturas.

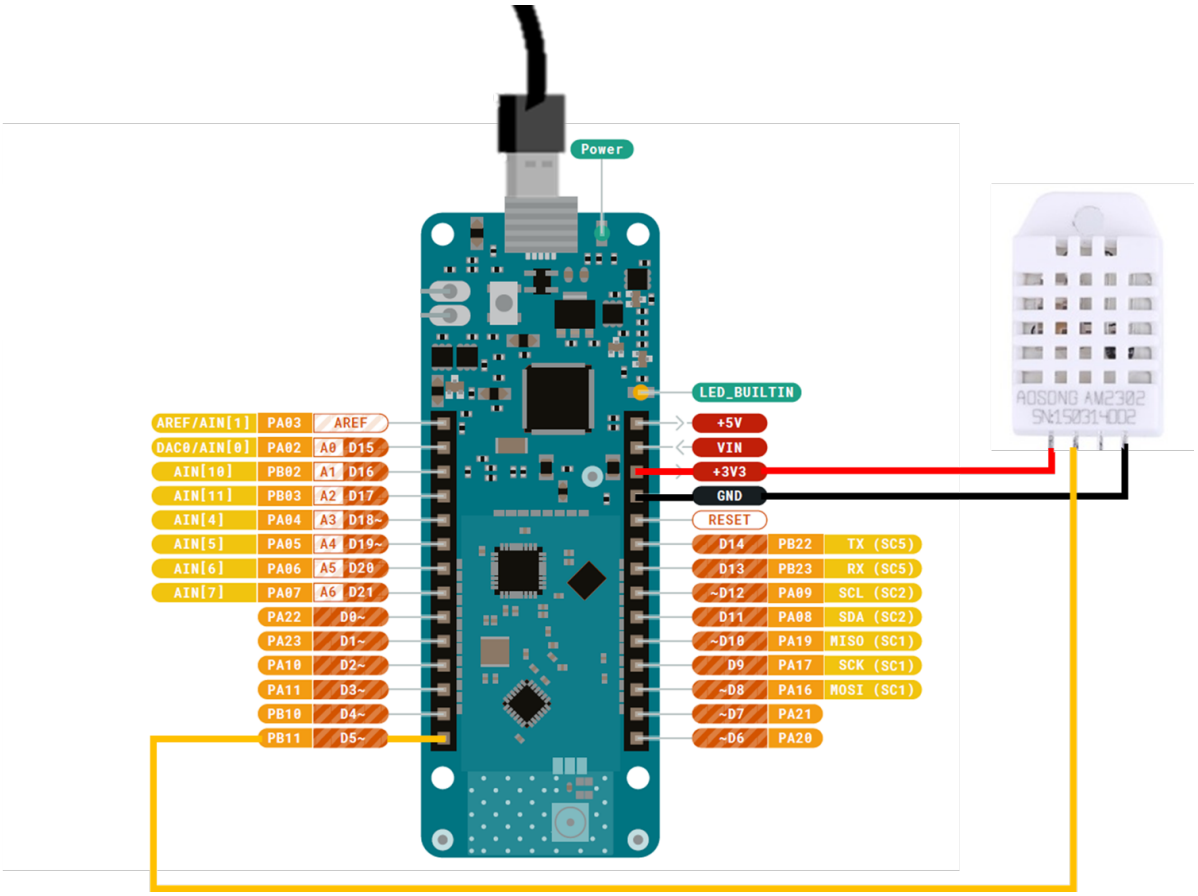


Figure 3.4. Cableado del sensor en la placa de Arduino.

El siguiente bloque del código se corresponde con las configuraciones del código, al correr el código pasa por ahí una vez y luego se mantiene corriendo el siguiente bloque que como su nombre indica *void loop* es un bucle. En el *void settup*, se inicializa el monitor serie, el sensor y la conexión con SigFox. Se añade un mensaje de error para el caso en el que no se haya inicializado correctamente el módulo de conexión a las redes de SigFox.

Por último en el bucle se lee la temperatura y la humedad, se manda un paquete con ambas variables a SigFox y se escriben tambien estas variables en el monitor serie para comprobar que las lecturas que se reciben en la nube de SigFox se corresponden a las leidas. Se añade un delay de 60 segundos hasta la siguiente medida.

Por otra parte, antes de poder enviar datos hay que registrar el dispositivo en el *backend* de SigFox. El *backend* [40] es la página de SigFox desde la cual se pueden gestionar los dispositivos [40]. Cada dispositivo es único y en cuanto se registra se activa la subscripción inicial gratuita, el dispositivo registrado para el desarrollo es el 1CF5D8. Una vez registrado se puede proceder a enviar datos.

3.2.3. De SigFox a Azure

Para tener una idea más visual del proceso que se está llevando a cabo, se tiene la Figure 3.5, donde se pueden ver las principales conexiones que se deben hacer para enviar los datos desde el sensor físico hasta la *blockchain*. La primera conexión ya se ha realizado que es entre el sensor y la nube de SigFox. La rama inferior del esquema hace referencia a los datos que se piden a través del Monitor Serie para comprobar que son iguales a los recibidos en el *backend*. En esta sección se va a tratar la siguiente conexión: entre SigFox y Azure Iot Hub.

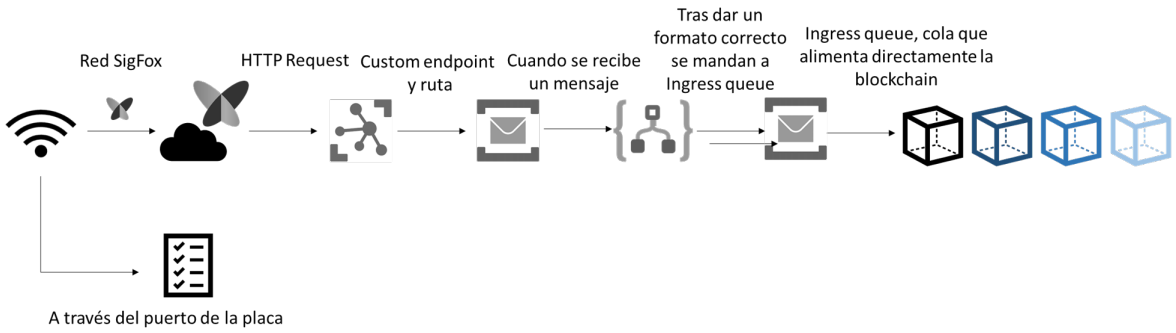


Figure 3.5. Trayectoria de los datos a lo largo de la aplicación.

Antes de proceder a configurar la redirección de los mensajes recibidos en SigFox, hay que preparar el IoT Hub. Este módulo de Azure permite conectar multitud de dispositivos de manera segura en su *back-end cloud* y trabajar con el resto de aplicaciones de Azure [9].

Para poder implementarlo se debe de tener un directorio en Azure con una subscripción activa, dentro de ese directorio se crea un grupo de recursos para albergar específicamente el IoT Hub de la aplicación. Para este trabajo se utiliza el directorio de carmenolleroshotmail.onmicrosoft.com y el usuario desde el que se va a desarrollar todo el trabajo es EmpresaTFM@carmenolleroshotmail.onmicrosoft.com. El grupo de recursos creado expresamente para el Iot Hub se llama IoTResource. El módulo Iot Hub desplegado tiene el nombre de IotHub-TFM.

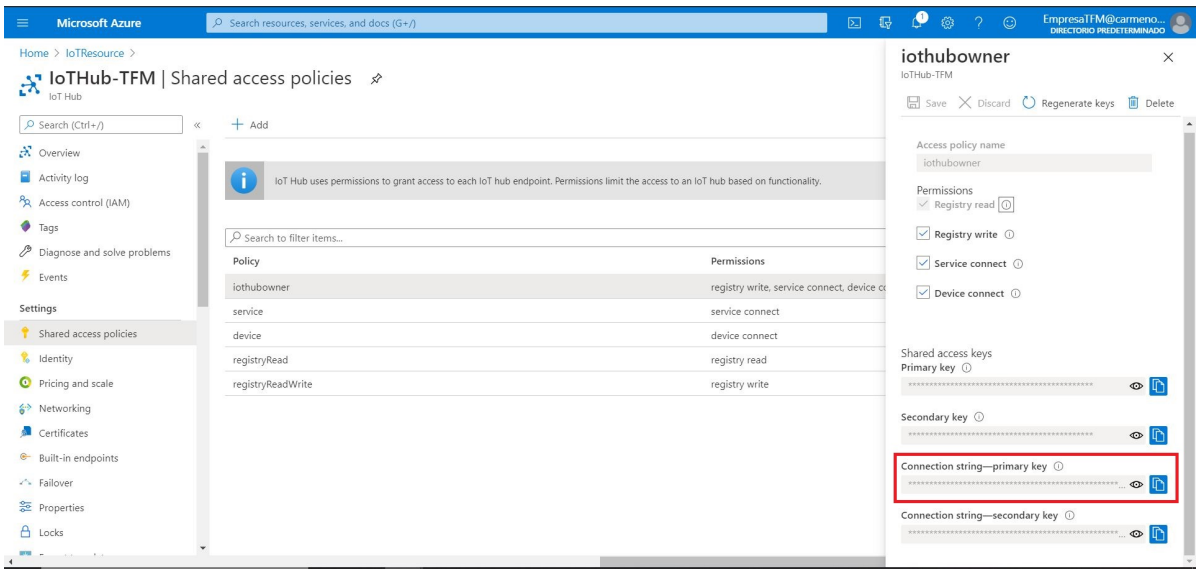


Figure 3.6. Llave *Connection string—primary key* de IoT Hub.

Una vez desplegado el IoT Hub, se puede realizar la conexión con SigFox, esto se hace a través de la llave: *Connection string—primary key*, se encuentra recuadrada en la Figure 3.6.

En SigFox, se prepara un *callback* que es una llamada de vuelta, es decir una accion que toma cuando se reciben datos. Para este desarrollo se utiliza un *callback* de tipo dato preparado para realizar una conexión con IoT Hub, como se vio en la Section 2.2 es uno de los cuatro *callback* predefinidos que tiene en *backend* de SigFox. En la configuración de este *callback* se introduce la llave copiada de IoT Hub (*Connection string—primary key*) y se escribe el codigo JSON que se desea envíar a Azure. Con tan solo estas dos configuraciones, que se pueden ver en la Figure 3.7, la conexión y el código queda configurado el envío de datos a IoT Hub.

A continuación se incluye el código JSON que se manda a IoT Hub. En el se pide que se envíen el identificados del dispositivo, los datos del usuario (en hexadecimal), el sello temporal (en formato unix), el numero de secuencia del mensaje, el identificador del tipo de dispositivo y las variables de interés: temperatura y humedad recibidas del sensor.

No se incluye la ubicación desde la que proviene el mensaje debido a que esto no se puede hacer en el tipo de *callback* que se está configurando.

```
{
"DeviceID" : "{device}",
"data" : "{data}",
"time" : "{time}",
"seqNumber" : "{seqNumber}",
"deviceTypeId" : "{deviceTypeId}",
"humidity" : "{customData#hum}",
"temperature" : "{customData#temp}"
}
```

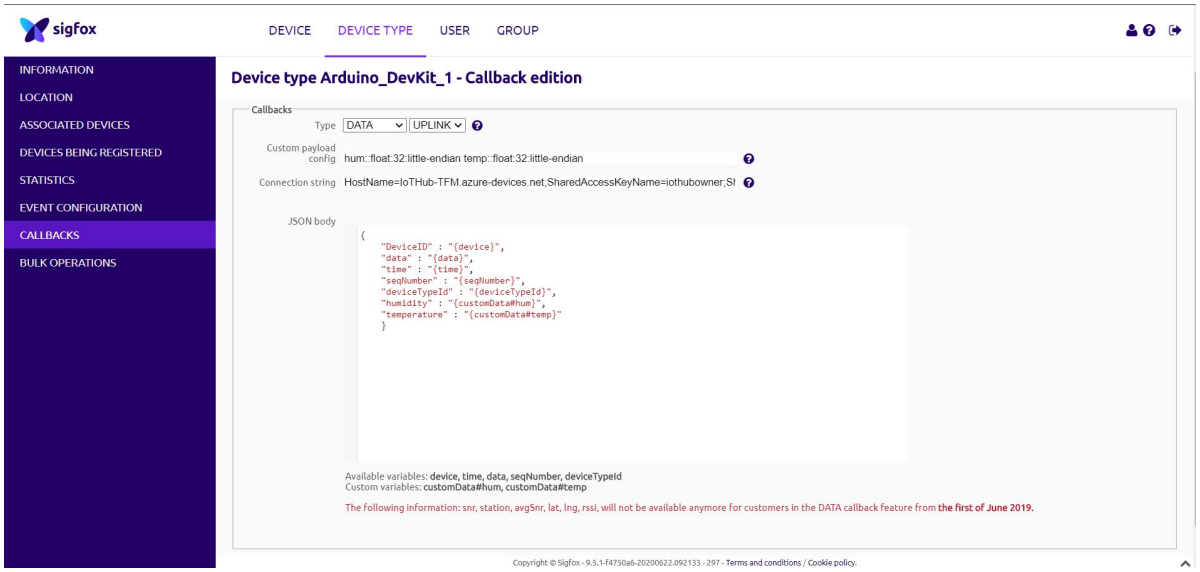


Figure 3.7. Configuración de la conexión a IoT Hub desde SigFox *backend*.

3.2.4. Procesamiento en Azure

Desde el Iot Hub hasta Azure *Blockchain Workbench* hay un paseo por diferentes módulos de Azure, que permiten el transporte de datos, y su modificación para adecuarlos al formato que es capaz de leer la *blockchain*. Este proceso se hace tomando de referencia el repositorio de Github "*Delivering Data from IoT Hub to Azure Blockchain Workbench*" [11]. Los diferentes módulos de Azure están desplegados en tres grupos de recursos diferentes, para no aglomerarlos, además se despliegan en dichos grupos de manera lógica y ordenada. En la Figure 3.8 se puede apreciar que hay desplegado en cada grupo de recursos.

De manera que hay tres grupos de recursos principales uno para la recepción de los datos desde SigFox, otro para la modificación de y transporte de los mismos y por último un grupo que contiene todos los módulos que aparecen al desplegar Azure *Blockchain Workbench*. A pesar de que en el segundo y tercer bloque aparecen muchos recursos, no todos ellos han de trabajarse para desplegar la aplicación. Hay que recordar la Figure 3.5, donde se ven los módulos que sí que necesitan ser trabajados.

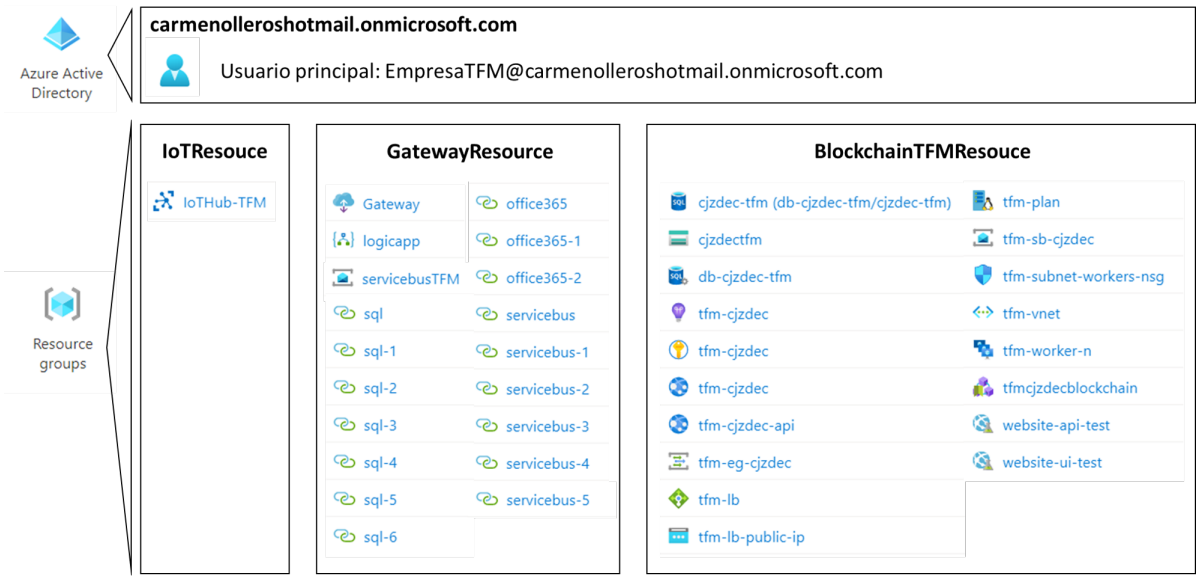


Figure 3.8. Estructura de los módulos de Azure desarrollados.

Tras tener el IoTHub-TFM funcionando y para continuar el camino hacia Azure *Blockchain Workbench* se crea el segundo grupo de recursos: *GatewayResource* donde se crea un *Service Bus* llamado servicebusTFM. Este módulo es una plataforma de mensajería en la nube también conocido como mensajería como servicio (MaaS). Permite enviar los mensajes recibidos en Iot Hub a una aplicación lógica que cambia el formato del mensaje recibido. Para poder realizar este envío de datos se debe crear una cola dentro de servicebusTFM, y generar una ruta en IoTHub-TFM que conecte los datos recibidos con dicha cola Figure 3.9. La cola en cuestión se ha llamado "*myqueue*".

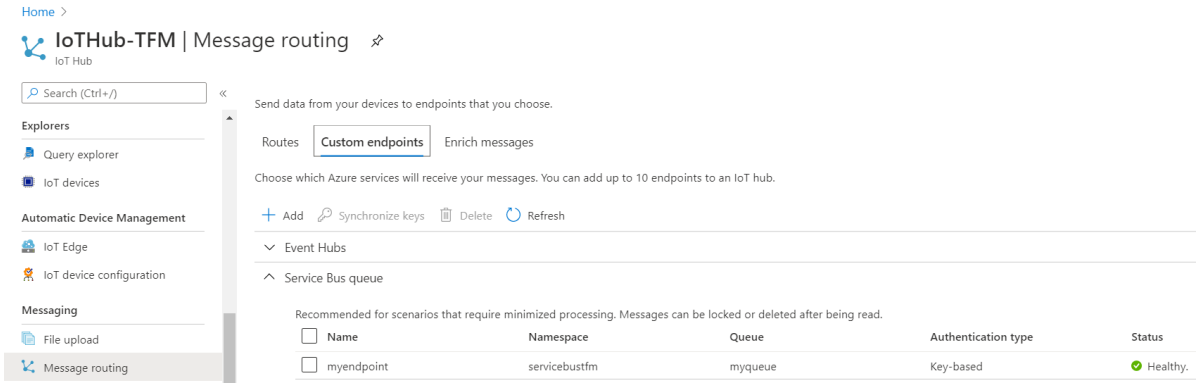


Figure 3.9. Ruta y *endpoit* que asocian IoTHub-TFM con la cola *myqueue*.

En el grupo de recursos GatewayResource se despliega un módulo llamado *Logic App* que es el que va a tratar los datos y prepararlos para adecuarlos al formato de *blockchain*. Este servicio permite captar datos de multitud de aplicaciones, pueden ser fuentes ajenas a Azure y de una manera dinámica, sin necesidad de escribir grandes códigos, editar y crear flujos de trabajo [12]. Para la aplicación, el servicio de *Logic App* va a comenzar con un detonante que una vez cada minuto revisa si hay mensajes nuevos en la cola *myqueue*.

Previo a continuar se debe desplegar Azure *Blockchain Workbench*, que se despliega en un grupo de recursos nuevo, BlochchainTFMResource. Este servicio realmente está formado por un cúmulo de módulos de Azure, ya preparados para funcionar. Entre todos los módulos desplegados (que se pueden apreciar en la Figure 3.8, se encuentra un servidor SQL: db-cjzdec-

tfm. Siguiendo los pasos del repositorio de GitHub mencionado más arriba, se debe lanzar una *query* o consulta para guardar una serie de procedimientos que deben de quedarse almacenados en la base de datos SQL. El procedimiento de interés es GetContractInfoForDeviceID, que para cada dispositivo va a obtener la información del *smart contract*. Pero para que el servicio de [12] pueda acceder a esta base de datos, debe de existir una conexión local al servidor SQL. Es decir hay que establecer una puerta de enlace al servidor SQL en el ordenador desde el que se vaya trabajar el servicio. Para ello existe una aplicación llamada *On-premises data gateway*, que permite configurar dicho enlace rápidamente siguiente las instrucciones de Azure [41].

Con Azure *Blockchain Workbench* desplegado y el *gateway* preparado se puede proceder a desarrollar el flujo que van a seguir los datos en *Logic App*. Los pasos para preparar los datos son los siguientes:

- Comprobar cada minuto si hay algún mensaje nuevo en la cola *myqueue*, si lo hay, dicho mensaje se saca de la cola y entra en la *Logic App* para ser editado Figure 3.10.

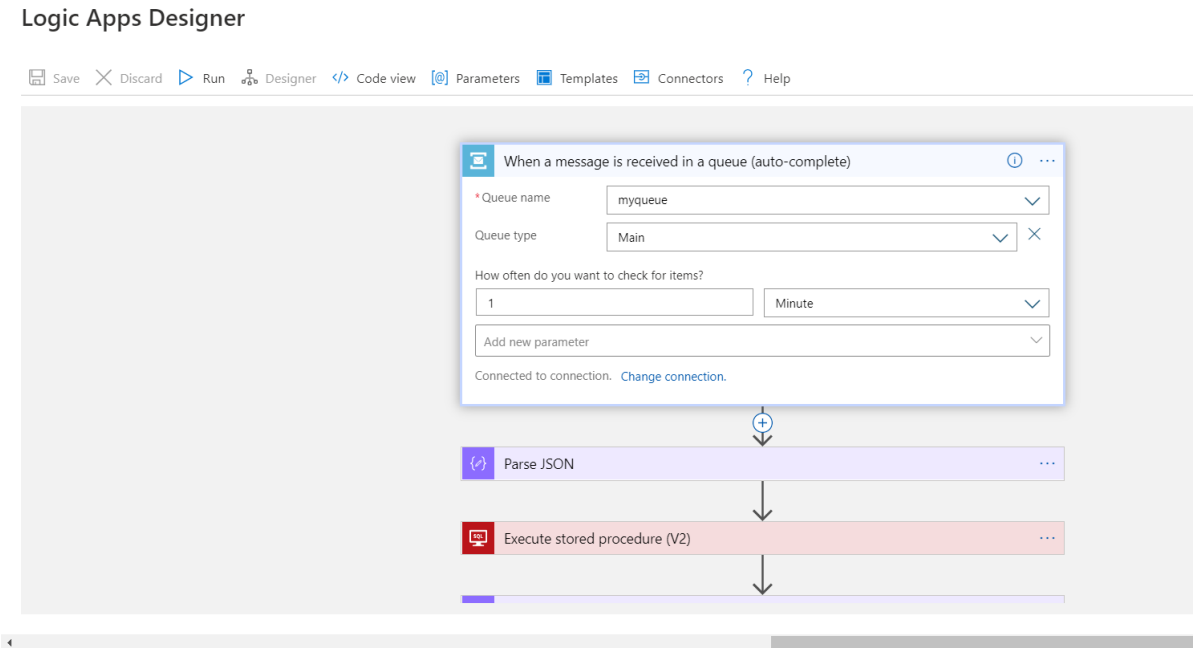


Figure 3.10. Módulo inicial que chequea Service Bus .

- El segundo paso es parsear los datos para poder leer el mensaje recibido: existe una opción dentro del bloque de parsear que dando el código JSON, te lo devuelve parseado. Este módulo se puede ver en la Figure 3.11. Este código se puede encontrar en la Section C.2.

- Obtener en función del id del dispositivo, la información del Smart Contract activo. Esto se hace con un módulo de SQL que busca dentro de los procedimientos almacenados en la base de datos. Aquí entra en juego el *gateway* instalado Figure 3.12.

- El siguiente módulo de parsear JSON prepara los datos para ser introducidos en *blockchain*. El código se muestra en la autorefsec:codbc y más abajo en la **??** se puede apreciar una imagen del módulo en cuestion.

- Se inicializa una variable, llamada *RequestID*, con ella se genera un id para poder trackear más adelante el proceso.

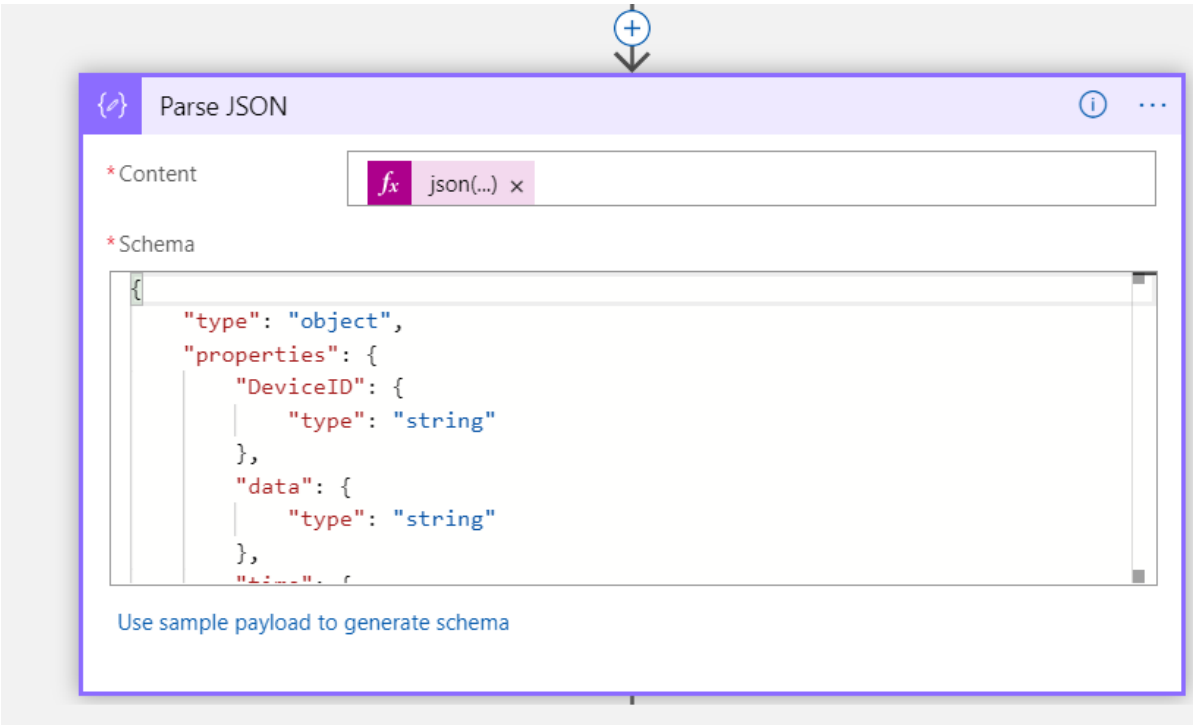


Figure 3.11. Segundo módulo de *Logic App*, que permite leer el mensaje recibido.

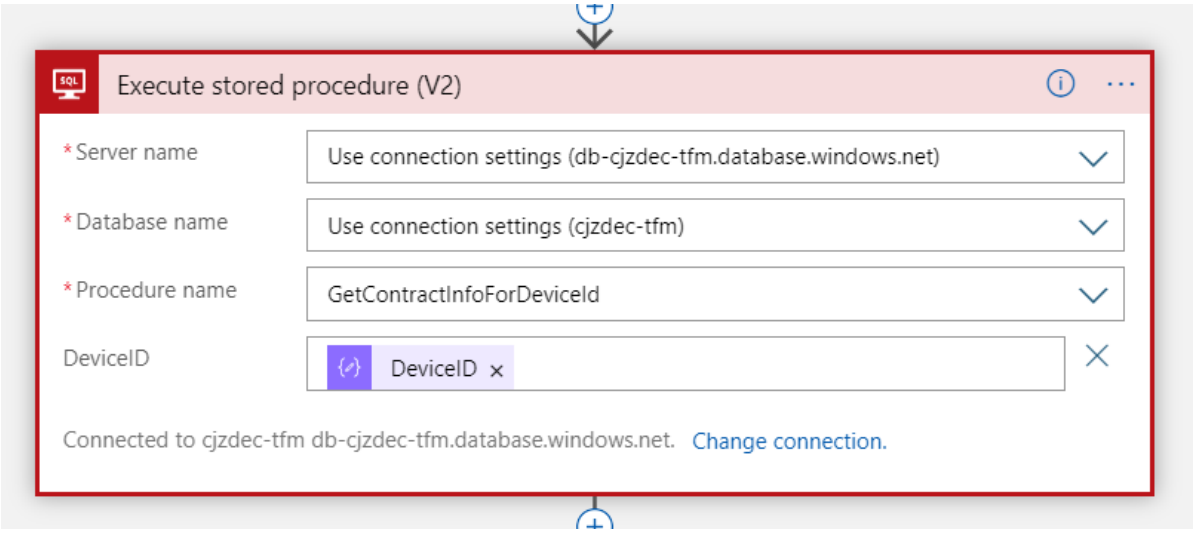


Figure 3.12. Módulo SQL para obtener la información del contrato inteligente.

- Se prepara el sello temporal en varios pasos, esto se puede ver en la Figure 3.14, el primer módulo se corresponde con la inicialización de RequestID, y los tres siguientes con el sello temporal en tiempo UNIX. En el primero se piden los ticks del momento, esto se corresponde con los segundos.

```
1    ticks(utcNow())
```

  El tercer bloque pide los ticks contados desde la medianoche UTC del 1 de enero de 1970.

```
1    ticks('1970-01-01')
```

  Y en el último se realiza la resta de ambos, de manera que ya se tiene el tiempo en formato UNIX.

Figure 3.13. Módulo para preparar los datos al formato de *blockchain*.

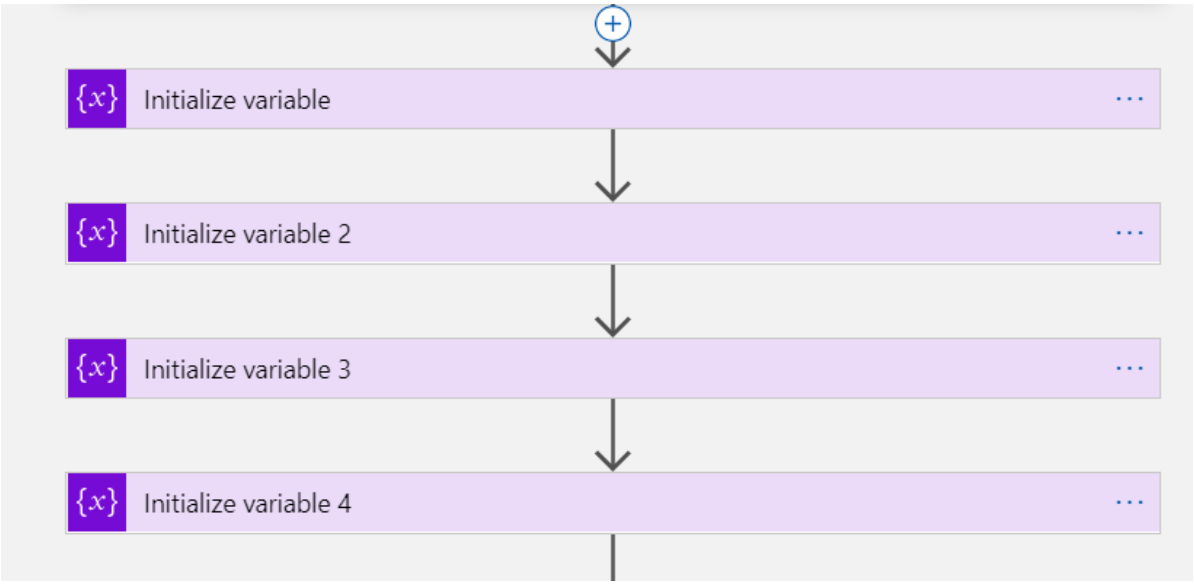


Figure 3.14. Módulos correspondientes a la incialización de RequestID y el sello temporal.

```
1    div(sub(variables('TicksNow'),variables('TicksTo1970'))
        ,10000000)
```

- Ahora el mensaje está preparado, unicamente falta meterlo en la cola de *blockchain*. Este envío es automático, la manera en la que se despliega Azure *Blockchain Workbench* hace que todo mensaje que entra en la *Ingress Queue* sea ingerido por la *blockchain*. Esta cola está dentro del *Service Bus* que se crea al desplegar Azure *Blockchain Workbench* y por tanto las conexiones están hechas. Todo mensaje que llega a esta cola debería ser cogido por el contrato activo pertinente. El módulo final es un módulo de envío a *Service Bus*. El código del mensaje que se envía se rellena con las variables trabajadas, tal y como se muestra en la Figure 3.15

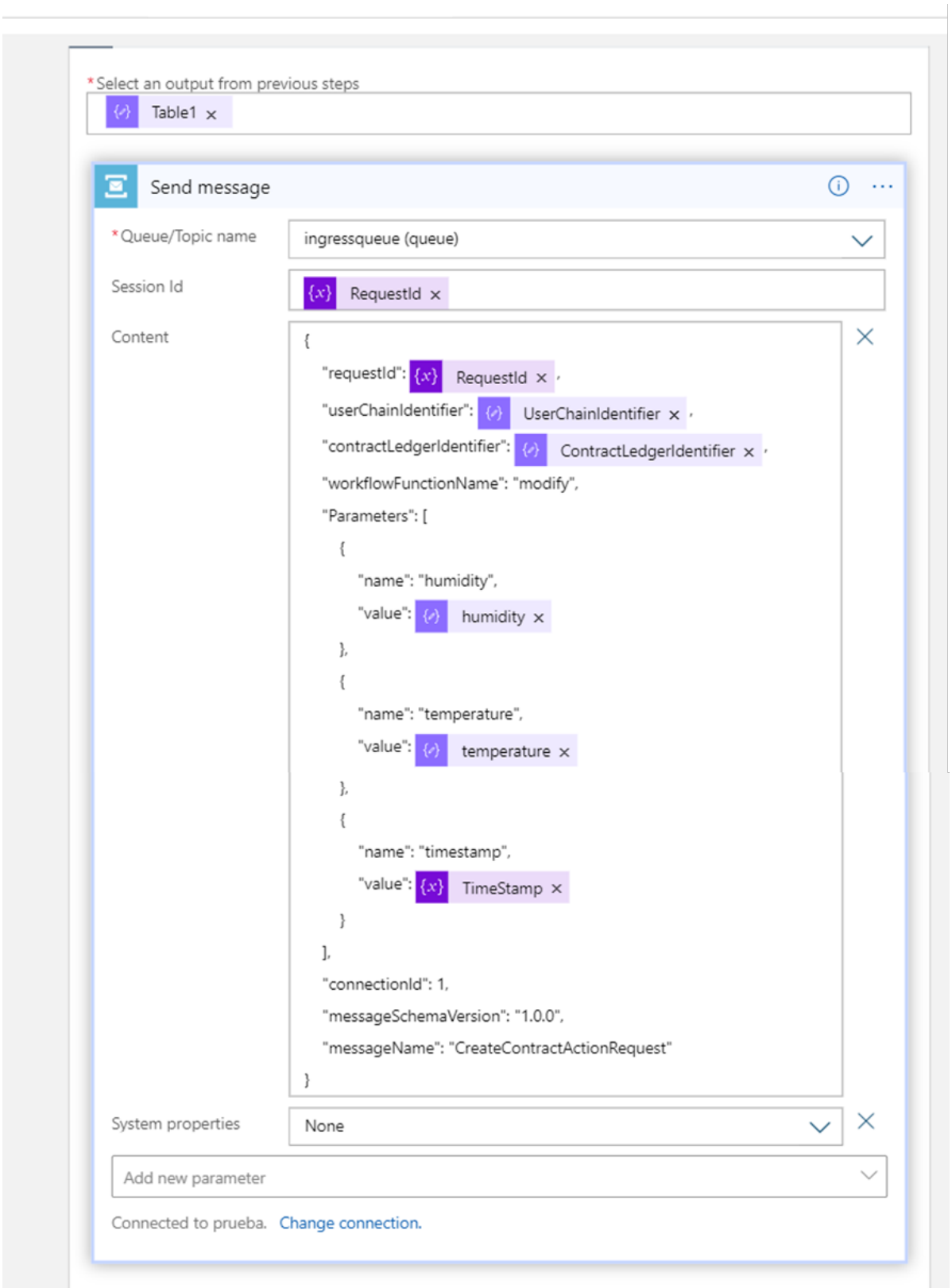


Figure 3.15. Módulos de envío a *blockchain*.

Con este proceso quedaría cerrado el proceso de envío de datos desde la cola *myqueue* a *blockchain*.

3.2.5. Preparación para MS Blockchain Workbench

Para la configuración de la plataforma DLT, en este caso concreto una *blockchain* se sigue el repositorio *Azure Blockchain Hands-on Lab – Microsoft Cloud Workshop* [42]. Los primeros pasos consisten en crear un directorio activo y un usuario administrador pero esto ya está creado, ya que para que funcione, toda la aplicación debe de estar bajo el mismo directorio y usuario principal que el resto de los módulos desplegados.

Para desplegar Azure *Blockchain Workbench* se pide un tipo de autenticación para el usuario principal. Se debe decidir entre utilizar una contraseña o SSH *Public Key* que es una clave criptográfica. Esta es una manera de proteger el sistema ya que la seguridad de una clave criptográfica es mayor.

Para generar dicha clave pública y la clave privada asociada se descarga un software llamado puttygen.exe que permite generar a partir del movimiento aleatorio del ratón dichas contraseñas, se puede ver en la Figure 3.16.

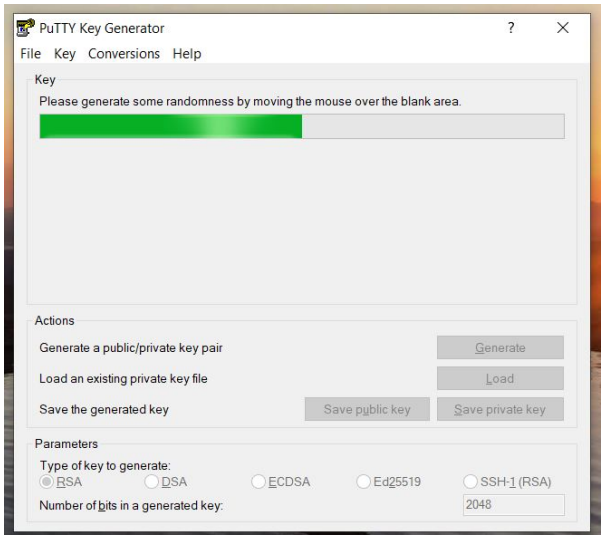


Figure 3.16. Aplicación para generar la clave pública y privada.

Una vez desplegado todo el sistema, en el módulo *App Service* se puede encontrar el link a la aplicación web creada. La primera vez que se entra en dicha aplicación web hay que configurarla, siguiendo los pasos del repositorio y en dos pasos ya se puede acceder directamente. En esta aplicación web es donde se van a subir los contratos inteligente.

En *Visual Studio Code* se prepara el contrato con el que se va a trabajar. Antes de mostrar el contrato se va a exponer qué se quiere conseguir y qué participantes involucra. La máquina de estados del contrato se muestra en la Figure 3.17. En ella se muestran los posibles cambios de estado del contrato. El actor que crea el contrato especifica quién va a ser el siguiente responsable del contrato, pasando de *CREATED* a *IN TRANSIT*. El contrato se mantendrá ahí hasta que un actor establezca que se ha completado el trayecto y se llega a *COMPLETED*. En todo momento, en los estados *CREATED* o *IN TRANSIT* el contrato está preparado para ingerir los datos de telemetría. Si en algún momento los datos no cumplen con las condiciones de temperatura y humedad establecidas, se pasa al estado *OUT OF COMPLIANCE*, si no pasa al estado *COMPLETED* cuando que se completa el trayecto [43].

Los actores involucrados en el proceso son:

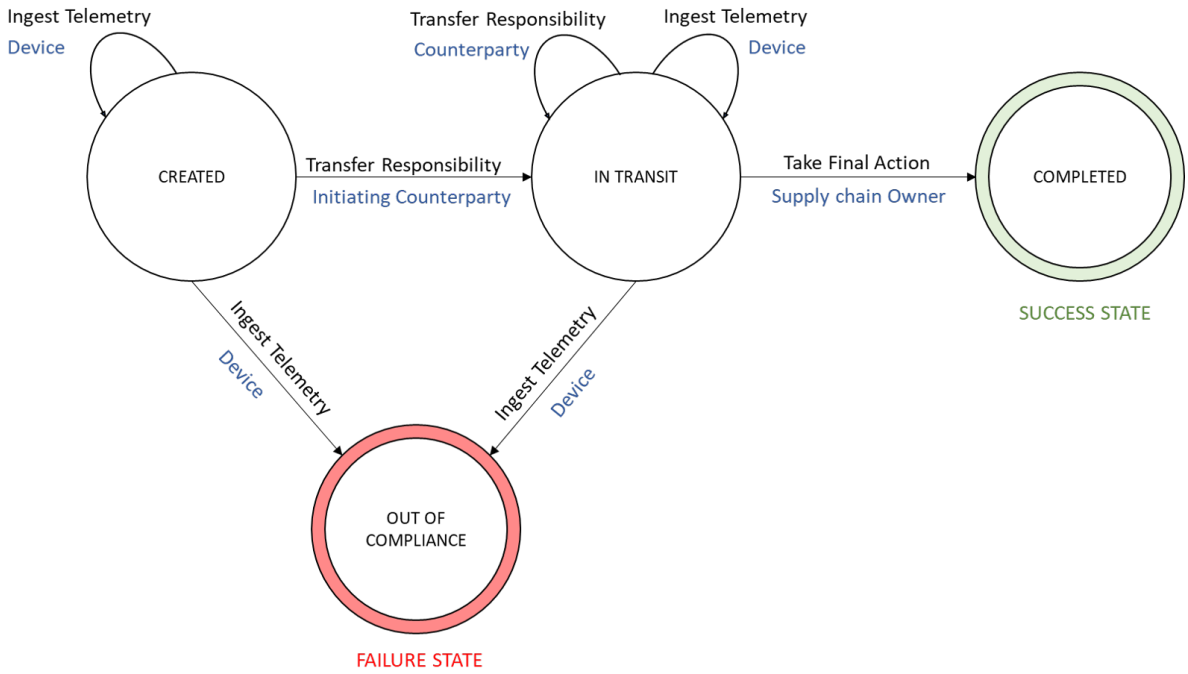


Figure 3.17. Máquina de estados del contrato.

- El usuario principal, ya visto más arriba EmpresaTFM@carmenolleroshotmail.onmicrosoft.com.
- El usuario que se corresponde con el dispositivo device@carmenolleroshotmail.onmicrosoft.com.
- Un usuario que actúa como si fuese el transportista transport@carmenolleroshotmail.onmicrosoft.com
- El correo personal carmen.olleros@hotmail.com que se utiliza como observador y auditor del proceso.

Es importante puntualizar cómo se asocia el usuario del dispositivo con el dispositivo real. Para hacer esta conexión, una vez creado el usuario del dispositivo, se debe acceder a la base de datos de Azure *Blockchain Workbench* y actualizar el id del email con el del dispositivo. De esta manera queda asociado dicho correo electrónico con el dispositivo.

En la Figure 3.18 se muestra la correspondencia entre usuarios y sus roles.

USUARIO	CORREO	ROL
Empresa TFM	EmpresaTFM@carmenolleroshotmail.onmicrosoft.com	Iniciating Counterparty
Transportista	transport@carmenolleroshotmail.onmicrosoft.com	Counterparty
Dispositivo	device@carmenolleroshotmail.onmicrosoft.com	Device
Usuario personal	carmen.olleros@hotmal.com	Auditor/Observer

Figure 3.18. Rol de cada usuario.

Ahora que queda explicado el funcionamiento del contrato, se puede proceder a la parte del código utilizado para desplegarlo. Un *smart contract* necesita dos archivos para funcionar un archivo en Solidity y un archivo en JSON. Solidity es un lenguaje orientado a objetos específico de los *smart contracts*, en este archivo se describe el contrato entero. Por otra JSON es un lenguaje programación intuitivo muy usado para la comunicación de datos, contiene

los metadatos del contrato inteligente y permite que se genere correctamente la aplicacion en *blockchain*.

Para poder trabajar el contrato de la aplicación se realizan dos cursos como forma de preparación y familiarización con estos lenguajes y el mundo de *blockchain*.

Por un lado se realiza un curso en CodeAcademy titulado *Learn the Basics of Blockchain with Python*, en español: Aprenda los básicos de *blockchain* con Python. Con este curso se busca afianzar el conocimiento de la estructura y propiedades principales de las cadenas de bloques. Se explica en profundidad por que esta tecnología proporciona tanta seguridad. Se afianzan dichos conocimientos mediante ejercicios prácticos, creando una pequeña *blockhain* [44].

Otro curso realizado para poder desarrollar mejor la aplicación es un curso de Solidity en una plataforma llamada Cryptozombies.io [45], que permite aprender este lenguaje mientras se desarrolla un juego. Es una manera interactiva y apetecible de aprender un conocimiento técnico. El juego tiene seis capítulos dentro del curso *Solidity Path: Beginner to Intermediate Smart Contracts* de los cuales con cursar los tres primeros se obtienen los conocimientos suficientes para desarrollar el contrato de la aplicación.

Con todo lo anterior y tomando de referencia un contrato proveniente de los ejemplos de aplicación de Azure [43] se tienen las bases para trabajar un contrato inteligente, en la Section C.4 se encuentra el código en Solidity y a continuación el código JSON que lo acompaña.

Para llegar a tener dichos archivos se ha utilizado la aplicación web de Azure *Blockchain Workbench* y Remix, un editor online que permite probar *smart contracts*.

Para el caso de uso de este trabajo interesa guardar los datos de telemetría. Se busca que estos datos se vayan guardando de tal manera que estén asociados a cada contenedor, es decir en función del sensor que los manda en un histórico que se pueda ver después almacenado en la base de datos asociada a *blockchain*. Este histórico de datos es el que va a permitir tener una trazabilidad y una mayor transparencia de lo que ocurre a lo largo del proceso. Con estos datos también es con lo que se puede demostrar que se ha cumplido la cadena de frío, esto interesa tanto a la empresa encargada del transporte, como al proveedor del producto refrigerado, al cliente final y a cualquier empresa auditora, de seguros o de regulación del producto transportado.

A continuación se explica la estructura del contrato de manera que logre guardar los datos en función del contenedor y luego se puedan pedir y visualizar. Lo primero se crea una estructura especial para almacenar la telemetría y se crea un mapping que es una asociación valor-llave. En el caso de esta aplicación la llave es el identificador del dispositivo y el valor el *array* de telemetría. En esta función se devuelve un parámetro 'posicion' que indica el lugar dentro del array que ocupa la estructura que se acaba de rellenar. este parametro es importante por si más adelante se quieren chequear estos parámetros.

Una vez esto está preparado, se trabaja la función IngestTelemety (desde la cual entran los valores del id del dispositivo, y los tres datos de telemetría). Esta función es donde se van a settear los valores. En ella se asocia el id a la llave, se rellena el *array* con la información de telemetría que entra en la función y se asocia dicha estructura con el identificador, haciendo el mappeo llave-valor.

Por último comentar la función getData donde se pueden recuperar y ver los valores previamente almacenados. Para poder determinar qué estructura del *array* se quiere visualizar es necesario indicar de qué dispositivo queremos la información y que posición dentro del *array* tiene la información que se quiere ver. Esta posición es la que se guarda en la función IngestTelemetry. Una vez se pasan estos dos parámetros se puede entrar en el array y obtener la información de humedad, temperatura y sello temporal correspondientes. El código completo se encuentra en la Section C.4.

4

Validación y Análisis de Resultados

En este capítulo se va a exponer lo obtenido con el desarrollo del capítulo anterior, las dificultades encontradas y las posibles soluciones a dichos problemas

4.1. Validación del desarrollo

Comenzando desde el *hardware*, en la Figure 4.1 se puede ver cómo los mensajes se reciben correctamente en SigFox. Esto verifica que la placa funciona correctamente, está bien alimentada, y que el cableado del sensor se ha hecho correctamente.

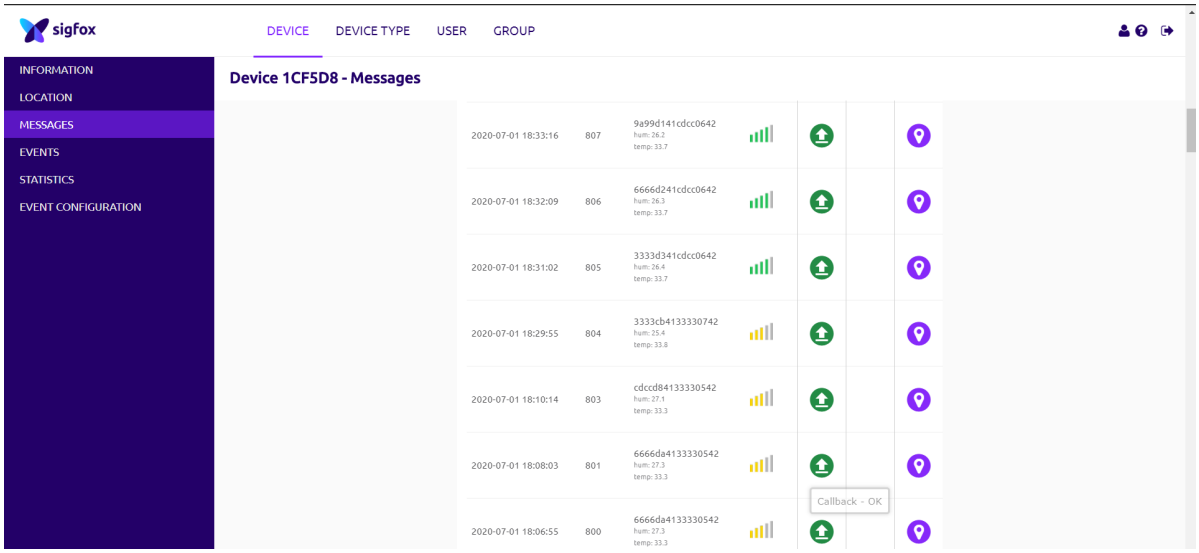


Figure 4.1. Mensajes recibidos en SigFox.

En la Figure 4.2 se puede ver con mayor claridad el detalle de un mensaje. En el se pueden ver las variables asociadas a cada mensaje: el sello temporal (momento en que es recibido el mensaje), los datos de temperatura y humedad enviados, la calidad de la señal, el estado del *callback* y la ubicación desde la cual se ha enviado el mensaje.

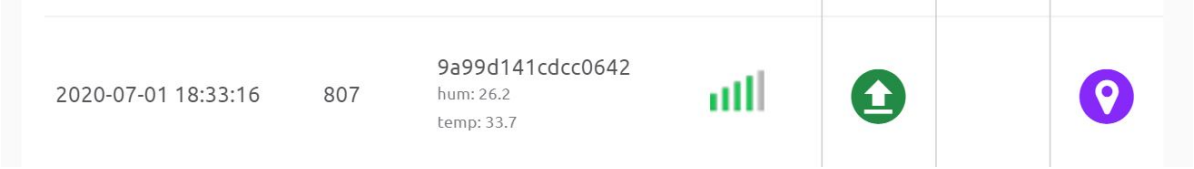


Figure 4.2. Detalle de un mensaje recibido en SigFox.

Una vez configurado el siguiente paso (la conexión con IoT Hub), se vuelven a mandar datos. Dichos datos enviados desde la placa, ahora llegan a SigFox, pero también son redirigidos a Azure IoT Hub. En la Figure 4.3 se ve como la conexión es correcta ya que aparece el sensor listado dentro de IohHub-TFM.

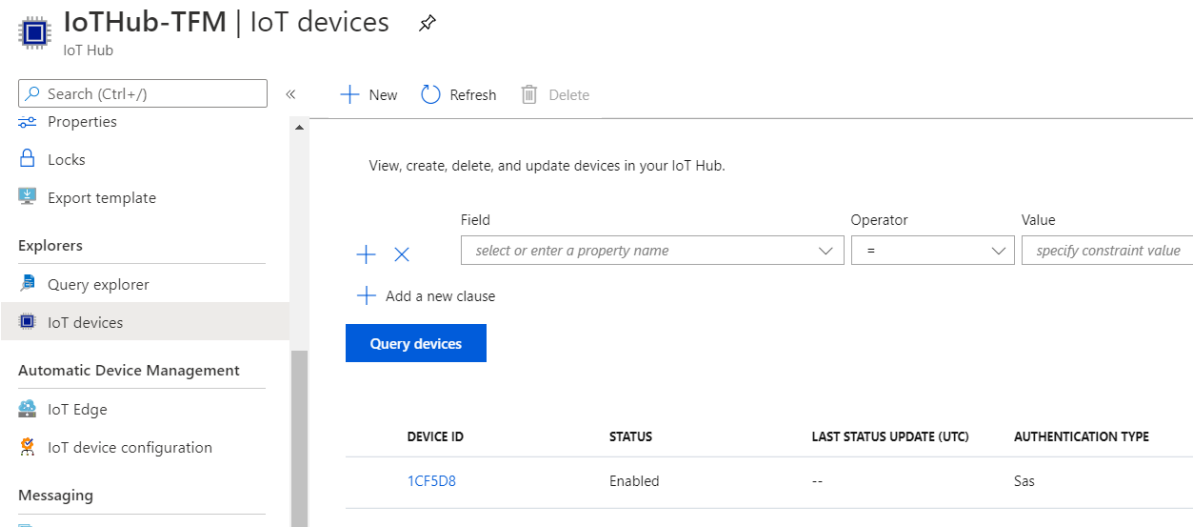


Figure 4.3. Confirmación de que el sensor se recibe en IotHub-TFM.

Del IoT Hub los mensajes pasan a una cola desde la cual toma los datos la *Logic App*. A continuación, en la figura Figure 4.4 se muestra cómo el flujo creado en el servicio *Logic App* funciona correctamente, se muestra como cada vez que se reciben mensajes el proceso ha sido exitoso.

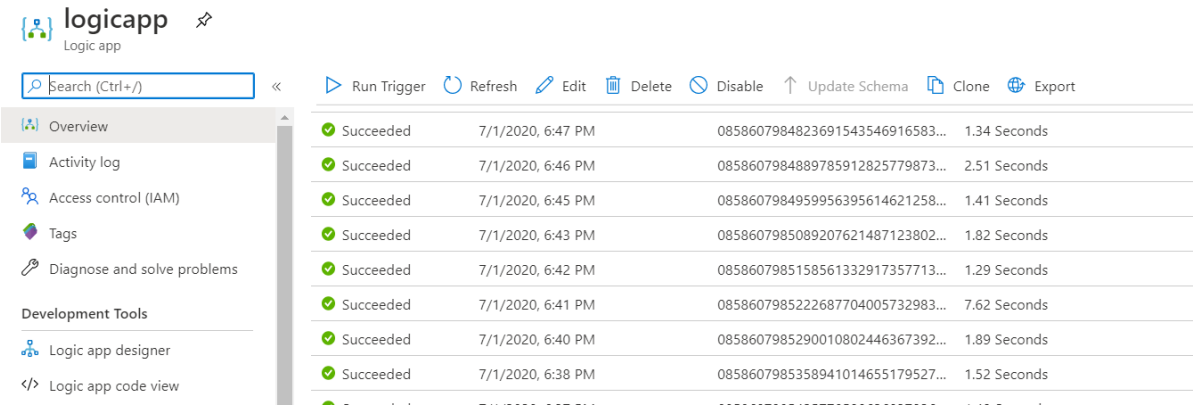


Figure 4.4. Histórico de los flujos de trabajo de *Logic App*.

Entrando en el detalle de cualquiera de los flujos mostrados en la Figure 4.4 se visualiza como cada bloque del flujo tiene una señal de éxito en la esquina superior derecha. En cada uno de ellos se puede pinchar y ver los detalles del proceso. Son de especial interés el primer módulo del flujo, en el que se toman los mensajes de la cola, el módulo que obtiene en función del

identificador del dispositivo la información del contrato y el último módulo que ingresa los datos en la cola de la aplicación *blockchain*. En el detalle del flujo de *Logic App* de la Figure 4.5 se pueden ver los tres módulos indicados y como todos ellos se han realizado exitosamente.

Figure 4.5. Detalle de un flujo de trabajo.

El primer módulo de la *Logic App* se muestra en la Figure 4.6. En él se puede ver como se toma correctamente el mensaje de la cola *myqueue*. Este paso confirma que IotHub manda correctamente los datos a la cola y que el enlace entre la cola y este servicio de flujos de trabajo funciona correctamente también.

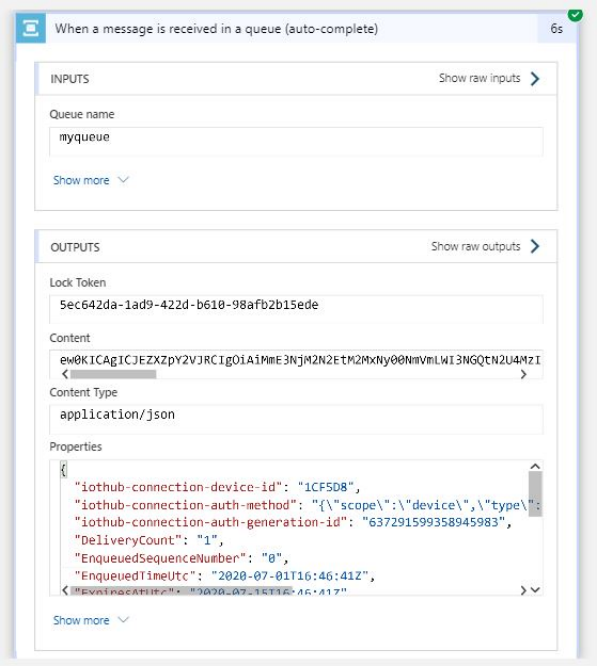


Figure 4.6. Obtención de los mensajes de la cola *myqueue*.

El siguiente módulo que es interesante visualizar corresponde al módulo que ejecuta el procedimiento que se añadió a la base de datos. Este procedimiento obtiene en función del id del dispositivo la información del contrato activo. Se incluye aquí la Figure 4.7 del módulo correspondiente, y como no se ve el código completo se añade a continuación. Es interesante verlo el código obtenido ya que muestra la información que saca de *blockchain*.

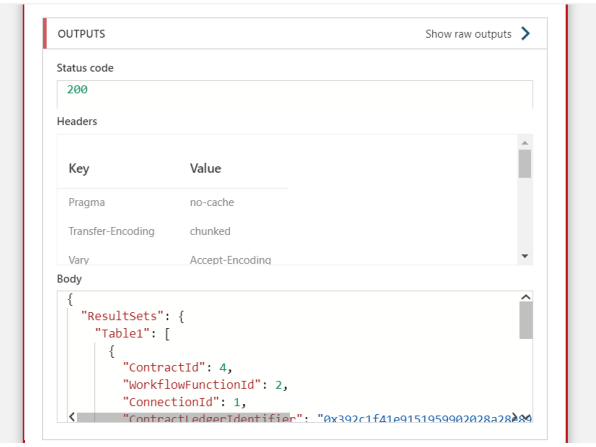


Figure 4.7. Obtención de la información del *smart contract*.


```
{
  "ResultSets": {
    "Table1": [
      {
        "ContractId": 4,
        "WorkflowFunctionId": 2,
        "ConnectionId": 1,
        "ContractLedgerIdentifier": "
           0x392c1f41e9151959902028a28e89820856e7d509",
        "ContractCodeBlobStorageUrl": "https://
           cjzdectfm.blob.core.windows.net/contract-code/7f666bbd-
           b7ac-4bea-b130-7c2643cf6039",
        "UserChainIdentifier": "
           0xfc1f3338be98a3f38c95c0c907160084da005f0f",
        "WorkflowFunctionName": "IngestTelemetry",
        "WorkflowName": "RefrigeratedTransportation",
        "IngestTelemetry_ContractWorkflowFunctionID": 2,
        "IngestTelemetry_ContractPersonaID": null,
        "IngestTelemetry_Humidity_WorkflowFunctionParameterID": 1,
        "IngestTelemetry_Temperature_WorkflowFunctionParameterID": 2
           ,
        "IngestTelemetry_Timestamp_WorkflowFunctionParameterID": 3
      }
    ]
  },
  "OutputParameters": {}
}
```


Por último se va a exponer el detalle del último módulo que prepara el mensaje y lo manda a la cola de ingreso de *blockchain*. Este módulo al igual que el resto funciona correctamente,

el envío parece que se hace bien y si se miran las métricas de la cola, se ve cómo se reciben mensajes.

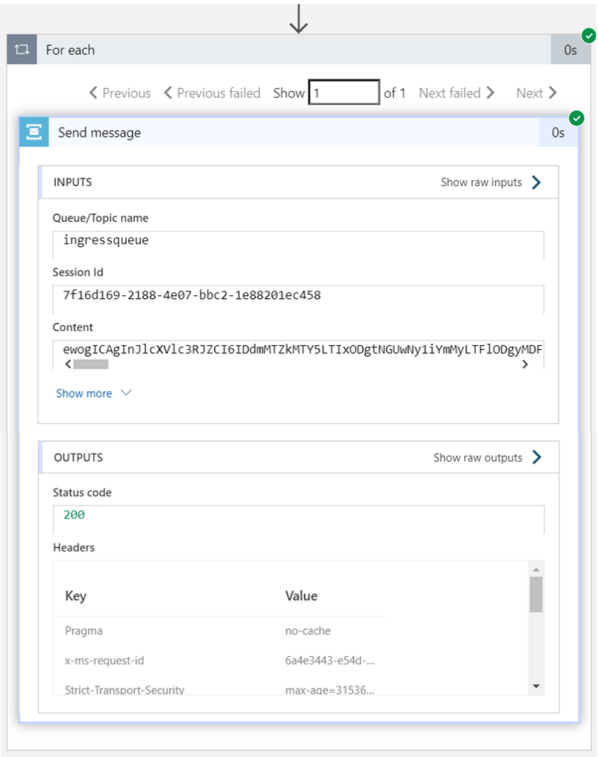


Figure 4.8. Envío de la información del mensaje preparado a *blockchain*.

En la cola de salida, se puede ver como entran mensajes pero no salen, en el momento en el que mejor funcionó fue a las 3:41am del 1 de julio que tras enviar datos a la cola se ve cómo hay un amago de envío, pero por lo general los mensajes mandados no parece que salgan de esta cola. En la Figure 4.9 se puede visualizar a la derecha el envío correcto de mensajes a las 3:41am y en la derecha una gráfica que muestra el Service Bus de *blockchain* con los mensajes de entrada en azul y los de salida en naranja. A las 3:41am se puede ver que la cola de salida emite 7 mensajes.

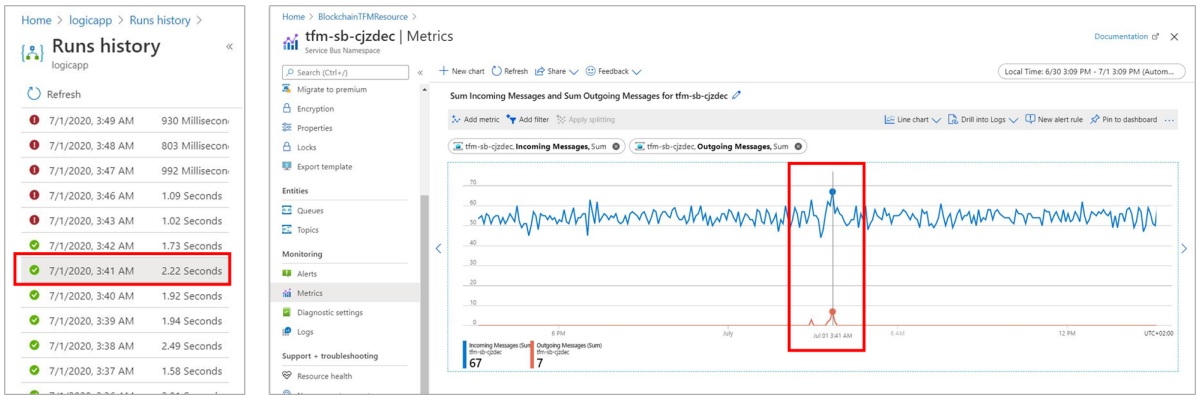


Figure 4.9. Paso de mensajes a través de la cola de entrada a *Blockchain*.

Respecto al *smart contract* se ha conseguido el objetivo deseado que era guardar los datos de telemetría de tal manera que estén asociados a cada contenedor. Al principio el contrato se podía probar en la aplicación web de Azure *Blockchain Workbench* pero se ha acabado en Remix debido a que se acabó la subscripción y se quería comprobar que el funcionamiento del contrato fuese correcto.

4.2. Problemas encontrados

En este trabajo se han encontrado algunos problemas a la hora de desarrollar la aplicación. Por un lado ha sido problemática la cuenta de Azure con la que trabajar, esto ha sido un problema más operativo ya que sin una cuenta no se podía continuar con el desarrollo. Por otro lado se han encontrado algunos problemas funcionales, ya que tras desplegar la aplicación completa y a lo largo del proceso, en puntos concretos no se conseguía el objetivo deseado.

Respecto al primer problema con la cuenta de Azure, en un principio se iba a trabajar con la cuenta educativa de la universidad, ya que todo el sistema de Comillas funciona con Microsoft. Además ofrecen un crédito de 100€ gratuitos a los alumnos. Pero al comenzar a desplegar distintos módulos de Azure, se comprobó que hay ciertas funciones que con este crédito no se pueden desplegar. El servicio problemático era Azure *Blockchain Workbench*. Al ser un servicio que despliega tantos módulos individuales Azure requiere que haya una cuenta con una subscripción más grande que una cuenta gratuita de estudiante con 100€ de crédito.

Al encontrar tal situación se paró el desarrollo hasta encontrar una solución y más adelante ya se encontró la manera de desplegar correctamente todos los módulos de Azure que requiere el desarrollo.

Por otra parte, los principales problemas funcional confrontados también han tenido lugar en la plataforma de Azure. Durante el desarrollo de la aplicación ha habido momentos de atasco en los que durante unos días no se avanzaba, la mayoría de ellos tras unos días de reflexión e investigación de soluciones similares, se ha podido continuar pero no en todos los casos.

Un momento problemático se dio durante el despliegue de Azure *Logic App*. En el momento de implementar el bloque que conectaba con la base de datos de Azure *Blockchain Workbench*, para sacar la información del contrato a partir del identificador del dispositivo, no se realizaba la conexión correctamente. Esto ocurría por que no estaba conectada la base de datos de la parte *blockchain* con el punto de acceso local desde el ordenador que se estaba trabajando. Faltaba descargar la aplicación que permitía establecer un *Gateway*.

Por último, el problema más importante se dio cuando tras conseguir desplegar toda la aplicación, siguiendo los diferentes documentos de apoyo ([11], [43], [42], [46], ...) no se consigue envíar los datos a Azure. Esto se intuye en la sección anterior, los mensajes no llegan a la aplicación de *blockchain*. Parece que entran en la cola pero no salen. La raíz de este problema no está clara, el problema puede encontrarse en la conexión del dispositivo con la cuenta que se crea para *blockchain*. Se puede estar dando el caso de que no se esté asociando correctamente el id del dispositivo con el id del email y por tanto los mensajes no saben dónde deben de ser publicados.

Otra opción valorada es que alguno de los pasos seguidos no se haya hecho correctamente, aunque se ha repasado concienzudamente todo el proceso explicado en los repositorios. Puede ser que no esté detallado explícitamente algún paso y que falte un paso que sea el que permita que todo funcione correctamente.

Debido a que la subscripción a Azure se acabó antes de finalizar el trabajo, para la parte del *smart contract* se tuvo que buscar una manera alternativa para probar el funcionamiento correcto.

4.3. Posibles soluciones

Las soluciones a los problemas anteriores se exponen en este apartado.

Respecto a la cuenta de Azure, tras averiguar que la subscripción de crédito por ser estudiante de Comillas no era suficiente para desplegar todos los servicios requeridos, se pidió ayuda a los directores. Estos se pusieron en contacto con una persona que trabaja en Azure, Sergio González, que pudo dar mayor visibilidad del problema y proponer una solución. Se propuso cambiar la subscripción de la cuenta de estudiante a Pago por Uso, muchos de los servicios de Azure son gratuitos o de coste muy bajo. Además esta nueva subscripción de Azure cobra según se usan los recursos no tiene una cuota fija, y por tanto era una solución buena.

Al cabo de unos días usando la cuenta de la universidad con esta nueva subscripción, se vio que el gasto que iba en aumento rápidamente, el servicio de Azure *Blockchain Workbench* si que supone un gasto mayor. De manera que cuando la tarifa ascendía a 62,23€ se para la subscripción y se busca una nueva solución (la tarifa está adjunta tras los anexos). Investigando más se observa que si se da de alta en Azure una cuenta que no ha hecho uso este servicio anteriormente, se ofrece un mes de prueba gratuita con un crédito de 170€. De esta manera, utilizando un correo de hotmail (correo de Microsoft) personal se procede a desplegar otra vez todos los módulos del desarrollo.

Respecto a los problemas de funcionamiento de los módulos, el primer obstáculo se supero al cabo de unos días. El problema era la conexión desde *Logic App* a la base de datos SQL, pero al ser un problema concreto fue sencillo buscar la solución. Otras personas que han tenido problemas similares han publicado sus dudas sobre el repositorio utilizado, dejando la solución a la vista del resto de personas.

Sin embargo, enfrentarse al siguiente problema no ha sido tan sencillo. Completar el desarrollo y ver que el funcionamiento no es el deseado ha sido frustrante ya que no se encontrada la raíz del problema. Se ha investigado muchísimo los repositorios y las preguntas y dudas publicadas sobre los mismos para ver si alguna de esas publicaciones aportaban algo de lucidez sobre lo que podía estar ocurriendo en la aplicación del trabajo. El problema radica en que para buscar la solución es necesario conocer específicamente donde está el fallo. Al no saber qué punto exacto es el que está generando los problemas es muy difícil buscar la solución. En el caso de que se hubiera encontrado qué causaba el fallo de envío de mensajes se habría buscado la soluciona través de uno de estos métodos:

- Utilizar el soporte técnico de Azure, preguntar a los desarrolladores de Azure cómo resolver el problema concreto.
- Publicar la duda específica en una comunidad de desarrolladores. Esto se podría hacer sobre el propio GitHub seguido.

Por último para poder trabajar el *smart contract* se ha buscado una plataforma alternativa a la aplicación web de Azure *Blockchian Workbench*, por recomendación de los directores se utiliza Remix una plataforma en la que se pueden probar contratos de manera rápida. Gracias a esta plataforma se puede continuar probando el contrato hasta comprobar que el funcionamiento es el deseado.

5

Conclusiones y trabajos futuros

En este capítulo se revisa si se cumplen los objetivos propuestos al comienzo del trabajo, se escriben las conclusiones y se establecen los pasos futuros.

5.1. Conclusiones del trabajo

El objetivo de este trabajo era llevar a cabo una aplicación que monitorice la cadena de frío de transporte refrigerado para asegurar su calidad. Este objetivo se ha cumplido. La aplicación se ha desarrollado de manera satisfactoria.

Comprobando las diferentes metas para llegar al objetivo principal, se tiene el estudio de las tecnologías IoT y *blockchain*. Se estudian las tecnologías y protocolos involucrados en el desarrollo de la aplicación en profundidad, especialmente enfocado a su aplicación en la parte logística de la cadena de suministro de productos que requieran cadena de frío. Los bloques analizados son: las plataformas *hardware*, las tecnologías y protocolos de comunicación, la plataformas en la nube y las principales plataformas *blockchain*.

Antes del desarrollo, se incluye una última sección (Section 2.5) que trata casos de uso de la combinación de estas dos tecnologías en otros sectores. Estos casos permiten tener una mayor visión de los beneficios que puede acarrear esta combinación de tecnologías.

Este extenso estudio de las posibilidades que hay para cada una de las capas de comunicación acompañado de los casos de uso, demuestra cómo hay una gran cantidad de opciones para desarrollar una aplicación que combine estas dos tecnologías: IoT y *blockchain*. A pesar de que no son tecnologías maduras cada vez están más presentes en las empresas, y con foco en el mundo logístico pueden suponer un cambio en el modelo de negocio importante: por un lado *blockchain* permite eliminar intermediarios, y por otro IoT aporta en las operaciones y movimientos visibilidad y trazabilidad, quedando registradas automáticamente en la cadena de bloques. Además en el caso desarrollado en este trabajo, las diferentes partes involucradas en la cadena de suministro pueden visualizar lo que está ocurriendo en tiempo real, eliminando complejidad y fomentando la buena praxis.

Respecto al desarrollo de la aplicación, se demuestra que es posible combinar una placa de Arduino, la nube de SigFox, y diferentes módulos de Azure hasta llegar a la plataforma de *blockchain* fructíferamente. Debido a que Arduino es una plataforma con la que se ha trabajado más a lo largo de la carrera y máster, fue rápido configurarlo. SigFox y Azure han supuesto el mayor reto a la hora de desarrollar, pero con ayuda de los directores y siguiendo las instrucciones de diferentes repositorios se ha llegado muy lejos.

En el momento de realizar la conexión con SigFox, me he podido apoyar en una práctica del curso de IoT de este año, donde se hacía una práctica similar, donde también se mandaban datos de Arduino a Azure a través de SigFox.

En Azure se utilizan módulos diferentes a los de la práctica del curso ya que se deben de preparar los datos para que estos puedan ser reconocidos por la aplicación *blockchain*. Destaco lo que he aprendido respecto a parsear datos, me parece muy interesante lo facilidad que da Azure *Logic App* para realizar estos cambios.

En resumen, de toda la aplicación, se ha conseguido la conexión de todas las plataformas y módulos, lo único que no se ha conseguido es recibir los datos del sensor en *blockchain*, el último enlace entre Azure *Logic App*, la cola de entrada de *blockchain* y la publicación de los datos ha dado algún problema. Se ha profundizado el conocimiento de Solidity y de las plataformas que permiten desplegar contratos inteligentes.

Por otra parte, hacer un TFM de desarrollo ha supuesto un reto personal, ya que en varias ocasiones me he encontrado frente a un problema que no sabía resolver pero con dedicación y esfuerzo, se encontraba el problema y se conseguía resolver y avanzar con el desarrollo. Respecto al problema del último punto donde no llegan a ingerirse los datos de la cola de *blockchain* y entrar en el *smart contract*, no se ha encontrado la solución por que no está claro dónde está el problema, ya que todo parece que se ha desplegado satisfactoriamente.

Personalmente, haber realizado este trabajo me ha permitido tener un conocimiento más profundo de los protocolos y tecnologías que rodean a Internet de las cosas y a *blockchain*. Me ha permitido mejorar mis conocimientos de Python y sobretodo he tenido la oportunidad de aprender Solidity, el lenguaje de los contratos inteligente. Finalmente comentar que he aprendido a no rendirme ante un problema que no entiendo y he mejorado mis capacidades de investigación.

5.2. Trabajos futuros

Si se decidiera continuar trabajando en este proyecto, los pasos a seguir están bien definidos. Lo principal es encontrar qué está ocurriendo en el desarrollo que impide que la aplicación funcione correctamente. Si se localizase dicho error se podría completar el desarrollo.

Una vez conseguido el funcionamiento correcto sería interesante incluir la ubicación del dispositivo que se envían los datos. Como se ha explicado anteriormente, no es posible hacerlo todo en el mismo *callback* de SigFox pero si que podría hacerse desde otro separado, y en Azure unir la información obtenida de ambos mensajes (los datos y la ubicación).

Otra mejora de la que se beneficiaría mucho la aplicación es una mejora en el contrato inteligente, se debería añadir al histórico de los datos no solo de qué camión provienen si no quien está conduciendo este camión en ese momento, por ejemplo para poder avisar directamente a esa persona en caso de salirse de rangos la temperatura o humedad. También es

interesante añadir la persona que lleva la mercancía como métrica que pueda ser estudiada por la empresa que contrata el servicio.

Una vez que la aplicación funciona con estas mejoras, realmente podría llegar a implementarse en el contenedor de un camión, ya que hasta el momento todo se hace con un sensor. Sería muy interesante comprobar el funcionamiento de la aplicación en movimiento. Para llegar al punto de colocar el sensor en un camión habría que preparar un encapsulado para el conjunto sensor mas placa que fuese fácil de colocar en la parte trasera de los vehículos.

Tras realizar la prueba con un único sensor, el objetivo siguiente sería organizar un estudio económico de la solución para escalar el proyecto y desplegarlo en una flota de camiones. Con el estudio preparado se podría llegar a un acuerdo o vender la aplicación a una empresa, y comercializar la solución desarrollada en el trabajo.

A

Relación con los Objetivos de Desarrollo Sostenible

En este capítulo se trata la relación y alineamiento del trabajo con los objetivos de desarrollo sostenible propuestos el 25 de septiembre de 2015 [24]. De los 17 objetivos establecidos, se tratan los que tienen más relevancia para el trabajo.

A.1. Alineación del proyecto con los ODS

De los 17 objetivos propuestos, que se encuentran en la Figure A.1, los que más se alinean con el trabajo son los que se tratan a continuación, además de nombrarlos se explica por qué están relacionados con el trabajo y cómo cumplen con el objetivo o prentenden cumplirlo:

Figure A.1. Objetivos de desarrollo sostenible.

- 3. Salud y bienestar: Con esta aplicación se desarrolla una monitorización de productos refrigerados, la mercancía transportada no siempre tienen por que ser alimentos, pero si esta aplicación se utiliza para bienes de consumo perecederos, puede garantizar el buen estado de los alimentos y por tanto la salud y bienestar de los compradores finales. La aplicación desarrollada aporta certeza del buen estado de los productos.
- 7. Energía asequible y no contaminante: La aplicación se ha desarrollado para utilizar redes LPWAN, es decir redes de bajo consumo que permiten a los dispositivos "dormir" siempre que no tengan que mandar datos.
- 8 Trabajo decente y crecimiento económico: Con esta aplicación se aporta una mayor transparencia ya no solo del producto que circula si no de las personas por las que pasa dicho producto, esto permite por ejemplo ver las horas que trabaja una persona, demostrando horarios decentes. Por otra parte el uso de estas tecnologías emergentes permiten un crecimiento de las empresas mejorando así su economía.
- 9. Industria, innovación e infraestructura: El proyecto ilustra la combinación de dos nuevas tendencias de la industria, y aunque hay proyectos similares en el mercado sigue siendo inovador, ademas con el proyecto se crea una nueva infraestructura para el sector logístico.
- 10. Reducción de las desigualdades: Al igual que para el punto ocho, se pueden demostrar horarios de trabajo decentes, también se pueden obtener estadísticas del porcentaje de personas de ambos sexos, discapacitados, personas en exclusión social, etc. implicados en el sector del transporte de productos refrigerados, permitiendo a las empresas igualar la balanza y ofrecer nuevas oportunidades de manera equilibrada.
- 12. Producción y consumo responsables: Como se ha mencionado antes, la solución desarrollada busca ser de consumo bajo, pero además se busca que tenga una vida útil larga. Haciendo referencia a las tres R del reciclaje: reducir, reutilizar y reciclar, si se desplegase el proyecto como se explicaba en la Section 5.2, se prepararían los encapsulados necesarios reduciendo el número a los dispositivos esenciales. Se reutilizarían pasándolos de una flota a otra según se contrate el servicio en una empresa y se deje de contratar en otra y se buscaría la manera de reciclar correctamente los dispositivos una vez llegado el fin de su vida útil.

B

Estudio Económico

En este anexo se van a tratar los recursos económicos necesarios para desplegar el caso de uso, esto puede dar una idea general al lector de lo costaría desplegar la aplicación en una flota de camiones completa.

B.1. Estudio económico de la aplicación desarrollada

Se van a analizar los costes de los recursos utilizados a lo largo de todo el proceso, desde el *hardware* hasta Azure *Blockchain Workbench*.

Dentro del *hardware* se encuentra la placa de Arduino y el sensor DHT22.

- Arduino MKR Fox 1200: 35€ en la tienda oficial de Arduino [47]
- DHT22: 19,15€ en Electrónica Embajadores, donde se compró. [5]

A continuación se analiza el coste de SigFox, que por un precio fijo permite acceder a sus redes, su nube y utilizar sus funciones de *callback*. Las tarifas varían en función de si se van a conectar menos o más de 1000 dispositivos, esto se puede ver en la Figure B.5. El Arduino MKR Fox 1200 viene con una subscripción gratis de un año a las redes de SigFox, pero en caso de querer utilizar dicho dispositivo más tiempo, se debe de pagar 16,13€ al año. Al renovar la subscripción se mantienen los mismos servicios que se ofrecían con la subscripción gratis.

El siguiente y último recurso de pago es la plataforma Azure. Aquí se cobran los recursos por uso, así que se van a analizar los precios de los módulos utilizados. Azure tiene una opción que permite ver el coste de los recursos. Sus tarifas son mensuales.

- Iot Hub: Para que la aplicación vaya más holgada se cogería la tarifa S1 (Standard 1), con un precio de 21,083€. Esta tarifa permite 400.000 mensajes al día y un tamaño de los mensajes de hasta 4KB. Se pueden ver cómo escalan las tarifas en función de cuántos mensajes se quieran mandar.
- Service Bus: Se cobra por los mensajes que pasan por el servicio. en la Figure B.3 se pueden verlos precios de los envíos de mensajes de las tarifas básica y estándar. Al igual

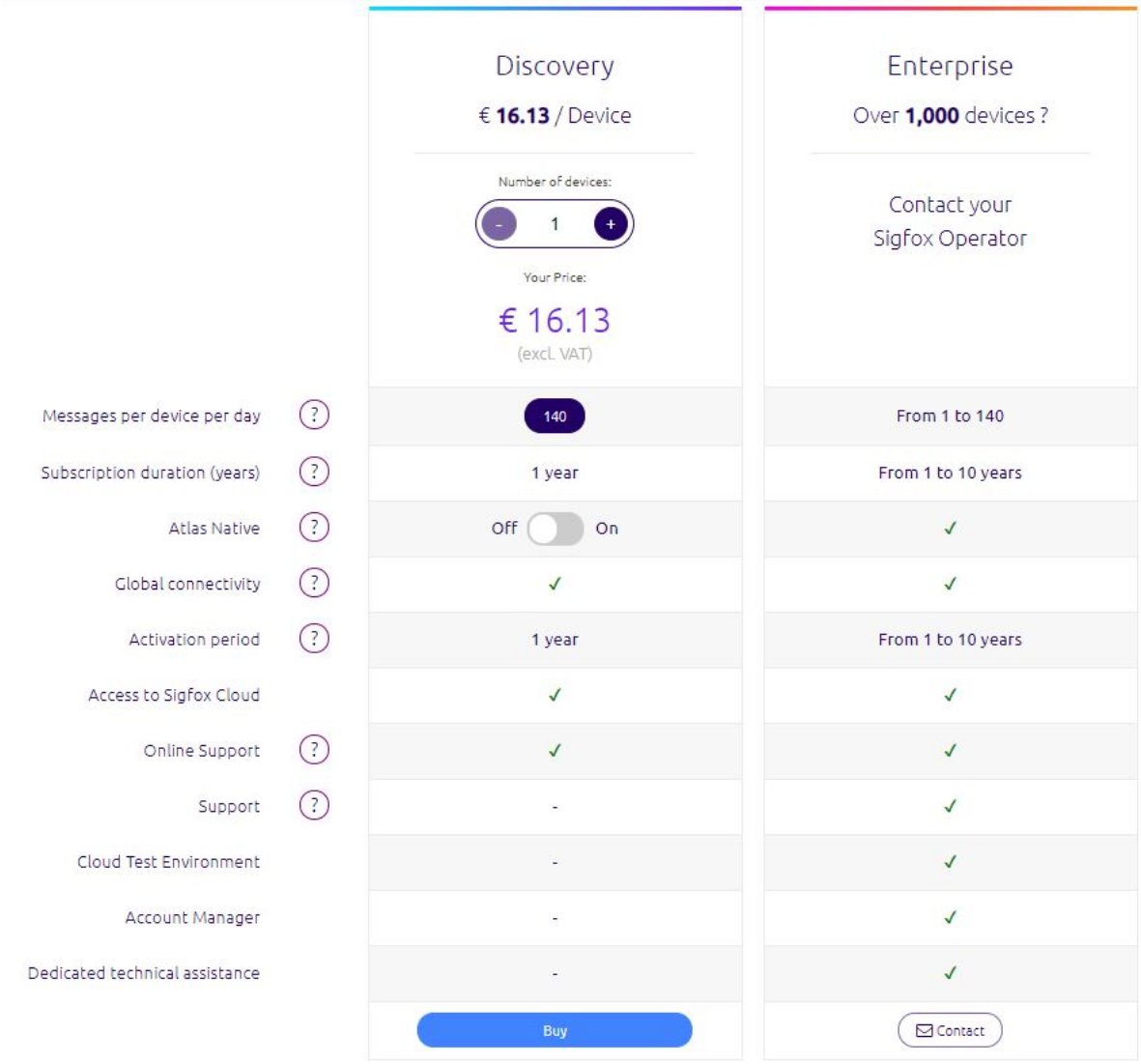
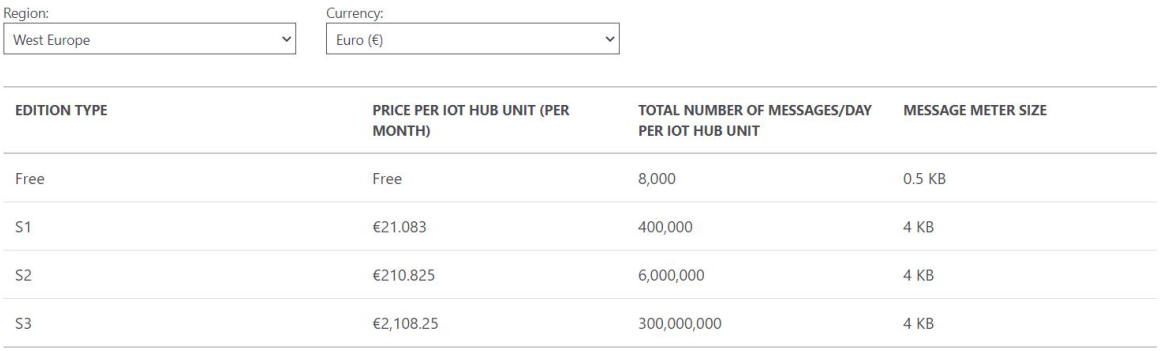


Figure B.1. Tarifas anuales de SigFox.

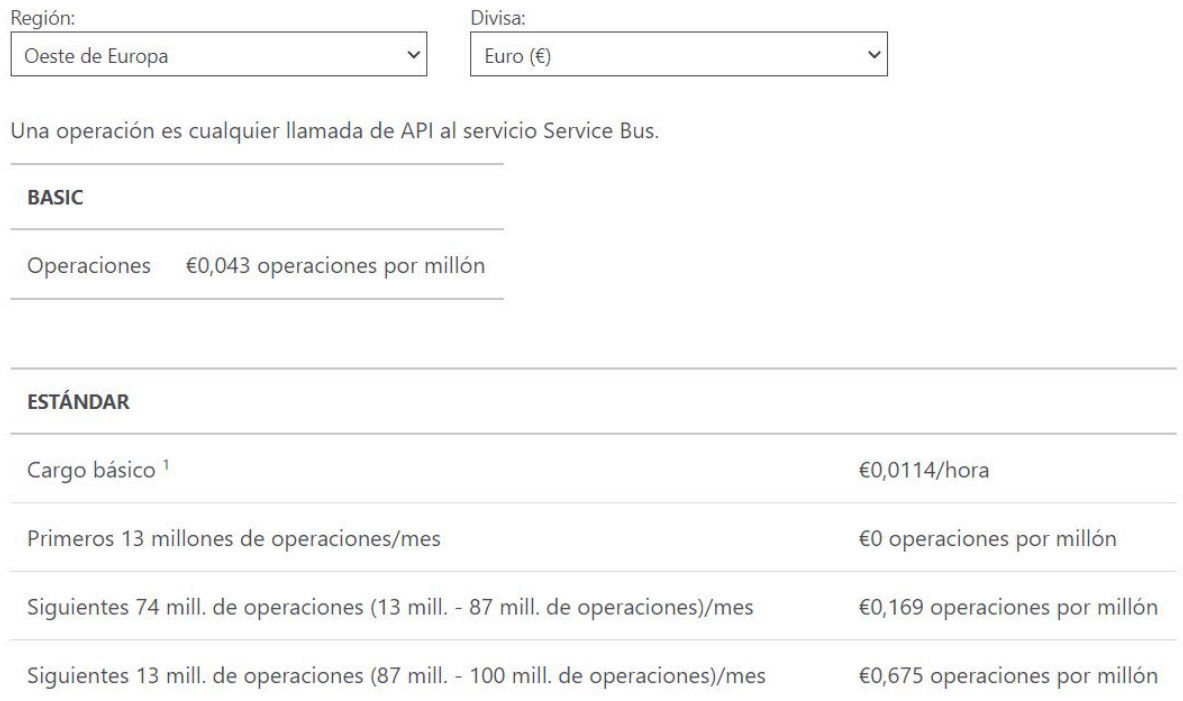


EDITION TYPE	PRICE PER IOT HUB UNIT (PER MONTH)	TOTAL NUMBER OF MESSAGES/DAY PER IOT HUB UNIT	MESSAGE METER SIZE
Free	Free	8,000	0.5 KB
S1	€21.083	400,000	4 KB
S2	€210.825	6,000,000	4 KB
S3	€2,108.25	300,000,000	4 KB

Figure B.2. Tarifas mensuales de IoTHub.

que para IoT se escoge la tarifa estándar, donde se cobra 0,01114€ por hora que está activo el servicio y luego las tarifas en función de la cantidad de mensajes enviados al mes. Hasta hacer 13 millones de operaciones el servicio es gratuito.

- Logic App: Cobra cada vez que ejecuta un flujo de trabajo, y también por la conexión. Interesa revisar por tanto los dos primeros costes de la ,Figure B.4.



Región: Oeste de Europa

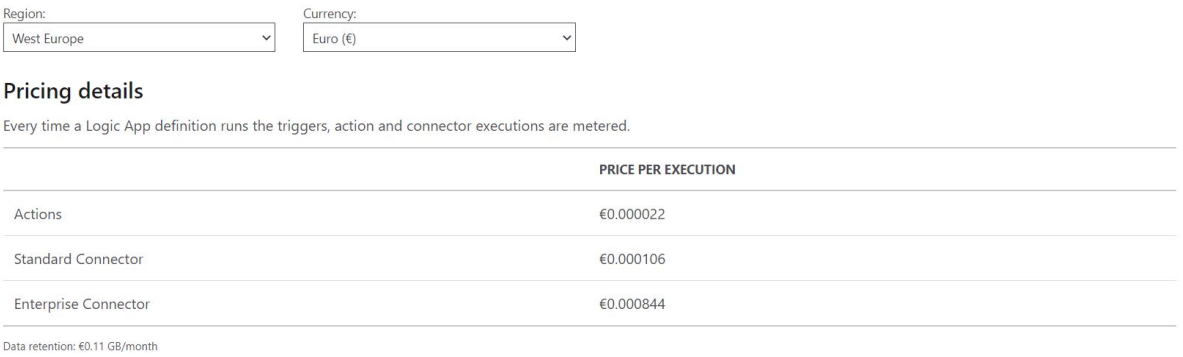
Divisa: Euro (€)

Una operación es cualquier llamada de API al servicio Service Bus.

BASIC	
Operaciones	€0,043 operaciones por millón

ESTÁNDAR	
Cargo básico [1]	€0,0114/hora
Primeros 13 millones de operaciones/mes	€0 operaciones por millón
Siguientes 74 mill. de operaciones (13 mill. - 87 mill. de operaciones)/mes	€0,169 operaciones por millón
Siguientes 13 mill. de operaciones (87 mill. - 100 mill. de operaciones)/mes	€0,675 operaciones por millón

Figure B.3. Tarifas de Service Bus.



Region: West Europe

Currency: Euro (€)

Pricing details

Every time a Logic App definition runs the triggers, action and connector executions are metered.

	PRICE PER EXECUTION
Actions	€0.000022
Standard Connector	€0.000106
Enterprise Connector	€0.000844

Data retention: €0.11 GB/month

Figure B.4. Tarifas por acción y conexión de Logic App.

- Azure Blockchain Workbench: Desplegar este servicio implica desplegar 25 módulos, de los cuales hay algunos que si que suponen un coste considerable. Se estiman tarifas entre los 300€ y 500€ al mes. Los servicios que más cuestan son las máquinas virtuales y la aplicación web desplegada.

Como resumen cada dispositivo cuesta un fijo de 55€, el primer año de SigFox es gratuito y a partir de ese momento 16€ anuales. Respecto a Azure, la cuenta mensual asciende a alrededor de 400€ mensuales. Esto puede parecer mucho pero con desplegar un único servicio de Azure *Blockchain Workbench* se pueden controlar un gran número de dispositivos. Además si se tuviese que desplegar esta aplicación seguramente al comprar al por mayor cierta cantidad de *hardware* se obtengan precios más económicos.

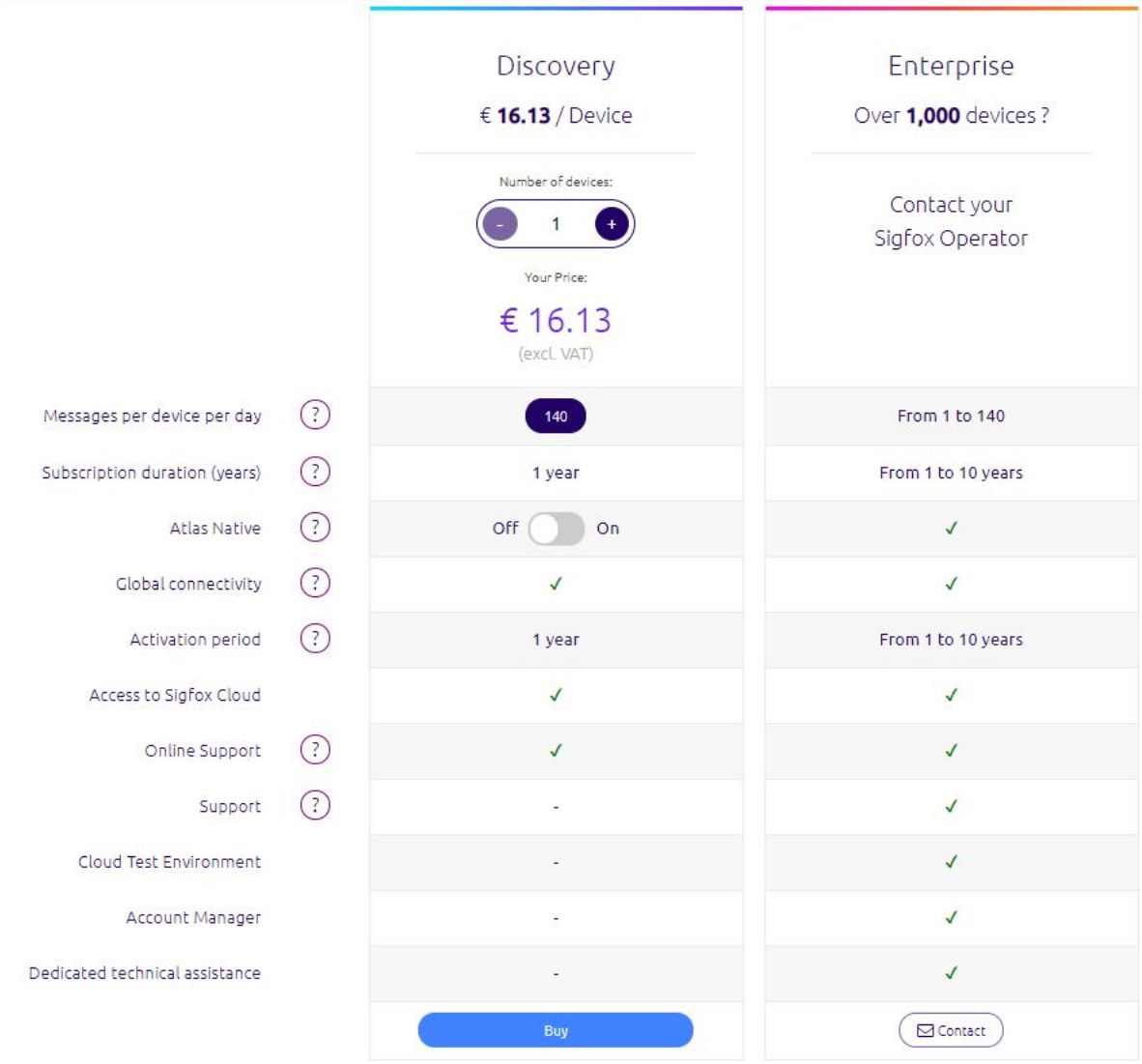


Figure B.5. Tarifas anuales de SigFox.

C

Códigos

En este anexo se encuentran los códigos utilizados a lo largo del proyecto. El código de Arduino subido a la placa y los dos archivos que componen el contrato inteligente de la aplicación *blockchain* desarrollada.

C.1. Código Arduino

```
1   //Libraries
2   #include <Adafruit_Sensor.h>
3   #include <DHT.h>
4   #include <SigFox.h>
5   #include <Arduino_JSON.h>
6
7   //Constants
8   #define DHTPIN 5            // Pin al que se conecta la salida
        digital del sensor
9   #define DHTTYPE DHT22       // Define el sensor DHT22  (AM2302)
10  DHT dht(DHTPIN, DHTTYPE);   // Inicializa el sensor DHT
11
12  float hum;                  // Variable que guarda la humedad
13  float temp;                 // Variable que guarda la temperatura
14
15  void setup() {
16    Serial.begin(9600);
17    dht.begin();
18
19    if (!SigFox.begin()) {    // Comienza la conexion con la red de
          SigFox
20      Serial.println("Unable to init the Atmel ATA8520 Sigfox
          chipset");
```

```
21          return;                 // Mensaje de error si no es capaz de
                inicializar el chip de SigFox
22        }
23      }
24
25      void loop()
26      {
27        SigFox.debug();
28        hum = dht.readHumidity();// Lee la humedad que mide el sensor
29        temp= dht.readTemperature(); // Lee la temperatura que mide el
              sensor
30        SigFox.beginPacket();
31        SigFox.write(hum);        // Mensaje que se manda a SigFox
32        SigFox.write(temp);
33        SigFox.endPacket();
34        Serial.print("HUM: ");    // Mensaje enviado al Monitor Serie,
             solo para verificar
35        Serial.print(hum);
36        Serial.print(" TEMP: ");
37        Serial.print(temp);
38        Serial.print('\n');
39        delay(60000);             // Se pide que espere 1 minuto antes
             de volver a leer y mandar datos
40      }
```

C.2. Código de SigFox parseado

El resultado del código JSON que se manda desde SigFox parseado es el siguiente:

```
1  {
2    "type": "object",
3    "properties": {
4      "DeviceID": {
5        "type": "string"
6      },
7      "data": {
8        "type": "string"
9      },
10     "deviceTypeId": {
11       "type": "string"
12     },
13     "humidity": {
14       "type": "string"
15     },
16     "seqNumber": {
17       "type": "string"
18     },
19     "temperature": {
20       "type": "string"
```

```
    },
    "time": {
      "type": "string"
    }
  }
}
```

C.3. Código de entrada a *blockchain*

Código de JSON parseado para prepara los datos a ser introducidos en *blockchain*.

```
{
  "type": "object",
  "properties": {
    "Table1": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "ConnectionId": {
            "type": "number"
          },
          "ContractCodeBlobStorageUrl": {
            "type": "string"
          },
          "ContractId": {
            "type": "number"
          },
          "ContractLedgerIdentifier": {
            "type": "string"
          },
          "IngestTelemetry_ContractPersonaID": {},
          "IngestTelemetry_ContractWorkflowFunctionID": {
            "type": "number"
          },
          "IngestTelemetry_Humidity_WorkflowFunctionParameterID": {
            "type": "number"
          },
          "IngestTelemetry_Temperature_WorkflowFunctionParameterID":
              {
            "type": "number"
          },
          "IngestTelemetry_Timestamp_WorkflowFunctionParameterID": {
            "type": "number"
          },
          "UserChainIdentifier": {
            "type": "string"
          },
```

```
            "WorkflowFunctionId": {
              "type": "number"
            },
            "WorkflowFunctionName": {
              "type": "string"
            },
            "WorkflowName": {
              "type": "string"
            }
          },
          "required": [
            "ContractId",
            "WorkflowFunctionId",
            "ConnectionId",
            "ContractLedgerIdentifier",
            "ContractCodeBlobStorageUrl",
            "UserChainIdentifier",
            "WorkflowFunctionName",
            "WorkflowName",
            "IngestTelemetry_ContractWorkflowFunctionID",
            "IngestTelemetry_ContractPersonaID",
            "IngestTelemetry_Humidity_WorkflowFunctionParameterID",
            "IngestTelemetry_Temperature_WorkflowFunctionParameterID",
            "IngestTelemetry_Timestamp_WorkflowFunctionParameterID"
          ]
        }
      }
    }
}
```

C.4. RefrigeratedTransport.sol

```
pragma solidity >=0.4.25 <0.6.0;
//pragma experimental ABIEncoderV2;

contract RefrigeratedTransportation
{
    //Set of States
    enum StateType { Created, InTransit, Completed, OutOfCompliance}
    enum SensorType { None, Humidity, Temperature }

    //List of properties
    StateType public  State;
    address public  Owner;
    address public  InitiatingCounterparty;
    address public  Counterparty;
    address public  PreviousCounterparty;
    address public  Device;
    address public  SupplyChainOwner;
    address public  SupplyChainObserver;
    uint256 public  MinHumidity;
```

```
20      uint256 public  MaxHumidity;
21      uint256 public  MinTemperature;
22      uint256 public  MaxTemperature;
23      SensorType public  ComplianceSensorType;
24      uint256 public  ComplianceSensorReading;
25      bool public  ComplianceStatus;
26      string public  ComplianceDetail;
27      uint256 public  LastSensorUpdateTimestamp;
28
29      address[] dispositivos;
30
31      struct Contenedor{
32          uint256 humidity;
33          uint256 temperature;
34          uint256 timestamp;
35      }
36
37      mapping(address => Contenedor[]) datos;
38
39      constructor(address supplyChainOwner, address supplyChainObserver, uint256
            minHumidity, uint256 maxHumidity, uint256 minTemperature, uint256
            maxTemperature) public
40      {
41          ComplianceStatus = true;
42          ComplianceSensorReading = 1;
43          InitiatingCounterparty = msg.sender;
44          Owner = InitiatingCounterparty;
45          Counterparty = InitiatingCounterparty;
46          SupplyChainOwner = supplyChainOwner;
47          SupplyChainObserver = supplyChainObserver;
48          MinHumidity = minHumidity;
49          MaxHumidity = maxHumidity;
50          MinTemperature = minTemperature;
51          MaxTemperature = maxTemperature;
52          State = StateType.Created;
53          ComplianceDetail = "N/A";
54
55      }
56
57      function IngestTelemetry(address device, uint256 humidity, uint256 temperature
            , uint256 timestamp) public returns(uint256 posicion)
58      {
59          // Separately check for states and sender
60          // to avoid not checking for state when the sender is the device
61          // because of the logical OR
62          device = msg.sender;
63          dispositivos.push(device);
64          posicion = datos[msg.sender].push(Contenedor(humidity, temperature,
                timestamp))-1;
65          return posicion;
66
67
68          if ( State == StateType.Completed )
69          {
70              revert();
71          }
72
73          if ( State == StateType.OutOfCompliance )
74          {
```

```
75              revert();
76          }
77
78          if (Device != msg.sender)
79          {
80              revert();
81          }
82
83          LastSensorUpdateTimestamp = timestamp;
84
85          if (humidity > MaxHumidity || humidity < MinHumidity)
86          {
87              ComplianceSensorType = SensorType.Humidity;
88              ComplianceSensorReading = humidity;
89              ComplianceDetail = "Humidity value out of range.";
90              ComplianceStatus = false;
91          }
92          else if (temperature > MaxTemperature || temperature < MinTemperature)
93          {
94              ComplianceSensorType = SensorType.Temperature;
95              ComplianceSensorReading = temperature;
96              ComplianceDetail = "Temperature value out of range.";
97              ComplianceStatus = false;
98          }
99
100         if (ComplianceStatus == false)
101         {
102             State = StateType.OutOfCompliance;
103         }
104     }
105
106     function getIdentificadores() external view returns(uint256){
107         return dispositivos.length;
108     }
109
110     function getData(address device, uint256 posicion) public view returns (
        uint256 humedad, uint256 temperatura, uint256 sello){
111         humedad = datos[msg.sender][posicion].humidity;
112         temperatura = datos[msg.sender][posicion].temperature;
113         sello = datos[msg.sender][posicion].timestamp;
114         return(humedad, temperatura, sello);
115     }
116
117
118     function TransferResponsibility(address newCounterparty) public
119     {
120         // keep the state checking, message sender, and device checks separate
121         // to not get cloberred by the order of evaluation for logical OR
122         if ( State == StateType.Completed )
123         {
124             revert();
125         }
126
127         if ( State == StateType.OutOfCompliance )
128         {
129             revert();
130         }
131
132         if ( InitiatingCounterparty != msg.sender && Counterparty != msg.sender )
```

```
        {
            revert();
        }

        if ( newCounterparty == Device )
        {
            revert();
        }

        if (State == StateType.Created)
        {
            State = StateType.InTransit;
        }

        PreviousCounterparty = Counterparty;
        Counterparty = newCounterparty;
    }

    function Complete() public
    {
        // keep the state checking, message sender, and device checks separate
        // to not get cloberred by the order of evaluation for logical OR
        if ( State == StateType.Completed )
        {
            revert();
        }

        if ( State == StateType.OutOfCompliance )
        {
            revert();
        }

        if (Owner != msg.sender && SupplyChainOwner != msg.sender)
        {
            revert();
        }

        State = StateType.Completed;
        PreviousCounterparty = Counterparty;
        Counterparty = 0x0000000000000000000000000000000000000000;
    }
}
```

C.5. RefrigeratedTransport.json

```
{
    "ApplicationName": "RefrigeratedTransportation",
    "DisplayName": "Refrigerated Transportation",
    "Description": "Application to track end-to-end transportation
       of perishable goods.",
    "ApplicationRoles": [
      {
        "Name": "InitiatingCounterparty",
        "Description": "First party who stores or transports a
           shipment."
      },
```

```
10        {
11          "Name": "Counterparty",
12          "Description": "A party who stores or transports a shipment.
              "
13        },
14        {
15          "Name": "Device",
16          "Description": "A device to track humidity and temperature."
17        },
18        {
19          "Name": "Owner",
20          "Description": "The owner who owns the end-to-end shipment."
21        },
22        {
23          "Name": "Observer",
24          "Description": "An observer who has oversight on the end-to-
              end shipment."
25        }
26      ],
27      "Workflows": [
28        {
29          "Name": "RefrigeratedTransportation",
30          "DisplayName": "Refrigerated Transportation",
31          "Description": "Main workflow to track end-to-end
              transportation of perishable goods.",
32          "Initiators": [ "Owner" ],
33          "StartState": "Created",
34          "Properties": [
35            {
36              "Name": "State",
37              "DisplayName": "State",
38              "Description": "Holds the state of the contract",
39              "Type": {
40                "Name": "state"
41              }
42            },
43            {
44              "Name": "Owner",
45              "DisplayName": "Owner",
46              "Description": "The owner of the end-to-end shipment.",
47              "Type": {
48                "Name": "Owner"
49              }
50            },
51            {
52              "Name": "InitiatingCounterparty",
53              "DisplayName": "Initial Party",
```

```
54              "Description": "First party who stored or transported
                   the shipment.",
55              "Type": {
56                "Name": "InitiatingCounterparty"
57              }
58            },
59            {
60              "Name": "Counterparty",
61              "DisplayName": "Current Party",
62              "Description": "Current party who is storing or
                   transporting the shipment.",
63              "Type": {
64                "Name": "Counterparty"
65              }
66            },
67            {
68              "Name": "PreviousCounterparty",
69              "DisplayName": "Last Party",
70              "Description": "Last party who stored or transported the
                    shipment.",
71              "Type": {
72                "Name": "Counterparty"
73              }
74            },
75            {
76              "Name": "Device",
77              "DisplayName": "Device",
78              "Description": "The device used to track humidity and
                   temperature of the shipment.",
79              "Type": {
80                "Name": "Device"
81              }
82            },
83            {
84              "Name": "SupplyChainOwner",
85              "DisplayName": "Owner",
86              "Description": "The owner of the shipment.",
87              "Type": {
88                "Name": "Owner"
89              }
90            },
91            {
92              "Name": "SupplyChainObserver",
93              "DisplayName": "Observer",
94              "Description": "The observer overseeing the shipment.",
95              "Type": {
96                "Name": "Observer"
97              }
```

```
98            },
99            {
100             "Name": "MinHumidity",
101             "DisplayName": "Min Humidity",
102             "Description": "Minimum humidity requirement.",
103             "Type": {
104               "Name": "int"
105             }
106           },
107           {
108             "Name": "MaxHumidity",
109             "DisplayName": "Max Humidity",
110             "Description": "Max humidity requirement.",
111             "Type": {
112               "Name": "int"
113             }
114           },
115           {
116             "Name": "MinTemperature",
117             "DisplayName": "Min Temperature",
118             "Description": "Min temperature requirement.",
119             "Type": {
120               "Name": "int"
121             }
122           },
123           {
124             "Name": "MaxTemperature",
125             "DisplayName": "Max Temperature",
126             "Description": "Max temperature requirement.",
127             "Type": {
128               "Name": "int"
129             }
130           },
131           {
132             "Name": "ComplianceSensorType",
133             "DisplayName": "Sensor Type",
134             "Description": "The type of IoT sensor used to read out
                  of compliance reading. Either humidity or
                  temperature.",
135             "Type": {
136               "Name": "enum",
137               "EnumValues": ["None", "Humidity", "Temperature"]
138             }
139           },
140           {
141             "Name": "ComplianceSensorReading",
142             "DisplayName": "Sensor Reading",
```

```
143             "Description": "The IoT sensor value for the out of
                    compliance reading.",
144             "Type": {
145               "Name": "int"
146             }
147           },
148           {
149             "Name": "ComplianceStatus",
150             "DisplayName": "Status",
151             "Description": "Boolean to indicate whether the shipment
                    is in compliance or not.",
152             "Type": {
153               "Name": "bool"
154             }
155           },
156           {
157             "Name": "ComplianceDetail",
158             "DisplayName": "Detail",
159             "Description": "A friendly string indicating the issue
                    and sensor reading.",
160             "Type": {
161               "Name": "string"
162             }
163           },
164           {
165             "Name": "LastSensorUpdateTimestamp",
166             "DisplayName": "Sensor Time",
167             "Description": "The time the sensor reading was taken.",
168             "Type": {
169               "Name": "int"
170             }
171           }
172         ],
173         "Constructor": {
174           "Parameters": [
175             {
176               "Name": "device",
177               "Description": "...",
178               "DisplayName": "Device",
179               "Type": {
180                 "Name": "Device"
181               }
182             },
183             {
184               "Name": "supplyChainOwner",
185               "Description": "...",
186               "DisplayName": "Owner",
187               "Type": {
```

```
                "Name": "Owner"
              }
            },
            {
              "Name": "supplyChainObserver",
              "Description": "...",
              "DisplayName": "Observer",
              "Type": {
                "Name": "Observer"
              }
            },
            {
              "Name": "minHumidity",
              "Description": "...",
              "DisplayName": "Min Humidity",
              "Type": {
                "Name": "int"
              }
            },
            {
              "Name": "maxHumidity",
              "Description": "...",
              "DisplayName": "Max Humidity",
              "Type": {
                "Name": "int"
              }
            },
            {
              "Name": "minTemperature",
              "Description": "...",
              "DisplayName": "Min Temperature",
              "Type": {
                "Name": "int"
              }
            },
            {
              "Name": "maxTemperature",
              "Description": "...",
              "DisplayName": "Max Temperature",
              "Type": {
                "Name": "int"
              }
            }
          ]
        },
        "Functions": [
          {
            "Name": "IngestTelemetry",
```

```
236             "DisplayName": "Ingest Telemetry",
237             "Description": "...",
238             "Parameters": [
239               {
240                 "Name": "humidity",
241                 "Description": "...",
242                 "DisplayName": "Humidity",
243                 "Type": {
244                   "Name": "int"
245                 }
246               },
247               {
248                 "Name": "temperature",
249                 "Description": "...",
250                 "DisplayName": "Temperature",
251                 "Type": {
252                   "Name": "int"
253                 }
254               },
255               {
256                 "Name": "timestamp",
257                 "Description": "...",
258                 "DisplayName": "Timestamp",
259                 "Type": {
260                   "Name": "int"
261                 }
262               }
263             ]
264           },
265           {
266             "Name": "TransferResponsibility",
267             "DisplayName": "Transfer Responsibility",
268             "Description": "...",
269             "Parameters": [
270               {
271                 "Name": "newCounterparty",
272                 "Description": "...",
273                 "DisplayName": "Party",
274                 "Type": {
275                   "Name": "Counterparty"
276                 }
277               }
278             ]
279           },
280           {
281             "Name": "Complete",
282             "DisplayName": "Complete",
283             "Description": "...",
```

```
            "Parameters": []
          }
        ],
        "States": [
          {
            "Name": "Created",
            "DisplayName": "Created",
            "Description": "...",
            "PercentComplete": 10,
            "Value": 0,
            "Style": "Success",
            "Transitions": [
              {
                "AllowedRoles": [],
                "AllowedInstanceRoles": [ "InitiatingCounterparty"
                   ],
                "Description": "...",
                "Function": "TransferResponsibility",
                "NextStates": [ "InTransit" ],
                "DisplayName": "Transfer Responsibility"
              },
              {
                "AllowedRoles": [],
                "AllowedInstanceRoles": [ "Device" ],
                "Description": "...",
                "Function": "IngestTelemetry",
                "NextStates": [ "OutOfCompliance", "Created" ],
                "DisplayName": "Ingest Telemetry"
              }
            ]
          },
          {
            "Name": "InTransit",
            "DisplayName": "In Transit",
            "Description": "...",
            "PercentComplete": 50,
            "Value": 1,
            "Style": "Success",
            "Transitions": [
              {
                "AllowedRoles": [],
                "AllowedInstanceRoles": [ "Counterparty" ],
                "Description": "...",
                "Function": "TransferResponsibility",
                "NextStates": [ "InTransit" ],
                "DisplayName": "Transfer Responsibility"
              },
              {
```

```
                "AllowedRoles": [],
                "AllowedInstanceRoles": [ "Device" ],
                "Description": "...",
                "Function": "IngestTelemetry",
                "NextStates": [ "OutOfCompliance", "InTransit" ],
                "DisplayName": "Ingest Telemetry"
              },
              {
                "AllowedRoles": [],
                "AllowedInstanceRoles": [ "Owner" ],
                "Description": "...",
                "Function": "Complete",
                "NextStates": [ "Completed" ],
                "DisplayName": "Complete"
              }
            ]
          },
          {
            "Name": "Completed",
            "DisplayName": "Completed",
            "Description": "...",
            "PercentComplete": 100,
            "Value": 2,
            "Style": "Success",
            "Transitions": []
          },
          {
            "Name": "OutOfCompliance",
            "DisplayName": "Out Of Compliance",
            "Description": "...",
            "PercentComplete": 100,
            "Value": 3,
            "Style": "Failure",
            "Transitions": []
          }
        ]
      }
    ]
  }
```

Referencias

[1] K. Schwab, «The Fourth Industrial Revolution: what it means, how to respond», World Economic Forum, Tech. Rep., 2016. [Online]. Available: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/.

[2] B. Villazán, *Lecture slides - Smart Systems I*, Sep. 2018.

[3] D. Ledger. Common protocol and standards used in IoT Systems, [Online]. Available: https://medium.com/@dledge/making-sense-of-the-myriad-of-iot-standards-and-protocols-88dc4792ba1f (visited on 03/26/2020).

[4] 2. Arduino. (Jun. 2020). What is Arduino? (Accessed on 09/06/2020).

[5] Electrónica Embajadores. DHT22 - Sensor de Humedad y Temperatura - AM2302. (Accessed on 12/07/2020), [Online]. Available: https://www.electronicaembajadores.com/es/Productos/Detalle/SSHUDHT22/sensores/sensores-de-humedad-agua/dht22-sensor-de-humedad-y-temperatura-am2302.

[6] SierraWireless. (Apr. 2018). LTE-M vs. NB-IoT: Make the Best Choice for Your Needs. (Accessed on 30/06/2020).

[7] AWS. AWS IoT. (Accessed on 12/07/2020), [Online]. Available: https://aws.amazon.com/es/iot/.

[8] O. Webinars. (Jun. 2020). Amazon Web Services vs. Microsoft Azure. (Accessed on 02/07/2020).

[9] Azure. IoT Hub. (Accessed on 07/07/2020), [Online]. Available: https://azure.microsoft.com/es-es/services/iot-hub/.

[10] M. Shower, *Top Blockchain-as-a-Service Providers for 2019*, https://mediashower.com/blog/blockchain-as-a-service-providers/, (Accessed on 07/02/2020), Nov. 2019.

[11] Bredalee, «Delivering Data from IoT Hub to Azure Blockchain Workbench», Jun. 2020. [Online]. Available: https://github.com/Azure-Samples/blockchain/blob/master/blockchain-workbench/iot-integration-samples/ConfigureIoTDemo.md.

[12] Azure. Logic App Service. (Accessed on 09/07/2020), [Online]. Available: https://azure.microsoft.com/es-es/services/logic-apps/.

[13] P. W. Kingsford, «James Watt | Biography, Inventions, Steam Engine», in *Encyclopaedia Britannica*, (Accessed on 03/18/2020), Springer.

[14] J. Elcacho, «El gas del cambio climático marca un récord nunca visto en la historia humana», May 2019. [Online]. Available: https://www.lavanguardia.com/natural/20190514/462242832581/concentracion-dioxido-cabono-co2-atmosfera-bate-record-historia-humanidad.html.

[15] (Jun. 2020). La Segunda Revolución Industrial. (Accessed on 28/06/2020).

[16] R. Black. (Jun. 2020). Las cicatrices del calentamiento global desde la revolución industrial. (Accessed on 28/06/2020).

[17] M. Lee, J. J. Yun, A. Pyka, D. Won, F. K. adn Giovanni Schiuma, H. Park, J. Jeon, K. Park, K. Jung, M.-R. Yan, S. Lee, and X. Zhao, «How to Respond to the Fourth Industrial Revolution, or the Second Information Technology Revolution? Dynamic New Combinations between Technology, Market, and Society through Open Innovation», vol. Journal of Open Innovation: Technology, Market, and Complexity, 2018. [Online]. Available: %5Curl%7Bhttps://www.mdpi.com/2199-8531/4/3/21%7D.

[18] D. Abood and A. Quilligan, «Industry X.0 Combine and Conquer. Unlocking the power of digital», Accenture, Tech. Rep., 2017.

[19] I. T. 2018. Digitally Transform your Business to Industry 4.0, [Online]. Available: http:www.iristechnologygroup.com/ (visited on 03/18/2020).

[20] J. L. Romero Ugarte, «Distributed ledger technology (DLT): introduction», Banco de España, Tech. Rep., 2018. [Online]. Available: https://www.bde.es/bde/es/.

[21] S. Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System», 2008.

[22] N. Szabo, «Smart Contracts», vol. Journal of Open Innovation: Technology, Market, and Complexity, 1994.

[23] K. Christidis and M. Devetsikiotis, «Blockchains and Smart Contracts for the Internet of Things», May 2016, (Accessed on 06/29/2020).

[24] N. Unidas.

[25] The Internet of Things is going mainstream, Microsoft survey finds | Transform. (Accessed on 04/23/2020).

[26] E. de Leyva and J. Oriol Dolz de Espejo, «Hardware Platforms Summary», Universidad Pontificia Comillas, Tech. Rep., 2019. [Online]. Available: https://sifo.comillas.edu/pluginfile.php/2632923/mod_resource/content/1/HW20Summary.pdf.

[27] 2. Arduino. (Jun. 2020). MKR Family. (Accessed on 07/06/2020).

[28] S. Rilo and M. Akim, «LPWAN», Universidad Pontificia Comillas, Tech. Rep., 2019. [Online]. Available: https://sifo.comillas.edu/pluginfile.php/2634696/mod_resource/content/1/LPWAN_Comparison.pdf.

[29] S. T. Help. (Jun. 2020). 10 Best IoT Platforms To Watch Out In 2020. (Accessed on 02/07/2020).

[30] AWS. AWS Kinesis. (Accessed on 12/07/2020), [Online]. Available: https://aws.amazon.com/es/kinesis/.

[31] Azure. Event Hubs. (Accessed on 12/07/2020), [Online]. Available: https://azure.microsoft.com/es-es/services/event-hubs/.

[32] G2. (Apr. 2020). Azure Blockchain Workbench Alternatives & Competitors. (Accessed on 02/04/2020).

[33] M. Azure. (Dec. 2016). Azure Blockchain Workbench. (Accessed on 02/04/2020).

[34] Trustradius. (Apr. 2020). Blockchain-as-a-Service (BaaS) Solutions. (Accessed on 02/04/2020).

[35] Leeway Hertz, «Blockchain IoT Use Cases», Jan. 2020. [Online]. Available: https://www.leewayhertz.com/blockchain-iot-use-cases-real-world-products/.

[36] B. Carson, G. Romanelli, P. Walsh, and A. Zhumaev, «Blockchain beyond the hype: What is the strategic business value?», Jan. 2018. [Online]. Available: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/blockchain-beyond-the-hype-what-is-the-strategic-business-value.

[37] D. O. Baecker and S. Jain, «Can blockchain accelerate Internet of Things (IoT) adoption?», Jan. 2020. [Online]. Available: https://www2.deloitte.com/ch/en/pages/innovation/articles/blockchain-accelerate-iot-adoption.html.

[38] triedral. Arduino Latex Listing. (Accessed on 07/07/2020), [Online]. Available: https://github.com/trihedral/ArduinoLatexListing.

[39] F. M. Lagos Suárez. Incluir código Arduino en documentos LaTeX. (Accessed on 07/07/2020), [Online]. Available: https://es.overleaf.com/latex/examples/incluir-codigo-arduino-en-documentos-latex/hthccwgsgktb.

[40] S. Backend. Device - List. (Accessed on 07/07/2020), [Online]. Available: https://backend.sigfox.com/device/list.

[41] Azure, «Install on-premises data gateway for Azure Logic Apps», Jun. 2020. [Online]. Available: https://docs.microsoft.com/es-es/azure/logic-apps/logic-apps-gateway-install.

[42] C. Pietschmann, «Azure Blockchain Hands-on Lab – Microsoft Cloud Workshop», Jan. 2020. [Online]. Available: https://build5nines.com/mcw-azure-blockchain/.

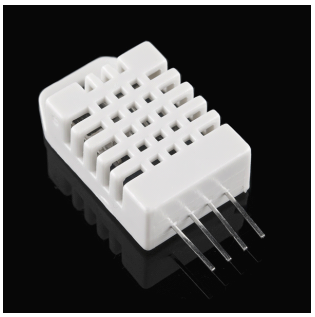
[43] D. Burela, «Refrigerated Transportation Sample Application for Azure Blockchain Workbench», Feb. 2019. [Online]. Available: https://github.com/Azure-Samples/blockchain/tree/master/blockchain-workbench/application-and-smart-contract-samples/refrigerated-transportation.

[44] CodeAcademy. Learn the Basics of Blockchain with Python. (Accessed on 14/10/2019), [Online]. Available: https://www.codecademy.com/learn/introduction-to-blockchain.

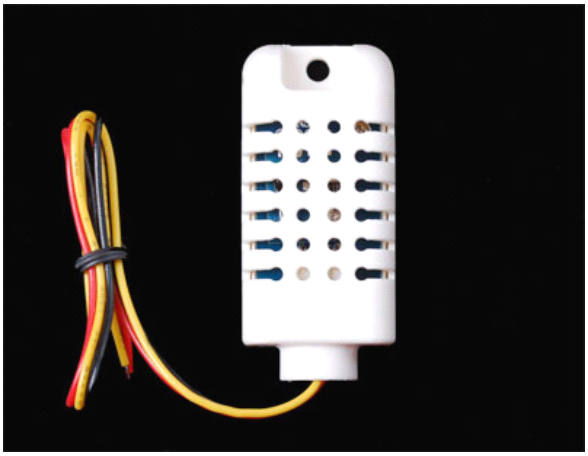
[45] Loom Netwoork. Solidity Path: Beginner to Intermediate Smart Contracts. (Accessed on 12/07/2020), [Online]. Available: https://cryptozombies.io/es/lesson/2/chapter/7.

[46] C. Gutierrez and A. Khizhniak, «IoT Meets Blockchain: Building a Supply Chain App on Microsoft Azure», Apr. 2019. [Online]. Available: https://www.altoros.com/blog/iot-meets-blockchain-building-a-supply-chain-app-on-microsoft-azure/.

[47] Arduino. Arduino MKR Fox 1200. (Accessed on 12/07/2020), [Online]. Available: https://store.arduino.cc/arduino-mkr-fox-1200-1408.



Standard AM2302/DHT22



AM2302/DHT22 with big case and wires

Digital relative humidity & temperature sensor AM2302/DHT22

1. Feature & Application:

*High precision
*Capacitive type
*Full range temperature compensated
*Relative humidity and temperature measurement
*Calibrated digital signal
*Outstanding long-term stability
*Extra components not needed
*Long transmission distance, up to 100 meters
*Low power consumption
*4 pins packaged and fully interchangeable

2. Description:

AM2302 output calibrated digital signal. It applys exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability. Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

Small size & low consumption & long transmission distance(100m) enable AM2302 to be suited in all kinds of harsh application occasions. Single-row packaged with four pins, making the connection very convenient.

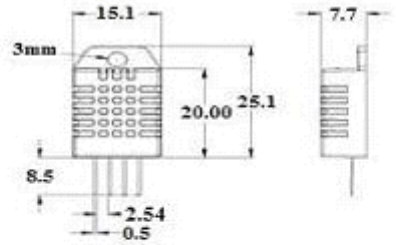
3. Technical Specification:

Model	AM2302
Power supply	3.3-5.5V DC
Output signal	digital signal via 1-wire bus
Sensing element	Polymer humidity capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	**humidity +-2%RH**(Max +-5%RH); temperature +-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Interchangeability	fully interchangeable

Thomas Liu (Sales Manager)

Email: thomasliu198518@aliyun.com

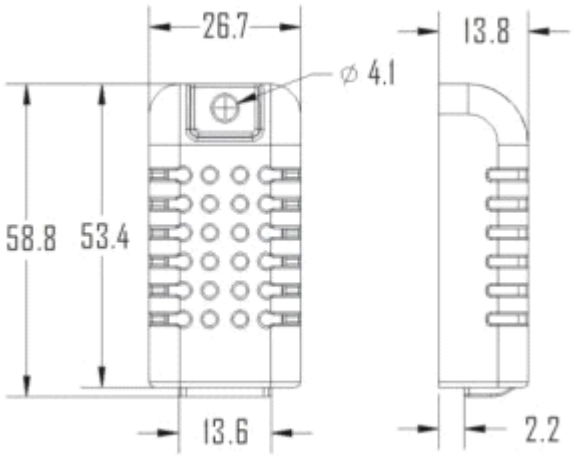
4. Dimensions: (unit----mm)



Pin sequence number: 1 2 3 4 (from left to right direction).

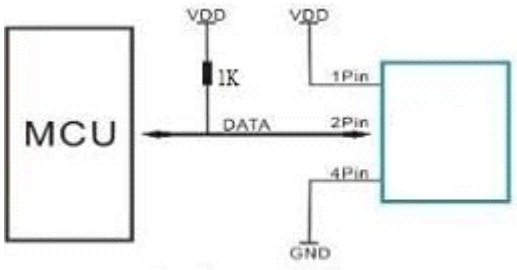
Pin	Function
1	VDD—power supply
2	DATA--signal
3	GND
4	GND

Standard AM2302's dimensions as above



Big case's dimensions as above
Red wire--power supply, Black wire--GND
Yellow wire--Data output

5. Electrical connection diagram:



6. Operating specifications:

(1) Power and Pins

Power's voltage should be 3.3-5.5V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status. One capacitor valued 100nF can be added between VDD and GND for wave filtering.

(2) Communication and signal

1-wire bus is used for communication between MCU and AM2302. (Our 1-wire bus is specially designed, it's different from Maxim/Dallas 1-wire bus, so it's incompatible with Dallas 1-wire bus.)

Illustration of our 1-wire bus:

DATA=16 bits RH data+16 bits Temperature data+8 bits check-sum

Example: MCU has received 40 bits data from AM2302 as

0000 0010 1000 1100 0000 0001 0101 1111 1110 1110

16 bits RH data 16 bits T data check sum

Here we convert 16 bits RH data from binary system to decimal system,

0000 0010 1000 1100 → 652

Binary system Decimal system

RH=652/10=65.2%RH

Here we convert 16 bits T data from binary system to decimal system,

0000 0001 0101 1111 → 351

Binary system Decimal system

T=351/10=35.1℃

When highest bit of temperature is 1, it means the temperature is below 0 degree Celsius.

Example: 1000 0000 0110 0101, T= minus 10.1℃

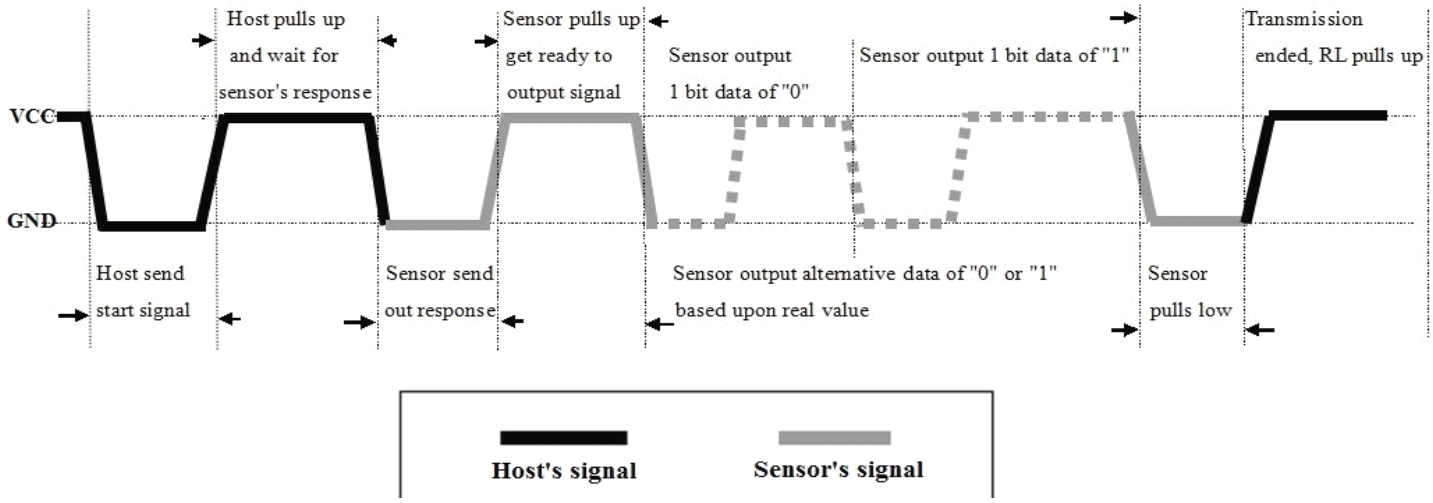
16 bits T data

Sum=0000 0010+1000 1100+0000 0001+0101 1111=1110 1110

Check-sum=the last 8 bits of Sum=1110 1110

When MCU send start signal, AM2302 change from standby-status to running-status. When MCU finishs sending the start signal, AM2302 will send response signal of 40-bit data that reflect the relative humidity and temperature to MCU. Without start signal from MCU, AM2302 will not give response signal to MCU. One start signal for one response data from AM2302 that reflect the relative humidity and temperature. AM2302 will change to standby status when data collecting finished if it don't receive start signal from MCU again.

See below figure for overall communication process, **the interval of whole process must beyond 2 seconds.**



---

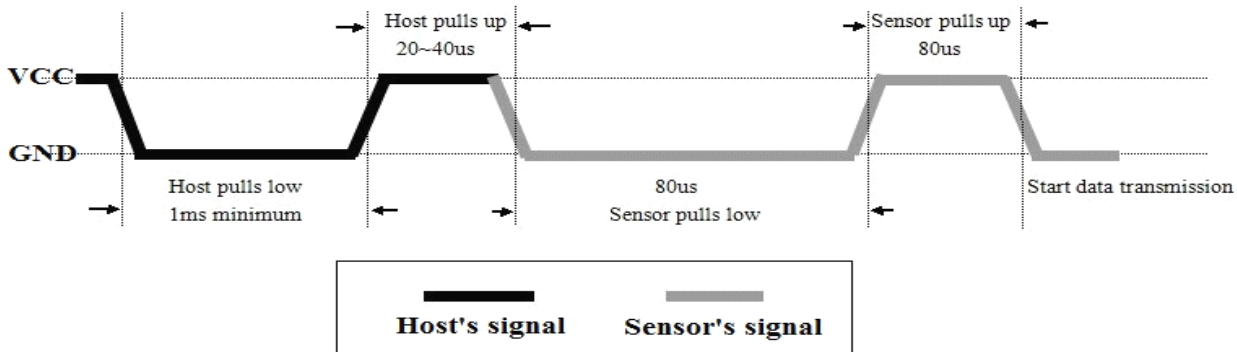
1) Step 1: MCU send out start signal to AM2302 and AM2302 send response signal to MCU

Thomas Liu (Sales Manager)

Email: thomasliu198518@aliyun.com

Data-bus's free status is high voltage level. When communication between MCU and AM2302 begins, MCU will pull low data-bus and this process must beyond at least 1~10ms to ensure AM2302 could detect MCU's signal, then MCU will pulls up and wait 20-40us for AM2302's response.

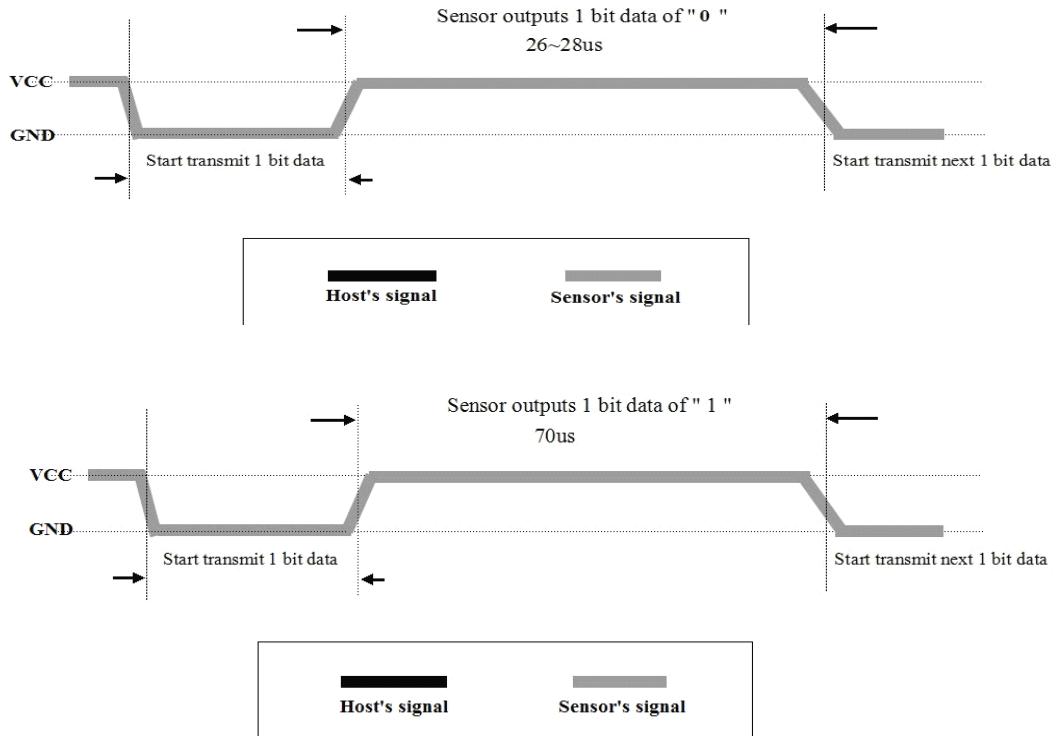
When AM2302 detect the start signal, AM2302 will pull low the bus 80us as response signal, then AM2302 pulls up 80us for preparation to send data. See below figure:



---

2). Step 2: AM2302 send data to MCU

When AM2302 is sending data to MCU, every bit's transmission begin with low-voltage-level that last 50us, the following high-voltage-level signal's length decide the bit is "1" or "0". See below figures:



Attention:

If signal from AM2302 is always high-voltage-level, it means AM2302 is not working properly, please check the electrical connection status.

Thomas Liu (Sales Manager)

Email: thomasliu198518@aliyun.com

7. Electrical Characteristics:

Items	Condition	Min	Typical	Max	Unit
Power supply	DC	3.3	5	6	V
Current supply	Measuring	1		1.5	mA
	Stand-by	40	Null	50	uA
Collecting period	Second		2		Second

8. Attentions of application:

(1) Operating and storage conditions

We don't recommend the applying RH-range beyond the range stated in this specification. The AM2302 sensor can recover after working in abnormal operating condition to calibrated status, but will accelerate sensors' aging.

(2) Attentions to chemical materials

Vapor from chemical materials may interfere AM2302's sensitive-elements and debase AM2302's sensitivity.

(3) Disposal when (1) & (2) happens

Step one: Keep the AM2302 sensor at condition of Temperature 50~60Celsius, humidity <10%RH for 2 hours;

Step two: After step one, keep the AM2302 sensor at condition of Temperature 20~30Celsius, humidity >70%RH for 5 hours.

(4) Attention to temperature's affection

Relative humidity strongly depend on temperature, that is why we use temperature compensation technology to ensure accurate measurement of RH. But it's still be much better to keep the sensor at same temperature when sensing.

AM2302 should be mounted at the place as far as possible from parts that may cause change to temperature.

(5) Attentions to light

Long time exposure to strong light and ultraviolet may debase AM2302's performance.

(6) Attentions to connection wires

The connection wires' quality will effect communication's quality and distance, high quality shielding-wire is recommended.

(7) Other attentions

* Welding temperature should be bellow 260Celsius.
* Avoid using the sensor under dew condition.
* Don't use this product in safety or emergency stop devices or any other occasion that failure of AM2302 may cause personal injury.

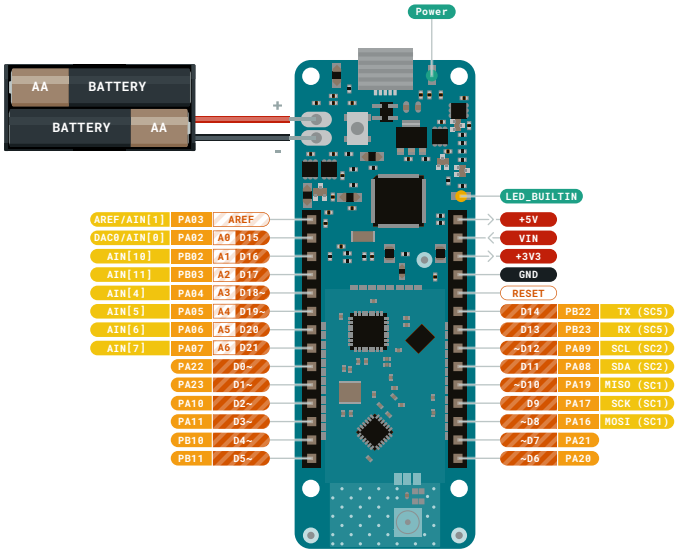
Thomas Liu (Sales Manager)

Email: thomasliu198518@aliyun.com

ARDUINO

ARDUINO
MKR FOX 1200

STORE.ARDUINO.CC/MKR-FOX-1200



- Ground
- Power
- LED
- Internal Pin
- SWD Pin
- Digital Pin
- Analog Pin
- Other Pin
- Microcontroller's Port
- Default

MAXIMUM current per pin is 7mA

MAXIMUM source current is 46mA

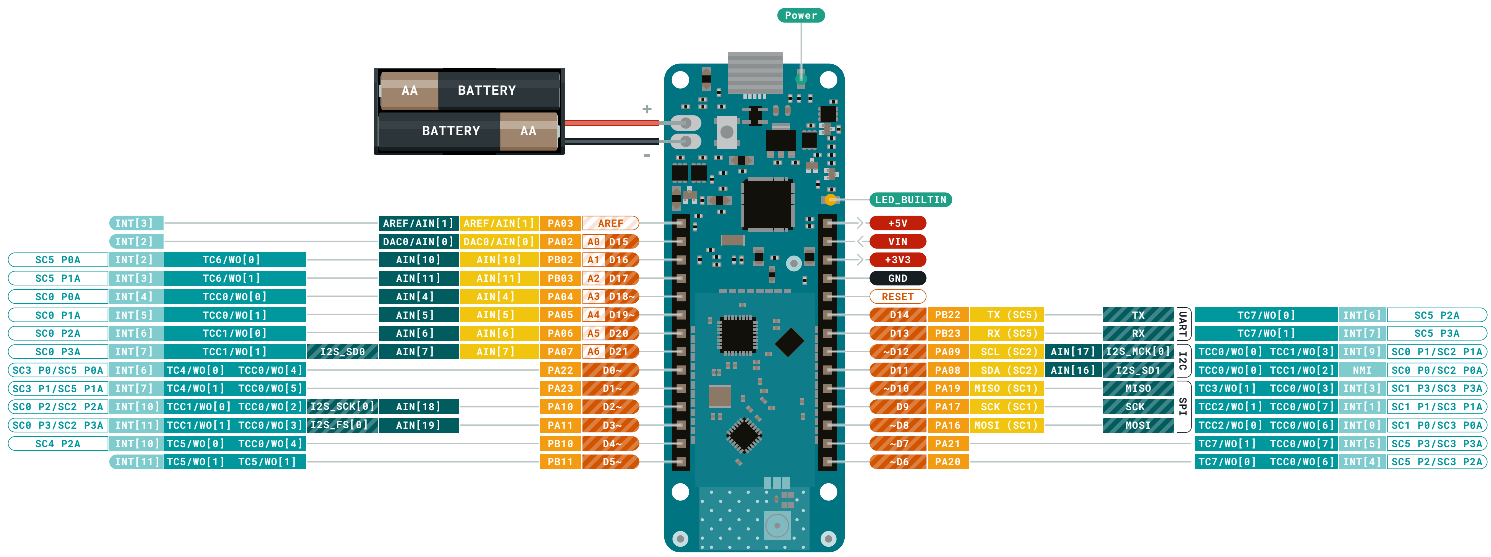
MAXIMUM sink current is 65mA per pin group

VIN Input voltage to the board.

ARDUINO.CC

Last update: 23/03/2020

ARDUINO
MKR FOX 1200
STORE.ARDUINO.CC/MKR-FOX-1200



- Ground
- Power
- LED
- Internal Pin
- SWD Pin
- Digital Pin
- Analog Pin
- Other Pin
- Microcontroller's Port
- Default
- Analog
- Communication
- Timer
- Interrupt
- Sercom

MAXIMUM current per pin is 7mA

MAXIMUM source current is 46mA

MAXIMUM sink current is 65mA per pin group

VIN Input voltage to the board.

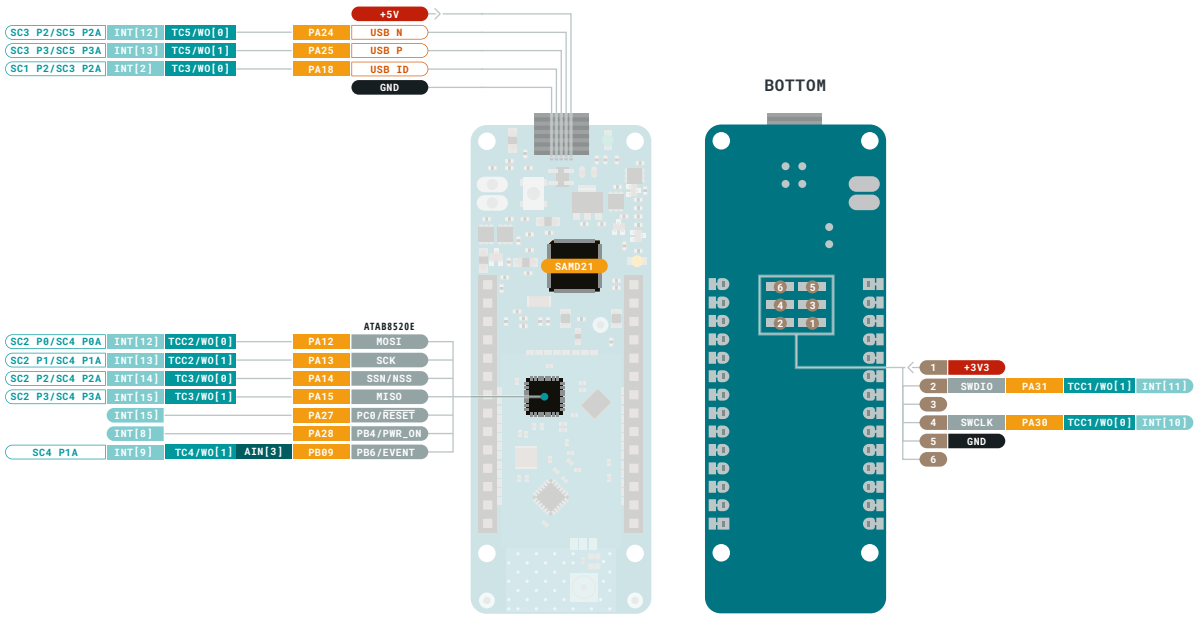
ARDUINO.CC

Last update: 23/03/2020

ARDUINO

ARDUINO
MKR FOX 1200

STORE.ARDUINO.CC/MKR-FOX-1200



- Ground
- Power
- LED
- Internal Pin
- SWD Pin
- Digital Pin
- Analog Pin
- Other Pin
- Microcontroller's Port
- Default
- Analog
- Communication
- Timer
- Interrupt
- Sercom

MAXIMUM current per pin is 7mA

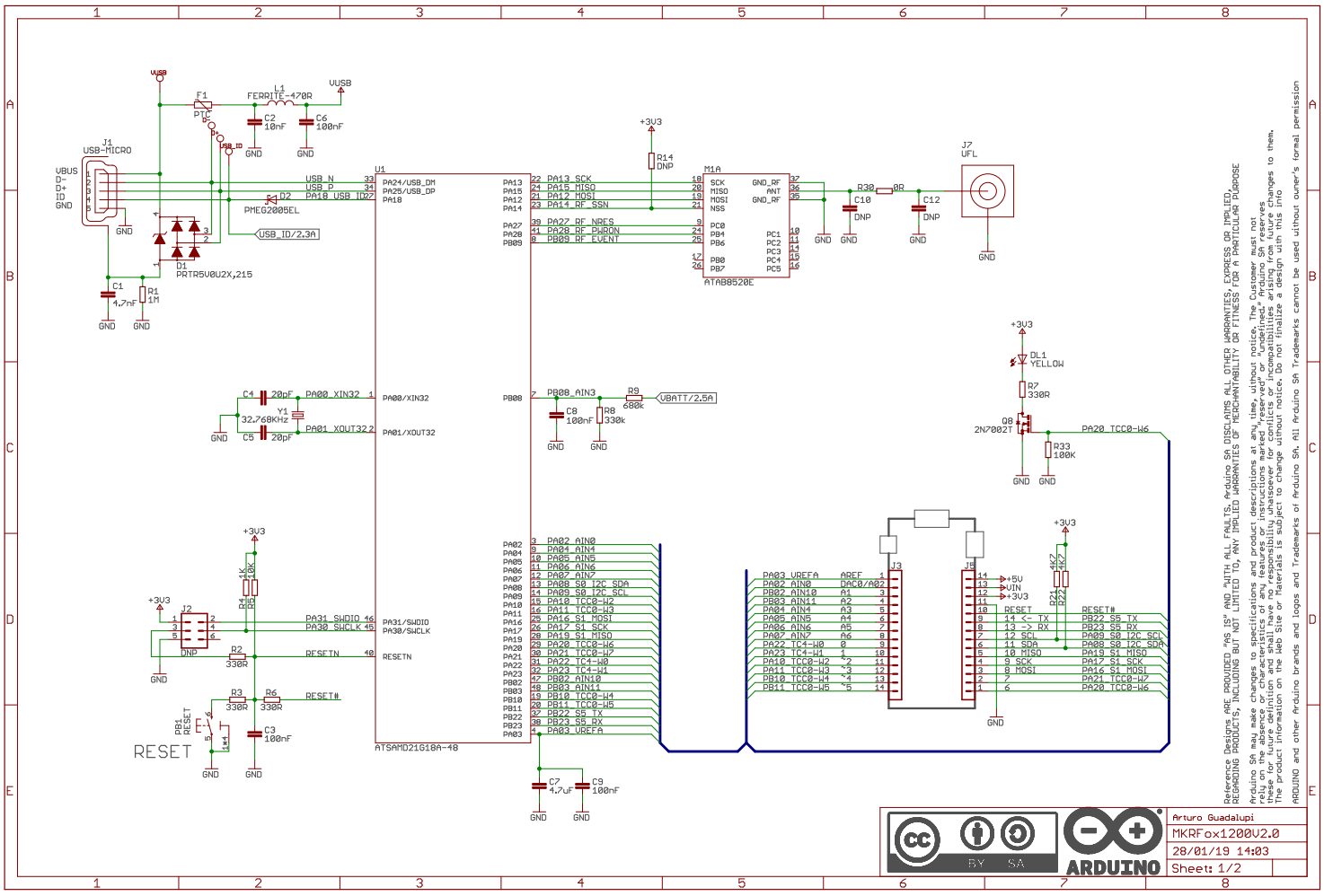
MAXIMUM source current is 46mA

MAXIMUM sink current is 65mA per pin group

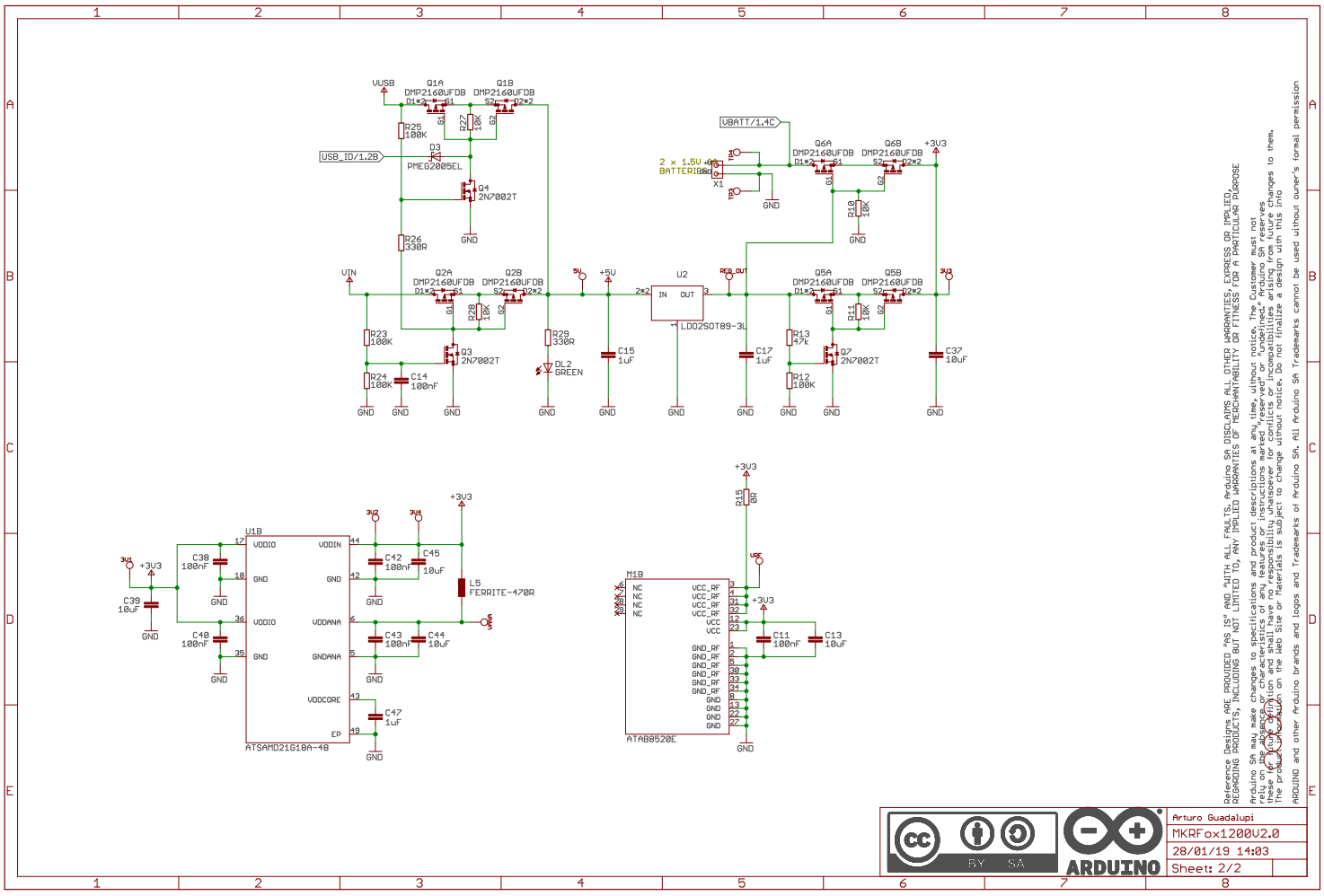
VIN Input voltage to the board.

ARDUINO.CC

Last update: 23/03/2020



Arturo Guadalupi	
MKRFox1200V2.0	
28/01/19 14:03	
Sheet: 1/2	



ARDUINO

Arturo Guadalupi	
MKRFox1200V2.0	
28/01/19 14:03	
Sheet: 2/2	