MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

LOGÍSTICA CERTIFICADA EN BLOCKCHAIN

Autor: Ainhoa García Echeverría

Director: Francisco Martín Martinez

Iñigo García de Mata

Madrid

Julio de 2019

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

LOGÍSTICA CERTIFICADA EN BLOCKCHAIN

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2018/2019 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.: Ainhoa García Fecha: 16/07/ 2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Iñigo García de Mata Fecha: 16/07/19

Francisco Martín Martínez

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 D._ Ainhoa García Echeverría DECLARA ser el titular de los derechos de propiedad intelectual de la obra: Logística certificada en Blockchain, 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 17 de Julio de 2019

ACEPTA

Fdo: 

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

LOGÍSTICA CERTIFICADA EN BLOCKCHAIN

Autor: García Echeverría, Ainhoa

Director: García de Mata, Iñigo

Martin Martínez, Francisco

Entidad Colaboradora: Universidad Pontificia de Comillas (ICAI)

RESUMEN DEL PROYECTO

Introducción

Dentro de lo que se conoce como la cuarta revolución industrial, la logística 4.0 se enfrenta entre otros muchos retos al de ser capaz de gestionar la trazabilidad de extremo a extremo en la cadena de suministro. La trazabilidad en la cadena de suministro se ha convertido de ser una ventaja competitiva a prácticamente un requisito con el objetivo de tener una cadena de suministro estable y sostenible. La exigencia creciente por parte de los compradores de la transparencia en el proceso, y la obligación de mejorar la seguridad y control desde el punto de vista de la salud pública, aporta a la trazabilidad una gran proyección de crecimiento en la que el blockchain se presenta como el enfoque perfecto para proporcionar ese valor añadido.

Las ventajas que hacen del blockchain la solución perfecta para los problemas a los que se enfrenta la logística [1]:

- Transparencia y Seguridad: la tecnología Blockchain incluye mecanismos para asegurar que los registros son precisos y, a prueba de manipulaciones. Todas las partes implicadas tienen acceso a un único conjunto de datos creando una única fuente real de información.
- Automatización y digitalización: blockchain permite que los documentos no puedan alterarse y además permite que sean localizados de manera digital.
- Confianza entre distintas partes: la información registrada es inmutable, por tanto el papel del intermediario cambia y la red se encarga de las verificaciones.
- Uso de Smart Contracts: permite agilidad en la cadena de suministro, reduciendo la interacción humana y digitalizando el proceso.

Actualmente, son muchas las empresas que han comenzado a incorporar nuevas tecnologías en sus cadenas de suministro para tener un mayor registro de sus productos, sin embargo, solo se está incorporando en determinadas etapas de la cadena de suministro y en muy pocos casos englobando la trazabilidad de extremo a extremo, aparte que la información que comparten estas empresas aún es muy escasa.

El objetivo principal del proyecto es explicar el proceso de almacenamiento de medidas externas mediante la tecnología Blockchain, desarrollando un prototipo de registro de trazabilidad usando Blockchain y el posterior análisis económico junto con las pruebas de

medición de etiquetas RFID para analizar ventajas tanto en tiempo cómo económicas respecto a métodos más tradicionales.

Metodología

El proyecto tiene dos partes diferenciadas, por un lado, el software relacionado con la creación de un nodo blockchain y la parte de hardware en el que el Arduino lea la información de una etiqueta de RFID. Para el desarrollo del prototipo completo se han seguido los siguientes pasos imprescindibles:

1. Configurar el hardware con la placa Arduino para la lectura de códigos RFID.
2. Configurar las conexiones entre la placa microcontroladora y el servidor:
    a. Se ha elegido cómo dispositivo para permitir las conexiones , una placa ethernet compatible con arduino. Se ha asignado a la placa una IP fija para evitar problemas en la comunicación.
    b. El protocolo de comunicación elegido entre el Arduino y la Raspberry Pi ha sido UDP.
3. Configurar la base de datos en el servidor.
    a. Se ha instalado el servidor web Apache para poder acceder a la base de datos
    b. Se ha instalado MySQL, para posteriormente definir las tablas que se rellenarán de manera automática con la información leída por el Arduino.
4. Configurar el nodo Blockchain. La información disponible sobre esta parte del prototipo es muy limitada. A lo largo del documento, se detallan todos los pasos y problemas que puede encontrarse al usuario para almacenar la información en la Blockchain. La síntesis de los pasos son los siguientes:
    a. Sincronizar la cadena de bloques que requiere de un importante uso de memoria, y por tanto, dependiendo de la placa computadora que se utilice será preciso usar memorias externas.
    b. Una vez con el nodo sincronizado, crear una cuenta con Ether para poder realizar las transacciones.
    c. Crear el contrato inteligente utilizando Remix.
    d. El programa para interactuar con la red tendrá que detallar la dirección del contrato, la cuenta desde donde se realiza la transacción, el ABI del contrato, y la dirección del nodo. Con todos los detalles definidos desde el principio, al enviar las transacciones se detalla la función dentro del contrato que se quiere ejecutar, junto con la cuenta de origen del envío.

Resultados

Las ventajas que se proponen con el prototipo creado en este proyecto incluyen la rapidez en la lectura de etiquetas y el almacenamiento seguro de la información.

Se han realizado pruebas para la medición de tiempos de lectura y escritura de las etiquetas RFID tanto en la base de datos de MySQL cómo en la Blockchain. Los tiempos de lectura de

las etiquetas RFID son de milisegundos, y el almacenamiento en la base de datos también se realiza en cuestión de milisegundos. El tiempo total para ver reflejada la información en la red de Blockchain dependerá de la velocidad de minado del bloque al que se envía la transacción con la información sobre el código leído.

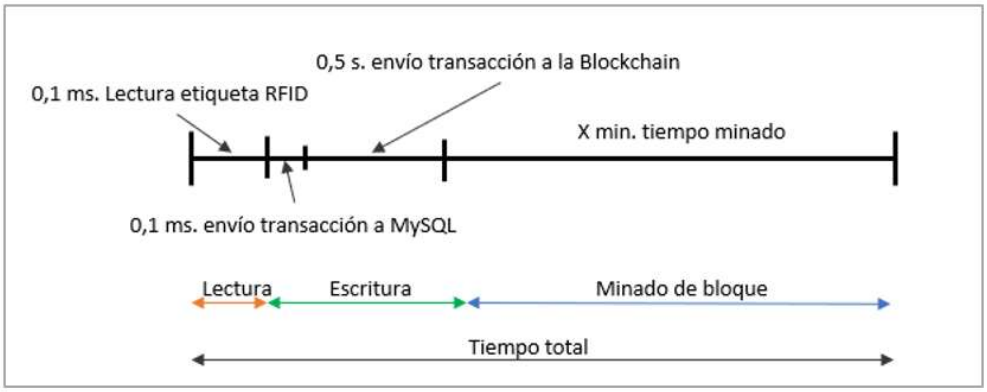


Ilustración 1. Prueba de tiempos de lectura y escritura de etiquetas RFID.

El tiempo de minado del bloque es el que se debe determinar en función de los costes o urgencia de tener desplegada la información en toda la red. En el caso de este prototipo las operaciones a realizar son transacciones simples de envío de información que suponen un gasto de 43061 de gas por cada envío. Se han analizado distintos momentos de congestión de la red catalogados como alta congestión, media y baja, y tres tiempos para ver la información en la red; media hora, una hora y más de una hora. También se ha tenido en cuenta diferentes escenarios dependiendo del número de transacciones que se envíen a un mismo bloque.

A continuación, se muestra la curva de costes para reflejar 22500 productos reflejándolos con una urgencia de 30 minutos. Cómo se observa atendiendo a la congestión de la red el precio puede aumentar hasta en un 580% de 5701USD a 982,5USD en el caso más favorable.

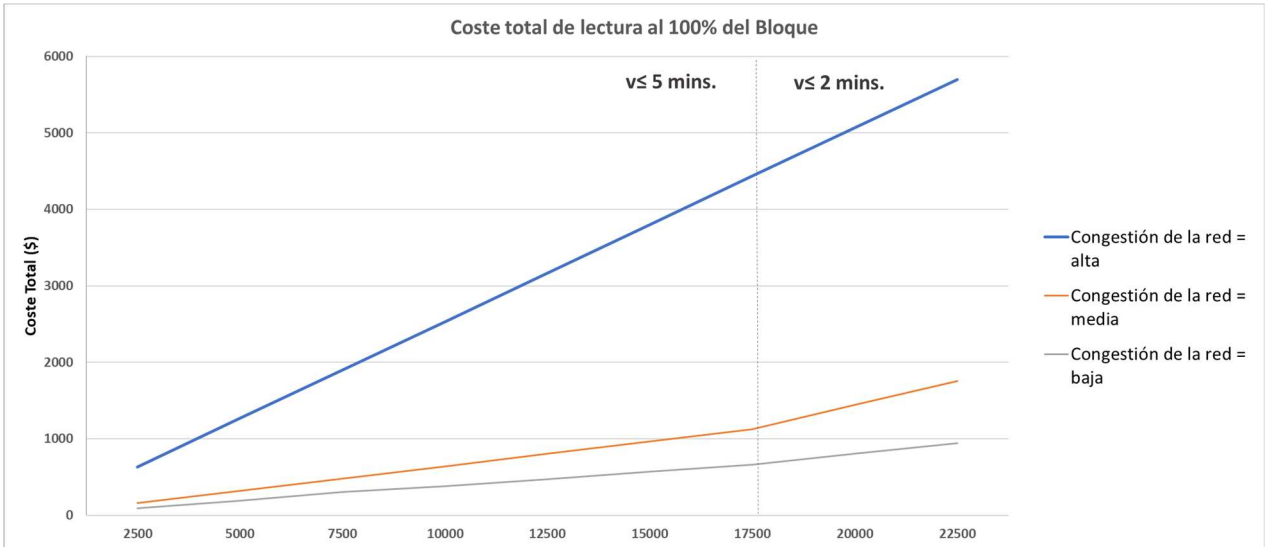


Ilustración 2. Coste por 22500 productos en función de la red.

Si aumentamos el número de productos a almacenar al máximo disponible por hora para el caso optimista, los costes en función de la urgencia de minado y de la congestión de la red son los siguientes:

	ALTA	MEDIA	BAJA
1 hora	$ 11.403,45	$ 2.411,93	$ 1253,65
+1 horas	$ 11.403,45	$ 760,5	$ 627,3

Tabla 1. Coste máximo de transacciones a la hora al 100% del Bloque.

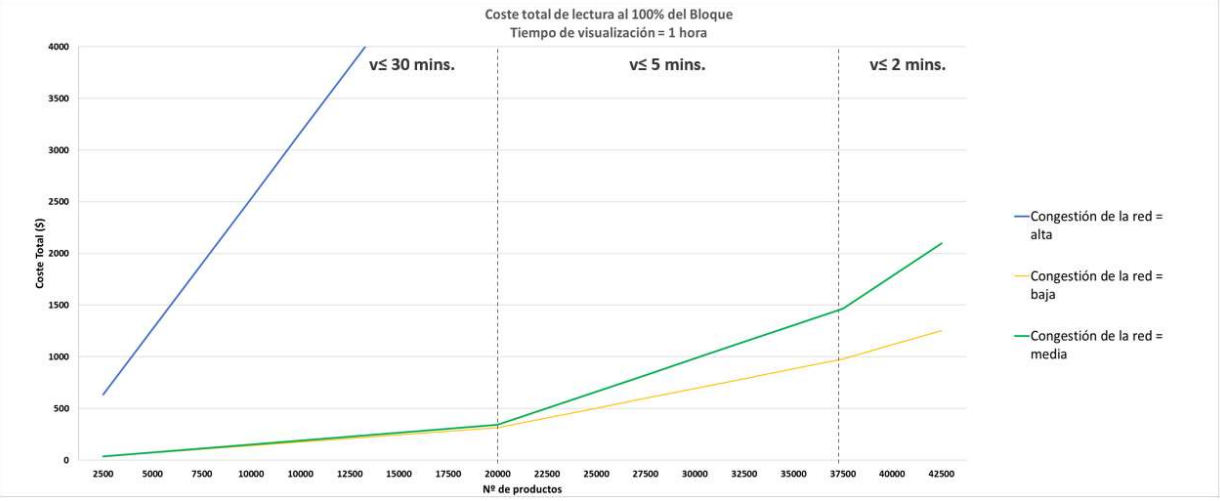


Ilustración 3. Coste por transacción al 100% del Bloque.

En esta Ilustración se observan los tres cambios de pendiente puesto que se muestra el caso límite con el número máximo de productos a procesar. Al querer la información reflejada a los 30 minutos, los costes son elevados, puesto que obliga a minar siempre a una velocidad media o baja. Sin embargo al poder tener un margen de una hora o más, permite que los primeros bloques se minen a velocidad baja (30 minutos), y los costes se reducen notablemente.

Si se busca un escenario más conservador en el que un bloque no puede almacenar todas las transacciones desde la misma cuenta, disminuye el número de productos a procesar en una hora y por tanto los costes. De 45000 productos se reduce a 36000 productos a la hora.

	ALTA	MEDIA	BAJA
1 hora	$ 8869,35	$ 1301,02	$ 970,45
+ 1 horas	$ 8869,35	$ 532,35	$ 487,5

Tabla 2. Coste máximo de transacciones a la hora al 80% del Bloque.

Conclusiones

Las conclusiones obtenidas una vez realizadas las pruebas muestran la viabilidad de este tipo de prototipos para la industria de la logística. Sin embargo, el coste computacional ligado al envío de las transacciones de manera individual a la red puede parecer elevado. Esto es porque, aunque el consumo de gas ligado al envío de tarjetas RFID es bajo, si se aplicase al control de

producción de líneas de grandes fábricas, el volumen de productos a procesar es tan elevado que aumenta el coste asociado. Es importante, que la información no se precise reflejada en la red de manera inmediata sino a lo largo del día de producción, de manera que los costes se reduzcan de manera considerable. Además para poder reducir el gasto es importante considerar aquellos momentos en los que la red está menos saturada, puesto que la tarifa aumenta en hasta un 500%.

Por otro lado, si se observan multitud de aplicaciones en las que el coste asociado sería menor puesto que requeriría de menor número de transacciones diarias, por ejemplo, envío de localización de productos en ciertos intervalos de tiempo. En el prototipo actual sería cambiar el sensor RFID, por uno de localización u otro de temperatura.

Para desarrollos futuros se puede optimizar el gas en la escritura de la Blockchain para reducir estos costes, igual que utilizar recursos más específicos para las funciones.

Referencias

[1] DHL trend research, "Blockchain in logistics". Disponible en: [https://www.logistics.dhl/content/dam/dhl/global/core/documents/pdf/glo-coreblockchain-trend-report.pdf]

CERTIFIED LOGISTICS ON BLOCKCHAIN

Author: García Echeverría, Ainhoa

Director: García de Mata, Iñigo

Martin Martinez, Francisco

Collaborating Entity: Pontifical University of Comillas (ICAI)

PROJECT SUMMARY

Introduction

Within what is known as the fourth industrial revolution, logistics 4.0 faces many challenges including being able to manage end-to-end traceability in the supply chain. Traceability in the supply chain has evolved from being a competitive advantage to a requirement with the aim of having a stable and sustainable supply chain. The growing demand by buyers for transparency in the process, and the obligation to improve security and control from the point of view of public health gives traceability a large projection of growth in which the blockchain is presented as the perfect approach to providing that added value.

The advantages that make blockchain the solution to the problems facing logistics [1]:

- Transparency and Security: Blockchain technology includes mechanisms to ensure that records are accurate and tamper-proof. All parties involved have access to a single dataset creating a single real source of information.
- Automation and digitization: blockchain allows documents not to be altered and also allows them to be located digitally.
- Trust between different parties: the registered information is immutable, therefore the role of the intermediary changes and the network is responsible for the verifications.
- Using Smart Contracts: enables agility in the supply chain, reducing human interaction and digitizing the process.

Currently, many companies have begun to incorporate new technologies into their supply chains to have a greater record of their products, however, it is only being incorporated at certain stages of the supply chain and in very few cases encompassing end-to-end traceability, apart from the fact that very limited information is shared between the different parts of the supply chain..

The main objective of the project is to explain the process of storing external measures using Blockchain technology, developing a prototype record of traceability using Blockchain and the subsequent economic analysis along with the testing of RFID tag measurement to analyze advantages both in time and economical compared to more traditional methods.

Methodology

The project has two distinct parts: the software related to the creation of a blockchain node and the hardware part in which the Arduino reads the information of an RFID tag. For the development of the complete prototype the following essential steps have been followed:

1. Configure the hardware with the Arduino board for reading RFID codes.
2. Configure the connections between the microcontroller board and the server:
    a. To allow connections between the Arduino and the Raspberry it has been chosen an ethernet board compatible with Arduino. A fixed IP has been assigned to the board to avoid communication problems
    b. The communication protocol chosen between the Arduino and the Raspberry Pi has been UDP.
3. Configure the database on the server.
    a. The Apache web server has been installed to access the database
    b. MySQL has been installed, to later define the tables that will be filled automatically with the information read by the Arduino.
4. Configure the Blockchain node. The information available on this part of the prototype was very limited. Throughout the document, there can be found in detail all the steps and problems that the user may encounter to store the information on the Blockchain. The synthesis of the steps are as follows:
    a. Synchronize the blockchain that requires significant memory usage, and therefore, depending on the computer board being used it will be necessary to use external memories.
    b. Once with the synchronized node, create an account with Ether to carry out the transactions.
    c. Create a smart contract using Remix.
    d. The program to interact with the network will have to detail the contract address, the account from which the transaction is made, the ABI of the contract, and the address of the node. With all the details defined from the beginning, sending the details of the transaction the function within the contract that you want to execute, along with the source account of the shipment.

Results

The advantages proposed with the prototype created in this project include speed in reading labels and securely storing information.

Tests have been performed for measuring reading and writing times of RFID tags both in the MySQL database and in the Blockchain. RFID tag reading times are milliseconds, and database storage is also done in a matter of milliseconds. The total time to view reflected the information on the Blockchain network will depend on the mining speed of the block to which the transaction is sent with the information about the read code.

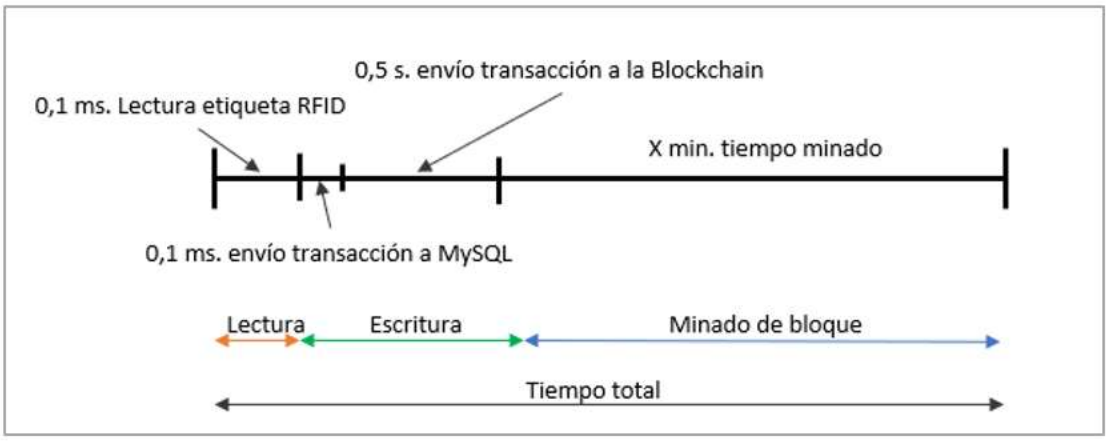


Figure 4. Test RFID tag reading and writing times.

The mining time of the block is the one that must be determined based on the costs or urgency of having the information deployed across the network. In the case of this prototype, the operations to be performed are simple transactions to send information that are an expense of 43061 gas per shipment. Different status of the network have been analyzed by classifying them as high congestion, medium and low, and three different times to display information on the network; half an hour, an hour and more than an hour. Different scenarios have also been considered depending on the number of transactions sent to the same block.

Below is the cost curve to reflect 22500 products by reflecting them with a 30-minute urgency. As observed in view of network congestion the price can increase by up to 580% from 5701USD to 982.5USD in the most favorable case.

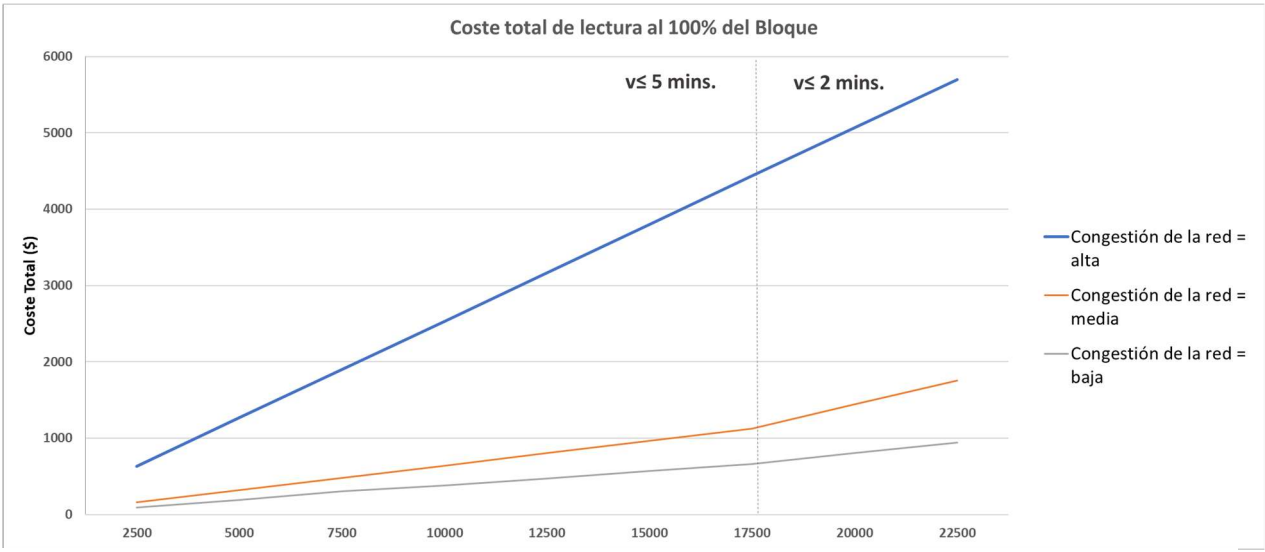


Figure 5 Cost per 22500 products depending on the network.

If we increase the number of products to be stored to the maximum available per hour for the optimistic case, the costs based on the urgency of mining and network congestion are as follows:

	High	Standard	Low
1 hour	$11,403.45	$2,411.93	$1253.65
+1 hours	$11,403.45	$760.5	$627.3

Table 3. The maximum cost of transactions at 100% of the Block.

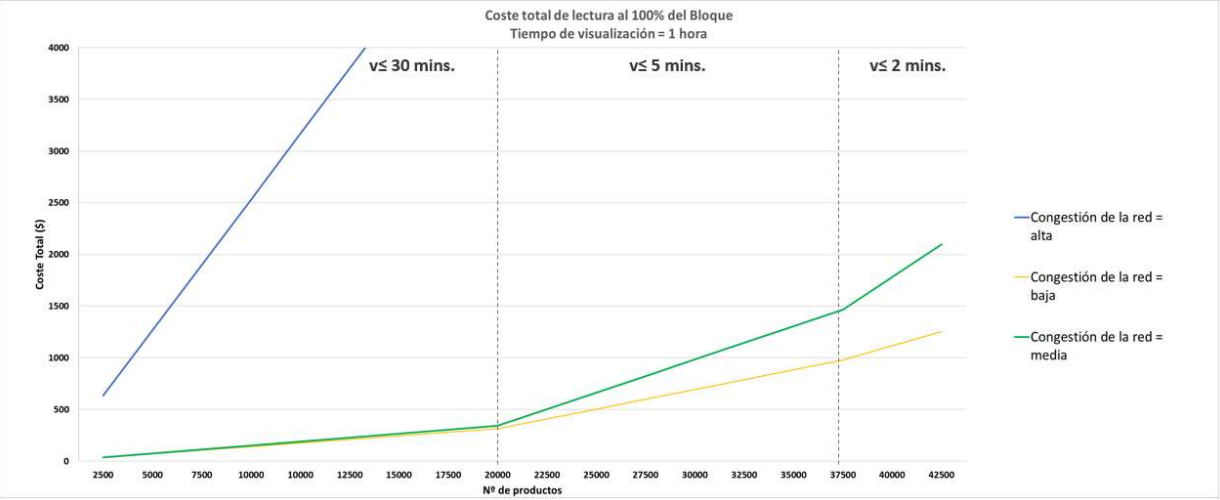


Figure 3. Cost per transaction depending on the network status.

When you want the reflected information within 30 minutes, the costs are high, since it forces you to always mine at an average or low speed. With a margin of one hour or more, it allows the first blocks to be mined at low speed (30 minutes), and costs are significantly reduced. This can be easily seen if compared figure 2 and 3, on the first one there's only one slope change between blocks mined at medium speed and blocks mined at the fastest speed. However, if there is no rush to have available the information products can be mined at the lowest speed, and if in Figure 3, there are two changes of slope.

If you are looking for a more conservative scenario in which a block cannot store all transactions from the same account, it decreases the number of products to be processed in an hour and therefore the costs. From 45000 products is reduced to 36000 products per hour.

	High	Standard	Low
1 hour	$8869.35	$1301.02	$970.45
+ 1 hours	$8869.35	$532.35	$487.5

Table 4. Maximum cost of transactions at the time at 80% of the Block.

Conclusions

The conclusions obtained after testing show the feasibility of such prototypes for the logistics industry. However, the computational cost associated with sending transactions individually to the network may seem high.

This is because, although the consumption of gas linked to the shipment of RFID cards is low, if applied to the production control of large factories lines, the volume of products to be processed is so high that the associated cost increases. It is important that information is not needed in the network immediately but throughout the day of production so that costs are significantly reduced. In addition, in order to reduce the expense, it is important to consider those times when the network is less saturated, since the rate increases by up to 500%.

On the other hand, if you see many applications where the associated cost would be lower since it would require fewer daily transactions, for example, shipping product localization at certain time intervals. In the current prototype would be to change the RFID sensor, by one location or another temperature.

For future developments, gas can be optimized in the blockchain writing to reduce these costs, as well as using more specific resources for functions.

References

[1] DHL trend research, "Blockchain in logistics". Available in: [https://www.logistics.dhl/content/dam/dhl/global/core/documents/pdf/glo-coreblockchain-trend-report.pdf]

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

LOGÍSTICA CERTIFICADA EN BLOCKCHAIN

Autor: Ainhoa García Echeverría

Director: Francisco Martín Martinez

Iñigo García de Mata

Madrid

Julio de 2019

Contenido

Índice de figuras

Índice de tablas

Introducción.

Dentro de lo que se conoce como la cuarta revolución industrial, la logística 4.0 se enfrenta entre otros muchos retos al de ser capaz de gestionar la trazabilidad de extremo a extremo en la cadena de suministro. Organizaciones como la FAO (Organización de las Naciones Unidas para la Alimentación y la Agricultura) y la OMS (Organización Mundial de la Salud) ya han indicado que la trazabilidad debería ser un elemento fundamental y regulado en todos los países. AECOC (La Asociación española de Fabricantes y Distribuidores) define trazabilidad como "el conjunto de procedimientos preestablecidos y autosuficientes que permiten conocer el histórico, la ubicación y la trayectoria de un producto o lote de productos a lo largo de la cadena de suministro, en un momento dado y a través de herramientas determinadas" [1]

La trazabilidad en la cadena de suministro se ha convertido de ser una ventaja competitiva a prácticamente un requisito con el objetivo de tener una cadena de suministro estable y sostenible. La exigencia creciente por parte de los compradores de la transparencia en el proceso, y la obligación de mejorar la seguridad y control desde el punto de vista de la salud pública, aporta a la trazabilidad una gran proyección de crecimiento en la que el blockchain se presenta como el enfoque perfecto para proporcionar ese valor añadido.

Ilustración 6.Esquema del concepto de trazabilidad.

Image from: R&B Traceability. Available at: http://rbrastreabilidade.com.br/en/traceability/

Los consumidores piden cada vez una mayor transparencia y controles en la cadena alimentaria ante posibles intoxicaciones o para conocer el origen de los productos. En otras industrias como la industria farmacéutica la trazabilidad es importante para garantizar la protección de los pacientes, detectando anomalías durante la fabricación. impidiendo productos falsificados en el mercado y permite retirar producto de manera efectiva ante problemas de calidad.

Las empresas apuestan en esta revolución de Industria 4.0 para digitalizar el entorno y así optimizar las operaciones. Con la digitalización son capaces de aprovechar las tecnologías más

desarrolladas en la Industria cómo inteligencia artificial, robótica o blockchain para aplicarlas al entorno de las operaciones permitiendo [1]:

- Cadenas de suministro más eficientes y eficaces
- Una mejor calidad de los productos finales para los consumidores
- Tomar decisiones a tiempo real de la cadena.
- Asegurar de manera eficiente la entrega de productos.
- Reducir costes y recursos.

La llegada de Internet permitió crear una red mundial para el intercambio y movimiento de información, sin embargo, la falta de fiabilidad y confianza entre las diferentes partes involucradas ha ido aumentando con el tiempo y es dónde la tecnología Blockchain puede incorporar sus características clave para definir la siguiente era de Internet. Aunque aún es incipiente, ya son muchos los sectores que lo han empezado a incluir [8]:

- Servicios públicos: blockchain ofrece sistemas seguros y facilita transacciones fiables permitiendo que la comunidad con DNI digitales tenga mejor acceso a diferentes actividades, cómo votar, seguros médicos, bancos, etc. Los gobiernos pueden tener estos sistemas de identificación encriptados para poder dar a sus ciudadanos estos servicios, y eliminar la falsificación de documentos.
- Venta al por menor: los minoristas y las grandes empresas de consumo utilizan la tecnología para poder proporcionar a sus consumidores de información cómo de dónde viene el producto, logrando una cadena de suministro más transparente.
- Sanidad: la mayor ventaja que ofrece el blockchain en este caso es el de proporcionar una red totalmente segura y fiable.
- Industria energética: los usos que podría ofrecer la tecnología blockchain son tales como facilitar el intercambio de energía entre particulares o gestionar las redes de autoconsumo.

Las ventajas que hacen del blockchain la solución perfecta para los problemas a los que se enfrenta la logística: [8]

- **Transparencia y Seguridad**: la tecnología blockchain incluye mecanismos para asegurar que los registros almacenados son precisos y a prueba de manipulaciones, y además de una fuente verificable. De esta manera, todas las partes implicadas tienen acceso a un único conjunto de datos creando una única fuente real de información, y no copias de cada una de las partes modificadas y alteradas. Todos los que trabajan con la información pueden estar seguros de que es la información no sólo más reciente sino además fiable. Toda transacción individual o mensaje que se envíe a través de esta red puede encriptarse reduciendo el riesgo de hackeos o manipulación de data. Al tratarse de una red distribuida sería necesario el hackeo de más del 50% para poder modificar el consenso de verificación aprobado por las partes.

- **Automatización y digitalización** para evitar errores en la gestión de la información y, reducir la complejidad existente en las cadenas de suministro actuales: blockchain permite que los documentos no puedan alterarse y además permite que sean localizados de manera digital. Esta tecnología permite la trazabilidad tanto a nivel artículo como para contenedores o pallets a lo largo de toda la cadena de suministro.
- **Confianza entre las distintas partes:** gracias a que la información se registra en la red de manera eficiente e inmutable, no es necesario un intermediario para cotejar su validez. El papel del intermediario cambia y ahora es la propia red la que se encarga de verificar.
- **El uso de Smart Contracts** permite una mayor agilidad en la cadena, reduciendo las interacciones humanas y digitalizando el proceso para mejorar tanto la trazabilidad como el seguimiento de la carga.

Por tanto, muchas de las ineficiencias que presenta la industria de la logística actualmente pueden ser solventadas gracias a la tecnología Blockchain o DLT (Distributed Ledger Technology o en español Tecnología de Contabilidad Distribuida en español). Esta industria crece exponencialmente año tras año y se espera que para 2023 su mercado alcance hasta los 15,5 billones de dólares, este crecimiento está ligado también al esperado crecimiento de la tecnología Blockchain orientada a la cadena de suministro, que se prevé que crezca en un 49% anualmente hasta 2024 cuando el mercado sea de 667 millones de dólares [30].

Es por todo esto, que las principales empresas logísticas del mundo han comenzado a explotar estas tecnologías y todos sus casos de uso. Con este proyecto se busca explicar el proceso de almacenamiento de medidas externas mediante la tecnología Blockchain, desarrollando un sistema de lectura de productos mediante tecnología RFID con envío cifrado a un servidor para su posterior registro en la Blockchain que simule un lector dentro de una cadena de suministro.

ESTADO DEL ARTE

LOGÍSTICA, TRAZABILIDAD Y GESTIÓN DE ACTIVOS

Investigadores de la mano de McKinsey han intentado evaluar la importancia estratégica de Blockchain en la mayoría de las industrias para determinar en qué casos el uso de la Blockchain tiene más oportunidades.

Así las oportunidades que ofrecen por sector son las siguientes:

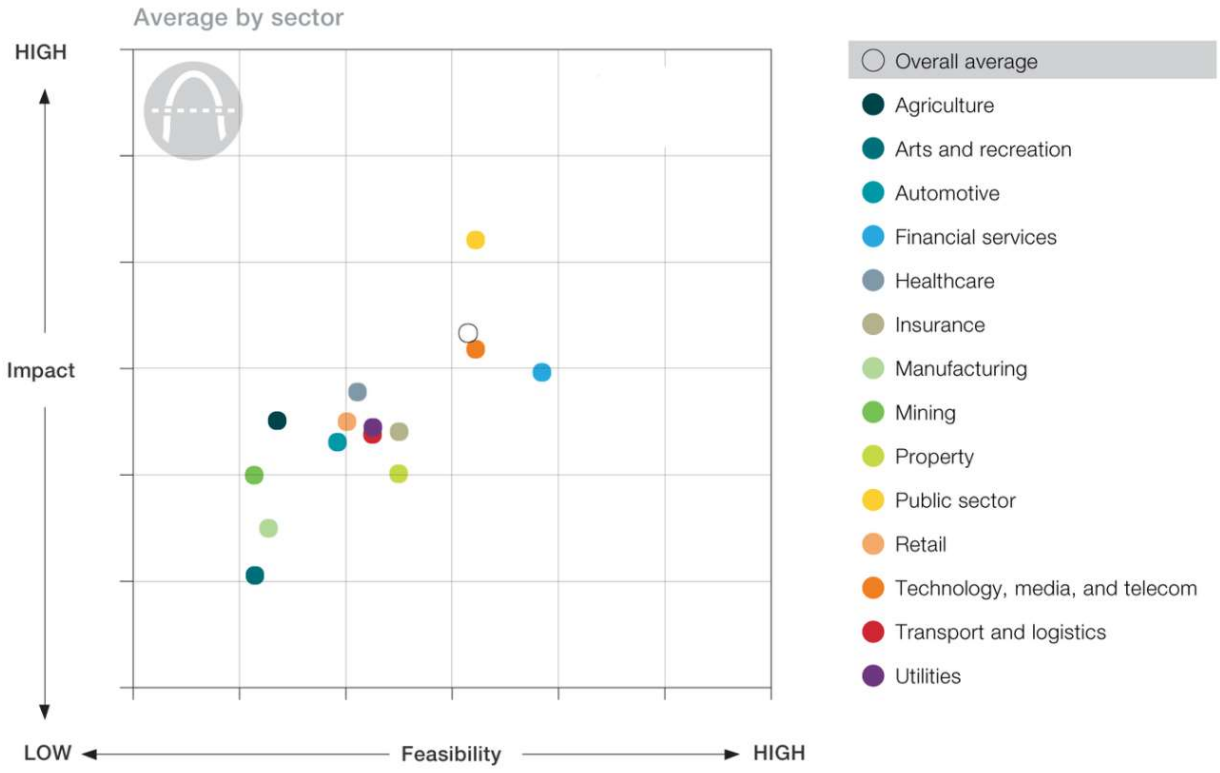


Ilustración 7. Oportunidades de Blockchain por industria.

Fuente: McKinsey&Company.

En el caso de la distribución, el transporte y la logística son muchos los casos de uso y la viabilidad es bastante alta. En el caso de la distribución, el principal uso estaría enfocado en la verificación de los productos y el segundo caso con mayor impacto sería el de conseguir información sobre el punto de venta. Por otro lado, para el sector del transporte el caso de uso con mayor valor estratégico está orientado a obtener información durante el envío.

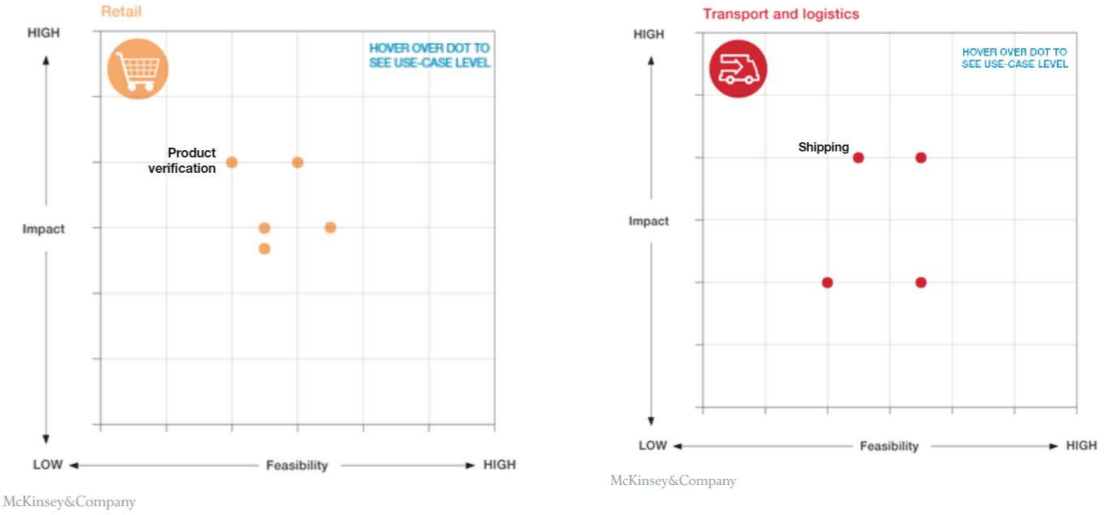


Ilustración 8. Casos de uso de Blockchain para transporte y distribución.

Fuente: McKinsey. "Blockchain beyond the hype: What is the strategic business value?

Según el informe publicado por la consultora McKinsey, la tecnología Blockchain tiene el potencial disruptivo para ser la base de los modelos operativos en el futuro, aunque el impacto inicial en todas las industrias es el de fomentar la eficiencia operativa. Los costes pueden reducirse eliminando de los procesos existentes los intermediarios, o el esfuerzo administrativo actúa para mantener los registros y la conciliación de las transacciones.

La clave del trabajo que se presenta en este proyecto es el de mejorar los problemas existentes en la actualidad en la logística y asegurar una mejor trazabilidad del producto a lo largo de la cadena de suministro. Primero, empezaremos por entender el concepto de logística. La logística corresponde a la función de planificar, implementar y controlar el flujo eficiente y eficaz de servicios, información y bienes entre el punto de origen y el de consumo [2]. La logística tiene centradas sus actividades fundamentalmente, en la gestión del transporte, de proveedores y de materiales, del almacenamiento y gestión del inventario para cumplir con los pedidos y de diseñar las redes logísticas.

La trazabilidad del producto según AECOC es el conjunto de medidas y procedimientos que permiten registrar e identificar cada producto en cualquier momento en su cadena de suministro. Gracias a ella se puede obtener una mejor gestión de la logística, puesto que permite tomar medidas a tiempo real tanto preventivas como correctivas, asegurando una mejor calidad del servicio aparte de optimizar costes. Asegurar la transparencia a lo largo de la cadena de suministro es fundamental para asegurar una buena trazabilidad del producto y poder superar los problemas actuales a los que se enfrenta.

Es importante distinguir los dos problemas generales a los que las cadenas de suministro se enfrentan actualmente y para los que el blockchain es una solución; por un lado, está la trazabilidad que es la posibilidad de que el consumidor desde fuera pueda conocer el origen del

producto, y por otro lado el seguimiento que está más relacionado con la visión interna dentro de la empresa. Para ambos problemas cómo se explicará más adelante el blockchain y la transparencia que este permite es una buena solución.

Dentro de los problemas que pueden solucionarse utilizando Blockchain se engloban los siguientes:

1. **Errores en la gestión de la información:** son muchas las operaciones que se llevan a cabo en los almacenes, como entradas y salidas de productos, movimientos internos, inventarios que deben ser perfectamente documentados, operaciones que incluyen actividad humana generando el riesgo de información incorrecta o manipulada. Para ello son necesarios sistemas de lectura no sólo fijos sino móviles. [3]
2. **Complejidad de la cadena de suministro:** el riesgo al que se enfrentan las cadenas de suministro globales es a la falta de visibilidad, son tantos los agentes que intervienen, tanto proveedores, como distribuidores, consumidores que se exponen a mayores riesgos, tanto de fraude cómo de cualquier tipo de infracción. [4]
3. **La falta visibilidad actual**: Una mayor visibilidad es requisito para poder conocer todos los movimientos del producto y así poder elaborar respuestas rápidas en cualquier momento ante cualquier impacto que pueda sufrir la cadena de suministro. [4]

Ilustración 9. Complejidad flujos de información en la cadena de suministro.

Image from: DHL trend research, Blockchain in Logistics [8].

4. **Falta de confianza entre las distintas partes implicadas en la cadena [4]:** para que el flujo e intercambio de información sea posible es necesario que exista una relación de confianza entre cada una de las partes implicadas en la cadena, este problema es posible mitigarlo mediante la red de Blockchain y los Smart Contracts. Además, es necesario que la cadena de suministro en cada una de sus etapas tenga la capacidad de recopilar la información de manera segura y exacta.
5. **Obsolescencia de las tecnologías [4]:** las cadenas de suministro requieren de un inversión continua en sus herramientas. La implementación de estás suele suponer un gran desafío para muchas de las empresas.
6. **Falta de digitalización y de sistemas de soporte para la tecnología [5]:** es necesario que la cadena de suministro esté completamente digitalizada para poder cumplir con la trazabilidad asegurando además la operabilidad entre los diferentes

sistemas de la cadena, para minimizar la actividad humana. Tecnología como la Blockchain abre una enorme salida a este problema, además a un bajo coste.

CONCEPTOS DE BLOCKCHAIN

La tecnología blockchain se está explorando actualmente en todos los sectores para transformar y solucionar los problemas de los modelos de negocios existentes. En el ámbito de la logística se encuentra aún en una fase inicial, pero ya es visible las potenciales ventajas que ofrece al sector.

Para entender que puede aportar *el Blockchain* al sector logístico es necesario entender en que consiste esta tecnología. *El blockchain* consiste en una cadena de bloques que permite la colaboración en un registro base distribuido, inmutable y público, garantizando de esta forma que todas las transacciones sean seguras y permitiendo que exista transparencia a lo largo del proceso de esta [6]. La transparencia que aporta la red de blockchain permite proporcionar una fuente única de verdad permitiendo que a partes sin ningún tipo de relación puedan confiar en la información. La red consiste en un conjunto de nodos conectados entre sí con el fin de validar y almacenar la misma información en toda la red creando así el "libro mayor de contabilidad".

El funcionamiento de la red es el siguiente; un usuario de la red envía una transacción, la red de nodos valida esta transacción que puede involucrar tanto criptodivisas, cómo Smart Contracts, archivos o cualquier otro tipo de información, una vez validada se crea un nuevo bloque de datos que se incluye en el Blockchain ya de manera permanente e inalterable [7].La blockchain se trata de una red pasiva en la que el usuario envía cosas y esta devuelve resultados.

En la blockchain se intercambian tokens, donde token se define como la representación digital de una unidad de valor. Según la Autoridad Supervisora del Mercado Financiero Suizo, los tokens pueden tener 3 funciones: tokens de pago (monedas criptográficas), tokens de utilidad (productos y servicios) y tokens de activos (que representan la propiedad de los activos). Es decir, pueden utilizarse para pagar por un trabajo, cómo un incentivo, para permitir servicios extra...

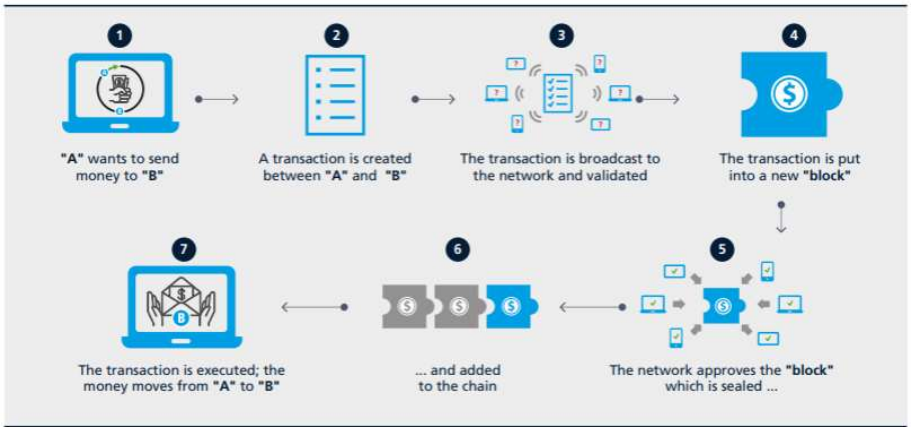


Ilustración 10. Funcionamiento del blockchain

Image from: DHL trend research, "Blockchain in logistics" [8].

Criptografía

En la red blockchain la información se envía encriptada a los nodos permitiendo únicamente acceso a los usuarios que se desee en caso de ser una red privada, y los usuarios ya podrán añadir información, pero nunca borrar la existente. Una de las mayores ventajas que proporciona el blockchain es la cronología en la que los cambios se registran en la red. Estos registros se encriptan en lo que se denomina hash o contraseña alfanumérica que referencia la relación entre los bloques, de manera que sólo los hash que sean válidos serán introducidos en la cadena y ya replicados a todos los miembros de la red. Cada registro de entrada o función hash proporciona una salida distinta. Por tanto, se trata de un proceso determinista, y siempre que se introduzca la misma información de entrada, el valor de salida será el mismo. Otra función característica de los registros hash es que se trata de conversiones de un único sentido y por tanto, no puedes revertir el hash para generar la entrada original.

La seguridad del mensaje en la red blockchain se consigue a través de la criptografía reconociendo dos tipos fundamentales [25]:

- Cifrado simétrico: se utiliza una clave tanto para cifrar cómo para descifrar la información. Es el método más sencillo y rápido al utilizar menos recursos informáticos.
- Cifrado asimétrico: en este caso se cifra la información con una clave pública y se descifra con la clave privada que posee el receptor. En este tipo de cifrados se utiliza la funcionalidad del hash para verificar la información del emisor.

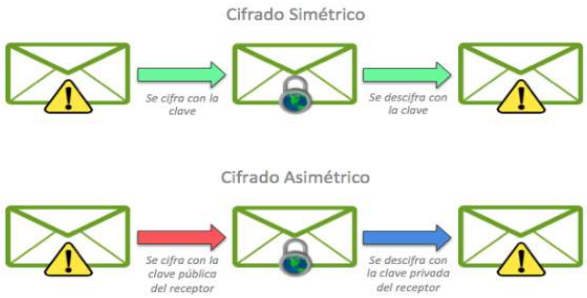


Ilustración 11. Tipos de criptografía

Image from: bigeek [25]

La red Blockchain se caracteriza también por ser inmutable y es debido a lo explicado anteriormente de los registros hash. Cada bloque dentro de la red posee la función Hash del bloque anterior, y por tanto no es posible modificar un bloque sin alterar toda la cadena.

Red distribuida

Los elementos de la red funcionan tanto como emisores como receptores respecto al resto de nodos de la red. En cada uno de ellos, existe una copa idéntica de la cadena de bloques, de esta manera se crea así la descentralización. De esta manera toda la naturaleza de la tecnología blockchain no reside en único punto de control. La falta de una única autoridad es lo que hace que el sistema sea más justo y considerablemente más seguro. Estas redes pueden ser tanto públicas como privadas, ambas son descentralizadas y los participantes poseen una réplica de toda la información sobre la que se almacena cualquier transacción. De manera que a este almacenamiento de información solo se le puede adjuntar nuevos bloques de información, pero nunca editar bloques pasados. Las redes públicas aparte de ser visibles para todos, permite que los usuarios puedan enviar transacciones. Por otro lado, las redes privadas solo permiten visibilidad y colaboración a los participantes que tengan el privilegio de hacerlo. Esta segunda opción es la que permite a las compañías mostrar su información únicamente con las partes implicadas en su ecosistema.

	Red pública Blockchain	**Red privada Blockchain**
Acceso	○ Cualquiera	○ Organización concreta
Participación	○ Anónima ○ Sin necesidad de permiso	○ Se conoce la identidad ○ Se requiere permiso

	Red pública Blockchain	Red privada Blockchain
Seguridad	○ Mecanismo de consenso ○ Prueba de trabajo/Prueba de participación	○ Participantes con aprobación previa. ○ Votaciones/ Sistema de consenso
Desempeño	○ Transacciones de baja velocidad	○ Red más ligera ○ Transacciones más rápidas.

Tabla 5. Características redes privadas y públicas en la red de Blockchain.

Mecanismo de consenso

Existe un **mecanismo de consenso** con el cual los diferentes nodos de la red se ponen de acuerdo con el fin de determinar qué información es veraz y cuál falsa. No solo verifica información, sino que determina el orden con el que suceden las cosas. Lo que se conoce como "marca temporal" es una secuencia de caracteres e información codificada que identifica cuando un cierto evento ha ocurrido.

El POW (Proof of Work), Prueba de Trabajo, es uno de los primeros procesos de validación y el primero usado por Bitcoin. Los usuarios a través de la red se envían tokens y la base de datos descentralizada se encarga de agrupar todas los movimientos, confirmar las transacciones y ordenarlas en bloques. A este proceso se le conoce como el de minar, y los nodos son los mineros. Los mineros se encargan de resolver complejos puzles matemáticos que requieren de alta potencia computacional, tanto que en los dos últimos años se estima que Bitcoin ha consumido 73.12 TWh lo que ha supuesto un coste de $3,656M (Según: "BitcoinEnergyConsumption.com"). La respuesta al problema de POW o ecuación matemática que se plantea a los mineros es el denominado Hash. Una vez se logra resolver se forma el bloque y de esta manera las transacciones se consideran confirmadas. A medida que la red crece, los algoritmos requieren cada vez de mayor potencia y por tanto la complejidad aumenta.

El bloque creado contiene la siguiente información:

- Hash: se trata de la función dónde queda registrada toda la información y que permite así una fácil verificación, pero impide reproducir la información de entrada. Estos apuntadores hash enlazan el bloque con el anterior y así hasta llegar al bloque génesis.
- Transacciones: el bloque contiene información también de las transacciones pasadas, y se agrupa con las anteriores formando lo que se conoce como estructura *Merkle Tree*.
- Nonce: se trata de un número único aleatorio emitido por los mineros a través del POW, sirve para verificar el bloque y evitar que la data se cambie sin realizar todo el trabajo nuevamente.

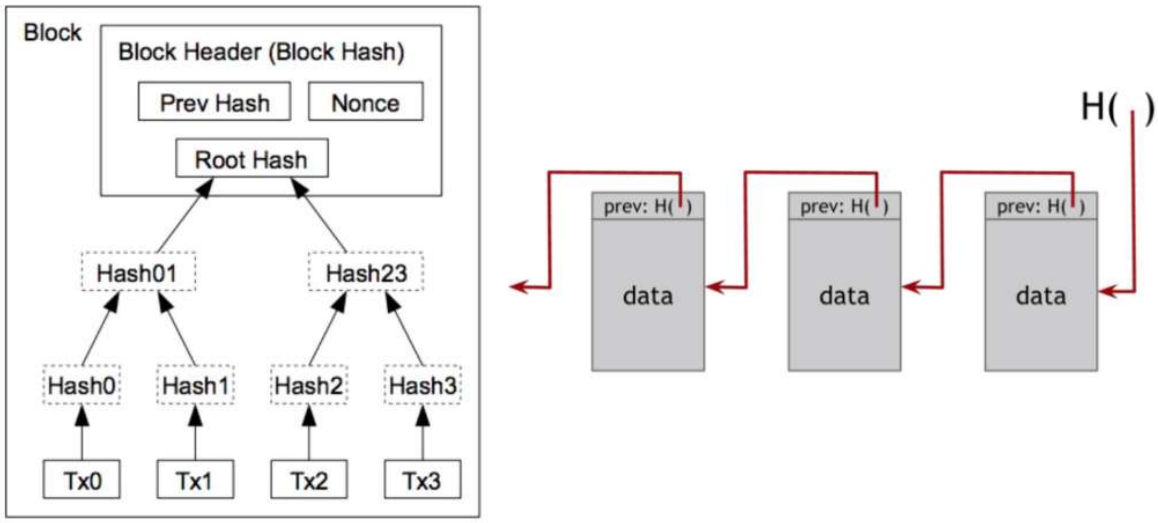


Ilustración 12. Información en un bloque en la blockchain.

Proceso de validación

Proceso de validación es el mecanismo por el cual un número representativo de nodos acredita que el resultado obtenido en la prueba de trabajo por el nodo ganador es correcto. Actualmente con el fin de motivar el comportamiento leal a la hora de validación de bloques, cada bloque validado supone 25 bitcoins. Aunque la tarea complicada consiste en encontrar cual es el Nonce que proporciona el hash en concreto, el proceso inverso es fácil. Los nodos comprueban de manera fácil que el resultado que el nodo calcula es correcto, y así, es como se valida un nuevo bloque [25].

Smart Contract

Los **Smart Contracts** son contratos autoejecutables en formato electrónico. Estos contratos se reparten entre los distintos usuarios de la red y lo verifican mediante unos mecanismos que poseen siempre que se cumplan unas determinadas condiciones. El término autoejecutable se debe a que no es necesario una tercera persona de confianza [6]. Se tratan de cualquier tipo de acuerdo en el que una vez se hayan formalizado todas o la mayor parte de las cláusulas mediante programas, una vez el acuerdo está finalizado y habiendo determinado uno varios eventos desencadenantes, si estos llegan a producirse se ejecutará de manera automática el contrato en su totalidad, sin la posibilidad de que sea modificado o bloqueado.

El acuerdo puede estar escrito a mano, pero una parte tendrá asociada un código de programación sobre el que se definen las reglas y las consecuencias de estas, como cualquier otro tipo de contrato con la diferencia que una vez se de el evento desencadenante, el mecanismo se ejecuta sin escuchar la voluntad de las partes, sino que actuará en función a las reglas ya

definidas anteriormente. La mayor ventaja de estos contratos autoejecutables es que una vez pasada la complejidad de la programación inicial, operan de manera rápida y sencilla y son inmodificables para no permitir el arrepentimiento.

POR QUÉ UTILIZAR BLOCKCHAIN EN LA CADENA DE SUMINISTRO

Son muchos los beneficios que Blockchain puede aportar a la gestión de la cadena de suministro y a las compañías para mejorar su transparencia y su responsabilidad de cuentas entre distribuidores, transacciones inmediatas gracias a los contratos inteligentes, etc.; y reducir el error humano a lo largo de la cadena gracias a una mejor automatización.[29]

1. **Transacciones más rápidas y baratas:** las inversiones de las compañías para acelerar sus transacciones son continuas. Por ejemplo, en 1999, GE ahorró casi $2 billones de dólares americanos cuando digitalizó sus medios de pago. En las fronteras los pagos también implican un importante consumo de tiempo, anualmente suelen ser hasta 1.5 trillones de dólares americanos. Todo esto demuestra que mejorar los procesos financieros conlleva a recompensas tangibles y substanciales, y que, por el contrario, ignorarlas implica un gran coste para toda la cadena de suministro. Utilizando Blockchain se puede permitir el pago entre dos entidades sin necesidad de un tercero, ejecutando ambos lados de la transacción de manera simultánea. Con menores tasas por evitar intermediarios y el bajo coste computacional añaden
2. **Auditorías más transparentes:** una buena auditoría depende en grande medida del auditor para demostrar y examinar la cadena de suministro con las herramientas adecuadas. La red de Blockchain es interoperable, y si se aplicase a todos los participantes de la cadena de suministro, se podría sacar dónde residen las mayores ineficiencias mientas se mantiene el anonimato de cada una de las cuentas. Esta combinación de accesibilidad y transparencia, y anonimato permite a los auditores hacer su trabajo de manera económica y comprensible, y Blockchain sirve como herramienta para identificar ineficiencias sin interrumpir en el flujo de procesos diarios de la cadena de suministro.
3. **Permite rastrear el impacto medioambiental:** 86% de los consumidores quieren que las compañías intervengan en los problemas medioambientales y sociales. Y no solo eso, el 66% de los consumidores y el 73% de los *millenials* pagarían más por un producto si la compañía por detrás se tratase de una marca sostenible. Blockchain permite rastrear los productos a lo largo de toda la cadena y ofrece un registro de fácil acceso para cada producto de manera que se pueda demostrar que en el camino del producto hasta la estantería no hubo explotación infantil o cualquier otro problema ético.
4. **Información más precisa y utilizable sobre el cálculo de costes**: el 39% de los managers de la cadena de suministro afirman que sin un buen software y tecnología en los procesos existe una barrera para obtener información sobre costes, creando

resistencia por parte de los departamentos de contabilidad y finanzas para cambiar sistemas. Con blockchain se puede solucionar siempre y cuando todas las partes que intervengan estén preparadas para este cambio. Si un registro compartido sobre los costes puede almacenarse y es accesible por todos, proporciona a los managers el tipo de tecnología que buscan para su cadena, mientras que también sirve como la ruptura de los sistemas de registros financieros externos a uno interoperable al cual todas las partes necesarias pueden acceder y supervisar.

5. **Mejor información sobre los pedidos:** en las encuestas a managers de la cadena de suministro se obtienen siempre un resultado en torno al 90% en el que afirman que tener acceso a información en tiempo real y mejores sistemas para compartir datos es indispensable. Por las características de la tecnología Blockchain y su proceso descentralizado, proporciona un modo uniforme de registro que es interoperable pro todos los participantes de la cadena de suministro y además se actualiza en tiempo real.
6. **Prevenir infracciones**: el costo de cumplimiento en los envíos es alto y cumplir con la linea suele ser complicado, pero ignorar los costes puede resultar en multas muy elevadas. Por tanto, cuantos más datos tengan los gerentes de las cadenas de suministro y más fácilmente puedan ver lo que sus proveedores lo están haciendo, es más probable que eliminen las infracciones de cumplimiento y, por tanto, eviten las multas. El Blockchain por ser descentralizado e interoperable permite que los gerentes no tengan solo lo que desean sino lo que necesitan para evitar multas e incluso tiempo en la cárcel.
7. **Origen/Proveniencia**: la tecnología Blockchain permite almacenar/grabar cada una de las transacciones de manera permanente, ya sea información financiera de intercambio de productos, lo que aporta innumerables beneficios. Ya sea por razones éticas, temor a productos falsificados o apoyar a marcas nacionales, los consumidores exigen cada vez un conocimiento mayor del origen de lo que consumen [33].
8. **Reducir el error humano:** entre $50,000 y $150,000 es lo que se estima que de manera anual pierde una empresa debido a errores de facturación. Aquellas compañías que reportan errores en su cadena de suministro reportan de manera anual un aumento de los costes de 11%, un 14% extra de inventario, y hasta una disminución de ventas de un 7%, lo que demuestra lo importante que es minimizar los errores humanos en los procesos de la cadena. Las plataformas de gestión con tecnología Blockchain están diseñadas para reducir la dependencia en los humanos, y mantener los registros, la entrada de datos y los sistemas de seguimiento de inventario de una manera que sea más rápida y asequible.
9. **Automatización**: un estudio realizado con 337 ejecutivos de empresas de producción globales afirman que el 33% están insatisfechos con la transformación digital de su propia cadena de suministro , y el 94% de los mismos afirman que herramientas que faciliten la visibilidad en la cadena son claves para el éxito. Todo está conectado, la visibilidad en la cadena está relacionado con la productividad, y la automatización permite a los managers una mayor velocidad de los datos lo que implica una mayor

visibilidad. Blockchain es una de las formas más seguras, avanzadas y seguras para automatizar procesos de la cadena de suministro y es por eso que tiene un papel central para automatizar transacciones y disponer dinero efectivo en cualquier momento.

10. **Automatización para compras y planificación**: una empresa tecnológica automatizó el 95% de sus procesos de pedido reduciendo el tiempo de procesamiento de extremo a extremo en un 60%. Un procesamiento más rápido dio lugar a la capacidad de reducir inventario y coordinar elementos de la cadena de suministro con mayor precisión, gracias a la mayor información que tenían a mano comparado con antes. Hay una buena razón por la que el 66% de los gerentes de la cadena de suministro claman por análisis avanzados de la cadena de suministro: en un entorno de consumo donde los pedidos deben llenarse casi al instante y la vida útil del producto sigue siendo baja, los análisis equivalen a importantes ahorros importantes. Es por eso qué el 44% de los encuestados están reforzando sus sistemas de planificación de recursos empresariales (ERP) para proporcionar una mayor visibilidad de la cadena de suministro. La cadena de bloques es otro medio para unificar, y por lo tanto ampliar, el registro de la cadena de suministro, mejorando la velocidad y la profundidad con la que se comparte la información para ayudar a los gerentes a tomar decisiones de compra más informadas y que ahorran costos. [32]

DESAFÍOS DE LA TECNOLOGÍA BLOCKCHAIN

Aunque son muchas las ventajas que proporciona esta tecnología también descubre grandes retos tanto económicos, como empresariales y de sostenibilidad [27][28].

1. Desafíos para adoptar la tecnología: para poder ser agente transformador y hacer frente a las cuestiones medioambientales será necesario que las aplicaciones Blockchain se puedan escalar de manera efectiva, además debería obtener una amplia difusión en la industria y una utilización por parte del usuario inmediata. La usabilidad de la tecnología es actualmente una barrera crucial de entrada, puesto que muchas de las interfaces de la tecnología son demasiado complejas para implementarlas. Las áreas específicas de mejora incluyen mejorar la experiencia del usuario, la velocidad del sistema y la falta de protocolos formalizados. El grado en el que los usuarios confían y entienden la tecnología también puede suponer una barrera para la adopción. Por ejemplo, en el caso de los inversores interesados en el enfoque de blockchain con finanzas se enfrentan a importantes obstáculos para participar, ya que las inversiones requieren de un conocimiento alto de la tecnología.
2. Barrera tecnológica: aunque la infraestructura y los casos de uso estén aun creciendo el despliegue de redes listas sigue siendo relativamente escaso. Hay todavía una serie de desafíos técnicos con el Blockchain, y la capacidad para superarlas puede determinar el alcance de su despliegue en los próximos años. La arquitectura descentralizada de la

red, por ejemplo, requiere que en todas las aplicación de POW (Proof-of-work) cada participante en la red debe procese cada una de las transacciones y, por tanto, las aplicaciones están limitadas al tiempo necesario para procesar cada transacción. Además, a medida que crece el tamaño de la red habrá más competencia para la capacidad limitada de transacción. Actualmente, esto se traduce en tiempos de espera mayores hasta confirmar transacciones y mayores cargos puesto que los usuarios tratan de pujar más que los demás para garantizar que sus transacciones se procesen primero. Hoy en día, tanto Bitcoin como Ethereum son capaces de manejar entre 3 y 30 transacciones por segundo. Todo esto implica un problema a la hora de escalar puesto que la red es limitada. Para lograrlo serían necesarios protocolos que desarrollasen mecanismos para limitar el número de nodos participantes que requieran validar una transacción, pero sin perder la confianza en la red de que todas las transacciones son válidas.

3. Riesgos de seguridad: una de las características básicas del Blockchain es que es "extremadamente difícil de hackear" debido a su compleja criptografía y por estar distribuida. Sin embargo, cualquier sistema de IT está sujeto a riesgos de ciberseguridad, y Blockchain no es la excepción. Por su diseño, en la red entre los participantes se comparte más información que en una base de datos más tradicional, ya que en muchas ocasiones se necesita compartir la información entre nodos. Por ello para acceder a muchos de los "libros de cuentas" se necesita tanto una llave privada como una pública, y por tanto, es casi imposible acceder a los datos sin la combinación correcta de ambas llaves, lo que aporta seguridad al sistema. Sin embargo, a su vez puede ser una debilidad puesto que la clave del hacker es encontrar las llaves para acceder a la información, y estas las poseen los usuarios y se encargan de almacenarlas de manera segura, lo que puede ser peligroso teniendo en cuenta el desconocimiento de muchos usuarios de esta tecnología.
4. Desafíos legales y regulatorios: en los próximos años los especialistas se van a enfocar en arreglar las limitaciones de la red actual y expandir la red actual y, por tanto, es necesario que también se desarrolle un marco legal que funcione de manera global. [34] Actualmente. los cambios necesarios dentro del marco regulatorio y legal para poder escalar esta tecnología son: Jurisdicción "distribuida": los marcos legales actualmente están definidos por cada jurisdicción. Sin embargo, las transacciones de Blockchain no se engloban dentro de ninguna localización y cada nodo puede estar localizado en un parte del mundo distinta. Por tanto, no sé sabe bajo que jurisdicción Blockchain entra y existe el riesgo de crear complejas exigencias a las entidades que utilizan esta tecnología como solución. El problema continua en caso de llegar a tribunales cuando se tenga que decidir qué normas y leyes son las que aplican y cuál será el tribunal que decida en temas más complicados.
    - Marco legal: si toda la tecnología Blockchain llegará a estar desplegada en "Smart Contracts" es necesario que el marco jurídico que rodea la formación y

el reconocimiento de los contratos evolucione para reflejar los avances tecnológicos. Primero de todo, Blockchain tendrá que ser entendido y reconocido por la ley como inmutable y, por otro lado, las bases legales actuales de validación de contratos tendrán que reconocer que los acuerdos "Smart" son igualmente válidos y ejecutables.

- Privacidad de datos: las características de la Blockchain hacen que una vez la información se almacena en un bloque esta misma no pueda ser alterada. Esto puede suponer problemas especialmente cuando la información es personal. Este problema es particularmente importante en las redes públicas con desafíos para balancear el derecho a la privacidad de cada usuario y el concepto de una gran y abierta red.

5. Riesgos de interoperabilidad: es indispensable que la integración de la tecnología Blockchain con el resto de los sistemas IT. Teniendo en cuenta que está en los primeros años de desarrollo, lo estándares no están aún del todo definidos, y por eso la interoperabilidad entre muchas de las plataformas y otros sistemas es muy limitada. Esto plantea un desafío en el coste operacional ya que los usuarios deberán establecer modelos de datos y procesos de negocio Blockchain para incorporar los nuevos protocolos de comunicación y autenticación.
6. Desafío de consumo de energía: los procesos de validación POW (el primero con Bitcoin) se diseñaron con un consumo de energía alto y si estos sistemas se escalan sin modificarse el impacto en el medioambiente puede ser negativo. Los desarrolladores afirman que han comenzado ya a abordar este problema. Por ejemplo, las soluciones de Ethereum ya tienen protocolos de confirmación distintos y que consumen mucha menos energía. Por tanto, se espera que a medida que esta tecnología vaya madurando reducirá el consumo de energía y por tanto las oportunidades que presenta para mejorar el planeta sean mucho mayores que las limitaciones que ofrecen desde el punto de vista energético.

TRAZABILIDAD EN LA CADENA DE SUMINISTRO ACTUAL

Actualmente cómo ya se ha mencionado en el apartado anterior hay algunas empresas que han comenzado a incorporar las nuevas tecnologías en sus cadenas de suministro para tener un mayor registro de sus productos, sin embargo, solo se está incorporando en determinadas etapas de la cadena de suministro y en muy pocos casos englobando la trazabilidad de extremo a extremo, aparte que la información que comparten estas empresas aún es muy escasa.

Las aplicaciones actuales de Blockchain son las siguientes, y respecto al total solo un 3% de las soluciones se están utilizando para la parte de manufactura. Con el uso de esta tecnología, los fabricantes pueden identificar el origen de los productos, las entregas y todas las actividades de producción a lo largo del largo proceso que engloba la cadena de suministro. Esto permite que los consumidores puedan confirmar el origen de los productos que están comprando, que puede contribuir en gran medida a retroceder artículos falsificados o alimentos tergiversados.

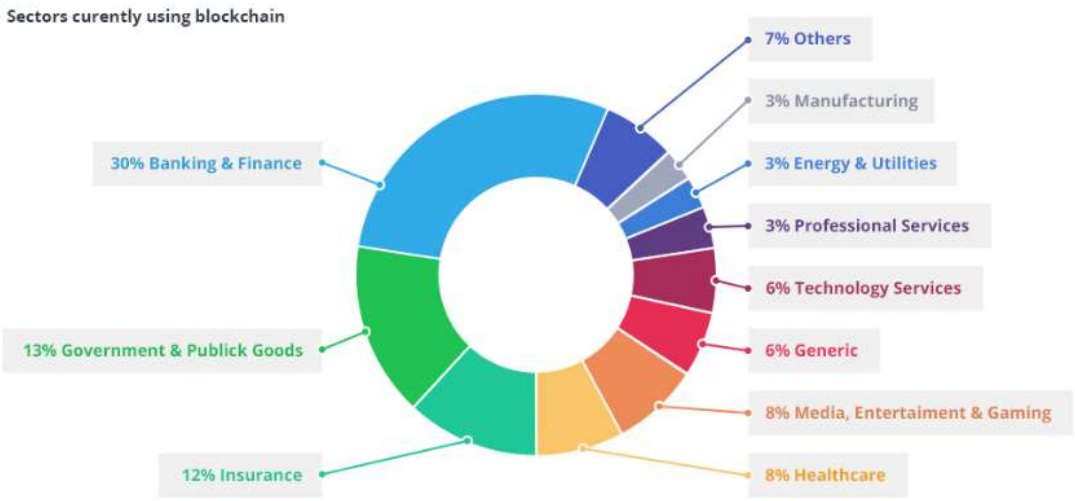


Ilustración 13. Sectores utilizando soluciones Blockchain.

Source: https://dzone.com/articles/the-future-of-the-blockchain-technology-use-cases.

A continuación, se enuncian algunos de las soluciones existentes ya en el mercado y las compañías que las están utilizando. Las soluciones existentes se pueden englobar en 5 sectores distintos:

A) Sector farmacéutico: uno de los mayores desafíos a los que se enfrenta al sector es a la falsificación de productos. Cada año, circulan 320 billones de $ en fármacos falsos:

- Blockverify: Esta empresa considera que la solución está en utilizar el blockchain, utilizando una red privada, fácil de escalar, transparente e inalterable. Cada producto tendrá un número de identificación único que se registrará en la base de datos de blockchain. [9]
- Mediledger: es un proyecto para implementar la gestión del seguimiento y rastreo de medicamentos recetados. A través de datos como el número de serie se podrá garantizar el origen y la autenticidad. La red tiene un directorio al que se accede a través de la red blockchain y permite a las compañías colaboradoras pedir información o verificar solicitudes de información. [10]
- Riddle and Code: Esta compañía ofrece servicios de logista usando Blockchain para la gestión de inventarios, identificación e integridad de productos en empresas farmacéuticas de bienes de consumo y compañías energéticas. Entre los productos que presenta, cuenta con etiquetas NFC e RFID paras sus soluciones de logística. Estas etiquetas se leen con el móvil con NFC o un lector de RFID respectivamente. Además de las etiquetas tiene una solución con un micro que almacena variables como la temperatura o humedad para garantizar la integridad del producto en el transporte.

Enviará dicha información por datos a la Blockchain, aunque no se detalla el método. [24]

B) Sector de los bienes de consumo:

- Walmart: usa tecnología RFID para identificar y conocer la trayectoria del inventario a lo largo de toda la cadena de suministro. La información se almacena en SAP. Junto a IBM está en la fase piloto de incorporar la tecnología Blockchain a su cadena de suministro, empezando con los productos frescos, para poder rastrear un producto hasta el origen de su plantación. [11]
- S-Group: es una compañía finlandesa que ha lanzado un piloto utilizando la plataforma de IBM para conocer la trayectoria de pescados frescos. Al finalizar la producción a cada lote se le asigna un código QR de manera que los consumidores puedan escanearlo y conocer toda la información que envuelve al producto (Dónde se pescó, el día, el tiempo que lleva de transporte...etc.). La captura de la información se realiza a través de la plataforma de IBM y de Hyperledger, de la Fundación Linux. [12]

C) Sector de artículos de lujo:

- Everledger: es una startup que nace con el objetivo de acabar con el fraude y el robo de diamantes, ya que existe una actividad criminal alrededor del negocio de las joyas con numerosos casos de engaño y alteración de registros oficiales, que además no hace más que aumentar a medida que crecen las compras online. La startup ha creado un registro global en el que recolecta una gran cantidad de información de referencia cruzada de los diamantes registrados, para poder proporcionar transparencia y eliminar los fraudes, ya que es inevitable que en los procesos basados en papel influya el factor humano. La tecnología detrás de esta base de datos es el blockchain, impidiendo así que cualquier impostor pueda alterar los registros de cualquiera de los diamantes del sistema. [13,14]
- Blockverify: cómo se ha mencionado en el apartado 1 de soluciones actuales, la empresa no pretende centrarse únicamente en fármacos sino también en electrónica, diamantes y productos de lujo. [9]

D) Sector del transporte:

- DS SMITH: es una empresa que fabrica embalajes a medida. DS Smith ofrece la posibilidad al cliente de realizar un seguimiento del embalaje. El cliente puede conocer en tiempo real la cantidad de embalajes que tiene y el tiempo que llevan ahí. Los embalajes se codifican con tecnología de radiofrecuencia, asignando a cada embalaje un código único, se leen en el momento de la recepción y la expedición a través de una APP móvil que transmite la lectura a una base de datos centralizada. Detrás de esta empresa hay otras como Amazon que usan sus embalajes. [15]
- Traxens: es la solución de contenedores en la red propuesta por el grupo naviero CMA CGM. Consiste en un dispositivo que se adhiere al contendor y transmite información sobre la localización del contenedor, las fluctuaciones de temperatura, los golpes que

recibe el contenedor y si se abre o se cierra la puerta del contenedor en algún momento. La información se transmite a través de una red mallada para optimizar el consumo de energía de estos dispositivos, que pueden comunicarse entre ellos. Toda la información se manda a un Gateway que a través de internet la manda a una plataforma en la nube de la cual las personas indicadas podrán obtener la información. [16,17]

- Tradelens: IBM y Maersk se han unido para crear una solución de blockchain llamada Tradelens que permite a varias empresas tener un único punto en el que poder rastrear envíos en barco en tiempo real, y autoriza a cada una de las partes involucradas a visualizar la transacción con transparencia, y no sólo de la posición del cargamento sino de todos los documentos de aduanas y facturas comerciales. [18,19]
- Kuovola Innovation: el proyecto nace por la falta de comunicación entre las diferentes compañías logísticas. Utiliza etiquetas RFID para comunicar los requisitos de envío entre los diferentes operadores. Permite que pujen por el envío los remitentes y se adjudica un contrato automáticamente al que mejor cumpla con el precio y las especificaciones. Todas estas transacciones quedan registradas en el blockchain y permite que se rastreen todos los envíos a lo largo de toda la cadena. [20,21]
- gZIM: se trata de una empresa israelí de transporte de contenedores con un piloto que permita digitalizar el documento de embarque que contiene información delicada como el destino, las características del producto, cantidad e información de facturación. Con este digitalización se pretende corregir los fallos actuales de información en las facturas que son alrededor del 10% del total de facturas emitidas al año, lo que supondría reducción de costes de la cadena de suministro.[31]
- Lynx International: se trata de una subsidiaria de Alibaba en la que han integrado Blockchain para poder rastrear

E) Sector de la minería:

- BHP Billiton: hasta el momento, en la minería la información se guardaba y rastreaba a través de emails, hojas de Excel que compartían y modificaban multitud de empleados y agentes públicos. En caso de extravío de muestras, no existía una fuente de información real y actualizada para tener como referencia. Esta empresa, ha desarrollado una aplicación web y móvil basada en blockchain que permite desde el momento inicial indicar la localización de cada muestra, de manera que la información queda registrada y no puede ser alterada de ninguna forma. [22].

En la siguiente tabla se muestra de manera resumida las soluciones existentes en el mercado mundial actual.

		SOLUCIONES EXISTENTES EN EL MERCADO						
		Blockverify	**Mediledger**	**WALMART**	**MAHOU**	**Bebidas y Alimentacion Getsion5sql**	**Ridde &Code**	**DS SMITH**
CARACTERÍSTICAS	Identificación de los productos	Número de identificación único	Número de serie	RFID QR	QR-escaneo de codigos	Lecturas con dispositivos de códigos de barras. Etiquetado EAN 128-GSI A cada lote se le asigna una etiqueta para su identificación.	RFID NFC	Código único de radio frecuencia
	Trasmisión de la información	-	-	-	-	-	-	A través de una app movil
	Almacenamiento de la información	Blockchain privada	Directorio en la red de blockchain Blockchain privada	QR Code (Uploaded on an e-certificate and linked to the product package via QR code) SAP	Web based	Servidor	Blockchain	Base de datos centralizada
SOLUCIÓN	Trazabilidad	SI	SI	SI	SI	SI	SI	SI
	Seguimiento	NO	SI	SI	NO	NO	SI	SI

	SOLUCIONES EXISTENTES EN EL MERCADO						**Solución propuesta por este TFM**
	TRAXENS	**EVERLEDGER**	**TRADELENS**	**S-GROUP** Pike-perch radar	**Kuovola innovation**	**BHP Billiton**	
CARACTERÍSTICAS	-	Mediante imágenes y escáneo de documentos	-	A cada lote se le asigna un código de barras.	RFID	Registro manual de datos basandose en la localización y añadiendo foto.	RFID Tag
	Red mallada de radio	-	Smartphone based sensor- GPS- Sensors		-		Ethernet
	Servidor: Plataforma de Big Data Servicios web y EDI	Blockchain con Smart Contracts	Blockchain	QR code & Hyperledger Plataforma de Blockchain de IBM	Blockchain	Aplicación WEB y móvil para la base de datos Blockchain privada	Raspberry pi (servidor) Blockchain
SOLUCIÓN	NO	SI	NO	SI	NO	SI	SI
	SI	SI	SI	SI	SI	NO	SI

Ilustración 14. Tabla resumen del estado de la cuestión.

MOTIVACIÓN DEL PROYECTO

Actualmente, son muchas las empresas que han comenzado a incorporar nuevas tecnologías en sus cadenas de suministro para tener un mayor registro de sus productos, sin embargo, solo se está incorporando en determinadas etapas de la cadena de suministro y en muy pocos casos englobando la trazabilidad de extremo a extremo, aparte que la información que comparten estas empresas aún es muy escasa. El sistema por desarrollar en este trabajo presenta las siguientes ventas competitivas con los sistemas actuales en el mercado:

1. **Trazabilidad de extremo a extremo**: el sistema permitirá tener un registro de los productos desde que salgan del almacén y se carguen en el medio de transporte elegido hasta que llegue al consumidor final. Cómo se ha dicho anteriormente, el objetivo es el de mejorar la visibilidad en toda la cadena de suministro, aumentando la confianza de los productores, transportistas y consumidores finales. De esta forma Con el dispositivo de lectura RFID que se va a desarrollar podrá colocarse en el punto de carga de productos y contabilizar sin error alguno la cantidad de producto que se introduce en el medio de transporte, siendo capaz de registrar también si algún producto se descarga antes del punto final del trayecto. Este método permite que no sea necesaria una relación de confianza entre el cliente y el proveedor, ya que la información no puede modificarse, y las facturas y pedidos se crean de acuerdo con información verídica.
2. Permite un **proceso transparente y accesible:** la información leída por el sistema RFID mandará la información cifrada para registrarla en la blockchain. Esto permitirá optimizar procesos y automatizarlos, ya que todos los usuarios involucrados en la cadena de suministro tendrán acceso a la información y permitirá conocer que mercancía se está transportando y en qué lugar se encuentra, mejorando la transparencia y la rapidez a lo largo de toda la cadena de suministro.
3. **Smart Contracts**: una vez alojada la información en la blockchain se pueden crear procesos automáticos entre las empresas. Generando contratos entre las diferentes partes de la cadena cómo bancos, aseguradoras, importadores/exportadores, proveedores, operadores logísticos sin intervención humana. Permitiría gestionar operaciones cómo las cartas de créditos, reduciendo los riesgos existentes de impagos, o los permisos de embarque de mercancías, y también permitiría la creación de albaranes y facturas.

OBJETIVOS

El objetivo principal del proyecto es explicar el proceso de almacenamiento de medidas externas mediante la tecnología Blockchain, desarrollando un prototipo de registro de trazabilidad usando Blockchain y el posterior análisis económico junto con las pruebas de medición de etiquetas RFID para analizar ventajas tanto en tiempo cómo económicas respecto a métodos más tradicionales.

1. Desarrollar el prototipo de bajo coste: se buscarán los sistemas que minimicen el coste del prototipo. El sistema estará formado por un Arduino con el cuál se realizará la lectura de RFID, y se enviará de manera cifrada a la Raspberry Pi. Se decidirá el tipo de Arduino que mejor pueda leer la información y almacenarla. Se decidirá también si la raspberry pi si se utilizará también como nodo ligero de blockchain o se empleará un servidor pasarela.
2. Introducir la tecnología Blockchain para mejorar la trazabilidad de los productos: los datos leídos por el Arduino se registrarán en la red de Blockchain, durante el proyecto se encontrará la mejor solución estudiando la viabilidad de utilizar una red Ethereum sin nodo, utilizar un servidor externo para la descarga e información o programar un nodo propio en la Raspberry Pi.
3. Analizar la mejora tanto económica como en cuestión de tiempo que la automatización de este proceso supone para la cadena de suministro.

METODOLOGÍA / SOLUCIÓN DESARROLLADA

El proyecto tiene dos partes claramente diferenciadas, por un lado, el software relacionado con la creación de un nodo blockchain y la parte de hardware en el que el Arduino lea la información de una etiqueta de RFID. En la primera parte del proyecto ambas partes pueden ir desarrollándose a la vez para finalmente diseñar la conexión entre ambas. Las actividades por desarrollar son las siguientes:

1. Revisión bibliográfica: se investigará cuáles son las soluciones existentes en el mercado relacionadas con el objeto de este estudio
2. Introducción al blockchain: se investigará las diferentes posibilidades que ofrece la tecnología blockchain para el trabajo aprendiendo si fuese necesario la creación de un nodo.
3. Buscar solución para el nodo de blockchain: se decidirá si se utilizará un servidor externo para la descarga de la información, o si por el contrario se programará el nodo según las posibilidades que ofrezca cada solución.
4. Programar la lectura por RFID: la lectura de la mercancía se hará por tecnología RFID, será necesario programar el Arduino para que pueda llevar a cabo la función.

5. Conexión servidor – Arduino: conectar el Arduino con el servidor que utilicemos (raspberry pi) para mandar la información del producto y poder almacenarla
6. Desarrollo del prototipo: una vez todas las partes estén funcionando se desarrollará el prototipo
7. Pruebas: se estudiará la eficacia del prototipo desarrollado.
8. Análisis de resultados: se analizarán la rapidez de lectura, la facilidad de descarga de información de la blockchain, etc.

RECURSOS Y HERRAMIENTAS EMPLEADAS

Los recursos y herramientas que se han empleado en el desarrollo del prototipo son los siguientes:

- Arduino
- Raspberry Pi 3
- RFID tags
- Antena MFRC522
- ProtoBoard y cables Macho-Hembra.
- Chip Ethernet para Arduino
- Monitor, Ratón y Teclado
- Disco externo de memoria de 1Tb
- Disipador para la Raspberry
- Cable ethernet
- Tutoriales de Blockchain para la creación de nodos:
    - Tutorial de Medium 1: Como instalar un nodo en la raspberry . Disponible en: https://medium.com/@jochemin/nodo-completo-bitcoin-en-raspberry-pi-3-3904c29d8ce1
    - Tutorial de Medium 2: Como instalar un nodo en la raspberry. Disponible en : https://medium.com/@thelucasmoore/part-one-building-an-ethereum-node-on-a-raspberry-pi-3b-481104974cf7
    - Tutorial de Medium 3: Como instalar un nodo en la raspberry https://medium.com/@thelucasmoore/part-two-building-an-ethereum-node-on-a-raspberry-pi-3b-a5154ae7a5b9
    - Tutorial para instalar Parity, cliente de Ethereum para conectar con el nodo. Disponible en: https://wiki.parity.io/Setup
    - FreeCodeMap para sincronizar el nodo. Disponible en: https://www.freecodecamp.org/news/how-to-sync-an-ethereum-node-using-geth-and-ethereum-wallet-81423d42a583/
- Tutoriales para instalar y configurar la Raspberry Pi:

  - Instalar Apache, MySQL y PHP: https://howtoraspberrypi.com/how-to-install-web-server-raspberry-pi-lamp/
  - Instalar Noobs. Disponible en: https://raspberryparatorpes.net/instalacion/noobs-paso-a-paso-instalar-el-sistema-operativo-en-la-raspberry-pi/
- Tutoriales Python
  - Socket tutorial disponible en: https://wiki.python.org/moin/HowTo/Sockets

Hardware

A lo largo de este apartado se detalla los sistemas y recursos utilizados en el desarrollo del prototipo.

El sistema está formado por un Arduino desde el cual se realiza la lectura de las tarjetas RFID, con una antena RC522. La información leída se enviará a un servidor, que en el caso de este prototipo se trata de una Raspberry Pi 3. Se ha utilizado una placa Ethernet Shield para permitir la comunicación entre el Arduino y la Raspberry. La Raspberry, a modo de servidor, alberga el nodo de Blockchain, al cual se envía la información leída por la placa Arduino. En la Raspberry, se instalará Apache como servidor web para poder conectar con la base de datos de MySQL, donde también se almacenará la información leída a tiempo real.

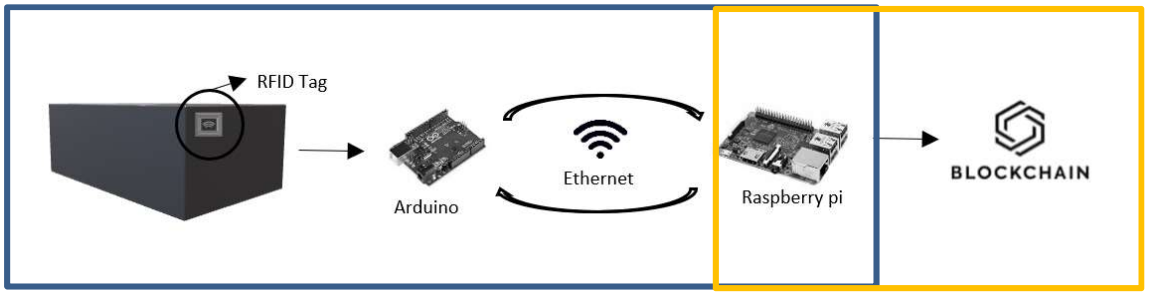


Ilustración 15. Esquema del prototipo a desarrollar en el TFM.

LECTOR RFID

El Lector RFID viene de las siglas Radio Frequency IDentification, que en español es Identificación por radiofrecuencia. Las aplicaciones como sistema de identificación son infinitas, se utiliza desde para sistemas de seguridad, para permitir accesos de personal, y sobre todo, en la logística para reconocimiento de productos e inventariado.

Su funcionamiento es sencillo, similar a los códigos de barras. el lector emite unas ondas de radio de unas frecuencias determinadas que permiten que las etiquetas al entrar en contacto con ellas envíen información sobre su ID. Las etiquetas ya tienen integradas tanto la antena como el microchip, y con las señales de radiofrecuencia obtiene la energía para comunicarse.

El módulo utilizado para este proyecto es el Módulo RFID RC522, con dos etiquetas una en forma de tarjeta y otra de llavero.

Ilustración 16. Diseño del prototipo: RC522

MFRC522 chip	
Frecuencia operativa	13.56 MHz
Voltaje de Entrada (Recomendado)	3.3 V
Corriente	13-26 mA
Rango de lectura	3cm
Velocidad de transferencia de datos	10Mbit/s
Dimensiones	60mmx39mm

Tabla 6. Especificaciones MFRC522 chip.

Fuente: www.hobbytronics.co.uk

Esta antena tiene limitado su alcance a escasos centímetros. El chip utilizado en el proyecto es de baja frecuencia y por ello, el límite de lectura a centímetros del lector. Un lector de etiquetas UHF permitiría aumentar el rango de lectura hasta 9 metros, pero sería necesario cambiar no solamente el lector pero las tarjetas, puesto que la lectores y las tarjetas no son intercambiables, y por tanto, una tarjeta LF no puede leerse con un lector UHF.

Existen lectores RFID de larga distancia, con potencias ajustables de hasta 27dBm lo que implica que pueden leer etiquetas a una distancia de hasta 5 metros con la antena correcta, y capaces de leer hasta 150 etiquetas/segundo, con precios alrededor de los 200$. Estos lectores serían necesarios para poder operar este prototipo en almacenes reales.

La conexión establecida entre el Arduino y el módulo RFID se ha realizado de la siguiente manera:

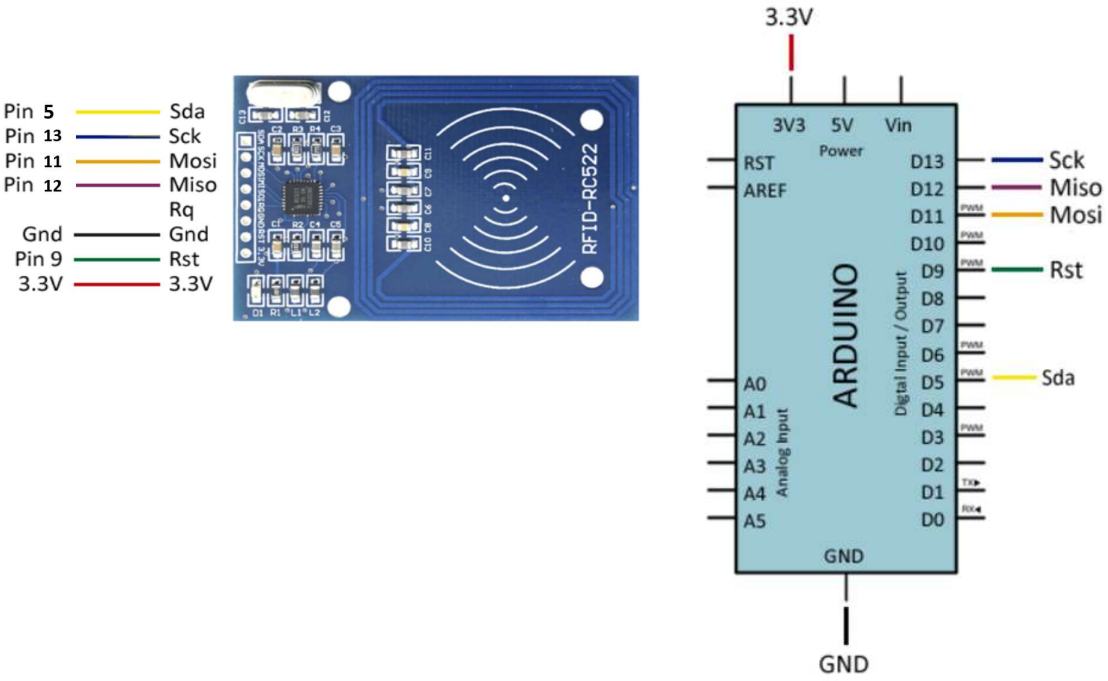


Ilustración 17. Conexión Arduino-RFID

Por encima de la placa Arduino se ha montado la placa Ethernet Shield que permite la conexión a la red local, y por tanto la conexión con el módulo RFID se realiza sobre los pines de esta. Por este motivo, puesto que el pin 10 de la placa Ethernet no puede estar ocupado para permitir la comunicación la conexión SDA se ha hecho a través de otro pin PWM, el 5.

Módulo RC522	**Arduino UNO**
SDA(SS)	5
SCK	13
MOSI	11
MISO	12
IRQ	-
GND	GND
RST	9
3.3 V	3.3 V

Tabla 7. Pines conexión Arduino-Modulo RC522.

ARDUINO

Con el prototipo se pretende simular como sería la lectura a tiempo real de productos/cajas con sus códigos de barras tanto por ejemplo de una línea de producción cómo del llenado de un camión. Para esta parte del proceso se ha elegido una tarjeta Arduino que tendrá dos funciones: por un lado, la lectura de las tarjetas y por otro la comunicación con el servidor.

El funcionamiento completo desarrollado en este trabajo simula la interacción entre una única antena lectora de tarjetas RFID y el servidor, sin embargo, a escala real serían muchas las antenas enviando información acerca de los códigos leídos o sensores midiendo temperaturas y localización.

Para ello, en la lectura de los códigos se ha asignado al Arduino una variable denominada antena, y se ha asignado el valor 1, para que el servidor pudiera diferenciar la información proveniente de cada Arduino al activar el protocolo de comunicación si se escalase el proyecto.

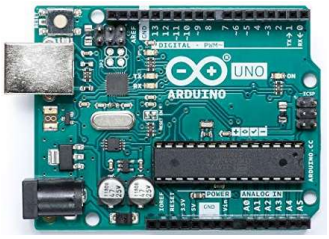


Ilustración 18. Diseño del prototipo: Arduino Uno

Para el proyecto se ha utilizado la tarjeta Arduino UNO, que consta de 14 pines tanto de entrada cómo salida digital, 6 entradas analógicas, un resonador cerámico de 16 MHz, un conector para USB, un Jack para la fuente, un conector ICSP y un botón de reset. Las especificaciones son las siguientes:

Microcontrolador	ATmega328
Voltaje Operativo	5V
Voltaje de Entrada (Recomendado)	7-12V
Pines de Entradas/Salidas Digital	14 (6 son salidas PWM)
Pines de Entradas Análogas	6
Memoria Flash	32 KB
SRAM	2 KB
EEPROM	1 KB

Velocidad del Reloj	16MHZ

Tabla 8. Especificaciones tarjeta Arduino UNO.

Para poder programar la placa y ejecutar la comunicación con el servidor se necesita tener instalado el IDE Arduino. Una vez cargado el programa final, únicamente necesita estar conectado a una fuente de alimentación externa.

Para la tarjeta microcontroladora se eligió el arduino por ser el recurso accesible, sin embargo existen otras opciones de bajo coste y bajo consumo que podrían realizar las funciones tanto de lectura de tarjetas RFID como de la conexión con la Raspberry Pi. Algunos de ellos son:

1. LaunchPad MSP430: prototipo de bajo coste de Texas Instruments con una memoria de 512 bytes (frente a los 2kB de Arduino) que funciona como buena alternativa en proyectos sencillos. El chip MSP430 tiene una función que permite el ahorro de energía y despierta de modo casi instantáneo cuando se necesita, lo que hace que sea una placa ideal para sensores que se utilicen de forma remota. El precio ronda los 4,30$.
2. STM32: se trata de otra alternativa de bajo coste. Su precio se encuentra alrededor de los 10$. Posee mayor memoria RAM con 8kB, pero el mayor inconveniente está en la limitada documentación y comunidad alrededor de este chip.
3. Teensy 2.0: trabaja con el software de Arduino, y por tanto, soporta las librerías de Arduino, lo que facilita su uso para aquellos usuarios de Arduino. La característica principal de esta tarjeta microcontroladora es su tamaño, similar al de una moneda, y un precio de unos 27$.

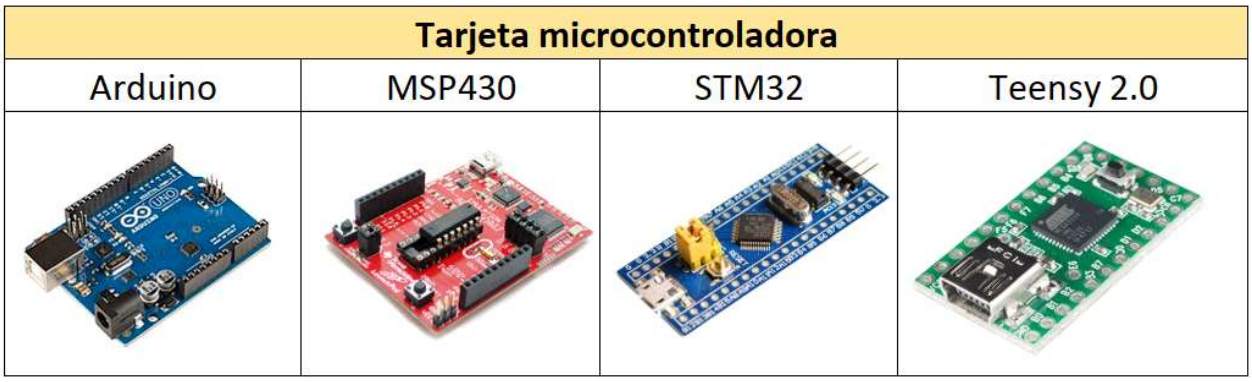


Ilustración 19. Tipos de tarjetas microcontroladoras.

ETHERNET SHIELD

El Arduino no tiene conexión por Ethernet ni Wifi para poder comunicarse con el servidor. Por tanto, se analizaron diferentes soluciones. Aquella que daba un mayor alcance era añadir una placa ethernet al Arduino.

Esta placa permite conectarse a Internet gracias a un chip Wiznet W5100 con pila de red IP capaz de TCP y UDP. Soporta hasta cuatro conexiones de sockets simultáneas. Se usa la librería Ethernet para programar. El W5100 está diseñado para facilitar la implementación de conectividad a internet. Se conecta mediante SPI, por lo que su conexión es sencilla a la mayoría de los procesadores. Algunos módulos incorporan hasta un lector de tarjeta SD, donde se pueden almacenar los ficheros en caso de trabajar como un servidor.

La conexión entre la placa Ethernet Shield y el Arduino es simplemente acoplando ambas placas.

Ilustración 20. Diseño del prototipo: Ethernet Shield

Otros dispositivos planteados:

- XBEE: se trata de un tipo de comunicación inalámbrica en la que los módulos se envían y reciben la información a través de la modulación de las ondas electromagnéticas. Por ello, para que la transmisión sea factible es necesario que ambos módulos estén a la misma frecuencia y red. Sin embargo, tenía muchas limitaciones de distancia e interoperabilidad.

Ilustración 21. Módulo de comunicación inalámbrica

- Arduino WiFI 101 Shield: se trata de una placa que permite la conexión a internet de manera inalámbrica. La conexión WiFi es tan sencilla como conectarse a una red en la que no se necesita mucha más configuración que la de añadir la contraseña y el SSID.
- Arduino MKR WAN 1300: se trata de una placa con conectividad MKR Zero y LoRa. Suele ser la solución recomendada para aquellos fabricantes diseñando proyectos de IoT con una mínima experiencia previa en las redes, y con un dispositivo de baja potencia.

RASPBERRY PI 3

Para el desarrollo del proyecto era necesario un servidor en el que poder ejecutar los diferentes programas que permitiesen la comunicación con el Arduino y almacenamiento de la información para posteriormente almacenarla tanto en una base de datos como en la red de Blockchain.

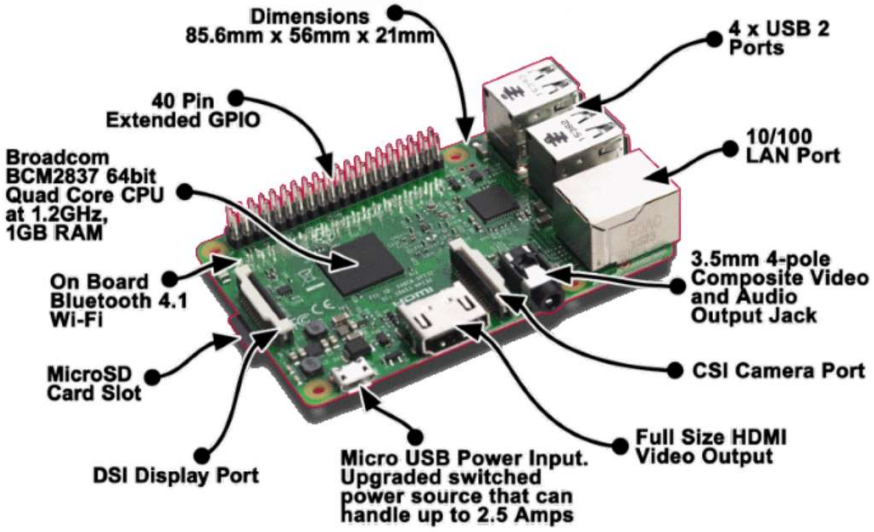


Ilustración 22. Diseño del prototipo: Raspberry pi 3

La Raspberry no deja de ser un ordenador, pero de placa pequeña. Para este proyecto se ha seleccionado la placa Raspberry Pi 3 Modelo B con procesador BCM2837 ARMv8 con 1,2 GHz de velocidad, Bluetooth, espacio para una tarjeta MicroSD y varios puertos USB. Además, a diferencia de las versiones anteriores, la placa tiene un chip BCM43143 que permite conectividad Wifi y Bluetooth de bajo consumo. El sistema operativo de la placa es una versión adaptada de Debian, denominada Raspbian, aunque puede permitir otros sistemas operativos. Raspbian ha sido el sistema utilizado para el proyecto, que se ha instalado en una tarjeta microSD de 32 GB.

La placa funciona con alimentación de 5V que, si comparamos con lo que consume cualquier otro ordenador, es un consumo mínimo, y a su vez, ocupa mucho menor espacio (85,6 mm x 53,98 mm), por lo que es perfecta para el desarrollo del prototipo.[35]

Para sincronizar el nodo de Ethereum en la Raspberry Pi, se han utilizado los siguientes elementos:

1. **Raspberry Pi3 Modelo B:** el último modelo detallado sus especificaciones en el punto 4.1.5.
2. **Un disco duro externo:** un disco duro de 1TGB de My Passport, para poder sincronizar toda la cadena de Blockchain.
3. **Micro SD:** con una capacidad de 32G en el que poder almacenar el sistema operativo, puesto que toda la información de la red estará en el disco externo.
4. **Disipador térmico:** el nodo necesita estar operando de manera continua y por tanto, necesita disipar el calor. Para poder mantener la temperatura baja, se ha utilizado uno de los puertos USB para un ventilador doble que apunte a la CPU de la Raspberry.

PRESUPUESTO PROTOTIPO

El presupuesto necesario para desarrollar este prototipo se ha dividido en la parte de lectura de los códigos, y la parte del servidor junto con los materiales necesarios para poder sincronizar la cadena de Blockchain.

El prototipo completo, suponiendo que el usuario no posea ninguno de los elementos necesarios sería de unos 228.4€. A continuación, se detallan los costes para cada una de las partes del prototipo:

MATERIAL	COSTE (€)
Arduino UNO	9,99€
Ethernet Shield	12,19€
RFID RC522	6,00€
Protoboard + Cables	4,50€

Tabla 9. Presupuesto prototipo parte lectura RFID.

MATERIAL	COSTE (€)
Raspberry Pi 3 Modelo B	35,30€
Tarjeta 32G	5,39€
Disco Duro Externo (min 500G)	44,50€
Pantalla	95€
Teclado	11€
Ratón	5€

Tabla 10. Presupuesto prototipo para el servidor.

Este presupuesto puede ser reducido si se hace un diseño propio con un microcontrolador encargado únicamente de leer el RFID y de las comunicaciones. De esta forma se disminuiría el coste para la empresa ya que se necesitarían N lectores para un solo servidor. También se podría utilizar algunas de las tarjetas microcontroladoras propuestas en el apartado de Hardware sobre el Arduino, de menor coste y con las mismas funcionalidades.

El prototipo inicial creado serviría para lectores fijos donde pueda haber acceso a Ethernet, por ejemplo, pórticos o zonas de entrada y salida de inventarios. Si se sustituyen por comunicaciones inalámbricas cuando se haga un diseño especifico se conseguiría no solo reducir el coste sino poder hacer un lector inalámbrico que podría ser usado por ejemplos en almacenes o plantaciones donde operarios lleven dicho equipamiento para leer la información de etiquetas más sofisticadas que posean medidas (Temperaturas, humedades etc..)

Software

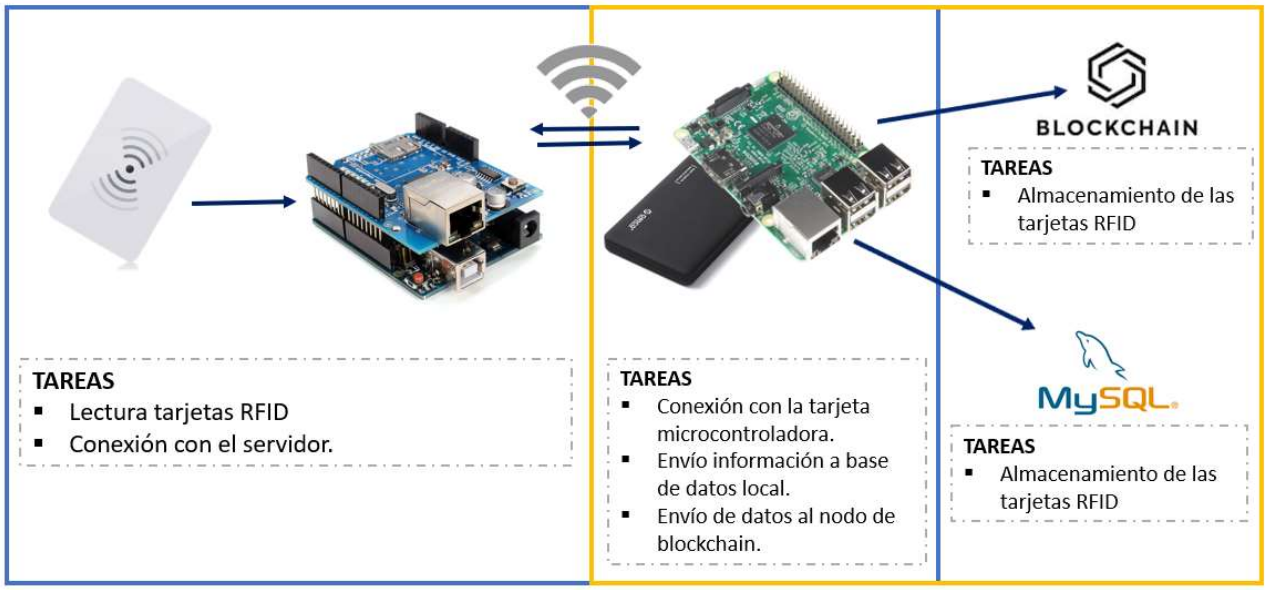


Ilustración 23. Esquema del prototipo a desarrollar en el TFM.

El esquema del prototipo desarrollado en este trabajo es el que se muestra en la *Ilustración 23. Esquema del prototipo a desarrollar en el TFM.*con dos partes diferenciadas para la lectura de las tarjetas y el posterior almacenamiento desde el servidor en una base de datos local y en la blockchain. La lectura de las tarjetas se realiza por RFID, la comunicación gracias a la placa de ethernet por wifi con el servidor, y desde la raspberry se almacena la información en la base de datos de MySQL y en la cadena sincronizada en el disco duro extraíble. En este capítulo se explicarán en detalle cómo se han realizado los códigos y en especial los pasos necesarios para establecer la red de Ethereum, desplegar contratos y almacenar medidas.

ARDUINO

El programa instalado en la placa Arduino está detallado al final del documento en el apartado de Anexos. El flujo que sigue es el indicado en la Ilustración 24. *Flujo lógico del prototipo final en el Arduino.*

Primero se configuran los puertos para leer las tarjetas inalámbricas RFID. Luego se establece la comunicación con el servidor, y una vez el servidor y el Arduino están conectados, el Arduino

estará a la espera de recibir la señal de disponibilidad del servidor para enviar los códigos RFID que vaya detectando.

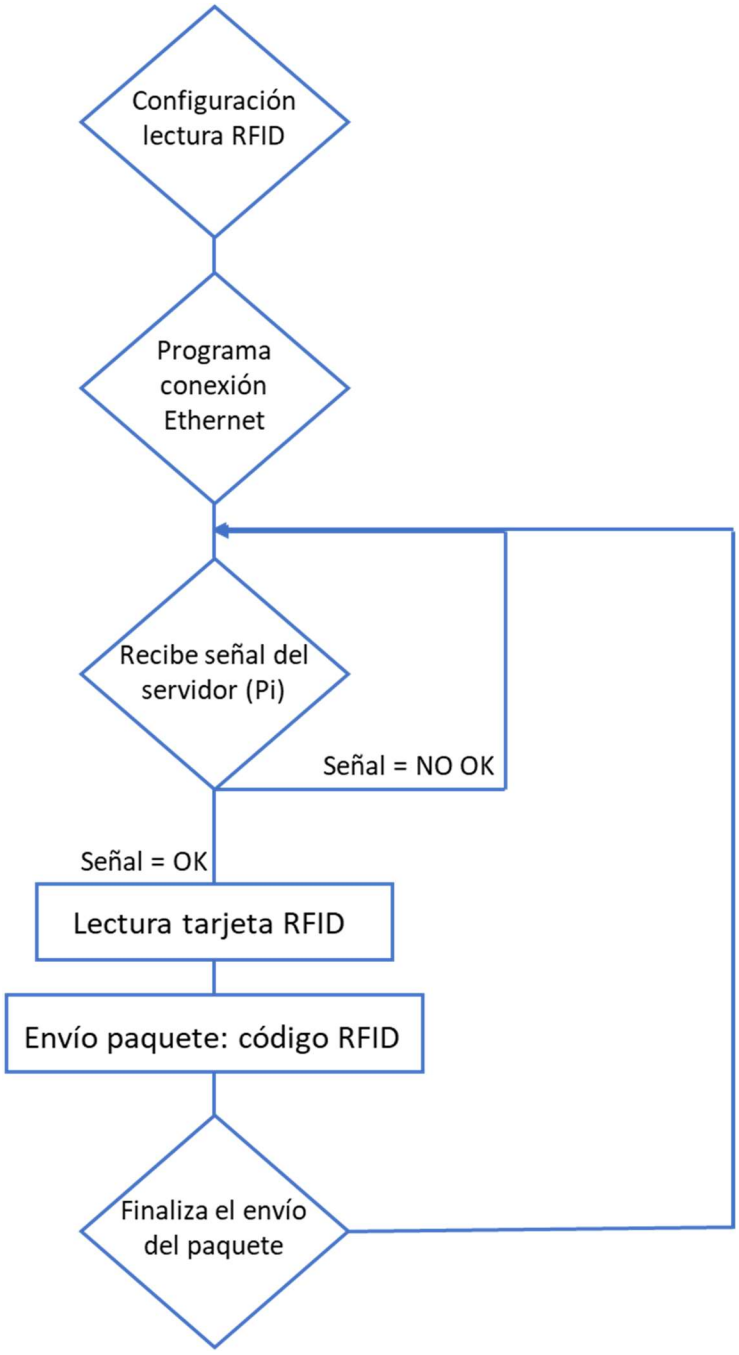


Ilustración 24. Flujo lógico del prototipo final en el Arduino.

En este apartado se explica cada elemento de la Ilustración 19.

RFID

El programa de lectura de los códigos de barras está en lenguaje C++ y utilizando el software de programación de Arduino. El código en detalle puede verse en el Anexo A.

La parte de lectura de tarjetas RFID proviene de los códigos ejemplo de Arduino, con modificaciones para distinguir cuales son los valores que sí deberían mandarse al servidor por considerarse una lectura correcta. En el programa se busca que además de leer los códigos, registre y envíe únicamente el valor cuando el código leído sea diferente al anterior, de manera que no se repitan valores y no se registren doblemente ni en la Blockchain ni en la base de datos de MySQL. A continuación, se explican algunas líneas del programa utilizado, aunque el programa completo se encuentra al final del documento en el apartado de Anexo.

En el código se inician los puertos para la antena RFID, y se establecen los valores de la placa ethernet , la IP y el puerto para permitir la posterior comunicación con el servidor. Cómo se explica más adelante se ha optado por asignar una IP fija a la placa ethernet, para no tener problemas en la comunicación de conectarse más dispositivos.

```
byte mac[] = {0xA8, 0x61, 0x0A, 0xAE, 0x00, 0x76};
IPAddress ip(192, 168, 1, 163);
unsigned int localPort = 8888;
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
String datReq;
int packetSize=0;
EthernetUDP Udp;

//Configuración RFID
const int RST_PIN = 9;              // Pin 9 para el reset del RC522
const int SS_PIN = 5;               // Pin 5 para el SS (SDA) del RC522
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Crear instancia del MFRC522

```

El puerto se asigna aleatoriamente, pero se hará coincidir con el puerto definido en el programa de la raspberry. Los pines utilizados para el RFID son el 9 y el 5, para no interferir en las comunicaciones de la Ethernet Shield. Y se inicializa también el buffer de memoria donde se almacena la información que posteriormente será leída, junto con la instancia que permita recibir y mandar paquetes por UDP.

```
char packetBuffer[UDP_TX_PACKET_MAX_SIZE
EthernetUDP Udp;
```

En el programa que se desarrolla en bucle, cada vez que el servidor esté listo para enviar información y en caso de recibir una nueva tarjeta RFID, esta se almacena en la variable rfid_uid

y se envía por conexión UDP al servidor. La variable codigo1 se encarga de almacenar la información para comparar con la lectura anterior, y confirmar que no se lee el mismo código repetidas veces.

```
1.
2.  void loop(){
3.
4.    packetSize = Udp.parsePacket();
5.    //Serial.println(packetSize);
6.


1.    if(packetSize>0){
2.      Serial.println("ok1");
3.
4.      Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
5.      String datReq(packetBuffer);
6.
7.      if(datReq == "codigo"){
8.
9.        Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());  //Initialize
10.       if (mfrc522.PICC_IsNewCardPresent()){
11.           if (mfrc522.PICC_ReadCardSerial()){
12.                 for (byte i = 0; i < mfrc522.uid.size; i++) {
13.                     Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
14.                     Serial.print(mfrc522.uid.uidByte[i], HEX);
15.                     String uid_part = String(mfrc522.uid.uidByte[i], HEX);
16.                     rfid_uid += uid_part;
17.                     ActualUID[i] = mfrc522.uid.uidByte[i];
18.                     codigo1= (ActualUID[1]);
19.                 }
20.                 Serial.println("");
21.                 delay(1);
22.                 // Finalizar lectura actual
23.                 mfrc522.PICC_HaltA();
24.            }
25.            Udp.print(rfid_uid);
26.            Serial.println(rfid_uid);
27.            Udp.endPacket(); //Se termina la transmisión del paquete
28.            rfid_uid = "";
29.        }
30.     }
31.   }
32.   memset(packetBuffer, 0, UDP_TX_PACKET_MAX_SIZE);
```

Para comparar el valor leído con el almacenado anteriormente en la variable que se envía al servidor se ha utilizado la siguiente función, que devolverá *Verdadero* en caso de que ambas lecturas sean la misma y por tanto, la información no se enviará al servidor.

```
1.  boolean compareArray(byte array1[],byte array2[]) {
2.    if(array1[0] != array2[0])return(false);
3.    if(array1[1] != array2[1])return(false);
4.    if(array1[2] != array2[2])return(false);
5.    if(array1[3] != array2[3])return(false);
6.    return(true);
7.  }
```

COMUNICACIONES

Para la comunicación entre el Arduino encargado de la lectura de los códigos y la Raspberry Pi, que hará de servidor se utilizará un protocolo de comunicación UDP. El protocolo de comunicación viene de las siglas User Data Protocol, protocolo de datos de usuario, y permite el envío de datos sin que haya una conexión previa puesto que ya contiene suficiente información sobre la dirección de envío en la cabecera. Al evitar el enlace que requieren otros protocolos como el TCP que exigen acuse de recibo entre el emisor y el receptor, mantiene un tiempo de transmisión bajo. En caso de una red muy saturada, con muchos Arduino enviando información de manera constante, como cada uno de ellos tendría que establecer la sesión antes de poder comenzar con el envío de información, podría llegar a provocar tiempos muy largos entre el envío de la petición de comunicación y el recibimiento del primer bit. Por tanto, para poder recibir la información leída por el Arduino de manera constante y sin retrasos, el protocolo UDP es el adecuado para este proyecto.

Para utilizar el protocolo UDP sin conexión se debe especificar tanto la dirección IP como el puerto de destino. El puerto se ha elegido aleatoriamente el 8888, con la única condición que quedase definido tanto en el Arduino como en la Raspberry [36].

Para asignar un IP de manera permanente se accede al router. Para poder asignarle la IP privada es necesario su dirección MAC, única para cada dispositivo, que en el caso de la placa de ethernet es: { 0xA8, 0x61, 0x0A, 0xAE, 0x00, 0x76}.

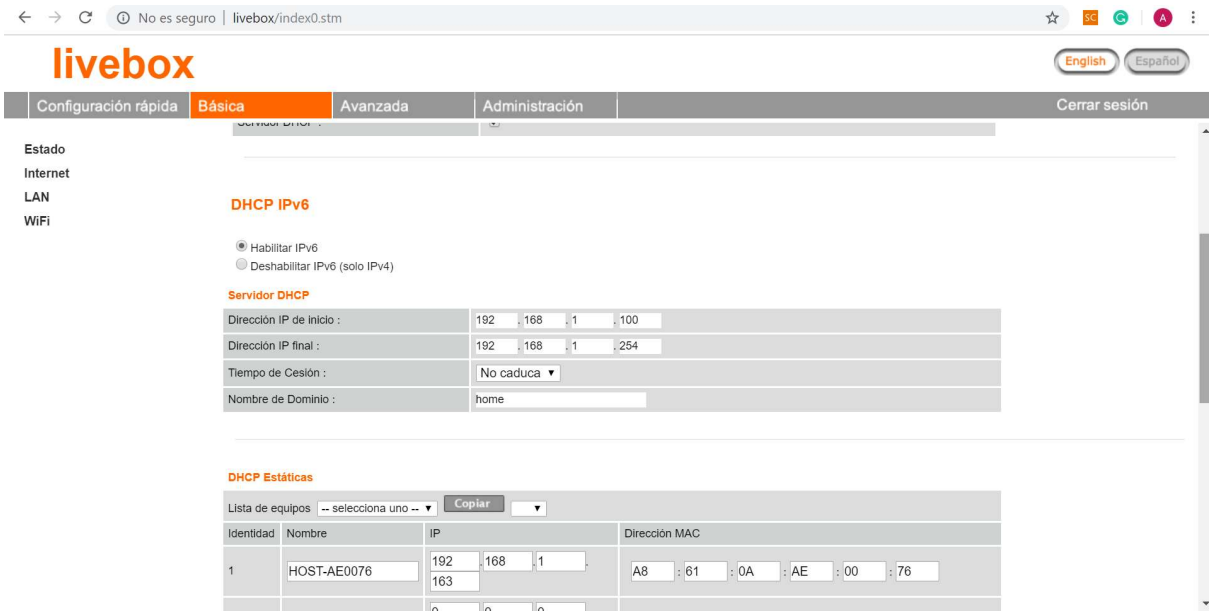


Ilustración 25. Configuración router IP fija.

Se accede al router abriendo en el explorador una pestaña e indicando, que suele estar asociado el IP del ordenador. De esta manera en el apartado de configuración DHCP indicando el MAC se asigna una IP a la placa ethernet, entre los IPs disponibles para ello El código detallado de comunicación entre el Arduino y el de la Raspberry Pi se puede encontrar en el Anexo A.

Durante el desarrollo del proyecto y al cambiar el prototipo del entorno de trabajo se encontraron algunas dificultades de la raspberry para conectarse a un nuevo router y asignarle una ip. En ese caso, una vez se asigna la ip de la forma que se acaba de mencionar será importante ejecutar los siguientes comandos también:

```
1. sudo ip route add default via 169.254.18.1
2. sudo dhcpcd wlan0
3. ip a
```

En el caso de este proyecto, la red a la que se intentaba conectar tenía como puerta de enlace predeterminada la 169.254.18.1., esa IP se modificará dependiendo del caso y de la red a la que se conecte el prototipo.

RASPBERRY PI

BASE DE DATOS LOCAL - MYSQL

Como se ha mencionado a lo largo del proyecto la información leída no se vuelca únicamente sobre la red de blockchain sino sobre una base de datos para almacenar toda la información.

MySQL se trata de un sistema de gestión de bases de datos. Se pueden crear bases de datos y tablas, insertar datos y modificarlos y eliminarlos, hacer consultas y otras muchas operaciones. En conclusión, administrar en su totalidad bases de datos. Para acceder y comunicarnos con el servidor se ingresan instrucciones desde la Terminal del servidor (Raspberry pi) en un lenguaje conocido como PHP. PHP es gratuito, y es uno de los lenguajes de programación más populares, es muy adecuado para el desarrollo web. Desde PHP accederemos al servidor, junto con esto instalaremos la aplicación de PhpMyAdmin que facilita la interacción con la base de datos.

El servidor web instalado en la Pi ha sido Apache. Es altamente personalizable y flexible al tener una estructura basada en módulos. Se trata de un servidor web de código abierto y gratuito, y con un software confiable y estable con un 60% del mercado, por lo que la documentación existente es extensa.

A continuación, se detallan los comandos utilizados para instalar Apache, PHP cómo intérprete, MySQL y PHPMyAdmin.

```
1. sudo apt update
2. sudo apt upgrade
3. sudo apt update
4. sudo apt install apache2 //Instala el servidor
5. sudo chown -R pi:www-
   data /var/www/html/ //Se cambian los permisos de lectura y de dueño
6. sudo chmod -R 770 /var/www/html/
```

Una vez instalado se comprueba que funciona Apache:

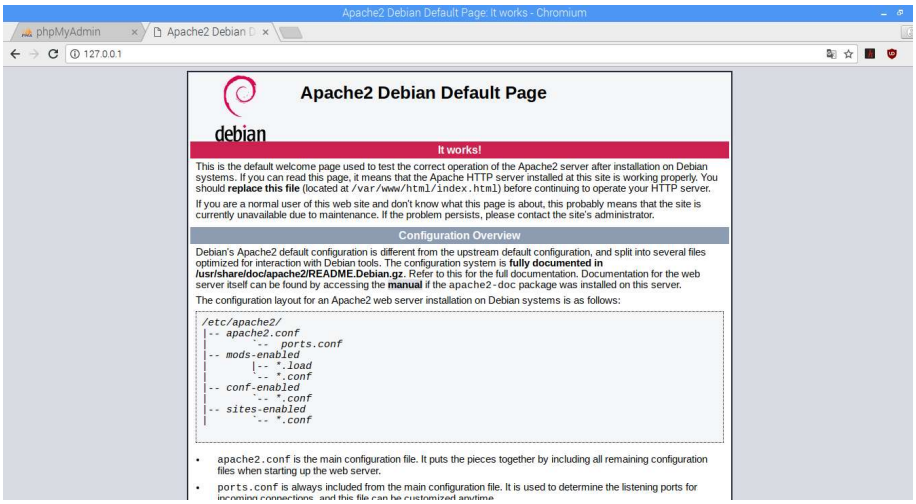


Ilustración 26. Servidor web APACHE.

Instalamos PHP:

```
1. sudo apt install php php-mbstring
2. sudo rm /var/www/html/index.html
3. echo "<?php phpinfo ();?>" > /var/www/html/index.php
```

Con PHP funcionando:

```
1. sudo apt install mysql-server php-mysql
2. sudo mysql --user=root //Para comprobar que todo sigue funcionando
3. DROP USER 'root'@'localhost'; //creamos el usuario
4. CREATE USER 'root'@'localhost' IDENTIFIED BY 'ainhoa';
5. GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost'
6. //Una vez creado se puede acceder con:
7. //mysql --user=root --password=ainhoa
8. sudo apt install phpmyadmin
9. //dbconfig-common - sin instalar.
10. sudo phpenmod mysqli
11. sudo /etc/init.d/apache2 restart
12. //Comprobamos la instalacion:
13. sudo ln -s /usr/share/phpmyadmin /var/www/html/phpmyadmin
```

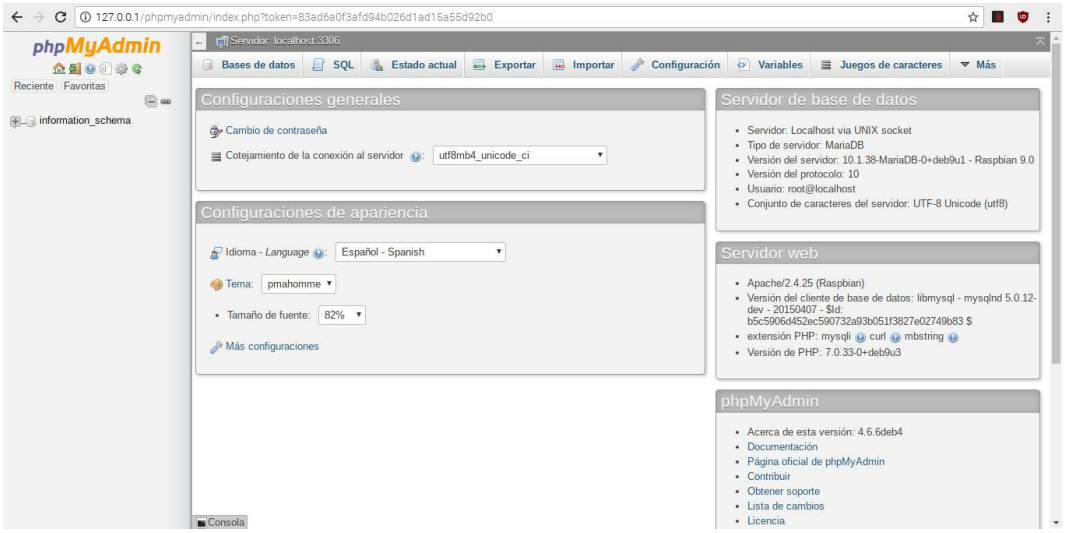


Ilustración 27. PHPMyAdmin interfaz.

Para el prototipo inicial se crean dos tablas distintas, una que se rellenará de manera automática con la información que el Arduino envíe a la Raspberry, y otra que funcionará simulando la base de datos de la fábrica.

Para la tabla que simula la base de datos, se crea una tabla sencilla con los siguientes campos:

```
mysql> DESCRIBE Lista;
+-------+-----------------+------+-----+---------+----------------+
| Field       | Type          | Null | Key | Default | Extra    |
+-------+-----------------+------+-----+---------+----------------+
| ID          | int(11)       | NO   |     | NULL    |          |
| CODIGO      | varchar(20)   | NO   | PRI | NULL    |          |
| DESCRIPCIÓN | varchar(100)  | NO   |     | NULL    |          |
| COSTE       | float         | NO   |     | NULL    |          |
+-------+-----------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

En la tabla, se rellenará todos los productos que puedan producirse o llegar a leerse por el dispositivo, con su código RFID, la descripción del producto y el coste. Para versiones futuras se podrán añadir tantos campos como se quiera.

Y para la tabla a rellenar de manera automática son los siguientes campos donde el código debería ser ForeignKey de la tabla anterior:

```
mysql> DESCRIBE Lecturas;
+-------+-----------------+------+-----+---------+----------------+
| Field     | Type          | Null | Key | Default | Extra          |
+-------+-----------------+------+-----+---------+----------------+
| ID        | int(11)       | NO   | PRI | NULL    | auto_increment|
| ANTENA    | int(11)       | NO   |     | NULL    |                |
| CODIGO    | varchar(20)   | NO   |     | NULL    |                |
| FECHA     | date          | NO   |     | NULL    |                |
+-------+-----------------+------+-----+---------+----------------+
4 rows in set (0.56 sec)
```

La propuesta es que se rellene el ID de manera automática, para contabilizar el número de productos/elementos que el dispositivo es capaz de leer. El campo: antena, para en el caso de existir diversos dispositivos se pueda reconocer desde cuál se está realizando la lectura. Los dos últimos campos son el código identificador RFID, y la fecha de la lectura. El identificador ID permitirá en caso de escalar el proyecto, se pueda relacionar la información disponible en las diferentes tablas.

El programa completo que permite la comunicación entre ambos dispositivos, así como las líneas que rellenan la tabla creada se encuentra en el apartado final del documento en Anexos. A continuación, se detallan algunas de las funciones y comandos claves que permiten enviar los datos a las tablas creadas con anterioridad. En primer lugar será importante que se disponga del paquete de mysql.connector (detallada su instalación en el Anexo). Este driver permite que desde el sistema se pueda acceder a la base de datos de MySQL.

```
1.  import mysql.connector
2.  from mysql.connector import Error
3.  from mysql.connector import errorcode
4.  try:
5.     connection = mysql.connector.connect(host='localhost',
6.                                  database='Portfolio',
7.                                  user='root',
8.                                  password='ainhoa')
9.
10.    sql_insert_query = (""" INSERT INTO `Lecturas`(`ANTENA`, `CODIGO`,`FECHA`) VALUES (%s
     , %s ,%s)""", (antena, codigo, fecha))
11.    cursor = connection.cursor()
12.    result  = cursor.execute(sql_insert_query)
13.    connection.commit()
14.    print ("La información se ha insertado correctamente en la tabla: Lecturas"")
15. except mysql.connector.Error as error :
16.     connection.rollback() #rollback if any exception occured
17.     print("Fallo al insertar la información en la tabla: Lecturas {}".format(error))
18. finally:
19.     #closing database connection.
20.     if(connection.is_connected()):
21.         cursor.close()
```

Para poder conectar con el servidor de MySQL será necesario indicar los argumentos propios de la base de datos, así como el usuario y la contraseña. Con el comando *sql_insert_query* se enuncian los campos a rellenar y posteriormente el tipo de datos a introducir, y en el caso de este proyecto las tres variables. Antena que indica desde que dispositivo se realiza el envío de la información (suponiendo que en un proyecto real serían muchos los dispositivos comunicándose con el servidor), el código de la tarjeta rfid leída, y fecha que se asigna automáticamente con el día, mes y año y la hora exacta en horas minutos y segundos.

BLOCKCHAIN – RED DE ETHEREUM

En este proyecto se busca desarrollar un prototipo funcional y una síntesis de cómo introducir medidas en la blockchain a través de contratos inteligentes simples. Para poder enviar esta información y almacenarla en la blockchain, son múltiples los pasos, programas y configuraciones a realizar.

a) Configuración correcta del sistema operativo en la Raspberry. Este proceso será necesario no únicamente para interactuar con la Blockchain, pero para permitir la comunicación con el Arduino y la Base de Datos.
b) Despliegue del nodo.
    1. Para poder interactuar con el nodo una vez este desplegado será necesario poseer una interfaz o consola para hacerlo. Para la red Ethereum son dos las más comunes:
        1.1 Geth: interfaz de línea de comandos para poder ejecutar el nodo de Ethereum.
        1.2 Parity: es la alternativa a Geth para conectar con el nodo, se trata de otro cliente para conectarse a las redes de Ethereum.

    La diferencia entre ambas reside en que Geth no cuenta con transacciones privadas, pero Parity si lo permite aunque hasta el momento únicamente permite una transacción privada por bloque. Por otro lado, la gran desventaja es que Geth no permite enviar transacciones con coste de gas 0, y Parity lo permite de manera que en el caso de existir transacciones sin costo no desencadene en reservas de Ether en cero. [44]

    Para el proyecto se ha decidido utilizar Geth, por ser más fácil su uso y existir más documentación disponible. Geth tiene un modo de desarrollo que establece una red de pruebas Ethereum de un solo nodo con una serie de opciones útiles, y no se necesitan de las ventajas de Parity respecto al coste de Ether para el desarrollo del proyecto.

Al sincronizar el nodo se debe elegir el tipo de nodo según su modo de sincronización:

    2.1 Full nodo: contienen la historia completa de las transacciones desde el bloque Génesis. Se encargan de verificar todas las transacciones y bloques
    2.2 Light nodo: solo contiene un subconjunto de transacciones, estos nodos se ayudan de los full nodos para obtener los datos que puedan necesitar.

Ambos tipos de nodos ejecutan al instalarse un software denominado Ethereum Virtual Machine, que se encarga de mantener todos los datos de la red de

Blockchain sincronizados. Aparte de estos dos modos de sincronización existe una funcionalidad al sincronizar la cadena de bloques utilizando –dev, que permite en modo desarrollador utilizar una red privada con un único nodo con una cuenta predefinida y saldo.

2. Los nodos requieren de cuentas para poder realizar las transacciones. Como se menciona si se opera en modo desarrollador (dev) las cuentas ya cuentan con un saldo, sin embargo, al sincronizar de manera "full" o "light", el envío de Ether se puede realizar a través de comandos o utilizando Ropsten, que te permite enviar 1 Ether a una cuenta determinada cada 24 horas. Todo se detalla más abajo en el apartado de Cuentas.

c) Desplegar el contrato. Una vez sincronizada la cadena, para poder enviar la información a la blockchain se desarrollan contratos inteligentes sencillos, y para ello se usa Remix. Se trata de un entorno de desarrollo y de compilación de estos contratos basado en un explorador Web.

d) Por último, para visualizar todas las transacciones una vez desarrollado el contrato, es útil usar Ganache una herramienta que simula una blockchain. De esta manera, se puede ir ejecutando y comprobando el uso de los contratos antes de finalizar la sincronización del nodo en cualquiera de sus modos, y permite hacer pruebas sin realizar gastos de gas.

A continuación, se detallan todos los pasos realizados para seguir el esquema arriba mencionado, junto con los problemas encontrados a evitar en futuros desarrollos.

CONFIGURACIÓN NECESARIA

Sistema operativo

Para la descarga del nodo y su configuración se han seguido múltiples manuales de Medium y Freecodemap, que se han añadido al final en el apartado de referencias [37][38][39][40] .

Cómo sistema operativo se ha utilizado ***Raspbian***, se trata de una versión de Debian adaptada para la arquitectura propia de la tarjeta ARM. Este sistema incluye todo lo necesario para hacer de la Raspberry un sistema totalmente funcional.

Ilustración 28. Logo Raspbian

De las opciones que ofrece se ha utilizado el instalador de sistemas operativos para la Raspberry Pi ***NOOBs*** Lite, que ocupa poco, pero requiere Internet para su instalación desde el principio.

Los pasos seguidos han sido los siguientes:

1. Acceder a la página oficial de Raspberry Pi para descargar el sistema operativo oficial *https://www.raspberrypi.org/downloads/* .
2. Insertar la tarjeta SD de 32G en el ordenador. La tarjeta SD se formatea y posteriormente se descomprime el fichero zip NOOBs.
3. Una vez todos los archivos se copian, se introduce la SD en la Raspberry Pi.
4. La Raspberry Pi, se conecta por HDMI a la pantalla, junto con el ratón, y el teclado y al cable de red, y la Raspberry arrancará de manera automática.
5. Al descargar el instalador, lo primero será elegir el sistema operativo en este caso; Raspbian. Se configura el idioma, la conexión Wifi y el resto de los parámetros y ya está lista para usar.

NODO DE ETHEREUM

Para poder comunicar la Raspberry con la red Ethereum se utiliza ***Geth.*** Se trata de una interfaz de línea de nodos para ejecutar un nodo Ethereum completo. Para la instalación los pasos a seguir son:

1. Acceder a la página oficial de Geth Ethereum y descargar la última versión existente de Geth, de arquitectura ARMv7 de la pestaña Linux. En el momento de la elaboración de este proyecto la versión Geth era la 1.8.27. y el enlace: https://gethstore.blob.core.windows.net/builds/geth-linux-arm7-1.8.27-4bcc0a37.tar.gz
2. Los comandos por ejecutar en la terminal de la Raspberry Pi son los siguientes:

```
$wget https://gethstore.blob.core.windows.net/builds/geth-linux-arm7-1.8.27-4bcc0a37.tar.gz //Última versión de Geth
$ tar -xvf geth-linux-arm7-1.8.27-4bcc0a37.tar.gz //Descomprime el archivo
$cd geth-linux-arm7-1.8.27-4bcc0a37
$ sudo mv geth /usr/local/bin/ //Se crea un directorio a Geth dentro de la carpeta
$ geth license //Comprobación que se ha instalado correctamente
```

Una vez instalado Geth, para evitar qué al sincronizar el nodo este cree los directorios de manera automática, se crea el directorio dentro del disco duro. Al tratarse de un sistema Linux, muchos de los discos duros externos únicamente soportan lectura. Para poder añadir lectura es necesario incluir NTFS-3G [41]

```
sudo apt-get install ntfs-3g //Permite NTFS para Linux
sudo mkfs.ntfs /dev/sda1 -f -v -I -L untitled //Permite escribir en el disco duro, y asigna untitled al disco duro.
```

Una vez el disco duro permite escribir sobre él ya se configura para poder comenzar la sincronización del nodo:

```
$ cd /media/pi/untitled/
$ mkdir Ethereum
$ cd Ethereum

//Se eligió para sincronizar el nodo el método "light"
$ mkdir light
% cd light
$ geth --syncmode=light --testnet --rpc --rpcapi "eth,net,web3,personal,admin" --
rpccorsdomain "*"  --rpcport 8545 --datadir ./ --bootnodes
"enode://6332792c4a00e3e4ee0926ed89e0d27ef985424d97b6a45bf0f23e51f0dcb5e66b875777506458a
ea7af6f9e4ffb69f43f3778ee73c81ed9d34c51c4b16b0b0f@52.232.243.152:30303,enode://94c15d1b9
e2fe7ce56e458b9a3b672ef11894ddedd0c6f247e0f1d3487f52b66208fb4aeb8179fce6e3a749ea93ed147c
37976d67af557508d199d9594c35f09@192.81.208.223:30303"
```

El comando para sincronizar la cadena se compone de Geth, necesario para interactuar con la red, y una serie de comandos.

Existen varios modos de sincronización:

- Syncmode=light: no descarga la red blockchain completa. Unicamente lo que hace es descargar los encabezados de los bloques para poder validad la autenticidad de las transacciones. Por esta razón, estos nodos son más fáciles de mantener y sincronizar. Estos nodos requieren de nodos completos para poder conectar con la red y realizar transacciones. Ocupan unos 500 MB de memoria y se sincroniza en cuestión de minutos, sin embargo, debido a la limitada potencia de la raspberry del prototipo el proceso puede extenderse a un día. El objetivo de estos nodos es que puedan funcionar en dispositivos integrados, y por ello se ha elegido para este proyecto. Aunque cada vez que el nodo quiera consultar una cadena de bloques, este debe pedir a un nodo completo que lo haga en su nombre.
- Syncmode=full: tiene una copia entera del estado completo de la cadena de bloques Ethereum y ejecuta cada transacción que se mine. Esto requiere mas de 120 GB de almacenamiento y más de 8Gbs de memoria. Para sincronizar un nodo completo en un ordenador con mucha potencia puede llevar más de 12 horas. Para el prototipo desarrollado en este proyecto, en el que se utiliza una raspberry pi, el tiempo de sincronización puede aumentar a varios días, y por ello se decidió no contemplar esta opción.
- Syncmode: --dev. Geth tiene un modo de desarrollo que establece una red de pruebas Ethereum de un solo nodo con una serie de opciones útiles.

En este Proyecto se han probado light y dev que se explican en detalle. Con el siguiente comando el método de sincronización elegido fue light:

```
$ geth --syncmode=light --testnet --rpc --rpcapi "eth,net,web3,personal,admin" --
rpccorsdomain "*"  --rpcport 8545 --datadir ./ --bootnodes
```

```
"enode://6332792c4a00e3e4ee0926ed89e0d27ef985424d97b6a45bf0f23e51f0dcb5e66b875777506
458aea7af6f9e4ffb69f43f3778ee73c81ed9d34c51c4b16b0b0f@52.232.243.152:30303,enode://9
4c15d1b9e2fe7ce56e458b9a3b672ef11894ddedd0c6f247e0f1d3487f52b66208fb4aeb8179fce6e3a7
49ea93ed147c37976d67af557508d199d9594c35f09@192.81.208.223:30303"
```

Cada vez que se ejecuta una transacción ligada a un contrato supone un gasto de gas, y por tanto un coste monetario. Hay veces que este coste puede llegar a ser prohibitivo si hay un sobreuso de la red en el momento de la transacción, y además la tecnología Blockchain por ser permanente implica que un error desplegado en un momento en la red continuará abierto lo que puede ser financieramente peligroso. Por esa razón, para realizar pruebas se usan Testnets que son copias de la red Ethereum totalmente idénticas en cada aspecto incluido los procesos de minar, a excepción del uso del Ether.

En los comandos de sincronización se establece cada una de las siguientes configuraciones:

1. El nodo se conecta a la red Testnet Ropsnet, una red de pruebas en la que probar el código antes de hacerlo sobre la red principal. La mayor diferencia con la red principal es que escribir sobre esta red es gratuito, pero es la más similar a la real.
2. Se habilitan posteriormente los APIS, con –rpc para habilitar JSON-RPC, y luego –rpcapi para habilitar todas las APIs ofrecidas por la interfaz JSON-RPC, que por defecto normalmente son eth, web3 y net.
   Estas APIs son el conjunto de comandos y funciones que permiten al desarrollador crear programas para determinados sistemas operativos. Permiten simplificar al programador escribir todo desde cero, puesto que suelen tener funciones predefinidas asociadas para interactuar con el sistema operativo o un programa, en este caso el nodo de Ethereum. Por ejemplo, como posteriormente se indica con el API de eth se crean nuevas cuentas para poder realizar el envió de información a la red.
3. Se indica el puerto para la comunicación con el nodo, por defecto localhost y puerto 8545. –datadir indicará el directorio donde escribir todos los valores de la cadena, que en este caso se ha elegido el directorio donde se ejecuta el comando dentro del disco duro extraíble.
4. Se añaden los pares, enumerando sus enodes que funcionan como nodos de arranque y realizan el descubrimiento de pares.

Para añadir más pares en el momento posterior a la sincronización, estos pueden ser algunos a utilizar. Cómo se ha mencionado anteriormente, para poder realizar transacciones habiendo sincronizado un *nodo light* se requiere de un nodo completo para poder actuar, o sino las transacciones no se podrán realizar. Existen multitud de listas con nodos disponibles en la red a los que poder conectarse, a continuación se enuncian 7, puesto que existe el riesgo de que algún nodo se inutilice y deje de funcionar.

```
1. admin.addPeer("enode://0b64924d478abaf6900ffed857dc066b29e6a9498c8a6604a159555bd08fe
   1ccf3c2cefdbd625b0e7cf93b49c3cc6d6e412a5cde92b4a7d2b8bfd6f10d56511e@136.144.129.222:
   30303");
2. admin.addPeer("enode://20c9ad97c081d63397d7b685a412227a40e23c8bdc6688c6f37e97cfbc22d
   2b4d1db1510d8f61e6a8866ad7f0e17c02b14182d37ea7c3c8b9c2683aeb6b733a1@52.169.14.227:30
   303");
```

```
3. admin.addPeer("enode://4fee548ed37a0aa07101479f0fb8672573b1396b100869eb8a6db77e05b87
   80293376c0bdd8ed41bfb503e2df20b0702c7b61ff795d9aa3eedf9d70b952643eb@34.195.230.169:3
   0303");
4. admin.addPeer("enode://5b190dc2848404a46ac47413c35e68e73fd5e4eb4fd681ed511ef3280d471
   69e703990ecd4e67a3c1eaa4e4788d56d182b3c62f32628bcc9fc772f6568ffc19e@104.236.178.16:3
   0303");
5. admin.addPeer("enode://6ce05930c72abc632c58e2e4324f7c7ea478cec0ed4fa2528982cf3448309
   4e9cbc9216e7aa349691242576d552a2a56aaeae426c5303ded677ce455ba1acd9d@13.84.180.240:30
   303");
6. admin.addPeer("enode://72c074af1e24caaa6904503fa88c3ef2881d319bebb32dbe8282f4be3dab0
   7ffd873970642a5ca7d8f03e3e14b8aea080811828c5ffbd183c94778ef53c904e2@188.214.30.138:3
   0303");
7. admin.addPeer("enode://a9e72b8b22f5f5ca858588a378af9f2214ac81f65e53b93e0e2bf00e6d36d
   353ad71a6f28b916155548922cf14401e4ab2950ec2a7a6870a512890e214dcbc76@139.59.155.74:30
   303");
```

Una vez el nodo este sincronizado será necesario crear una nueva cuenta, y protegerla con contraseña antes de poder interactuar con el nodo:

```
1. > personal.newAccount()
2. Passphrase:
3. Repeat passphrase:
4. ["0xaa484749a0fec1c24037922b862e3d8a00152579"]
```

Las cuentas son imprescindibles para comunicarse con la red blockchain. Cómo se especificará en el apartado de los contratos inteligentes, para poder realizar el envío de información dos campos son imprescindibles, uno la dirección del contrato inteligente, y el otro la cuenta desde la que se realiza la transacción.

De manera automática dentro del directorio light se crea una carpeta *keystore*, en la que se especifican en archivos de texto todos los detalles de la cuenta. El nodo ya está preparado para actuar sobre él. Se pueden crear tantas cuentas como se necesite. En este caso, con una cuenta desde la que realizar el envío de los códigos RFID es suficiente, sin embargo, a la hora de escalar este proyecto con diferentes sensores, la información a enviar se podría hacer desde cuentas diferentes puesto que el envío de información se realice en distintos intervalos dependiendo del sensor.

En el modo de sincronización –dev, al tratarse de una simulación las transacciones no suponen un gasto de gas, pero con el modo light es necesario desde el momento inicial que la cuenta tenga ether para poder realizar las transacciones. En la consola Geth se ejecuta:

```
1. > address1 = ('0x7f999f948fb8ea004d2b961e73a2ff7e4a49642b')
2. > eth.sendTransaction({from: address1, to: eth.accounts[0], value: web3.toWei 1,
   “ether”)})
```

Otra manera más sencilla de mandar ether a una cuenta, pero limitada a una transacción cada 24 horas es utilizando Ropsten:

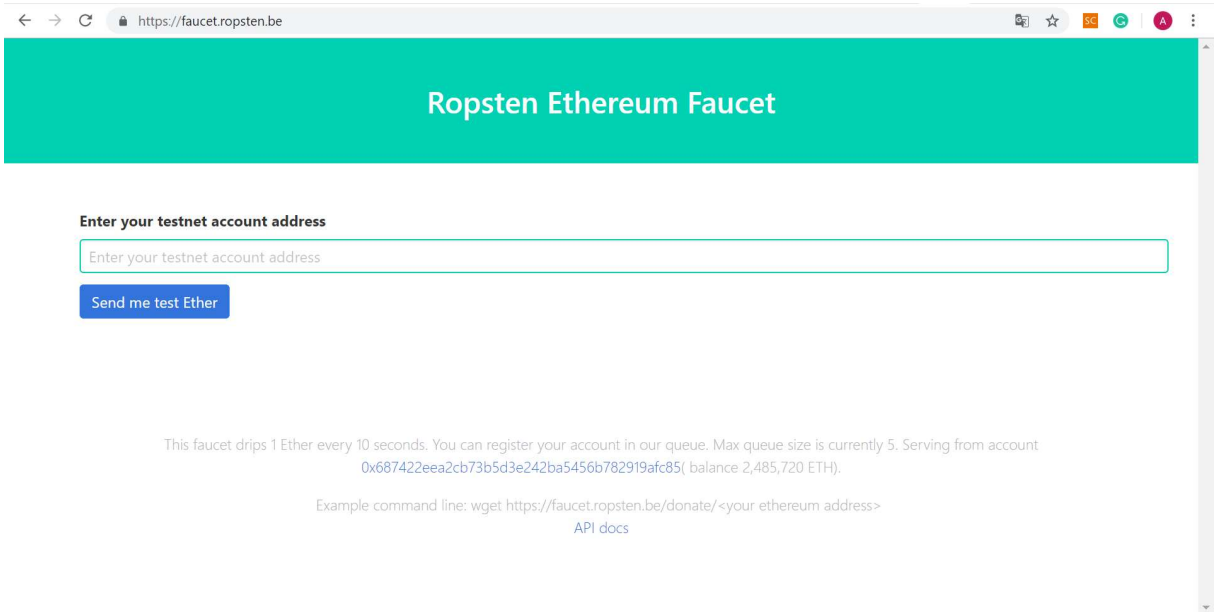
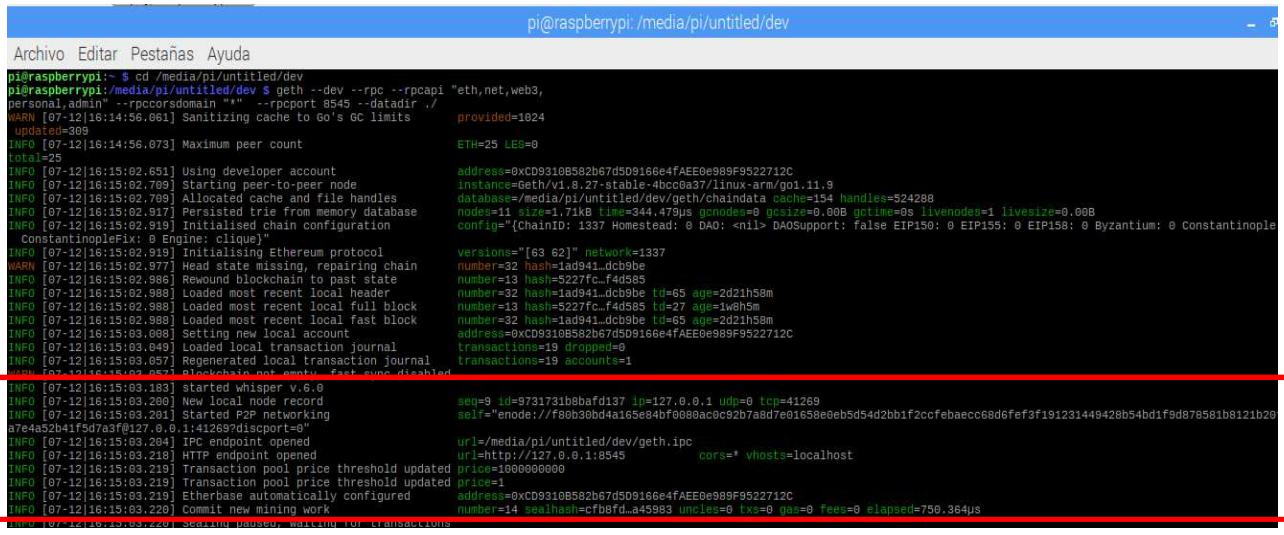


Ilustración 29. Ropsten Ethereum interfaz.

Tras la realización de este prototipo, se considera que es importante que las primeras pruebas se realicen sincronizando la cadena en modo dev, en el que se simula la interacción con la cadena y sincroniza mucho más rápido.

```
1. $ geth --dev --rpc --rpcapi "eth,net,web3,personal,admin" --rpccorsdomain "*"  --
   rpcport 8545 --datadir ./
```



Se trata del modo desarrollador, con una red privada de un único nodo con la cuenta ya predefinida que se indica nada más iniciar la sincronización en la terminal de la raspberry y con saldo. Al iniciar las primeras pruebas con este modo de sincronización, se pueden comprobar

los comandos y se observa como las transacciones se envían en cuestión de milisegundos puesto que no hay un gasto de gas realmente asociado a la transacción.

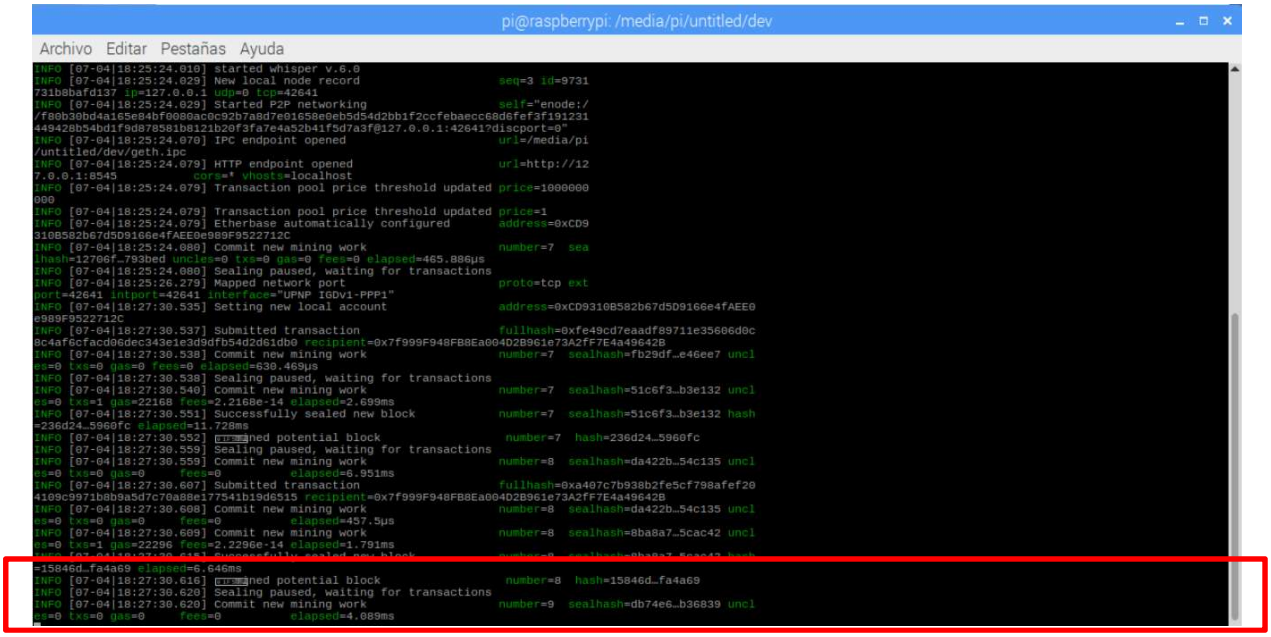


Ilustración 30. Transacción en modo dev.

En el momento que comienza la descarga del nodo, se puede interactuar gracias a la consola *Geth.* En otra terminal se ejecutan los siguientes comandos:

```
1.  $ pwd
2.  /media/pi/untitled/Ethereum/light/geth
3.  $ geth attach geth.ipc
```

Una vez la conexión con Geth.ipc se realizan consultas sobre el nodo de Ethereum. Por ejemplo, con el comando eth.syncing, se monitoriza cual es el último bloque sincronizado, y una vez toda la cadena ya esté sincronizada, devuelve: false. Es en ese momento, en el que es posible la interacción con el nodo para crear nuevas cuentas, o administrar los peers.

Durante la sincronización si se usasen comandos para la creación de cuentas, o para asociar peers al nodo existente, la raspberry entra en error como se ha mencionado debido a su potencia limitada, y aparece un mensaje de EOF (“End of File”), y será necesario reiniciar el proceso.

La información que se obtiene al conectarse a la consola Geth para visualizar el progreso de la sincronización se muestra en la siguiente imagen:

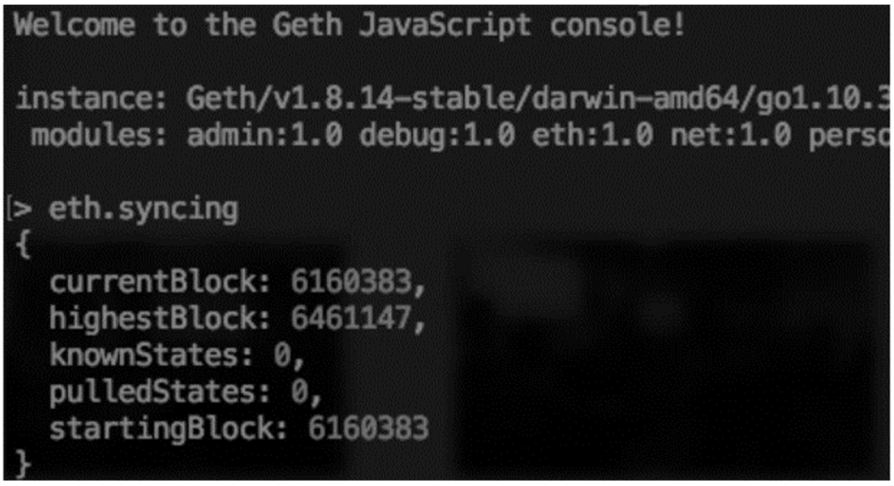


Ilustración 31. Geth console en la Raspberry Pi 3.

Una vez la cadena esté sincronizada, se comprueba que la sincronización se completó de manera correcta:

```
5.  > eth.getBlock("latest")
```

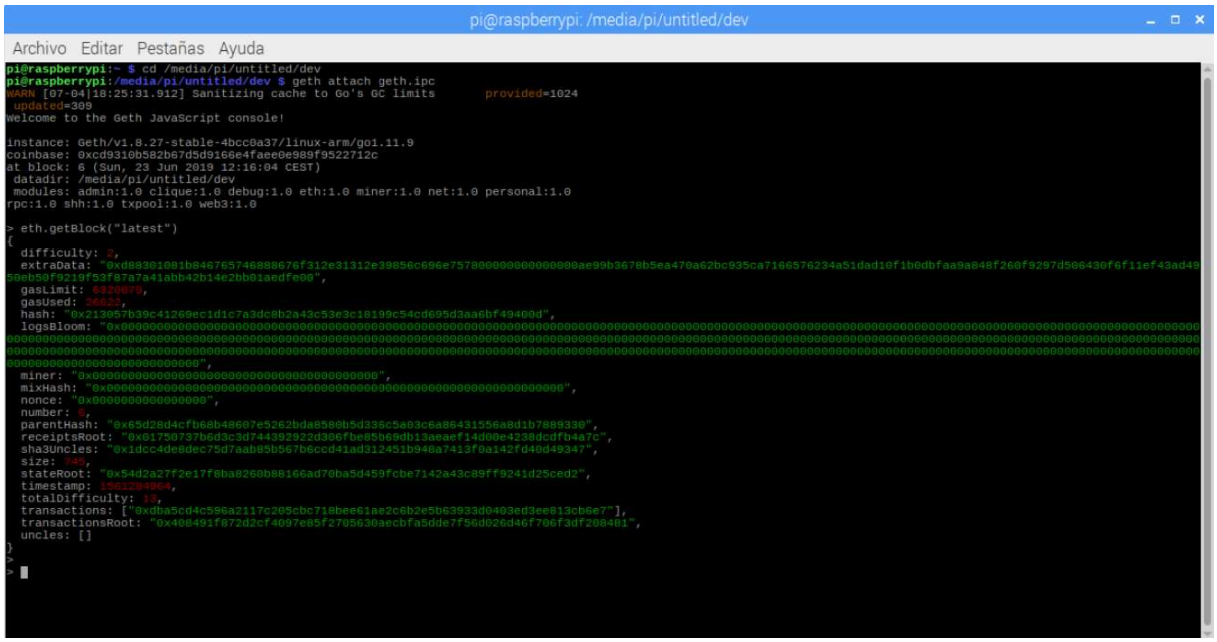


Ilustración 32. Último bloque sincronizado en la Blockchain.

Problemas en la configuración del nodo.

Es importante mencionar los problemas encontrados durante la sincronización del nodo para desarrollos futuros, puesto que la información pública disponible es limitada y más aún cuando se trabaja con sistemas menos potentes como la raspberry.

Los problemas encontrados durante la sincronización de la cadena se deben fundamentalmente a la raspberry. Es importante, considerar la capacidad limitada que tiene para sincronizar toda la cadena. Durante el tiempo de sincronización en modo light, que puede llegar a suponer de un periodo de hasta 3 días, la raspberry está "inutilizada", de manera que intentar llevar procesos paralelos puede implicar que la raspberry se reinicie, y, por tanto, sea necesario comenzar con el proceso desde el principio. La limitación no es únicamente de potencia sino de almacenamiento, sincronizar toda la cadena supone creación de carpetas para albergar las claves, las cuentas y la cadena y, por tanto, es estrictamente necesario indicar al sincronizar la cadena la dirección donde albergar toda la información, y que esta sea en el disco duro externo. Aunque el disco externo tenga capacidad suficiente como para almacenar la sincronización de la cadena en modo completo, se considera que las limitaciones de usar una Raspberry impiden en este modo de sincronización, puesto que puede llegar a tardar hasta 10 días, y es posible que algún bloque no se sincronice de manera correcta y corrompa todo el proceso, y posteriormente no pueda ser utilizado.

DISEÑO Y DESPLIEGUE DEL SMART CONTRACT

El contrato inteligente para interactuar con la red está escrito en un lenguaje de programación denominado Solidity. Para empezar a programar se necesita un entorno de desarrollo que en este proyecto se utiliza Remix. Remix es un entorno para desarrollar, compilar y posteriormente desplegar los contratos inteligentes. Para el proyecto se ha utilizado la versión online, aunque es posible instalarlo en el ordenador siguiendo las indicaciones de :

https://github.com/ethereum/remix-ide.

La versión online tiene la siguiente interfaz:

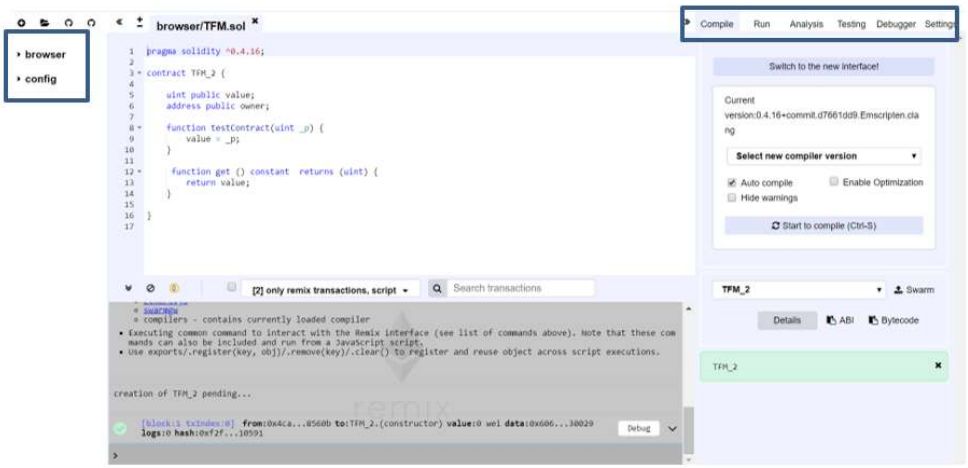


Ilustración 33. Remix IDE .

En la parte central se escribe el código:

```
pragma solidity ^0.4.16;

contract TFM{

    string public value ;
    address public owner;
    string public date;

    function testContract(string _p){
        value = _p;
    }

    function testDate(string _d){
        date = _d;
    }

    function get() constant returns (string){
        return value;
    }
    function getDate () constant returns (string){
        return date;
    }
}
```

Una vez se compila sin errores, ya se puede desplegar. En la pestaña de run situada en la esquina superior derecha permite elegir el entorno de trabajado donde desplegarlo. Las opciones son:

2.1. JavaScript VM : todas las transacciones se ejecutan en una cadena de bloques en un espacio reservado en el navegador. Sin embargo, no persiste nada, y una vez se recarga la página se reinicia un nuevo bloque desde cero. Este modo se ha utilizado en las pruebas iniciales con ganache para entender las transacciones y verificar el correcto funcionamiento.

2.2. Injected provider: remix se conecta a cliente como MetaMask o Ropsten. No se ha utilizado en ningún momento a lo largo del proyecto.

2.3. Web3Provider: remix se conecta a un nodo. Para ello es necesario indicarle la dirección URL en donde se encuentra el nodo privado. En el caso del proyecto en http://127.0.0.1:7545. , como se ha indicado al sincronizar la cadena.

En este proyecto, se usa la última opción. Se elige el contrato a desplegar, en el *combo box,* en este caso es “TFM, y pulsando sobre deploy, se despliega sobre el nodo seleccionado:

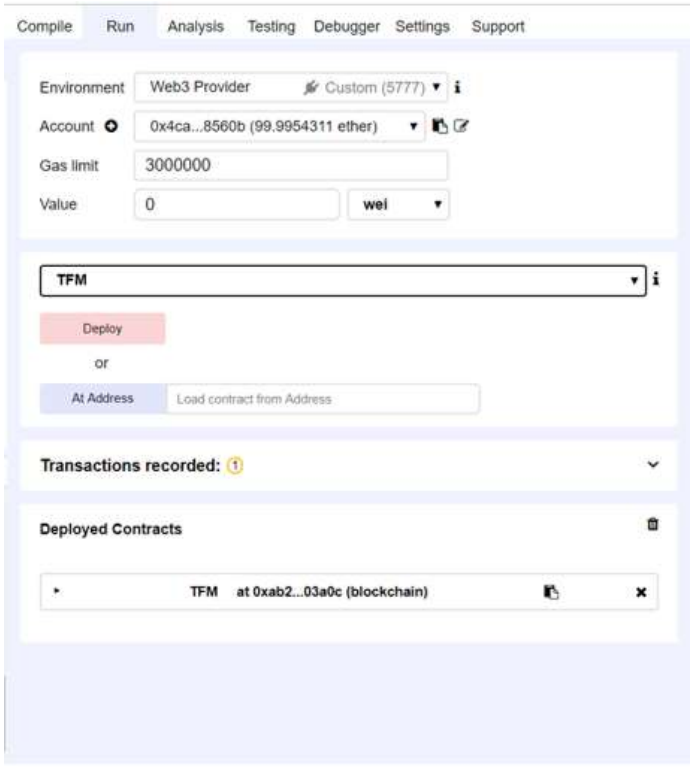


Ilustración 34. Interfaz REMIX

Una vez todo se ejecuta de manera correcta, en la parte inferior de esa misma pestaña devuelve los siguientes comandos:

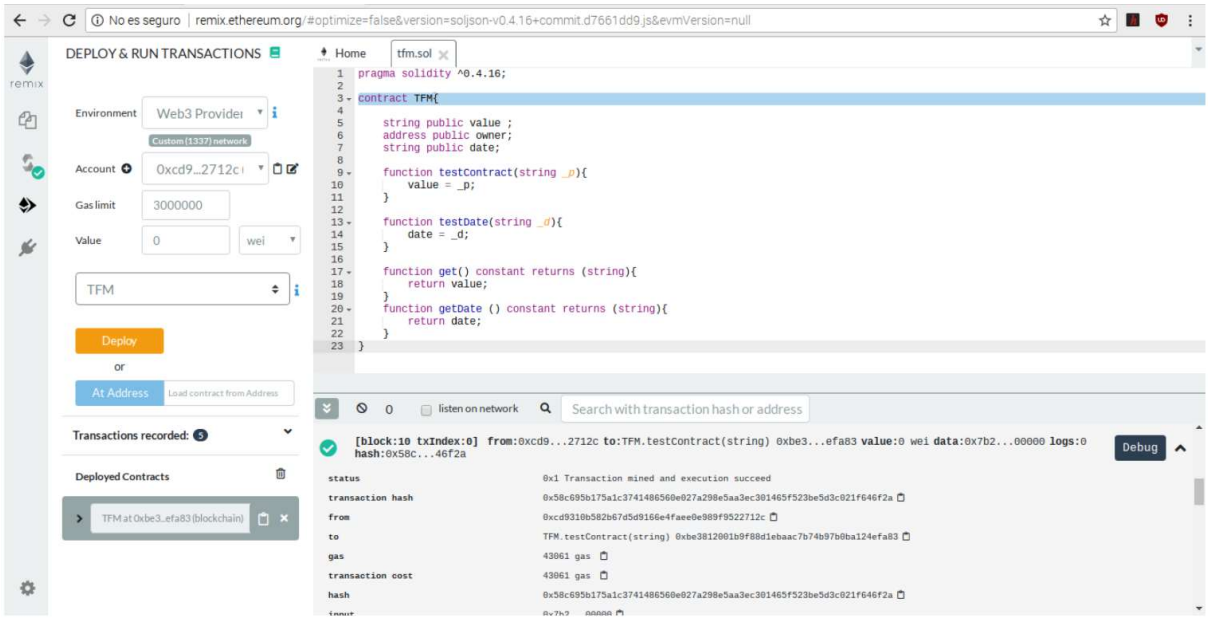


Ilustración 35. Remix interfaz contrato desplegado

Ahí se pueden observar los detalles del contrato, como el gas utilizado en la primera transacción del contrato y la dirección del contrato. En la línea que inicia con un to: se puede observar cual es la dirección en la que está alojado el contrato, información que es imprescindible detallar en el programa de la raspberry, para que las transacciones se realicen. El status muestra que la transacción se realizo y se mino de forma correcta. Puede llegar a ocurrir que la transacción no se realice por un exceso uso de gas, es decir, si una transacción lleva asignado un coste de gas mayor al permitido en los bloques de la red, la transacción no se realizaría.

El contrato se despliega una única vez al iniciar la primera comunicación. A partir de ese momento se generan transacciones por cada envío de información a la red. El contrato utilizado en este prototipo es muy sencillo, por un lado, permite enviar no solo los códigos leídos a la red, sino la fecha y la hora exacta del envío, y aunque con los *hash* que se obtienen de cada transacción se puede consultar la información en cualquier momento y de manera gratuita, se han incluido dos funciones de consulta. Por ejemplo, se quiera en un determinado momento saber cual fue el último momento de lectura o la última vez que se midió la temperatura de una planta.

VISUALIZACIÓN

Se ha utilizado ***Ganache,*** una herramienta que permite simular y testear con la red de Ethereum. Durante el proyecto se usó para poder realizar pruebas del funcionamiento de los contratos y su interacción con el nodo se utilizó Ganache. Ante un desconocimiento del funcionamiento de la red, la herramienta permite entender como se suceden las transacciones y la información que se puede obtener de ellas previamente a tener instalado todo en la Raspberry. Además al existir la versión web de Remix, y Ganache poder instalarse en el propio ordenador sin dificultades, permite hacer las pruebas sin necesidad de desplegar toda la parte de Hardware y permite trabajar en el proyecto desde cualquier punto únicamente con el ordenador.

Para poder utilizar Ganache se descarga la herramienta desde:

http://trufflеframework.com/ganache

y se accede a un espacio de trabajo con 10 cuentas que te permiten interactuar con la cadena de bloques. Te permite conocer el saldo, y las transacciones realizas en cada una de las cuentas, explorar bloques y las transacciones, de manera que permite un conocimiento del funcionamiento del sistema.[42]

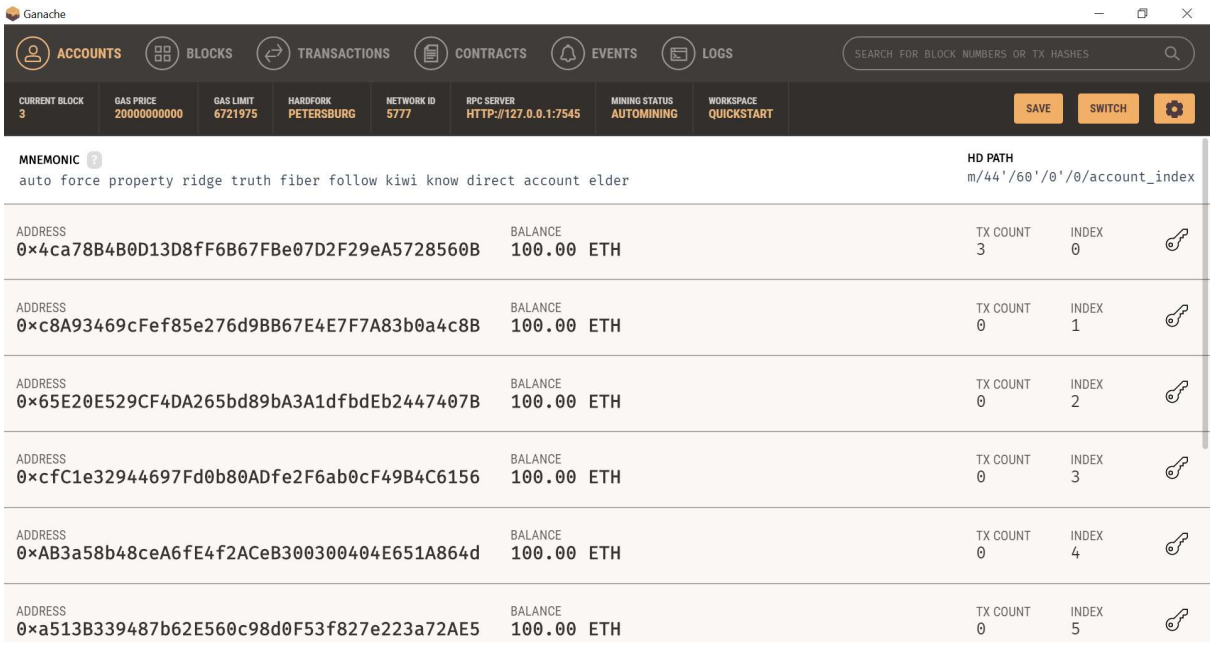


Ilustración 36. Área de trabajo de Ganache.

En la parte superior del área de trabajo se identifica nuestra red y el RPC server,(127.0.1:7545) que nos permite conectarnos a la herramienta desde Remix, el entorno IDE, dónde se desarrollan los contratos inteligentes basados en Solidity.

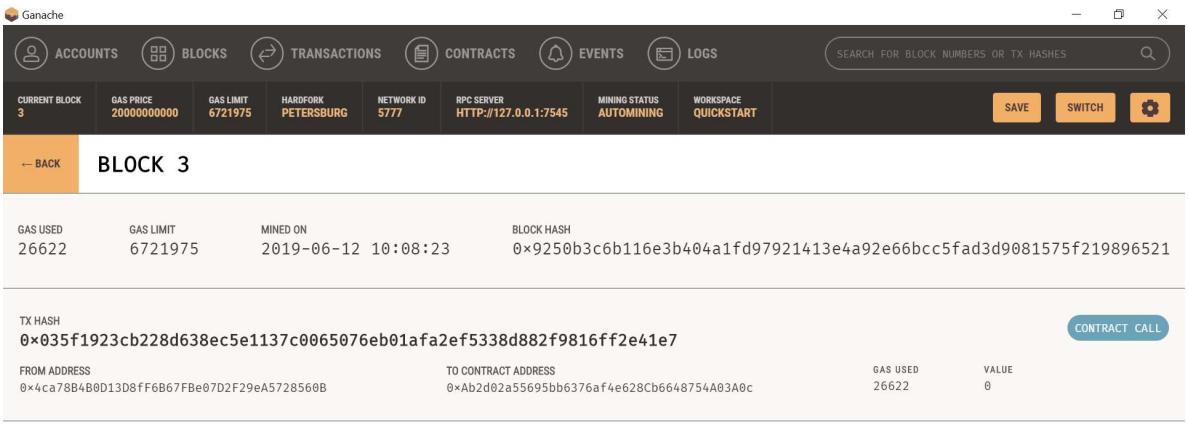


Ilustración 37. Detalles de un bloque creado a través de una transacción.

Si analizamos el bloque para entender la información que nos proporciona, se observa la fecha en la que fue minado, el número de transacciones que posee y también el Gas usado, con el que se puede estimar el coste de cada transacción. En el detalle se puede visualizar, el Hash de la transacción, así como la dirección Ethereum, y la dirección del Smart Contract.

USO DEL SMART CONTRACT

En la Raspberry Pi, el flujo del código es el que se muestra en la Ilustración 33. Una vez se configura la comunicación entre la Raspberry y el Arduino, se establece la conexión con el nodo local indicándole no sólo la dirección de la cuenta desde la que se realizan las transacciones sino además la dirección del contrato creada en remix.

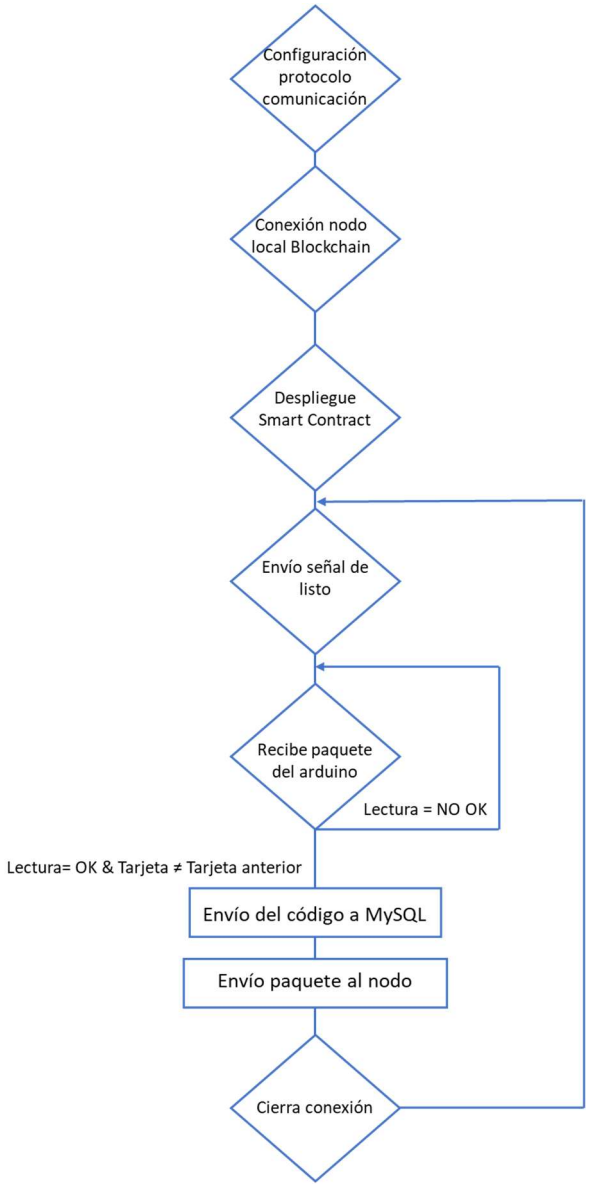


Ilustración 38. Flujo lógico del prototipo final en la Raspberry Pi.

Con la dirección del contrato también es necesario enviar el ABI del contrato, la interfaz, que se genera con el Smart Contract. ABI viene de Application Binary Interface (Interfaz binaria). El contrato se almacena como un código binario en la blockchain bajo una dirección

determinada que es la que se conoce como dirección del contrato. Este ABI se necesita para poder acceder al código binario. El ABI define qué funciones se pueden llamar y garantiza que la función devolverá los datos en el formato que se espera.

Una vez el contrato esté compilado, se pueden acceder a todos los detalles que se necesitan especificar para el programa de la raspberry pi. En la versión web de remix, en el apartado de compilar, se pueden acceder a los detalles de todo el contrato.

Será necesario importar estas características desde la plataforma de remix:

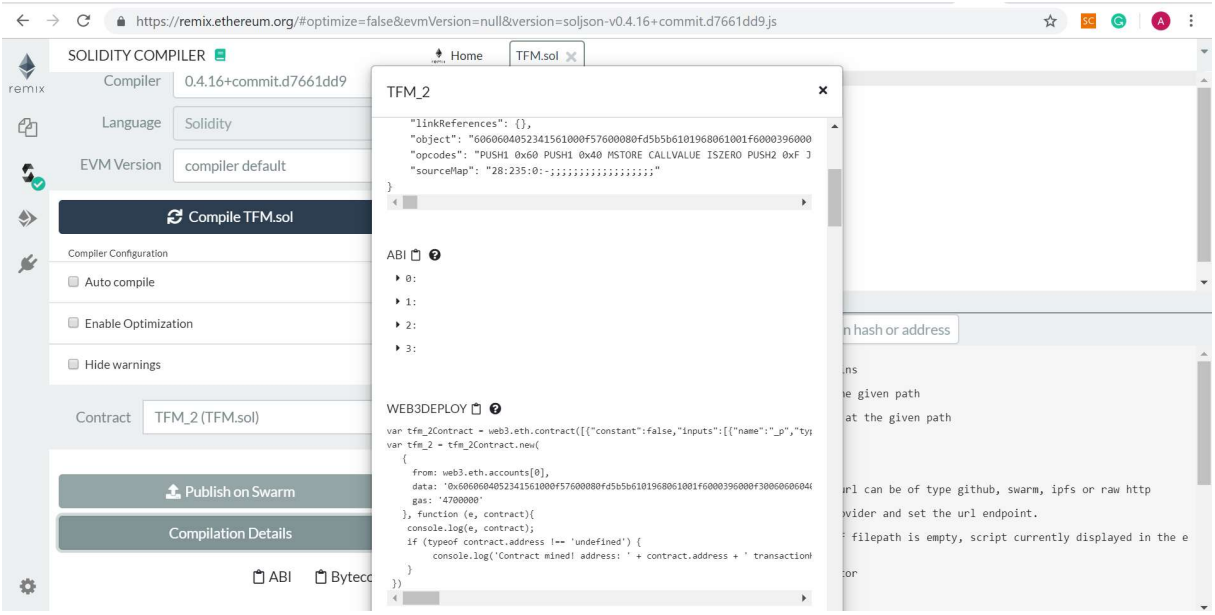


Ilustración 39. Detalles de un contrato inteligente en remix.

Fuente: remix.ethereum.org

La inserción de los datos en la red Ethereum se realiza de la siguiente forma:

```
1.  contractInstance.functions.testContract(aux).transact({'from': account1})
2.  contractInstance.functions.testDate(fecha).transact({'from': account1})
```

Se indica la función y la variable que se quiere enviar, puesto que la función ya había sido programada para recibir una variable como entrada, y se indica la cuenta desde la que se realiza la transacción.

Para que estas dos lineas puedan ejecutarse correctamente, previamente se define:

1. El lugar en el que está alojada toda la información de la red de Ethereum y conectarse a la consola para poder conectarse con el nodo.
2. La dirección del contrato (address1)

3. La cuenta (eth.accounts [0])
4. El ABI del contrato

```
web3 = Web3(Web3.IPCProvider("/media/pi/untitled/nodoLight/geth.ipc"))
address1 = web3.toChecksumAddress('0x7f999f948fb8ea004d2b961e73a2ff7e4a49642b')
account1 = web3.eth.accounts[0]
web3.middleware_stack.inject(geth_poa_middleware, layer=0)

false = False
true = True

abiTFM= [
        {
        "constant": false,
        "inputs": [
        {
        "name": "_p",
        "type": "string"
        }
        ],
        "name": "testContract",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
        },
        {
        "constant": false,
        "inputs": [
        {
        "name": "_d",
        "type": "string"
        }
        ],
        "name": "testDate",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
        },
        //ALL ABI CODE
]
contractClass = web3.eth.contract(abi=abiTFM)
contractInstance = contractClass(address=address1)

```

Una vez desplegado el contrato y el protocolo de comunicación, el servidor envía la señal al Arduino para empezar la transmisión de paquetes. Es precio instalar los sockets, para el envío de la información, definir la IP y el puerto que se hayan establecido en el arduino.

```
import socket
from socket import *
import time
from time import time


```

```
address= ('192.168.1.163', 8888)
client_socket = socket(AF_INET,SOCK_DGRAM)

while(1):
    data = b"codigo"
    client_socket.sendto( data, address)
    try:
            rec_data, addr = client_socket.recvfrom(2048)
            aux=(rec_data)
            if(aux!=0):
                print(aux)
    except:
        pass
```

No todos los códigos leídos por el Arduino se envían a la red de Ethereum y a la base de datos, solo aquellos que sean distintos del último valor leído o distintos de cero, que se supone una lectura incorrecta. Se supone que, en una aplicación real en una línea de producción, cada producto tendrá un código de identificación único y por tanto, de estar leyendo el mismo código de manera continua supone un error en la lectura. Si el código leído es considerado correcto, se envía a la base de datos de MySQL con el día y hora de lectura, y a su vez a la red de blockchain con la fecha y hora detallada en la transacción también. Una vez se envía el código, se cierra la conexión y vuelve a esperar una lectura correcta del próximo código.

```
    if aux!=0:
        if(aux2!=aux):
            try:
                    connection= mysql.connector.connect(host='localhost', database='
Portfolio', user='root', password='ainhoa')
// CONTINUA EL PROGRAMA
```

De esta manera, utilizando variables auxiliares, solo se envía la información a la base de datos y a la red cuando el dato leído es distinto de cero y distinto del código anterior.

Pruebas y análisis económico

Las ventajas que se proponen con el prototipo creado en este proyecto incluyen la rapidez en la lectura de etiquetas y el almacenamiento seguro de la información.

Se han realizado pruebas para la medición de tiempos de lectura y escritura de las etiquetas RFID tanto en la base de datos de MySQL cómo en la Blockchain. Los tiempos de lectura de las etiquetas RFID son de milisegundos, posteriormente el envío de la información a la base de datos de MySQL es también inmediato para este prototipo, aunque en el caso de comunicaciones inalámbricas o despliegue real donde la antena y el servidor estén más distanciadas habría que estudiar la latencia, y de manera paralela la información se manda al nodo de Blockchain.

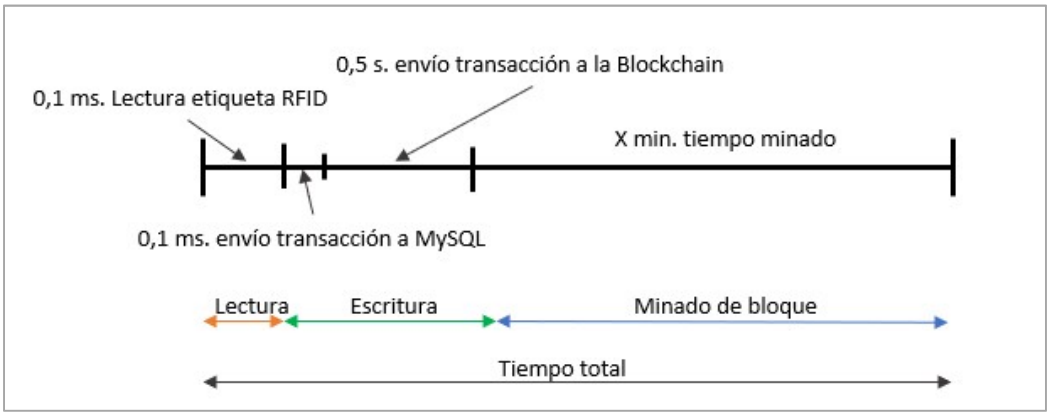


Ilustración 40. Prueba de tiempos de lectura y escritura de etiquetas RFID.

FUNDAMENTOS PARA EL CÁLCULO DEL COSTE:

El tiempo de minado del bloque es el que se debe determinar en función de los costes o urgencia de tener desplegada la información en toda la red. Cada transacción de envío de información no es gratis, sino que supone un gasto en gas. Este Gas es el coste del esfuerzo de los ordenadores para ejecutar las transacciones en el que se tiene en cuenta tanto el consumo eléctrico como el tiempo para realizar la transacción. Cuanto más compleja sea la operación por realizar, se requiere mayor uso de recursos y por tanto mayor será el gasto en Gas. Este gasto permite recompensar a los mineros por los recursos que hayan utilizado y evita procesar transacciones no ejecutables.
En el caso de este prototipo las operaciones a realizar serán de envío de transacciones individuales y de lectura de la información y, por tanto, supone un bajo coste de gas. El envío

de transacciones implica un gasto de 43061 de gas, y el de lectura de 0. Los bloques dentro de la red tienen un límite de gas que pueden procesar que ronda los 8000000 de gas de media, y por tanto una vez superado ese límite, la información se deposita en un nuevo bloque.

La siguiente ilustración muestra el estado de la red Blockchain a jueves 27 de Junio. En el que la media de procesamiento de un bloque es de alrededor de los 14 segundos, y el límite de gas por bloque de 8000000.

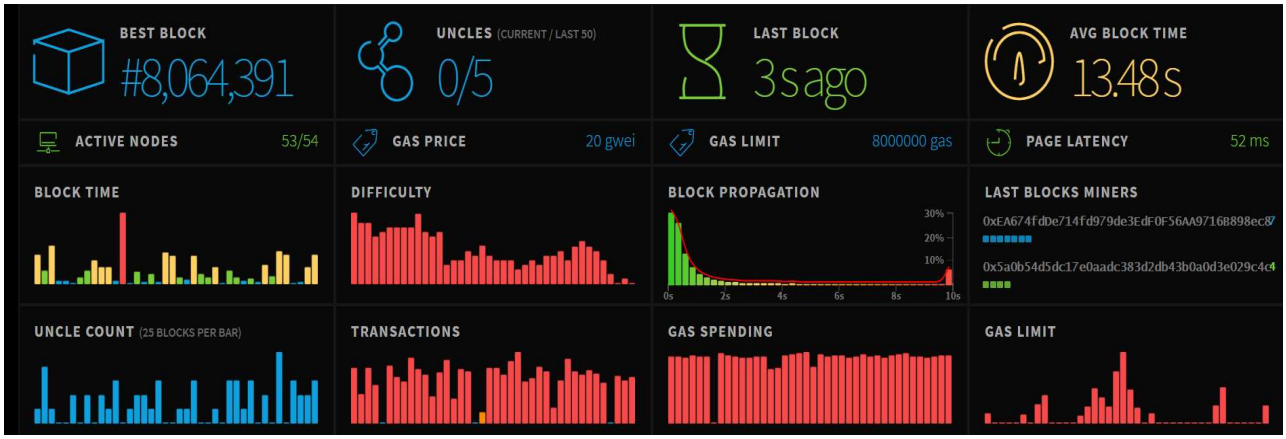


Ilustración 41. Estado de la red Blockchain.

Un nuevo bloque se procesa cada 14 segundos, sin embargo, el tiempo que ha llevado a que el bloque se mine, depende del estado de la red y del coste al que estés dispuesto a incurrir en el proceso. Por ejemplo, a continuación, vemos el estado de la red el mismo 27 de Junio a las 10:34 am, en el que, para minar el bloque en menos de 2 minutos, el coste de cada transacción supondría un $0,25341 y si no existiese urgencia en ver el bloque desplegado en la red de manera rápida, el coste por transacción sería de unos $0,127.

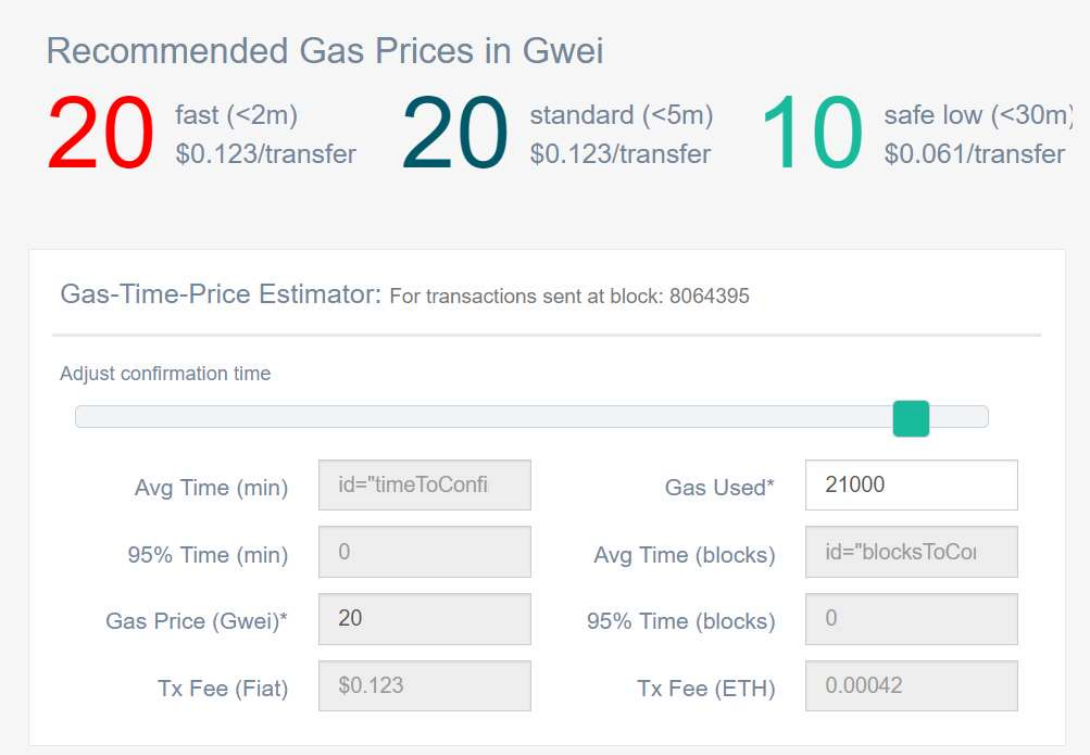


Ilustración 42. Precios en Gas según el estado de la red.

El caso mostrado refleja una situación de la red bastante saturada en el que los costes por transacción son bastante altos. Durante el mismo día se analizó la red y los costes por transacción y se obtuvo la siguiente tabla:

		Tiempo de minado		
		Fast (<2min)	**Standard (<5 min)**	**Low (<30 min)**
Tráfico en la red	**Alto**	0,25341	0,25341	0,25341
	Medio	0,12672	0,0642	0,01521
	Bajo	0,055	0,038	0,01394

Tabla 11. Coste en $ por transacción.

Para obtener el coste de las transacciones por tanto depende de: lo congestionada que esté la red en ese momento, y el "gas Price" de las transacciones que determinarán la prioridad a la hora de ejecutarlas. Cuanto mayor sea la tasa que pagar, más rápido será la aprobación de la transacción.

Los mineros recolectan comisiones por cada transacción procesada, y por tanto competirán con los demás mineros para conseguir aquellas transacciones con mayores ganancias. Esto no implica, que sea necesario establecer una cantidad de gas alta para asegurar las transacciones, puesto que si una única transacción ocupa mucho espacio de un bloque (Cómo se ha comentado, cada bloque tiene un límite de gas de alrededor de 8000000), para un minero puede no ser rentable procesarla puesto que reconocerá que no puede recolectar mucho GWEI.

El gwei (de giga wei) equivale a 0.000000001 ETH. El wei se considera como la unidad más pequeña del Ether, como un céntimo, de hecho 1 Ether equivale a 10^{18} wei, por ello se utiliza Gwei cómo la unidad más común para hablar de costes de transacción. [43]. El gwei se multiplica por el precio de gas de la transacción en concreto del contrato inteligente desplegado y posteriormente por el precio del Ether en ese momento.

$$Precio\ de\ la\ transacción = gas\ de\ la\ transacción\ x\ GWEI\ x10^{-9} x\ precio\ del\ ether$$

En función de la prioridad de la red a la hora de ejecutar la transacción se obtienen los diferentes precios mostrados en las Tablas 5 y 6, teniendo en cuenta que el precio del gas de nuestra transacción es de 43061, se multiplica por el Gwei dependiendo de la saturación de la red y rapidez requerida para ver reflejada la información en la red y por el precio del Ether.

		Tiempo de minado		
		Fast (<2min)	**Standard (<5 min)**	**Low (<30 min)**
Tráfico en la red	**Alto**	20	20	20
	Medio	10	4	2
	Bajo	8,8	3	1

Tabla 12. Coste en Gwei por transacción

Actualmente, el precio del Ether ronda los 280USD, aunque sus máximos históricos han llegado incluso a los 1448USD*. Durante los últimos seis meses, el precio del ether no ha hecho más que subir:

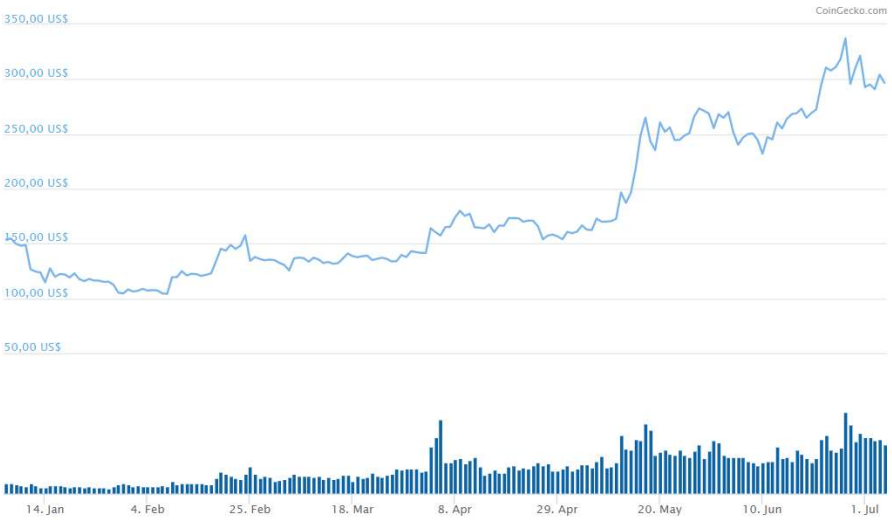


Ilustración 43. Precio del Ether entre Enero-Julio 2019.

Fuente: https://www.coingecko.com/es/monedas/ethereum

Para el caso de estudio reflejado en esta memoria, se han tomado valores del Ether de 291US$.

Para entender el coste asociado a enviar las lecturas de manera constante se han hecho los siguientes cálculos, considerando flujos de entrada de productos que pueden variar desde 1 producto a la hora hasta 45000 productos/hora. Además como se ha explicado anteriormente, cada bloque tiene un límite de gas y, por tanto, teniendo en cuenta que cada una de nuestras transacciones de manera individual ocupa 43061 de gas, el límite de transacciones por bloque está limitado y el número de productos a recepcionar también. Se analizan diferentes casos desde el más optimista en el que el 100% del bloque almacene la información leída por el Arduino, a que la información suponga únicamente un 20% del bloque.

En la tabla se muestran las transacciones que puede gestionar un bloque a la hora. En el momento del estudio, se procesaba un bloque cada 15 segundos, y por tanto en cada bloque si se llenase al 100% cada 15 segundos podría recepcionar 185 productos, que a lo largo de una hora serían hasta casi 45000. Sin embargo, es poco realista pensar que se puede completar un bloque al completo, puesto que ningún minero querría competir por esas transacciones, y acabaría tardando mucho el proceso de minado.
Se han calculado las transacciones máximas procesadas en función del llenado de bloque, obteniendo un mínimo de productos a recepcionar cerca de los 9000 a la hora.

Gas limit	8000000
Avg. block time (seg)	14,97
Gas por transaccion	**43061**

		nº transacciones en un bloque	En una hora	En media hora
BLOQUE	**100%**	185	44488	22244
	90%	167	40160	20080
	80%	148	35591	17795
	60%	111	26693	13346
	40%	74	17795	8897
	20%	37	8897	4448

Tabla 13. Transacciones admisibles por hora en función del llenado del bloque.

Para los distintos casos de estudio se supone que los productos llegan de manera uniforme de forma que en las gráficas que se presentan a continuación el producto N llega cuando t = Nx(hora/productos) en hora. A continuación, se presentan los resultados obtenidos en el caso más optimista de llenado de un bloque al 100%, un caso intermedio de llenado al 80%, y uno en el que las transacciones supongan tan solo el 20% del bloque.
El número máximo de productos a analizar a la hora sucede en el caso optimista de llenado de bloque al 100%. Con 45000 productos/hora, al año considerando solo los días laborables y el funcionamiento de la línea 12 horas al día, supondría un total de producción anual de 135 millones de productos en una única linea, lo que supone comparar esta producción con la de

una gran industria. El caso menos optimista, en una misma línea el prototipo sería capaz de almacenar 27 millones de productos al año. Por tanto, en todos los casos que se discuten a continuación suponen producciones muy elevadas, seguramente proporcionales a lo que serían industrias de medio y gran tamaño.

Los costes que supondrían enviar individualmente las transacciones de los productos se han calculado atendiendo a dos variables:

1. Congestión de la red: cómo se ha reflejado en *Tabla 11. Coste en $ por transacción.* , en función de la red los costes por transacción pueden variar en gran medida. Se ha analizado 3 momentos de la red distintos, categorizados como congestión alta, media y baja.
2. Tiempo reflejado en la red: un bloque es minado cada 15 segundos, sin embargo, el proceso previo puede suponer desde escasos minutos hasta 30 minutos. Por tanto, dependiendo de la urgencia a la hora de verse reflejada la información en la red se pueden incurrir en diferentes costes. Por ello, se han analizado diferentes escenarios, en los que se precisa la información a la media hora , a la hora, y más de una hora.

ESCENARIO OPTIMISTA: 100% BLOQUE

Primero analizamos el caso ideal en el que el bloque pueda ser completado en su totalidad de gas por las transacciones que envían los códigos de las etiquetas.

Para este caso, si el escenario planteado requiriese de visualizar la información en la red a la media hora de comenzar las lecturas el número máximo de productos a procesar sería de 22500 productos y los costes totales en los que se incurriría dependiendo de la congestión de la red serían los siguientes:

Limite	ALTA	MEDIA	BAJA
30 mins.	$ 5.701,25	$ 1.757,1	$ 982,50

Tabla 14. Coste incurrido por transacciones reflejadas en 30 minutos.

Cómo se observa en la *Tabla 14. Coste incurrido por transacciones reflejadas en 30 minutos.*, atendiendo a la congestión de la red, el coste puede aumentar hasta en un 580% cuando la congestión de la red es alta.

Los costes mostrados en la tabla surgen del caso en el que se busca que desde el momento inicial de lectura y a los 30 minutos, se puedan ver reflejados hasta 22500 productos en la red, lo que supone 12 productos al segundo.

Se tendrá en cuenta que, al querer los productos reflejados a los 30 minutos de empezar la primera lectura, los primeros bloques si se podrán minar a una velocidad media de menos de 5 minutos, pero será necesario que los últimos bloques se minen rápido (velocidad alta de media dos minutos de minado) para verse reflejados con la urgencia que se requiere.

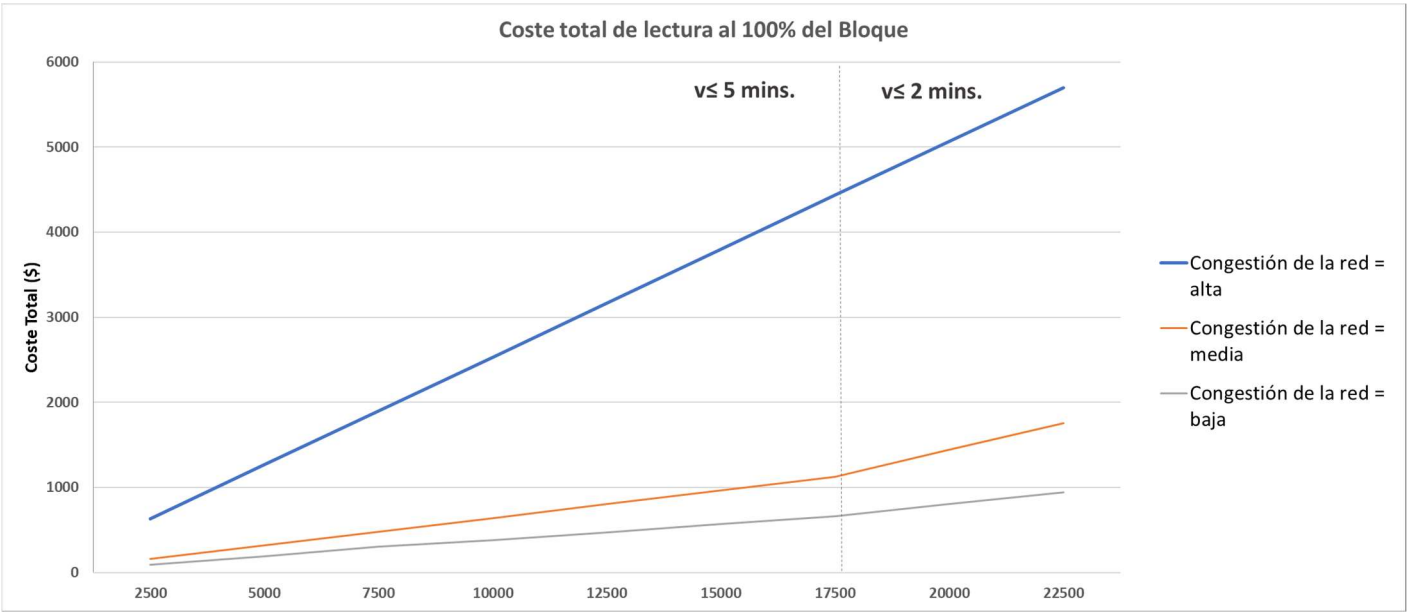


Ilustración 44. Coste por 22500 productos en función de la red.

En caso de la red estar congestionada, como el precio por las transacciones no varía según las velocidades de minado, independientemente del número de productos, el coste es proporcional al número de productos leídos por el precio estándar de la transacción durante el periodo de tráfico alto en la red, es por ello qué en la Ilustración 39, la linea de costes para un tráfico de red alto es totalmente lineal, independientemente de la velocidad de minado. Sin embargo, para un tráfico medio o incluso bajo de la red, los precios entre los distintos tiempos de minado si varían, y por tanto, los primeros productos pueden minarse más lentamente obteniendo una pendiente de costes inferior a la curva de costes al minar los últimos productos a una mayor velocidad.

Aun así siempre que la situación requiriese visualizar la información en periodos inferiores a 30 minutos, la velocidad de minado se va a limitar a media o alta y por tanto los gastos en los que se incurra serán más altos. Por tanto, si no fuese necesario mostrar los productos de manera

tan urgente y, se pudiesen intervalos de hasta 1 hora o de más de una hora desde el momento inicial de lectura el precio disminuye considerablemente:

	ALTA	MEDIA	BAJA
30 mins.	$ 5.701,25	$ 1.757,51	$ 982,50
1 hora	$ 5.701,25	$ 342,23	$ 313,65
+ 1 de hora	$ 5.701,25	$ 342,23	$ 313,65

Tabla 15. Coste por 22500 productos en diferentes intervalos de tiempo.

Cómo se puede observar en un estado de la red bajo, con precios de 1Gwei por minar en menos de 30 minutos, el precio se reduce más de la mitad. Por el contrario, en momentos de congestión alta de la red, la velocidad de minado no influye en el precio de las transacciones y por tanto el coste es independiente de la velocidad, siendo bastante elevado.

Dentro de los dos escenarios de tiempo de una hora o más de una hora de la Tabla 11 no existen diferencias puesto que para ambos casos los 22500 productos pueden minarse en todo momento a baja velocidad. El coste medio por transacción en cada uno de los casos reflejados en la Tabla 11 serían los siguientes:

	ALTA	MEDIA	BAJA
30 mins.	$ 0,25341	$ 0,0781	$ 0,04366
1 hora	$ 0,25341	$ 0,0152	$ 0,01394
+ 1 de hora	$ 0,25341	$ 0,0152	$ 0,01394

Tabla 16. Coste medio por transacción en función de tráfico de red y tiempo de minado.

Para congestión alta el precio medio no varía, y para reflejar la información en 30 minutos, el precio medio por transacción está entre los valores de minado medio y alto de la Tabla 7, puesto que los primeros bloques se minan a una velocidad media de 5 minutos y los últimos a velocidad media de 2 minutos.

En la figura que se muestra a continuación se ve reflejado los costes (mostrados en la Tabla 11) en los que se incurre dependiendo de la llegada de entre 1 a 22500 productos, y dependiendo de la rapidez de minado que se exige.

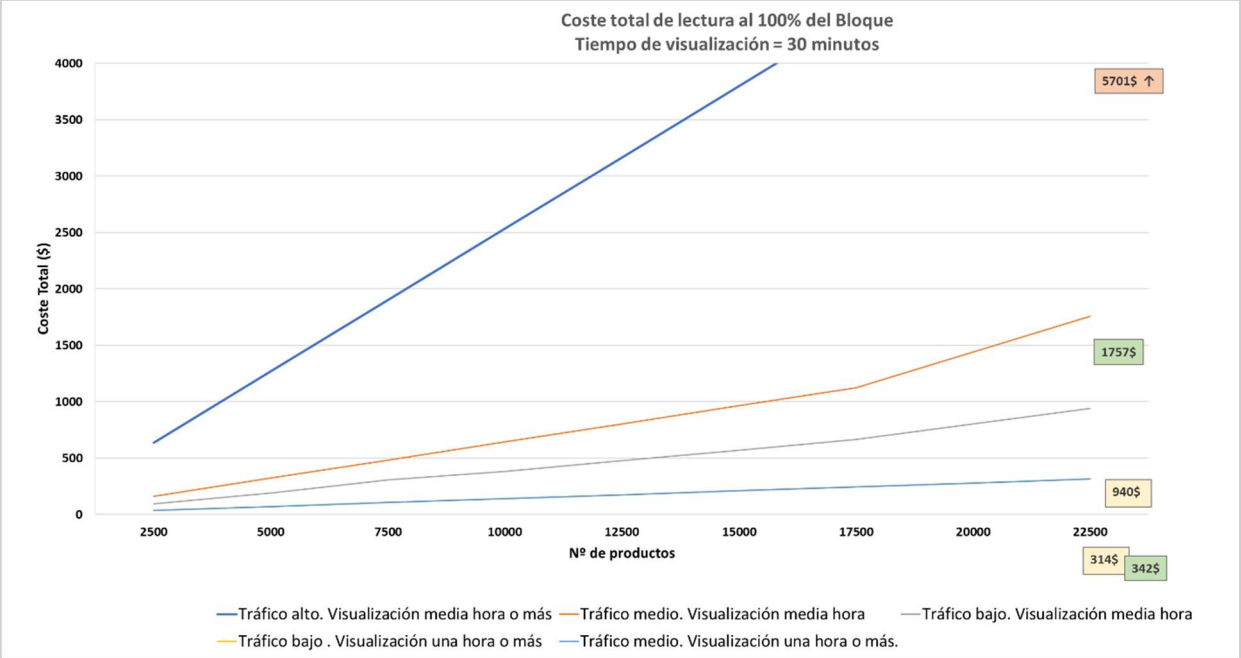


Ilustración 45. Coste por 22500 productos en función de la red y el tiempo de minado.

En esta figura respecto a la anterior, se añaden las curvas de coste si el tiempo para visualizar la información desplegada en la red fuese más flexible. Cómo se observa en la figura, cuando la lectura de productos es inferior a 17500 productos, la línea de costes es lineal para todos los casos. Sin embargo, llega un momento de lectura en el que sería necesario minar más rápido si se exige 30 minutos desde el inicio de la primera lectura, y los costes aumentan proporcionalmente al coste asociado a la nueva velocidad de minado. En el caso, de disponer de un tiempo flexible de visualización de una hora o más se aprecia en las curvas de los costes la importante disminución de precios. Se ha utilizado una leyenda de precios en la gráfica mostrando en amarillo el coste si la congestión de la red es baja, en color verde los precios si la red está en funcionamiento habitual, y en naranja el coste si la red está muy congestionada.

Cómo se ha mencionado la curva de costes para un tráfico de la red muy alto se mantiene lineal independientemente de la velocidad de minado, y es por ello, que se ha decidido mostrar en mayor detalle los casos de media y baja velocidad, y se ha ajustado el eje de costes para poder visualizarlo.

Con esos costes y tiempos de minado, si se completarán los casi 45000 productos capaces de procesar en una hora los costes serían los siguientes:

	ALTA	MEDIA	BAJA
1 hora	$ 11.403,45	$ 2.099,25	$ 1253,65
+1 horas	$ 11.403,45	$ 684,45	$ 627,3

Tabla 17. Coste por máximo de transacciones a la hora.

Los precios con un tráfico de red alta son muy elevados y constantes independientemente del tiempo de minado. En un escenario con un tráfico de red medio (el más usual), los costes por hora en los que tendría que incurrir la planta serían de hasta 2100€. Los costes medios por transacción se muestran en la siguiente Tabla:

	ALTA	MEDIA	BAJA
1 hora	$ 0,25341	$ 0,0466	$ 0,01394
+ 1 de hora	$ 0,25341	$ 0,0152	$ 0,01394

Tabla 18. Coste medio por transacción con número máximo de productos (45000)

En caso de disponer de más de una hora para ver la información desplegada en la red, los costes por transacción se ajustan al coste por transacción mínimo impuesto en la red en ese momento. Con el caso límite de una hora, el coste medio por transacción se trata de una media no aritmética entre los costes por transacción en ese momento de la red en las tres velocidades de minado distintas. La mayoría de los bloques pueden minarse a velocidad baja y media, y es por eso que el coste medio por transacción está más próximo al coste a velocidad standard de minado.

El importe al poder aumentar el tiempo de minado, con el mismo número de productos a leer, disminuye notablemente. En el primer caso en el que se exigía tener la información disponible a los 30 minutos, el tiempo de minado mínimo tenía que ser de 5 minutos, pero al poder aumentarlo a 30 minutos como en este caso, el precio de las transacciones con una red de tráfico normal se reducen a la mitad. Los primeros 22500 bloques pueden minarse a la mínima velocidad, y es por ello por lo que el coste se reduce.

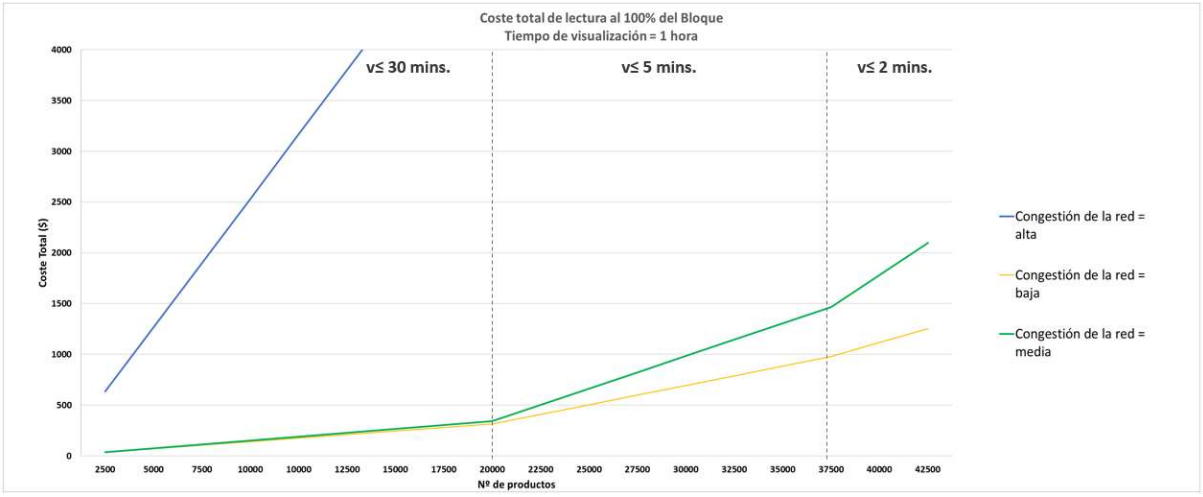


Ilustración 46. Coste incurrido por transacciones reflejadas a la hora al 100% del Bloque.

En esta Ilustración se observan los tres cambios de pendiente puesto que se muestra el caso límite con el número máximo de productos a procesar.

De este análisis se puede obtener una primera conclusión, que aunque las transacciones de manera individual sean baratas, al estar estudiando un sistema con lectura constante de códigos y por tanto, envío de los mismos la tarifa para la empresa aumentaría frente a una simple digitalización del proceso. Sin embargo, si no se requiere que la información esté desplegada en la red de manera inmediata, el caso de estudio para la empresa es mucho más económico, puesto que permite un minado lento sea cual sea el estado de la red, incluso se puede evitar minar en aquellos momentos en los que la red esté congestionada.

ESCENARIO REALISTA: 80% BLOQUE

Para este caso, si el escenario planteado requiriese de visualizar la información en la red a la media hora de comenzar las lecturas, los costes en los que se incurriría dependiendo de la congestión de la red serían los siguientes:

	ALTA	MEDIA	BAJA
30 mins.	$ 4434,67	$ 1.436,1	$ 750

Tabla 19. Coste incurrido por transacciones reflejadas en 30 minutos al 80% del Bloque.

En este caso, el número de transacciones en el bloque disminuyen a 150 por bloque y por tanto, de requerir una velocidad de minado alta para que la información esté desplegada en la red a la media hora, se limita el número de productos de lectura a 18000 productos cada media hora.

Respecto al caso optimista anterior, la mayor diferencia se observa en el número de productos a procesar pero el comportamiento de los costes es similar.

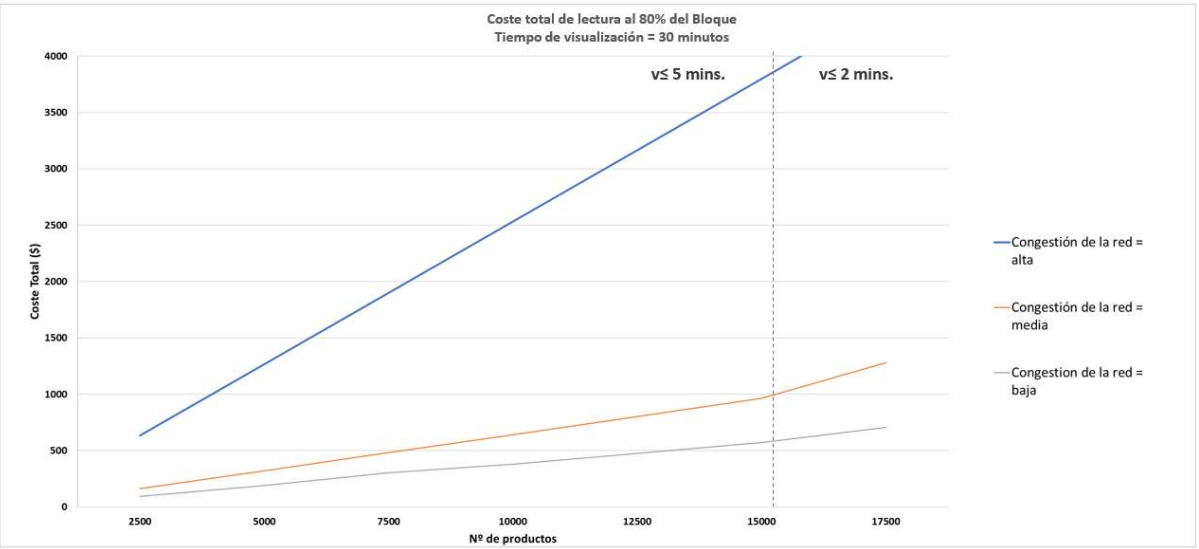


Ilustración 47. Coste incurrido por transacciones reflejadas en 30 minutos al 80% del Bloque.

El comportamiento de los costes es totalmente similar al caso anterior. La mayor diferencia es el número máximo de transacciones (productos) que pueden almacenarse en un único bloque. Ante una congestión alta de la red, se observa el mismo comportamiento lineal proporcional al número de productos leídos. Por otro lado, tanto para un tráfico medio cómo para un tráfico bajo se necesita minar a dos velocidades distintas, y por tanto dos pendientes de costes distintas.

	ALTA	MEDIA	BAJA
30 mins.	$ 0,25341	$ 0,08206	$ 0,04285

Tabla 20. Coste medio por transacción al 80% del Bloque.

Los costes medios por transacción son muy similares al caso anterior, y las pequeñas diferencias vienen de la aproximación que se ha hecho en los cálculos para trabajar en series de 2500 productos.

Si se muestran los costes máximos al procesar un número máximo de 17500 productos, el resultado es el que se muestra en la siguiente ilustración. De la misma forma que en el caso anterior, se han utilizado leyendas para los costes máximos en naranja para el tráfico mayor de red, verde para el más habitual, y amarillo para una congestión de la red muy baja. Con 17500 productos, si existiese flexibilidad de visualización se minan a velocidad mínima en todo momento y por tanto, las curvas de costes son lineales y los precios resultantes los más bajos dentro de cada categoría de congestión de la red.

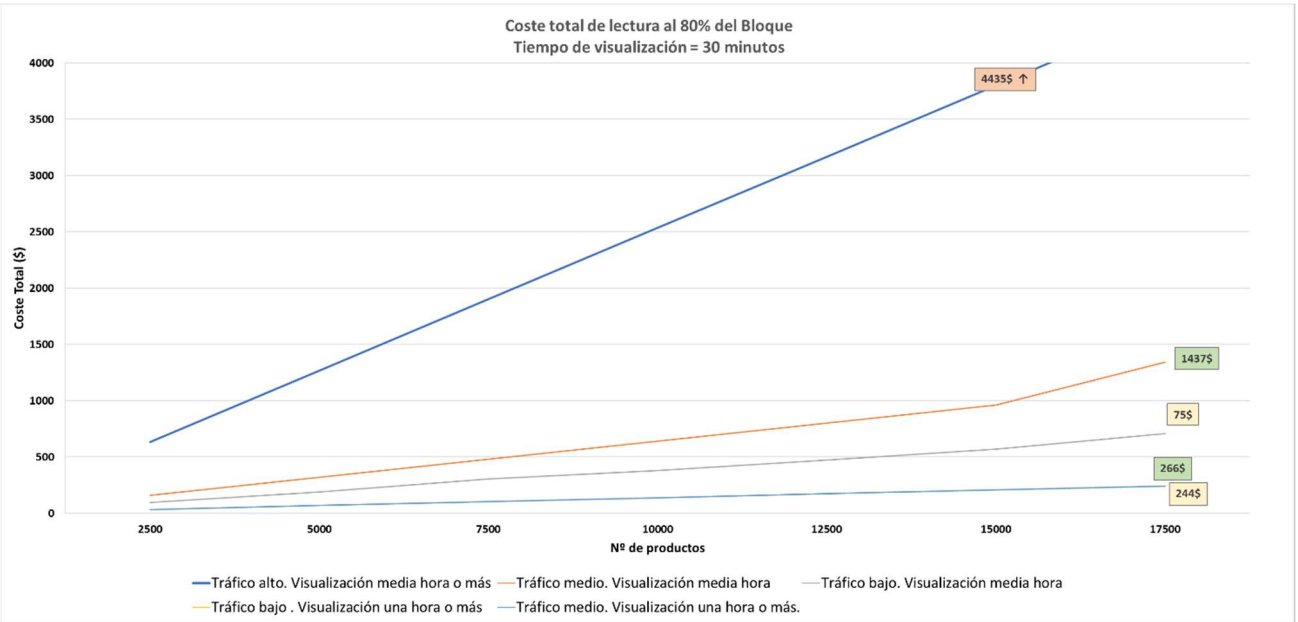


Ilustración 48. Coste por 17500 productos en función de la red y el tiempo de minado.

En este caso de envío de transacciones sin superar el 80% del bloque, el límite de productos a procesar está en 36000 a la hora, y los costes por verlo reflejado a la hora o más son los siguientes:

	ALTA	MEDIA	BAJA
1 hora	$ 8869,35	$ 1828,99	$ 1048,95
+ 1 horas	$ 8869,35	$ 532,35	$ 487,5

Tabla 21. Coste por máximo de transacciones a la hora

Con una media de lectura a la hora de 36000 productos, y buscando ver reflejada la información en la red al finalizar la hora, los costes se comportan de la siguiente manera:

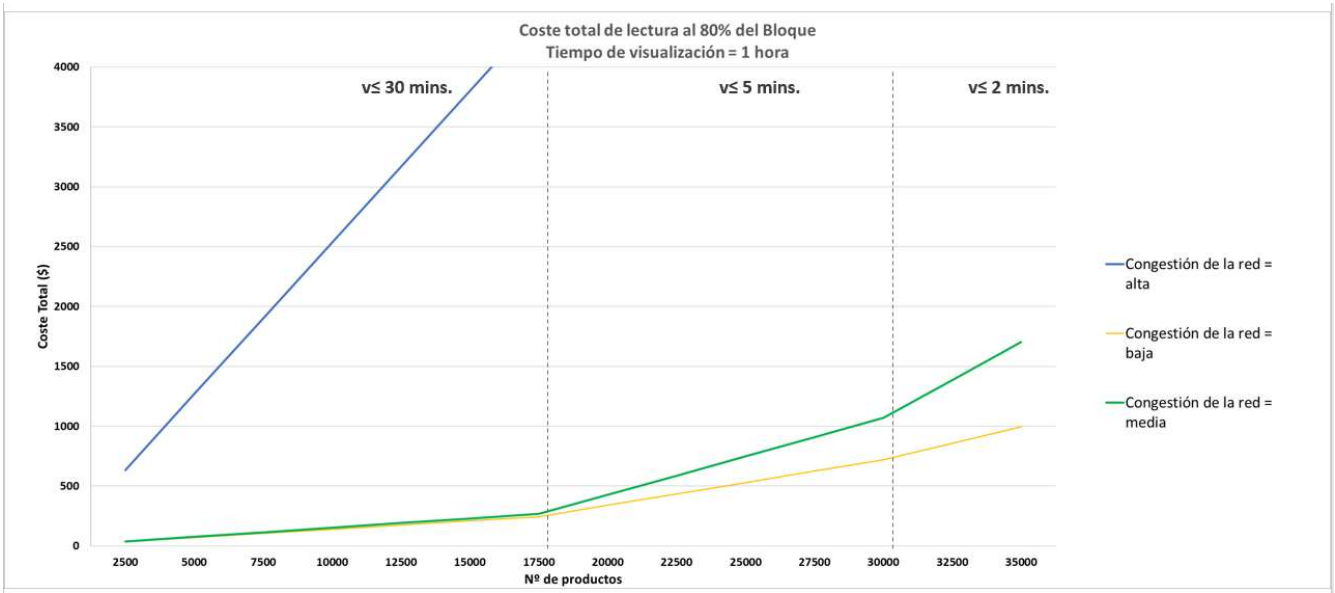


Ilustración 49. Coste total por lectura de 36000 productos/hora

En esta Ilustración se observan los tres cambios de pendiente puesto que se muestra el caso límite con el número máximo de productos a procesar. En la Ilustración 42, al tratarse de un número inferior de productos que el máximo capaz de almacenar en el caso de estudio, las velocidades de minado pueden permanecer bajas. En este caso los primeros productos, y por tanto, los primeros bloques pueden minarse a velocidad baja, a partir de un momento los productos ya requieren una velocidad de minado más alta, y los últimos la velocidad máxima de minado disponible en la red.

Para los estados de red de media y baja congestión son muy visibles los cambios de pendiente con relación al cambio de precio por exigir una mayor velocidad de minado. Al necesitar los productos reflejados en la red al finalizar las lecturas, permite que los primeros bloques con hasta 150 transacciones vayan minándose a velocidad baja y por tanto coste inferior, hasta que sobrepasada la media hora, un minado del bloque a velocidad baja podría suponer no tener disponible la información en el tiempo establecido. Es en ese momento en el que es necesario minar a una velocidad superior y la pendiente se inclina, hasta que los últimos códigos que se identifiquen y una vez todas las transacciones estén en el bloque es necesario que éste mine a la mayor velocidad posible, detectando en la gráfica otro aumento de pendiente.

ESCENARIO: 20% BLOQUE

Para este caso, si el escenario planteado requiriese de visualizar la información en la red a la media hora de comenzar las lecturas, los costes en los que se incurriría dependiendo de la congestión de la red serían los siguientes:

	ALTA	MEDIA	BAJA
30 mins.	$ 1140,34	$ 320,16	$ 179,5

Tabla 22. Coste incurrido por transacciones reflejadas en 30 minutos al 20% del Bloque.

En este caso, el número de transacciones en el bloque disminuyen a 50 por bloque y por tanto, de requerir una velocidad de minado alta para que la información esté desplegada en la red a la media hora, se limita el número de productos de lectura a 4500 productos cada media hora.

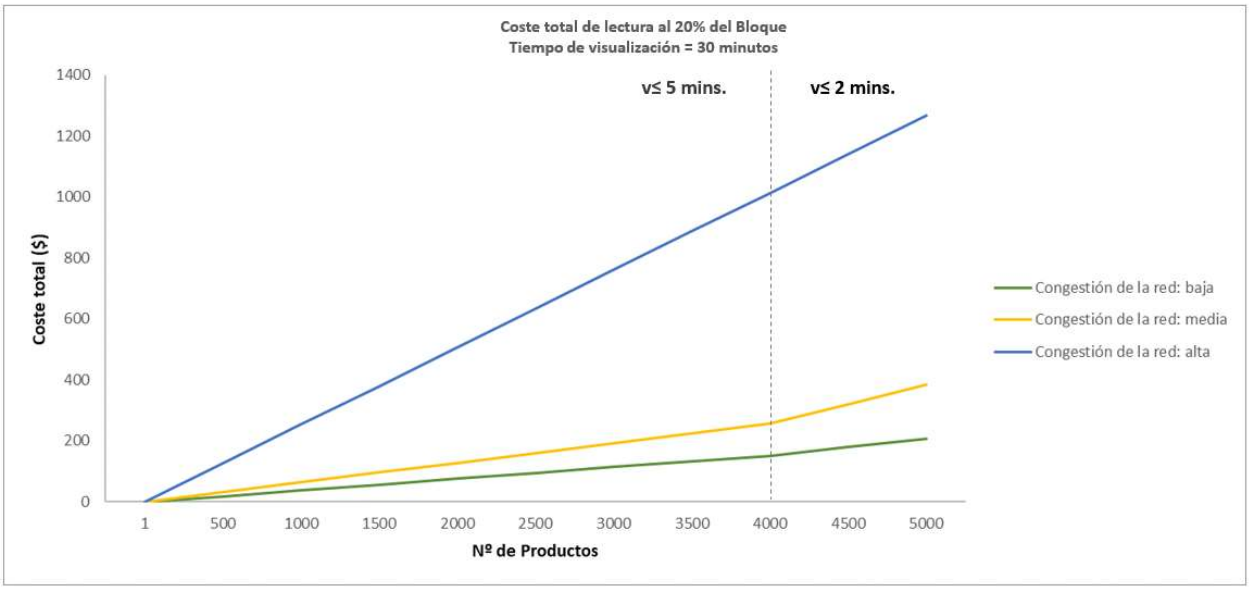


Ilustración 50. Coste por transacciones reflejadas en 30 minutos al 20% del Bloque.

Por tanto, al ir disminuyendo la capacidad de llenado de los bloques, la cantidad de productos a recepcionar disminuye notablemente, desde 22500 productos cada media hora al 100% hasta 4500 productos. En este último caso, más realista si hablamos de productos que salen de una línea de producción a una media 3 productos al segundo, el coste total por hora en un estado de congestión de la red medio o incluso bajo, sería de unos 420$ de media, que a lo largo de un día laboral supondrían de unos 4200$.

Resultados, Conclusiones y Desarrollos futuros

Cómo se ha detallado a lo largo del proyecto, las aplicaciones del Blockchain son infinitas y en todos los sectores. Este proyecto se ha centrado en las aplicaciones para la industria de la logística, ya sea para almacenamiento, control de la producción, o muchas de las actividades relacionadas con el transporte de mercancías.

A continuación se resumen algunos de los resultados de este proyecto sobre el análisis económico del prototipo junto con un listado de los pasos necesarios para almacenar información en la Blockchain.

Como principales resultados se destacan:

- Síntesis de los pasos necesarios para el almacenamiento de información en la blockchain

Para poder almacenar la información en la red Ethereum será necesario:

1. Sincronizar la cadena de bloques que requiere de un importante uso de memoria, y por tanto, dependiendo de la placa computadora que se utilice será preciso usar memorias externas.
2. Una vez con el nodo sincronizado, crear una cuenta con Ether para poder realizar las transacciones.
3. Crear el contrato inteligente utilizando Remix.
4. El programa para interactuar con la red tendrá que detallar la dirección del contrato, la cuenta desde donde se realiza la transacción, el ABI del contrato, y la dirección del nodo. Con todos los detalles definidos desde el principio, al enviar las transacciones se detalla la función dentro del contrato que se quiere ejecutar, junto con la cuenta de origen del envío.

- Desarrollo de un prototipo completo

Los pasos imprescindibles seguidos a lo largo del proyecto:

1. Configurar el hardware con la placa Arduino para la lectura de los códigos RFID.
2. Configurar las conexiones entre la placa microcontroladora y el servidor.
    1. Será necesario encontrar un dispositivo, para permitir las conexiones, en el caso de este proyecto la placa de ethernet.
    2. Definir el protocolo de comunicación. Para este proyecto UDP.
    3. Asignar IPs y puertos para la comunicación
3. Configurar la base de datos en el servidor.
    1. Instalar el servidor web. Para este proyecto Apache.
    2. Instalar el sistema de bases de datos. Para este proyecto MySQL.
    3. Definir las tablas donde almacenar la información.
4. Configurar el nodo Blockchain en el servidor.

- Análisis económico

Las conclusiones obtenidas una vez realizadas las pruebas muestran la viabilidad de este tipo de prototipos para la industria de la logística. Sin embargo, el coste computacional ligado al envío de las transacciones de manera individual a la red puede parecer elevado. Esto es porque, aunque el consumo de gas ligado al envío de tarjetas RFID es bajo, si se aplicase al control de producción de líneas de grandes fábricas, el volumen de productos a procesar es tan elevado que aumenta el coste asociado. Es importante, que la información no se precise reflejada en la red de manera inmediata sino a lo largo del día de producción, de manera que los costes se reduzcan de manera considerable. Además para poder reducir el gasto es importante considerar aquellos momentos en los que la red está menos saturada, puesto que la tarifa aumenta en hasta un 500%.Por otro lado, si se observan multitud de aplicaciones en las que el coste asociado sería menor puesto que requeriría de menor número de transacciones diarias, por ejemplo, envío de localización de productos en ciertos intervalos de tiempo. En el prototipo actual sería cambiar el sensor RFID, por uno de localización u otro de temperatura.

Los costes asociados a un desarrollo como el realizado en este trabajo son solo con relación al envío de información, y por tanto, la consulta de esta no supone ningún gasto.

Para desarrollos futuros:

2.4. Limitar el acceso a posible información sensible utilizando una red privada.En el proyecto se ha utilizado una red pública que implica que aquellos usuarios conectados a la red puedan acceder a la misma información, que en el caso de ser competidores, la información a la que tendrían acceso podría llegar a ser muy sensible.

2.5. Uso de placas computadoras más potentes. Las pruebas realizadas están limitadas por el uso de una Raspberry como servidor. Sincronizar la cadena de bloques entera supone un consumo de memoria muy elevado y de potencia para el cual la raspberry no está preparada. Son muchos los problemas a la hora de crear los entornos de trabajo que puede tener raspberry. Es importante, como se ha mencionado anteriormente que las pruebas futuras con prototipos de este estilo se utilicen discos duros extraíbles en los que depositar la información, además permite que ante posibles reinicios de la raspberry no se pierda información.

Se pueden sugerir distintas líneas de estudio después de lo expuesto en este trabajo:

i. Ampliar las funcionalidades del prototipo. Por ejemplo, incluir:
   1. Señal de GPS, útil para el sector del transporte, tanto para distribuidores como para clientes finales, que tienen mayor control de las entregas.
   2. Sensor de temperatura, útil para el sector farmacéutico o de la alimentación, que requieren temperaturas determinadas para la conservación de los productos. Para asegurar una llegada óptima de los

productos, cualquier sensor que además de la temperatura pueda detectar cambios de presión o acústica aportaría información importante.

ii. Diseño propio de la parte del microprocesador

iii. Optimizar el gas utilizado. Existen maneras de optimizar los contratos, desde el tipo de variables a utilizar (uint8 utiliza más gas que la variable uint256, cuando las dos se tratan de enteros)

iv. Crear una red privada. En este trabajo se ha desarrollado en una red pública a la que cualquiera tiene acceso para escribir o participar. La seguridad en estas reside en que al estar totalmente descentralizadas no se puede cambiar la información, una vez se haya validado. En las redes privadas, sin embargo, es necesario dar accesos a la red y a las transacciones para poder participar.

v. Incluir funcionalidades como el cobro automático cuando el producto salga de tienda o la creación de albaranes.

BIBLIOGRAFÍA

[1] Los retos de la logística del futuro, AECOC 2017. Disponible en: https://www.aecoc.es/articulos/los-retos-de-la-logistica-del-futuro/

[2] OBS Business School, Qué es logística y por qué se confunde con gestión de cadena de suministro. [https://www.obs-edu.com/es/bloginvestigacion/operaciones/que-es-logistica-y-por-que-se-confunde-con-gestion-de-cadena-de-suministro]

[3] TRAZA. Identificación, Movilidad, Codificación y Marcaje. Transparencia y trazabilidad en el almacén. Disponible en: [https://www.traza.com/blog/post/comopuede-trazabilidad-ayudarme-a-mejorar-los-procesos-en-mi-almacen/]

[4] Deloitte, "Using blockchain & Internet-of-Things in supply chain traceability". Disponible en: [https://www2.deloitte.com/content/dam/Deloitte/lu/Documents/technology/lublockchain-internet-things-supply-chain-traceability.pdf]

[5] Martin & Servera, Katios Future, Axfoundation & SKL Kommentus. "Blockchain use cases for food traceability and control". Disponible en: [https://www.sklkommentus.se/globalassets/kommentus/bilder/publication-engblockchain-for-food-traceability-and-control-2017.pdf]

[6] Minsait, "Cómo impacta blockchain en la Logística 4.0". Disponible en: [https://www.minsait.com/sites/default/files/newsroom_documents/informe_blockchain_logistica_uno_e_0.pdf]

[7] _Advisor Abbate. Definición de Blockchain. Disponible en: [https://advisorabbatees.wordpress.com/2017/06/11/que-es-el-blockchain/]

[8] DHL trend research, "Blockchain in logistics". Disponible en: [https://www.logistics.dhl/content/dam/dhl/global/core/documents/pdf/glo-coreblockchain-trend-report.pdf]

[9] Bitcoin exchange Guide. https://bitcoinexchangeguide.com/blockverify/

[10] Mediledger. Disponible en: https://www.mediledger.com/solution-protocols

[11] IBM Food TRUST: trust and transparency in our food. disponible en: https://www.ibm.com/blOCKCHAIN/solutions/food-trust

[12] S Group. Disponible en: https://www.s-kanava.fi/web/s/en/s-ryhma-lyhyesti

[13] Diamantes, blockchain y bancos: la historia de Everledger. Disponible en: https://www.bbva.com/es/diamantes-blockchain-y-bancos-la-historia-de-everledger/

[14] Everldger, página oficial : https://www.everledger.io/

[15] DS Smith Software Tracking-pack. Información disponible en : https://www.dssmith.com/es/tecnicarton/productos/embalaje-LOGISTICO/SOFTWARETRACKING-PACK

[16] Fallah, Michael. 2016, Nov 24. Traxens: "Smart Containers & Multimodal Transport. Disponible en: http://actionsincitatives.ifsttar.fr/fileadmin/uploads/recherches/geri/sticits/2016-11-24/FALLAH.pdf

[17] Traxens technology. Disponible en: http://www.traxens.com/en/technology

[18] Cadena de suministro noticias. 'TradeLens' la solución de Blockchain de Maersk para mejorar la transparencia en la cadena de suministro. Disponible en: http://www.cadenadesuministro.es/noticias/tradelens-la-solucion-de-blockchain-demaersk-para-mejorar-la-transparencia-en-la-cadena-de-suministro/

[19]Tradelens solución. Disponible en: https://www.tradelens.com/solution/

[20] Top Blockchain use cases for Supply Chain Management. https://www.verypossible.com/blog/top-blockchain-use-cases-for-supply-chainmanagement

[21] Smartlog, blockchain logistics.https://smartlog.kinno.fi/articles/project-smartlogblockchain-logistics

[22] BHP Billiton : https://blockapps.net/wp-content/uploads/2018/05/BHP-BlockAppsUse-Case.pdf

[23] García Guzmán, Daniel. "Potencial de la tecnología blockchain en el mercado eléctrico".

[24] Riddle and Code Disponible en: https://www.riddleandcode.com/faq/

[25] Guerrero, Francisco: Seguridad en Sistemas de Información. Disponible en: https://blog.bi-geek.com/seguridad-sistemas-informacion-parte-ii/

[26] Echebarria Saenz, M. "Contratos electrónicos autoejecutables (Smart Contract) y pagos con tecnología Blockchain". 2017

[27] World economic Forum.]http://www3.weforum.org/docs/WEF_Building-Blockchains.pdf

[28]https://pdfs.semanticscholar.org/305e/dd92f237f8e0c583a809504dcec7e204d632.pdf

[29] Blockchain in Supply Chain Management: 13 Possible Use Cases, Mire Sam, 2018, Disponible en: https://www.disruptordaily.com/blockchain-use-cases-supply-chain-management/

[30] La tecnología blockchain puede revolucionar toda la cadena de suministros si la industria de la logística apuesta por ella. Business Insider, Mayo, 2019. Disponible en : https://www.businessinsider.es/informe-blockchain-industria-logistica-cadena-suministro-375519

[31] Blockchain Success cases: Supply Chain and Logistics, Zigurat. Disponible en : https://www.e-zigurat.com/innovation-school/blog/blockchain-success-cases/

[32] The rout to no-touch planning: Taking the human error out of supply chain planning. Felix, I., Kuntze, C., Tobias, E., Agosto, 2018. Disponible en : https://www.mckinsey.com/business-functions/operations/our-insights/the-route-to-no-touch-planning

[33] Why promoting provenance makes sense. Tyller, Rebecca July 2018. Disponible en: https://www.cips.org/en/supply-management/analysis/2018/july/promoting-provenance/

[34] Blockchain: Benefits for the supply chain, September, 2018. Disponible en:

https://www.openaccessgovernment.org/blockchain-supply-chain/51937/

[35] Características Raspberry Pi. Disponible en: https://www.raspberryshop.es/raspberry-pi-3.php

[36] Protocolo comunicación UDP. Disponible en: http://neo.lcc.uma.es/evirtual/cdd/tutorial/transporte/udp.html

[37] Instalar Geth en Raspberry. Disponible en: https://miethereum.com/wp-content/uploads/2018/01/GETH-Instalaci%C3%B3n-y-puesta-en-marcha-TRADUCCI%C3%93N-AL-ESPA%C3%91OL.pdf

[38] Tutorial: como instalar un nodo en la raspberry . Disponible en: https://medium.com/@jochemin/nodo-completo-bitcoin-en-raspberry-pi-3-3904c29d8ce1

[39] Tutorial: como instalar un nodo en la raspberry. Disponible en : https://medium.com/@thelucasmoore/part-one-building-an-ethereum-node-on-a-raspberry-pi-3b-481104974cf7

[40] Tutorial: como instalar un nodo en la raspberry https://medium.com/@thelucasmoore/part-two-building-an-ethereum-node-on-a-raspberry-pi-3b-a5154ae7a5b9

[41] Adaptar discos duros para escritura y lectura. Disponible en : https://devtidbits.com/2013/03/21/using-usb-external-hard-disk-flash-drives-with-to-your-raspberry-pi/

[42] Tutorial Ganache. Disponible en: https://transformaciondigital.izertis.com/blog/como-arrancar-y-testear-tu-propia-blockchain-con-ganache

[43] https://gwei.io/es/

[44] Adrian Pareja, “Parity vs Geth vs Quorum en Redes Permisionadas”. https://medium.com/everis-blockchain/parity-vs-geth-vs-quorum-en-redes-permisionadas-a4c5c9688437

ANEXO A

CÓDIGO LECTURA RFID EN EL ARDUINO:

```
/RST           D9
//SDA(SS)      D5
//MOSI         D11
//MISO         D12
//SCK          D13

#include <SPI.h>
#include <MFRC522.h>

const int RST_PIN = 9;            // Pin 9 para el reset del RC522
const int SS_PIN = 5;            // Pin 5 para el SS (SDA) del RC522 - El pin 10 se necesita para la comunicación Ethernet
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Crear instancia del MFRC522
unsigned char codigo;
unsigned int prueba;
byte ActualUID[4]; // Para almacenar el código leído
byte BeforeUID[4] = {0, 0, 0, 0};


void setup()
{
   Serial.begin(9600);      //Inicializa la velocidad de Serial
   SPI.begin();             //Función que inicializa SPI
   mfrc522.PCD_Init();      //Función que inicializa RFID
   Serial.println("Lectura código de barras");
}

void loop()
{
   // Detectar tarjeta
   if (mfrc522.PICC_IsNewCardPresent()){
        if (mfrc522.PICC_ReadCardSerial())
        {
           Serial.print(F("Card UID:"));
           //printArray(mfrc522.uid.uidByte, mfrc522.uid.size);
           for (byte i=0; i< mfrc522.uid.size; i++) {
                 Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
                 Serial.print(mfrc522.uid.uidByte[i], HEX);
                 ActualUID[i] = mfrc522.uid.uidByte[i];
           }
           Serial.println();
           if(compareArray(ActualUID, BeforeUID)){
            Serial.println("No registrar");
           }else{
            Serial.println("Registrar");
           }
           for (byte i=0; i< mfrc522.uid.size; i++) {
            BeforeUID [i] = ActualUID[i];
           }
           // Finalizar lectura actual
           mfrc522.PICC_HaltA();
        }
   }

}
```

```
62.
63. boolean compareArray(byte array1[],byte array2[]) {
64.   if(array1[0] != array2[0])return(false);
65.   if(array1[1] != array2[1])return(false);
66.   if(array1[2] != array2[2])return(false);
67.   if(array1[3] != array2[3])return(false);
68.   return(true);
69. }
```

CÓDIGO COMUNICACIÓN

En el Arduino:

```
1.  #include <Ethernet.h> //Load Ethernet Library
2.  #include <EthernetUdp.h> //Load UDP Library
3.  #include <SPI.h> //Load the SPI Library
4.  #include <MFRC522.h>
5.
6.  byte mac[] = { 0xA8, 0x61, 0x0A, 0xAE, 0x00, 0x76}; //Dirección MAC
7.  IPAddress ip(192, 168, 1, 163); //IP de la placa shield
8.  unsigned int localPort = 8888; // Puerto aleatorio, que coincida con la pi
9.  char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
10. String datReq;
11. int packetSize;
12. EthernetUDP Udp;
13.
14. unsigned char codigo;
15. unsigned int antena = 1;
16. const int RST_PIN = 9;                // Pin 9 para el reset del RC522
17. const int SS_PIN = 5;                 // Pin 5 para el SS (SDA) del RC522
18. MFRC522 mfrc522(SS_PIN, RST_PIN);    // Crear instancia del MFRC522
19. byte ActualUID[4]; // Para almacenar el código leído
20. byte BeforeUID[4] = {0, 0, 0, 0};
21. int codigo2=0;
22. int codigo1=0;
23.
24. void printArray(byte *buffer, byte bufferSize) {
25.    for (byte i = 0; i < bufferSize; i++) {
26.       Serial.print(buffer[i] < 0x10 ? " 0" : " ");
27.       Serial.print(buffer[i], HEX);
28.       ActualUID[i] = mfrc522.uid.uidByte[i];
29.       }
30. }
31.
32. void setup() {
33.   Serial.begin(9600); //Turn on Serial Port
34.   Ethernet.begin(mac, ip); //Initialize Ethernet
35.   Udp.begin(localPort); //Initialize Udp
36.   SPI.begin();
37.   mfrc522.PCD_Init();
38. }
39.
40. void loop() {
41.
42.   packetSize = Udp.parsePacket(); //Read the packetSize
43.
44.   if(packetSize>0){ //Check to see if a request is present
45.     Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE); //Reading the data request on
    the Udp
```

```
    String datReq(packetBuffer); //Convert packetBuffer array to string datReq

    if(datReq=="Codigo:"){
        Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());  //Initialize
        if (mfrc522.PICC_IsNewCardPresent()){
          if (mfrc522.PICC_ReadCardSerial())
              {
                for (byte i = 0; i < mfrc522.uid.size; i++) {
                    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
                    Serial.print(mfrc522.uid.uidByte[i], HEX);
                    ActualUID[i] = mfrc522.uid.uidByte[i];
                    codigo1= ActualUID[1];
                }
                Serial.println("");
                delay(1);
                // Finalizar lectura actual
                mfrc522.PICC_HaltA();
              }
              Udp.write(codigo1);
              Serial.println("Código leído");
              Udp.endPacket(); //Se termina la transmisión del paquete
        }
     }

  }
  memset(packetBuffer, 0, UDP_TX_PACKET_MAX_SIZE);
}
```

En la Raspberry Pi

```
import socket
import time
from socket import *

address= ('192.168.1.163', 8888)
client_socket = socket(AF_INET,SOCK_DGRAM)
client_socket.settimeout(1)

while(1):

    data = b"codigo"
    client_socket.sendto( data, address)
    try:
        rec_data, addr= client_socket.recvfrom(2048)
        aux = float(rec_data)
        #print (rec_data.decode('unicode_escape'))
        #aux = rec_data.decode('unicode_escape')
        if (aux!=0):
            print (aux)
    except:
        pass

    time.sleep(1)
    print("")
```

ENVÍO DATOS A BASE DE DATOS DE MYSQL

Una vez definidas las tablas, se comprueba que es posible la comunicación y enviar información a la tabla Lecturas, creada en una base de datos denominada Portfolio de la siguiente manera:

```
1. create DATABASE Portfolio;
2. use Portfolio;
3. Create table Lecturas( ID int NOT NULL PRIMARY KEY auto_increment, ANTENA int NOT
   NULL, CODIGO varchar(20) NOT NULL, FECHA date);
```

Para la comunicación es necesario instalar el paquete de mysql.connector

```
1. // Instalar el paquete mysql.connector
2. pip3 search mysql-connector | grep --color mysql-connector-python
3. mysql-connector-python (8.0.16)       - MySQL driver written in Python
4. mysql-connector python-rf (2.0.2)     - MySQL driver written in Python
5. mysql-connector-python-dd (2.0.2)     - MySQL driver written in Python
6. pip3 install mysql-connector-python-rf
```

Y el programa es el siguiente:

```
22. import mysql.connector
23. from mysql.connector import Error
24. from mysql.connector import errorcode
25. try:
26.    connection = mysql.connector.connect(host='localhost',
27.                                 database='Portfolio',
28.                                 user='root',
29.                                 password='ainhoa')
30.    sql_insert_query = """ INSERT INTO `Lecturas`
31.                           (`ID`, `ANTENA`, `CODIGO`, `FECHA`) VALUES (1,'1', '26 F
    A F8 08','2019-06-15')"""
32.    cursor = connection.cursor()
33.    result  = cursor.execute(sql_insert_query)
34.    connection.commit()
35.    print ("La información se ha insertado correctamente en la tabla: Lecturas"")
36. except mysql.connector.Error as error :
37.     connection.rollback() #rollback if any exception occured
38.     print("Fallo al insertar la información en la tabla: Lecturas {}".format(error
    ))
39. finally:
40.     #closing database connection.
41.     if(connection.is_connected()):
42.         cursor.close()
43.         connection.close()
44.         print("Conexión MySQL cerrada")
```

CÓDIGO FINAL: PROTOTIPO COMPLETO

En el Arduino el código utilizado es :

```
#include <Ethernet.h> //Load Ethernet Library
#include <EthernetUdp.h> //Load UDP Library
#include <SPI.h> //Load the SPI Library
#include <MFRC522.h>

#define PACKET_SIZE  384

byte mac[] = {0xA8, 0x61, 0x0A, 0xAE, 0x00, 0x76};
IPAddress ip(192, 168, 1, 163);
unsigned int localPort = 8888;
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
String datReq;
int packetSize=0;
EthernetUDP Udp;
int prueba;
String rfid_uid = "";

//Configuración RFID
const int RST_PIN = 9;                // Pin 9 para el reset del RC522
const int SS_PIN = 5;                 // Pin 5 para el SS (SDA) del RC522
MFRC522 mfrc522(SS_PIN, RST_PIN);     // Crear instancia del MFRC522
byte ActualUID[4]; // Para almacenar el código leído
byte BeforeUID[4] = {0, 0, 0, 0};
float codigo1=0;


void setup(){
  Serial.begin(9600);
  Ethernet.begin( mac, ip);
  mfrc522.PCD_Init();     //Función  que inicializa RFID
  Udp.begin(localPort);
  //delay(1500);
}

void loop(){

  packetSize = Udp.parsePacket();
  //Serial.println(packetSize);

  if(packetSize>0){

    Serial.println("ok1");

    Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
    String datReq(packetBuffer);

    if(datReq == "codigo"){

      Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());  //Initialize
      if (mfrc522.PICC_IsNewCardPresent()){
          if (mfrc522.PICC_ReadCardSerial()){
                for (byte i = 0; i < mfrc522.uid.size; i++) {
                    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
                    Serial.print(mfrc522.uid.uidByte[i], HEX);
                    String uid_part = String(mfrc522.uid.uidByte[i], HEX);
                    rfid_uid += uid_part;
                    ActualUID[i] = mfrc522.uid.uidByte[i];
                    codigo1= (ActualUID[1]);
                }
                Serial.println("");
                delay(1);
                // Finalizar lectura actual
                mfrc522.PICC_HaltA();
           }
           Udp.print(rfid_uid);
           Serial.println(rfid_uid);
```

```
            Udp.endPacket(); //Se termina la transmisión del paquete
                rfid_uid = "";
            }
          }
      }
      memset(packetBuffer, 0, UDP_TX_PACKET_MAX_SIZE);
    }
```

Y en la Raspberry Pi 3, el siguiente:

```
import mysql.connector
from mysql.connector import Error
from mysql.connector import errorcode

import datetime
from datetime import datetime

import web3
import json
from web3 import Web3
from web3.middleware import geth_poa_middleware

import socket
from socket import *
import time
from time import time


address= ('192.168.1.163', 8888)
client_socket = socket(AF_INET,SOCK_DGRAM)
client_socket.settimeout(1)
antena_leida = "1"
aux2=1.0
aux =0.0


web3 = Web3(Web3.IPCProvider("/media/pi/untitled/dev/geth.ipc"))
address1 = web3.toChecksumAddress('0x7f999f948fb8ea004d2b961e73a2ff7e4a49642b'
)
account1 = web3.eth.accounts[0]
web3.middleware_stack.inject(geth_poa_middleware, layer=0)

false = False
true = True

abiTFM= [
        {
        "constant": false,
        "inputs": [
        {
        "name": "_p",
        "type": "string"
        }
        ],
        "name": "testContract",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
        },
        {
        "constant": false,
        "inputs": [
        {
```

```
        "name": "_d",
        "type": "string"
        }
        ],
        "name": "testDate",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
        },
        {
        "constant": true,
        "inputs": [],
        "name": "date",
        "outputs": [
        {
        "name": "",
        "type": "string"
        }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
        },
        {
        "constant": true,
        "inputs": [],
        "name": "get",
        "outputs": [
        {
        "name": "",
        "type": "string"
        }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
        },
        {
        "constant": true,
        "inputs": [],
              "name": "getDate",
              "outputs": [
              {
              "name": "",
              "type": "string"
        }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
        },
        {
        "constant": true,
        "inputs": [],
        "name": "owner",
        "outputs": [
        {
        "name": "",
        "type": "address"
        }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
        },
```

```
        {
        "constant": true,
        "inputs": [],
        "name": "value",
        "outputs": [
        {
        "name": "",
        "type": "string"
        }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
        }
        ]
        contractClass = web3.eth.contract(abi=abiTFM)
        contractInstance = contractClass(address=address1)


        while(1):
            data = b"codigo"
            client_socket.sendto( data, address)
            try:
                    rec_data, addr = client_socket.recvfrom(2048)
                    aux=(rec_data)
                    if(aux!=0):
                        print(aux)
            except:
                pass

            codigo_leido= aux
            now= datetime.now()
            year = now.strftime("%Y")
            month = now.strftime("%m")
            day = now.strftime("%d")
            time = now.strftime("%H:%M:%S")
            date_time = now.strftime ("%d%m%Y, %H:%M:%S")
            date = now.strftime("%d%m%Y")

            if aux!=0:
                if(aux2!=aux):
                    try:
                                connection= mysql.connector.connect(host='localhost
    ', database='Portfolio', user='root', password='ainhoa')
                                antena=antena_leida
                                codigo=aux
                                aux2 = aux
                                fecha= date_time
                                sql_insert_query = """ INSERT INTO `Lecturas`(`ANT
    ENA`, `CODIGO`,`FECHA`) VALUES (%s , %s ,%s)""", (antena, codigo,fecha))
                                cursor= connection.cursor()
                                result = cursor.execute(*sql_insert_query)
                                connection.commit()
                                print("OK")
                                contractInstance.functions.testContract(aux).transa
    ct({'from': account1})
                                contractInstance.functions.testDate(fecha).transact
    ({'from': account1})
                                fecha =0
                    finally:
                        #closing database connection.
                        if(connection.is_connected()):
                            cursor.close()
                            connection.close()
                            print("MySQL connection is closed")
```