UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

DISEÑO Y DESARROLLO DE PROPIEDADES INTELIGENTES: APLICACIÓN DE LA BLOCKCHAIN A INTERNET DE LAS COSAS

Guillermo Ruiz Álvarez
Tutor: Óscar Delgado Mohatar
Ponente: Eloy Anguiano Rey

Junio 2016

UNIVERSIDAD AUTÓNOMA DE MADRID
Escuela Politécnica Superior
Francisco Tomás y Valiente, nº 11
Madrid, 28049
España

Diseño y desarrollo de propiedades inteligentes: aplicación de la blockchain a Internet de las Cosas
Guillermo Ruiz Álvarez, Junio 2016

IMPRIMIDO EN ESPAÑA – PRINTED IN SPAIN

Diseño y desarrollo de propiedades inteligentes: aplicación de la blockchain a Internet de las Cosas

AUTOR: Guillermo Ruiz Álvarez
TUTOR: Óscar Delgado Mohatar
PONENTE: Eloy Anguiano Rey

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Junio 2016

Agradecimientos

En primer lugar, me gustaría dar las gracias a Óscar Delgado, tutor de este trabajo, no sólo por haberme dado la oportunidad de realizarlo, sino por abrirme las puertas a un mundo tan fascinante como es el de la seguridad, por todo lo que me ha enseñado y lo que he aprendido gracias a él y por mostrarme las cosas tal y como son. Por otro lado, no podría no mencionar a Eloy Anguiano, ponente de este trabajo, por haber trabajado en un proyecto de LaTeX sin el cual este no sería como es y abrirlo a todo el que lo necesite.

También tengo que dar las gracias a mis amigos, por haber estado ahí siempre. Y por último, y no por ello menos importante, a mi familia. En especial a mi madre y a mi hermana, porque sin ellas muy pocas cosas tendrían sentido, y a mi abuelo, por haber sido mi modelo a seguir desde el principio de mi vida hasta el final de la suya, y aún continuar siéndolo.

Resumen

El *Internet de las Cosas (Internet of Things, IoT)*, la red de interconexión de los objetos cotidianos mediante el uso de internet, es un concepto cuya realidad avanza hacia el campo de la innovación disruptiva debido al continuo avance tecnológico. En muchos casos, la conectividad entre personas y objetos está permitiendo que se tomen decisiones en función de los datos recogidos y analizados por los mismos sin que sea necesaria la intervención humana. El resultado de esta evolución de las tecnologías es la apertura de un nuevo campo de numerosas aplicaciones que están basadas en el uso de estos dispositivos.

En este contexto, la seguridad toma un papel de vital importancia. Los dispositivos conectados a la red tienen un estrecho vínculo con sus propietarios y cualquier vulnerabilidad existente puede traer importantes problemas de privacidad debido a la gran cantidad de información que estos manejan y a las decisiones que puedan tomar en base a ella. Es por ello que se hace necesario disponer de un sistema preciso que permita al propietario de un dispositivo proporcionar una prueba para identificarse como tal.

Hasta ahora, los sistemas de este tipo han requerido de terceras partes en las que hay que confiar la responsabilidad de mantener un registro de propiedad. Sin embargo, este problema fue resuelto a finales del año 2008, con una publicación bajo el seudónimo de *Satoshi Nakamoto* que trajo consigo varios conceptos que incluso a día de hoy siguen suscitando gran controversia. Uno de ellos fue el *bitcoin*, la primera *criptodivisa* que prescinde de autoridad central y que registra el historial de transacciones en un sistema público y distribuido conocido como *blockchain* (cadena de bloques).

Esta tecnología tiene un notable potencial en su relación con el *Internet de las Cosas*, y su evolución ha hecho surgir el concepto de *smart property* (*propiedad inteligente*), un dispositivo cuya propiedad es gestionada mediante el uso de la cadena de bloques.

En este trabajo se utiliza la cadena de bloques para construir un sistema que permita mantener un registro distribuido de propiedad, realizando un estudio previo de las tecnologías utilizadas.

Asimismo, se reutiliza no sólo el concepto de cadena de bloques, sino la red completa que usa *Bitcoin* para implementar el sistema descrito anteriormente. Tal sistema ha permitido diseñar y construir un prototipo de *propiedad inteligente*, con el uso del microcontrolador *Arduino Yún*, que utiliza dicho sistema para conocer en todo momento a quién pertenece y aceptar únicamente las peticiones realizadas por su propietario.

Palabras clave: Internet de las Cosas, bitcoin, cadena de bloques, propiedad inteligente, ciberseguridad, criptografía.

Abstract

Internet of Things (IoT), the global network of everyday objects, is a concept that, as the technology continues to advance, has the potential to create digital disruption. In many cases, connectivity between people and objects allows devices to make decisions based on the data that is collected and analyzed without human intervention. As a result, a new field of applications is being developed, in which the term *smart* is frequently used to refer to these devices.

In this context, security takes on a very important role. Devices connected to the network are strongly related to their owners, and any existing vulnerability can lead to significant privacy issues, due to the large amount of information they handle and the decisions they could make based on it. This is the reason why it is necessary to build an accurate system that allows the current owner of such devices to provide a proof that he is, indeed, the owner.

Up to now, such systems had to rely on trusted third-parties to be responsible for keeping track of ownership information. However, by the end of 2008, a publication under the pseudonym *Satoshi Nakamoto* provided a solution. Several concepts that even today continue to attract much controversy were introduced. One was the *bitcoin*, the first *cryptocurrency* that dispenses with a central authority and whose transaction history is recorded in a public distributed ledger, known as *blockchain*.

This technology has significant potential in its relationship with the *Internet of Things*, where the concept of *smart property* arises. A *smart property* is a device whose ownership is managed by using the *blockchain*. In other words, it is said that *smart properties* are to ownership what *bitcoin* is to money.

To get a correct design of a system that keeps distributed records of property, a preliminary study of the technologies to be used will take place.

In this way, this project reuses not only the concept of *blockchain*, but the whole network that *Bitcoin* uses to implement a system as the one described. Such system has allowed us to design and develop a complete prototype of *smart property* with the *Arduino Yún* microcontroller board, which uses this system to always know who its owner is, and only respond to requests made by him.

Key words: Internet of Things, bitcoin, blockchain, smart property, cybersecurity, cryptography.

ÍNDICE

Listas

Lista de figuras

Lista de tablas

GLOSARIO

ataque de man-in-the-middle
: Un *ataque de man-in-the-middle* corresponde a una forma de ataque en la que un usuario malintencionado intercepta una conexión y adquiere la capacidad de leer, modificar e insertar mensajes de forma fraudulenta.

ataque de repetición
: Un *ataque de repetición* es una forma de ataque de red en la que un usuario malintencionado intercepta una transmisión de datos válida y la reinyecta de forma fraudulenta.

Bitcoin
: Término utilizado para referirse tanto a la red completa como al protocolo que utiliza la criptodivisa bitcoin.

bitcoin
: Unidad monetaria digital que utiliza la red Bitcoin. El símbolo utilizado para referirse a esta moneda digital es ₿.

criptografía asimétrica
: La criptografía asimétrica o criptografía de clave pública comprende una serie de métodos para el intercambio de mensajes. Cada usuario posee un par de claves, una pública que el resto de usuarios puede utilizar para cifrar los mensajes dirigidos al propietario y una privada, utilizada tanto para descifrar esos mensajes como para proporcionar autenticación de origen de los enviados.

criptomoneda
: Una *criptomoneda* es una unidad monetaria de intercambio digital que utiliza métodos criptográficos para asegurar las transacciones y la creación de nuevas unidades.

framework
: Un *framework* es un entorno que ofrece una funcionalidad definida y da soporte como base para la organización y el desarrollo de software.

fuerza bruta
: El término *fuerza bruta* se refiere a una forma de encontrar un valor por medio de probar todas las combinaciones posibles del mismo hasta dar con el buscado.

hash
: Función que utiliza un algoritmo para transformar un conjunto de elementos de entrada, normalmente cadenas, a un conjunto de salida finito. Estas funciones se suelen utilizar para obtener una representación compacta de dicha entrada. El término hash se utiliza aquí para referirse tanto a las funciones como a las salidas de las mismas.

metadatos
: Los metadatos son datos estructurados que proporcionan información sobre otros datos.

red peer-to-peer

Arquitectura de red que funciona prescindiendo de servidores centrales, en la que los nodos se conectan entre sí de forma directa.

satoshi

Un *satoshi* es el equivalente a 10^{-8}₿.

script

Un *script* es un programa que automatiza la ejecución de un conjunto de tareas. Los lenguajes que soportan este tipo de programas se denominan lenguajes de *scripting*.

torrent

Un *torrent* es un archivo que almacena metadatos sobre ficheros que están distribuidos por la red. Dependiendo del contexto, el mismo término puede utilizarse para referirse al contenido de dichos ficheros distribuidos.

Acrónimos

API Application Programming Interface.

CA Certification Authority.

ECDSA Elliptic Curve Digital Signature Algorithm.

IoT Internet of Things.

P2P peer-to-peer.

RPC Remote Procedure Call.

SSH Secure SHell.

TLS Transport Layer Security.

UTXOs Unspent Transaction Outputs.

1 Introducción

En los últimos años, el desarrollo de la computación de bajo coste y la disponibilidad de nuevos dispositivos electrónicos han logrado que internet comience a expandirse hacia una nueva dimensión, la de los objetos inteligentes. El continuo avance de las tecnologías está permitiendo a estos dispositivos adquirir cierta autonomía e inteligencia, y su interconexión por medio de internet resulta en una relación directa entre el mundo físico y el digital. La cantidad de objetos cotidianos que están interconectados a través de la red está creciendo de forma considerable, y es en este contexto donde se introduce una nueva visión que está revolucionando el entendimiento de internet tal y como lo conocemos ahora.

El *Internet de las Cosas* (*Internet of Things*, *IoT*) es un concepto referido a la interconexión de objetos cotidianos por medio de internet. Su auge se fundamenta no sólo en la capacidad que adquieren los dispositivos inteligentes con el avance de la tecnología, sino en el continuo crecimiento de la cantidad de objetos físicos que comienzan a utilizar las redes para comunicarse entre sí [14].

No obstante, esta revolución puede verse mermada por importantes problemas que se están afrontando en la actualidad. Los dispositivos electrónicos que recogen información de su entorno por medio de sensores están en continuo desarrollo, y a medida que su uso se hace más común, crece la amenaza contra la seguridad de los mismos. La información que estos dispositivos manejan y analizan para la toma de decisiones está directamente relacionada con sus propietarios, y un acceso no autorizado de terceras partes puede comprometer de forma importante su privacidad y su control sobre estos objetos.

La ausencia de nuevas soluciones para asegurar aspectos tales como la privacidad de los usuarios, la información sensible o incluso la creación de nuevos modelos de negocio podrían frenar el avance del *IoT*. Muchas de las soluciones propuestas implican establecer y mantener confianza con autoridades centrales que tienen acceso al control de los dispositivos. Sin embargo, en una red en continua expansión de la escala del *IoT*, la privacidad y el anonimato deben integrarse en su diseño otorgando a los usuarios el control de sus propios dispositivos. Los modelos de seguridad actuales basados en código cerrado se están quedando obsoletos y deben ser reemplazados por un nuevo enfoque, el de la seguridad a través de la transparencia [12].

1.1. Motivación

Para cualquier tipo de propiedad de valor es importante mantener registros precisos que permitan a los propietarios identificarse como tal, de manera que puedan ser utilizados para proteger sus derechos, resolver disputas, asegurar que las transferencias de propiedad son realizadas y prevenir fraudes. Hasta ahora, estos sistemas han necesitado de terceras partes con las que hay que establecer confianza para el mantenimiento de estos registros, tales como fabricantes o agencias gubernamentales [12] [16].

En 2008, una publicación bajo el seudónimo de *Satoshi Nakamoto* [1] planteó una solución a este problema. En ella se describía el *bitcoin*, la primera *criptomoneda* que prescinde de autoridad central para realizar transferencias entre usuarios, con la que se introdujo una nueva tecnología conocida como *blockchain* (cadena de bloques).

La cadena de bloques proporciona un registro público y distribuido de las transacciones de esta moneda digital, asegurándolas mediante métodos basados en criptografía asimétrica y haciendo que sea computacionalmente impracticable alterar los datos que ya han sido registrados. Su potencial se ha visto reflejado en la creación de nuevos modelos económicos por medio de las monedas digitales, y un continuo análisis por parte de diversos grupos está permitiendo que se extiendan sus aplicaciones más allá de este campo. Como resultado de la relación que existe entre el concepto de cadena de bloques y el de *IoT*, surge el término de *propiedad inteligente* [13], referido a aquellos activos cuya gestión de la propiedad se realiza por medio del uso de esta tecnología.

La cadena de bloques ofrece la posibilidad de prescindir de una autoridad central para gestionar estos registros, manteniendo los datos de forma pública y distribuida y permitiendo utilizar monedas digitales que sirvan como representación de los activos, cuya propiedad puede ser verificada con métodos criptográficos. El concepto de *propiedad inteligente*, junto a la evolución que ha sufrido la cadena de bloques, ha hecho que se desarrollen protocolos que permiten esta gestión de activos por medio de la representación de las propiedades a través de las monedas digitales. Uno de estos protocolos, llamado *Open Assets*, permite tanto la emisión como la transferencia de activos digitales entre usuarios de la red *Bitcoin*.

En este marco, surge la idea de que sean los propios dispositivos los que determinen a quién pertenecen con el uso de la información contenida en la cadena de bloques, puesto que esta constituye un registro público y distribuido cuya estructura no permite modificar los datos registrados, manteniendo su integridad.

Idealmente, estos objetos deberían ser completamente autónomos para poder verificar la información contenida en dicha cadena de forma independiente. Sin embargo, los dispositivos de bajo coste carecen de la capacidad de procesamiento necesaria para realizar esta tarea, puesto que la cadena de bloques mantiene el histórico de todas las transacciones realizadas. Por tanto, se hace necesario implementar un sistema de registro de propiedades que permita a estos dispositivos consultar la información sin tener que realizar un análisis completo de dicha cadena, para poder conocer en todo momento el propietario al que corresponden y permitir al mismo identificarse como tal.

1.2. Objetivos

En este contexto, uno de los objetivos de este proyecto es implementar un sistema que mantenga un registro de propiedades público y distribuido, haciendo uso de la cadena de bloques de *Bitcoin* para mantener el registro de los datos y del protocolo *Open Assets* para representar, mediante monedas digitales, la emisión y transferencia de activos.

De este modo, el sistema de registro de propiedades deberá:

- Mantener una copia propia de la cadena de bloques de *Bitcoin*, que será verificada y se utilizará para extraer los datos relativos a las propiedades.
- Detectar y procesar las transacciones que contengan monedas digitales que utilicen el protocolo *Open Assets* para extraer la información relativa a las propiedades que representan.
- Ofrecer a los dispositivos un sistema de acceso a la información extraída, de modo que puedan consultar periódicamente el identificador de su propietario. De esta manera, si se realiza la transferencia de una propiedad entre usuarios, el propio dispositivo tendrá independencia para advertir que su dueño ha cambiado y poder reconocer a su nuevo propietario.
- Ofrecer a los propietarios un sistema de acceso a los datos, de modo que puedan consultar la información relativa a la definición de una propiedad.
- Mantener la información actualizada, de manera que las nuevas emisiones y transferencias de propiedades que sean realizadas sean identificadas por el sistema tras la validación de las transacciones recibidas.

Asimismo, en el objetivo principal de este proyecto se incluye diseñar y desarrollar un prototipo de *propiedad inteligente* con el uso del microcontrolador *Arduino Yún*, cuya propiedad pueda ser gestionada y transferida utilizando la cadena de bloques de *Bitcoin*. De esta manera, el sistema permitirá que el dispositivo conozca en todo momento a quién pertenece sin tener que realizar un análisis completo de la cadena de bloques, pudiendo autenticar las peticiones recibidas por parte de su propietario.

1.3. Estructura del documento

En el capítulo 2 se realiza un estudio de la red *Bitcoin*, necesario para introducir el concepto de cadena de bloques y comprender su funcionamiento. Asimismo, se describe cómo ha evolucionado la cadena de bloques desde que se introdujo con la llegada de *Bitcoin* hasta su estado actual, permitiendo el desarrollo de un conjunto de protocolos destinado a la gestión de activos. Finalmente, se profundiza en el que se utiliza para el desarrollo de este proyecto, el protocolo *Open Assets*.

En el capitulo 3 se ofrece una descripción general del proyecto y se proporcionan los detalles de diseño e implementación tanto del sistema propuesto como del prototipo de *propiedad inteligente*.

El capítulo 4 está dedicado a la descripción de las pruebas realizadas y de los resultados obtenidos en base al funcionamiento del prototipo construido en conexión con el sistema implementado.

Para terminar con el cuerpo del documento, en el capítulo 5 se presentan tanto las conclusiones obtenidas con la realización de este proyecto como una descripción del trabajo futuro.

2

Estudio de las tecnologías a utilizar

Este capítulo comienza con un estudio previo del funcionamiento de la red *Bitcoin*, que se hace necesario tanto para introducir el concepto de cadena de bloques como para entender cómo se evita la alteración de los datos y cómo se aseguran las transacciones realizadas entre usuarios. En segundo lugar, se realiza un análisis de la evolución que ha experimentado el concepto de cadena de bloques, pasando de sostener un sistema de monedas digitales a extenderse a otros sectores para producir una gran cantidad de aplicaciones. Posteriormente, se describen varios protocolos que surgen gracias a esta evolución y que permiten realizar la emisión y transferencia de activos con el uso de esta cadena, profundizando en el análisis del funcionamiento del que será utilizado en este proyecto, el protocolo *Open Assets*.

2.1. Bitcoin

A finales del año 2008, una publicación bajo el seudónimo de *Satoshi Nakamoto* trajo consigo la primera *criptomoneda* que **prescinde de autoridad central** para la realización de transacciones, el *bitcoin* [1]. El historial de todas las transacciones realizadas se registra en un **sistema público y distribuido** que preserva el **anonimato** de los usuarios, y que utiliza métodos criptográficos para proporcionar **seguridad** y controlar la creación de nuevas unidades de esta moneda digital.

Bitcoin es una *criptomoneda* que utiliza una *red peer-to-peer* (*P2P*) para permitir el envío de dinero electrónico entre pares prescindiendo de terceras partes que actúen como intermediarios. Para ello, utiliza el sistema conocido como cadena de bloques, un registro público y distribuido de transacciones que permite **evitar la alteración de los datos** ya registrados.

La cadena de bloques se entiende como una de las grandes innovaciones que *Bitcoin* trajo consigo. La clave de esta innovación es que los usuarios confían en el conjunto de nodos que componen y mantienen la red, en lugar de tener que establecer y mantener confianza con la contraparte. La **confianza en la red** se fundamenta en el **consenso** de los nodos que la conforman. Cada uno de ellos tiene, de forma independiente, una copia de la cadena de bloques, donde se almacena el histórico completo de las transacciones realizadas. Cuando un usuario quiere realizar una transacción difunde la misma al resto de nodos de la red, de modo que si esta es válida será incluida en un bloque la cadena. Mediante normas de validación comunes, los nodos llegan a tener copias idénticas de la cadena, alcanzando el estado conocido como **consenso**.

De este modo, los nodos no necesitan confiar en ningún tipo de agente central que actúe como interventor para la realización de las transacciones, puesto que estas son aseguradas mediante métodos basados en criptografía asimétrica.

En los siguientes apartados se describirá la estructura de la cadena de bloques y mecanismos utilizados para evitar la alteración de los datos registrados y proporcionar seguridad.

2.1.1. La cadena de bloques

La cadena de bloques es una estructura [4] que proporciona un registro de transacciones ordenado cronológicamente. Los bloques que conforman dicha estructura almacenan un conjunto de transacciones representado mediante el uso de un árbol de tipo *Merkle*.

La razón principal del uso de este tipo de árbol es que un pequeño cambio en cualquier transacción altera completamente la raíz del mismo, permitiendo identificar que ha habido una modificación de los datos. La construcción de los árboles de tipo *Merkle* de detalla a continuación:

- En primer lugar se colocan las **transacciones como nodos hoja** del árbol *Merkle*.
- A continuación, se toman pares de los nodos hoja del árbol, se concatenan y al resultado se le aplica una función *hash*. De este modo, **a partir de un par** de nodos hoja **se obtiene un nuevo nodo**, que será el padre de dicho par. En caso de que el número de nodos sea impar, la última hoja será concatenada consigo misma. La función utilizada por *Bitcoin* es *double-SHA-256*, que consiste en aplicar dos veces la función *SHA-256*.
- Iterando el proceso se forman nuevos niveles superiores del árbol hasta llegar a un único nodo, llamado *raíz Merkle*. **La *raíz Merkle* se almacena en la cabecera del bloque** que contiene al conjunto de transacciones.

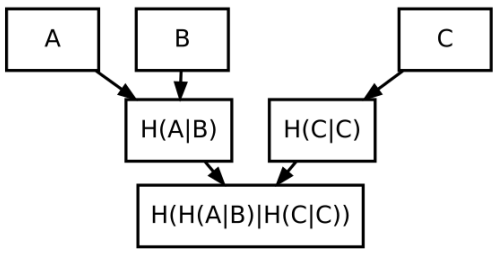


Figura 2.1: Ejemplo de construcción de un árbol Merkle.

En la figura 2.1 se puede observar un ejemplo de construcción de un árbol de *Merkle*. Las transacciones se concatenan por pares para aplicarles una función *hash* y construir un nivel más del árbol. En este caso, dado que el número de transacciones es impar, la última transacción se concatena consigo misma. Finalmente, tras aplicar el mismo proceso al nuevo nivel, se obtiene la *raíz Merkle*. En la imagen se utiliza el símbolo "|" para representar el operador concatenación y el término H para referirse a la función *hash*.

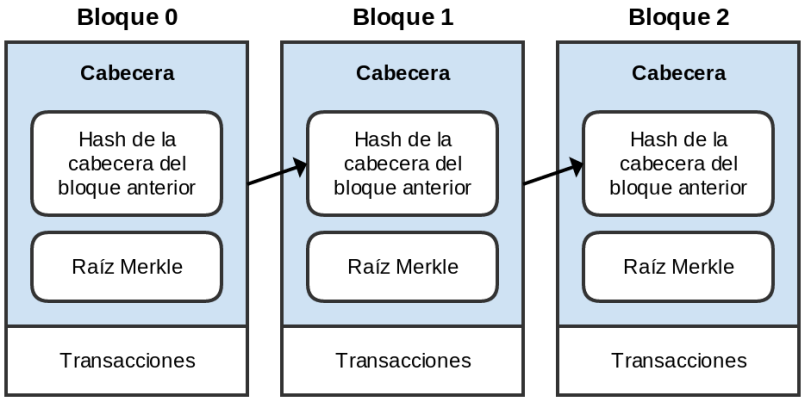


Figura 2.2: Versión simplificada de una cadena de bloques.

Cada bloque, a parte de la *raíz Merkle*, guarda en la cabecera el *hash* de la cabecera del bloque anterior, encadenándose así con el mismo. Es por ello por lo que se denomina a esta estructura cadena de bloques. De esta manera se consigue evitar que se modifiquen los datos de un bloque sin tener que modificar todas las cabeceras de los bloques siguientes, puesto que **cualquier alteración en una transacción altera el resultado de la *raíz Merkle*** y por tanto la cabecera del bloque que la contiene. En la figura 2.2 se muestra un ejemplo simplificado de una cadena de bloques.

Se observa, por tanto, que la cadena de bloques está estructurada de manera que la alteración de los datos relativos a un bloque hace necesaria la reconstrucción de la cabecera de todos los bloques posteriores al mismo. Para evitar que un usuario malintencionado pueda llevar a cabo tal tarea, la creación de un nuevo bloque de la cadena requiere proporcionar lo que se conoce como **prueba de trabajo**.

Cuando un usuario quiere realizar una nueva transacción realiza una **difusión** de la misma a todos los nodos de la red *P2P*, de forma si esta es válida será incluida en un nuevo bloque de la cadena. La tarea de crear nuevos bloques y de mantener la red es conferida a cierto conjunto de nodos de la misma. Los usuarios que pertenecen a este conjunto, conocidos como **mineros**, se encargan de confirmar las transacciones e incluirlas en un nuevo bloque. Los bloques creados por los mineros podrán ser añadidos a la cadena si estos son capaces de proporcionar una **prueba de trabajo**.

Entre los campos que la cabecera de un bloque posee se encuentra uno denominado *nonce*, se trata de 32 bits que el minero puede elegir al azar modificándose así el *hash* de la cabecera del bloque con cada valor elegido. **La prueba de trabajo consiste en encontrar un valor para este campo** tal que el *hash* de la cabecera esté por debajo de un umbral dado, conocido como *target* (objetivo). De este modo, **mediante la fuerza bruta**, los mineros tratan de encontrar un valor para el *nonce* que les permita crear un bloque nuevo. Una vez conseguido realizan una difusión del bloque, y cuando el resto de nodos lo recibe, lo analiza y declara su validez intentando encontrar uno nuevo que se sitúe tras el mismo, sirviendo este nuevo bloque como **confirmación** del anterior.

Puede ocurrir que varios mineros generen un bloque a la vez o que un bloque sea rechazado por el resto de la red por tener datos inválidos. En este caso, el resto de nodos debe

decidir cuál será el que se elegirá para seguir generando bloques nuevos tras él, hecho que posibilita que se produzcan bifurcaciones en la cadena. Los mineros elegirán la cadena más larga, pues es la que mayor prueba de trabajo representa, y **una mayor prueba de trabajo representa el consenso de la mayoría de la red**. En esta situación, los bloques de la otra cadena serán consideraros inválidos por todos los nodos de la red. En la figura 2.3 se puede ver un ejemplo del caso recién descrito: el bloque 3b queda invalidado cuando el resto de nodos decide continuar la cadena a partir del bloque 3.

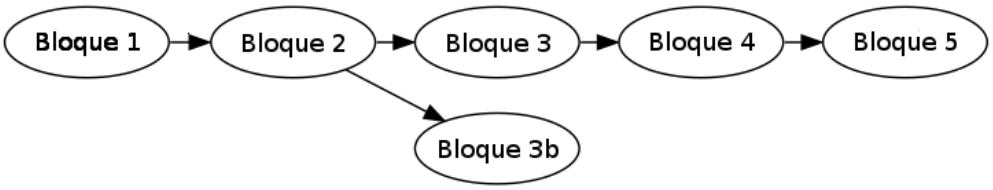


Figura 2.3: Ejemplo de una bifurcación en la cadena de bloques.

De esta manera, **para realizar una alteración** de los datos de un bloque **se necesitaría controlar la mayoría de la fuerza de cómputo** de la red, pudiendo realizar lo que se conoce como *ataque del 51 %*. Esto hace que la confianza en la misma se fundamente en que la mayoría de nodos son honestos y no se ponen de acuerdo para corromper la red.

A cambio del cómputo necesario, los bloques nuevos incluyen una transacción especial denominada *coinbase* que genera una cantidad estipulada de *bitcoins* para el minero. El propósito de la red es que, entre todos ellos, sean capaces de generar un nuevo bloque aproximadamente cada 10 minutos. Dado que crear nuevos bloques genera beneficios económicos, los mineros invierten cada vez más en capacidad de cómputo y es por ello por lo que el *target* cambia periódicamente (cada cierto número de bloques generados) con el objetivo de mantener la tasa media de generación en 10 minutos.

De esta forma, la cadena de bloques mantiene una tasa estable de incorporación de nuevos datos y hace impracticable la alteración de los correspondientes a un bloque a medida que se generen confirmaciones del mismo.

2.1.2. Transacciones

Las transacciones son estructuras [4] que permiten a los usuarios de la red transferirse *bitcoins* entre sí. Para ello, cada una de estas estructuras dispone de una serie de entradas y salidas que se utilizan para realizar la operación de transferencia. De este modo, los *bitcoins* realmente se mueven entre transacciones: cada una de ellas gasta los *bitcoins* recibidos previamente por una o varias transacciones, de manera que cada entrada de una transacción se enlaza con una salida de otra anterior mediante una referencia. En la figura 2.4 se muestra un ejemplo de cómo se produce este enlace.

Las transacciones pueden gastar en sus salidas tantos *bitcoins* como acumulen con sus entradas para ser consideradas válidas. Cuando la suma de las salidas es menor que la suma de las entradas, la diferencia se acumula en la *coinbase* del bloque que la almacena, que

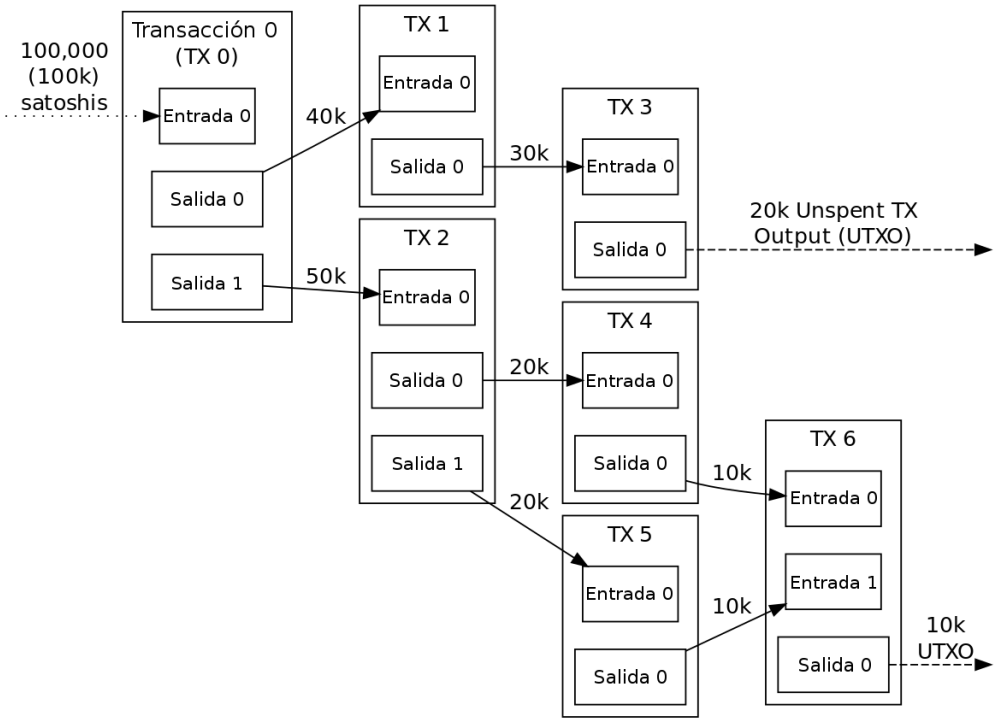


Figura 2.4: Ejemplo de transferencia de bitcoins entre transacciones.

podrá ser gastada por el minero correspondiente. De esta manera, los usuarios tienen un método para poder pagar una tarifa a los mineros para que procesen sus transacciones con mayor prioridad que otras. En el ejemplo de la figura, todas las transacciones pagan una tarifa de 10.000 *satoshis*. Las salidas de transacciones que aún no se han utilizado como entradas de otras, es decir, que aún no han sido gastadas, están en un conjunto denominado *Unspent Transaction Outputs (UTXOs)*. Cuando una transacción utiliza como entrada una salida que no está en este conjunto, se está produciendo un intento de **doble gasto**, es decir, de utilizar salidas ya gastadas, y los nodos considerarán la transacción inválida.

Las salidas de las transacciones tienen, entre sus campos, la cantidad que pueden gastar y un *script*, llamado *pubkey script*, que indica mediante el uso de códigos de operación quién puede usar la salida como entrada de otra transacción. De este modo, **cualquier usuario que pueda satisfacer las condiciones** que impone dicho *script* para retornar un valor verdadero **podrá gastar los correspondientes *bitcoins***.

A continuación se describe un ejemplo de cómo son estas condiciones. Un usuario, llamado *Alice*, quiere realizar una transferencia a otro usuario, llamado *Bob*. En primer lugar, *Bob* generará un par de claves pública y privada. Para ello, *Bitcoin* utiliza *ECDSA* (*Elliptic Curve Digital Signature Algorithm*), un algoritmo de firma digital mediante el uso de curvas elípticas, y la curva *secp256k1* [6]. *Bob* genera un número aleatorio de 256 bits para crear una clave privada y utiliza la curva *secp256k1* para obtener la clave pública asociada. A dicha clave pública se le aplica un *hash* y se transfiere a *Alice* para que pueda añadirla al *script* de la transacción (ver figura 2.5). Normalmente, las claves públicas se transfieren en forma de lo que se conoce como *dirección Bitcoin* [3], que no son más que codificaciones mediante la aplicación de funciones *hash* que incluyen sumas de verificación.

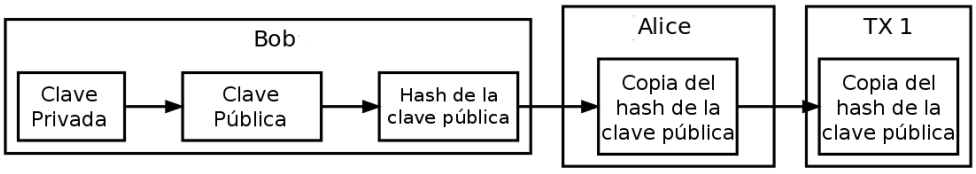


Figura 2.5: Obtención de un hash de clave pública para recibir un pago.

Una vez *Alice* realice una difusión de la transacción y esta sea validada, *Bob* podrá utilizar la salida que le corresponde en cualquier momento, ya que será incluida en el conjunto de *UTXOs*. Cuando *Bob* quiera utilizar sus *bitcoins*, realizará otra transacción con una entrada que tenga una referencia a dicha salida. Para ello, creará un *script* conocido como *signature script* para **autentificarse** como el usuario que tiene permiso para realizar el gasto.

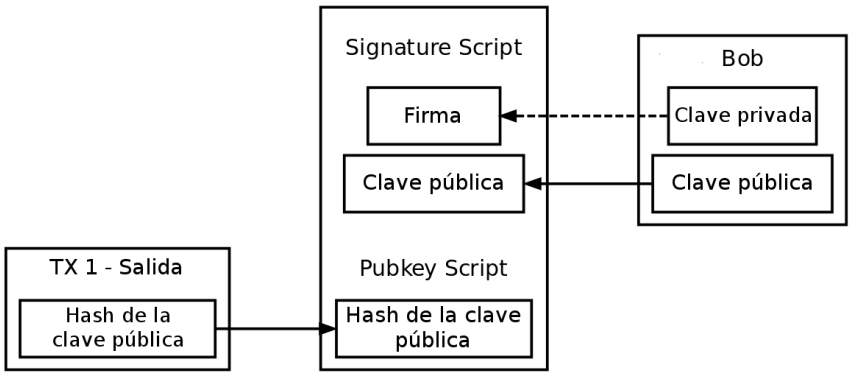


Figura 2.6: Esquema del proceso de gasto de la salida de una transacción.

Por un lado, *Bob* debe proporcionar su clave púbica completa de forma que el *script* de la salida de la transacción pueda comprobar que el *hash* de la misma coincide con el que tiene almacenado. Asimismo, *Bob* creará una firma de parte de la información de la transacción para demostrar que él es el dueño de la clave privada y por tanto satisface las condiciones necesarias para realizar el gasto. En la figura 2.6 se muestra un esquema del proceso recién descrito.

Se tiene, por tanto, que la red **asegura** criptográficamente las transacciones preservando el anonimato de sus usuarios, puesto que las claves públicas no revelan nada acerca de la identidad de los mismos.

2.2. Blockchain 2.0

Como se ha visto, la llegada de *Bitcoin* introdujo un nuevo concepto que sirvió de base para construir un sistema monetario digital que ha traído consigo nuevos modelos económicos. Debido al gran potencial que posee, la cadena de bloques ha sido analizada con el objetivo de extender los beneficios que esta tecnología puede proporcionar [11].

A raíz de dicho análisis, y dado que el diseño de la estructura desarrollada para *Bitcoin* es extremadamente extensible, surge lo que se conoce como *Blockchain 2.0*, una segunda versión del concepto de cadena de bloques. *Blockchain 2.0* es la evolución de la cadena de bloques diseñada para *Bitcoin*, y dado que aún sigue en desarrollo, existen diferentes nociones sobre este concepto. Sin embargo, hay un acuerdo general en que incluye los conceptos de *propiedades* y *contratos inteligentes* [11] [19].

Las *propiedades* y *contratos inteligentes* fueron propuestos por primera vez en 1997 por Nick Szabo [2]. Tal y como se ha descrito, el término *propiedad inteligente* se refiere a ciertos dispositivos cuya propiedad puede gestionarse y transferirse por medio de la cadena de bloques. Por otro lado, el concepto de *contrato inteligente* se refiere a un tipo de transacción que va mas allá de la transferencia de activos, comprendiendo un método de acuerdo entre usuarios de la cadena.

Como consecuencia de esta evolución, *Bitcoin* lanzó una nueva versión del protocolo que ofrece en su lenguaje de *scripting* un código de operación especial, denominado **OP_RETURN** [17], con el que se permite introducir hasta un máximo de 80 bytes de metadatos en una transacción. A raíz de este gran cambio se han creado protocolos que permiten utilizar la cadena de bloques para conseguir definir, emitir y transferir *propiedades inteligentes*, surgiendo así el concepto de *monedas coloreadas*.

2.3. Protocolos de emisión y transferencia de activos

El término *moneda coloreada* está asociado con el uso de protocolos construidos sobre la cadena de bloques de *Bitcoin* para representar activos digitales.

La llegada del código de operación **OP_RETURN** ha supuesto una revolución en el concepto de cadena de bloques como una estructura que puede ser utilizada en aplicaciones que van más allá de la transferencia de monedas digitales. Este código de operación implica que las transacciones puedan tener una salida adicional que no esté en el conjunto de *UTXOs* y que almacene hasta 80 bytes de metadatos.

A continuación se detallan los protocolos que han surgido a raíz del concepto de *moneda coloreada*.

- El protocolo ***EPOBC*** fue la primera implementación de las *monedas coloreadas*. Su desventaja principal es que no utiliza el código **OP_RETURN** para introducir datos en la cadena, sino que se aprovecha de otros campos de las transacciones que están presentes pero no son utilizados.
- El protocolo ***Open Assets*** es uno de los más sencillos dentro de esta familia. Este protocolo sí que utiliza el código de operación **OP_RETURN** para almacenar toda la información necesaria relativa tanto al protocolo en sí mismo como a los activos que define y maneja. Los metadatos relativos a los activos emitidos son referenciados desde la cadena de bloques y almacenados en la web.
- El protocolo ***CoinSpark*** utiliza el mismo código de operación que *Open Assets* para almacenar metadatos. Sin embargo, su complejidad es mayor que la del protocolo an-

terior, a cambio de ofrecer funcionalidades adicionales como el envío de mensajes o la certificación de correos electrónicos.

- La implementación de monedas coloreadas de ***Colu*** es la más reciente hasta el momento. Su diferencia con las demás radica en que utiliza *torrents* para guardar la información relativa a los activos, manteniéndola así descentralizada. La referencia a estos ficheros es almacenada en la cadena de bloques por medio del uso del código de operación **OP_RETURN**.

Debido a su simplicidad y gran potencial, este proyecto utilizará el protocolo *Open Assets* para realizar la definición de *propiedades inteligentes*. Por tanto, se hace necesario un estudio previo del mismo para comprender cómo asegura la emisión y transferencia de activos.

2.4. El protocolo Open Assets

Uno de los protocolos más potentes para la definición y transferencia de activos es el protocolo *Open Assets* [9], que está implementado sobre las salidas de las transacciones de la cadena de bloques. Dentro de este protocolo, existen dos campos fundamentales:

- El campo denominado **identificador de activo** es un *hash* de 160 bits que se utiliza para identificar al activo de forma única.
- El campo denominado **cantidad de activos** es un entero sin signo que se utiliza para representar el número de activos que guarda la salida de la transacción.

El **identificador de activo** se obtiene como sigue:

1.– Se toma la salida referenciada por la **primera** entrada de la transacción que se utiliza para emitir el activo.
2.– Se obtiene el *script* de dicha salida y se le aplica la función *hash SHA-256*.
3.– Al resultado del paso anterior se le aplica la función *hash RIPEMD-160*, obteniendo el **identificador de activo**.

Dado que una transacción puede tener múltiples salidas, estas se clasifican en dos tipos, *coloreadas* y *no coloreadas*:

- Las salidas *coloreadas* son aquellas que contienen un **identificador de activo** y un campo de **cantidad de activos** no nulo.
- Las salidas *no coloreadas* mantienen estos campos indefinidos. Este tipo puede utilizarse como salidas normales para la transferencia de *bitcoins*.

Las transacciones que usen el protocolo deben tener una salida especial, denominada *marker output* (*marcador*). Esta es la salida que contiene el *script* que utiliza el código **OP_RETURN** para introducir los datos necesarios que definen los parámetros del protocolo. Además, sirve como medio para identificar que una transacción utiliza el protocolo *Open Assets*.

Campo	Descripción	Tamaño
Prefijo OAP	Prefijo utilizado para indicar que la transacción sigue el protocolo Open Assets. Su valor es fijo: 0x4f41.	2 bytes
Número de versión	El número de versión del protocolo que se está utilizando. En la fecha en la que se escribe este documento sólo existe una versión, cuyo valor es: 0x0100.	2 bytes
Tamaño de la lista de cantidades.	Un entero de un tipo propio de Bitcoin denominado compactSize [5] que representa el tamaño de la lista de cantidades de activos.	1-9 bytes
Lista de cantidades	Una lista de cero o más enteros sin signo codificados con LEB128A. Esta lista representa la cantidad de activos para cada salida de la transacción, incluyendo el marcador, en cuyo caso será cero.	Variable
Longitud de los metadatos.	Un entero de tipo compactSize que representa la longitud del campo de metadatos.	1-9 bytes
Metadatos	Metadatos asociados a esta transacción. Este campo puede ser vacío.	Variable

Tabla 2.1: Contenido del marcador en el protocolo Open Assets.

En la tabla 2.1 se describe cuál es la información requerida que se encuentra en el *marcador*. Dependiendo de su posicionamiento, se identifican dos tipos de salida dentro de este protocolo:

- Las salidas que se encuentren posicionadas antes del *marcador* serán entendidas por el protocolo como **salidas de emisión**, es decir, las utilizadas para emitir un nuevo activo.
- Las salidas que se encuentren posicionadas tras el *marcador* serán las **salidas de transferencia**, es decir, las utilizadas para transferir un activo a otra dirección.

En ambos casos, si la cantidad de activos indicada es nula, la salida sera considerada como *no coloreada*. De este modo, además de indicar que la transacción sigue el protocolo y ofrecer información relativa al activo, el *marcador* sirve como separador entre los dos tipos de acciones que se pueden realizar, la de emisión y la de transferencia.

Para dar seguridad a las transacciones que manejan activos, **el protocolo no incluye el identificador de activo** directamente en los campos que lo definen (ver la tabla 2.1). De esta forma, para poder realizar la transferencia de un activo se comprueba que se dispone de una salida de transacción que lo posee. Para ello, se utilizan los enlaces hacia atrás de las transacciones hasta alcanzar la salida de emisión del mismo. En ese momento, se computa el **identificador de activo** de acuerdo al protocolo y se permite la transferencia.

3 Diseño y desarrollo

En este capítulo se ofrece, en primer lugar, una visión general del sistema completo incluyendo su relación con los dispositivos asociados al mismo. Posteriormente, se proporcionan los detalles de diseño y desarrollo tanto del sistema propuesto como del prototipo de *propiedad inteligente* construido con el uso del microcontrolador *Arduino Yún*.

3.1. Descripción general

El sistema propuesto en este proyecto constituye un registro de propiedades público y distribuido que utilizará la cadena de bloques de *Bitcoin* y el protocolo *Open Assets* para obtener la información relativa a las *propiedades inteligentes*. Al permitirse la emisión y transferencia de activos por parte de cualquier usuario de la red, será necesario que el sistema sea capaz de extraer, analizar y mantener actualizada toda la información relativa a las *propiedades inteligentes* a partir de los datos contenidos en la cadena y el uso de este protocolo.

La información relativa al dueño de cada *propiedad inteligente* deberá poder ser consultada por estas de forma periódica para que puedan mantener identificado a su dueño actual en todo momento.

Al ser emitidos, los activos corresponden a un propietario inicial, y este tiene la capacidad de transferir su propiedad a otros usuarios por medio de la red. Cuando las *propiedades inteligentes* sean transferidas y las transacciones sean validadas e incluidas en la cadena, mediante las consultas periódicas que estas realizan advertirán que su dueño ha cambiado y dejarán de responder al propietario anterior.

De esta forma se les consigue otorgar cierta **inteligencia**, dado que no se les podrá indicar a quién pertenecen porque tienen **independencia** para consultarlo por sí mismas.

Para conseguir todo esto, el sistema estará compuesto de varias partes que interactuan entre sí para poder ofrecer las funcionalidades requeridas. En la figura 3.1 se muestra una visión general del sistema descrito:

- El sistema dispone de un nodo completo de la red *Bitcoin* que se encarga de validar y mantener actualizada una copia propia de la cadena de bloques.
- Un componente extrae la información relativa a las transacciones que utilicen el protocolo *Open Assets* a medida que lleguen nuevos bloques y estos sean validados.

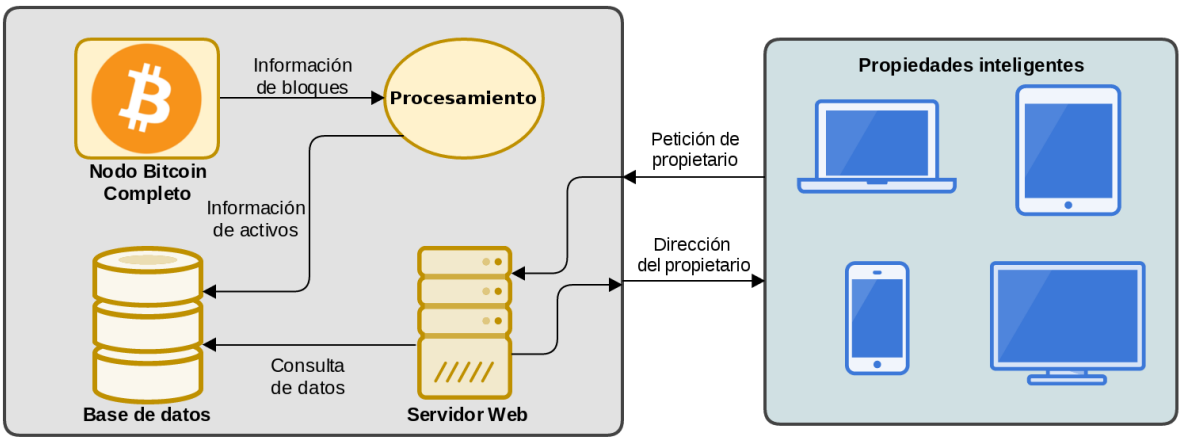


Figura 3.1: Visión general del sistema.

- Los datos recogidos serán procesados y la información de las *propiedades inteligentes* se almacenará en una base de datos.
- Un servidor web se encargará de exponer esta información de modo que pueda ser accedida por los dispositivos asociados a este sistema para que puedan conocer en todo momento el identificador de su propietario actual.

De esta manera, y dado que se utiliza la red *Bitcoin* como base de este sistema, las *propiedades inteligentes*, al consultar el identificador de su dueño, obtendrán su *dirección Bitcoin*. Puesto que estas direcciones están asociadas a una clave privada que sólo posee el propietario, los dispositivos pueden autentificar el origen de las peticiones al requerir que se firme cierta información proporcionada cuando estas sea recibidas.

3.2. Sistema de registro de propiedades

El sistema está dividido en varios componentes que se integran entre sí para obtener las funcionalidades requeridas:

- El nodo *Bitcoin* se encarga de mantener una copia actualizada de la cadena de bloques, validando y almacenando en disco cada nuevo bloque recibido.
- Un componente denominado ***OASync*** dispone de varios módulos que se encargan de la lectura de los datos de la cadena de bloques y de la identificación y procesamiento de las transacciones que utilicen el protocolo *Open Assets* para extraer la información relativa a las *propiedades inteligentes*. Asimismo, mantiene estos datos actualizados a medida que lleguen nuevos bloques y estos sean validados.
- Un componente denominado ***OADB*** define las estructuras de datos y sirve de interfaz para el acceso y el almacenamiento de la información en la base de datos.
- Un componente denominado ***OAServer*** se encarga de mantener disponible en un servidor web parte de la información recopilada para peticiones de consulta.

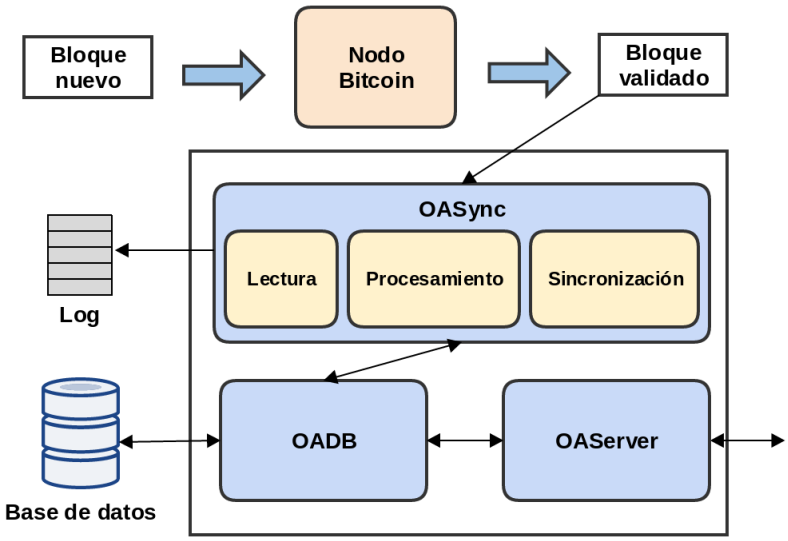


Figura 3.2: Estructura del sistema.

En la figura 3.2 se observa un esquema de la relación de los componentes dentro del sistema. Tras la llegada de un nuevo bloque, el nodo *Bitcoin* se encarga de su validación y almacenamiento en disco. *OASync* leerá la información de los bloques y detectará aquellas transacciones que usen el protocolo *Open Assets*, registrando su actividad. Estas transacciones serán procesadas y la información relativa a las *propiedades inteligentes* será almacenada en una base de datos mediante la interfaz proporcionada por *OADB*. Además, el componente se encargará de mantener actualizados los datos a medida que lleguen nuevos bloques y estos sean validados. Por otro lado, *OAServer* recibirá peticiones externas de la información almacenada y utilizará el acceso a la base de datos para poder proporcionarla.

A continuación se describen los distintos componentes que conforman el sistema, cómo se configura dicho sistema y qué plataformas han sido utilizadas para su implementación.

3.2.1. Nodo Bitcoin

Bitcoin ofrece un cliente, denominado *Bitcoin Core* que permite establecer una conexión con varios nodos para obtener una copia de la cadena de bloques y poder validarla. Este cliente permite elegir mediante previa configuración entre varios entornos que disponen de cadenas de bloques independientes de estructura similar:

- La red ***mainnet*** es la red principal de *Bitcoin* en la que las monedas digitales tienen valor económico real.
- La red ***testnet*** es un entorno **global** que se ofrece a los desarrolladores para realizar pruebas.
- La red ***regtest*** es un entorno **local** que se ofrece a los desarrolladores para realizar pruebas.

La información de los bloques se escribe en varios ficheros de forma secuencial a medida que estos son recibidos por el cliente. Los bloques se almacenan consecutivamente siguiendo la estructura de datos que se indica en la figura 3.3.

Todos los campos se almacenan en *little-endian* exceptuando aquellos que sean resultado de una función *hash*, esto es, el *hash* de la cabecera de un bloque, la *raíz Merkle* y el *hash* de cada transacción, que sirve para su identificación.

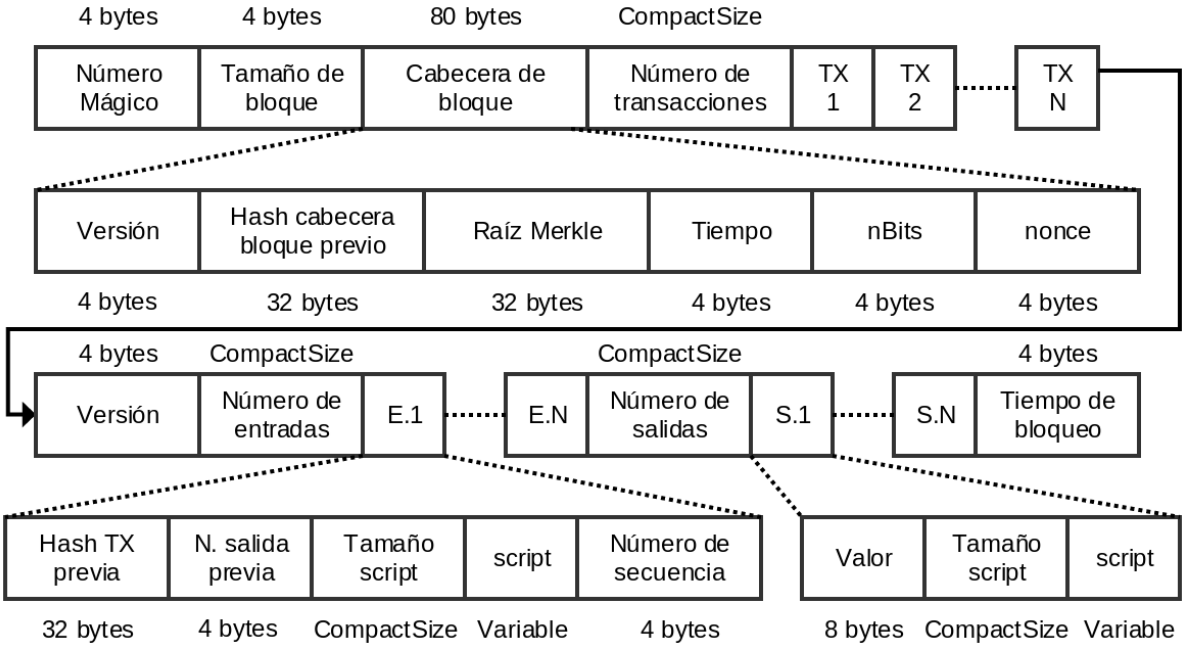


Figura 3.3: Estructura de un bloque en los ficheros binarios.

A excepción del *número mágico* y el *tamaño de bloque*, una descripción del resto de los campos puede ser consultada en la especificación del protocolo [5]. El *número mágico* es una secuencia de 4 bytes, cuyo valor se puede consultar en la tabla 3.1, que sirve como indicador del comienzo un bloque nuevo. El *tamaño de bloque* simplemente indica la cantidad de bytes que ocupa toda la estructura.

Red	Valor
mainnet	0xD9B4BEF9
testnet	0x0709110B
regtest	0xDAB5BFFA

Tabla 3.1: Valor del número mágico en función de la red.

Dado que *Bitcoin* utiliza una red *P2P* para obtener los bloques en paralelo, estos son almacenados de forma desordenada, por lo que si se realiza una lectura secuencial de uno de estos ficheros binarios es posible encontrar bloques cuyos predecesores se encuentran más adelante en el fichero o incluso en uno posterior. Puesto que las transacciones que utilizan el protocolo *Open Assets* deben utilizar las referencias a transacciones anteriores para identificar a los activos que manejan, el componente que realiza la lectura y el procesamiento de las mismas deberá solventar este problema.

3.2.2. Componente OASync

OASync es el componente del sistema que tiene acceso directo a los datos de la cadena de bloques. Cuando llega un bloque nuevo y el cliente *Bitoin Core* lo valida, los datos del mismo se escriben en disco en formato binario. Este componente tiene que leer todos los bloques recibidos y almacenados, analizar todas las transacciones, detectar aquellas que utilicen el protocolo *Open Assets* y extraer de las mismas la información relativa a las propiedades inteligentes. Para ello, dispone de varios módulos que se encargan de realizar estas funciones.

Para facilitar la recuperación ante fallos del sistema, este componente podrá ser suspendido. De esta manera, con la lectura de cada bloque se comprobará si se ha recibido la señal de interrupción y se almacenará una referencia al último bloque procesado para tener un punto de partida en el momento del inicio del sistema.

Módulo de lectura

Este módulo se encarga de la lectura de los datos y la identificación de las transacciones que utilizan el protocolo *Open Assets* para su posterior procesamiento.

Como ya se ha explicado, los bloques en los ficheros se encuentran en desorden, lo que imposibilita que el procesamiento de las transacciones se realice a la par que la lectura, pues el uso del protocolo *Open Assets* requiere el acceso a los datos de transacciones anteriores a una dada para poder identificar el activo que se maneja en una transferencia.

Para solventar esta situación, en primer lugar, este módulo lee los bloques de forma secuencial, calculando el *hash* de sus cabeceras para obtener los identificadores y almacenando su localización en el disco. Cuando esto se ha realizado, pasa al análisis de las salidas de las transacciones con el objetivo de identificar aquellas que siguen el protocolo *Open Assets*. Para ello, se comprueba que las transacciones contengan una salida en cuyo *script* se defina correctamente el *prefijo OAP* y el resto de campos del *marcador* especificados por el protocolo (ver sección 2.4).

Una vez se haya reconocido una transacción que use el protocolo, se calcula el *hash* de la estructura completa para obtener el identificador de la misma y poder almacenar su localización en el disco además de una referencia al bloque que la contiene.

De esta forma, se consigue construir un índice de todas las transacciones que siguen el protocolo y permitir que el módulo que se describe en el próximo apartado lo utilice para realizar un procesamiento de las mismas y extraer la información relativa a las *propiedades inteligentes*.

Módulo de procesamiento

Cuando se completa la tarea de lectura e identificación de todas las transacciones que siguen el protocolo *Open Assets*, este módulo realiza un procesamiento de las mismas que tiene como objetivo extraer la información relativa a las *propiedades inteligentes*.

Todas las *salidas coloreadas* de estas transacciones deberán ser analizadas para poder computar el identificador del activo que manejan.

En el caso de las **emisiones** la operación es sencilla, puesto que basta con obtener el *script* de la salida referenciada por la primera entrada de la transacción que realiza la emisión y aplicarle las funciones *hash SHA-256* y *RIPEMD-160* respectivamente, obteniendo así el identificador.

Sin embargo, en el caso de las **transferencias**, hay que utilizar los enlaces de las transacciones hacia atrás para llegar a la transacción de emisión y realizar el cálculo recién descrito. El proceso de asignación de identificadores de activo a las salidas comienza con la primera entrada de la transacción y la primera salida colocada tras el *marcador*:

- Cada entrada que referencie a una *salida coloreada* tendrá asociados sus campos de **identificador de activo** y de **cantidad de activos**. Se considera que las entradas que referencien a *salidas no coloreadas* tienen un campo de **cantidad de activos** nulo.
- El **identificador de activo** de cada unidad de las entradas se va asignando en orden a cada unidad de las salidas hasta que todas las unidades indicadas por el campo de **lista de cantidades** en el *marcador* hayan recibido un identificador. Si hay menos unidades en las entradas que las indicadas en el *marcador*, la operación será considerada inválida.
- Si a todas las unidades que contiene una salida se les ha asignado el mismo **identificador de activo**, se considerará que dicha salida tiene tal identificador. En caso contrario, la operación será considerada inválida.

En el primer punto del proceso recién descrito, el campo de **identificador de activo** de la salida coloreada referenciada deberá ser obtenido de forma recursiva repitiendo el proceso sobre la transacción que la contiene, hasta llegar a la salida de emisión.

Esto puede suponer un problema en cuanto al tiempo necesario de computación si un activo ha sido transferido múltiples veces, por el hecho de tener que volver a realizar el recorrido para localizar la emisión con cada una de las transacciones que compongan la cadena de transferencias.

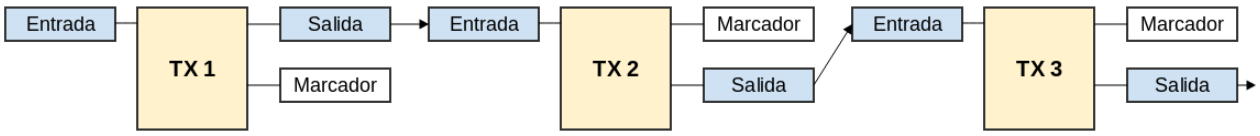


Figura 3.4: Ejemplo simple de transferencias de activos.

En el ejemplo de la figura 3.4 se observa que para calcular el identificador de activo manejado por la salida de la transacción 2 hay que ir hacia atrás para llegar a la salida de la transacción 1, que se corresponde con la transacción de emisión de dicho activo. Sin embargo, para calcular el identificador del activo que maneja la salida de la transacción 3, habría que analizar de nuevo la salida de la transacción 2, lo que supone repetir un proceso que ya ha tenido lugar.

Para evitar tener que repetir el cálculo de identificadores, dado que la información de las

propiedades inteligentes se encuentra en estas salidas, se almacenan todos sus datos una vez completada la tarea de validación y procesamiento. De esta forma, cuando se acceda a una salida que ya ha sido procesada, se podrán consultar los datos almacenados.

Por otro lado, el procesamiento de las transacciones incluye la extracción del *hash* de la clave pública del usuario al que va dirigida cada una de sus salidas, a partir de los *scripts* que contienen las mismas. Asimismo, del *marcador* de cada transacción que contenga una salida de emisión se extraen los metadatos asociados a los activos emitidos.

Finalmente, se lleva un registro de qué transacciones han sido ya procesadas para evitar repetir operaciones y tener un punto de partida en caso de suspensión y reinicio del sistema.

Módulo de sincronización

Una vez se hayan obtenido los datos relativos a las *propiedades inteligentes*, el componente debe comprobar la llegada de nuevos bloques para poder repetir los procesos de lectura y procesamiento y así mantener actualizada la información.

Este módulo se encarga de mantener una conexión con el nodo *Bitcoin* y realizar una llamada, cada cierto tiempo establecido por configuración, a un procedimiento remoto (*RPC*) que ofrece el cliente. Dicho procedimiento permite obtener el *hash* de la cabecera del último bloque recibido, que se compara con el del último bloque analizado. En caso de que difieran, el módulo se encarga de iniciar las operaciones de lectura y procesamiento para, una vez se hayan completado, volver a comprobar si se ha producido la llegada de un nuevo bloque.

Este proceso continua de forma indefinida mientras no se suspenda el componente por medio de una señal de interrupción, consiguiéndose así que la información que se almacena en la base de datos esté actualizada de acuerdo a los datos más recientes contenidos en la cadena de bloques.

3.2.3. Componente OADB

OADB sirve de interfaz de acceso a la base de datos para el resto de componentes del sistema.

Para ello, define estructuras de datos y ofrece métodos de acceso y almacenamiento que se utilizan para manejar la información de los bloques, transacciones y salidas de transacciones, facilitando así los procesos de extracción de datos de la cadena de bloques y la exposición de los mismos a través del servidor web.

Estas estructuras son definidas de acuerdo al modelo utilizado para almacenar la información en la base de datos. En la figura 3.5 se muestra un diagrama que representa dicho modelo.

Los atributos propios de las salidas se utilizan para la identificación de las mismas por otras transacciones y evitar la repetición de las operaciones de procesamiento, tal y como se ha descrito. Además, dado que contienen la información de los activos, se utilizan para atribuir la propiedad de los mismos a las direcciones que identifican a los dueños.

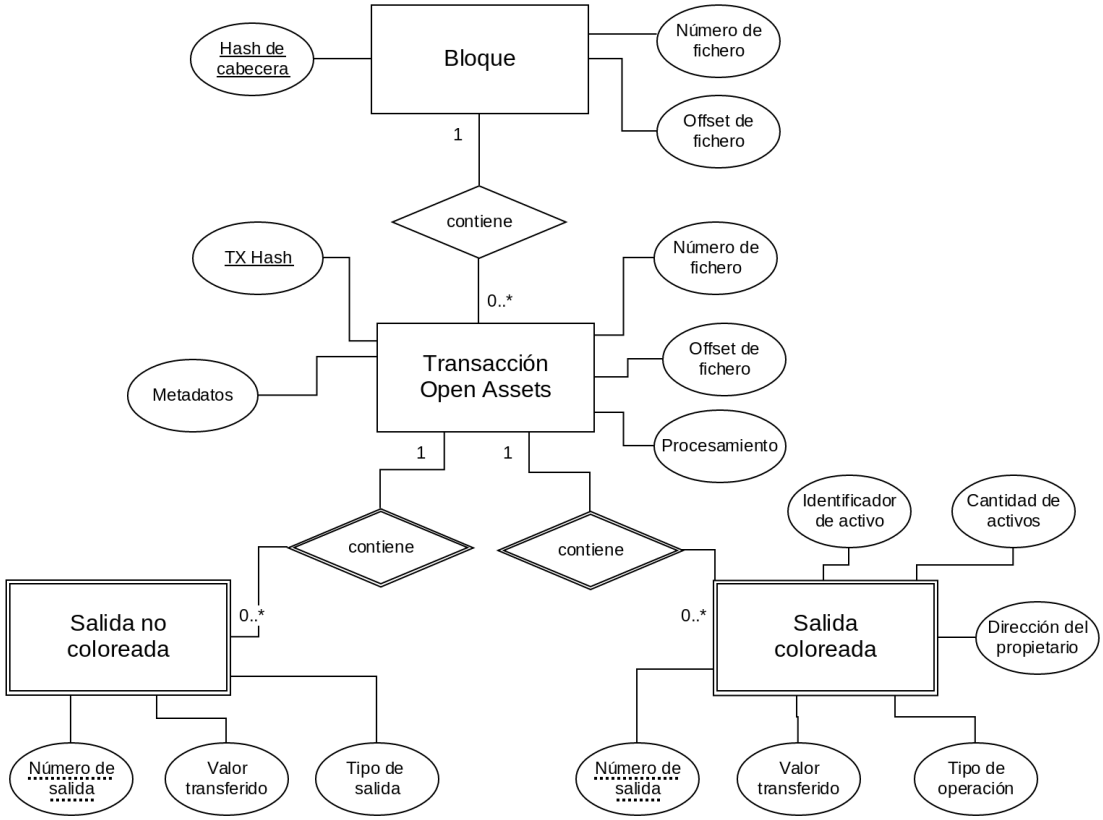


Figura 3.5: Diagrama entidad-relación de la base de datos.

Además de las estructuras descritas en el diagrama, el componente almacenará una referencia al último bloque procesado, de manera que sirva como punto de partida en caso de suspensión y reinicio del sistema y para comprobar si se han recibido nuevos bloques a través del cliente *Bitcoin Core*.

3.2.4. Componente OAServer

Este componente se encarga de exponer los datos extraídos de la cadena de bloques a petición de consulta, de manera que las *propiedades inteligentes* puedan conocer en todo momento el identificador de su propietario. Consiste en un servidor que ofrece una *API* sencilla que permitirá acceder a dichos datos mediante una conexión segura, para evitar ataques de suplantación de identidad.

La información referente a los propietarios y *propiedades inteligentes* se extrae a partir de las *salidas coloreadas* y de las transacciones que siguen el protocolo. De esta manera, cuando se recibe una petición, se utiliza la interfaz de acceso a la base de datos para obtener la información que el componente *OASync* mantiene actualizada.

Las relaciones de contención entre salidas, transacciones y bloques, permiten reconocer la operación más reciente relativa a una *propiedad inteligente* identificada de forma única, utilizando para ello las confirmaciones del bloque al que pertenezcan, que son obtenidas por medio del cliente a partir del *hash* de su cabecera.

El servidor permite obtener el identificador del dueño de una *propiedad inteligente* y los *metadatos* asociados al mismo. Asimismo, ofrece un método para permitir que los dispositivos comprueben que la firma de un mensaje recibido corresponde a una dirección dada. Una descripción de estos métodos puede ser consultada en la tabla 3.2.

El primer método es el que utilizará el prototipo para comprobar, a partir de su identificador, cuál es la dirección del dueño que le corresponde. Se ofrece un parámetro opcional que tiene el propósito de establecer un número mínimo de confirmaciones del bloque que contiene la última transferencia realizada del activo indicado. El valor por defecto tomará como válido hasta el último bloque recibido por el sistema.

Método	Descripción	Parámetros	Valor
/owner	Permite a los dispositivos consultar en todo momento cual es su propietario actual.	**asset_id**: Identificador de un activo. **min_conf**: Número mínimo, más uno, de confirmaciones del bloque que contiene la última transferencia relativa al activo. Parámetro opcional. Valor por defecto: 1.	Identificador del dueño actual de una propiedad inteligente.
/metadata	Permite a los usuarios conocer la información de un activo definida por cada emisor.	**asset_id**: Identificador de un activo.	Metadatos relativos a un activo.
/verify	Permite verificar que un mensaje corresponde a una firma realizada con la clave privada asociada a una dirección.	**address**: Dirección con la que se firma el mensaje. **signature**: Firma del mensaje. **message**: Mensaje del que se verifica la firma.	El resultado es *true* si el mensaje se ha firmado con la clave privada asociada a la dirección o *false* en caso contrario. Si los parámetros introducidos no corresponden a una dirección o firma válidas, el resultado será *null*.

Tabla 3.2: Métodos de la API que ofrece el servidor.

Este servidor ofrece la información mediante una conexión segura, utilizando el protocolo *TLS*, de manera que pueda ser autenticado por los dispositivos que accedan a la información y evitar *ataques de man-in-the-middle*.

Para ello, se ha creado una autoridad de certificación propia (*Certification Authority*, *CA*) que emite un certificado para el servidor. Los dispositivos que usen el sistema mantendrán confianza en esta entidad de modo que podrán autentificar al servidor al que se conectan.

3.2.5. Configuración del sistema

El sistema dispone de un fichero de configuración en el que se indican ciertos parámetros necesarios para realizar todas las funcionalidades detalladas.

A continuación se describen estos parámetros:

- Los parámetros de **usuario**, **contraseña**, ***host***, y **puerto** son necesarios para establecer la conexión con el cliente. Los valores de estos parámetros vienen especificados en el fichero de configuración del mismo.
- Se indica la **red** sobre la que se está trabajando, ya sea *mainnet*, *testnet* o *regtest*, y el **directorio** donde se encuentran los ficheros que contienen los bloques para que el sistema pueda acceder a los mismos.
- Se indica, mediante otro parámetro, cada cuánto **tiempo** el módulo de sincronización debe realizar peticiones al cliente para consultar el identificador del último bloque recibido.
- Se especifica el **fichero** que utiliza la **base de datos** (ver apartado 3.2.6, descripción de *SQLite*) donde se almacena la información extraída.
- Se incluyen también la **ruta** y el **número** y **tamaño máximo** de los ficheros que se van a utilizar para el **registro de actividad**.

3.2.6. Plataformas utilizadas

Para el desarrollo del sistema de registro se ha utilizado el lenguaje de programación ***python*** en su tercera versión, un lenguaje interpretado que soporta orientación a objetos y programación imperativa.

Para el mantenimiento de una copia actualizada de la cadena de bloques se ha utilizado el cliente ***Bitcoin Core***, que valida y almacena los bloques recibidos y ofrece métodos para obtener información sobre los mismos.

En el componente de extracción de datos se usa una librería, denominada *python-bitcoinlib* [20] que sirve como interfaz al protocolo y que además contiene una implementación que permite realizar peticiones al cliente *Bitcoin Core*. Asimismo, se ha utilizado la librería *openassets* [8] como una referencia a la implementación del protocolo, ya que dispone de una licencia libre permisiva que ha posibilitado realizar la implementación del procesamiento de las salidas de transacciones tal y como se ha descrito.

Para el almacenamiento de datos se utiliza ***SQLite*** y la correspondiente librería *sqlite3* de *python*. *SQLite* es un proyecto de dominio púbico que implementa un sistema de gestión de bases de datos relacionales. A diferencia de otros sistemas, *SQLite* no es un motor de bases de datos con arquitectura cliente-servidor, sino que pasa a formar parte íntegra de la aplicación. Esto reduce la latencia en el acceso a la base de datos debido a que las llamadas a funciones son más eficientes que las comunicaciones entre procesos. La base de datos completa se almacena en un único fichero en la máquina local.

Finalmente, para el servidor se ha utilizado ***CherryPy***, un *framework* que utiliza el lenguaje de programación *python* para permitir a los desarrolladores construir aplicaciones web de forma sencilla. La creación de la autoridad certificadora y la firma de certificados se ha realizado utilizando ***OpenSSL***, un proyecto de software libre que proporciona un conjunto de herramientas para manejar certificados digitales, entre otras funcionalidades.

3.3. Prototipo de propiedad inteligente

El prototipo de *propiedad inteligente*, construido con el uso del microcontrolador *Arduino Yún*, hará uso del sistema de registro implementado para conocer en todo momento el identificador de su propietario y poder autenticar el origen de las peticiones que reciba.

En primer lugar, en esta sección se ofrece una descripción del *hardware* utilizado para su implementación, incluyendo la plataforma *Arduino Yún*. Posteriormente, se proporcionan los detalles de dicho prototipo.

3.3.1. Descripción del hardware

En este apartado se ofrece una descripción del *hardware* empleado para construir el prototipo de *propiedad inteligente*, incluyendo la plataforma *Arduino Yún* y las interfaces de las que dispone el dispositivo para mostrar la información.

Con el objetivo de integrar estos componentes se ha utilizado una *impresora 3D* para construir un soporte para el dispositivo, tal y como se muestra en las figuras de la sección de resultados (ver sección 4.2).

Arduino Yún

Arduino Yún es un microcontrolador que ha sido diseñado para desarrollar proyectos relacionados con el *IoT*. Dispone de dos chips independientes que están conectados entre sí: el chip *ATmega32u4*, propio de *Arduino*, y el procesador *Atheros AR9331*, que soporta una distribución *Linux* basada en *OpenWrt*. Para la comunicación entre ambos se utiliza una librería, denominada *Bridge*, que permite lanzar procesos, ejecutar scripts y obtener los resultados en el lado de *Arduino*. Para las conexiones de red, dispone de *WiFi* (IEEE 802.11b/g/n) y *Ethernet* (IEEE 802.3 10/100Mbit/s). En la figura 3.6 se muestra un esquema de la plataforma.

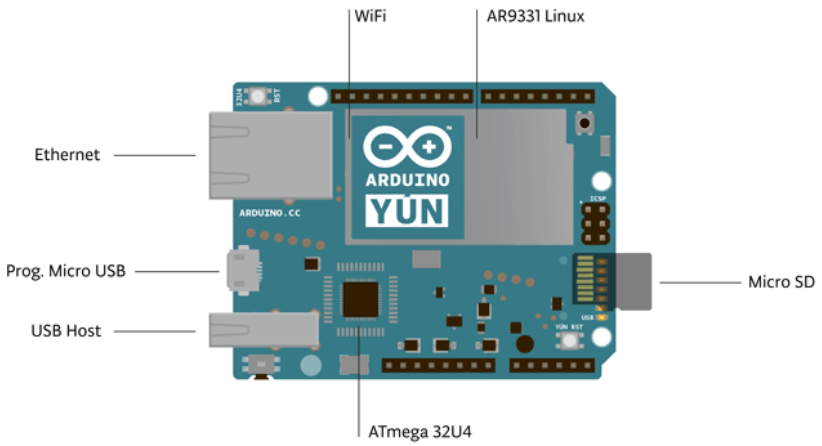


Figura 3.6: Esquema del Arduino Yún.

El hecho de que *Arduino Yún* utilice una distribución *Linux* es lo que le diferencia del resto de microcontroladores de *Arduino*. Con el uso de comandos como *cURL* y la posibilidad de ejecutar *scripts* usando el lenguaje *python* se otorga robustez a las comunicaciones, permitiéndose así realizar conexiones seguras.

Interfaz

Para mostrar la información, el dispositivo dispondrá de dos interfaces. En primer lugar, se utiliza la pantalla *LCD Nokia 5110* que se muestra en la figura 3.7. Asimismo, se utilizará un *led RGB* para indicar estados relativos a conexión y autenticación, como se explica más adelante.

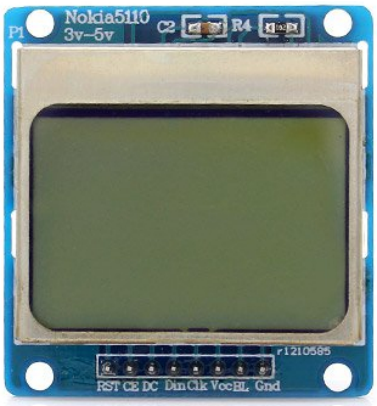


Figura 3.7: Pantalla LCD Nokia 5110.

La pantalla *LCD* dispone de 8 pines que se describen en la tabla 3.3. Para el control de la pantalla se utiliza la librería *Adafruit PCD8544 Nokia 5110 LCD*, que ofrece una interfaz sencilla para enviar datos a la misma. Esta librería permite crear un objeto que representa la pantalla y utilizar funciones para imprimir cadenas de caracteres, figuras geométricas y mapas de bits. Para poder utilizarla, basta crear un objeto e indicar en qué pines del microcontrolador se han conectado los pines 1-5 descritos en la tabla 3.3.

Número de pin	Nombre de pin	Descripción
1	RST - Reset	Permite reiniciar el dispositivo.
2	CE - Chip Enable	Permite la entrada de datos.
3	DC - Data/Command	Permite seleccionar entre entrada de comandos o de datos.
4	Din	Entrada para la línea de datos.
5	Clk	Clock. Entrada de reloj.
6	Vcc	Alimentación.
7	BL - Backlight	Alimenta la retroiluminación.
8	Gnd	Tierra.

Tabla 3.3: Descripción de los pines de la pantalla LCD Nokia 5110.

3.3.2. Descripción del prototipo de propiedad inteligente

El prototipo de *propiedad inteligente* dispondrá de un identificador único, que corresponde al de un activo emitido con el uso del protocolo *Open Assets*. De manera periódica, realizará peticiones al sistema para obtener la dirección *Bitcoin* de su propietario a partir de su propio identificador. Estas peticiones serán realizadas mediante una conexión segura para evitar *ataques de man-in-the-middle*.

Mientras mantenga la conexión con el sistema, el dispositivo puede autenticar a su propietario actual a partir de una firma que debe realizarse con la clave privada asociada a su *dirección Bitcoin*, indicando mediante la interfaz de la que dispone si dicha autenticación resulta exitosa.

Con cada petición, el prototipo actualizará el identificador de su propietario, por lo que si se le inhabilita la conexión no podrá realizar dicha actualización. En este momento, entrará en un modo en el que intentará restablecer las comunicaciones con el sistema y no podrá autenticar las peticiones recibidas. Una vez lo consiga, volverá a aceptar la entrada de peticiones externas. De esta forma, se evita que si el dispositivo no puede actualizar el identificador de su dueño debido a que se le ha inhabilitado la conexión, se mantenga el último recibido de forma indefinida.

Identificador y definición del prototipo

Como se ha descrito, el prototipo debe tener un identificador único, que será mostrado por medio de la interfaz en la etapa de inicialización. Este identificador, una vez generado, se define en el software del microcontrolador y sólo es consultado para su lectura.

Para su generación se utiliza la herramienta *colorcore* [10], un *software* que funciona sobre *Bitcoin Core* y que es compatible con el protocolo *Open Assets*. Esta herramienta permite realizar las operaciones de emisión y transferencia de activos sobre cualquiera de las redes de *Bitcoin* mediante previa configuración.

Además, con cada emisión se permite introducir ciertos metadatos codificados en *UTF-8* que sirven para referenciar la *URL* donde se proporciona la definición del dispositivo. Los desarrolladores de *Open Assets* ofrecen un protocolo [7] para esta definición que introduce como metadatos una cadena denominada *asset definition pointer* (puntero de definición de activo). Entre los varios formatos que ofrecen, se utiliza el siguiente por considerarse el más apropiado:

```
u=<URL de definición>&sha256=<sha256 hash>
```

El primer parámetro que contiene el puntero referencia a la *URL* donde se encuentra la definición del activo. El segundo parámetro, resultado de aplicar la función *SHA-256* a la definición, permite comprobar que la información referenciada no ha sido modificada. Por otro lado, la emisión del mismo se realiza con un valor de uno para el campo de cantidad de activos del protocolo. De este modo, se utiliza el identificador del dispositivo generado para reconocer al mismo de forma única.

Consulta de propietario y autenticación

El prototipo realiza peticiones de forma continua al sistema de registro para consultar la dirección de su propietario a partir de su propio identificador. Esta conexión se realiza utilizando el protocolo *TLS*, de manera que se pueda autenticar al servidor del sistema y evitar *ataques de man-in-the-middle*.

Es necesario que el dispositivo confíe en la autoridad certificadora que ha creado el certificado para dicho servidor, de forma que se pueda realizar la autenticación correctamente. Para ello, *Arduino Yún* permite realizar conexiones por *SSH* e instalar la cadena de certificación generada en el lado de *OpenWrt*.

Una vez hecho esto, mediante el uso de la librería *Bridge*, el dispositivo ejecuta el comando *cURL*, indicando mediante la opción `--cacert` el fichero que contiene los certificados que se utilizarán para verificar al servidor con cada petición. La respuesta de este comando será devuelta al microcontrolador, de forma que pueda almacenar el identificador de su propietario actual y mostrarlo a través la pantalla *LCD*. Asimismo, se indicará mediante el *led* que se ha recibido respuesta del servidor, tal y como se muestra en la tabla 3.4.

Mientras el prototipo mantenga el identificador de su propietario, podrá recibir peticiones cuyo origen será autenticado utilizando un protocolo simple de reto-respuesta en el que el prototipo ya conoce de antemano la clave pública (codificada en forma de dirección *Bitcoin*) de su propietario [15] [18]:

$$A \rightarrow B : m \tag{3.1}$$
$$A \leftarrow B : r_B \tag{3.2}$$
$$A \rightarrow B : S_A(r_B) \tag{3.3}$$

En primer lugar, el usuario que quiere ser autenticado (A) envía un mensaje al prototipo (B) con la petición, que constará de la cadena “led” (3.1). En segundo lugar, el prototipo responde con un número aleatorio de 8 bytes generado a partir de `/dev/random`, utilizando para ello el módulo de *Linux* (3.2). Finalmente, el usuario realiza una firma del número aleatorio recibido utilizando su clave privada y lo envía de vuelta al prototipo (3.3). De esta manera, por medio de una petición al servidor, el prototipo podrá comprobar si la firma recibida corresponde al número aleatorio firmado con la clave privada de su propietario actual, evitando así que se puedan producir *ataques de repetición*.

Las peticiones serán recibidas por medio de la *consola* de *Arduino Yún*, accesible a través de la conexión *WiFi*. Cuando el dispositivo verifique la firma recibida, notificará su validez por medio del *led RGB*, tal y como se indica en la tabla 3.4.

Descripción	Led RGB
Sin respuesta del servidor	Apagado
Respuesta recibida	Azul
Propietario autenticado	Verde
Propietario no autenticado	Rojo

Tabla 3.4: Comportamiendo del led RGB.

4

Pruebas y resultados

En este capítulo, en primer lugar, se realiza una descripción de las distintas pruebas realizadas durante las etapa de desarrollo del sistema. Posteriormente, se muestra el funcionamiento del prototipo de *propiedad inteligente*.

4.1. Pruebas del sistema de registro

Durante la implementación del sistema de registro se han realizado distintas pruebas con el objetivo de detectar fallos en el desarrollo y comprobar cómo se comportan los distintos módulos a la hora de realizar las tareas de lectura de bloques, identificación de transacciones que utilicen el protocolo *Open Assets*, procesamiento y extracción de la información y mantenimiento de los datos actualizados.

Estas pruebas se han realizado utilizando los entornos de *test* que ofrece *Bitcoin*: *regtest* y *testnet*, puesto que la estructura de la cadena de bloques que manejan es similar a la de la red principal.

A continuación, se describe en qué han consistido estas pruebas, atendiendo a las distintas tareas descritas anteriormente.

Lectura de bloques e identificación de transacciones

Mediante el uso de ambos entornos, se han realizado varias pruebas sobre la lectura de bloques e identificación de transacciones.

En primer lugar, se han realizado pruebas mediante la **generación de bloques**. Estas pruebas han consistido en la construcción una cadena de bloques propia utilizando el entorno *regtest* de manera que el sistema (previamente configurado para funcionar sobre esta red) realizara una lectura completa de dicha cadena, produciéndose así el almacenamiento de la información relativa a los bloques en la base de datos. De esta forma, se ha podido comprobar, mediante métodos de inspección, que los bloques generados son correctamente identificados.

Asimismo, se han **generado transacciones** que usan el protocolo para comprobar que estas son identificadas. Estas pruebas han consistido en el uso de *scripts* que emplean la herramienta *colorcore* para crear transacciones de tres tipos: transacciones de emisión, transacciones de transferencia y transacciones que no utilizan el protocolo. De esta manera, con la

inclusión de estas transacciones en distintos bloques, se ha podido comprobar que el sistema puede detectar aquellas que sí utilizan el protocolo.

Procesamiento de las transacciones y extracción de información

Para comprobar que el sistema procesa las transacciones de forma correcta, se han realizado varias pruebas mediante generación de **emisiones**, **transferencias** y el uso de herramientas que actúan como **exploradores** de la cadena.

La emisión de activos se ha realizado para hacer una primera comprobación del procesamiento de las transacciones, en el que el sistema tiene que extraer la información de los identificadores, los metadatos asociados a los activos y las direcciones de los propietarios iniciales. Asimismo, se han construido cadenas de transferencias cuyas transacciones tienen que ser analizadas por el sistema para realizar el proceso de asignación de identificadores a las *salidas coloreadas* y extraer los datos relativos al propietario.

Por otro lado, se han utilizado herramientas en línea, que actúan como exploradores de la cadena de bloques y que ofrecen toda la información relativa a cualquier bloque o transacción, incluyendo la de los *scripts* correspondientes. De esta manera, tomando conjuntos de transacciones identificadas, se ha podido comprobar que los datos que contienen han sido correctamente procesados.

Actualización de datos

Para comprobar que el sistema mantiene los datos actualizados se han realizado pruebas mediante generación de bloques. Una vez comprobado, utilizando el registro de actividad, que el sistema entraba en la fase de sincronización, se realizaban múltiples emisiones, transferencias y generaciones de bloques para comprobar que los nuevos activos emitidos se añadían a la base de datos y que la información relativa a la propiedad de los mismos era actualizada conforme se producía la llegada de transacciones de transferencia.

4.2. Prototipo de propiedad inteligente

En esta sección se muestran resultados del funcionamiento del *prototipo de propiedad inteligente* y de cómo utiliza el sistema para conocer, en todo momento, el identificador de su propietario actual.

Para este prototipo se ha realizado la emisión de un activo con el uso de la herramienta *colorcore*, de forma que su propietario inicial correspondiese a una dirección propia de la que se dispone la clave privada asociada.

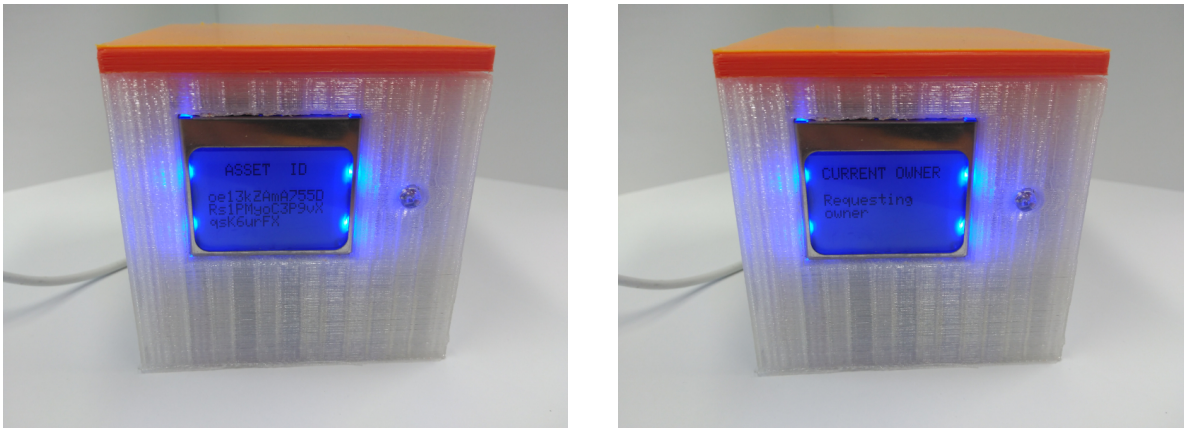
De este modo, la información del prototipo, una vez emitido, es la siguiente.

- Identificador único del prototipo: oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX
- Dirección *Bitcoin* del propietario inicial: mjiCDSFH3mLgLQaYohYgtGj6wwQZpuhEhf

A continuación se muestran los resultados del funcionamiento del prototipo, en cuanto a la obtención del identificador del propietario, la conexión con el sistema y la autenticación del propietario.

4.2.1. Obtención del identificador del propietario

El prototipo, una vez inicializado, muestra su identificador único por medio de la pantalla *LCD* (ver figura 4.1(a)). Una vez hecho esto, realiza una petición al servidor para obtener, por primera vez, la dirección *Bitcoin* que identifica a su propietario. Mientras obtiene la respuesta del servidor, muestra que se está requiriendo dicho identificador (ver figura 4.1(b)) y no acepta peticiones externas.



(a) El prototipo muestra su identificador de propiedad.

(b) Primera petición de identificador de propietario.

Figura 4.1: Etapa de inicialización del prototipo.

Una vez obtenido, mediante el *led* en color azul, se indica que se ha obtenido respuesta y se muestra de forma inmediata el identificador del propietario actual a través de la pantalla *LCD*, como se puede observar en la figura 4.2.

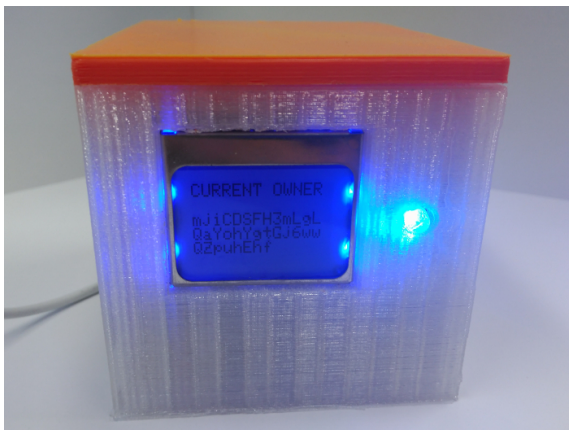


Figura 4.2: Conexión con el sistema y obtención del propietario actual.

4.2.2. Conexión con el sistema

Como ya se ha descrito, el prototipo realiza peticiones de forma continua, requiriendo el identificador de su propietario. Si en algún momento no obtiene respuesta, el *led* azul se apaga y se indica mediante la pantalla *LCD* que se está intentando restablecer las comunicaciones con el servidor (ver figura 4.3). En este momento, el prototipo no puede actualizar la información de su propietario, por lo que sigue realizando peticiones hasta poder obtener su identificador, sin aceptar solicitudes de autenticación externas.

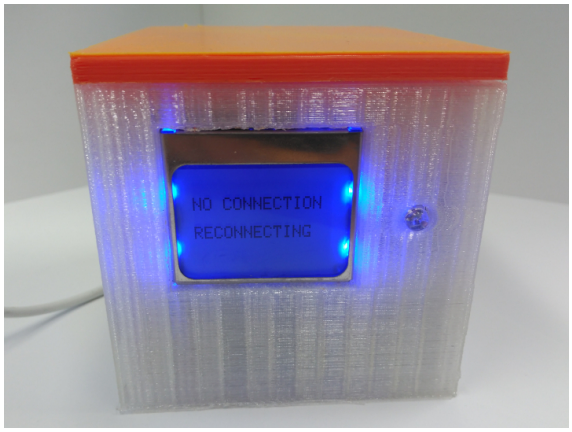


Figura 4.3: Restableciendo las comunicaciones con el servidor.

En la figura 4.4, se puede observar que durante cierto periodo, el servidor recibía peticiones del prototipo hasta que su conexión le fue inhabilitada durante aproximadamente un minuto (en la imagen, hasta la petición marcada en rojo). Durante ese tiempo, el prototipo se encontraba en el estado de la figura 4.3, en el que intenta restablecer las comunicaciones, hasta que se volvió a habilitar su conexión. En ese momento, el servidor comenzó de nuevo a recibir peticiones (en la imagen, a partir de la petición marcada en azul), de forma que el prototipo pudo volver al estado de la figura 4.2, en el que obtiene y muestra el identificador de su propietario.

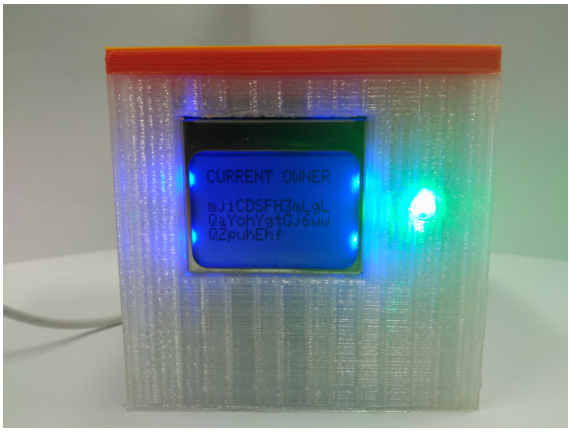
```
192.168.240.1 - - [24/Jun/2016:12:14:33] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:14:37] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:14:42] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:14:47] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:14:52] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:14:57] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:15:03] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:15:08] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:16:07] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:16:08] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:16:11] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:16:16] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:16:21] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:16:26] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
192.168.240.1 - - [24/Jun/2016:12:16:31] "GET /owner/?asset_id=oe13kZAmA755DRs1PMyoC3P9vXqsK6urFX HTTP/1.1" 200 34 "" "curl/7.29.0"
```

Figura 4.4: Respuesta de peticiones por parte del servidor.

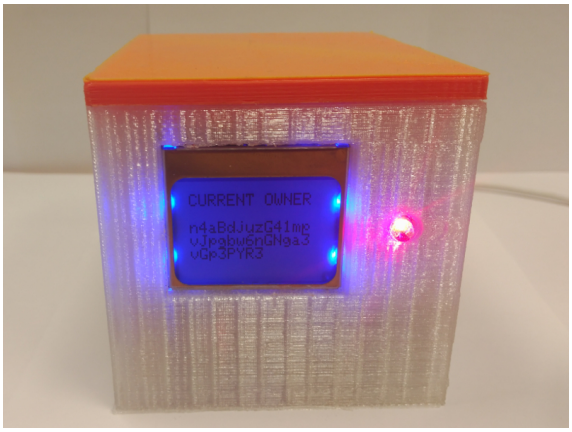
4.2.3. Autenticación del propietario

Como se indica al principio de esta sección, el prototipo se emitió con un propietario inicial del que se dispone la clave privada asociada a su dirección. Mediante un programa escrito en el lenguaje *python* que realiza una conexión con la consola del *Arduino Yún*, se envió una petición de autenticación cuya respuesta se firmó con la clave privada y fue devuelta al dispositivo. De esta forma, este dispositivo pudo comprobar mediante el uso del servidor que la firma correspondía al propietario que tenía registrado como actual, indicando la autenticación exitosa por medio del *led* en color verde, como se puede mostrar en la figura 4.5(a).

Una vez hecho esto, mediante la herramienta *colorcore*, se realizó una transferencia de la propiedad del dispositivo a otra dirección generada, de forma que cuando el bloque que contenía la transacción con la transferencia fue recibido por el sistema, el identificador del propietario actual se vio modificado. Tras ello, al solicitar la autenticación con la clave del propietario anterior, el dispositivo indicaba, mediante el *led* rojo, que la firma recibida no correspondía a su propietario actual (ver figura 4.5(b)).



(a) El prototipo reconoce a su propietario como origen de la petición.



(b) Tras realizar una transferencia, el prototipo deja de responder al propietario anterior.

Figura 4.5: Solicitudes de autenticación.

5 Conclusiones y trabajo futuro

La cadena de bloques proporciona un mecanismo para registrar los datos de transacciones monetarias digitales de forma pública y distribuida, de manera que resulta computacionalmente impracticable realizar una modificación de la información contenida en ella a no ser que se disponga de una capacidad de cómputo superior a la de la mayoría del resto de nodos de la red. Con ello, la confianza que había que establecer y mantener en una autoridad central para intervenir en las transacciones y en la verificación de los datos se ha transformado en una confianza hacia la red en sí misma, lo que ha generado que el concepto de cadena de bloques haya sido fuertemente analizado para extender sus aplicaciones más allá de las transacciones puramente monetarias. Gracias a la evolución que está experimentando, se han desarrollado, entre otras muchas aplicaciones, protocolos que permiten representar cualquier activo real mediante estas monedas digitales, dándole sentido al concepto de *propiedad inteligente*.

En este proyecto se ha estudiado la idea de aprovechar la capacidad que ofrece la cadena de bloques con el objetivo de construir un prototipo de *propiedad inteligente* que pudiera ser gestionado y transferido mediante el uso de la misma, y que pudiera utilizar la información contenida en ella para identificar a su propietario. Para ello, y dado que la plataforma utilizada sirve como representación de un dispositivo de baja capacidad de cómputo que no puede procesar la cadena de bloques por sí mismo, se ha conseguido desarrollar un sistema que sirve como acceso a dicha información, tanto para dispositivos como para usuarios, manteniéndola constantemente actualizada y cumpliendo así con los objetivos principales del trabajo.

Sin embargo, el estudio que ha requerido este proyecto, su desarrollo y los resultados finales, han llevado a considerar varias mejoras y extensiones que permitirían ampliar las posibles aplicaciones que pudieran derivar de su uso.

En primer lugar, el uso del protocolo *Open Assets* ha servido para registrar el prototipo de *propiedad inteligente* en la cadena de bloques, otorgándole un identificador y utilizando una cantidad de activos de una unidad para representarlo de forma única. Sin embargo, el sistema podría extenderse para utilizar este campo como método para permitir que un dispositivo disponga de más de un propietario, de manera que el emisor podría reutilizar su clave privada para generar más unidades y transferirlas a otros usuarios. Esto abriría un nuevo campo de aplicaciones en el que se permitiría compartir la propiedad de un sólo dispositivo, pudiendo este requerir, por ejemplo, de la autorización de todos sus propietarios para realizar ciertas tareas.

Asimismo, el campo de metadatos podría aprovecharse, en cada transferencia, para alojar información más allá del uso de protocolos de definición, consiguiendo de este modo nuevas funcionalidades. Dependiendo de las necesidades que se presenten, se podrían diseñar protocolos para almacenar datos relativos al estado de cada propiedad. Un primer ejemplo podría ser un dispositivo que deje de responder y realizar sus actividades porque detecta que ha sido robado, gracias a la información contenida en el campo de metadatos que su propietario añadió en la última transacción realizada.

Entre estas consideraciones de trabajo futuro también se incluye la de extender la *API* que ofrece el servidor y el sistema en sí mismo para que se comporte como un explorador completo de la cadena de bloques. De este modo, permitiría acceder a más información aparte de la relativa a las propiedades, incluyendo el acceso a todos los datos referidos tanto a los bloques como a las transacciones.

Por otro lado, la decisión de implementar un sistema que permita el acceso directo a la información alojada en la cadena de bloques ha sido una consideración de diseño previo que surgió debido a que el proyecto se enfocó hacia el campo de los dispositivos de bajo coste, lo que crea cierta dependencia por parte de dichos dispositivos hacia este sistema. Este hecho abre una nueva cuestión, que trata de cómo conseguir minimizar esa dependencia y que deberá ser estudiada.

Con la realización de este proyecto queda claro que la cadena de bloques tiene un gran potencial en el ámbito de la propiedad. El hecho de que los datos se mantengan íntegros y distribuidos y de que la confianza se deposite en la red, hace de la cadena de bloques una herramienta que permite reducir la confianza en terceras partes y proteger el control que los propietarios tienen sobre sus dispositivos.

Referencias

[1] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008. https://bitcoin.org/bitcoin.pdf (Último acceso: 30/05/2016).

[2] Nick Szabo. *The Idea of Smart Contracts*, 1997. http://szabo.best.vwh.net/idea.html (Último acceso: 01/06/2016).

[3] Bitcoin Core Documentation. *Technical background of version 1 Bitcoin addresses*. https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses (Último acceso: 15/06/2016).

[4] Bitcoin Project. *Bitcoin Developer Guide*. https://bitcoin.org/en/developer-guide (Último acceso: 15/06/2016).

[5] Bitcoin Project. *Bitcoin Developer Reference*. https://bitcoin.org/en/developer-reference (Último acceso: 15/06/2016).

[6] Daniel R. L. Brown. *SEC 2: Recommended Elliptic Curve Domain Parameters*. http://www.secg.org/sec2-v2.pdf (Último acceso: 03/06/2016).

[7] Flavien Charlon. *Asset Definition Protocol*. https://github.com/OpenAssets/open-assets-protocol/blob/master/asset-definition-protocol.mediawiki (Último acceso: 15/06/2016).

[8] Flavien Charlon. *Reference implementation of the Open Assets Protocol*. https://github.com/OpenAssets/openassets (Último acceso: 15/06/2016).

[9] Flavien Charlon. *Open Assets Protocol (OAP/1.0)*, 2012. https://github.com/OpenAssets/open-assets-protocol/blob/master/specification.mediawiki (Último acceso: 03/06/2016).

[10] Colorcore. *Colored coins wallet compatible with the Open Assets Protocol*. https://github.com/OpenAssets/colorcore (Último acceso: 15/06/2016).

[11] Sander Duivestein, Menno van Doorn, Thomas van Manen, Jaap Bloem, and Erik van Ommeren. *Design to disrupt. Blockchain: cryptoplatform for a frictionless economy*. Sogeti, 2015.

[12] IBM Institute for Business Value. *Device democracy. Saving the future of the Internet of Things*, 2015. http://www-935.ibm.com/services/multimedia/GBE03620USEN.pdf (Último acceso: 15/06/2016).

[13] Mike Hearn. *Smart property*. https://en.bitcoin.it/wiki/Smart_Property (Último acceso: 01/06/2016).

[14] Hermann Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications (Real-Time Systems Series)*. Springer, 2011.

[15] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[16] Alex Mizrahi. *A blockchain-based property ownership recording system*. http://chromaway.com/papers/A-blockchain-based-property-registry.pdf (Último acceso: 01/06/2016).

[17] Toby Padilla. *BIP 74. Allow zero value OP_RETURN in Payment Protocol*. https://github.com/bitcoin/bips/blob/master/bip-0074.mediawiki (Último acceso: 01/06/2016).

[18] Bruce Schneier. *Applied Cryptography. Protocols, Algorithms, and Source Code in C*. Wiley, 1996.
[19] Melanie Swan. *Blockchain: Blueprint for a new economy*. O'Reilly, Sebastopol, CA, 2015.
[20] Peter Todd. *Python Bitcoin library*. https://github.com/petertodd/python-bitcoinlib (Último acceso: 15/06/2016).