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

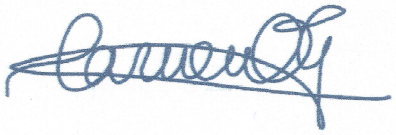
Autora: Carmen Olleros Guerrero

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.



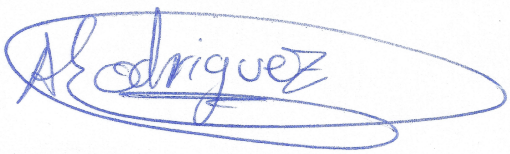
Fdo.: Carmen Olleros Guerrero Fecha: 13/ 07/ 2020

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



AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor Dña. Carmen Olleros Guerrero DECLARA ser el titular de los derechos de propiedad intelectual de la obra: 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, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente)*.

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

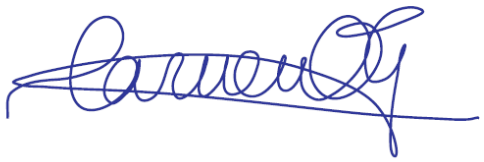
6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 13 de julio de 2020

ACEPTA



Fdo: Carmen Olleros Guerrero

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

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 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 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 es el DHT22 [5].

A continuación, en la Figura 3 se aprecia el cableado de la placa y el sensor.

	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.

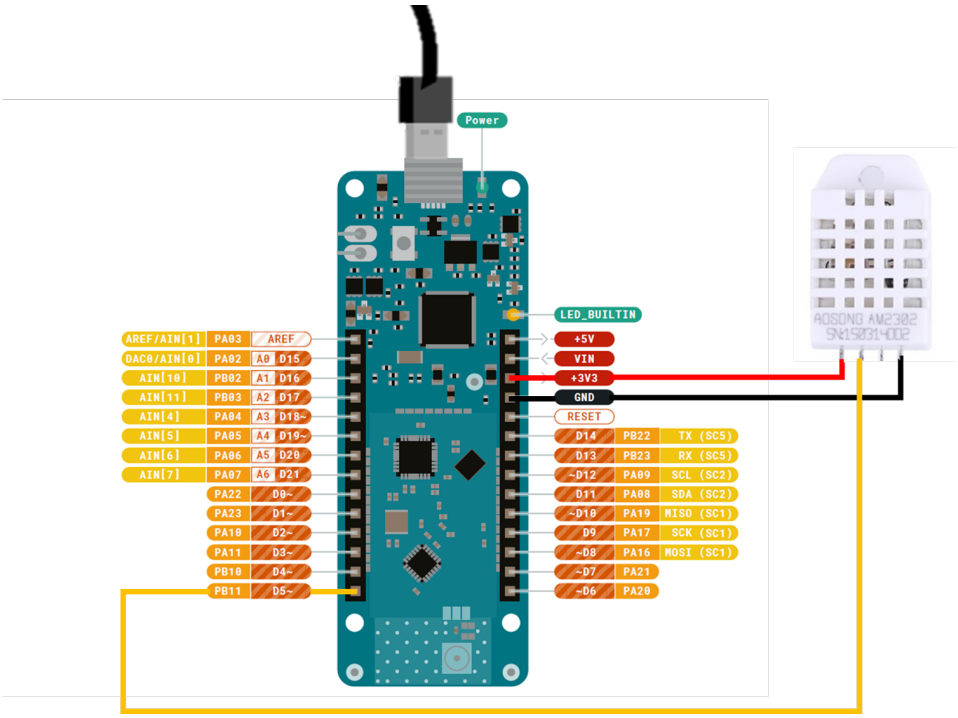


Figura 3. Cableado del sensor en la placa de Arduino.

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 4Antes 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.

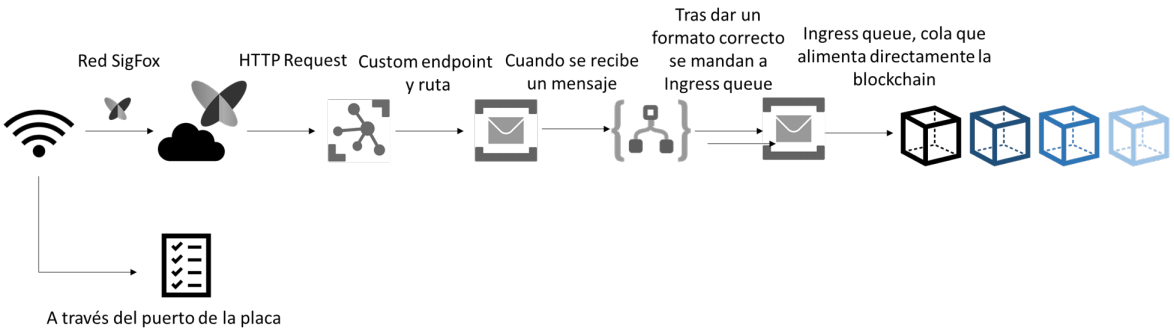


Figura 5. Trayectoria de los datos a lo largo de la aplicación.

Para tener una idea más visual del proceso que se está llevando a cabo, en la Figura 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 6 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 6 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 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 [13].

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 7.

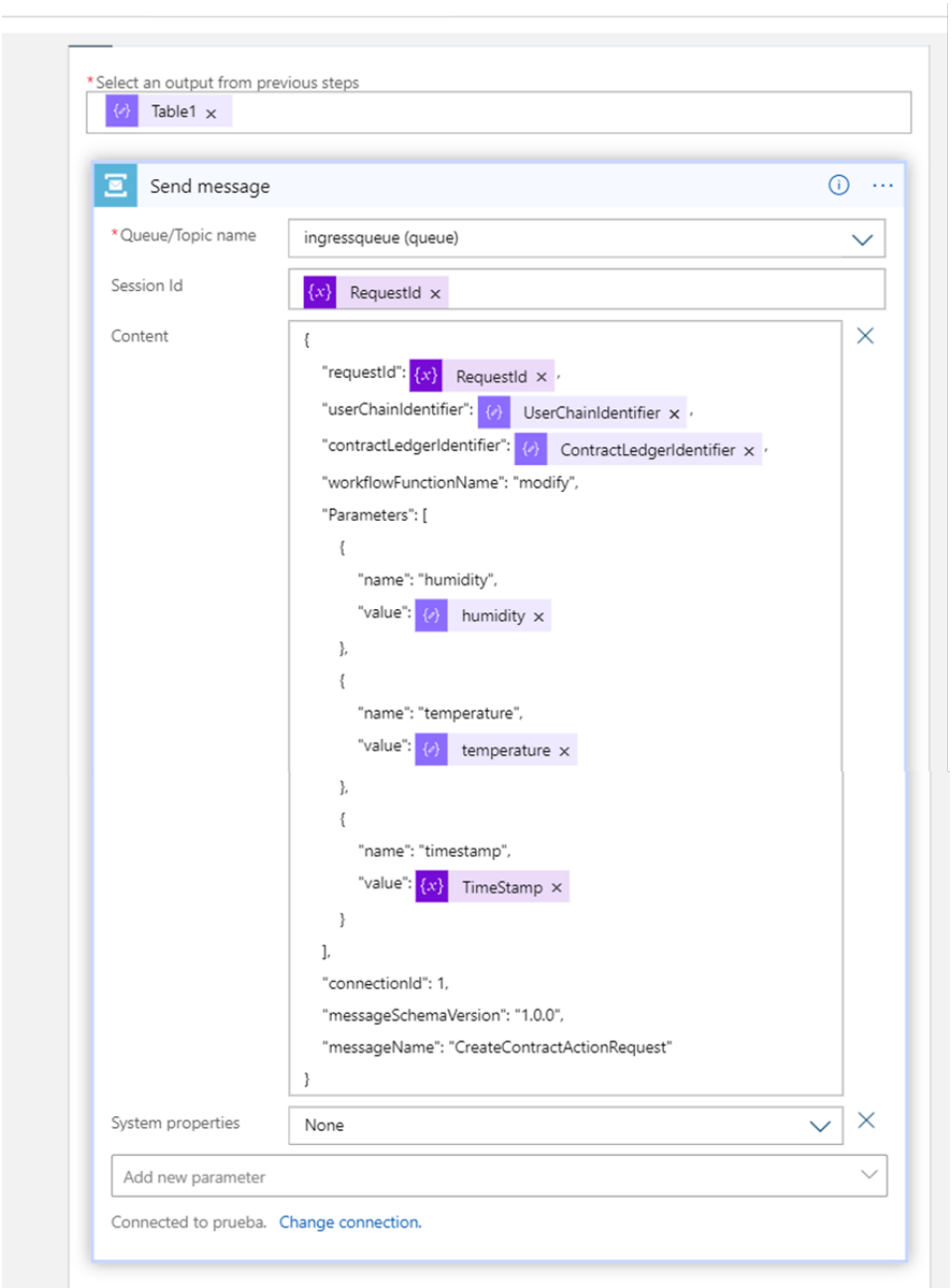


Figura 7. Módulos de envío a *blockchain*.

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 8. En ella se muestran los posibles cambios de estado del contrato.

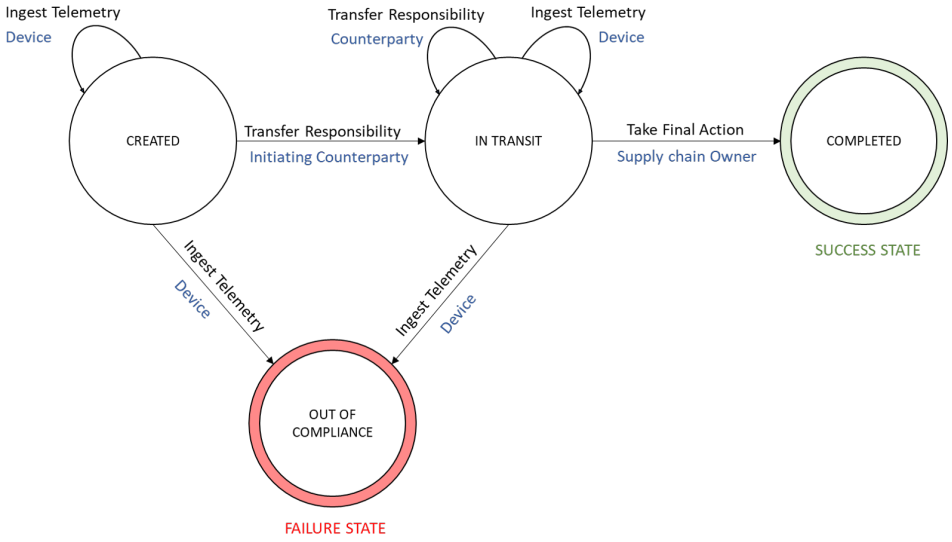


Figura 8. Máquina de estados del contrato.

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 9 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 9 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 10 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.

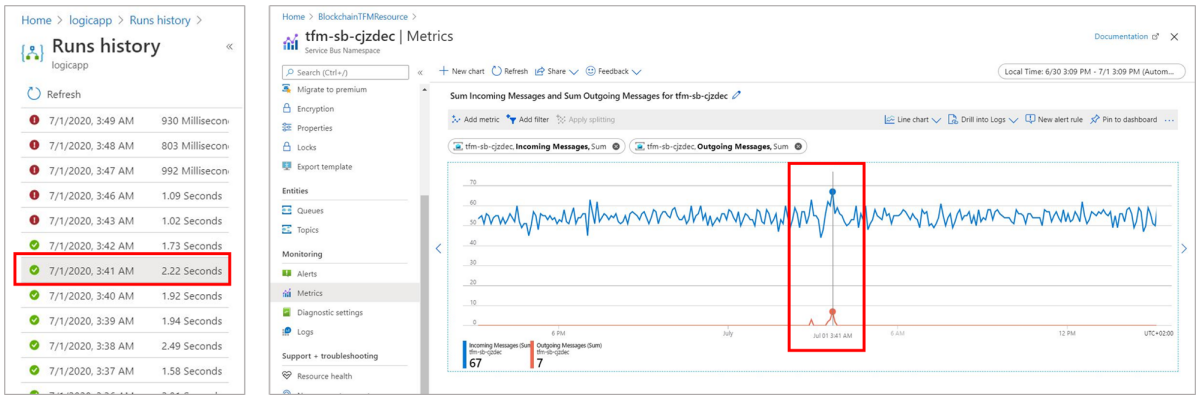


Figura 10. Obtención de la información del *smart contract*.

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.

Referencias

[1] K. Schwab, «The Fourth Industrial Revolution: what it means, how to respond», World Economic Forum, inf. téc., 2016. dirección: 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. de 2018.

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

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

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

[6] SierraWireless. (abr. de 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), dirección: https://aws.amazon.com/es/iot/.

[8] Open Webinars. (jun. de 2020). Amazon Web Services vs. Microsoft Azure. (Accessed on 02/07/2020).

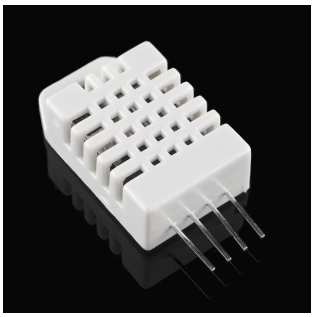
[9] Azure. IoT Hub. (Accessed on 07/07/2020), dirección: https://azure.microsoft.com/es-es/services/iot-hub/.

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

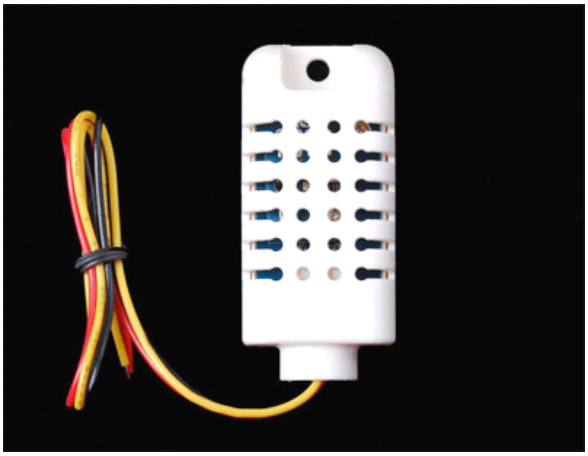
[11] Bredalee, «Delivering Data from IoT Hub to Azure Blockchain Workbench», jun. de 2020. dirección: https://github.com/Azure-Samples/blockchain/blob/master/blockchain-workbench/iot-integration-samples/ConfigureIoTDemo.md.

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

[13] M. Azure. Install on-premises data gateway for Azure Logic Apps. (Accessed on 07/07/2020), dirección: https://docs.microsoft.com/es-es/azure/logic-apps/logic-apps-gateway-install.



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.

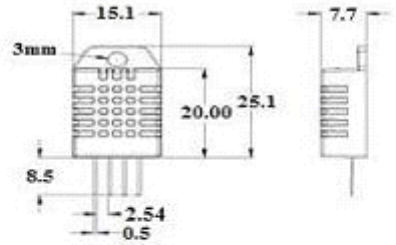
3. Technical Specification:

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

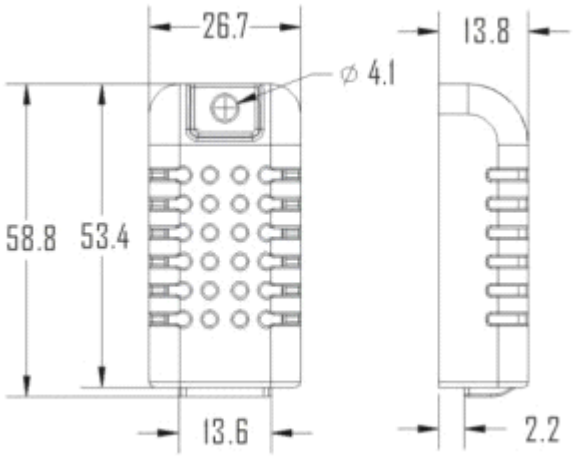
4. Dimensions: (unit----mm)



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

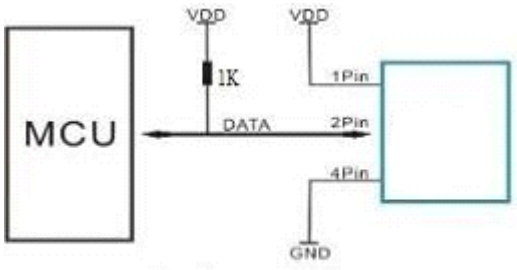
Pin	Function
1	VDD—power supply
2	DATA--signal
3	GND
4	GND

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℃

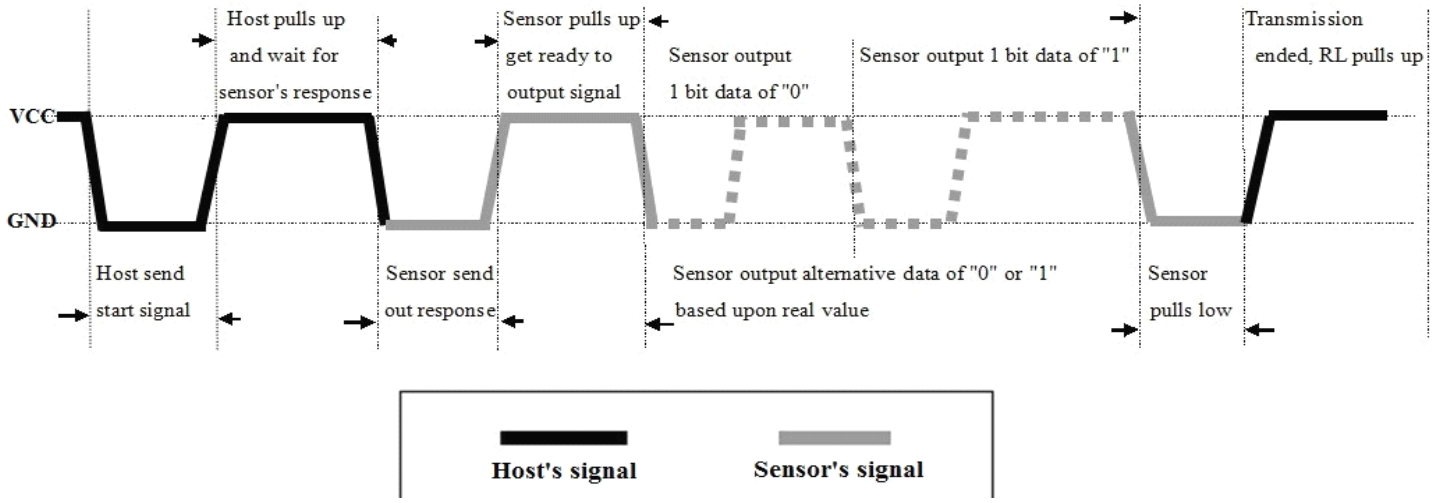
16 bits T data

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.**



---

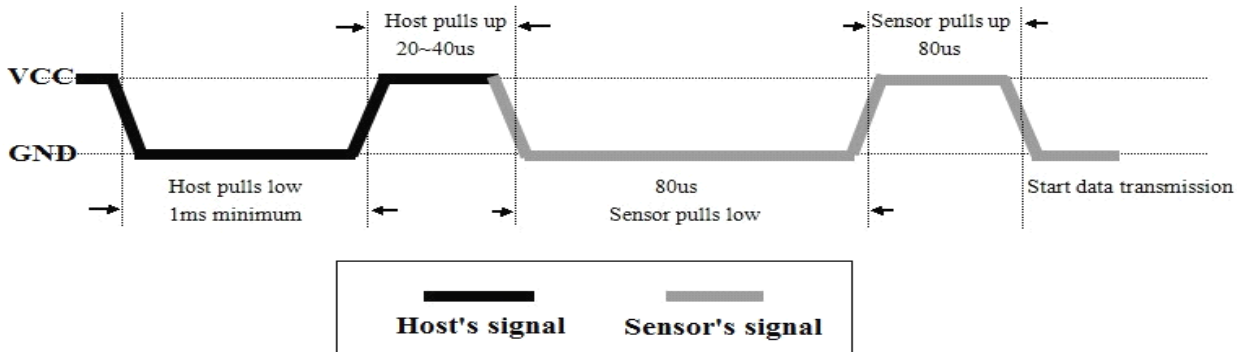
1) Step 1: MCU send out start signal to AM2302 and AM2302 send response signal to MCU

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.

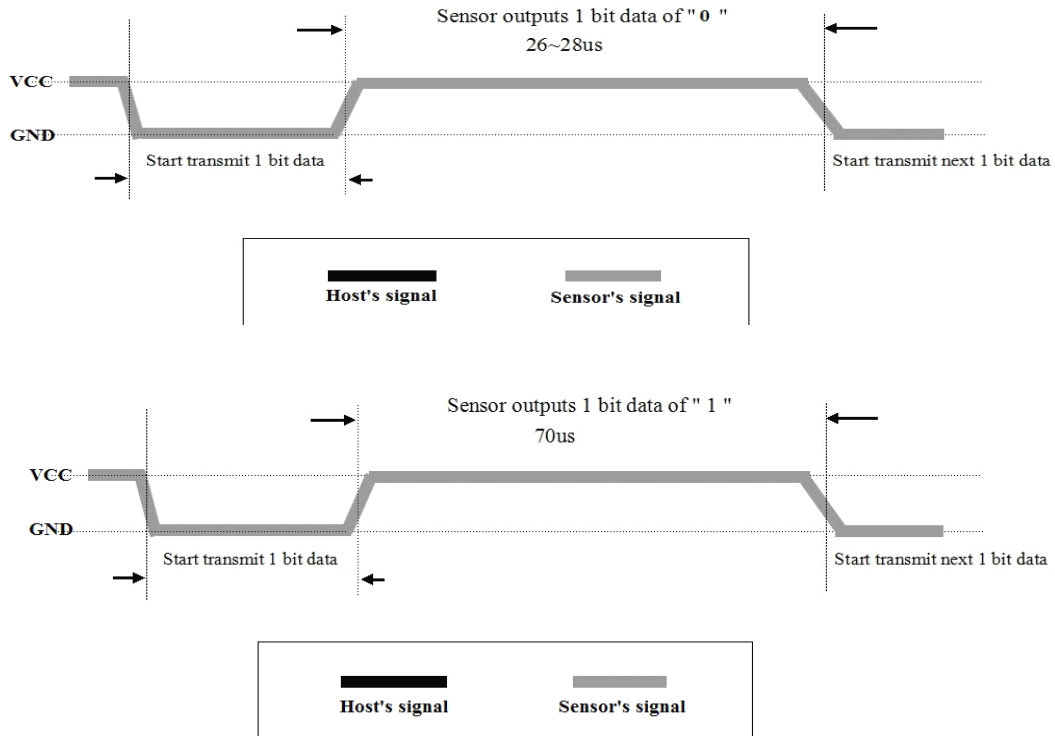
When AM2302 detect the start signal, AM2302 will pull low the bus 80us as response signal, then AM2302 pulls up 80us for preparation to send data. See below figure:



---

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.

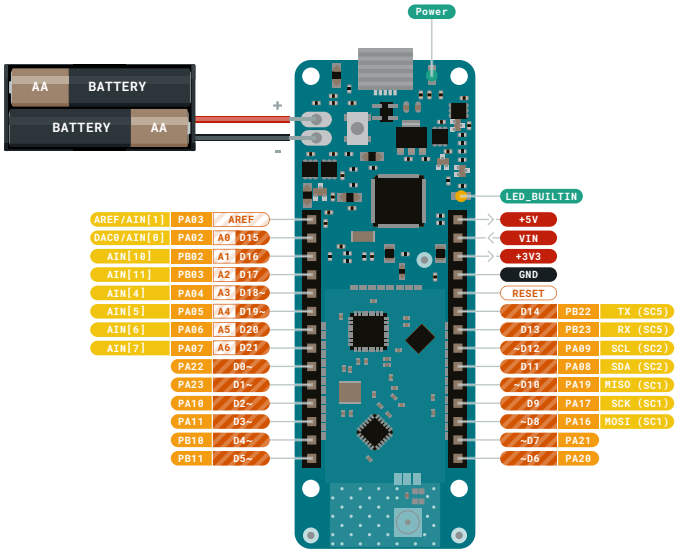
Thomas Liu (Sales Manager)

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

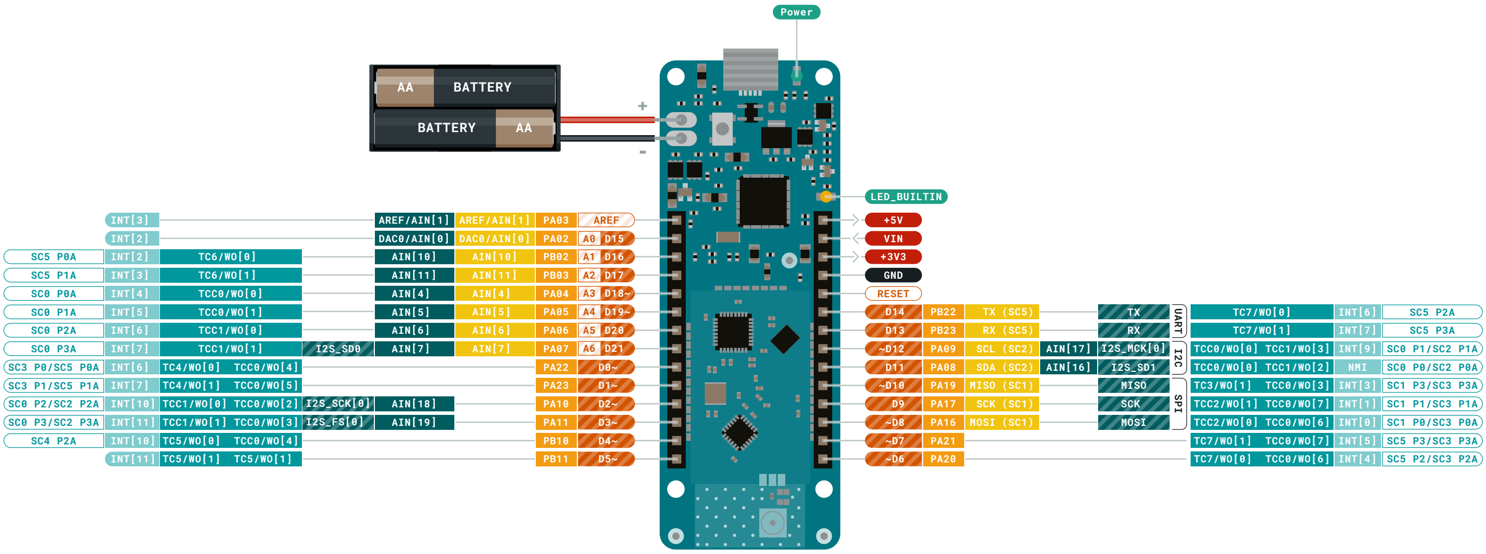
MAXIMUM sink current is 65mA per pin group

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.

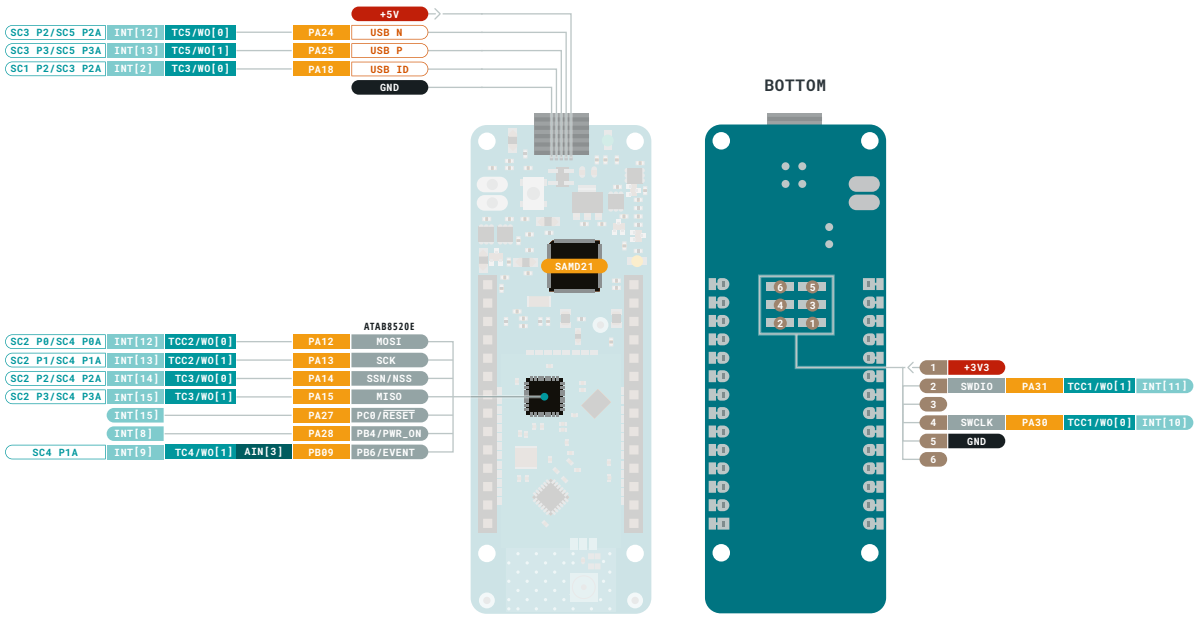
ARDUINO.CC

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