



Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Industriales
Grado de Ingeniería Tecnologías Industriales

Inserción de Datos de Sensores en Cadenas de Bloques Empleando Raspberry Pi y Tecnologías Blockchain

Trabajo Fin de Grado

Víctor Román Aragay

2018

Agradecimientos

Este trabajo está dedicado a las personas que más me han ayudado a lo largo de la carrera. En primer lugar, quiero agradecer profundamente a mis padres su apoyo, ánimos y comprensión que me han dado la fuerza necesaria para superar todos los obstáculos que he encontrado en mi vida. Terminar el grado de Ingeniería en Tecnologías Industriales es lo más complicado que he hecho hasta ahora con una gran diferencia, y les debo a ellos todo lo que he conseguido.

También se lo dedico a mis compañeros, con los que he compartido penas, alegrías y litros de café. Juntos nos hemos enfrentado a grandes retos y, cada uno a su ritmo, hemos conseguido salir victoriosos de ellos. Siempre con la conciencia de que los tropezones sufridos a lo largo del camino nunca supusieron fracasos, sino importantes oportunidades de aprendizaje, tremadamente valiosas para actuar mejor en el futuro.

A mis profesores, por ponerme a prueba de maneras inimaginables y cuya superación de las mismas han estimulado mi crecimiento y aprendizaje hasta límites que antes creía inalcanzables.

Por último, quiero dedicarle el trabajo a mi tutor Joaquín Ordieres por haberme brindado la oportunidad de embarcarme en el apasionante viaje que ha supuesto este proyecto y por introducirme al prometedor mundo de la tecnología Blockchain.

Resumen

Con la realización de este proyecto se busca estudiar las posibilidades de la integración de las tecnologías punteras del Internet de las cosas, junto con las bases de datos descentralizadas y distribuidas que son las Blockchain, con el objetivo de construir un sistema capaz de monitorizar en tiempo real las condiciones ambientales de una cierta área u objeto.

Con la llegada de la 4^a revolución industrial se ha conseguido, entre otras cosas, la ubicuidad de multitud de tecnologías que fueron disruptivas y claves en la consolidación y mejora de los procesos productivos e industriales como pueden: ser la conexión permanente a internet, la captación de datos por redes de sensores inalámbricas y la automatización de operaciones. La implementación y el desarrollo de estas tecnologías han supuesto la optimización de la producción y la fabricación de bienes, reduciendo significativamente los tiempos y costes necesarios para llevarlos a cabo.

En este contexto, surgen nuevas necesidades y problemas, que constituyen a su vez oportunidades para continuar la mejora y la optimización de las operaciones. En concreto, se considera el caso de una empresa distribuidora de mercancía y sus potenciales clientes. Estos agentes, tienen un gran interés el estado de la mercancía almacenada, transportada y distribuida, pues una deficiencia en cualquiera de los anteriores pasos podría corromper la mercancía y por tanto generar problemas con los acuerdos comerciales establecidos entre dichas partes.

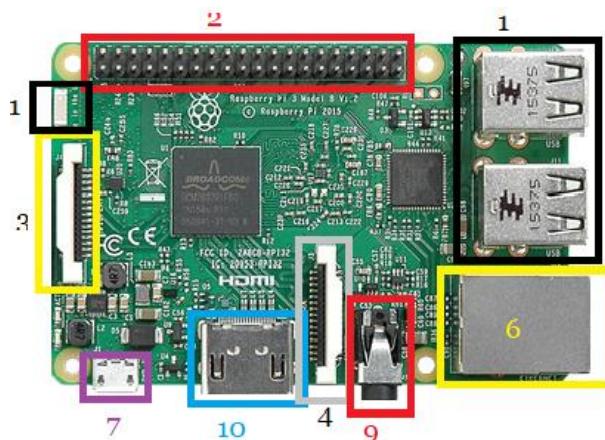
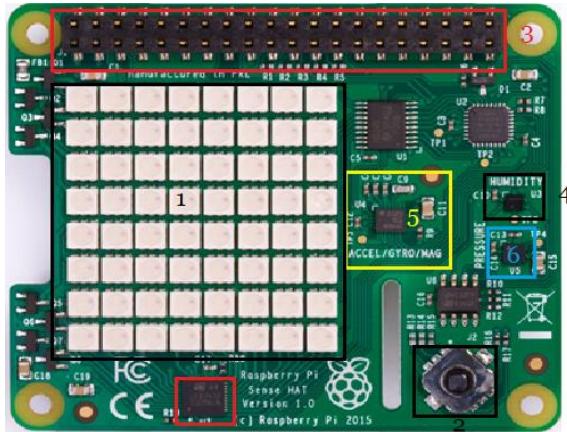
Para atacar este problema, se propone un sistema capaz de llevar a cabo una monitorización en tiempo real de las condiciones ambientales a las que se somete dicha mercancía, es decir: temperatura, presión, humedad, campos magnéticos, aceleraciones y orientación, que pueden dar información de su estado en todo momento y prevenir en caso de que alguna de las condiciones ambientales se vayan a salir de unos límites previamente considerados y establecidos entre las entidades involucradas.

El sistema propuesto, estará formado por un ordenador de placa única Raspberry Pi, un sensor conectado a dicha placa, un server local y otro descentralizado que actuarán como bases de datos en las que se registrarán las mediciones obtenidas por el sensor y por último un Libro de Cuentas Distribuido o Distributed Ledger Technology (DLT) al que se publicará un resumen estadístico temporal periódico de dichas mediciones y que permitirá realizar una control preciso a tiempo real de las condiciones ambientales requeridas.

El sistema se desarrollará en el lenguaje de programación de Python, pues resulta idóneo para la interacción y la creación de aplicaciones en la Raspberry Pi.

Tras una evaluación y selección de las distintas tecnologías disponibles para ser utilizadas en el proyecto, finalmente se decide por la siguiente combinación:

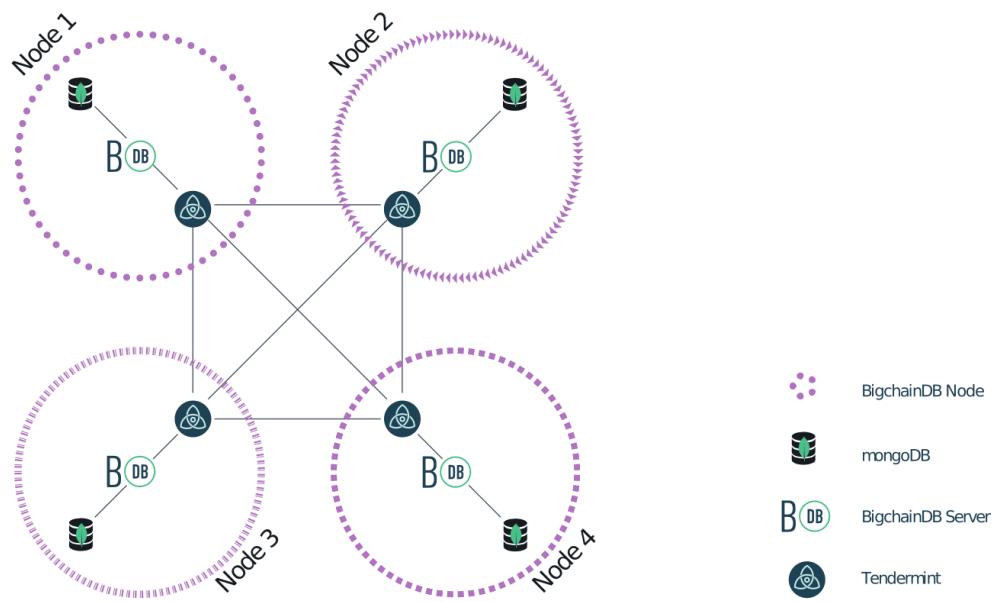
- Un sensor SenseHat, específicamente creado para la Raspberry Pi, utilizado en misiones espaciales y capaz de medir de forma precisa las condiciones ambientales previamente descritas.



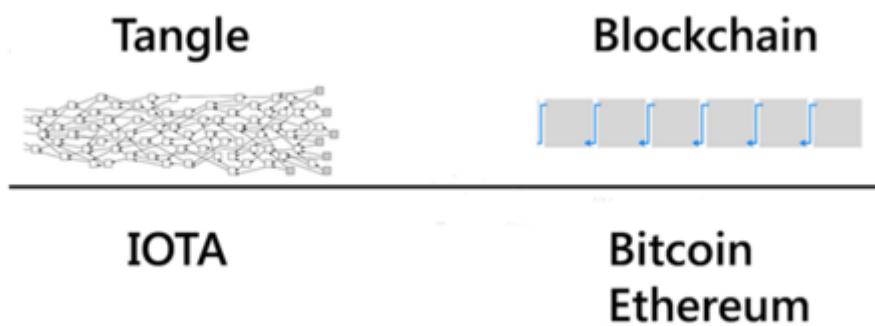
Nº	Identificador
1	Panel Led
2	Joystick
3	Pines de conexión
4	Sensor de humedad
5	Sensor de presión
6	Sensor de aceleraciones, giroscopio y magnetómetro

- El protocolo de comunicación entre máquinas Message Queue Telemetry Transport o MQTT, que tiene una estructura bróker-cliente y servirá como canal de intercambio de mensajes entre la Raspberry Pi y un ordenador con sistema operativo Linux desde el que se tratará la información recogida y se insertará en los distintos servidores. Este resulta un protocolo extremadamente ligero y por tanto ideal para el intercambio de grandes flujos de información entre dispositivos de bajas capacidades.
- La base de datos MariaDB, gratuita y de código abierto, como servidor local en el que registrar las mediciones ambientales recogidas y enviadas por la Raspberry Pi, siendo la primera capa de la solución.

- La base de datos descentralizada y distribuida con características de Blockchain que es BigchainDB, para llevar a cabo las mismas acciones que en MariaDB, pero aprovechando las ventajas que ofrecen las tecnologías de Blockchain de registrar la información y que esta sea inmutable, integra y visible únicamente por aquellos usuarios escogidas por las entidades involucradas en el levantamiento de la red de nodos. Este software se vale de servidores MongoDB que se configuran en cada nodo participante de la red, y del protocolo de comunicación Blockchain Tendermint para realizar las transacciones entre dichos nodos.

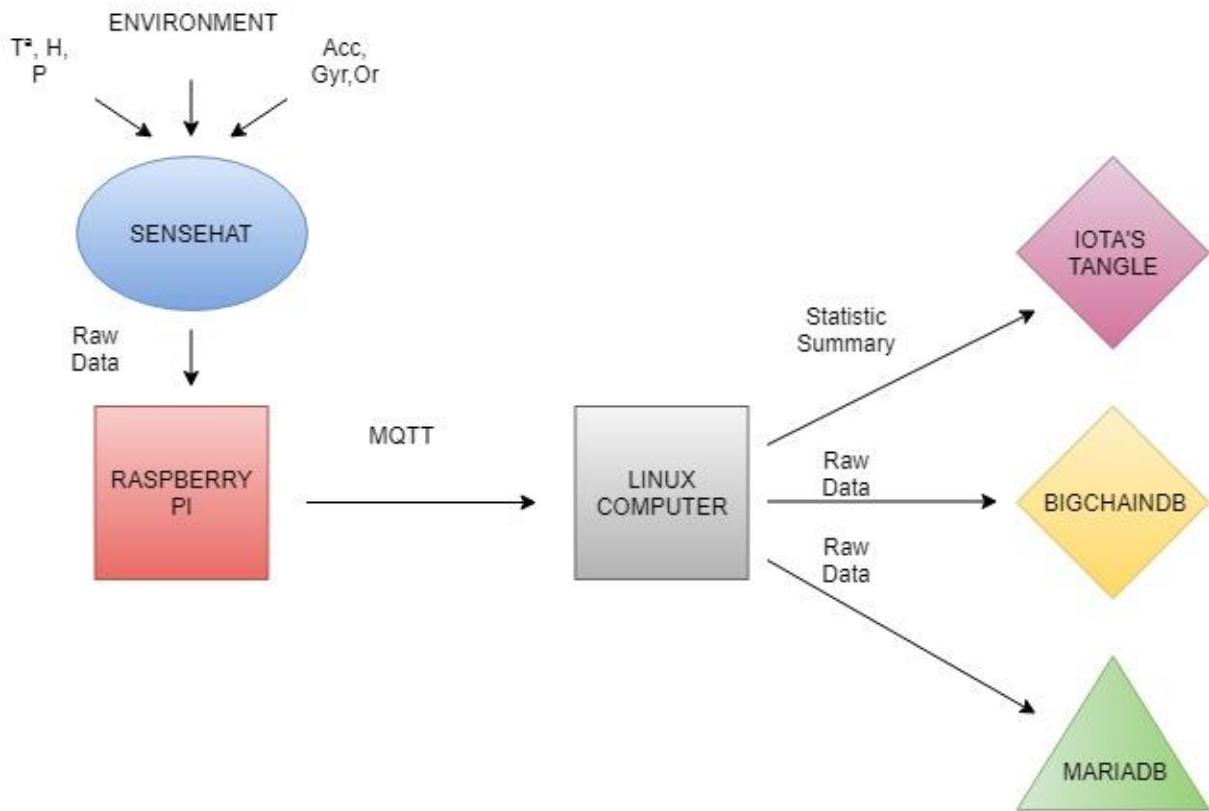


- La DLT de 3^a generación IOTA, que, con su Tangle, revoluciona completamente el paradigma de las bases de datos descentralizadas y distribuidas y sustituye la cadena de bloques por una red de transacciones interconectadas que es gratuita, prácticamente instantánea, escalable (en cuanto a que funciona mejor cuanta más participación haya en el sistema) y que proporciona protección ante ataques realizados por computadores



cuánticos.

Quedando la arquitectura del sistema configurada de la siguiente manera:



Una vez definido el sistema propuesto, se procede a la configuración e instalación de los programas librerías y drivers necesarios para lograr la interacción adecuada en entre los diferentes componentes del sistema.

1) En el lado servidor:

- Instalación de las librerías de Python del sensor SenseHat.
- Instalación de los drivers necesarios para conectarse al servidor Paho MQTT.

2) En el lado cliente:

- Instalación de los drivers necesarios para conectarse al servidor Paho MQTT.
- Instalación de la base de datos MariaDB y configuración de las tablas donde se registrarán los datos ambientales.
- Instalación de los drivers de Python para BigchainDB
- Levantar y configurar nodo de BigchainDB a través de Docker-compose
- Generación de claves criptográficas públicas y privadas de BigchainDB
- Instalación de drivers de Python para IOTA, PyOTA, con extensión en C, que mejora el desempeño de las funciones criptográficas hasta 60 veces.
- Instalar módulo de Python Pandas, para el cómputo del resumen estadístico.

Una vez completados todos los requisitos, se ejecutarán los programas desarrollados:

- Se tomará una muestra de los datos ambientales cada 5 segundos, obteniéndose y transmisiéndose un diccionario de la siguiente forma:

```
root@UDPRBP02:~/Escritorio/HAT/Data# python pubsql.py
Connected with result code 0
Data recorded: {'Data_Collection_Time_Stamp': '2018-06-20 11:30:08', 'Temperature': 38.109092712402344, 'Acceleration_y': 0.6851606369018555, 'Acceleration_x': 0.015274092555046082, 'Gyroscope_x': 0.004145707935094833, 'Yaw': 151.26422437817544, 'Magnetic_Field_y': -9.62348747253418, 'Humidity': 29.86866569519043, 'Pressure': 944.28466796875, 'Magnetic_Field_z': 4.333179473876953, 'Magnetic_Field_x': -18.230669021606445, 'Pitch': 359.193469180509, 'Gyroscope_z': 0.0025741183198988438, 'Gyroscope_y': -0.0034309150651097298, 'Roll': 43.42333745970627, 'Acceleration_z': 0.7215863466262817}
Data recorded: {'Data_Collection_Time_Stamp': '2018-06-20 11:30:13', 'Temperature': 38.21818161010742, 'Acceleration_y': 0.6841894388198853, 'Acceleration_x': 0.01406186334152508, 'Gyroscope_x': 0.002639155834913254, 'Yaw': 151.5494530779152, 'Magnetic_Field_y': -23.936538696289062, 'Humidity': 29.234420776367188, 'Pressure': 944.28271484375, 'Magnetic_Field_z': 9.98161506652832, 'Magnetic_Field_x': -45.10332489013672, 'Pitch': 358.3445476275136, 'Gyroscope_z': 0.0014231132809072733, 'Gyroscope_y': -0.002239307388663292, 'Roll': 44.31297301419501, 'Acceleration_z': 0.7240233421325684}
```

- Dichos datos se reciben en el cliente con el mismo formato:

```
vroman@pcudp05:~/HAT/programas$ python3 mqtt.py
Connected with result code 0
test/data 0 b'{"Data_Collection_Time_Stamp": "2018-06-20 11:31:00", "Temperature": 38.21818161010742, "Acceleration_y": 0.6837038993835449, "Acceleration_x": 0.014304309152066708, "Gyroscope_x": 0.004105187952518463, "Yaw": 151.5492208514834, "Magnetic_Field_y": -27.209548950195312, "Humidity": 31.060279846191406, "Pressure": 944.274658203125, "Magnetic_Field_z": 13.4045991897583, "Magnetic_Field_x": -50.68513488769531, "Pitch": 357.405392561212, "Gyroscope_z": 0.0008658710867166519, "Gyroscope_y": -0.0009315479546785355, "Roll": 43.636521324080704, "Acceleration_z": 0.7225611209869385}'
```

- Una vez recibidos, estos diccionarios de datos se introducen en 3 programas distintos que ejecutan 3 acciones simultaneas:
- Se inyectan en MariaDB:

	+ Options	Data_Storage_Time_Stamp	Data_Collection_Time_Stamp	Temperature	Pressure	Humidity	Pitch	Roll	Yaw	Magnetic_Field_X	Magnetic_Field_Y	Magr
2018-06-20 11:32:30		2018-06-20 11:32:22		38.2545	944.291	27.4022	357.879	44.2795	150.323	-50.4894	-26.9957	
2018-06-20 11:32:30		2018-06-20 11:32:28		38.1455	944.309	31.0411	357.353	44.0495	150.206	-50.3316	-27.5808	
2018-06-20 11:32:33		2018-06-20 11:32:33		38.2182	944.376	30.977	357.771	43.1817	150.052	-50.0781	-27.2729	
2018-06-20 11:32:38		2018-06-20 11:32:38		38.2364	944.285	29.7053	357.894	43.4764	149.901	-49.9835	-26.871	
2018-06-20 11:32:43		2018-06-20 11:32:43		38.2364	944.313	31.5376	358.143	43.9368	149.872	-50.2384	-27.4523	
2018-06-20 11:32:48		2018-06-20 11:32:48		38.2364	944.314	31.6881	358.443	43.6487	149.969	-50.3694	-26.9097	
2018-06-20 11:32:53		2018-06-20 11:32:53		38.2727	944.293	28.046	359.273	43.958	150.301	-50.675	-26.9751	

- Se Inyectan en BigchainDB:

localhost:9984/api/v1/transactions/c9aa71f7f60a2314a7b9eb0787b3578fcffd4acc0df96cd1959f5abb7 90%

JSON Datos sin procesar Cabeceras Guardar Copiar

```

inputs:
  0:
    owners_before:
      0: "9Dw9Pc3P6zA74Dtt9uuQaDnUpHxEsxJoc7gX8Tfqrx8W"
    fulfills: null
    fulfillment: "pGSAIHotr8mun9r-Wxau0q7we_XhTZ_zGUxViEsDpZNga7XguAKKz-emUjgRAm6cPtpVvWP31xY1H6aMlsBha0HJYUS9i2B4MB8YTquhrUe6vtmBT5ffsGPfxdCZU_layV8U1sP"
outputs:
  0:
    public_keys:
      0: "9Dw9Pc3P6zA74Dtt9uuQaDnUpHxEsxJoc7gX8Tfqrx8W"
    condition:
      details:
        type: "ed25519-sha-256"
        public_key: "9Dw9Pc3P6zA74Dtt9uuQaDnUpHxEsxJoc7gX8Tfqrx8W"
        uri: "ni:///sha-256;q003kn0Li16iFUtKI0-6NUN0xmJKKENN4eD0ymP2Fk?fpt=ed25519-sha-256&cost=131072"
        amount: "1"
      operation: "CREATE"
metadata:
  Data_Collection_Time_Stamp: "2018-06-20 11:31:00"
  Temperature: 38.21818161010742
  Acceleration_y: 0.6837038993835449
  Acceleration_x: 0.014304309152066708
  Gyroscope_x: 0.004105187952518463
  Yaw: 151.5492208514834
  Magnetic_Field_y: -27.209548950195312
  Humidity: 31.060279846191406
  Pressure: 944.274658203125
  Magnetic_Field_z: 13.4045991897583
  Magnetic_Field_x: -50.68513488769531
  Pitch: 357.405392561212
  Gyroscope_z: 0.0008658710867166519
  Gyroscope_y: -0.0009315479546785355
  Roll: 43.636521324080704
  Acceleration_z: 0.7225611209869385
asset:
  data:
    raspberry pi:
      serial_number: "RPI01"
      owner: "UPM"
      StorageTimeStamp: "2018-06-20 11:31:00"
version: "2.0"
id: "c9aa71f7f60a2314a7b9eb0787b3578fcffd4acc0df96cd1959f5abb72767efc"

```

6) Se calcula el resumen estadístico temporal, se formatea adecuadamente y se publica al Tangle de IOTA:

	Temperature	Acceleration_y	Acceleration_x	Gyroscope_x	Yaw	\
count	2.0000	2.0000	2.0000	2.0000	2.0000	
mean	37.3969	0.1657	-0.0761	0.0006	144.7547	
std	0.0643	0.0002	0.0005	0.0009	0.0526	
min	37.3455	0.1656	-0.0765	0.0000	144.7175	
50%	37.3969	0.1657	-0.0761	0.0006	144.7547	
max	37.4364	0.1658	-0.0758	0.0012	144.7919	
	Magnetic_Field_y	Humidity	Pressure	Magnetic_Field_z	\	
count	2.0000	2.0000	2.0000	2.0000		
mean	-42.8126	26.9249	942.9504	12.3581		
std	0.0970	0.4711	0.0435	0.2890		
min	-42.8811	26.5917	942.9197	12.1537		
50%	-42.8126	26.9249	942.9504	12.3581		
max	-42.7440	27.2580	942.9812	12.5624		
	Magnetic_Field_x	Pitch	Gyroscope_z	Gyroscope_y	Roll	\
count	2.0000	2.0000	2.0000	2.0000	2.0000	
mean	-60.1859	3.7309	0.0007	-0.0002	7.8209	
std	0.0039	0.9633	0.0004	0.0013	0.2380	
min	-60.1886	3.0497	0.0004	-0.0011	7.6527	
50%	-60.1859	3.7309	0.0007	-0.0002	7.8209	
max	-60.1831	4.4121	0.0010	0.0008	7.9892	
	Acceleration_z					
count	2.0000					
mean	0.9782					
std	0.0007					
min	0.9777					
50%	0.9782					
max	0.9787					

```
Message sent to tangle: {"Num": 6, "Temperature": [35.9, 0.0707, 35.8182, 35.8818, 36.0], "Acceleration_y": [0.4499, 0.0011, 0.4487, 0.4495, 0.4516], "Acceleration_x": [0.0196, 0.0019, 0.0179, 0.019, 0.0233], "Gyroscope_x": [-0.0016, 0.0021, -0.0042, -0.0017, 0.0016], "Yaw": [144.6136, 0.2877, 144.3218, 144.5763, 145.0923], "Magnetic_Field_y": [-38.42, 0.1462, -38.686, -38.4137, -38.2796], "Humidity": [27.1518, 0.2357, 26.7327, 27.1571, 27.4342], "Pressure": [938.0862, 0.0649, 938.0007, 938.0945, 938.145], "Magnetic_Field_z": [15.8246, 0.8381, 14.7016, 15.8784, 16.8803], "Magnetic_Field_x": [-56.2405, 0.2446, -56.5078, -56.2651, -55.9386], "Pitch": [359.3264, 0.43, 358.8901, 359.2653, 359.9761], "Gyroscope_z": [0.0006, 0.0008, -0.0007, 0.0007, 0.0017], "Gyroscope_y": [-0.0009, 0.0019, -0.0042, -0.0007, 0.0013], "Roll": [26.1115, 0.5811, 24.9579, 26.307, 26.5397], "Acceleration_z": [0.8899, 0.026, 0.8856, 0.8905, 0.8924]}
```

Time: 11:31:25 data transferred! in: 13.5 seconds

<https://thetangle.org/transaction/WQFFRFZNYGWNYWCZABHKI099MOCKGXNFVXPUGTSVCAGAAAYTEREGWEXZ9HAFBXURSPIRAFJPAOYZ9999>

Transaction

WQFFRFZNYGWNYWCZABHKI099MOCKGXNFVXPUGTSVCAGAAAYTEREGWEXZ9HAFBXURSPIRAFJPAOYZ9999

June 20, 2018 11:31:38 - 5 minutes and 49 seconds ago

Value	0 i	Pending	
Conversion	0 USD	Index in bundle	0 / 0
Tag	RBRRYPISNRSRSMRYFRSTVSN99999	Weight magnitude	14

Address EABEHEOVIFMSKSUWKZNWDHSUDVAMGSFNEYFODWPIDHTXCGNHKAALKUVIRANAHTQFCHQOUUX9NKVMLAXUFDQJSBCA

Bundle ANJYHVNPYZEEBBULQYDYJLFFXZNGAVXJVXGNYRKFLCUJHLTNBBLEYMTVEBYGBRZZ9RBDZCZIGJVZEL9

Nonce P9MKOGHLEPEEF9TBIKXHQTGUBJG

Message {"Num": 1, "Temperature": [38.2545, NaN, 38.2545, 38.2545, 38.2545], "Acceleration_y": [0.6847, NaN, 0.6847, 0.6847, 0.6847], "Acceleration_x": [0.0141, NaN, 0.0141, 0.0141, 0.0141], "Gyroscope_x": [0.0018, NaN, 0.0018, 0.0018, 0.0018], "Yaw": [151.534, NaN, 151.534, 151.534, 151.534], "Magnetic_Field_y": [-26.88, NaN, -26.88, -26.88, -26.88], "Humidity": [30.032, NaN, 30.032, 30.032, 30.032], "Pressure": [944.2878, NaN, 944.2878, 944.2878, 944.2878], "Magnetic_Field_z": [13.0227, NaN, 13.0227, 13.0227, 13.0227], "Magnetic_Field_x": [-50.4104, NaN, -50.4104, -50.4104, -50.4104], "Pitch": [357.4617, NaN, 357.4617, 357.4617, 357.4617], "Gyroscope_z": [0.0009, NaN, 0.0009, 0.0009, 0.0009], "Gyroscope_y": [-0.0008, NaN, -0.0008, -0.0008, -0.0008], "Roll": [44.6103, NaN, 44.6103, 44.6103, 44.6103], "Acceleration_z": [0.7213, NaN, 0.7213, 0.7213, 0.7213]}

Parent transactions

Trunk	Branch
KWOHLRNXBHDAMHDIJKQ9MB9JAECM9NEWKZWIAOIXRIGWPYK WRJIMIQWIRINZ9AOCRPEM99999	GYGQSQTEHFAKXQHLLTFGUETIQGXKHHWD0F9YNMUTX9YXNSFWFL9 MBGHVFSRTUWSKOOXUPUAVGZ9999

Los resultados obtenidos arrojan las siguientes conclusiones:

- Por la funcionalidad satisfactoria del sistema, se validan las tecnologías utilizadas como las adecuadas para desarrollar la solución propuesta.
- El sensor SenseHat proporciona mediciones de temperatura erróneas por encontrarse demasiado cerca de la placa Raspberry Pi y captar su temperatura como si fuera la ambiental.

- La utilización de estas tecnologías de código libre y gratuitas a pueden formar un sistema altamente cualificado para optimizar procesos productivos, industriales y de distribución y por lo tanto con una inversión muy baja obtener un gran beneficio.
- Hay una carencia significativa de recursos y tutoriales para realizar proyectos con tecnologías Blockchain, lo que limita su adopción por el momento por el público general.
- Existe una gran dispersión en los tiempos de publicación de los datos en el Tangle de IOTA. Esta dispersión se debe a que influyen variables como: el poder de computación del nodo que realiza el PoW necesario y la actividad de la red en el momento de publicar la transacción. Los mejores tiempos obtenidos han sido de 1.8 segundos, y los peores de hasta 43 segundos, llegándose incluso a no publicar algunas transacciones (quedándose pendientes de confirmación).
- Las Snapshots de IOTA (borrados periódicos de transacciones con valor o miotas), hacen que el Tangle no sea un lugar de almacenamiento permanente de los resúmenes estadísticos calculados y limitan estos a la monitorización en tiempo real.
- Resulta muy laborioso estructurar las transacciones en BigchainDB, en especial aquellas del tipo TRANSFER. A pesar de ello, BigchainDB funciona de manera superlativa, tanto en cuanto a tiempos de inserción como con respecto a la incorruptibilidad de los datos.

El proyecto podría tener mayor capacidad de solucionar problemas más amplios y complejos si en el futuro se desarrollan las siguientes mejoras:

- Integración de sistema de seguimiento GPS
- Implementación de funcionalidad MAM en los mensajes de IOTA
- Utilización de placa que sustituya a la Raspberry Pi y posea mayores especificaciones técnicas que ésta. Esto permitirá la automatización completa del sistema y se dejará de depender de un ordenador al que mandar los datos recogidos. Los datos se podrán publicar en IOTA desde la propia placa, ésta podrá actuar de nodo completo capaz de realizar la prueba de trabajo (o PoW) necesaria y se podrán insertar desde ésta los datos en MariaDB y en BigchainDB.
- Utilización de un sensor de temperatura externo y más fiable.
- Implementar Blockchain capaz de establecer y ejecutar contratos inteligentes automáticamente.

Índice General

Capítulo 1 Introducción	1
1.1 Planteamiento del Problema:	1
1.2 Objetivos	3
1.3 Estructura del Documento	4
Capítulo 2 Estado del Arte.....	5
2.1 Contexto.....	5
2.2 Comunicación en Redes IoT:	9
2.2.1 CoAP (Constrained Application Protocol)	10
2.2.2 TinySOA	11
2.2.3 MQTT (Message Queue Telemetry Transport)	12
2.2.4 Conclusiones:	12
2.3 Almacenamiento de Información en Sistemas IoT	13
2.3.1 Almacenamiento Local	13
2.4 Blockchain.....	18
2.4.1 Introducción	18
2.4.2 Orígenes de Blockchain[12]	20
2.4.3 Criptografía y Consenso en la Blockchain[13]	22
2.4.4 Consenso en la Blockchain[12]	26
2.4.5 Blockchain y la Industria 4.0[12].....	31
2.4.6 Distributed Ledger Technologies (<i>Datos</i>) y Blockchain Adecuadas para el Proyecto	34
Capítulo 3 Plataformas Utilizadas	38
3.1 Raspberry Pi 3 B:	38
3.1.1 ¿Qué es Raspberry Pi?.....	38

3.1.2 Modelos y Especificaciones Técnicas:	39
3.1.3 Raspberry Pi 3 Modelo B[20]	40
3.1.4 Puesta en Marcha del Sistema:.....	46
3.2 Sensor Sense HAT:.....	48
3.2.1 ¿Qué es el sensor HAT?:	48
3.2.2 Especificaciones Técnicas:	50
3.3 Protocolo MQTT	51
3.3.1 Introducción	51
3.3.2 El protocolo	51
3.3.3 Brokers	54
3.3.4 Clientes	54
3.3.5 Requisitos	54
3.4 BigchainDB	59
3.4.1 Introducción	59
3.4.2 Diseño de BigchainDB 2.0	59
3.4.3 Campos de Aplicación de BigchainDB.....	62
3.4.4 Funcionamiento de BigchainDB	63
3.4.5 Transacciones en BigchainDB.....	64
3.5 IOTA: La espina dorsal de IoT	74
3.5.1 ¿QUÉ ES IOTA?[29]	74
3.5.2 Consenso.....	75
3.5.3 Diseño y Transacciones en IOTA[30]	75
Capítulo 4 Solución Adoptada	102
4.1 Introducción	102
4.2 Configuración del Sistema	103
4.2.1 Requisitos lado servidor	103

4.2.2 Requisitos lado Cliente	104
4.2.3 Captación de Datos – Lado Servidor	111
4.2.4 Desarrollo del programa	113
4.2.5 Recepción de Datos – Lado Cliente	117
4.2.6 Vida de una Recogida de Datos	126
Capítulo 5 Conclusiones, Líneas Futuras y Difusión	133
5.1 Conclusiones generales	133
5.2 Líneas Futuras.....	135
5.3 Difusión:.....	136
Capítulo 6 Gestión del Proyecto	137
6.1 Introducción	137
6.2 Planificación Temporal.....	137
6.3 Presupuesto.....	140
Capítulo 7 Bibliografía	142

Índice de Figuras

Figura 2.1: Ejemplo de Arquitectura IoT	6
Figura 2.2: IOTA vs Blockchain.....	9
Figura 2.3: Arquitectura CoAP	10
Figura 2.4: Arquitectura de TinySOA.....	11
Figura 2.5: Arquitectura MQTT	12
Figura 2.6: Interfaz de InfluxDB	15
Figura 2.7: Interfaz de usuario de MongoDB.....	16
Figura 2.8: Ejemplo de funcionamiento de función hash	23
Figura 2.9: Modelo Merkle Tree.....	24
Figura 2.10: Ejemplo mensaje enviado mediante criptografía asimétrica	26
Figura 3.1: Logo de la Marca Raspberry.....	38
Figura 3.2: Raspberry Pi.....	40
Figura 3.3: Puertos Raspberry Pi 3 modelo B	41
Figura 3.4: Pines GPIO Raspberry Pi	42
Figura 3.5: Logo Ubuntu Mate	43
Figura 3.6: Diferencias sistemas kernel monolíticos y microkernel	45
Figura 3.7: Descarga de Ubuntu Mate.....	47
Figura 3.8: Instalación Ubuntu Mate.....	48
Figura 3.9: Sensor SenseHat	49
Figura 3.10: Arquitectura de MQTT cortesía de Eurotech	51
Figura 3.11: Flujo de intercambio de paquetes entre bróker y cliente en MQTT	53
Figura 3.12: Códigos de respuesta de conexión en MQTT	53
Figura 3.13: Flujo de datos de desuscripción en MQTT	53

Figura 3.14: Autenticación y autorización en MQTT	56
Figura 3.15: Quality of Service en MQTT	57
Figura 3.16: Logo BigchainDB.....	59
Figura 3.17: Red de 4 nodos de BigchainDB	64
Figura 3.18: Transacción en BigchainDB	65
Figura 3.19: Logo IOTA	74
Figura 3.20: Proceso para publicar una transacción en el Tangle de IOTA.....	81
Figura 3.21: Simulación red Tangle de IOTA.....	91
Figura 3.22: simulación Tangle con 17 nodos y ratio de transacción $\lambda = 1.5$	92
Figura 3.23: Simulación red Tangle con 120 nodos y ratio de transacción $\lambda = 7.1$	92
Figura 3.24: Simulación red Tangle con $\lambda = 0.1$	93
Figura 3.25: Simulación red Tangle con λ muy elevado	93
Figura 3.26: Simulación red IOTA con algoritmo Unweighted Random Walk	94
Figura 3.27: Representación de la estrategia Weighted Random Walk (1).....	95
Figura 3.28: Representación de la estrategia Weighted Random Walk.....	96
Figura 3.29: Representación de la estrategia Super-Weighted Random Walk.....	96
Figura 3.30: Weighted Random Walk, cada transacción azul muestra su peso acumulado y la probabilidad de ser escogido como el siguiente paso en el recorrido	97
Figura 3.31: Preferencia de los tips con mayor peso acumulado en el Tangle	100
Figura 4.1: Arquitectura general de la solución propuesta	103
Figura 4.2: Tabla de inserción de datos ambientales en MariaDB.....	107
Figura 4.3: Prueba de operatividad de nodo BigchainDB	109
Figura 4.4: Imágenes de BigchainDB, Tendermint y MongoDB levantadas.....	109
Figura 4.5: SenseHat y tornillos	111
Figura 4.6: Visualización en la terminal de la Raspberry Pi de los datos recogidos .	126
Figura 4.7: Datos recibidos en el cliente Linux.....	127

Figura 4.8: Visualización a través de HTTP del nodo levantado en el cliente Linux	127
Figura 4.9: Activo creado e insertado en BigchainDB, cuyos metadatos constituyen los datos ambientales recogidos y transmitidos al cliente Linux	128
Figura 4.10: Datos.....	129
Figura 4.11: Datos registrados en MariaDB	129
Figura 4.12: Datos registrados en MariaDB	130
Figura 4.13: Resumen estadístico periódico de los datos recolectados	131
Figura 4.14:Mensaje publicado al Tangle de IOTA	131
Figura 4.15: Tiempo que se ha tardado en realizar el PoW necesario para publicar el mensaje en el Tangle de IOTA.....	131
Figura 4.16: mensaje publicado al Tangle de IOTA, visualizado en texto	132
Figura 5.1: Mínimo tiempo obtenido en la publicación de un mensaje en el Tangle ..	134
Figura 6.1: Diagrama EDP de las fases del proyecto.....	138
Figura 6.2: Diagrama de Gantt del proyecto.....	139

Índice de Tablas

Tabla 3.1: Comparación especificaciones técnicas distintos modelos de Raspberry ..	39
Tabla 3.2: Elementos Raspberry	40
Tabla 3.3: Requisitos Ubuntu Mate	44
Tabla 3.4: Elementos SenseHat.....	49
Tabla 3.5: Especificaciones técnicas SenseHat	50
Tabla 3.6: Paquetes MQTT [6]	52
Tabla 3.7: Tabla comparativa servidores MQTT [6]	54
Tabla 3.8: Comparativa BigchainDB con bases de datos y otras Blockchain[27]	60
Tabla 3.9: Unidades típicas IOTA	79
Tabla 3.10: Campos transacción IOTA.....	81
Tabla 6.1: Tabla de desglose de las fechas de inicio y final de las fases del proyecto	140
Tabla 6.2: Desglose de Coste Material del proyecto	140
Tabla 6.3: Honorarios por la realización del proyecto.....	141
Tabla 6.4: Presupuesto de Ejecución por Contrata.....	141

Capítulo 1 Introducción

En este capítulo se pretende dar una visión global de las tecnologías de *Internet de las Cosas*, *Blockchain* y como la sinergia de ambas está llamada a ser fundamental en el futuro ecosistema de la industria, los procesos productivos y las cadenas de suministro. Se establecerá el marco en el que se desarrollará su implementación para la solución de un problema actual y, además, se plantearán los objetivos del presente proyecto, así como la estructura que seguirán el mismo y el documento asociado a él.

1.1 Planteamiento del Problema:

En la actualidad, nos encontramos inmersos en la cuarta revolución industrial, en la que la automatización y digitalización de los procesos productivos, las tecnologías de la información y la conexión permanente a internet se han convertido en tecnologías ubicuas en nuestro entorno y nuestras vidas.

En este contexto, surgen nuevas necesidades y problemas, que constituyen a su vez oportunidades para continuar la mejora y la optimización de las operaciones.

Consideremos el caso de una empresa de distribución de mercancía delicada, como pueden ser alimentos. Si dicha empresa tiene como tarea el almacenamiento, transporte y entrega de sus productos, resulta extremadamente útil conocer el estado de dicha mercancía a lo largo de toda la cadena. Tanto para la empresa distribuidora, para su propia gestión interna de los distintos procesos y operaciones que lleva a cabo, como para la empresa receptora del producto y para el consumidor final. Todas las partes involucradas estarán profundamente interesadas en que la mercancía llegue en estado óptimo, y es en este contexto en el que la aplicación de las tecnologías punteras del Internet of Things (IoT) y Blockchain cobran especial relevancia. Su implementación puede tener un efecto sinérgico y lograr un incremento significativo en la transparencia y el flujo de datos en una gran parte de las operaciones a las que dicha mercancía se ha sometido, logrando así una palpable mejora en la optimización y control.

En concreto, en el presente proyecto, se propone un sistema capaz de llevar a cabo una medición en tiempo real de las condiciones ambientales a las que se somete dicha mercancía, es decir: temperatura, presión, humedad, campos magnéticos, aceleraciones y orientación, que pueden dar información de su estado en todo momento y permitir la actuación adecuada en caso de que alguna de las condiciones ambientales se salga de unos límites previamente considerados y establecidos.

Para ello, se utilizarán unos sensores conectados a un ordenador de placa única (Single Board Computer o SBC) que no solo sea capaz de captar y procesar dichos datos, sino que además pueda transmitirlos en tiempo real, vía internet, para que puedan ser almacenados y analizados, de manera que se puedan tomar las decisiones más acertadas en cada momento. Si, por ejemplo, se está transportando una carga de carne congelada y se necesita que se mantenga por debajo de cierta temperatura para completar un pedido a un cliente y, la temperatura aumenta más de lo establecido por las partes interesadas, la empresa distribuidora podría programar en tiempo real la partida de otro transporte capaz de suplir la mercancía corrupta y por tanto salvar una oportunidad de negocio.

Por otro lado, la empresa receptora de dicha mercancía también tendría un gran interés en conocer el estado de la carne, pues si se sale de los parámetros preestablecidos podría conocerlo de antemano y no aceptar dicho pedido. Para conseguir este objetivo, entra en juego la tecnología Blockchain. La Blockchain es una base de datos descentralizada y distribuida cuya principal aplicación y por lo que resulta una tecnología de grandísimo interés en una enorme cantidad de áreas e industrias diferentes, es porque permite las transferencias de información, contratos y valor entre distintos actores que no tienen por qué confiar unos en otros. En este sistema se elimina la necesidad del arbitraje de organismos reguladores centralizados, pues es la propia red la que asegura la inmutabilidad de los datos escritos en dicha base de datos (también denominada comúnmente ledger o libro de cuentas). Esto lo consigue debido a que la información registrada en el sistema se vuelve completamente inmutable, lo que confiere una garantía total a la historia de los datos escritos.

De esta forma, los datos recogidos por los sensores y enviados a las distintas bases de datos podrán ser inmediatamente insertados en esta Blockchain, quedando la información perfectamente visible tanto para la empresa distribuidora, como para la receptora de la mercancía y para el cliente final. Así, todas las partes involucradas podrán tener la confianza y seguridad de que la carga se encuentra en el estado acordado por dichas partes y solucionaría una gran cantidad de problemas de condiciones incumplidas en este tipo de transacciones.

Así el sistema propuesto constituiría una solución eficaz y eficiente para resolver y provenir un problema relevante en el marco actual de la industria. Además, dada la naturaleza de los componentes utilizados, el sistema desarrollado supondría un bajo coste y consumo, no necesita de terceras partes controladoras, es descentralizado y garantiza un flujo de información íntegro e inmutable. Todo ello podría contribuir significativamente al aumento de la creación de valor y es precisamente su posible aplicación el objetivo principal del proyecto presente.

1.2 Objetivos

En este trabajo se propone la integración de un sistema IoT con una serie de tecnologías Blockchain, con el fin de obtener un sistema lo más descentralizado, integro, transparente y económicamente ventajoso posible.

Para ello, se contará con un sensor, conectado a una Raspberry Pi, que recogerá datos ambientales, una base de datos en un server local, como primera capa de la solución, y la implementación de tecnologías Blockchain para dotar al sistema de un servidor descentralizado en el que registrar la información recogida y para publicar un resumen estadístico temporal periódico de dicha información de forma que se pueda obtener una monitorización tanto a tiempo real como periódica del estado de la carga de interés. De forma que se conseguirá un sistema flexible, dinámico y potente que permita respaldar a una gran cantidad de servicios y aplicaciones, orientadas sobre todo a la mejora de la eficacia y productividad de los procesos productivos, transporte de mercancías, de mantenimiento predictivo y cadenas de suministro en el sector industrial.

Todo esto se conseguirá desarrollando los correspondientes programas, codificados en el lenguaje de programación Python. Quedando el desglose de los objetivos individuales tal y como sigue:

- **Objetivo 1** – Recopilar los datos ambientales de: presión, temperatura, humedad, aceleración, campos magnéticos y orientación; a partir del sensor conectado a una Raspberry Pi.
- **Objetivo 2** – Establecer un canal de comunicación bróker-cliente, *Machine to Machine* (M2M), entre la Raspberry Pi y otra máquina Linux, de manera que los datos recopilados por el sensor se publiquen, a través del protocolo de comunicación entre dispositivos, y, a través de una suscripción a dicho canal, puedan ser tratados convenientemente por la máquina Linux.
- **Objetivo 3** – Recibir los datos publicados a través del mencionado protocolo y registrarlos con su correspondiente marca temporal en la base de datos situada en un servidor local en la máquina Linux.
- **Objetivo 4** - Calcular, valiéndose del módulo de Python Pandas, un resumen estadístico temporal que incluya: Mínimo, Media, Mediana, Máximo y Desviación Típica; y publicarlo a través de una tecnología DLT.
- **Objetivo 5** – Aportar una última capa de solución al sistema, en el que se registre la información recibida en el objetivo 3, paralelamente en un servidor descentralizado y distribuido, valiéndose de una tecnología Blockchain adecuada.

1.3 Estructura del Documento

En este apartado se hará una explicación del contenido de los siguientes capítulos, teniendo en cuenta que el actual (Capítulo 1) es la introducción:

Capítulo 2: Estado del Arte

En este capítulo se hará una se pondrá en contexto la situación actual de la tecnología relacionada con la industria 4.0, se hará una breve introducción a dichas tecnologías. También se hará una descripción de su historia y su actual grado de desarrollo, así como de los principales problemas y oportunidades que presentan, y como su aplicación sinérgica resulta idónea para solucionar el problema planteado. Por último, se realizará una evaluación de las distintas herramientas disponibles en cada tecnología, y se seleccionarán aquellas que mejor se adapten a las necesidades del proyecto en términos de eficiencia y eficacia, de facilidad de uso e implementación y de coste.

Capítulo 3: Plataformas Utilizadas

En este capítulo, se realizará una descripción en detalle de las tecnologías utilizadas en el proyecto explicando su funcionamiento, aplicaciones y su papel específico en el proyecto.

Capítulo 4: Solución Adoptada

En este capítulo, se describirá en detalle la solución adoptada y la arquitectura del sistema propuesto. Se realizarán las pruebas pertinentes para demostrar su correcto funcionamiento, y su validez como solución al problema planteado.

Capítulo 5: Conclusiones, Líneas Futuras y Difusión

En este capítulo, se comentarán los resultados obtenidos y se destilarán conclusiones de ellos. Se describirán posibles vías futuras de desarrollo del sistema, así como soluciones integradas con otras tecnologías para acometer una variedad aún más amplia de problemas. Además, se hablará de la difusión del proyecto en diferentes medios.

Capítulo 6: Gestión del Proyecto

En este capítulo, se adjuntarán los diagramas EDP y Gantt empleados en la organización del proyecto, así como el presupuesto del mismo.

Capítulo 7: Bibliografía

Capítulo 2 Estado del Arte

En este capítulo se llevará a cabo una introducción de las últimas tecnologías y soluciones disponibles en los terrenos de: comunicación en redes de IoT, almacenamiento de la información en IoT, Blockchain y aplicaciones de Blockchain con IoT. Además, en cada categoría se escogerá una o varias tecnologías específicas para desarrollar el proyecto.

2.1 Contexto

En 1969, se dio la primera conexión entre ordenadores, conocida como *ARPANET*, y la humanidad fue testigo de una verdadera revolución: había nacido *internet*. Nuestra relación con la información y su tratamiento cambiaron para siempre. Llegó una nueva forma de transmisión y almacenamiento de datos, haciéndose posible la comunicación sin fronteras espaciotemporales entre diversas partes.

Esta revolución dio paso a un desarrollo y crecimiento de tecnologías sin precedentes, impulsando de manera notoria la creación de soluciones novedosas a problemas relacionados con el trato y seguridad de la información. El desarrollo de la totalidad de estas tecnologías se sale del espectro del presente proyecto, y hay multitud de recursos en las redes y bibliografía específica sobre el tema. Por lo tanto, dando un salto en el tiempo, se llega hasta una de las tecnologías más relevantes y que sin duda impulsará el nuevo paradigma de la Industria 4.0, el nacimiento del *internet de las cosas* o *Internet of Things* (IoT).

El término IoT, fue acuñado en el MIT en 1999 por Kevin Ashton[1], y se entiende como la red de dispositivos, vehículos y aplicaciones del hogar integrados con electrónica, software, sensores y actuadores que están interconectados con el objetivo de recolectar, almacenar y compartir información, con la posibilidad de realizar ciertas acciones con respecto a ella[2].

Actualmente, el concepto va mucho más allá de la comunicación *Maquina a Maquina* o *Machine to Machine* (M2M) y describe un sistema de conexión avanzada de dispositivos, sistemas y servicios que cumple una amplia variedad de protocolos, dominios y aplicaciones.

Dada la reciente implantación de la tecnología, existen multitud de arquitecturas propuestas para el IoT, pero en general se puede dividir en 3 capas[3]:

1^{a)}) Capa de percepción o sensores: en la que se recopila información de dispositivos, etiquetas de identificación de radiofrecuencia o RFID, cámaras, GPS... En esta capa los sensores inteligentes inalámbricos pueden captar e intercambiar automáticamente datos a través de diferentes dispositivos y controlarlos remotamente.

2^{a)}) Capa de red: en la que principalmente se transmite y procesa información. Su rol es el de conectar todos los objetos entre sí y permitir el intercambio de datos.

3^{a)}) Capa de aplicación: En la que se desarrollan aplicaciones para el IoT.

Un esquema de arquitectura típica de IoT se refleja en la siguiente figura:



Figura 2.1: Ejemplo de Arquitectura IoT

La información recopilada, tradicionalmente se comparte en la nube llevando a cabo diferentes funciones: análisis de datos, ejecuciones en regla, gestión de consultas, informes...

El IoT es extremadamente útil, sobre todo en las siguientes situaciones:

- Aplicaciones que requieren multi-organización de datos compartidos.
- Alta variedad y distribución de sensores/datos recogidos/fuentes de datos emergentes.
- Aplicaciones basadas en la nube.
- Software para gestionar multitud de objetos (casa, oficinas, talleres...)
- Aplicaciones que involucran personalización/customización de productos/ordenes de producción/dispositivos.

El IERC (*European Research Cluster on the Internet of Things*) considera las siguientes áreas como las principales de aplicación del IoT: *Smart Energy, Smart*

Health, Smart Cities, Smart Farming y Smart Industry[4]. En el contexto del sector industrial surge el *Industrial Internet of Things* o IIoT. En concreto, el término se refiere a la aplicación del IoT en industrias como la manufacturera, logística, cadenas de suministro, mantenimiento predictivo, gas, petróleo, transporte, energía, química y de aviación entre otras.

La aplicación e integración de las tecnologías IoT e IIoT, permitirán la automatización y eliminación de gran cantidad de agentes intermediarios en los procesos productivos. Esto generará, según organizaciones como IDC, Boston Consulting Group y Bain [5] un valor de unos 300 Billones de dólares americanos en el año 2020 para el IoT, y se estima que el sector de las IIoT, tendrá un valor aproximados de 85 Billones para el mismo año.

Pero no son todo bondades, actualmente estas tecnologías se enfrentan a retos de gran magnitud. En particular, actualmente existen ciertas barreras que limitan el crecimiento y la implantación de sistemas IoT:

Uno de ellos es el direccionamiento, debido a la enorme cantidad de dispositivos conectados a internet que configuran las redes IoT, el espacio de direcciones Ipv4 (y sus posibles 4,294,967,296 direcciones) no van a ser capaces de soportar el tráfico que se alcanzará[1]. Por este motivo, se está realizando una migración al espacio Ipv6 (que provee hasta 340,282,366,920,938,463,463,374,607,431,768,211,456 direcciones) que, se espera, sea capaz de abastecer a la totalidad de la futura red de redes.

Otro gran problema es la ausencia de un estándar común de interoperabilidad entre dispositivos y aplicaciones y de una arquitectura única, según la cual se implementen y funciones dichos sistemas[6].

La integración de la información también supone un área de preocupación para el desarrollo de estas tecnologías ya que, la tarea de transformar los datos recogidos en valor de negocio no es para nada trivial. Además, hay una carencia significativa de trabajadores capaces de manejar dichas tecnologías y ello resulta un gran impedimento para la adopción por parte de las compañías.

Pero quizás, el mayor reto al que se enfrenta el IIoT sea el problema de la ciberseguridad y la protección de datos. Éste, es un tema de máxima prioridad en la industria, dada la creciente vulnerabilidad a los ataques y a las brechas de seguridad. La información que se recopila y gestiona a través del IIoT se considera de muy alta sensibilidad, debido a la importancia de los datos sobre productos, estrategias de negocio e información interna de organizaciones y, cuyo robo o manipulación puede repercutir en graves pérdidas económicas y reputacionales.

Como ejemplo, el viernes 21 de octubre de 2016 se produjo la cadena de ciberataques más graves de la última década[7]. Estos ataques se dirigieron hacia la empresa Dyn, una de las mayores proveedoras de internet, que controla los servicios de páginas tan importantes como Amazon, Twitter, Spotify, Reddit, Airbnb, PayPal, Netflix... y medios de comunicación como *The New York Times*, *Financial Times*; *The*

Guardian... Estos ataques fueron de naturaleza DDoS o *Denegación de Servicios*, afectaron a más de 1.000 millones de usuarios. Los objetivos del ataque fueron precisamente dispositivos IoT y se realizó a través del botnet Mirai.

El informe de amenazas trimestrales de FortiGard Labs de finales del 2017[8], reveló que el número de ciberataques está incrementando a una velocidad sin precedentes. En el último trimestre de 2017 se detectaron una media de 274 ataques por compañía, un 82% más que en el anterior trimestre. Estando dirigidos la mayoría de dichos ciberataques a redes de IoT.

Según John Chambers, CEO de Cisco, “Las empresas se dividen en dos categorías: las que han sido hackeadas y las que no lo saben”. Por otro lado, IBM estima que las organizaciones reciben 1.400 ciberataques en promedio a la semana y a nivel mundial, el cibercrimen genera 440.000 millones de dólares al año[9].

Ante este panorama, surgen tecnologías que aseguran la integridad y la seguridad de los datos que se generan e intercambian a través de internet, siendo una de las más útiles y funcionales la denominada tecnología Blockchain (cadena de bloques). La Blockchain es una base de datos distribuida y descentralizada, que se encuentra dividida entre todos los nodos que conforman la red. Esta característica otorga a la red una gran capacidad de protección ante ciberataques, pues, para resultar exitoso, un ciberataque debería romper toda la seguridad del sistema. Cuantos más nodos forman parte de la red, más complejo se vuelve poder acceder al sistema[9].

Además, el sistema se auto-revisa y auto-repara. Los ordenadores que participan como nodos, chequean lotes de transacciones (o bloques) y, una vez que se validan, se registran en la base de datos, donde toda la red tiene acceso a él. La Blockchain es pública en cuanto a que todos los participantes pueden acceder a las transacciones realizadas, pero las personas que intervienen en ella son anónimas a ojos del usuario. Este concepto se explicará con más detalle más adelante.

Los datos registrados en las cadenas de bloques se revisan constantemente y, esto les confiere un carácter inmutable pues, cualquier cambio realizado es inmediata y permanentemente visible y, si no es congruente con la historia registrada por el resto de los participantes, no será aceptado. Esto, infiere integridad y asegura la confianza en la validez de la información registrada. Y por ello resulta tremadamente interesante para gestionar los datos generado por el IoT.

El problema, es que tecnologías Blockchain “tradicionales” como Bitcoin, requieren de una prueba de trabajo o *Proof of Work* (PoW) para llevar a cabo el proceso de verificación y registro de la información de las transacciones. Y, dada la naturaleza del sistema, actualmente este PoW se traduce en un enorme gasto en poder de computación que conlleva grandes consumos de tiempo, energía y dinero. Por ejemplo: en Finlandia se dedica actualmente más energía eléctrica a minar Bitcoin que en consumo de hogares[10].

Además, a pesar de que originalmente la idea era que se trataran de bases de datos descentralizadas, actualmente tan solo pueden llevar a cabo el PoW entidades con acceso a enormes capacidades computación, como en el caso de China, donde existen bloques de apartamentos repletos de clúster de ordenadores enteramente dedicados al minado de Bitcoin. Por tanto, la capacidad de minado de bloques queda centralizada en unos pocos organismos, lo que rompe con la filosofía original de la tecnología.

Ante esta situación, surgen bases de datos descentralizadas o *Distributed Ledger Technologies* (DLT), llamadas de 3^a generación, con IOTA como buque insignia. Esta tecnología revoluciona completamente el concepto de Blockchain ya que no se trata de una cadena de bloques, sino que está basada en una red interconectada de nodos o *Tangle*, en la cual cada transacción, para poder quedar registrada en la DLT, ha de validar previamente otras dos transacciones previamente realizadas. Esto confiere al sistema un carácter totalmente descentralizado y permite acabar con el minado o PoW, lo que se traduce en transacciones casi instantáneas y gratuitas, y hace que la red funcione mejor cuanto mayor sea el número de nodos que formen parte del sistema. No solo eso, sino que incorpora una tecnología Curl-p que aporta protección ante ciberataques provenientes de computadores cuánticos.

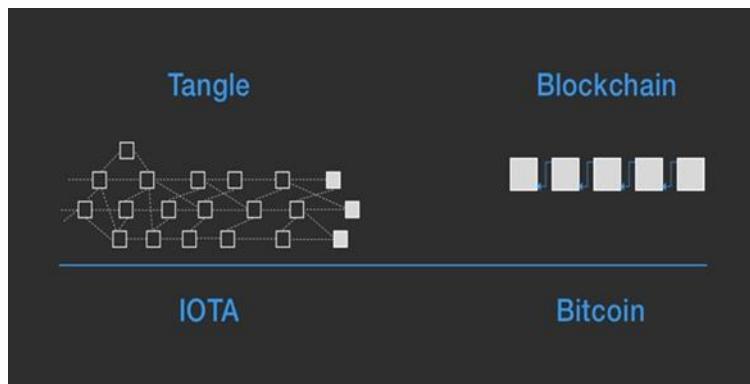


Figura 2.2: IOTA vs Blockchain

2.2 Comunicación en Redes IoT:

El conjunto de tecnologías que constituyen el IoT se encuentra en un estado de desarrollo muy temprano y, una muestra de ello es la ausencia de un protocolo de comunicación estándar según el cual interoperen los dispositivos y aplicaciones que forman el sistema. Además, es crucial que dichos protocolos sean capaces de operar a niveles de red diferentes, generando e intercambiando gran cantidad de eventos[6].

Para ello, se está llevando a cabo el desarrollo de middleware, que sea capaz de soportar las operaciones anteriormente mencionadas. Las líneas de investigación de dicho middleware pasan por las redes de sensores inalámbricas (WSN), por las comunicaciones maquina a máquina (M2M) y por las etiquetas de identificación de

radio frecuencia (RFID)[6]. En este punto se hará una pequeña introducción a los middlewares más utilizados y se elegirá uno para realizar el proyecto. Las principales características que se buscan en este tipo de protocolos son: ancho de banda reducido, bajo consumo energético y mecanismos de seguridad efectivos.

2.2.1 CoAP (Constrained Application Protocol)

Es un protocolo ampliamente utilizado en las redes de sensores (WSN) y comunicación M2M[6]. Está diseñado como una traducción sencilla de HTTP, implementando el modelo REST: Los servidores hacen disponibles los recursos bajo un URL y los clientes acceden a dichos recursos a través de métodos como GET, PUT, POST y DELETE. Dado que HTTP y CoAP comparten el modelo REST, pueden conectarse fácilmente utilizando proxys de protocolo cruzado independientes de la aplicación. Se utiliza para simplificar la integración dispositivo-red con requisitos específicos como baja carga, consumo y simplicidad[6].

Sus características principales son las siguientes:

- Es una traducción simplificada de HTTP, ampliamente compatible con ésta.
- Permite el intercambio asíncrono de información.
- Baja sobrecarga de cabeceras para reducir la complejidad al analizar el mensaje.
- Incorpora un protocolo de seguridad DTLS equivalente a 3072-bit llaves de RSA, pero que se ejecuta con facilidad aún en los nodos con baja capacidad de procesamiento.
- Soporta IPv6.

Un esquema de su arquitectura:

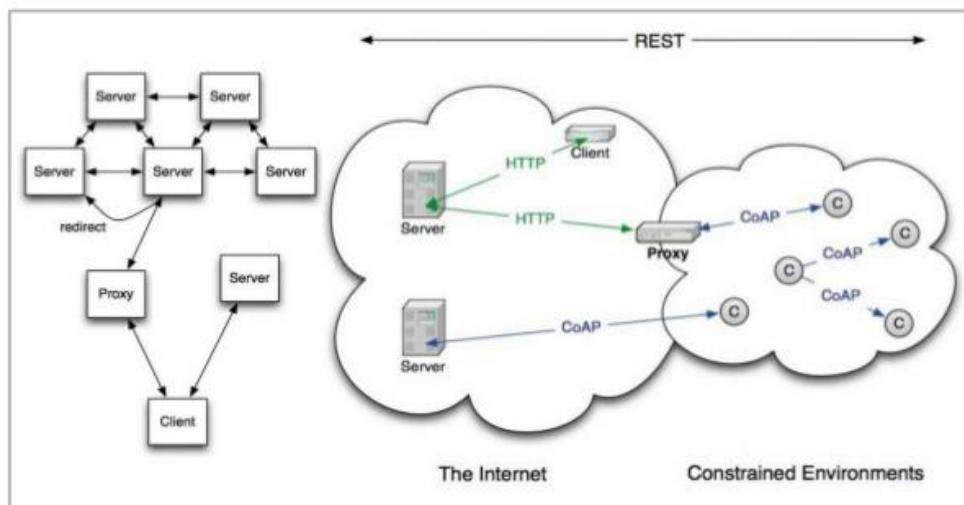


Figura 2.3: Arquitectura CoAP

2.2.2 TinySOA

Es un middleware utilizado en redes de sensores (WSN). Proporciona un conjunto de servicios web que facilita el envío y recepción de datos por parte de los sensores. Está formado por cuatro partes principales: Nodos, Gateway, registro y servidor[6].

El nodo reúne toda la información de un nodo sensor. Tiene algunos servicios como el descubrimiento de recursos o la comunicación con el gateway. El gateway en las redes WSN, tiene la función de conectar la red WSN con la red externa, es decir con Internet. Entre otras funcionalidades, usa la información del nodo sensor para registrarlos en el servicio de registro. Una vez registrados, el gateway se suscribe a los servicios proporcionados por el sensor. El servidor es otra de las partes principales de este middleware, y su función es usar el servicio de registro para establecer un servicio web[6]

Arquitectura de TinySOA:

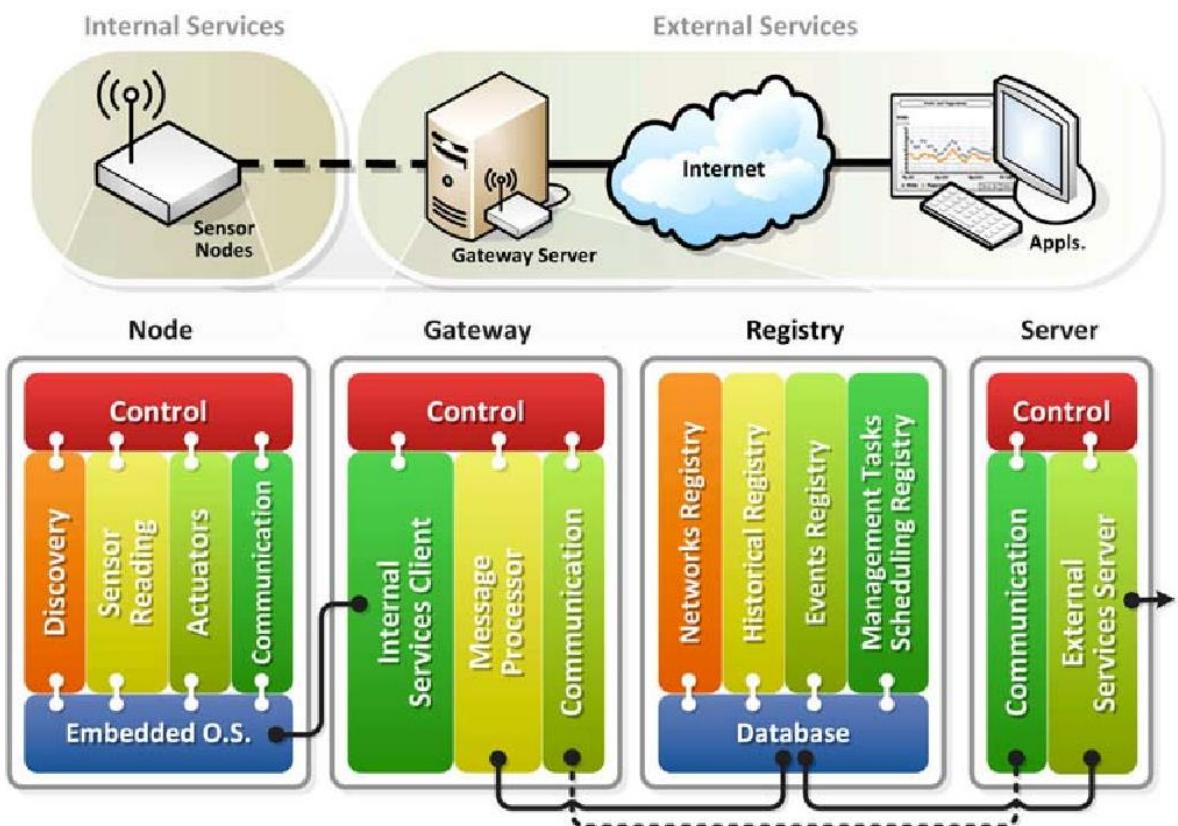


Figura 2.4: Arquitectura de TinySOA

2.2.3 MQTT (Message Queue Telemetry Transport)

Es un protocolo de código abierto desarrollado y optimizado para la comunicación máquina a máquina (M2M), en particular, para dispositivos de bajo consumo, redes de bajo ancho de banda, alta latencia y poco confiables[11]. Es un protocolo, extremadamente ligero, de transporte de mensajería basado en el esquema publicación/subscripción cuyas características principales son las siguientes:

- Puede ser utilizado por la mayoría de los dispositivos empotrados con pocos recursos
- La arquitectura sigue una topología de estrella, con un nodo central que hace de servidor o bróker y tiene capacidad de hasta 10000 clientes.
- Permite cifrar la información, garantizando la seguridad en la transmisión de esta.
- Tiene reconocimiento de sesión continua, apoyándose en TCP.
- Permite elegir entre varias opciones de calidad de servicio, a mayor calidad, más pesada será la comunicación, pero más confiable.

Arquitectura de protocolo MQTT:

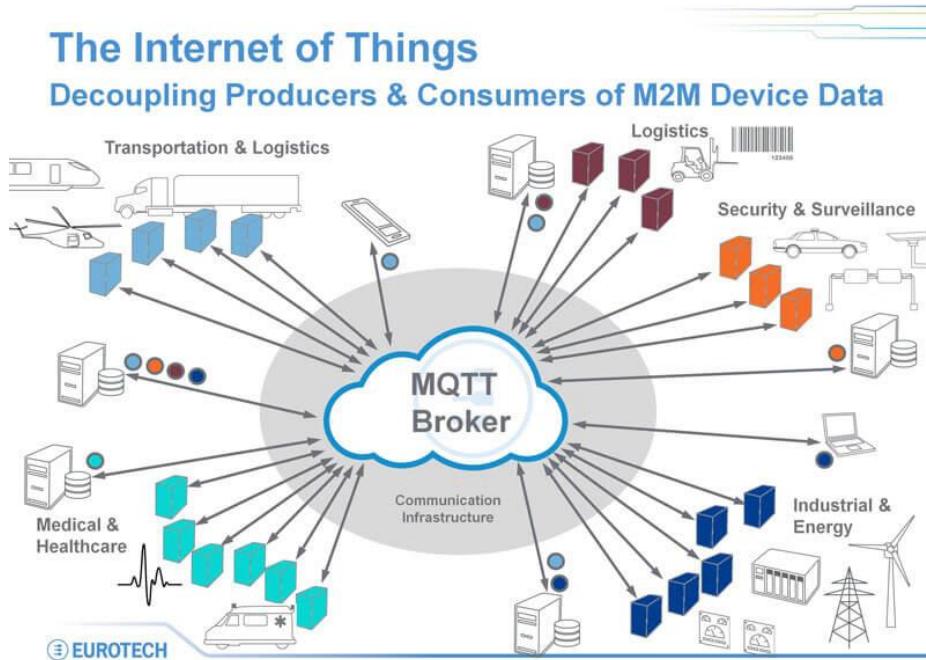


Figura 2.5: Arquitectura MQTT

2.2.4 Conclusiones:

Tras un análisis de los diferentes protocolos de comunicación disponibles para redes de sensores y dispositivos IoT, se ha escogido MQTT como el servicio de mensajería a utilizar, entre la Raspberry desde la que se captan los datos ambientales, y la maquina

Linux que actúa como cliente, desde la que se tratará la información recopilada. Las principales causas que han motivado esta decisión son:

- Se trata de un protocolo de código abierto que presenta un gran apoyo por parte de la comunidad de desarrolladores de aplicaciones IoT, por lo que: hay una substancial cantidad de recursos disponibles para entender y aplicar la tecnología y que el protocolo esté continuamente actualizándose y adaptándose a las necesidades de la industria.
- Es un protocolo extremadamente ligero que resulta idóneo para la transmisión del tipo de información que se trata.
- Permite establecer distintos grados de seguridad y calidad de servicio en función de las necesidades del desarrollador.
- Es relativamente sencillo de utilizar y programar.

2.3 Almacenamiento de Información en Sistemas IoT

Por la arquitectura y la funcionalidad inherente al IoT, se genera una cantidad enorme de información que es preciso almacenar para su posterior tratamiento y análisis. Ante esta situación, se consideran diferentes métodos de almacenamiento y se detallan sus ventajas e inconvenientes, realizándose finalmente una selección del más adecuado para el correcto desarrollo del proyecto.

2.3.1 Almacenamiento Local

Por las características de bajos recursos de los dispositivos que recogen los datos en las redes IoT, y el gran volumen de datos generados, generalmente, el guardado de dichos datos en los propios dispositivos resulta mala idea. Además, dichos dispositivos se suelen ver sometidos a ciertos fenómenos ambientales que pueden poner en peligro el hardware y por tanto pueden poner en compromiso dicha información.

Por tanto, el almacenamiento local que se considera es en el cliente que recibe los datos generados por los sensores, en el caso del presente proyecto sería la maquina Linux de la que se dispone. Y ante esta situación se tienen en cuenta las siguientes alternativas.

2.3.1.1 Archivos CSV

Los archivos CSV, o Comma-Separated Values, son un tipo especial de documentos en formato sencillo para representar datos en forma de tabla. En dichas tablas, las columnas se separan por comas y las filas por saltos de línea. Resultan de especial interés para almacenar de forma compacta gran cantidad de información, y que

posteriormente, es fácil de tratar con programas como Excel, que disponen de métodos de reconocimiento y traducción automáticos a formato tabla.

Un ejemplo de archivo CSV:

```
Año,Marca,Modelo,Descripción,Precio  
1997,Ford,E350,"ac, abs, moon",3000.00  
1999,Chevy,"Venture ""Extended Edition""","",4900.00  
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00  
1996,Jeep,Grand Cherokee,"MUST SELL!  
air, moon roof, loaded",4799.00
```

Y su representación en formato tabla:

Año	Marca	Modelo	Descripción	Precio
1997	Ford	E350	ac, abs, moon	3000.00
1999	Chevy	Venture "Extended Edition"		4900.00
1999	Chevy	Venture "Extended Edition, Very Large"		5000.00
1996	Jeep	Grand Cherokee	MUST SELL! air, moon roof, loaded	4799.00

2.3.1.2 Bases de Datos:

A pesar la utilidad de los archivos CSV, se quedan pequeños para las tareas a las que se ven sometidas los sistemas de IoT, y no resultan del todo adecuadas para gestionar la cantidad de información generada. Además, en caso de modificaciones en la colecta de datos, es preciso cambiar también el código del programa asociado a dicha operación, lo que puede suponer problemas de incompatibilidades del sistema en su conjunto. Por ello, generalmente se utilizan bases de datos en del desarrollo de aplicaciones y sistemas de IoT.

En general, se busca que las bases de datos tengan una serie de características muy concretas para ser consideradas de utilidad para el IoT. Estas son:

- Escalabilidad
- Capacidad para gestionar grandes cantidades de información a la velocidad adecuada.
- Estructura flexible.
- Compatibilidad con herramientas de análisis, seguridad y coste.
- Capacidad de tolerar fallos y estar continuamente disponible.
- Capacidad de almacenar los datos en un sistema de mensajes si algún servidor se encuentra caído o la frecuencia de escritura sobrepasa ciertos límites.

A continuación, se realizará una pequeña introducción a algunas de estas bases de datos y se escogerá la más adecuada para el desarrollo del proyecto.

- **InfluxDB**

InfluxDB es una base de datos distribuida de código abierto, desarrollada por InfluxData. Está escrito en lenguaje Go y se basa en LevelDB. Es una base de datos con estructura clave-valor y proporciona una interfaz HTTP y bibliotecas a los usuarios para la interacción de la base de datos. Su principal ventaja es la facilidad de automatizar la agregación de valores en intervalos de tiempos definidos.

Se puede acceder a ella mediante software como Grafana, una potente herramienta front-end que provee funciones de visualización para datos en series temporales. No tiene dependencias externas y las consultas de tipo SQL se realizan a través de una estructura de mediciones, series y puntos. Cada punto consta de pares clave-valor que pueden ser: enteros de 64 bits, puntos flotantes de 64 bits, cadenas de caracteres y booleanos. Los datos se pueden almacenar a través de HTTP, TCP y UDP.

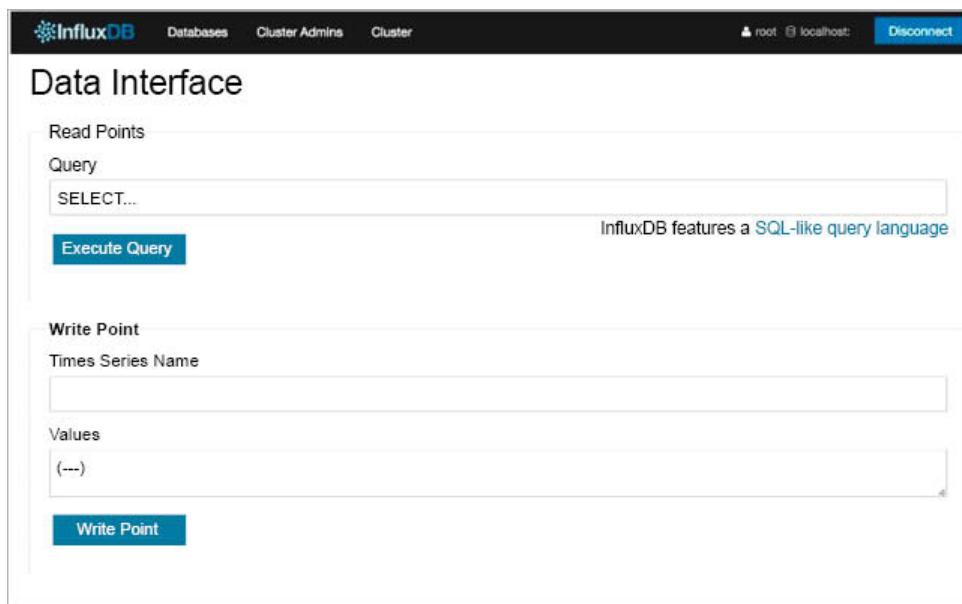


Figura 2.6: Interfaz de InfluxDB

- **MongoDB**

MongoDB es una base de datos de código abierto, flexible y gratuita. Está orientada a documentos y tiene capacidad de escalar funciones como: índices secundarios, consultas de rango, clasificación, agregaciones e índices geoespaciales. Se clasifica como una base de datos NoSQL ya que utiliza documentos similares a JSON con esquemas.

MongoDB agrega relleno dinámico a los documentos y asigna previamente archivos de datos para intercambiar espacio de uso adicional. Hace un rendimiento eficiente de la

memoria RAM para el almacenamiento en caché y la corrección de consultas para índices. Admite un lenguaje de consulta enriquecido para llevar a cabo operaciones de lectura y escritura (CRUD), así como agregación de datos, búsqueda de texto y consultas geoespaciales. Además, admite colecciones TTL (Time-To-Live) para datos que deben caducar después de cierto periodo de tiempo y tiene una herramienta de servicio de administración MMS que permite rastrear bases de datos y realizar copias de seguridad de los datos.

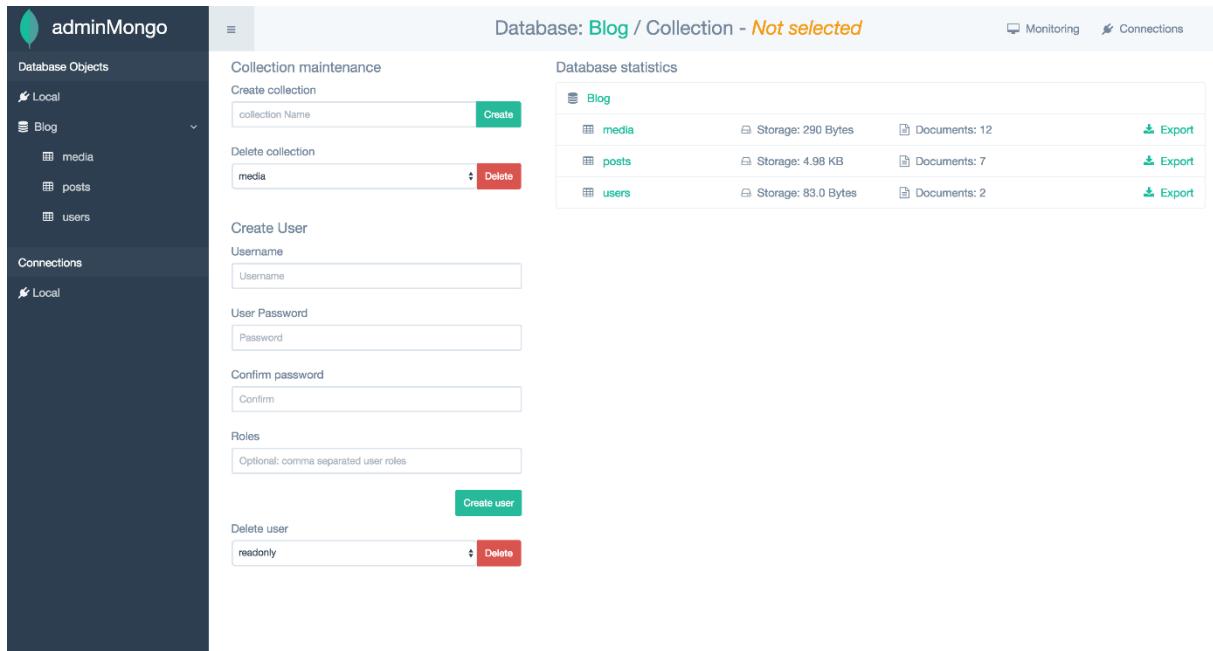


Figura 2.7: Interfaz de usuario de MongoDB

- **MySQL**

MySQL es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS) escrito en lenguaje C y C++. MySQL es un componente central de la pila de software de aplicaciones web de código abierto LAMP. MySQL también se utiliza en muchos sitios web a gran escala y de alto perfil, como Google (aunque no para búsquedas), Facebook, Twitter, Flickr, y YouTube.

Proporciona el servidor como un programa separado para su uso en un entorno de red cliente-servidor, y como una biblioteca que puede integrarse en aplicaciones

independientes. Tales aplicaciones se pueden usar de forma aislada o en entornos donde no hay una red disponible. Tiene una gran portabilidad y trabaja con un amplio conjunto de ANSI SQL 99, así como con extensiones. Posee soporte multiplataforma y un sistema de procedimientos almacenados, y permite la adición de motores de almacenamiento muy diversos (InnoDB, MyISAM, CSV, Blackhole y NDB Cluster entre otros). El diseño de servicio es multicapa con módulos independientes. Y tiene un esquema de rendimiento que recopila y agrega estadísticas sobre la ejecución del servidor y el rendimiento de las consultas con fines de supervisión. Soporte de SSL y caché de consultas. Biblioteca de bases de datos incorporada. Soporte de Unicode.

• MariaDB

MariaDB es una bifurcación (o *fork*) muy popular de MySQL, creado por los desarrolladores originales de MySQL. Ofrece soporte tanto para tareas de procesamiento de datos pequeñas como para ámbitos corporativos. Su propósito es reemplazar por completo a MySQL, haciendo la transición entre ambas muy sencilla, simplemente desinstalando la primera e instalando MariaDB. Ofrece las mismas características de MySQL y muchas más. Las más importantes son:

- Todo MariaDB se encuentra bajo GPL, LGPL o BSD.
- Incluye una amplia variedad de motores de almacenamiento de alto rendimiento y que permiten trabajar con otras bases de datos relacionales.
- Utiliza un lenguaje estandarizado y extendido de consulta.
- Se ejecuta en prácticamente todos los sistemas operativos y trabaja bajo una gran cantidad de lenguajes de programación.
- Ofrece soporte para PHP, uno de los lenguajes de desarrollo web más populares.
- Ofrece la tecnología clúster Galera, que soporta formatos esclavo-maestro y maestro-maestro.
- Soporta muchas operaciones y comandos que no están disponibles en MySQL, y elimina o reemplaza funcionalidades que afectan negativamente el desempeño.



• Conclusión

Para llevar a cabo el almacenamiento de los datos recogidos por el sensor conectado a la Raspberry Pi, una vez mandados a través del protocolo MQTT, se utilizará la base de datos tipo MySQL. En particular, se utilizará MariaDB, que es una interfaz compatible con subprocesos para el servidor de MySQL y que proporciona la API de base de datos

de Python. Y, dado que el proyecto se desarrollará en lenguaje Python, resulta ideal por ese y los siguientes motivos:

- Compatibilidad con objetos JSON.
- Es gratis y de código abierto.
- Transacciones de información seguras.
- Escalabilidad bajo demanda, pudiendo ofrecer una flexibilidad sin igual que facilita la gestión de aplicaciones integradas aun con grandísimas cantidades de información.
- Facilidad de instalación y uso.
- Alta disponibilidad, está diseñado para soportar millones de consultas y miles de transacciones.
- Alta confiabilidad, SSH y SSL proporciona confianza en torno a las conexiones y posee herramientas potentes para la encriptación de los datos.

2.4 Blockchain

2.4.1 Introducción

Por Blockchain, se entiende una base de datos distribuida entre diferentes participantes (o nodos), protegida criptográficamente, inmutable y organizada en bloques de transacciones relacionados matemáticamente entre sí. El principal atractivo que posee esta tecnología es que permite que entidades entre las que no existe confianza plena, puedan sostener un consenso sobre la existencia, estado y evolución de una serie de elementos compartidos.[12]

Éste consenso es el fundamento que permite que todos los integrantes confíen en la información que se encuentra grabada en él, y es la clave que convierte a los sistemas Blockchain en herramientas con un potencial enorme para llevar a cabo transformaciones en sectores claves de la industria y la sociedad. Pues permite prescindir de organismos reguladores y centralizados que verifiquen la integridad de la información registrada en las Blockchain, ya que es la propia red la que se encarga de ello.

Estos sistemas están construidos a partir de una red global de ordenadores que gestionan una gigantesca base de datos, y haciendo una primera distinción, nos podemos encontrar con redes abiertas a la participación de cualquier entidad (o Blockchain públicas) o con redes limitadas a ciertos participantes (Blockchain privadas).

2.4.1.1 Elementos Básicos de una Blockchain[12]

- **Nodo:** Por nodo se entiende a un participante activo de la red. Puede tratarse de un ordenador personal, un computador de placa base reducida (tipo Raspberry pi o Asus TinkerBoard), o incluso de una mega computadora).
- **Protocolo estándar:** Del mismo modo que existen protocolos tan conocidos como el TCP/IP para internet o el SMPT para el correo electrónico, todos los nodos que forman parte del sistema, han de compartir un software común para poder comunicarse entre sí, independientemente de sus características técnicas y su poder de computación.
- **Una red P2P:** es decir, una red de nodos conectados en una misma red.
- **Un sistema descentralizado:** En el que todos los nodos controlan de manera equitativa la red, es decir no existe una jerarquía en la que haya entidades reguladoras y centralizadas que controlen la información.

2.4.1.2 Las claves de la tecnología Blockchain[12]

- **La criptografía:** Es un método en el cual, a partir de un algoritmo con clave de cifrado, se transforma un mensaje de forma que resulte incomprensible para toda aquella entidad que no posea la dicha clave. En las tecnologías Blockchain, es aquello que tiene la responsabilidad de proveer un mecanismo infalible para la codificación segura de las reglas del protocolo que rigen el sistema. Además, se encarga de las importantes tareas de evitar la manipulación, hurto o introducción errónea de información en la red, así como de la generación de claves (públicas y privadas) e identidades digitales.
- **Cadena de bloques:** Es la base de datos diseñada para el almacenamiento de los registros realizados por los usuarios. Todas las Blockchain han de actuar bajo las mismas reglas o protocolos para dar validez al bloque e incorporarlo a la cadena. Y, una vez hecho esto, la cadena continuará con la emisión del siguiente bloque, permaneciendo inalterable la información registrada a través de la criptografía, lo que elimina la necesidad de terceras partes reguladoras.
- **Consenso:** Está sustentado en un protocolo común que verifica y confirma las transacciones realizadas, y asegura la irreversibilidad de las mismas. Es fundamental que proporcione a todos y cada uno de los nodos participantes de la red una copia inalterable y actualizada de la cadena de bloques.

2.4.1.3 Blockchain privadas y públicas[12]

Las Blockchain se pueden dividir en dos grandes grupos, las privadas y las públicas. Las primeras Blockchain, fueron de carácter público y estuvieron diseñadas para ser:

- **Públicas:** Cualquier persona puede acceder y consultar la cadena de bloques si necesidad de ser usuario.

- **Abiertas:** Cualquier persona puede convertirse en usuario y participar en el protocolo.
- **Descentralizadas**
- **Pseudoanónimas:** Las direcciones asociadas a los usuarios y a las que se realizan las transacciones son rastreables y de carácter público, pero no así las identidades de los usuarios propietarios de dichas transacciones.

Las unidades de cuenta que se utilizan en las Blockchain públicas se dominan tokens, y son cadenas alfanuméricas, aceptadas por el consenso de la red, del tipo 3jpskd72387r3gdgedhqdb que representan un registro en la Blockchain.

Posteriormente han ido surgiendo otro tipo de Blockchain de carácter privado y sus principales características son:

- **Privadas:** Sólo los usuarios participantes de la red pueden acceder y consultar todas o algunas de las transacciones realizadas.
- **Cerradas:** Se necesita una invitación para poder participar en la Blockchain. Otorgando diferentes tipos de acceso a los usuarios, en función de los fines perseguidos.
- **Distribuidas:** En general, la fortaleza de una Blockchain radica en la cantidad de los nodos que la protegen y en los incentivos que se reciben por cumplir dicho papel. A mayor número de nodos operativos, menor es la probabilidad de que se realice un ataque con éxito sobre la cadena. En las Blockchain privadas, los participantes se comprometen a mantener la estabilidad del sistema.
- **Anónimas:** o no, dependiendo del nivel de privacidad que se quiera instaurar en la red.

2.4.2 Orígenes de Blockchain[12]

2.4.2.1 El boom de la criptografía

En apenas 50 años, la criptografía en Estados Unidos pasó de ser monopolio absoluto de las agencias de inteligencia como la National Security Agency (NSA), a formar parte del dominio público y por tanto sujeto a una gran cantidad de mejoras. Los cypherpunks y el movimiento que lideraron en pos de la liberación de estas técnicas jugaron un papel fundamental y fueron inspirados en gran parte por personajes como: Ron Rivest, Adi Shamir y Leonard Adleman con su algoritmo RSA o David Chaum con DigiCash.

En 1969, Whitfield Diffie, co-creador del protocolo Diffie-Hellman, ya comentó: “*La criptografía es vital para la privacidad humana*”. Este concepto de la protección de la privacidad y el uso del dinero anónimo son vitales para entender los intereses subyacentes al movimiento cypherpunk y la creación de la primera Blockchain, Bitcoin.

En concreto, fue precisamente Whitfield Diffie, junto con Hellman (al que conoció en Standford mientras estudiaban métodos criptográficos novedosos), el que en la década

de los 70 desarrollaría el concepto de criptografía de clave pública (Public Key Cryptography) y que en la actualidad es el método que se utiliza en comunicaciones electrónicas por internet vía SSL/TLS y en las Blockchain como Bitcoin. La unión posterior a ambos de Ralph Merkle y su concepto de los Merkle Tree también fueron fundamentales en el desarrollo de la criptografía.

En la década de los 90, la idea de la sustitución del dinero en metálico por dinero digital se encontraba en el camino de ser aceptada por la mayoría del público, y ante esta situación, David Chaum fundó DigiCash, una empresa de dinero electrónico que utilizaba los métodos criptográficos previamente comentados para asegurar la viabilidad y la confianza en las transacciones. A pesar de lo revolucionario de la idea, esta fracasó y en 1998 se declaró en bancarrota. Inicialmente había captado la atención de compañías tan importantes como Microsoft y VISA, pero Chaum cometió una serie de errores que desembocaron en el fallo de su tecnología:

- El primero de ellos fue centrarse demasiado en grandes corporaciones, sin tener antes una base sólida de usuarios que sustentase el sistema.
- Otro, fue emplear tecnología propia, protegida con patentes y licencias, lo que dificultaba enormemente la adopción de la solución, así como la revisión, mejora y adaptación de esta. Además, estas empresas, debían confiar plenamente en que el protocolo ideado por Chaum no fuese a volverse en su contra en algún determinado momento.

Como dato curioso, uno de los colaboradores de Chaum, Zooko Wilcox-O'Hearn, fue el promotor y creador de Z-Cash, que, desde su lanzamiento en 2016, asegura el anonimato de sus tokens y actualmente es una de las criptomonedas con mayor cotización en el crypto-mercado y más adoptada por entidades financieras.

2.4.2.2 Hacktivismo y los Hackers

Por hacktivismo, se entiende el software destinado a facilitar la comunicación segura entre personas de distintas naciones y resistente al espionaje de los gobiernos[12]. Además, también se usa para referirse a la libertad de expresión en las redes y a la posibilidad de manifestarse y criticar al sistema sin temor a las represalias.

Los hackers, son aquellos individuos que hacen del hacktivismo su filosofía de vida y se sienten entusiasmados por la creación de máquinas más sofisticadas. Creen firmemente que la tecnología tiene el deber de aportar al mundo y por este motivo, comparten e intercambian ideas y códigos para encontrar soluciones a problemas técnico-informáticos de distinta índole.

Por otro lado, los cyberpunks son aquellos individuos, que llevan a cabo una labor más intelectual, y defienden con vehemencia la libertad de expresión, de información y la privacidad de las comunicaciones. De ellos, surgen los cypherpunks en 1992, que se valieron de la criptografía y la tecnología para alcanzar dichos objetivos en el mundo

digital. Su programa e ideología se encuentra recogida en *The Crypto-Anarchist Manifesto (1992)* y bebe de las ideas de los movimientos del hacktivismo y los creadores de métodos criptográficos previamente mencionados.

Dentro de los casos del activismo realizado por los cyberpunks y que fueron claves en la liberación y desarrollo de técnicas criptográficas, destaca el hackeo a la versión internacional del navegador más popular de la década de los 90, Netscape. Este software utilizaba la encriptación RC4 de 40 bits, que, en comparación con la versión estadounidense de 128 bits, se quedaba realmente corta en cuanto a capacidad de protección. El motivo por el cual existía esta diferencia es que Estados Unidos había prohibido a las empresas la exportación de productos con tecnologías potentes de cifrado. Fue un estudiante francés el que consiguió romper la seguridad de Netscape y se demostró lo absurdo de las leyes promovidas por la NSA que tenían la consecuencia de limitar el desarrollo de la criptografía en otras partes del mundo. Este caso, abrió los ojos en cuanto al tema, y permitió la liberación de las técnicas criptográficas que concluirían en la creación de tecnologías Blockchain.

2.4.3 Criptografía y Consenso en la Blockchain[13]

2.4.3.1 Criptografía

Por criptografía se entiende el arte de transformar un mensaje legible, en un ilegible. A este proceso se le llama “cifrado”, y al que se encarga de recomponer un mensaje en formato legible, “descifrado”. En la actualidad hay tres tipos principales de criptografía y todas se usan en el ecosistema Blockchain: Hashing, criptografía simétrica (o convencional) y criptografía asimétrica (o de clave pública).

2.4.3.2 Hashing

Hash es un término anglosajón, cuya traducción literal es moler o picar, y se trata de una imagen gráfica, ya que la criptografía que se basa en esta técnica consiste en “moler” unos contenidos hasta obtener una secuencia de caracteres que tengan una longitud previamente determinada, se trataría entonces del equivalente a una firma digital de un mensaje.

En la práctica, el hash de un determinado contenido se obtiene mediante la aplicación de una función matemática unidireccional. Al aplicar esta función sobre el mismo dato de entrada, siempre se obtendrá la misma salida. Por tanto, si se modifica dicha entrada, el hash resultante cambiará por completo y ésta es una característica de vital importancia para el desarrollo de tecnologías Blockchain, ya que se utiliza para generar firmas digitales y verificar la integridad de los datos, al detectar si se ha modificado la información, pues en ese caso, el hash también cambiaría.

Otra característica fundamental de estas funciones unidireccionales es que, a pesar de conocer el procedimiento del cálculo del hash en la dirección de salida, conocer el proceso inverso, es decir, hallar el dato de entrada a partir de hash generado, resulta tremadamente complicado. Es prácticamente imposible realizar dicho cálculo en un tiempo razonable, como ejemplo para explicar esto, es típico el caso de la factorización de los números enteros: Cualquier ordenador actual es capaz de multiplicar con facilidad números primos muy grandes o miles de bits, pero no se conoce ningún algoritmo eficiente para hallar los números primos iniciales a partir de su multiplicación.

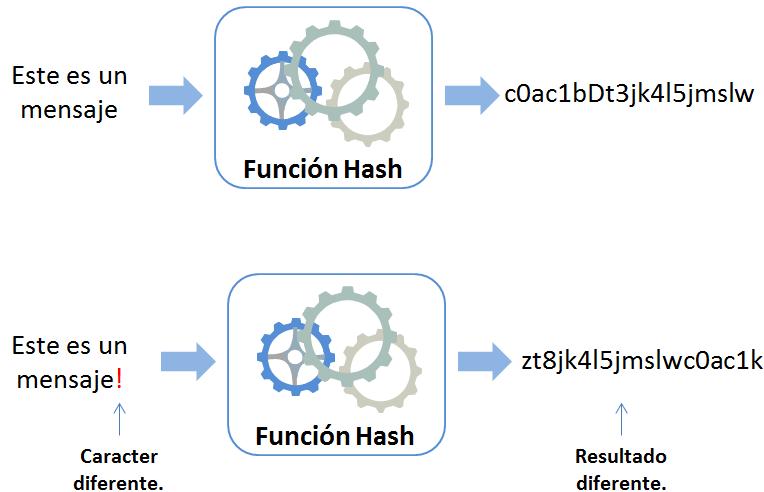


Figura 2.8: Ejemplo de funcionamiento de función hash

Las características fundamentales de este tipo de funciones son:

- **Eficiencia de cálculo:** Generan hash rápidamente a bajo coste.
- **Resistencia a preimagen:** Que no se pueda prever el hash que se va a generar con un mensaje de entrada, o que sea *extremadamente* complicado a nivel de computación.
- **Resistencia a segunda preimagen y a colisión:** Que sea computacionalmente muy complicado obtener el mismo hash a partir de dos mensajes de entrada distintos.

La mejora en funciones hash se ha ido consiguiendo partiendo de pruebas de ataque realizadas por los criptoanalistas, esto hace que los algoritmos de funciones que se utilizaban en la década de los 90 se encuentren “*rotos*” o significativamente amenazados. En el caso de Bitcoin, se utilizan dos tipos de algoritmos de hash:

- SHA-256, como función hash principal, que significa *Secure Hash Algorithm* y pertenece a la familia de funciones de hash SHA-2 diseñada por la NSA. Estas funciones son las aceptadas por el *National Institute of*

- *Standards and Technology* (NIST) como el estándar de algoritmos hash para la administración estadounidense.
- RIPEMD-16 (*RACE Integrity Primitives Evaluation Message Digest*), que se usa en el proceso de creación de direcciones, cuando se requiere un hash de menor longitud. Dispone de longitudes de salida de 128, 160 (la utilizada por Bitcoin), 256 y 320 bits.

Protocolos Blockchain como Bitcoin recurren a *doble hash*, es decir, aplicar dos veces un algoritmo hash. En 1979 Ralph Merkle con su invención del *Merkle Tree* o “árbol de Merkle”, dio con una estructura piramidal de hashes en la que cada hash es el resultado de aplicar una función hash sobre los hashes inferiores hasta llegar a un nodo raíz o Merkle Root. Estos árboles permiten verificar de forma eficiente y segura la integridad e inyección de grandes datos de información y resulta idónea para las cadenas de bloques. La cabecera de cada bloque contiene, entre otros datos, el nodo raíz (Root Hash) de todas las transacciones incluidas en el bloque. Más adelante se explicará en detalle la estructura de los Merkle Trees y de los bloques.

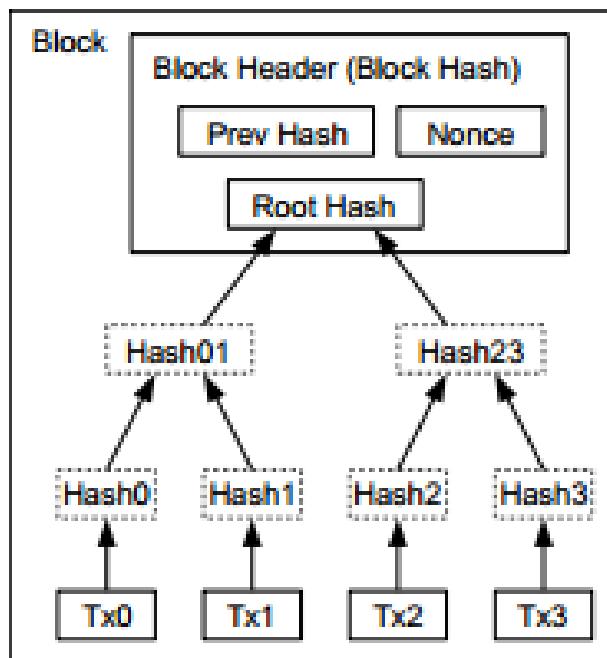


Figura 2.9: Modelo Merkle Tree

2.4.3.3 Criptografía Simétrica o Convencional

Antes de explicar en qué consiste este tipo de criptografía es importante llevar a cabo una distinción entre *Claves* y *Contraseñas*.

- Por contraseña se entiende como el “código secreto que se introduce en una máquina para poder accionar un mecanismo o para acceder a ciertas

- funciones informáticas”. Por ello, comúnmente son cortas y están pensadas para poder ser memorizadas por una persona, y por ello, vulnerables.
- Por otro lado, una clave es un “código de signos convencionales y cifrados que se emplea para escribir y leer mensajes secretos”. Son más seguras que las contraseñas por su gran tamaño y por su generación, que suele ser aleatoria.

Dado que las claves no son memorizables, se suelen guardar en archivos cifrados o en dispositivos hardware específicamente pensado para ello (*Hardware Wallets*).

La criptografía simétrica o criptografía de clave secreta es un método criptográfico que utiliza una sola clave, tanto para cifrar un mensaje como para descifrarlo, entre un emisor y un receptor. Ambas partes involucradas han de ponerse de acuerdo en la clave a utilizar. Para que sea seguro, es imprescindible que sea muy difícil adivinar el tipo de clave y es por ello que la seguridad ha de centrarse en la clave, y no en el algoritmo que la genera. En concreto, cuanto mayor sea el número de claves posibles de generar, más robusto es el sistema criptográfico.

Un algoritmo muy extendido para el cifrado de clave simétrica es el AES (*Advanced Encryption Standard*) que utiliza claves de hasta 56 bits, es decir, proporciona 2^{56} combinaciones distintas posibles de ceros y unos, un numero representable con una cifra decimal de 77 cifras de largo, que resulta tremadamente seguro por su gran longitud.

2.4.3.4 Criptografía Asimétrica o de Clave Pública y Clave Privada

La criptografía asimétrica se vale de dos claves:

- Una Privada: es un numero aleatorio tan grande que resulta probabilísticamente imposible generar otra igual y se usa para cifrar y descifrar mensajes, y para calcular las claves públicas a partir de algoritmos como RSA o ECDSA (*Elliptic Curve Digital Signature Algorithm*).
- Una Pública: que conocerá todo el mundo y que se utiliza para cifrar y enviar mensajes.

Un ejemplo de aplicación para aclarar el funcionamiento:

Sean dos personas, Alicia y Bruno, cada uno de ellos con sus correspondientes claves privadas y públicas. Si Alicia quisiera enviar un mensaje a Bruno, buscaría la clave pública de éste y cifraría el mensaje con ella. Bruno, una vez que recibiera dicho mensaje, utilizaría su propia clave privada para descifrarlo. La característica fundamental del sistema es que *la clave con la que se cifra un mensaje no sirve para descifrarlo*. Conceptualmente, la clave pública actuaría como un candado al mensaje, y la clave privada sería la llave de dicho candado.

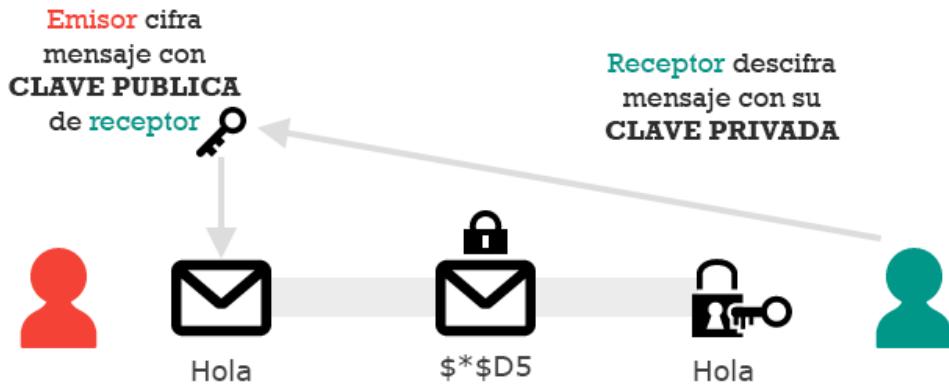


Figura 2.10: Ejemplo mensaje enviado mediante criptografía asimétrica

Las claves privadas en el contexto Blockchain suelen ser números aleatorios de 256 bits, y que se suelen representar en el sistema hexadecimal. En el caso de IOTA se utiliza el sistema trinario para representar las claves privadas y públicas, y las direcciones.

Se detallarán más adelante los algoritmos específicos de generación de claves tanto privadas como públicas y de direcciones que utilizan IOTA y BigchainDB.

2.4.4 Consenso en la Blockchain[12]

Las reglas de consenso son las reglas que deberán cumplir los bloques para ser admitidos en una cadena y, para poder llegar un consenso en una red descentralizada del tipo Blockchain, es fundamental conocer dichas reglas con anterioridad.

Los nodos que forman la red son los responsables de actualizar la Blockchain y quienes verifican los bloques antes de incorporarlos a la cadena. Los mineros son los encargados de crear los bloques y por tanto los que deciden cuál es la cadena legítima en caso de discrepancia. En el caso de las Blockchain públicas cualquiera puede ser minero y bastará con reunir los requisitos y recursos necesarios.

Es en este consenso, donde se ha dado un cambio de paradigma radical. Dado que no existen autoridades centrales a las que acudir en caso de discrepancias y que sirvan de “árbitros” del juego, han de ser las propias reglas del juego las que proporcionen a los “jugadores” los incentivos adecuados para que actúen de manera honesta. De hecho, la presunción de partida es que los participantes actuarán de manera “deshonesta”, o, en otras palabras, la manera en la cual su beneficio se maximizará. Es por ello que elegir cuidadosamente las reglas del consenso es un tema de máxima importancia y sumamente delicado. Este principio tiene su base en la teoría de juegos, y es la clave para la estabilidad de todo el sistema.

En el contexto de las Blockchain públicas, cualquier cambio que se realice en la cadena (como puede ser la inserción de un nuevo bloque), ha de contar con el respaldo y la verificación de la mayoría de los participantes. Para alcanzar este respaldo, actualmente se utilizan principalmente dos algoritmos distintos: Proof of Work (PoW) y Proof of Stake (PoS).

2.4.4.1 Pruebas de trabajo (Proof of Work)

Es recurrente en la literatura Blockchain recurrir al algoritmo Hashcash para explicar el funcionamiento de las pruebas de trabajo o PoW. Inventado en 1997 por Adam Back, el objetivo principal de este algoritmo era combatir el correo basura y para ello se estableció un mecanismo para verificar que el emisor del mensaje había “pagado” por emitir dicho mensaje, con tiempo de computación de su CPU, para poder firmar con una marca dicho mensaje. Así, en caso de querer enviar grandes volúmenes de correos, el emisor tendría que soportar altos costes y ello le disuadiría de llevar a cabo dicha actuación. Estas pruebas de trabajo consistían en el cálculo de raíces cuadradas de números muy grandes o en encontrar colisiones parciales de hash.

Unos años más tarde, Hal Finney desarrolló un sistema de pruebas de trabajo reutilizable o RPOW (*Reusable Proof of Work*). La idea de Finney se basaba en la generación de tokens digitales por parte de los usuarios que posteriormente se podrían usar o vender.

Esas pruebas de trabajo reutilizables y universales es lo que hoy día llamamos criptomonedas. El problema era que, como cualquier objeto del mundo digital, dichos tokens eran susceptibles a ser duplicados y/o falsificados. Este problema en concreto, la posibilidad de mandar el mismo token a dos direcciones (o usuarios) distintos, se conoce como “doble gasto” y es una de las principales amenazas a solventar a la hora de elaborar una Blockchain efectiva.

La solución a estos problemas vino de mano de Satoshi Nakamoto cuando, en 2008, publicó el Whitepaper de Bitcoin revolucionando por completo el mundo de la moneda digital. El protocolo Bitcoin se valía de una red descentralizada del tipo P2P, junto con el algoritmo Hashcash para resolver algunos de estos problemas.

Bitcoin es un registro inmutable con la historia de todos los tokens desde su creación inicial. Los usuarios del sistema se valen de un método criptográfico asimétrico, y a partir de sus claves públicas y privadas realizan transacciones mediante un protocolo de firma electrónica de las mismas. El cumplimiento de las normas del sistema sería fácilmente verificable, ya que, al ser el historial de las transacciones de carácter público, cualquiera podría verificar el origen de los tokens, su validez y legitimidad.

En este contexto, alguien con suficiente poder de minado podría atacar la red, cambiando las reglas del sistema sin haber alcanzado un consenso previo. Si se diera esto, los usuarios de la red se encontrarían con dos cadenas de bloques distintas, la original, y la creada por la entidad “maliciosa”.

La solución propuesta en el Whitepaper de Bitcoin para determinar la legitimidad de las cadenas de bloques en estos casos consistiría en crear una cadena de hashes cuyo objetivo sea verificar la acumulación de pruebas de trabajo, de esta manera, la cadena de bloques con más confirmaciones será aquella con un mayor número de pruebas de trabajo acumuladas, y, por tanto, la aceptada por los usuarios.

Para poder comprender el sistema en su totalidad, es necesario introducir los conceptos criptográficos de nonce y dificultad:

- **Nonce:** Por nonce, en criptografía, se entiende un número arbitrario, aleatorio o secuencial que se utiliza una única vez al realizar una operación. En el caso de Blockchain, en cada intento de cálculo del hash correcto, el nonce cambiará y por tanto se asegurará que el próximo hash calculado será distinto ya que el hash está directamente relacionado con los datos de entrada, y el nonce es uno de ellos.
- **Dificultad:** Se refiere a lo complicado que resultará hacer una prueba de trabajo para verificar un bloque determinado, es decir, definir en términos de computación y por tanto temporales cuánto costará encontrar el hash correcto de un nuevo bloque para ser insertado en la cadena. En general, esto se hace especificando un número de ceros concretos con los que debe empezar el hash. A dicho número se le denomina target u objetivo.

Si el hash calculado comienza con un número de ceros distinto al impuesto por la dificultad del consenso, se modifica el nonce y se vuelve a intentar. Se repetirá este proceso un numero aleatorio de veces hasta que se dé con el hash que contenga el número de ceros requerido. Al ser imposible determinar de antemano el hash a calcular, habrá que realizar millones de pruebas hasta dar con el hash acertado, lo que se traduce en un tiempo determinado mínimo necesario.

Esto último se puede entender fácilmente, utilizando un ejemplo con una moneda. Dado que las probabilidades de obtener cara son del 50%, si queremos obtener 50 caras, habría que realizar en promedio 100 pruebas. Si establecemos un tiempo mínimo de tirada (incluyendo el tiempo de preparación y caída de la moneda) de 2 segundos, se necesitarán en promedio 200 segundos para obtener 50 resultados cara.

En concreto en el sistema Bitcoin, el consenso determina que se tarde unos 10 minutos en calcular el hash correcto. Para ello, se calcula el promedio del tiempo invertido en calcular los hashes en las últimas dos semanas y, en caso de que este tiempo sea menor de 10 minutos, ello implica que los mineros están aportando una gran capacidad de computación al sistema, y, por tanto, es necesario aumentar la dificultad, incrementando el número de ceros necesarios. Del mismo modo, si se tardase más de 10 minutos, se rebajaría la dificultad del cálculo de hash. De esta manera, el sistema se autorregula en función del poder de computación dedicado a estos cálculos, y se mantiene el tiempo especificado.

Si el hash calculado no comienza con el número de ceros especificado, el nonce cambia y se intenta de nuevo, así hasta que aleatoriamente (por fuerza bruta) el ordenador da

con un hash que tenga ese número de ceros delante. Recordemos que es imposible saber de antemano cuál será el hash, por lo que habrá que hacer miles o millones de intentos antes de dar con el hash requerido.

Mantener un tiempo de cálculo de referencia es de vital importancia para poder discernir que cadena de bloques es la legítima en caso de discrepancias y para determinar qué minero ha sido el que ha conseguido realizar la prueba de trabajo primero.

El sistema establece que después de la orden de emisión de validación e inserción de un bloque, en un tiempo determinado, se realizará un recuento de las confirmaciones que posee la cadena, y aquella que posea más, será la legítima, y el minero que haya verificado dicho bloque, será el recompensado con el incentivo propuesto por la red.

En general, las Blockchain disponen de los siguientes elementos:

Los nodos: son los validadores de los bloques y transacciones, y los que actualizan la Blockchain.

Los wallets: generan las transacciones, las firman y las envían a los nodos para ser validadas y puestas en una lista de espera (mempool, o piscina de memoria) a disposición de los mineros.

Los mineros: generan los bloques de la cadena. Para ello obtienen el merkle tree de las transacciones que quieran escoger entre aquellas que ya se encuentran en la lista de espera. El root hash de ese árbol de merkle lo usan como parte del mensaje a hashear con un algoritmo de prueba de trabajo.

Cuando un minero logre insertar un bloque con su hash correcto, ese será el primer voto válido y el primer eslabón de esa cadena. Enviará el número de voto, el bloque, el nonce y el hash a todos los demás usuarios de la red. A continuación, el resto de los mineros comenzarán a buscar el segundo hash, utilizando el hash anterior como parte del contenido a hashear.

El minero que encuentre el siguiente hash (el segundo voto de la cadena) lo enviará a los demás para que alguno de ellos logre crear el tercer voto y así sucesivamente. De esta manera, se empieza a formar una cadena de hashes o de pruebas de trabajo, esta cadena es necesaria para llevar la contabilidad de los votos y permite la verificación independiente de los bloques por cada uno de los participantes.

Ante un nuevo bloque y antes de sumarlo a su propia copia de la cadena, los mineros realizan una serie de comprobaciones:

1. ¿El hash del nuevo bloque comienza con el número de ceros especificados?
2. ¿Se obtiene ese hash con los contenidos del hash?
3. ¿Es el bloque el mismo que el del voto anterior?

Si responde afirmativamente a las 3 preguntas, el bloque obtiene una confirmación y por tanto es susceptible de ser insertado por consenso en la cadena. De esta forma, todos los usuarios van construyendo una versión actualizada de la cadena común.

Dos cadenas

En el caso de que otro minero hubiese publicado de manera simultánea otra cadena en la cual él fuera el que ha realizado la prueba de trabajo de determinado bloque en primer lugar, los participantes del sistema podrían encontrarse con dos cadenas distintas. En cuyo caso, cada minero elegirá aleatoriamente una de las cadenas y según vayan obteniendo confirmaciones por el resto de los mineros, habrá una que se diferencie claramente sobre la otra.

Dado que a la mayoría les interesa llegar a un acuerdo, la cadena con menos confirmaciones será ignorada y los mineros seguirán confirmando y tratando de insertar nuevos bloques en la cadena que cuente con más confirmaciones al cabo de un tiempo determinado, como el tiempo de cálculo de los hashes determina el número posible de confirmaciones que se obtienen en ese tiempo, la cadena correcta será la que posea dicho número de confirmaciones.

Incentivos

La pregunta lógica a la que se enfrentará el lector en este punto es: ¿Por qué los mineros incurren en este gasto de tiempo, computación y por tanto dinero?

La respuesta es que, en la mayor parte de las Blockchain basadas en PoW, por cada bloque minado con éxito, es decir, insertado en la cadena, el sistema recompensa al minero que ha realizado dicha acción. En protocolos en los que existe una generación continua de tokens como Bitcoin, esta recompensa viene determinada por una cantidad de tokens específica (y que también varía con el tiempo). En cambio, en aquellos protocolos con una cantidad fija de tokens, en los que no se generan más, la recompensa viene de la mano de comisiones pagadas por parte de los usuarios que quieren realizar las transacciones. En estos casos, los mineros eligen los bloques que ofrece una mayor comisión y esos serán los que minarán.

En la actualidad, en el protocolo Bitcoin hay una recompensa de 12.5 tokens BTC por cada bloque minado con éxito, como la computación y el gasto energético asociada a ella se encuentra en valores muy elevados, esa recompensa apenas rentabiliza el minado. Este es el motivo de que actualmente en Bitcoin se recompense por ambas vías, cantidad fijada por el protocolo más las comisiones que ofrezcan los usuarios que deseen inscribir sus bloques en la cadena.

2.4.4.2 Proof of Stake (PoS)

Actualmente, las pruebas de participación o Proof of Stake (PoS), son la alternativa principal a las pruebas de trabajo. En lugar de tener una capacidad de creación de

bloques en función de la capacidad de proceso que posean, con PoS, los mineros pueden crear bloques de manera proporcional a la cantidad de tokens que posean.

Para seleccionar al minero que realizará la creación del nuevo bloque, se suelen seguir dos procedimientos distintos:

- **Selección aleatoria:** El minado de cada bloque se asigna aleatoriamente por la red a cualquiera de los tokens existentes. Si el propietario de ese token está online, recogerá las transacciones y las comisiones, creará el bloque y lo enviará al resto de la red. Si no está online, la red seleccionará otro token. Este sistema beneficia claramente a aquellos que posean más tokens, pero a la vez son los más interesados en el buen funcionamiento del protocolo y que su token asociado tenga un alto valor.

- **Selección por antigüedad de la moneda:** Con este sistema se asigna el bloque a quien haya realizado una transacción con las monedas que acumulen más tiempo desde la última vez que se transfirieron.

En general, las Blockchain basadas en PoS, no suelen generar nuevos tokens y por tanto las recompensas obtenidas por minar bloques consisten únicamente en las comisiones.

Las ventajas fundamentales de este método son:

- 1) Menor coste de minado. Los procesadores de consumo para minar tienen un coste mucho menor que los utilizados en redes como Bitcoin, lo que el coste de mantener la red es mucho menor, y con ello, las comisiones a pagar a los mineros.
- 2) Se reduce las posibilidades de un ataque del 51%: Ya que el atacante debería hacerse con el 51% de los tokens, situación muy complicada, aunque no imposible.

Pero estos sistemas no están exentos de problemas, el principal es que no sirven para alcanzar un consenso en caso de bifurcación de la cadena, ya que, debido a que no supone un gasto demasiado cuantioso, los mineros podrían seguir minando en ambas cadenas, sin tener ningún incentivo por decantarse por alguna. Es por ello que la mayoría de los nuevos protocolos Blockchain utilizan sistemas híbridos de PoW-PoS.

2.4.5 Blockchain y la Industria 4.0[12]

A lo largo de la historia, la industria y los procesos productivos han experimentado una serie de revoluciones:

- **La primera revolución industrial:** utilizó la energía del agua y del vapor para la producción mecanizada. Estos cambios modelaron los mercados y permitieron pasar de economías localizadas a descentralizadas, a gracias al ferrocarril y a las naves autopropulsadas. Las industrias se desarrollaron en

los sectores textil, metalúrgico y siderúrgico, llegando a desarrollar dispositivos mecánicos para sustituir labores manuales.

- **La segunda revolución industrial:** se inicia cuando se emplea la energía eléctrica aprovechada dentro del entorno productivo para mejorar la fabricación masiva en serie.
- **La tercera revolución industrial:** ha empleado la electrónica y las tecnologías de la información para automatizar la producción. Se caracteriza por una fusión de tecnologías que está borrando los límites entre las esferas físicas, digitales y biológicas.

Actualmente nos encontramos a las puertas de la 4^a revolución industrial, en la que la digitalización y la coordinación cooperativa de las unidades productivas van a suponer un desarrollo y una creación de valor sin precedentes.

En la sociedad moderna, la demanda es cada vez más sofisticada y personalizada, lo que para la industria se traduce en que, para poder satisfacer los requerimientos del público, es imprescindible llevar a cabo una producción más eficiente e inteligente, maximizando la rentabilidad de todos los procesos implicados, y disminuyendo radicalmente costes y tiempos marginales de fabricación.

Para ello tanto las empresas productoras, como la industria manufacturera, han de dar el salto a la creación de soluciones personalizadas y dirigirse hacia modelos de negocio basados en el servicio y el pago por uso. Así, la cuarta revolución industrial viene marcada por la creación de fábricas inteligentes en las sea posible producir miles de configuraciones distintas de producto y poder fabricar lotes muy pequeños a un coste muy reducido.

Esta industrial inteligente estará basada en la digitalización de absolutamente todas y cada una de las partes que integran y forman parte de los procesos productivos y se valdrán de tecnologías como el internet de las cosas (IoT), las comunicaciones Machine to Machine (M2M), plataformas en la nube y la impresión 3d, entre otras.

En este nuevo paradigma, los datos no serán más una consecuencia del proceso de fabricación, sino que tendrán valor de negocio por sí mismos. Como ejemplo de la fuerza y la velocidad con la que se está realizando esta transformación de la industria, es destacable la existencia de plataformas como Industrial Data Platfrom, Digital Manufacturing Comons o Skywise, cuyo objetivo es el intercambio de datos de producción y fabricación y para facilitar tareas de mantenimiento predictivo.

En particular, la tecnología del IoT resulta de especial interés en esta revolución, hasta el punto de que se acuñado un nuevo término específico para referirse a la aplicación de dicha tecnología al contexto 4.0: Industrial Internet of Things o IIoT. Los dispositivos que formen parte de estos ecosistemas escalables e integrables deben estar sujetos a una gestión extremadamente efectiva pues, la autenticación y las comunicaciones entre los diferentes sistemas ciberfísicos cobran una importancia mayúscula, fallos en la recolecta, tratamiento o almacenamiento de los datos generados

pueden tener consecuencias catastróficas en toda la cadena de producción e incluso repercutir en pérdidas humanas.

Es por ello, que la gestión descentralizada, inmutable e integra de los datos es de capital importancia y es en este contexto donde las tecnologías Blockchain pueden aportar un valor diferencial. Gracias a la aplicación de esta tecnología se puede registrar la actividad y la identidad de cada dispositivo integrado en el sistema de producción, sin miedo a la manipulación de los datos y a sus consecuencias.

Además, las Blockchain pueden integrarse a través de protocolos de comunicación entre maquinas, permitiendo la creación de una nueva economía entre los propios dispositivos en los que podrán llegar a acuerdos de suministros de materias primas, energía, piezas, mantenimiento e incluso logísticos a partir de contratos inteligentes o Smart Contracts cuyo pago se ejecutará automáticamente una vez se cumplan las condiciones previamente establecidas. Ya existen ejemplos de micropagos a través de la Blockchain o el Tangle con sensores que venden sus datos, y coches eléctricos que comercian con energía eléctrica con los puntos de recarga.

Esta integración supone la agilización y automatización de cientos de procesos que actualmente requieren de una gran cantidad de pasos intermedios que entorpecen y encarecen los procesos productivos actuales y, junto con el dramático descenso de la necesidad de intervención de terceras partes reguladores (y humanas), disminuirá enormemente el consecuente gasto que implica. De manera que, se podrán alcanzar la reducción en costes marginales necesarias para poder hacer frente a las necesidades de producción personalizadas y unitarias. Siendo la clave la desintermediación del proceso productivo, de manera que las empresas puedan recibir las peticiones de un portal descentralizado, incorruptible y fácilmente accesible por todas las partes involucradas.

Es destacable el caso de la empresa de transporte internacional de mercancía contenerizada, Maersk. Maersk es una compañía de origen Danés y se trata de la compañía con mayor cuota de mercado en su sector, con un rango entre el 18 y el 20% del mismo[14]. Actualmente, el coste de la gestión del transporte de un contenedor, debido a los permisos y trámites a necesarios con los países involucrados y las autoridades pertinentes, resulta superior al coste del transporte físico de dicho contenedor. Por lo que se trata de un claro caso de uso en el que la integración de las tecnologías Blockchain para automatizar procesos, y que todas las artes involucradas tengan acceso instantáneo a la información involucrada, además de la conciencia de que se trata de información inmutable e integra puede agilizar y por tanto reducir significativamente los costes de gestión. Además, en este contexto, la integración con tecnologías IoT permitiría el rastreo y el control de toda la carga transportada y por tanto la actuación conveniente en caso de necesidad.

2.4.6 Distributed Ledger Technologies (*Datos*) y Blockchain Adecuadas para el Proyecto

2.4.6.1 Bitcoin

Como declaró Satoshi Nakamoto en el whitepaper de Bitcoin, Bitcoin aparece como una solución, para tener un libro mayor descentralizado y distribuido donde las transacciones se pueden hacer sin comisiones e instantáneamente.

Inicialmente, en la creación de Bitcoin (2009), para cada bloque validado con éxito, había una recompensa de 50 tokens de Bitcoin (BTC). Esta cantidad se ha ido reduciendo a la mitad cada 210,000 bloques minados (lo que lleva alrededor de 4 años), y actualmente, la recompensa es 12.5 BTC. Esta recompensa es cómo se crean nuevos tokens y la recompensa al proceso de inserción de los bloques en la cadena, o minado, llevado a cabo por los mineros. Hay una cantidad fija de BTC, que estará disponible y el número exacto es de 21 millones [9], que se alcanzará en 2140.

Actualmente hay 16.7 millones de tokens disponibles. A medida que la cuantía de la recompensa disminuye con el tiempo, es más costoso en potencia de cálculo y, por tanto, en consumo de energía y tiempo, realizar el PoW necesario. Esto significa que hoy en día, el consumo de energía en la minería BTC está alcanzando niveles insostenibles. Por ejemplo: Islandia actualmente está dedicando más energía eléctrica a la minería Bitcoin que a la vivienda residencial.

En consecuencia, las únicas entidades que pueden hacer esta Prueba de Trabajo de una manera rentable (obteniendo más de los tokens recompensados que el coste de la energía requerida para hacer la minería), son los que tienen acceso a un gran poder de cómputo. Se conoce el caso de edificios enteros de apartamentos en China dedicados por completo a la minería de Bitcoin. Esto se conoce como "Mining Pools", y han supuesto que el poder de minado de Bitcoin y por tanto el potencial de creación y reparto se encuentre únicamente en manos de unos pocos, lo que lleva a una centralización del sistema.

Además, para obtener la recompensa que sirve como una tarifa de transacción, se vuelve mucho más difícil a medida que pasa el tiempo y la cadena de bloques se hace más grande. En consecuencia, la red se vuelve más costosa de usar y el tiempo de transacción se hace más largo. Por todos los motivos mencionados anteriormente, la idea original de Satoshi Nakamoto y la filosofía de Bitcoin están corruptas, ya que encarna ahora muchos de los problemas que quería resolver en sus orígenes.

2.4.6.2 Ethereum

Existen otros ledger distribuidos de segunda generación, como Ethereum, que no solo sirven como almacenamiento de valor, sino que también permiten contratos inteligentes entre dos o más participantes, utilizando un concepto Blockchain similar para deshacerse de terceros y partes centralizadas que validen esos contratos.

Ethereum también sirve como plataforma de desarrollo para otras criptomonedas y soluciones Blockchain. Actualmente, hay numerosos proyectos que se están construyendo sobre él. Siendo algunos de los más relevantes actualmente: EOS, Tron, Vechain, ICX...

Pero este tipo de Blockchain también enfrentan actualmente problemas. Recientemente se ha revelado el caso del juego Crypto-Kitties [13] que causó un aumento inesperado y dramático en el tráfico neto de Ethereum, que supuso el colapso del sistema. Este hecho puso de manifiesto los enormes problemas de escalabilidad que enfrentan las DLT.

2.4.6.3 IOTA

En este marco, aparecen DLTs de tercera generación con IOTA como su buque insignia. Revoluciona por completo el concepto de Blockchain, ya que su tecnología se basa en un Tangle o red interconectada de nodos, que permite que el sistema esté completamente descentralizado, tenga transacciones gratuitas, funcione mejor a medida que más usuarios participen en la red y proporcione protección contra los ciberataques cuánticos. La red de IOTA permite la transferencia de valor, sirve como una plataforma para el desarrollo y la configuración de contratos inteligentes. También está diseñado para permitir transacciones de datos y almacenamiento de información.

2.4.6.4 Hyperledger

Las anteriores tecnologías DLT comentadas, constitúan Blockchain públicas, es decir, en las que cualquiera puede acceder a las mismas y ver las transacciones realizadas por los usuarios, aunque no se pueda acceder a la identidad de los usuarios de forma explícita. Son consideradas por tanto Blockchain pseudoanonimas.

Hyperledger, por otro lado, se trata de una Blockchain privada, y considerando dicho subconjunto, de las más relevantes actualmente ya que empresas de la talla de IBM y organizaciones como la Fundación Linux, se valen de ella para el desarrollo de aplicaciones.

Se trata de una red de código abierto, nacida en 2015 y desarrollada por la Fundación Linux. Nace con el objetivo de servir como ecosistema en el que se puedan crear aplicaciones de código abierto, concretamente para el área empresarial y corporativo.

Uno de sus principales atractivos, es la herramienta Hyperledger Composer que permite hacer aplicaciones abstrayéndose de la parte de programación de Hyperledger Fabric, gracias a la cual tan solo hay que centrarse en la lógica de diseño de activos, participantes y transacciones. Esto lleva a que el desarrollo de Dapss ('distributed applications' o aplicaciones distribuidas) se realice de una manera mucho más rápida y eficiente.

Entre sus beneficios encontramos:

- Permite realizar un MVP (mínimo producto viable) de manera rápida y pudiéndose ejecutar para realizar tests.
- Disminución del tiempo de desarrollo.
- Reduce el riesgo por la reusabilidad de sus componentes.
- Flexibilidad.
- Sistema de canales que garantiza la privacidad de los datos compartidos entre los usuarios predeterminados.

2.4.6.5 BigchainDB:

BigchainDB es un software de código libre que aúna características propias de una base de datos junto con otras propias de una red Blockchain. A grandes rasgos, el sistema consiste en una red, entre un número determinado de ordenadores o nodos, en los cuales se levantan servidores de MongoDB que funcionan como base de datos locales y la comunicación entre dichos nodos y la inserción de los datos, se lleva a cabo a través del protocolo Blockchain de Tendermint.

Por la parte de la base de datos, aporta las características de MongoDB de: alta capacidad de inserción de la información, baja latencia y funciones de indexación y consulta. En cuanto a las funciones Blockchain, aporta la descentralización y distribución entre los nodos que forman la red, inmutabilidad e integralidad y tolerancia ante fallas bizantinas.

El sistema permite la creación de redes altamente personalizables en las que la creación e intercambio de activos están completamente a disposición de los nodos formadores, así como la decisión del perfil privado o público de dicha red.

Tiene una comunidad de desarrolladores muy activa, especialmente por su canal de Gitter.

2.4.6.6 Conclusiones

Tras un análisis de las DLTs propuestas, se determina que tanto Bitcoin como Ethereum carecen de utilidad en el sistema propuesto pues, la Bitcoin sirve únicamente como una reserva de valor u “oro digital” y, Ethereum, sirve principalmente como plataforma de desarrollo de otras aplicaciones Blockchain y para el establecimiento de contratos inteligentes.

Las opciones restantes son: Hyperledger, IOTA y BigchainDB. Entre ellas, se ha escogido a IOTA y BigchainDB, debido a que por sus características resultan idóneas para su aplicación y consecución de los objetivos propuestos en el presente proyecto.

Por un lado, IOTA se utilizará para la publicación del resumen estadístico temporal de los datos recogidos por el sensor conectado a la Raspberry Pi, pues las características de la Tangle, tanto en cuanto a su perfil público (que permite la visualización de los datos transmitidos con gran facilidad a partir de herramientas como www.thetangleexplorer.com), así como que la publicación de los datos sea gratuita e instantánea se ajustan perfectamente a la tecnología buscada.

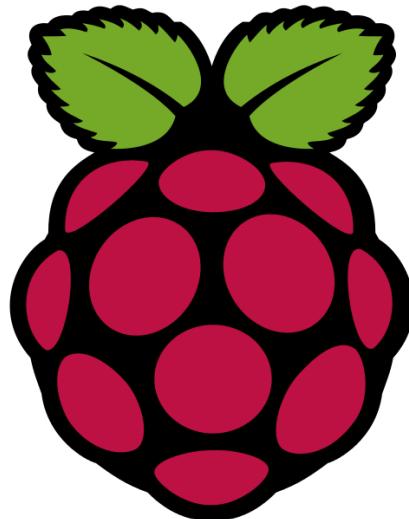
Por otro lado, BigchainDB y su naturaleza dual de base de datos y Blockchain, también se amolda perfectamente al objetivo de establecer un servidor descentralizado en el que registrar la información recogida, de manera que la misma quede escrita de forma íntegra e inmutable y sea fácilmente consultada.

Capítulo 3 Plataformas Utilizadas

3.1 Raspberry Pi 3 B:

3.1.1 ¿Qué es Raspberry Pi?

La Raspberry Pi es un computador de placa reducida o única de tipo SBC y bajo coste, desarrollado en Reino Unido por la Fundación Raspberry Pi. El objetivo original de la creación de este dispositivo fue el de estimular la enseñanza de ciencias de computación (CS) en las escuelas, acercando un sistema sencillo de usar y con un bajo consumo de recursos a la población más joven, para que se familiaricen desde edades tempranas con este tipo de tecnologías, tan importantes en los tiempos que corren.[15][16][17]



Aunque no se especifica de manera clara si es de hardware libre o con derechos de marca, la empresa da a entender que se trata de un producto con propiedad registrada, manteniendo el control de la plataforma, pero permitiendo su uso libre tanto a nivel particular como educacional.

Por otro lado, el software si es de código libre, siendo el sistema operativo oficial una versión adaptada de Debian, denominada Raspbian. Aun así, La computadora permite usar otros sistemas operativos como se comentará más adelante.

En mayo de 2009 se creó la Fundación Raspberry Pi en Caldecote, South Cambridge, Reino Unido como una asociación caritativa regulada por la comisión de caridad de Inglaterra y Gales. [18]

El lanzamiento oficial de la Raspberry Pi 1, con los modelos A y B, se realizó el 29 de febrero de 2012, convirtiéndose en un éxito de ventas desde el primer momento y teniendo una gran aceptación entre el público[19]. Y desde entonces, se ha beneficiado de diversas modificaciones y actualizaciones que, aun aumentando sus características técnicas, no han supuesto un incremento del precio del producto. Otro motivo por el cual la comunidad de usuarios y desarrolladores tienen en tan alta estima a la placa.

3.1.2 Modelos y Especificaciones Técnicas:

En la actualidad existen 3 modelos distintos, los modelos Raspberry Pi 3 B, B+ y una versión de características y precio aún más reducido que las anteriores denominada Raspberry Pi Zero W (Wireless).

El modelo B+ es la actualización más reciente del popular modelo B de la marca, y se puso a la venta en marzo de 2018. Como mejoras, presenta una CPU 200 MHz más rápida, manteniendo el consumo energético. Admite bandas de 5GHz en su conectividad Wifi y actualiza el sistema de bluetooth, siendo compatible con la versión BLE 4.2. Pero quizás la mejora más significativa se encuentre en el aumento de la velocidad máxima vía Ethernet, de 100 a 300 Mbps.

A continuación, se presenta una tabla en la que se comparan las características técnicas de los diferentes modelos:

Tabla 3.1: Comparación especificaciones técnicas distintos modelos de Raspberry

Características	Modelo B	Modelo B+	RPI Zero W
SoC	Broadcom BCM2837	Broadcom BCM2837Bo	Broadcom BCM2835
CPU	1.2GHz 64-bit quad-core ARMv8	1.4GHz 64-bit quad-core ARMv8	1GHz ARM11
GPU	Broadcom VideoCore IV 250MHz OpenGL ES 2.0		no
RAM	1 GB LPDDR SDRAM 450 MHz	1 GB LPDDR2 SDRAM 450 MHz	512 MB LDDPR 400 Mhz
USB 2.0	4	4	1 Micro USB
Wifi	802.11.n 2.4 GHz	802.11.n 2.4 & 5 GHz	802.11n 2.4 Ghz
Bluetooth	BLE 4.1	BLE 4.2	BLE 4.1
Salidas de vídeo		HDMI 1.4 @ 1920x1200 píxeles	
Almacenamiento		microSD	
Ethernet	10/100 Mbps	Sí, 10/300 Mbps + POE support	Sí, 10/100 Mbps
Consumo energético	800 mA (3.0W)	800 mA (3.0W)	150 mA
Tamaño	85,60x56,5 mm	85,60x56,5 mm	65x30x5 mm
Peso	45g	45g	8.8g

Las características, el precio y la disponibilidad en el momento del comienzo del presente trabajo llevaron a la utilización para la realización del mismo de la Raspberry Pi 3 modelo B. Cuyos componentes y funcionalidades serán detallados a continuación.

3.1.3 Raspberry Pi 3 Modelo B[20]

3.1.3.1 Hardware

Sacada a la luz en el año 2016. Con respecto a su predecesora, la Raspberry Pi 2 B, renueva procesador, una vez más de la compañía Broadcom, de nuevo un Quad-Core, pero pasando de 900MHz a 1.20GHz. Mantiene la RAM en 1GB. Su mayor novedad fue la inclusión de Wi-Fi y Bluetooth (4.1 Low Energy) sin necesidad de adaptadores.

Tabla 3.2: Elementos Raspberry

Nº	Identificador
1	Salida UDB 2.0
2	Pines GPIO
3	Conector DSI Display Module
4	Conector CSI Camera Module
5	Ranura para tarjeta Micro SD
6	Puerto Ethernet (rj45)
7	Alimentación micro USB
8	Leds Indicadores
9	Salida de audio
10	Salida HDMI

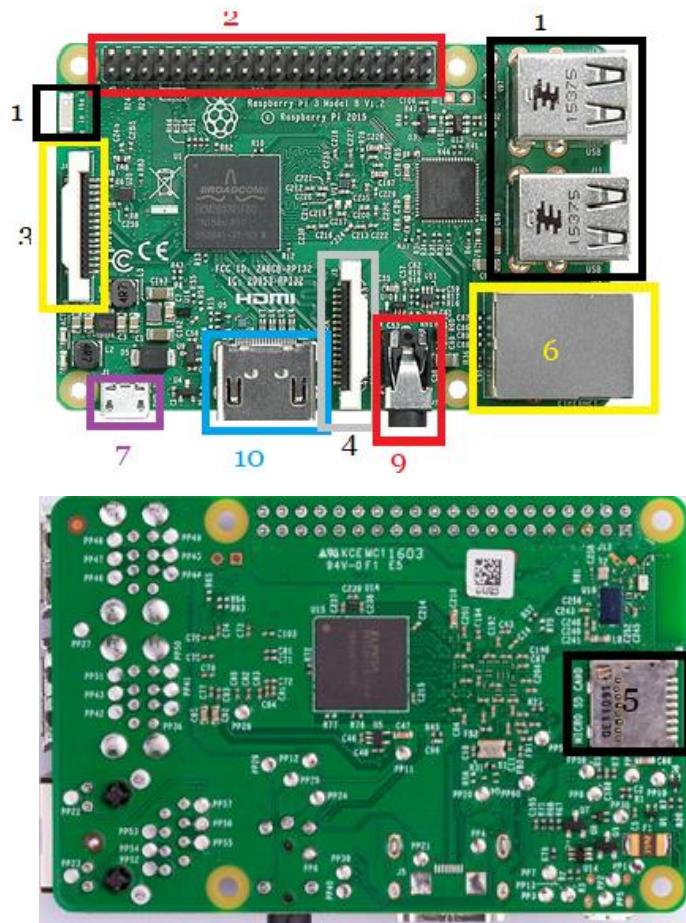


Figura 3.2: Raspberry Pi

- **Puertos USB:**

El modelo B posee cuatro puertos USB del estándar 2.0 gestionados por el microchip LAN9514. Este chip esta especialmente diseñado para casos como el de Raspberry Pi en el que se tiene que integrar puertos en una pequeña placa. El hecho de que en este modelo se haya aumentado la cantidad de puertos permite conectar a la vez ratón, teclado y conexión WIFI USB.

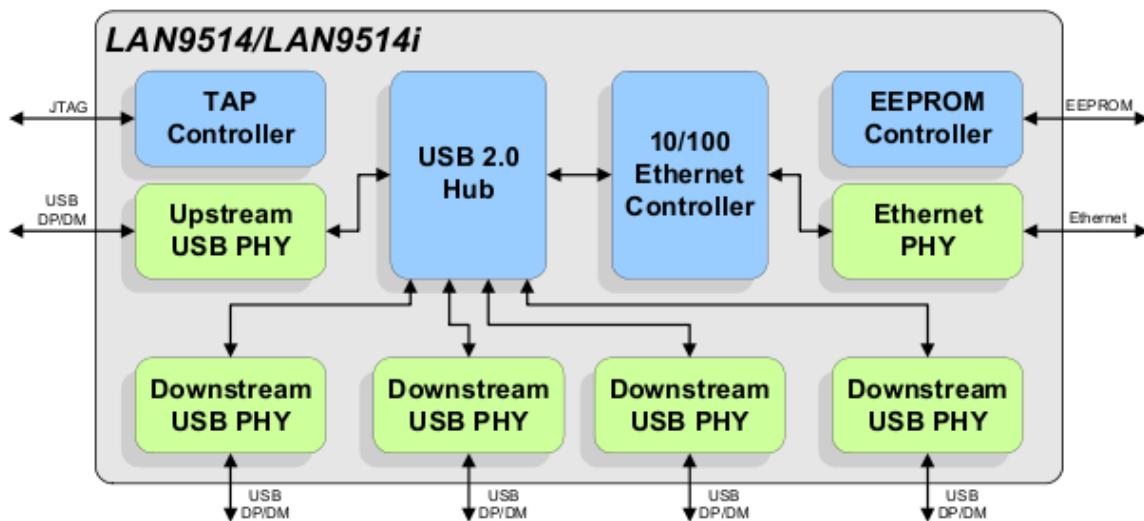


Figura 3.3: Puertos Raspberry Pi 3 modelo B

teclado y conexión WIFI USB.

- **Pines GPIO:**

Como se puede observar en la Figura 2-5 la Raspberry Pi B+ cuenta con dos filas de 20 pines cada una. Estos pines sirven de interfaz entre la Raspberry Pi y el exterior. GPIO (General Purpose Input/Output) es un sistema de Entrada/Salida de propósito general que te permite dar un uso más amplio y realizar multitud de proyectos.

Con estos pines se pueden controlar multitud de dispositivos como luces leds, motores o sensores.

De los 40 pines que contiene la Raspberry Pi, 12 de ellos son de alimentación (5V o 3.3V) o tierra y después hay tanto pines GPIO normales, que permiten una programación específica para el proyecto a realizas, como pines GPIO especiales para usar como puerto UART, I2C o SPI.

Raspberry Pi 3 Model B (J8 Header)		
GPIO#	NAME	GPIO#
	3.3 VDC Power	1
8	GPIO 8 SDA1 (I2C)	2
9	GPIO 9 SCL1 (I2C)	4
7	GPIO 7 GPCLK0	9
	Ground	
0	GPIO 0	8
2	GPIO 2	15
3	GPIO 3	16
	3.3 VDC Power	1
12	GPIO 12 MOSI (SPI)	21
13	GPIO 13 MISO (SPI)	22
14	GPIO 14 SCLK (SPI)	23
	Ground	24
30	SDA0 (I2C ID EEPROM)	25
21	GPIO 21 GPCLK1	26
22	GPIO 22 GPCLK2	27
23	GPIO 23 PWM1	28
24	GPIO 24 PCM_FS/PWM1	29
25	GPIO 25	
	Ground	
39	37	

Figura 3.4: Pines GPIO Raspberry Pi

- **Puerto Ethernet:**

Se dispone de un conector RJ-45 conectado al integrado LAN9514 de SMSC, nombrado en el apartado 2.1.1, que proporciona conectividad a 10/100 Mbps. Es posible conectar el dispositivo directamente a un PC sin pasar por un router conectando ambos equipos de manera directa con un cable RJ45.

- **Almacenamiento:**

La Raspberry Pi B no dispone de un disco duro, siendo ésta una de las causas fundamentales de su reducido precio. Para lidiar con este tema, trae un lector/ranura para memorias microSD, un sistema de almacenamiento en estado sólido. El arranque del sistema se hará desde la propia tarjeta microSD, donde se encuentra instalado el sistema operativo que el usuario haya escogido. Por este motivo, se precisa una capacidad mínima de dichas tarjetas SD de 2GB.

- **Alimentación:**

La placa no cuenta con botones de encendido/apagado. Para su alimentación, dispone de un conector micro USB tipo B que proporciona 5V de tensión.

- **Salidas de audio:**

Para la salida de audio posee un conector de audio Jack de 3,5mm, además del propio HDMI. Si se está usando el puerto HDMI de la Raspberry Pi, obtener el audio es sencillo: cuando está configurado apropiadamente, el puerto HDMI transporta ambas señales, la de video y la de audio. Si el display no tiene entrada HDMI se tendría que utilizar la salida de audio Jack.

- **Conecotor HDMI:**

Permite la conexión de un dispositivo compatible con la interfaz HDMI 1.3 y 1.4 para la extracción de vídeo y audio.

- **Broadcom BCM2837:**

El diseño incluye un System-on-a-chip Broadcom BCM2837, que contiene un procesador central (CPU) ARMv8 a 1.2 GHz, un procesador gráfico (GPU) VideoCore IV y 1 GB de memoria RAM todo en un mismo integrado.

3.1.3.2 Software:

La Raspberry Pi está diseñada para ejecutar el sistema operativo de código abierto GNU/Linux[20]. Actualmente existen numerosas versiones de Linux, conocidas como distribuciones, compatibles con el dispositivo, siendo algunas de ellas:

- Raspbian OS, la oficial y más extendida.
- RISC OS, uno de los pocos no basados en Linux.
- OpenELEC, diseñada para crear un centro multimedia barato con la Raspberry Pi.



Figura 3.5: Logo Ubuntu Mate

En el caso del presente proyecto se ha escogido el sistema operativo Ubuntu Mate, cuyas características se detallarán a continuación.

- **Ubuntu Mate[21][22]**

Ubuntu Mate es una distribución Linux ligera, multilenguaje, basada en Ubuntu, y específicamente desarrollada para sistemas con especificaciones técnicas bajas. Se

trata de un software libre y de código abierto. Está reconocida oficialmente por Canonical y es mantenida por la comunidad.

Fue fundado por Martin Wimpress y Alan Pope, y comenzó como un derivado no oficial de Ubuntu, usando como base a la versión 14.10 del mismo para su primer lanzamiento. Poco tiempo después se lanzó la versión 14.04 LTS. En febrero de 2015, fue reconocida por Canonical Ltd. Como sabor oficial de Ubuntu, coincidiendo con el lanzamiento de la versión 15.04 Beta 1[23].

La versión 16.04 LTS tiene imágenes específicamente desarrolladas para los dispositivos Raspberry Pi 2 y 3.

Actualmente, la última versión estable disponible es la 17.10, lanzada en octubre de 2017, y la que se utiliza para realizar el proyecto.

- **Requisitos:**

Tabla 3.3: Requisitos Ubuntu Mate

Requisitos Hardware	de Mínimos	Recomendados
Microprocesador	Pentium III 750 MHz	Core 2 DUO 1,6 GHz
Memoria RAM	1,5 GB	2 GB
Disco Duro (espacio libre)	8 GB	16 GB
Resolución	1024 x 768 o superior	1366 x 768 o superior

- **Características:**

Como se ha indicado antes, el núcleo es Linux y es de tipo monolítico, es decir, es una arquitectura de sistema operativo donde éste en su trabajo en su totalidad en espacio del núcleo, estando tan solo en modo supervisor. Se diferencia de otras arquitecturas, como micronúcleo o núcleo híbrido, en que solo define una interfaz de alto nivel sobre el hardware del ordenador.

Un conjunto primitivo de llamadas al sistema implementa todos los servicios propios del sistema operativo tales como: la planificación de procesos, gestión de la memoria o el sistema de archivos

. Este tipo de núcleos están programados de forma no modular y pueden tener un tamaño muy considerable, además cada vez que se añade una nueva funcionalidad el sistema debe ser recompilado y reiniciado. Todos los componentes del sistema tienen acceso a todas sus estructuras de datos internas y funcionalidades, por lo que un fallo podría propagarse por la totalidad del mismo[20].

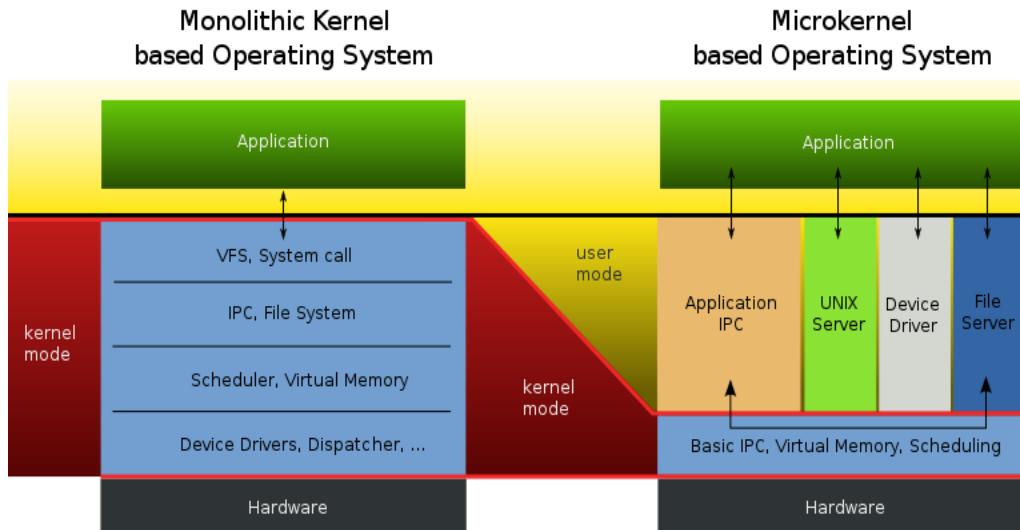


Figura 3.6: Diferencias sistemas kernel monolíticos y microkernel

Las plataformas soportadas por éste sistema operativo son x86, x86-64, PowerPC y ARMv7, siendo esta ultima la correspondiente a la Raspberry Pi 3 B.

El sistema de gestión de paquetes es de tipo **dpkg**, una herramienta de bajo nivel, por lo que se cuenta con un frontal de alto nivel para traer los paquetes desde lugares remotos o resolver conflictos complejos en las dependencias de paquetes. Esta herramienta es la *apt* o *aptitude*. Las herramientas que Debian debe llamar para construir un paquete son[24]:

- **dpkg-source** Empaque y desempaque los archivos fuentes de un paquete Debian.
- **dpkg-gencontrol** Lee la información de un árbol fuente Debian desempaquetado y genera un paquete binario de control, generando una entrada para éste en el fichero `debian/files`.
- **dpkg-shlibdeps** Calcula las dependencias de ejecutables respecto a bibliotecas.
- **dpkg-genchanges** Lee la información de un árbol fuente Debian desempaquetado y ya construido, generando un fichero de control de los últimos cambios (`un.changes`).

- **dpkg-buildpackage** Es un script de control que se puede utilizar para automatizar la construcción del paquete.
- **dpkg-distaddfile** Añade una entrada de un fichero a debian/files.
- **dpkg-parsechangelog** Lee el fichero de cambios (*changelog*) de un árbol fuente Debian desempaquetado y genera una salida con la información de estos cambios, convenientemente preparada.

3.1.4 Puesta en Marcha del Sistema:

3.1.4.1 Instalación del Software:

Uno de los pilares fundamentales que sustentan a Raspberry Pi y a todo su entorno, es la flexibilidad y personalización. Por ello el dispositivo según sale de la caja viene sin fuente de almacenamiento y dado que es ahí donde se encontrará la imagen del sistema operativo que instalará, será elección del usuario el sabor concreto que desee implementar y usar.

En este caso se ha elegido Ubuntu Mate como software debido a la fluidez de su uso y a las herramientas ofimáticas y de programación que trae por defecto.

- **Requisitos para instalar el sistema operativo:**

- Tarjeta SD de 8 GB mínimo.
- Adaptador de microSD a USB.
- Fichero .img del sistema operativo.

- **Descarga del sistema operativo:**

Se necesitará otro computador, donde se descargará la imagen del sistema operativo que se desea instalar. Para ello se accederá a la página oficial de Ubuntu Mate y en la sección Download, se seleccionará la versión para la Raspberry Pi. En este caso, se seleccionará la versión 16.04 LTS (Xenial)



Download

Choose your architecture

The page displays four options for choosing architecture:

- 64-bit**: Ideal for computers with:
 - More than 3 GB of RAM.
 - 64-bit capable Intel and AMD processors.
 - UEFI PCs booting in CSM mode.
 - Modern Intel-based Apple Macs.
- 32-bit**: Ideal for computers with:
 - Less than 2 GB of RAM.
 - Intel and AMD processors.
 - Ageing PCs with low-RAM resources.
 - Older Intel-based Apple Macintosh systems.
- PowerPC**: For hardware like:
 - Apple Macintosh G3, G4 and G5
 - Apple iBooks and PowerBooks
 - IBM OpenPower 7xx Machines
- Raspberry Pi**: For aarch32 (ARMv7) computers, like:
 - Raspberry Pi 2
 - Raspberry Pi 3



Download

Which release would you like?

for a Raspberry Pi system

The page shows a list of releases for a Raspberry Pi system:

- 16.04.2 (Xenial)**: Bring the traditional desktop experience to your Raspberry Pi!
Supported until April 2019

Choose a different architecture



Figura 3.7: Descarga de Ubuntu Mate

• Instalacion del sistema operativo:

Para realizar este paso es necesario:

- Descargar en primer lugar el programa Win32DiskImager que se encargará de escribir la imagen descargada en la tarjeta SD.
- Una vez descargado e instalado, se abre y se selecciona la unidad donde se tiene pinchada la tarjeta SD.
- Se selecciona la imagen a escribir.

- Se pulsa el botón escribir.

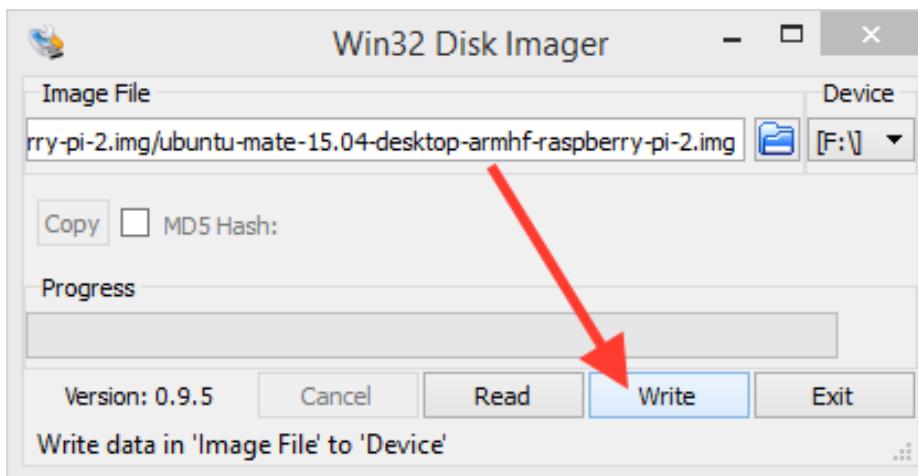


Figura 3.8: Instalación Ubuntu Mate

- Una vez escrita la imagen del sistema operativo, se inserta la tarjeta SD en la Raspberry y ya queda lista para ser usada y configurada.

3.2 Sensor Sense HAT:

3.2.1 ¿Qué es el sensor HAT?:

Es una tarjeta add-on o modulo adicional, diseñada específicamente para la Raspberry Pi y una de las piedras angulares del proyecto a realizar.

Originalmente desarrollada para la misión Astro Pi lanzado a la Estación Espacial Internacional en 2015. El objetivo de dicha misión es el de realizar una competición orientada a los jóvenes para escribir aplicaciones sobre un sistema Raspberry Pi con una cámara integrada y este sensor HAT. [25]

El dispositivo permite recopilar multitud de medidas con un bajo coste. La tarjeta cuenta con un panel LED RGB de 8 x 8, un joystick de 5 botones (arriba, abajo, izquierda, derecha y medio) e incluye los siguientes sensores:

- Temperatura de 16 bits de resolución.
- Humedad de 16 bits de resolución.
- Presión barométrica de 24 bits de resolución.

- Magnetómetro, acelerómetro, y giroscopio de 16 bits de resolución y 9 grados de libertad.

Como apunte, la temperatura se puede recoger tanto del sensor de humedad como el de presión. Los señores pueden tomar muestras de forma periódica según esquema de cola FIFO (First In First Out), pudiendo tomar valores en algunos señores con frecuencia máxima de 25 muestras por segundo. En particular el magnetómetro, acelerómetro, y giroscopio pueden recopilar hasta 952 muestras por segundo, haciendo de este un complemento ideal para estabilizar el vuelo de un dron por ejemplo.[26]

La conexión a la Raspberry Pi se realiza a través de los 40 pines GPIO y los sensores se conectan a través de I2c. La programación de la placa se hace mediante la interfaz SPI de la Raspberry, por lo que podemos reprogramarla o agregarle firmware.

Las diferentes partes se recogen en la siguiente imagen y tabla

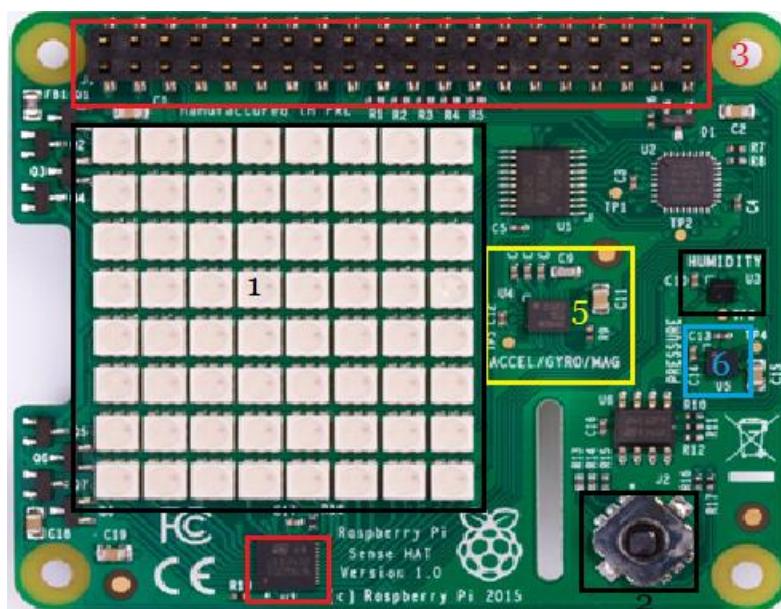


Tabla 3.4: Elementos SenseHat

Nº	Identificador
1	Panel Led
2	Joystick
3	Pines de conexión
4	Sensor de humedad
5	Sensor de presión
6	Sensor de aceleraciones, giroscopio y magnetómetro

Figura 3.9: Sensor SenseHat

Sus reducidas, dimensiones, peso, coste y el conjunto de todos sus sensores y actuadores hacen de éste el dispositivo ideal para capturar datos ambientales. Todo ello, unido a la completa librería en Python disponible, llamada SenseHat dotan a su combinación con la Raspberry Pi de una gran versatilidad y facilitan el desarrollo de multitud de proyectos distintos a poder realizar.

3.2.2 Especificaciones Técnicas:

Tabla 3.5: Especificaciones técnicas SenseHat

Módulo	Especificaciones
Giroscopio	Sensor de ratio angular: ±245/500/2000dps
Acelerómetro	Sensor de aceleración lineal: ±2/4/8/16 g
Magnetómetro	Sensor magnético: ±4/8/12/16 gauss
Barómetro	260 – 1260 hPa (la precisión depende de la temperatura y la presión, ±0.1 hPa en condiciones normales)
Sensor de temperatura	Precisión en la temperatura de ± 2 °C en el rango de 0-65 °C
Sensor de humedad	Precisión de ±4.5% en el rango de 20-80%r H, precisión de ± 0.5 °C en el rango de 15- 40 °C)
Matriz de Leds	8x8 elementos con 15 bits de resolución

3.3 Protocolo MQTT

3.3.1 Introducción

MQTT son las siglas correspondientes a Message Queue Telemetry Transport y es un protocolo de comunicación entre máquinas o Machine to Machine (M2M). Desarrollado en 1999 en una colaboración entre IBM y Eurotech (La actual Cirrus Link), se trata de un protocolo extremadamente ligero, con un ancho de banda eficiente y que consume muy pocos recursos. Características que lo convierte en un protocolo de comunicación ideal para dispositivos del IIoT.

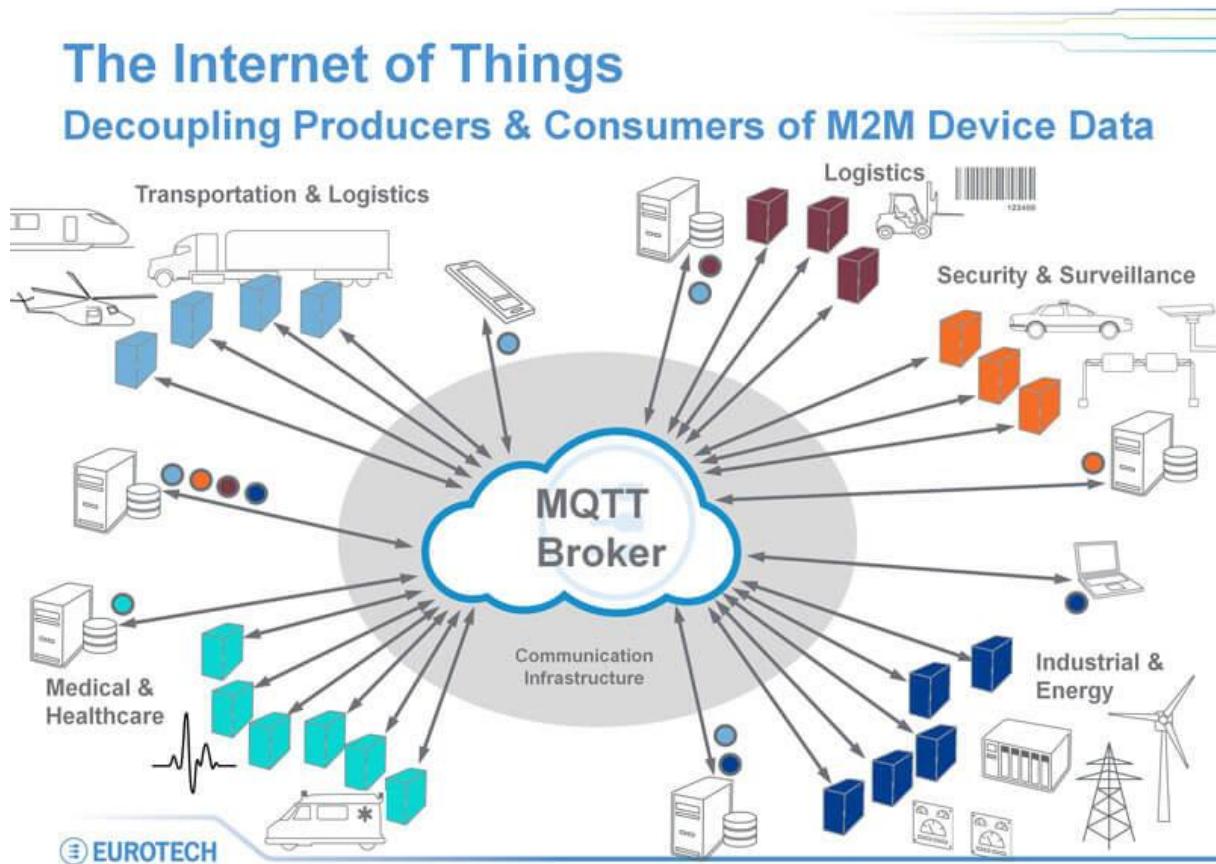


Figura 3.10: Arquitectura de MQTT cortesía de Eurotech

3.3.2 El protocolo

El protocolo utiliza una arquitectura de intercambio de mensajes publicación/subscripción sobre TCP/IP, claramente diferenciado de otras arquitecturas como la de HTTP, basadas en solicitud/respuesta. Esta arquitectura publicación/subscripción, está basada en eventos y consta de un servidor centralizado de comunicación o *bróker*, desde la que se gestionan los paquetes intercambiados.[6]

La comunicación está basada en *topics* o temas. Los clientes que publican mensajes lo hacen especificando un topic concreto, y los nodos que deseen recibir dichos mensajes, han de suscribirse a dicho topic. Este topic es la información de enrutamiento para el bróker, y constituye el canal de comunicación entre emisores y receptores de los mensajes. Cada cliente que desee recibir mensajes ha de suscribirse a un cierto topic.

El esquema para el envío y recepción de paquetes considerando el código y el tipo de paquete se refleja en la siguiente tabla.

Tabla 3.6: Paquetes MQTT [6]

Paquete	Enumeración	Descripción
Reservado	0	Reservado
CONNECT	1	Cliente solicita conectarse al servidor
CONNACK	2	Confirmación de mensaje CONNECT
PUBLISH	3	Mensaje PUBLISH
PUBACK	4	Confirmación del paquete PUBLISH
PUBREC	5	PUBLISH recibido ()
PUBREL	6	PUBLISH reléase ()
PUBCOMP	7	PUBLISH completado
SUBSCRIBE	8	Petición del cliente para suscribirse a un topic
SUBACK	9	Confirmación del paquete SUBSCRIBE
UNSUBSCRIBE	10	Petición del cliente para desuscribirse
UNSUBACK	11	Confirmación de UNSUBSCRIBE
PINGREQ	12	PING
PINGRESP	13	Respuesta de PING
DISCONNECT	14	Cliente va a desconectarse
Reservado	15	Reservado

La conexión a nivel de red se realiza mediante el envío de los paquetes CONNECT y CONNACK. Una vez satisfecha esta condición, el cliente manda un paquete SUBSCRIBE con el que se subscribe a un topic determinado. La suscripción se confirma a través del envío de un paquete SUBPACK, y una vez confirmada, el cliente puede empezar a enviar mensajes. El flujo de datos se refleja en la siguiente figura:

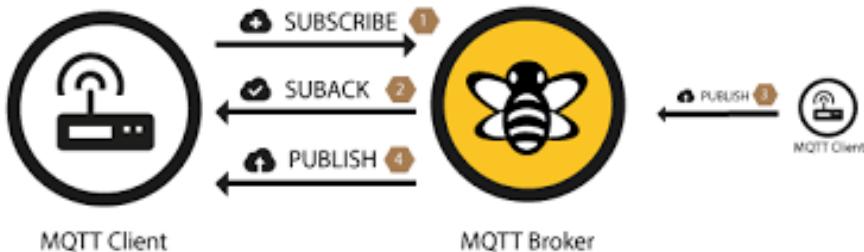


Figura 3.11: Flujo de intercambio de paquetes entre bróker y cliente en MQTT

Una vez realizada satisfactoriamente la conexión, se devolverá un código que reflejará el nivel de servicio especificado en el topic. Los códigos de respuesta son los siguientes:

Return Code	Return Code Response
0	Success - Maximum QoS 0
1	Success – Maximum QoS 1
2	Success – Maximum QoS 2
128	Failure

Figura 3.12: Códigos de respuesta de conexión en MQTT

En caso de que el cliente desee desuscribirse de un topic, enviará el paquete UNSUBSCRIBE y deberá esperar la confirmación del bróker UNSUBACK.

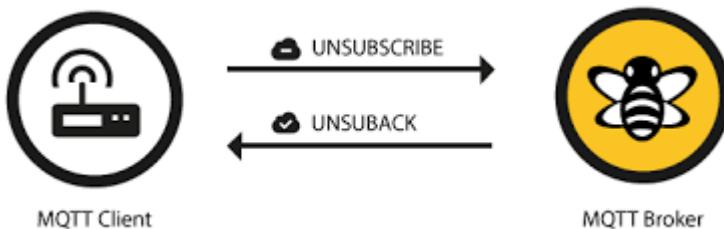


Figura 3.13: Flujo de datos de desuscripción en MQTT

3.3.3 Brokers

Son los servidores de MQTT, recibe los mensajes de los clientes y se encargan de gestionar la red y los envíos a los nodos suscriptores al topic correspondiente. Se verifica periódicamente la conexión a través del canal mediante el envío de paquetes PINGREQ, que devuelven una respuesta PINGRESP.

Actualmente, existen numerosos brokers MQTT, tanto comerciales (HiveMQ o CloudMQTT) como de código abierto (Mosquitto o Mosca)[6]. En el proyecto se trabajará con Mosquitto, al ser un software libre, gratuito, muy completo y con el respaldo de una gran comunidad. A continuación, se presenta una tabla en la que se

	QoS 0	QoS 1	QoS 2	Bridge	SSL	Cluster	Websockets	Plugin System
mosquitto	✓	✓	✓	✓	✓	✗	✗	✓
Mosca	✓	✓	✗	?	?	?	✓	✗
RabbitMQ	✓	✓	✗	✗	✓	?	?	?
HiveMQ	✓	✓	✓	✓	✓	✓	✓	✓
ActiveMQ	✓	✓	✓	?	?	?	✓	?
RSMB	✓	✓	✓	✓	✗	✗	✗	?
moquette	✓	✓	✗	?	?	✗	✗	✗

refleja una comparación entre las características de los brokers más utilizados:

3.3.4 Clientes

Son los encargados de enviar mensajes al bróker y recibir mensajes de los topics a los que se han suscrito previamente. Del mismo modo que en el caso de los brokers, existen numerosos tipos de clientes MQTT distintos, tanto comerciales como de código abierto, y para una amplia variedad de lenguajes de programación. Uno de los clientes de código libre más utilizados, y con el que se trabajara en el presente proyecto, es Eclipse Paho Python Client.

3.3.5 Requisitos

Dada la naturaleza middleware de IoT del protocolo MQTT, hay una serie de requisitos que deben ser cumplimentados para su implementación. Para la realización del

proyecto, algunos de los que se comentan a continuación son indispensables, mientras que otros no lo son. [6]

3.3.5.1 Requisitos Funcionales

- Control de recursos**

Se realiza un control básico a nivel de red, sobre las conexiones y desconexiones de los clientes. A través del temporizador “Keep alive” en los envíos de PINGREQ, el bróker es capaz de detectar si se ha producido una desconexión inesperada de un cliente y, en ese caso, publicará un mensaje llamado Last Will Testament o LWT a todos los clientes suscritos al topic determinado. Cada cliente, al conectarse con el bróker, pueden definir el mensaje LTW y hacer una petición de almacenamiento al bróker.

- Control de Datos**

El bróker lleva a cabo un control de datos como el almacenamiento y el filtrado. En caso de que un cliente se encuentre offline, el bróker puede almacenar los paquetes del topic al que esté suscrito dicho cliente y, una vez que se retome la conexión, enviarlos. Además, también se puede aplicar un filtrado en los topics, de manera que se pueden aplicar niveles de prioridad a los mismos.

- Single Level +

Resulta más sencillo explicar este nivel con un ejemplo:

Un bróker publica mensajes a través de los topics:

myoffice/meetingsroom/+/temperature

Y los clientes suscritos a los topics:

myoffice/meetingsroom/firstfloor/temperature

myoffice/meetingsroom/secondfloor/temperature

Recibirán dichos mensajes, al contrario que el cliente suscrito a:

Myoffice/meetingsroom/firstfloor/humidity

- Multi Level: #

Se denota de la forma:

Myoffice/meetingsroom/#

Los mensajes en este caso llegarán a cualquier cliente suscrito a partir del nivel *meetingsroom*.

3.3.5.2 Requisitos no funcionales

- **Escalabilidad**

MQTT tiene una gran escalabilidad, que dependerá en gran parte de la calidad de los nodos que se establezcan y la topología utilizada. La configuración de uno o dos brokers permite al sistema tener una expansión horizontal significativa, dada la facilidad de adhesión de nuevos clientes, se puede construir una gran red de comunicación IIoT sin demasiados recursos.

- **Seguridad y Privacidad**

La seguridad y la privacidad son temas cruciales en los ecosistemas IIoT y, MQTT los gestiona a través de comunicación TLS con autentificación de usuario/contraseña. Además, existen métodos de denegación de determinados topics o de restricciones de operaciones.

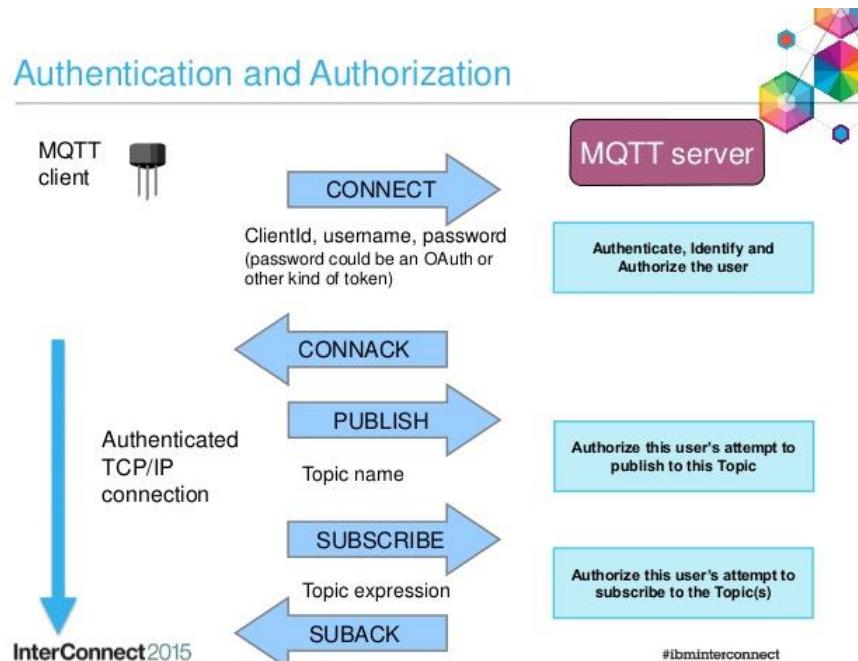


Figura 3.14: Autenticación y autorización en MQTT

- **Disponibilidad**

Al basarse en una arquitectura centralizada en torno al bróker, el sistema es fundamentalmente dependiente del mismo, y su disponibilidad es muy limitada, ya que está directamente ligada a la disponibilidad del bróker.

Existen herramientas como el MQTT clustering o los balanceadores de carga que permiten el uso de la red en caso de fallo de algunos de los brokers, y que, por tanto, consiguen aumentar la disponibilidad y robustez del sistema.

- **Confiabilidad**

En MQTT, la confiabilidad viene determinada por el Quality of Service o QoS, y existen 3 niveles distintos para el intercambio de paquetes:

- 1) **QoS 0: Como mucho una vez**

Se envía el mensaje PUBLISH y el bróker no lanza ningún reconocimiento. El mensaje llega al bróker una o ninguna vez, y en caso de llegar, tan solo será recibido por los subscriptores una o ninguna vez.

- 2) **QoS 1: Al menos una vez**

Se envía el mensaje y el nivel de calidad asegura que llega al bróker al menos una vez. Este reconocimiento tiene lugar al enviar el bróker un mensaje PUBACK en el que reconoce la recepción del paquete. En caso de que no se reciba el PUBACK, se reenvía el primer mensaje con identificadores distintos, hasta que se reciba el reconocimiento. Existe la posibilidad de recibir paquetes duplicados.

- 3) **QoS 2: Exactamente una vez**

Se asegura que el paquete llega exactamente una vez. A través de los paquetes PUBREC y PUBREL.

Un resumen de los diferentes niveles de servicio:

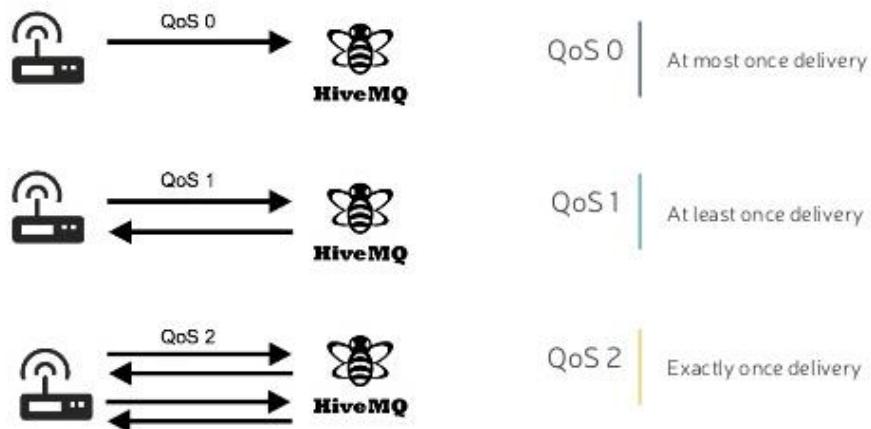


Figura 3.15: Quality of Service en MQTT

- **Facilidad de Implementación, Popularidad y Comunidad**

MQTT tiene una gran comunidad de desarrolladores trabajando con él y desarrollando aplicaciones sobre el mismo. Esto hace que exista una gran cantidad de recursos disponibles para aprender a construir redes basadas en MQTT de manera sencilla, escalable y segura.

3.3.5.3 Requisitos Arquitectónicos

- **Ligereza**

Una de las claves de la extensión en el uso de MQTT como protocolo de comunicación en redes IoT, es su bajo consumo de procesamiento y de ancho de banda. Estos resultan tan pequeños, que es resulta muy sencillo de implementar en prácticamente cualquier red de dispositivos-sensores.

- **Interoperabilidad**

Gracias a la poca capacidad de procesamiento que requiere, resulta un protocolo extremadamente versátil y heterogéneo, capaz de funcionar a través de numerosos dispositivos con características y propiedades muy distintas. Funciona en placas tipo Raspberry Pi, Arduino, en ordenadores personales e incluso en *smartphones*.

3.4 BigchainDB



Figura 3.16: Logo BigchainDB

En esta sección se introducirá la tecnología Blockchain BigchainDB, se hará una descripción de su arquitectura, funcionamiento y se detallará por qué su aplicación como base de datos descentralizada y distribuida resulta ideal en el ecosistema IIoT.

3.4.1 Introducción

A grandes rasgos BigchainDB es un software de código libre que posee características tanto de una Blockchain como de una base de datos[27]. Lanzado por primera vez en 2016, actualmente se encuentra en la versión 2.0 que incorpora substanciales avances respecto de las anteriores versiones como la tolerancia a fallas Bizantinas (*Bizantine Tolerance Fault o BFT* en inglés), gracias a la cual, aunque fallen hasta un tercio de los nodos, el resto del sistema continuará siendo capaz de llegar a un consenso y funcionar con normalidad.

Es importante reseñar que el sistema está concebido para levantar redes Blockchain entre una serie de partes previamente especificadas y que se conocen y confían entre sí. El proceso sería el siguiente, una entidad coordinadora levanta el primer nodo y es el que configura la adhesión del resto de nodos, conociendo previamente las claves públicas de dichos nodos. Esto, hace que el sistema sea idóneo para el desarrollo de aplicaciones industriales, para la creación y transferencia de gemelos digitales, e incluso para crear propios tokens con los que hacer transacciones.

3.4.2 Diseño de BigchainDB 2.0

Anunciado por primera vez en febrero de 2016, BigchainDB se autodenomina desde el primer momento como una base datos con propiedades de Blockchain, y buscó la convergencia de ambas naturalezas de una manera sinérgica. Por el lado de Blockchain el sistema sería descentralizado, inmutable y con activos únicamente controlados por los propietarios declarados, además heredaría las características propias de una base

datos como baja latencia, alta capacidad de transacción y herramientas de consulta e indexación de los datos estructurados[27].

Las primeras versiones funcionaban, pero tenían una serie de vulnerabilidades que era imprescindible solventar. El sistema era incapaz de soportar fallos arbitrarios, no era tolerante a fallas Bizantinas, y era susceptible de ser completamente gestionada por el nodo primario y, en caso de ser un agente “malicioso”, podría borrar la base de datos entera con un solo comando.

Estas vulnerabilidades quedan solucionadas con la nueva versión 2.0. En la siguiente tabla se reflejan sus características generales y una comparación con otros sistemas de almacenamiento de información.

Tabla 3.8: Comparativa BigchainDB con bases de datos y otras Blockchain[27]

	Blockchain típica	Base de datos distribuida típica	BigchainDB
Descentralización	✓		✓
Tolerancia a fallas Bizantinas	✓		✓
Inmutabilidad	✓		✓
Activos controlados por los propietarios	✓		✓
Alta capacidad de transacción		✓	✓
Baja latencia		✓	✓
Indexación y Consulta		✓	✓

- **Descentralización completa y Tolerancia a Fallas Bizantinas**

BigchainDB 2.0 utiliza Tendermint para realizar las transacciones y alcanzar el consenso. Cada nodo posee una base de datos local MongoDB y la comunicación entre los nodos se hace a través del protocolo Tendermint, el cual es BFT, por lo que el sistema resulta ser BFT a su vez. Otra consecuencia es que, si un agente “maligno” consigue privilegios de administrados de uno de los nodos locales MongoDB, en el peor

de los casos tan solo podrá corromper o borrar la información registrada en esa base de datos local, y los demás nodos no se verán afectados.

El sistema se trata de una red descentralizada ya que, al estar cada nodo controlado y ser operado por una entidad diferente, no tiene un único propietario, único punto de control y único punto de fallo.

- **Inmutabilidad**

Una vez que la información se registra en el sistema BigchainDB, no puede ser modificada o eliminada, o al menos sin gran dificultad. Lo interesante es que, si se diera ese caso, sería fácilmente detectable. Para alcanzar esta quasi-inmutabilidad, el sistema utiliza diferentes estrategias:

- No existen APIs para modificar o eliminar la información almacenada.
- Todas las transacciones se firman criptográficamente. Una vez registradas, cualquier cambio realizado modificará a su vez la firma lo cual es fácilmente detectable (a no ser que la clave pública también cambie, pero eso también sería detectable dado que cada bloque de transacciones está firmado por el nodo almacenador, y todas las claves públicas de los nodos que forman la red se conocen).

- **Activos Controlados por los Propietarios Declarados**

Como en la mayoría de los sistemas Blockchain, en BigchainDB tan solo el propietario (o propietarios) de un activo puede transferir dicho activo. La principal diferencia es que, a diferencia de otras Blockchain “tradicionales” en las que los únicos activos disponibles son los tokens que proporciona la red (como en Bitcoin o Ethereum), en BigchainDB se permite la creación de tantos activos como se desee o necesite. Por ejemplo, el usuario Juan puede realizar una transacción del tipo CREATE con la que creará 1000 “Juan tokens”, para ello, la firmará con su clave privada y la enviará a la red BigchainDB. Una vez validada la transacción, Juan podrá transferir hasta 1000 Juan tokens. Si decidiera enviar 200 al usuario Laura, realizaría otra transacción, esta vez del tipo TRANSFER, que tendría dos *outputs* o salidas, la primera de 200 tokens con la condición de que tan solo Laura podrá transferirlas, y la segunda con la condición de que los 800 token restantes solo podrán ser transferidos por Juan.

La red comprueba cada transacción para asegurarse que no se está tratando de transferir una salida previamente transferida (el famoso problema del “doble gasto” al que ya se ha hecho referencia previamente en el documento). También se comprueban otros aspectos, que se especificarán cuando se explique con más detalle el proceso de las transacciones.

- **Alta Capacidad de Transacción y Baja latencia**

El protocolo de comunicación entre nodos Tendermint proporciona un desempeño excepcional en este sentido. En pruebas realizadas con 64 nodos distribuidos a través de 7 centros de datos en 5 continentes, el consenso Tendermint puede procesar miles de transacciones por segundo, con una latencia del orden de 1-2 segundos[27]. Esta capacidad de transacciones se mantiene incluso en condiciones adversas como de colapso de los agentes validadores o existencia de confirmaciones de agentes maliciosos.

- **Indexación y Consulta De la Información Estructurada**

Al poseer cada nodo de BigchainDB una base de datos local MongoDB, el operador de dicho nodo tiene acceso a todo el potencial de consulta e indexación de la información registrada (transacciones, activos, metadatos, todos ellos cadenas JSON), características de MongoDB. Además, cada operador puede determinar cuánto de este potencial pone a disposición de usuarios externos que deseen acceder a los registros.

Por defecto, BigchainDB crea ciertos índices MongoDB y la API HTTP incluye ciertos *endpoints* para realizar consultas básicas.

- **Resistencia a ataques Sybil**

En algunas redes Blockchain como Bitcoin, cualquiera puede añadir su nodo al sistema. Ello puede repercutir en que algún agente incluya tantos nodos que llegue a hacerse en control de toda la red, lo que se conoce como un ataque Sybil. Bitcoin se protege ante ello haciendo este tipo de ataque prohibitivamente costosos. En BigchainDB, la organización gobernadora controla la lista de miembros, lo que hace que el ataque Sybil no sea un problema.

3.4.3 Campos de Aplicación de BigchainDB

A partir de las características y funcionalidades previamente comentadas, se deduce que BigchainDB puede servir en multitud de casos de uso. Donde sea que se necesite información que sea inmutable y que represente a activos digitales, BigchainDB puede usarse. En concreto, hay una serie sectores que pueden beneficiarse directamente de las propiedades de BigchainDB, como, por ejemplo:

- **Cadenas de suministro**

Comúnmente en las cadenas de suministro existen diferentes entidades que colaboran e intercambian información entre sí. En principio dicha información es concerniente en cuanto a los diferentes procesos a los que se ven sometidos los productos producidos y a llevar un seguimiento de estos a lo largo de toda la cadena, hasta el final de esta (minoristas/consumidores).

Actualmente, el mayor reto al que se enfrentan las cadenas de suministro es la gestión y la protección de la información que se comparte. Es en estos ámbitos en los que las tecnologías Blockchain y DLTs cobran interés y pueden ayudar a organizar la información en un ecosistema compartido de manera que se facilite la gestión de esta. Además, la inmutabilidad que proporcionan estas redes proporciona una capa de confianza entre las entidades colaboradoras, de manera que, aunque no confíen entre sí, podrán confiar en la integridad de la información registrada.

Donde BigchainDB se diferencia con respecto a otras Blockchain en este contexto, es en sus características de base de datos de alta capacidad de transacción y su gran potencial de consulta e indexación.

- **Gemelos Digitales e IoT**

Los gemelos digitales son representaciones digitales de objetos físicos que pueden ser rastreados en cuanto a la verificación de autenticidad y proveniencia, prueba de propiedad, trazabilidad de ciclos de vida, e inputs de dispositivos y sensores IoT. Para gestionar toda esta escala de información, es útil otorgar a cada producto u objeto una historia e identidad propia. BigchainDB proporciona el entorno y las herramientas idóneas para llevar a cabo estas labores.

- **Identidad**

La identidad es uno de los aspectos más críticos cuando nos referimos a la información específica de usuario, más aún en el contexto del IIoT y los gemelos digitales, donde cada máquina, dispositivo y sensor tiene una identidad propia. Actualmente, el robo de la identidad de este tipo de objetos es una de las principales preocupaciones en este contexto y es imprescindible asegurarse de que la identidad de un humano o una máquina es autogobernada y a prueba de hackeo.

3.4.4 Funcionamiento de BigchainDB

Se puede obtener una comprensión bastante cercana del funcionamiento de BigchainDB a partir de seguir el proceso de una transacción. A continuación, se ilustran los componentes principales de una red de cuatro nodos BigchainDB y como se comunican entre sí.

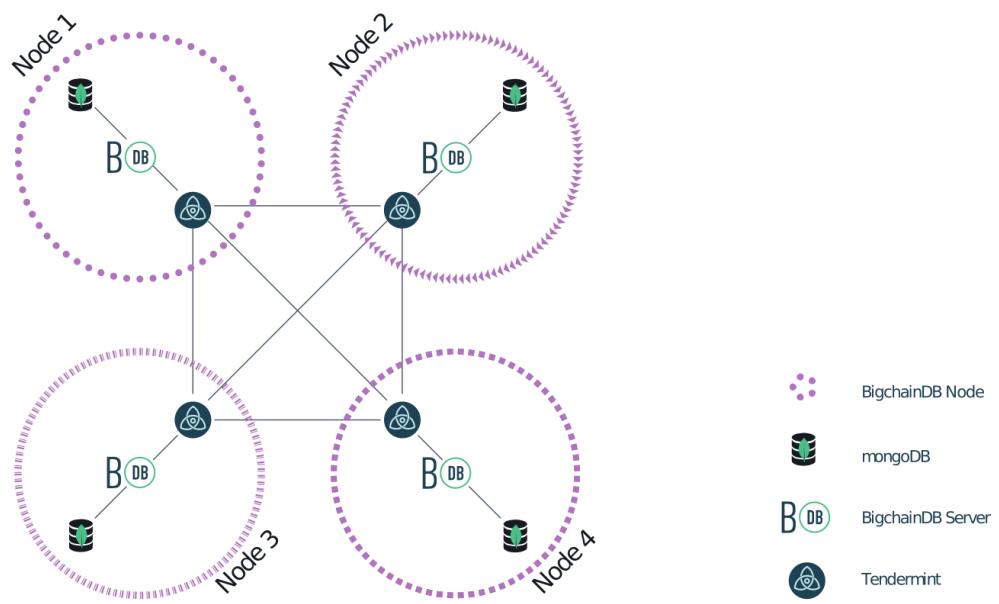


Figura 3.17: Red de 4 nodos de BigchainDB

3.4.5 Transacciones en BigchainDB

Una transacción en BigchainDB es una cadena JSON que cumple una serie de especificaciones concretas, que están claramente detalladas en la documentación de BigchainDB. Se utilizan para registrar, crear o transferir activos y hay dos tipos básicos: transacciones de tipo CREATE y de tipo TRANSFER. En particular, una transacción tiene la siguiente forma:

```

{
  "id": "3667c0e5cbf1fd3398e375dc24f47206cc52d53d771ac68ce14d_"
  → df0fde806a1c",
  "version": "2.0",
  "inputs": [
    {
      "fulfillment": "pGSAIEGwaKW1LibaZXx7_NZ5-V0alDLvrguGLyL_"
      → RkgmKWG73gUBJ2Wpnab0Y-4i-kSGFa_VxxYCcctpT8D6s4uTG0O_"
      → F-hVR2VbbxS35NiDrwUJXYCHSH2IALYUoUZ6529Qbe2g4G",
      "fulfills": null,
      "owners_before": [
        "5RRWzmZBKPM84o63dppAttCpXC3wqYqL5niwNS1XBFyY"
      ]
    }
  ],
  "outputs": [
    {
      "amount": "1",
      "condition": {
        "details": {
          "public_key": "
          → "5RRWzmZBKPM84o63dppAttCpXC3wqYqL5niwNS1XBFyY",
          "type": "ed25519-sha-256"
        },
        "uri": "ni:///sha-256;d-_huQ-eG-QQD-GAJpvrsSy71LJqyNh_"
        → tUAs_own7aTY?ftp=ed25519-sha-256&cost=131072"
      },
      "public_keys": [
        "5RRWzmZBKPM84o63dppAttCpXC3wqYqL5niwNS1XBFyY"
      ]
    }
  ],
  "operation": "CREATE",
  "asset": {
    "data": {
      "message": "Greetings from Berlin!"
    }
  },
  "metadata": null
}

```

Figura 3.18: Transacción en BigchainDB

- **Transacciones CREATE:**

Las transacciones de tipo CREATE se puede usar para registrar, crear o iniciar la historia de un objeto único (o activo) en BigchainDB. Por ejemplo, se puede registrar una identidad o un trabajo creativo, o un bien de cualquier tipo. Los objetos registrados se denominan activos.

Estos activos pueden tener una propiedad compartida entre varios usuarios, lo que implicaría que podrían ser transferidos por todos los propietarios. Para ello, debe estar firmada por todos los propietarios.

- **Transacciones TRANSFER:**

Las transacciones de tipo TRANSFER pueden activos previamente creados y deben tener inputs que apunten a transacciones tipo CREATE previas. Estos outputs deben estar asociados al mismo activo y solo se pueden transferir partes de un activo cada vez.

3.4.5.1 Componentes de una Transacción

Una transacción en BigchainDB se implementa como un vector asociativo en casi cualquier lenguaje de programación, en el caso de Python, correspondería a la estructura de un diccionario clave-valor.

Una transacción tiene la siguiente estructura básica:

```
{  
    "id": id,  
    "version": version,  
    "inputs": inputs,  
    "outputs": outputs,  
    "operation": operation,  
    "asset": asset,  
    "metadata": metadata  
}
```

- **ID de la transacción:**

Corresponde al hash SHA-256 de la transacción y es una cadena que adopta una forma similar a:

"oe7a9717ad8966584b40148630ec412c6bffdbe8d788a9047fdf39eb5ee03518"

- **Inputs:**

Conceptualmente, un input es un puntero a un output de una transacción previa. Señala a quien pertenecía anteriormente el activo en cuestión y proporciona prueba de que se cumplen las condiciones necesarias para poder transferir la propiedad del activo.

Una transacción de tipo CREATE debe tener exactamente un input, que especifica quien es el usuario que está registrando el activo. En una transacción de tipo TRANSFER ha de tener como mínimo un input, que contiene pruebas de que el usuario está autorizado a transferir o actualizar la información del activo en cuestión.

En la práctica, el valor de “inputs” es una lista de outputs de transacción implementados como un diccionario de la siguiente forma:

```
{  
    "fulfills": {  
        "transaction_id": transaction_id,  
        "output_index": output_index  
    },  
    "owners_before": [public_key_1, public_key_2, etc.],  
    "fulfillment": fulfillment  
}
```

- “fullfills”:

Si la transacción es de tipo TRANSFER, corresponde a un puntero que señala al output que es transferido. En concreto, se trata de un vector asociativo con dos parejas clave/valor: “transaction_id”, que corresponde al ID de la transacción donde se localiza el output; y “output_index”, que corresponde al índice del output que es transferido, se trata de un número entero. Un ejemplo:

```
{  
    "transaction_id":  
    "107ec21f4c53cd2a934941010437ac74882161bcbefdf7664268823fc347996",  
    "output_index": 0  
}
```

Si la transacción fuera del tipo CREATE, el valor de “fullfills” sería nulo (*None* en Python), ya que no existe un output previo al que se pueda referir.

- “owners_before”:

Se trata de una lista de claves públicas, codificadas en Base58, según claves Ed25529 (ver anexos).

Si la transacción es de tipo CREATE, corresponde a una lista de claves públicas, con las que el registrador del activo debe firmar la transacción.

Si la transacción es de tipo TRANSFER, la lista debe coincidir con el valor de la lista “public_keys” del output del activo transferido.

- “fulfillment”:

Se trata de una cadena alfanumérica del tipo:

```
"pGSAIDgbT-nnN57wgI4Cx17gFHv3UB_pIeAzwZCk10rAjs9bgUDxyNnXM1-  
5PFgSI0rN7br2Tz59MiWe2XY0z1C7LcN52PKhpmdRtcr7GR1PXuTfQ9dE3vGhv7LHn6QqDD6qYHYM"
```

Si se trata de una transacción de tipo CREATE, el valor de “fulfillment” debe cumplimentar de manera implícita n-de-n condiciones de firma.

Si se trata de una transacción de tipo TRANSFER, el valor de “fulfillment” debe cumplimentar la condición del output que está siendo transferido.

Los detalles específicos de cómo se computan los cumplimientos y las condiciones se salen del espectro del presente documento y no son necesarios para comprender la estructura y funcionamiento de las transacciones. Existen suficientes recursos en la documentación de BigchainDB en caso de querer profundizar más.

- **Output:**

Conceptualmente, el output de una transacción determina las condiciones que deben ser cumplimentadas para que se modifique la propiedad o la información correspondiente a un activo específico. Para poder transferir un archivo, el propietario debe firmar la transacción con su clave privada, que implícitamente contiene información sobre la clave pública asociada a dicha clave privada.

En el caso de “activos divisibles”, las transacciones pueden tener múltiples outputs cada uno de ellos apuntara a la propiedad de cada parte divisible del activo.

En la práctica, el valor de “outputs” corresponde a una lista de outputs de transacciones, representado en un diccionario clave-valor. Cada output indica las condiciones criptográficas que deben ser satisfechas por cualquiera que desee transferir un activo, además, indica el número de partes de dicho activo asociadas al output. Tienen la siguiente forma:

```
{  
    "condition": condition,  
}
```

```

    "public_keys": [public_key_1, public_key_2, etc.],
    "amount": amount
}

```

- “condition”:

Corresponde a un diccionario clave/valor de la siguiente forma:

```
{
  "details": subcondition,
  "uri": uri
}
```

“details”, corresponde a la subcondición criptográfica y guarda información acerca del tipo de algoritmo criptográfico utilizado para calcular el hash de la transacción (puede tratarse de ED25519-SHA-256 o THRESHOLD-SHA-256, preferentemente el primero) y la clave pública del usuario que desea crear/transferir el activo.

“URI”, una dirección del tipo ni://sha-256;atoMY6Ye8yvidsgL9FrnKmsVzXoXrNNXFmuAPF4bQeU?fpt=ed25519-sha-256&cost=131072.

- “public keys”:

Lista de claves publicas codificadas en Base58 en claves Ed25519, debe ser consistente con el contenido de “condition”.

- “amount”:

Se trata de la cantidad del activo asociado al output en cuestión. Se puede utilizar para transferir diferentes cantidades del activo a diferentes usuarios, y la suma de los output de “amounts” debe coincidir con la suma de los outputs que se transfieren.

- **Operation**

La clave “Operation”, indica el tipo de transacción de la que se trata y, por tanto, como debe ser validada. Debe ser una cadena y los valores admitidos son “CREATE” y “TRANSFER”.

- **Asset**

En una transacción del tipo CREATE, un activo puede ser nulo o None en Python, o un vector asociativo que contenga exactamente una pareja clave/valor. La clave debe ser “data”, y debe tener una forma similar a la siguiente:

```
{  
    "data": {  
        "Asset type": "Car",  
        "Manufacturer": "BMW",  
        "Model": "M4 GTS",  
        "Owner": "Víctor Román"  
    }  
}
```

En una transacción del tipo TRANSFER, el valor asociado a “asset”, puede ser nulo o un vector asociativo que contenga exactamente una pareja clave-valor. La clave debe ser “id” y el valor asociado debe ser el de “transaction ID”, es decir, una cadena alfanumérica de 64 caracteres válida.

- **Metadata**

Corresponde a los metadatos asociados al activo en concreto. Son definidos por el creador del activo y pueden actualizarse mediante transferencias del tipo TRANSFER, en las que el destinatario de la transacción sea el mismo propietario. Se trata también de un diccionario clave/valor que puede contener tantas parejas como se desee. Es habitual representarlas y codificarlas en formato JSON.

3.4.5.2 Construir una Transacción[28]

Pasos a seguir:

- 1) Establecer la variable “version” con valor válido (actualmente “2.0”)
- 2) Establecer la variable “operation” con un valor válido (“CREATE” o “TRANSFER”).
- 3) Establecer la variable “asset” con un valor válido.
- 4) Establecer la variable “metadata” con un valor válido.
- 5) Generar o recuperar las claves públicas requeridas
- 6) Construir una lista “outputs” con todos los outputs que deben formar la transacción.
- 7) Construir una lista “unfulfilled_outputs” con todos los outputs que deben formar la transacción. Todas las cadenas “fulfillment” que estén contenidas en la lista deben estar en valor nulo.
- 8) Construir un vector asociativo llamado “unfulfilled_tx” de la forma:

```
{
  "id": ctnull,
  "version": version,
  "inputs": unfulfilled_inputs,
  "outputs": outputs,
  "operation": operation,
  "asset": asset,
  "metadata": metadata
}
```

- 9) Convertir “unfulfilled_tx” a una cadena JSON estándar de BigchainDB llamada “utx_json”.
- 10) Crear “inputs” como una copia de “unfulfilled_inputs”.
- 11) Cumplimentar o *fulfill* la lista “inputs” utilizando un método de implementación de condiciones criptográficas.
- 12) Construir un nuevo vector asociativo “tx” a partir de una copia de “unfulfilled_tx”.
- 13) En “tx”, cambiar el valor de “inputs” al valor recientemente calculado.
- 14) Computar el “transaction ID” de la transacción y llamarlo “computed_id”.
- 15) En “tx”, cambiar el valor de “id” por el de “computed_id”.

El resultado final “tx”, es una transacción correctamente cumplimentada y que puede ser enviada a través de BigchainDB. Mas adelante, en la sección de arquitectura del sistema, se detallarán ejemplos de transacción enviada, codificada en Python.

3.4.5.3 Enviar una Transacción a la red BigchainDB[27]

Una vez correctamente construida una transacción, ésta puede ser enviada a la red BigchainDB usando la API HTTP, o a través de un driver BigchainDB. En concreto, actualmente existen drivers en JavaScript y Python para llevar a cabo esta operación.

Si se decide realizar a través de la API HTTP se puede escoger el modo de envío, utilizando una de las siguientes terminaciones en la solicitud HTTP, que puede ser enviada a tantos nodos como se especifique:

POST /api/v1/transactions

POST /api/v1/transactions?mode=async

POST /api/v1/transactions?mode=sync

POST /api/v1/transactions?mode=commit

En el siguiente punto se explicará que significa cada modo.

3.4.5.4 Llegada de una Transacción a un Nodo[27]

Asumiendo que la solicitud HTTP llega exitosamente al servidor web Gunicorn que se encuentra dentro del nodo BigchainDB, Gunicorn muestra una interfaz estándar *Web Server Gateway Interface* (WSGI) que permite la comunicación de aplicaciones Python. BigchainDB utiliza la aplicación de desarrollo web Flask para simplificar la interacción WSGI/Gunicorn.

Se usa Flask para dirigir la solicitud a un método Python para gestionar la terminación especificada de posteo de transacción. Este método verifica la validez de la transacción. Si no es válida, entonces ese es el final de la vida de la transacción y ésta se pierde, el estado de respuesta del código HTTP es 400 (Error), y el cuerpo del error aporta información sobre la naturaleza del motivo por el cual la transacción ha resultado inválida.

En el caso de que se verifique la validez de la transacción, será convertida a Base64 y registrada en una nueva cadena JSON con cierta información adicional (como el modo escogido de la transacción). En este punto, el nodo BigchainDB envía la cadena a la instancia Tendermint local en el cuerpo de una solicitud POST de HTTP. Esta solicitud utiliza la API Broadcast de Tendermint.

3.4.5.5 Llegada de la Transacción a la instancia local Tendermint

Una vez que llega una transacción a la instancia Tendermint local, se le pasa el método CheckTx, que corresponde a una API que lee el modo de envío escogido. Si pasa el método, la transacción será incluida en una cola de espera (*mempool*), hasta que sea incluida en un nuevo bloque.

Se recuerda los diferentes modos en los que es posible realizar una transacción en función de la terminación escogida en la solicitud POST de HTTP:

- 1) /broadcast_tx_async: Corresponde a enviar la transacción sin esperar la respuesta de CheckTx de si es válida o no.
- 2) /broadcast_tx_sync: Devuelve el resultado de CheckTx.
- 3) /broadcast_tx_commit: Esperará hasta que la transacción sea incluida en un bloque o hasta que se alcance cierto tiempo límite, previamente establecido. Además, en caso de que no pase la verificación CheckTx, devolverá dicho resultado automáticamente.
 - **CheckTx**

CheckTx se utiliza para determinar la validez de una transacción e incluirla en un bloque. En concreto, el método realiza el siguiente recorrido:

- 1) InitChain
- 2) Info
- 3) CheckTx

- 4) BeginBlock
- 5) DeliverTx
- 6) EndBlock
- 7) Commit

Tendermint se ocupa de proponer nuevos bloques, en los que cada uno se incluye un conjunto de transacciones, y se cerciora de todos los nodos llegan a un consenso en cuanto al siguiente bloque en ser incluido en la cadena de manera resistente a fallas Bizantinas. Cada instancia de BigchainDB sigue la trayectoria del nuevo bloque que está siendo construido mediéndote la recolección de transacciones enviadas (por DeliverTx) entre BeginBlock y EndBlock.

Una vez que Tendermint envía una transacción a una instancia BigchainDB a través de DeliverTx, BigchainDB verifica de nuevo la validez de la transacción, y, si resulta válida, la incluye en memoria, pero no la registra en el servidor local de MongoDB todavía. Antes de registrarla, espera la respuesta de Commit, y, una vez recibida, incluirá el nuevo bloque (con sus correspondientes transacciones) en MongoDB.

Cuando se registra una transacción en MongoDB, BigchainDB elimina el valor asset.data y la incluye en una colección separada de activos. Esta operación se lleva a cabo para facilitar la búsqueda de activos. Un proceso similar ocurre con los metadatos asociados a dicho activo.

Cuando recibe confirmación de Commit, Tendermint registra el bloque en la cadena de bloques (almacenada en un servidor local LevelDB).

3.4.5.6 BigchainDB en el Proyecto

Como se ha comentado a lo largo del punto 3.5, las características combinadas de BigchainDB de Blockchain y base de datos, lo convierten en la opción idónea para registrar información proveniente de dispositivos IoT. Y por ello ha resultado escogida para implementar en el proyecto.

La idea es que cada captación de datos ambientales realizado por el sensor conectado a una Raspberry Pi se inyecte en un nodo BigchainDB mediante una operación del tipo CREATE, es decir, cada toma de datos constituirá un activo en sí mismo, y por tanto susceptible de ser transferido. En este sentido, las aplicaciones de uso en el sector industrial son de diversa índole, siendo la más coherente con la naturaleza del sistema, la posibilidad de realizar intercambios económicos por la información registrada en la red, que podría constituir a la información ambiental de una planta de producción o a la monitorización de los datos de una carga de transporte.

3.5 IOTA: La espina dorsal de IoT



Figura 3.19: Logo IOTA

3.5.1 ¿QUÉ ES IOTA? [29]

IOTA es una base de datos descentralizada de última generación, que utiliza una novedosa tecnología llamada la *Tangle* como núcleo de funcionamiento.

Esta Tangle es una nueva estructura de datos basada en “Grafo Acíclico Dirigido” (*Directed Acyclic Graph o DAG*). Gracias a esto, rompe con el paradigma de las Blockchain descentralizadas existentes hasta el momento, ya que este sistema permite que no existan bloques, cadena ni mineros.

IOTA es una de las tecnologías Blockchain denominadas de 3^a generación. Si Bitcoin es entendida como “oro digital” y la primera generación de criptomonedas, y Ethereum, con su entorno de desarrollo eERC20 y la implementación de contratos inteligentes, se entiende como una criptomoneda de 2^a generación; IOTA revoluciona completamente la tecnología y la usabilidad de estas monedas permitiendo, además de todo lo anterior, realizar transacciones de valor o iotas (el token de IOTA) y por tanto, el intercambio de información entre dispositivos de manera prácticamente instantánea, gratuita y descentralizada, convirtiéndose así en una herramienta de posibilidades infinitas para el desarrollo de aplicaciones en el campo del Internet of

Things, y esto es lo que ha motivado a la realización del proyecto. En las siguientes páginas se desarrollará como consigue esto.

3.5.2 Consenso

A parte de la estructura de los datos, otro gran aspecto en el que se diferencia IOTA es la manera en que se consigue el consenso y como se realizan las transacciones. Tal y como se ha mencionado anteriormente, no hay mineros. Esto supone que cada participante de la red que desea realizar una transacción debe participar activamente en el consenso de la misma, mediante la aprobación de dos (o más) transacciones previas. Esto asegura el correcto funcionamiento de la red y dota a IOTA de una serie de características únicas:

- *Escalabilidad:* IOTA puede lograr un alto volumen de transacciones simultáneas debido a la estructura paralela de validación de transacciones. Gracias a ella, no existe límite en el número de transacciones que pueden ser confirmadas en un intervalo de tiempo determinado (a diferencia de otras Blockchain).
- *Descentralización:* Cada participante de la red que realiza una transacción, participa activamente en el consenso. Dicha transacción, queda registrada y puede ser observada por cada nodo que integra el sistema. Por tanto, IOTA es la más descentralizada de las Blockchain existentes.
- *Sin comisiones por Transacciones:* IOTA no tiene mineros. Por lo tanto, no existe un sistema de compensación por el Proof of Work necesario para la validación de transacciones que otras Blockchain precisan (ya que la compensación es la posibilidad de realizar la propia transacción) y esto permite que no haya comisiones.
- *Inmunidad Cuántica:* IOTA usa un generador hash trinario llamado Curl-p, que es inmune ante decodificadores cuánticos. Se basa en un sistema de trytes, compuestos por cadenas de caracteres alfanuméricos en mayúsculas y el número 9.

3.5.3 Diseño y Transacciones en IOTA[30]

Como se ha comentado anteriormente, una de las principales características que diferencian a IOTA de otras bases de datos descentralizadas y distribuidas o DLTs, como Bitcoin y Ethereum, es la naturaleza homogénea de los usuarios de la red. No hay una división entre mineros y emisores de transacciones, cada usuario participa activamente de ambos roles ya que, para poder incluir la propia transacción en el sistema, es necesario “minar”, o validar y añadir, al menos otras dos transacciones anteriores a la red, convirtiéndose esto en el incentivo de hacer la prueba de trabajo o PoW. Dicho PoW, está configurado para que sea mucho menos costoso computacionalmente que el utilizado en otras Blockchain, como Bitcoin.

Este sistema de validar dos transacciones previas da pie a que, en vez de una Blockchain, o cadena de bloques, el sistema se construya como un Grafo Acíclico Dirigido o *Directed Acyclic Graph* en inglés, a partir de ahora este tipo de grafo se denotara como *DAG*. La DAG de IOTA, se denomina Tangle y se realizará una descripción más detallada del funcionamiento y la estructura del Tangle en el siguiente punto.

Es destacable el hecho de que existen ciertas discrepancias entre el diseño y funcionamiento teórico del Tangle en IOTA, tal y como viene explicado en su *Whitepaper*, y la implementación práctica del mismo. Las principales diferencias se comentarán a continuación.

3.5.3.1 Nodos Completos (*Full Nodes*)

En teoría, los nodos completos almacenan, actualizan y verifican las transacciones realizadas desde la última *Snapshot*, que ocurren con una frecuencia aproximada de un mes. Dado que la red de IOTA es asíncrona, un nodo completo no tiene por qué registrar el estado global de la Tangle por sí solo. Cada nodo completo es responsable de propagar y controlar las transacciones a los nodos vecinos, y de esta manera, la red de nodos forma colectivamente la Tangle.

Al no haber incentivo económico directo por operar un nodo completo, actualmente los usuarios realizan esta tarea sobre todo por motivos altruistas y para contribuir a la seguridad y al correcto funcionamiento de la red. Los nodos completos permiten, a los usuarios de IOTA, verificar las transacciones realizadas desde más allá del último *milestone*, hasta el último *Snapshot*. Si el usuario confía en que los Snapshots realizados en el pasado, se hicieron correctamente, pueden tener la seguridad que están verificando tan solo transacciones válidas.

En el futuro, los desarrolladores de IOTA planean incentivar a los nodos completos a registrar la historia completa de la Tangle (más allá de los Snapshots) mediante un concepto llamado *Permanodes*, los cuales recibirán una recompensa por cada consulta acerca del estado de la red.

3.5.3.2 Clientes Ligeros

Los clientes ligeros son dispositivos que dependen de la red de nodos completos para poder interactuar con el Tangle. No llevan a cabo ningún tipo de registro del estado de la red. Sin tener acceso al Tangle, la selección de tips no puede ser realizada, por lo tanto, cuando un cliente ligero (como un dispositivo IoT o una cartera de un usuario) interactúa con un nodo completo, el cual lleva a cabo el algoritmo de selección de tips, tan solo tienen acceso a la parte del Tangle que ese nodo completo tiene registrada. Por ello, los clientes ligeros corren el riesgo de interactuar con nodos maliciosos que propaguen transacciones falsas a lo largo del sistema. Así, un incentivo para operar un nodo completo es no tener que depender en otros nodos controlados por entidades

desconocidas para realizar transacciones al Tangle, esto se vuelve especialmente interesante en el ecosistema IoT, ya que se requiere una confianza del 100% en que las transacciones sean aceptadas por la red.

3.5.3.3 El Coordinador y las Milestones

Un tercer tipo de nodo que forma parte actualmente del Tangle de IOTA, es el Coordinador. El protocolo de IOTA, en el momento presente, opera un nodo Coordinador que es un tipo especial de nodo completo, operado por la IOTA Foundation, que emite una transacción (llamada *milestone*) cada dos minutos aproximadamente, en la que se referencia tan solo a las transacciones válidas. Todos los nodos que ejecutan el software oficial de IOTA reconocen la firma del coordinador y tratan las transacciones referenciadas como válidas.

Por tanto, la transacción milestone constituye un tipo especial de transacción publicada por el coordinador. El objetivo de esta transacción es reducir el número de transacciones que los clientes ligeros tienen que verificar, y así prevenir que el Tangle se ramifique demasiado transversalmente, lo que resultaría en diferentes estados de la red presentes en finales concretos del Tangle simultáneamente. Lo hace atrayendo las ramificaciones al centro de la Tangle y resolviendo el problema potencial del doble gasto a la vez que previene las inconsistencias del sistema.

Por consiguiente, una transacción en IOTA es tratada como una “transacción confirmada” si, y solo si, está directa o indirectamente referenciada por una transacción milestone. Esto reduce la cantidad de computación necesaria en el proceso de validación para el resto de los nodos. El algoritmo de selección de tips realizado por el nodo Coordinador al publicar transacciones milestones no está revelado en el presente. La IOTA Foundation sostiene que las transacciones milestones son necesarias como un mecanismo de protección ante ataques del 34%, hasta que la red sea lo suficientemente grande como para que este tipo de ataques no supongan un peligro real. Se supone que, en ese momento, el papel del coordinador no será necesario, y el Tangle podrá operar sin él.

La IOTA Foundation sostiene a su vez, que el objetivo último es llevar a cabo una transición a un protocolo en el que no se necesite el papel del Coordinador, por tanto, en ese punto se desharían de él, y la red podría actuar de forma verdaderamente descentralizada. Ya que, actualmente la implementación del protocolo posee características que la clasifican como una red altamente centralizada (a pesar de que anuncian todo lo contrario), al ser el Coordinador el único punto de fallo del sistema.

3.5.3.4 Snapshots

Otra característica que no se menciona en el Whitepaper pero que se utiliza actualmente en el protocolo, son los Snapshots. Los Snapshots son una especie de barrido o poda que se realiza cada cierto tiempo (entre uno y dos meses) y que sirve

para reducir el tamaño del Tangle. Borra todas las direcciones (y por tanto sus contenidos), que contengan un balance igual a cero y guarda aquellas cuyo balance es positivo.

Esto crea una nueva transacción génesis desde la que el Tangle puede volver a expandirse. La motivación principal de los Snapshots es mantener una red ligera y con un tamaño reducido y, por tanto, manejable. Estos Snapshots permiten que la red trabaje más rápida y eficientemente, a costa de la auditabilidad, ya que los usuarios normales pierden la capacidad de poder ver la totalidad de la historia de la red desde la transacción Génesis original.

3.5.3.5 Proof of Work (PoW):

IOTA utiliza PoW para protección de spam, similar en espíritu al PoW utilizado en Hascash. Pero no debe ser confundido con el costoso PoW que se emplea en *DLTs* basadas en mineros como Bitcoin, ya que se trata de una operación computacional significativamente más corta.

- Magnitud de Peso Mínima (Min Weight Magnitude):

La dificultad del PoW reside en una variable llamada Magnitud de Peso Mínima (MWM), la cual se refiere al número de ceros arrastrados (en trits) en la encriptación de una transacción y es proporcional a la dificultad del PoW.

El dispositivo que realiza este PoW empleara fuerza bruta de computación para encontrar un “nonce” que tenga el numero correcto de ceros arrastrados. Cada cero arrastrado extra que deba de ser encontrado multiplicará por un factor de 3 la dificultad del PoW.

Los valores típicos de MWM en el Tangle son:

- MWM de la red principal (Mainnet): 14
- MWM de la red de prueba (Testnet): 9

3.5.3.6 El Token de IOTA (MIOTA)

El token de IOTA es un “cryptotoken”, construido sobre la tecnología de la Tangle y se beneficia de muchas de las ventajas que aporta la estructura de la misma. Opera en una Tangle “libre de permiso” (permission-less) conocida como Mainnet. Los usuarios interactúan con dicha red al operar con un nodo completo que se encuentra conectado con vecinos. Alternativamente también se puede operar con nodos de la Mainnet a través de una tercera parte, que constituyen nodos ligeros, pueden ser instalados y utilizados en una gran variedad de dispositivos electrónicos.

3.5.3.7 Generación de Tokens

Todos los IOTAs que existirán se crearon en la transacción génesis. Esto significa que la cantidad total de IOTAs disponibles se mantendrá siempre constante y no se podrán “minar”. Por lo que al realizar PoW en IOTA, no se estarán generando nuevos IOTAs sino verificando otras transacciones.

La cantidad total de IOTAs existentes es de $(3^{33}-1)/2$, es decir, “2779530283277761” = ~2.8 Peta IOTAs. Al estar diseñada específicamente para comunicación entre máquinas, esta alta cantidad de oferta la hace ideal para nano-transacciones. Y esta cantidad se ajusta adecuadamente al valor “MAX_SAFE_INTEGER” en JavaScript.

Actualmente, la criptomoneda de referencia es Bitcoin, de la cual se dispondrán aproximadamente 21 billones BTC en el año 2140. Si se dividen en su unidad mínima, el máximo número de unidades disponibles de BTC será de 2.1 Peta Satoshi's. Esto se traduce en que habrá un ~32% más de IOTA's disponibles cuando se alcance el máximo de BTC.

3.5.3.8 Etimología de la Oferta

Dado que el número de tokens es ciertamente elevado, se utilizan prefijos del Sistema Internacional para referirse a grandes cantidades de IOTA. Algunas de las más utilizadas son:

Tabla 3.9: Unidades típicas IOTA

Unidad	Nombre	Cantidad	Potencia
Pi	peta IOTA	1.000.000.000.000.000	10^{15}
Ti	terra IOTA	1.000.000.000.000	10^{12}
Gi	Giga IOTA	1.000.000.000	10^9
Mi	Mega IOTA	1.000.000	10^6
Ki	Kilo IOTA	1.000	10^3

3.5.3.9 El sistema Trinario

Siempre que se refiera a lo largo del documento a seeds, direcciones, encriptaciones... basadas en trinario, significa que la cadena de caracteres solo puede consistir en las letras del alfabeto latino en mayúsculas y el número 9.

- Todos los posibles valores “9ABCDEFGHIJKLMNPQRSTUVWXYZ”.

Debido al funcionamiento del Kerl (el algoritmo criptográfico que utiliza IOTA), una encriptación siempre consiste en 81 trits (caracteres).

- Trits validos: “VBVEUQ”
- Trits inválidos “Vaadh32”

3.5.3.10 Seeds, Claves Privadas y Cuentas[31]

3.5.3.10.1 Seeds y Cuentas

Las seeds son la base del sistema. Para crear una cuenta con una clave privada es necesaria una seed segura. La seed consiste en 81 Trytes (o menos, pero no se recomienda), y es el único acceso del usuario a la clave para su cuenta y, por lo tanto, los fondos e información almacenados en la misma.

Existen tres niveles de seguridad, siendo el alto el recomendado para utilizar en según los desarrolladores. La librería para clientes hace posible seleccionar fácilmente el nivel de seguridad entre los siguientes:

- Bajo: 81 trits
- Medio: 162 trits
- Alto: 243 trits

3.5.3.10.2 Claves Privadas y Cuentas

Las claves privadas se derivan de los índices de las claves de seed. Desde esa clave privada se genera la dirección de la cuenta. Es importante tener en cuenta que todas las funciones vulnerables en cuanto a la seguridad se implementan desde el lado del cliente. Esto quiere decir que se pueden generar llaves privadas y direcciones de forma segura desde el explorador o desde un ordenador offline. Todas las librerías existentes en los diferentes lenguajes de programación (Python, JavaScript...) aportan esta funcionalidad.

3.5.3.11 Publicar una Transacción en IOTA[30]

A continuación, se hará una introducción de alto nivel al proceso requerido para publicar una transacción en el Tangle de IOTA, y más adelante, se procederá a explicar cada paso en detalle.



Figura 3.20: Proceso para publicar una transacción en el Tangle de IOTA

- 1) Firma: El nodo firma la transacción con su clave privada
- 2) Selección de tips: El nodo selecciona aleatoriamente dos transacciones que no han sido validadas (tips), según un algoritmo determinado. En el Whitepaper de IOTA, los creadores declaran que no imponen ninguna regla específica de selección de tips, pero recomiendan encarecidamente el método Cadena Monte Carlo Markov (MCMC). Mas detalle en el siguiente punto.
- 3) Validación: Una vez seleccionados los tips, el nodo ha de verificar que las transacciones no son conflictivas con la historia de la Tangle hasta el momento, y en caso de serlo, seleccionar nuevos tips.
- 4) Prueba de Trabajo: A diferencia del PoW de bitcoin, los nodos no compiten por la recompensa, la dificultad de resolución del PoW se mantiene constante ya que esta sirve únicamente como un mecanismo de protección ante ataques SPAM.
- 5) Publicación: Una vez completados los pasos 1-4, la transacción se publica al Tangle.

3.5.3.11.1 Campos de la Transacción[32]

Tabla 3.10: Campos transacción IOTA

Nombre	Tipo - Longitud	Descripción
Address	Cadena de caracteres - 27	Output = dirección del recipiente. Input = dirección desde la que se manda la transacción.
Value	Entero -	Valor de la transacción.

signatureMessageFragment	Cadena de caracteres - 2187	En caso de haber una cantidad gastada como input, la dirección generada por la llave privada se guarda aquí. Si no se requiere firma, está vacía y puede ser usada para guardar el valor del mensaje al realizar la transacción.
Tag	Cadena de caracteres - 27	Tag definido por el usuario
Bundle	Cadena de caracteres - 81	Encriptación del Bundle actual, que es usado para agrupar transacciones. Sirve para identificar transacciones que formaban parte del mismo conjunto
hash	Cadena de caracteres - 81	Encriptación en trinario de una única transacción.
nonce	Cadena de caracteres - 27	La Nonce es un requisito indispensable de la transacción para poder ser aceptada por la red. Se genera mediante el PoW, ya sea mediante la llamada API “attachToTangle” o mediante una librería externa.
branchTransaction	Cadena de caracteres - 81	Transacción siendo aprobada.
trunkTransaction	Cadena de caracteres - 81	Transacción siendo aprobada.
currentIndex	Entero -	Índice de transacción del paquete.
lastIndex	Entero -	Índice de la última transacción del paquete

Sea el caso en que A quiere enviar 80i a la dirección[o] de B: QQQQ....QQQ.

1) Hacer un Bundle de transacción:

El Bundle, o paquete, es la unidad de transacción, que incluye tres tipos de transacciones: Input, Output y meta-transacciones.

- Input: el valor de la transacción es negativo.
- Output: el valor de la transacción es positivo.
- Meta-transacción: el valor de la transacción es oi, puede contener la firma de la transacción, u otro mensaje en el campo correspondiente.

Para el escenario estudiado, en primer lugar, se ha de preparar el output de la transacción, es decir, especificar que se quieren enviar 80i a la dirección de B.

Transaction

Address :	QQQQQQ.....QQQ
Value :	80
Tag :	VISUALTRANSAC
Timestamp:	CurrentTime()
Index :	
LastIndex:	
Bundle :	
Nonce :	
Message :	WELCOME9T09IOTA

A continuación, se ha de preparar el input de la transacción, en el escenario estudiado, es imprescindible que las cuatro direcciones de A que contienen tokens de IOTA, sumen una cantidad igual o superior a la cantidad que se quiere enviar, para poder cumplimentar el output. Es decir: $10 + 5 + 25 + 60 \geq 80$.

Dado que el nivel de seguridad por defecto es 2, se necesita una meta-transacción de valor `oi`, que contenga la firma de la transacción.

Transaction

Address : `AAAAAA.....AAA`
Value : `-10`
Tag : `VISUALTRANSAC`
Timestamp: `CurrentTime()`

Index :
LastIndex:
Bundle :
Nonce :

Message :

Transaction

Address : `CCCCCC.....CCC`
Value : `-25`
Tag : `VISUALTRANSAC`
Timestamp: `CurrentTime()`

Index :
LastIndex:
Bundle :
Nonce :

Message :

Transaction

Address : `BBBBBBB.....BBB`
Value : `-5`
Tag : `VISUALTRANSAC`
Timestamp: `CurrentTime()`

Index :
LastIndex:
Bundle :
Nonce :

Message :

Transaction

Address : `DDDDDDD.....DDD`
Value : `-60`
Tag : `VISUALTRANSAC`
Timestamp: `CurrentTime()`

Index :
LastIndex:
Bundle :
Nonce :

Message :

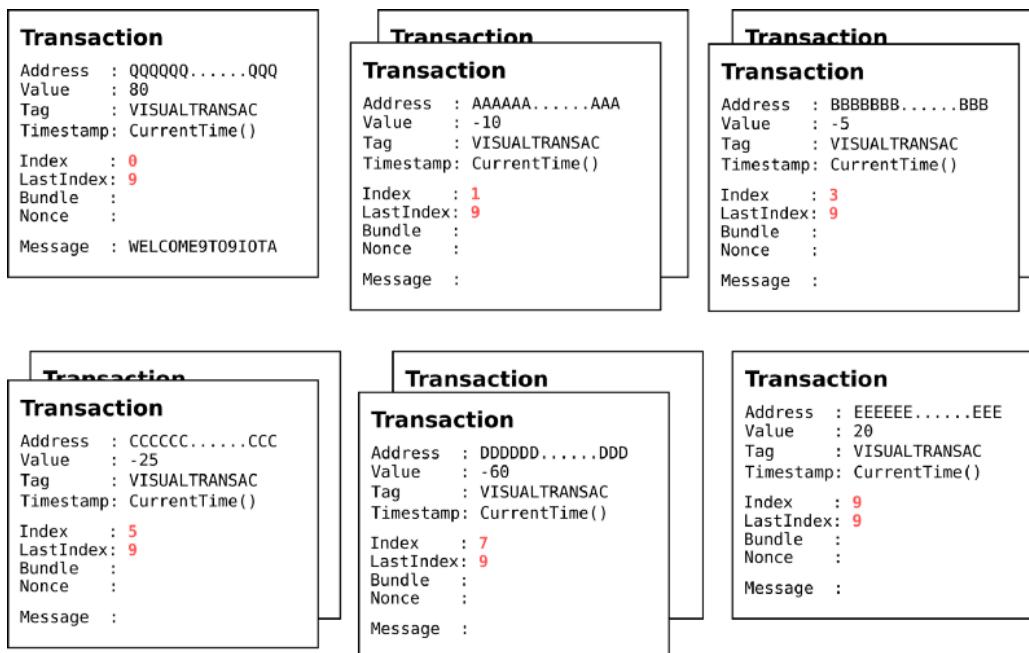
Se aprecia que el balance de los valores entre las transacciones input y output, es positivo ya que: $10 + 5 + 25 + 60 = 100 > 80$. Por lo tanto, hay `2oi` que no se gastan y hay que construir otra transacción adicional para enviar dicha cantidad remanente a otra dirección propiedad de A.

Transaction

```
Address : EEEEEEE.....EEE  
Value   : 20  
Tag     : VISUALTRANSAC  
Timestamp: CurrentTime()  
  
Index   :  
LastIndex:  
Bundle  :  
Nonce   :  
  
Message :
```

2) Finalizar el Bundle:

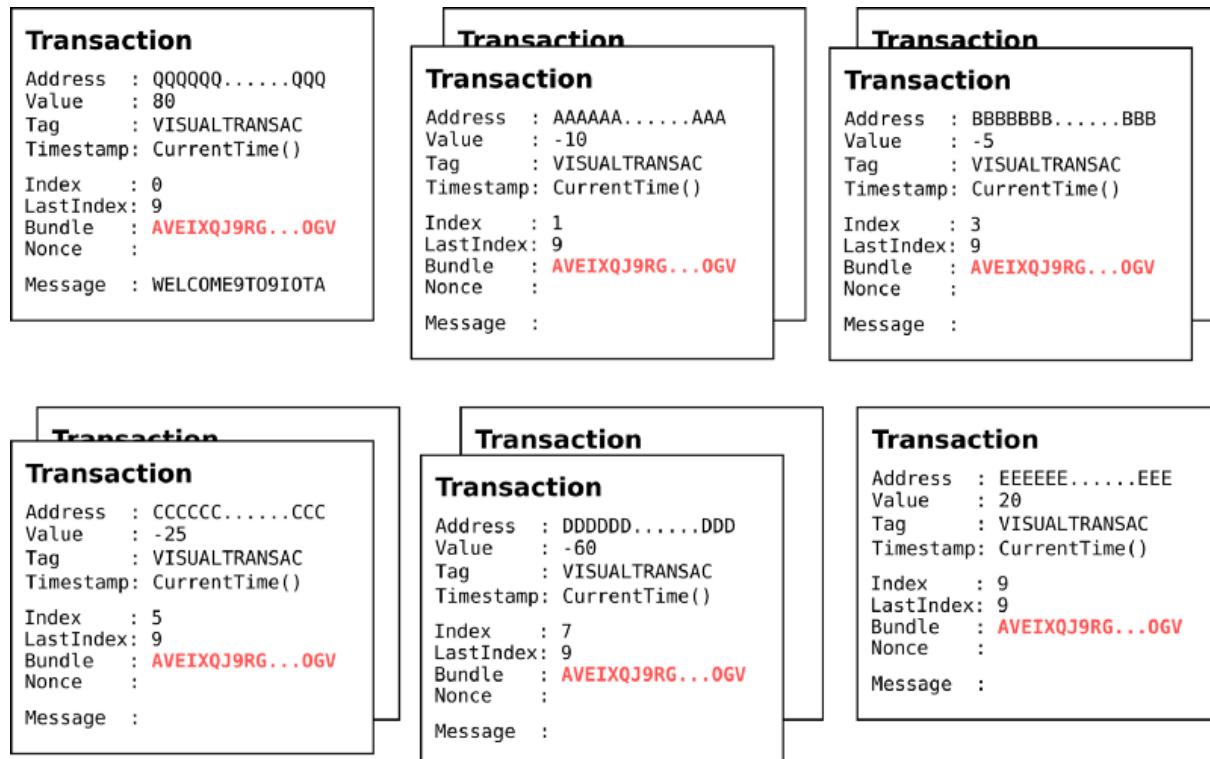
En este paso, se completan los campos de “transaction_index”, “last_index” y se genera el hash del paquete con la función hash Kerl.



Se validan los campos “address”, “value”, “obsolete tag”, “timestamp”, “index” y “last index”. La función hash Kerl, usa una construcción tipo esponja, es decir, absorbe los campos de validación por orden y expulsa el resultado. Se sigue un método parecido al del algoritmo criptográfico SHA-3.

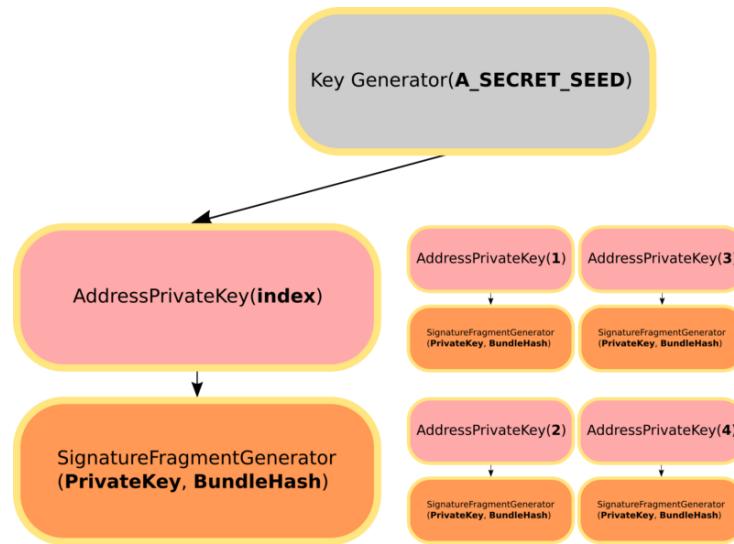
Además, en el proceso de creación del hash del paquete, chequeará si el hash es seguro o no. Si no lo es, incrementará el valor del campo “obsolete tag” y generará un nuevo hash.

Una vez generado un hash satisfactorio, se incluirá en el campo correspondiente en todas las transacciones del paquete. Obteniendo:



3) Firma de las transacciones input:

A continuación, se firman las transacciones input con las claves privadas correspondientes a cada transacción. Se pueden obtener a partir de un generador de claves y la seed de A. para obtener la clave privada de una dirección, se puede usar también *Signature Fragment Generator*, con la clave privada y el hash del bundle como inputs.

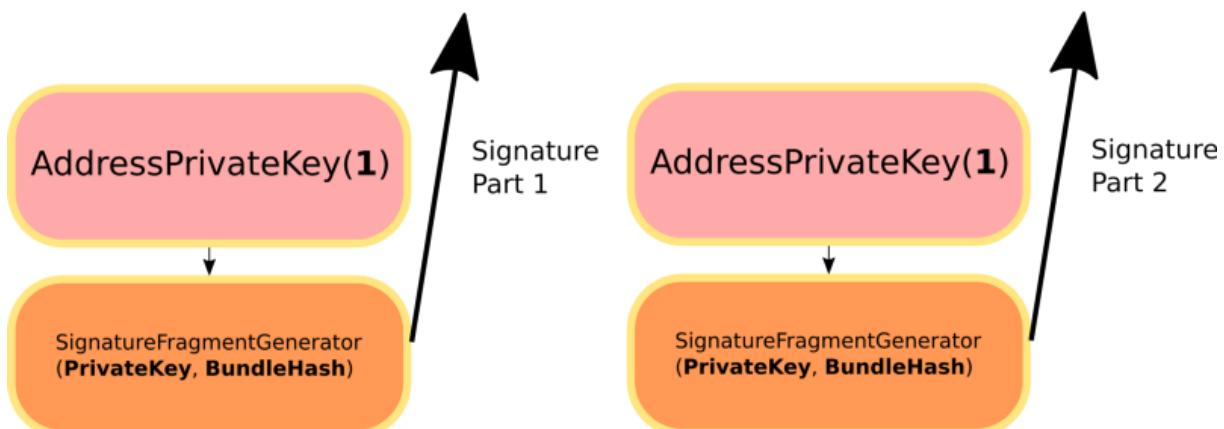


Transaction

Address	:	AAAAAAA.....AAA
Value	:	-10
Tag	:	VISUALTRANSAC
Timestamp	:	CurrentTime()
Index	:	1
LastIndex	:	9
Bundle	:	AVEIXQJ9RG...0GV
Nonce	:	
Message	:	SIGNNELMAOV09IQ...

Transaction

Address	:	AAAAAAA.....AAA
Value	:	0
Tag	:	VISUALTRANSAC
Timestamp	:	CurrentTime()
Index	:	2
LastIndex	:	9
Bundle	:	AVEIXQJ9RG...0GV
Nonce	:	
Message	:	...CVEKMVIELQOAL



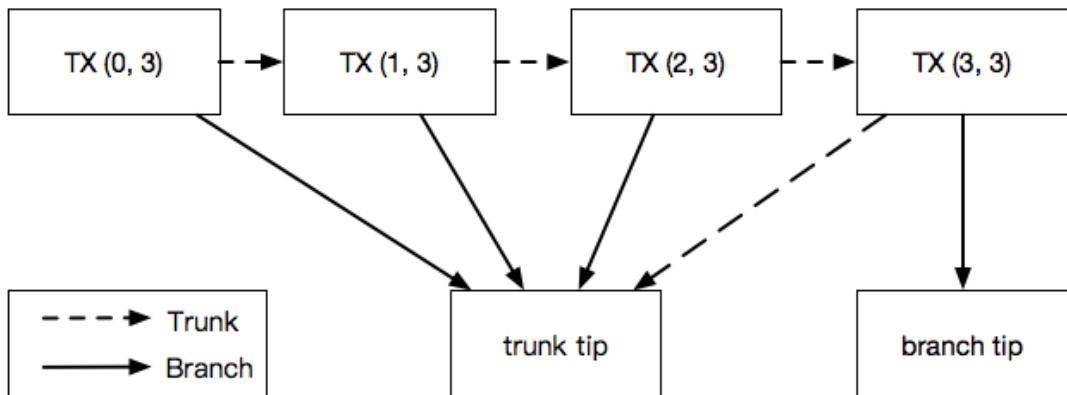
Una vez añadidos las claves privadas, el paquete queda completo.

4) Selección de tips:

Éste, es un proceso complejo, y se explicará en profundidad en un punto posterior.

5) Proof of Work:

En este paso, se completarán los campos “last index”, “trunk”, “Branch” y “nonce”, en cada transacción del paquete.



6) Publicación de la transacción:

Una vez que esté todo correcto, la transacción se publica en el Tangle satisfactoriamente.

3.5.3.12 La Tangle en Profundidad. Estructura y Selección de Tips[34]

Para entender adecuadamente la *Tangle*, se ha de introducir primero el concepto utilizado en las ciencias de computación de *Directed Graph*, que son una colección de vértices o nodos (correspondientes a los cuadrados en la ilustración adjunta) que están conectados entre sí mediante *edges* (flechas).

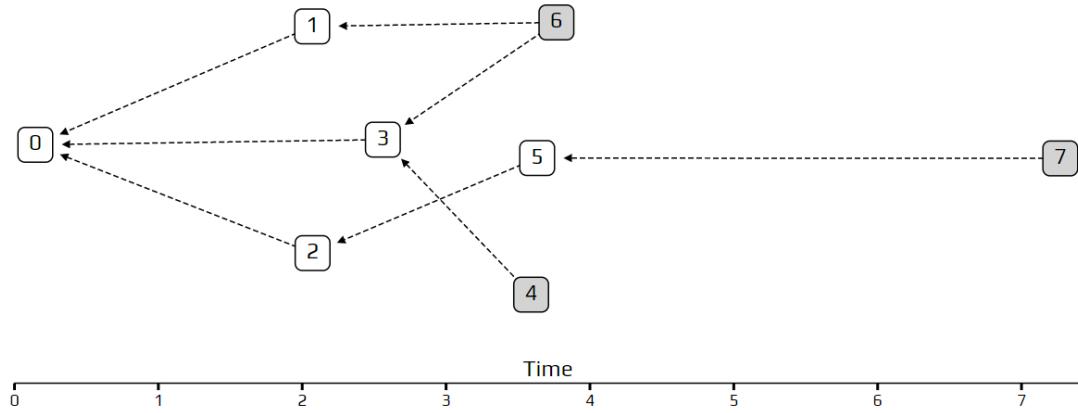


Figura 3.21: Simulación red Tangle de IOTA

La Tangle, que es la estructura de datos que sustenta a IOTA, es una clase particular de Directed Graph, a través de la cual se realizan transacciones. Cada transacción se representa como un vértice en el gráfico. Cuando una nueva transacción se une a la Tangle, elige dos transacciones previas que aprobar o validar, añadiendo dos nuevos edges al gráfico. En el ejemplo anterior, la transacción número 5 aprueba las transacciones 2 y 3. Estas transacciones son información que se computa de manera similar “Persona A envía a la persona B 100 IOTAs”.

Las transacciones que todavía no se han validado, se entienden como *tips*. En el ejemplo, la transacción nº 6 correspondería a un *tip*. Cada nueva transacción entrante ha de seleccionar dos *tips* previas para validarlas antes de que dicha transacción sea elegible para ser validada (es decir, se convierta en un *tip*).

3.5.3.12.1 Estrategias de Selección:

La estrategia para elegir *tips* es una de las claves de IOTA y posteriormente se desarrollará en profundidad, de momento se asumirá que se utiliza la estrategia más simple:

- 1) Eligiendo dos *tips* de manera aleatoria entre los disponibles (*Uniform Random Tip Selection*).

A continuación, se muestran imágenes de una simulación, creada por el equipo de desarrolladores de IOTA, que ilustra esta estrategia de selección aleatoria de *tips*. La simulación genera redes aleatorias, con la primera de ellas (o *génesis*) situada en el extremo izquierdo (es la que tiene el número 0) y avanzando hacia las más recientes en el tiempo a través del eje inferior. La estructura de la red por lo tanto viene determinada por el número de transacciones que se dan en la misma y por el ratio de transacciones (λ).

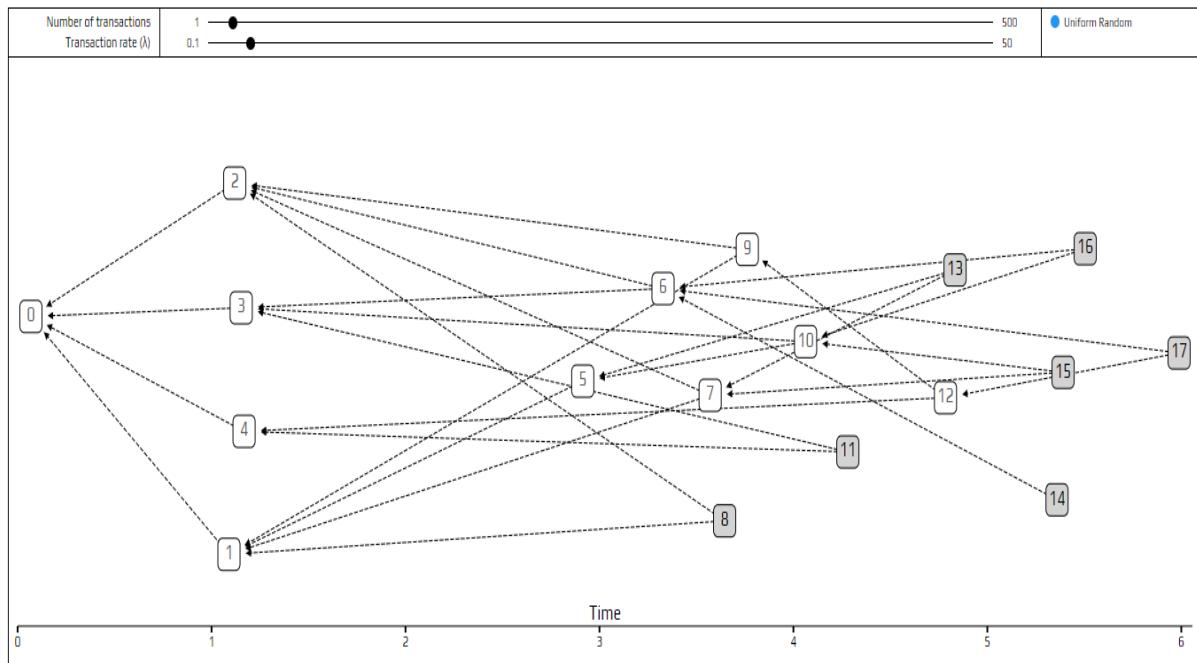


Figura 3.22: simulación Tangle con 17 nodos y ratio de transacción $\lambda = 1.5$

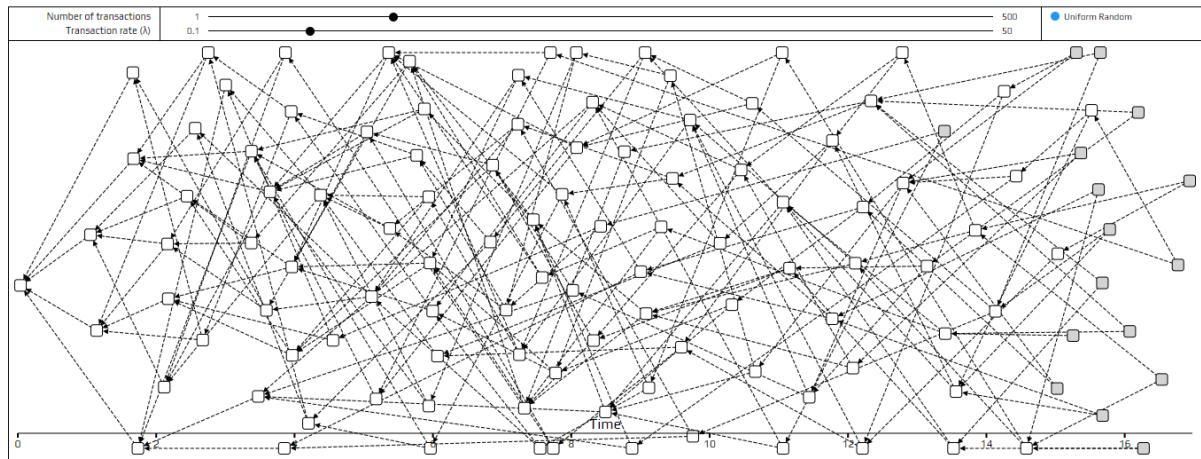


Figura 3.23: Simulación red Tangle con 120 nodos y ratio de transacción $\lambda = 7.1$

Como se puede observar las transacciones no están repartidas uniformemente en el tiempo, existen ciertos periodos que están más “ocupados” que otros. Esta aleatoriedad, que hace el modelo más realista, se consigue mediante la aproximación en la frecuencia de aparición de nuevas transacciones, a un Proceso de Poisson, un tipo de método matemático que consiste en distribuir una serie de puntos localizándolos aleatoriamente en un espacio matemático, con el que se asume que cada transacción es independiente de las demás. Este tipo de modelos son muy adecuados para analizar sucesos aleatorios independientes como: cuantos clientes entran en un supermercado

en un intervalo de tiempo determinado o cuantas llamadas de teléfono se realizan a un call center.

En la Figura 15, las transacciones 4, 5 y 6 llegan casi simultáneamente y después de la transacción numero 6 hay una larga pausa.

Para entender este método de Distribución de Possion se considera que, de media, el ratio de transacciones entrantes es constante y se representa por lambda λ . Como ejemplo, si se considera $\lambda = 2$ y el número de transacciones igual a 100, el tiempo de simulación total será de 50 unidades de tiempo.

Si se establece un λ suficientemente pequeño, se obtiene una cadena. Una cadena es una *Tangle* en la que cada transacción únicamente se valida por otra, en vez de por dos. Esto ocurre porque las transacciones llegan a un ratio tan bajo que en un instante de tiempo determinado tan solo existe un *tip* disponible para ser aprobado.

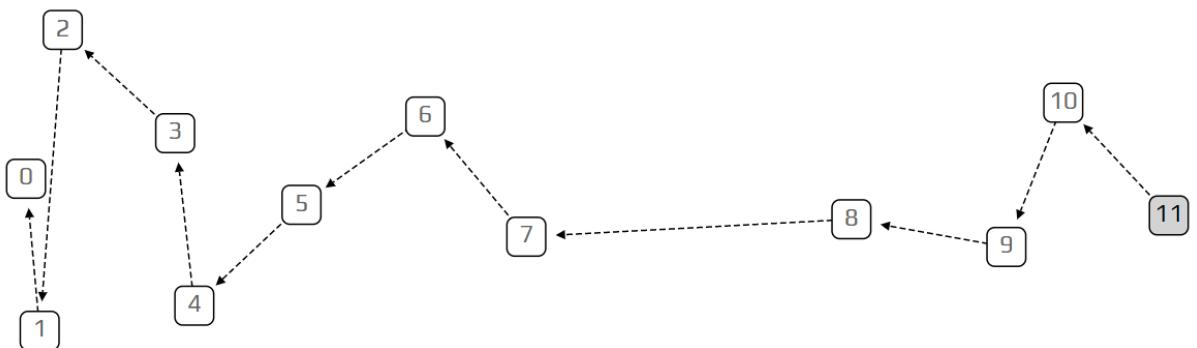


Figura 3.24: Simulación red Tangle con $\lambda = 0.1$

En el otro caso extremo, con un λ suficientemente elevado, todas las transacciones llegan tan rápidamente que el único *tip* que ven es el *génesis*. Esto es únicamente una limitación de la simulación en la que se considera un intervalo de tiempo muy corto.

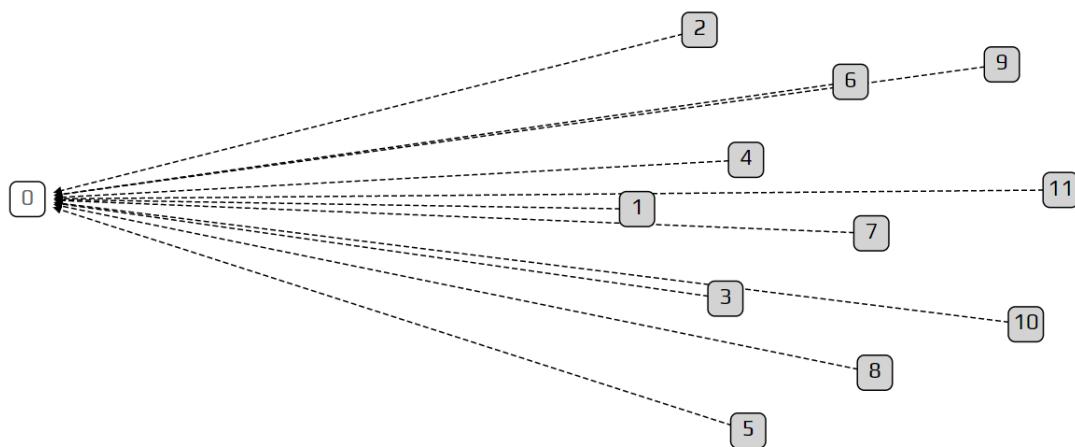


Figura 3.25: Simulación red Tangle con λ muy elevado

Cuando se dice que una transacción no reconoce a una anterior, en el modelo se hace que cada transacción sea invisible durante un intervalo de tiempo determinado después de su llegada, denotado por h . Éste retraso representa el tiempo que le toma a una transacción para ser preparada y propagada a través de la red. En la simulación, se establece $h = 1$. Esto quiere decir que solo se pueden validar transacciones que se encuentran como mínimo una unidad de tiempo en el pasado. Este retraso es una propiedad fundamental de la *Tangle*, con la que se busca modelar de forma más cercana a la realidad las transacciones.

2) Unweighted Random Walk:

Este algoritmo, recorre la *Tangle* desde el *génesis* a través de los *tips*. En cada paso se va saltando a una de las transacciones que valida directamente aquella en la que se encuentra. El método de elección de la transacción a la que se salta es de manera aleatoria, teniendo ambas transacciones las mismas probabilidades de ser escogidas

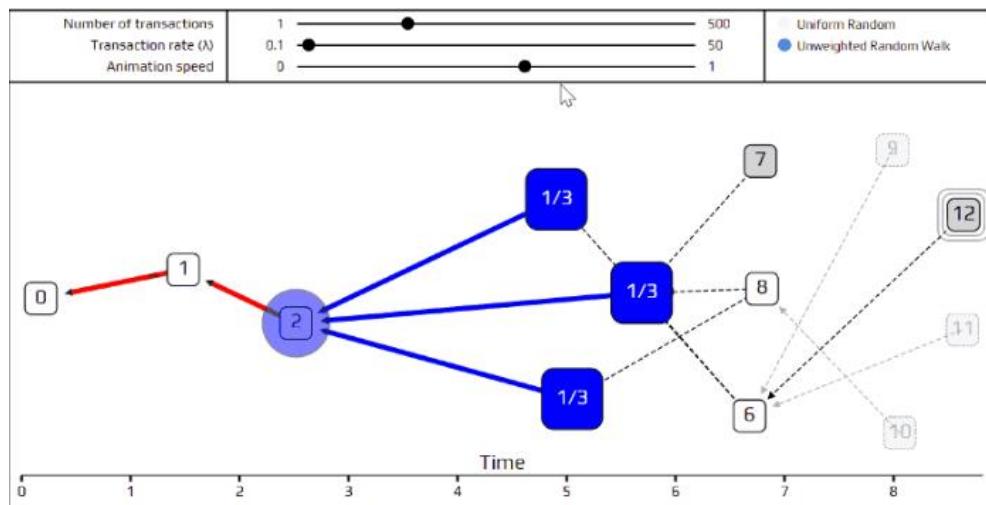


Figura 3.26: Simulación red IOTA con algoritmo Unweighted Random Walk

(*unweighted* o sin peso).

El camino recorrido es el marcado en color rojo, y en cada intersección los posibles caminos a recorrer se marcan en azul, las transacciones que son tan recientes que se tratan como “invisibles” por la red se representan en color transparente. El número que se encuentra en el interior de los posibles nodos a recorrer es el correspondiente a la probabilidad que posee dicho nodo de ser escogido.

3) Weighted Random Walk:

Uno de los principales aspectos que se buscan en el algoritmo para la selección del *tip* es evitar “*tips vagos*”. Éstos son *tips* que validan transacciones antiguas antes que las más recientes. Se les llama vagos porque no se preocupan por mantenerse actualizados ante los estados de la red *Tangle*, lo que perjudica a la red ya que provoca que no se validen las nuevas transacciones.

Además, al *tip* que toma este comportamiento (perfectamente factible en la anterior estrategia de selección de transacciones que validar), no se le penaliza de ninguna manera. Para lidiar con este problema surge un nuevo algoritmo llamado *Weighted Random Walk*, que incentiva aquellos *tips* que no se comportan de dicha manera. Se toma este acercamiento ya que tampoco se puede forzar a los *tips* a validar solo las transacciones más recientes (lo que rompería con la idea de descentralización), además, tampoco se posee una manera fiable de saber exactamente el tiempo en el que llega una nueva transacción.

La estrategia a seguir, por tanto, será la de sesgar el recorrido aleatorio, de forma que sea menos probable elegir *tips* vagos. Se usará el término *peso acumulado* para determinar como de importante es una transacción, y en el sistema, será más probable que se elija una transacción con un peso elevado antes que una “ligera”. La definición de *peso acumulado* es la siguiente: se cuentan el número de nodos “validadores”, tanto directos como indirectos, que tiene una transacción y se le suma uno. En el ejemplo siguiente, la transacción 3 tiene un peso acumulado de cinco, porque tiene cuatro transacciones que la validan (5 directamente, 7,8 y 10 indirectamente)

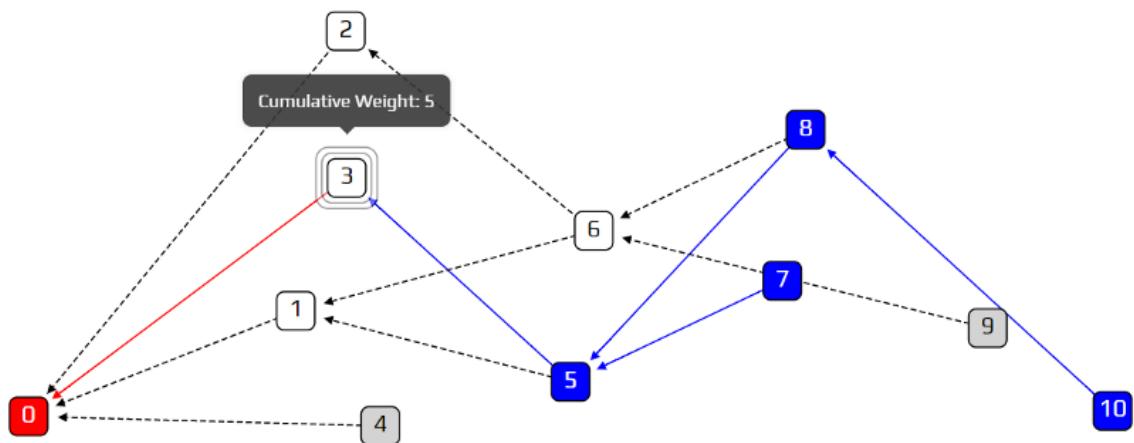


Figura 3.27: Representación de la estrategia Weighted Random Walk (1)

Para ilustrar de manera más clara el funcionamiento, se verá otro ejemplo: en la siguiente figura, el *tip* 16 es uno vago. Para que pueda ser validado, se debe recorrer la transacción 7, y después se debe elegir la transacción 16 antes de la 9. Esto es muy improbable que ocurra dado que la transacción 16 tiene un peso acumulado de 1, mientras que la 16 uno de 7.

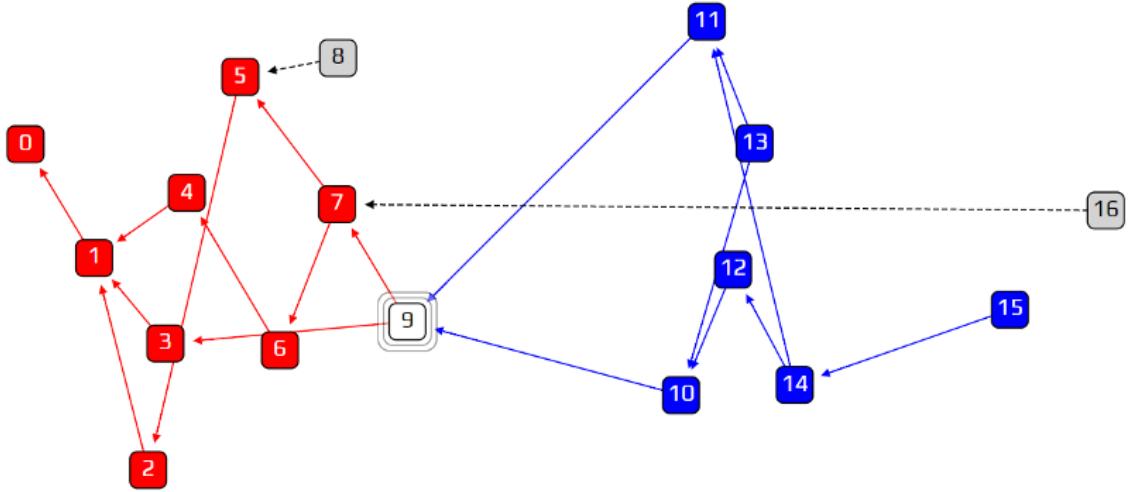


Figura 3.28: Representación de la estrategia Weighted Random Walk

Ante este punto podría surgir la pregunta de ¿Por qué no establecer un *Super-Weighted Random Walk* en el cual ante cada intersección se escoja la transacción con mayor peso acumulado sin involucrar probabilidades? Ante esta situación se obtendría algo parecido a la siguiente figura:

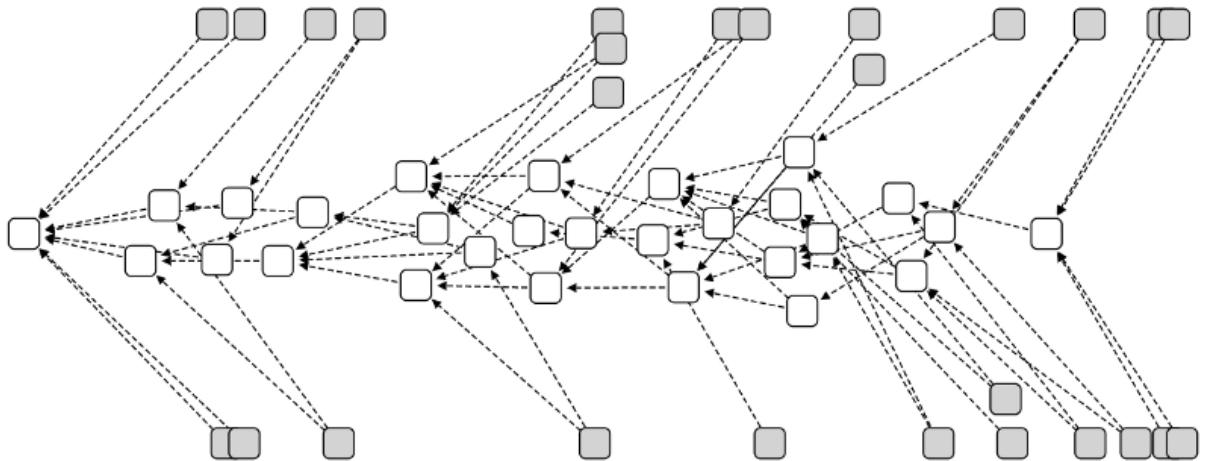


Figura 3.29: Representación de la estrategia Super-Weighted Random Walk

Los *tips* grises representan transacciones sin “validadores”. Mientras que es normal tener algunos *tips* a la derecha del diagrama sin validar, es un problema tener tantos repartidos por la línea temporal. Estos son *tips* que se han quedado atrás y nunca serán validados. Esto es la consecuencia de elegir *solo* las transacciones con mayor peso acumulado en todo momento.

Se necesita un método para definir como de probable es que se elija un recorrido concreto hacia un “validador” particular en un momento determinado. Se sesgará positivamente aquellas transacciones con un mayor peso relativo acumulado. Esto introduce un nuevo parámetro α que determina como de importante es el peso acumulado de una transacción.

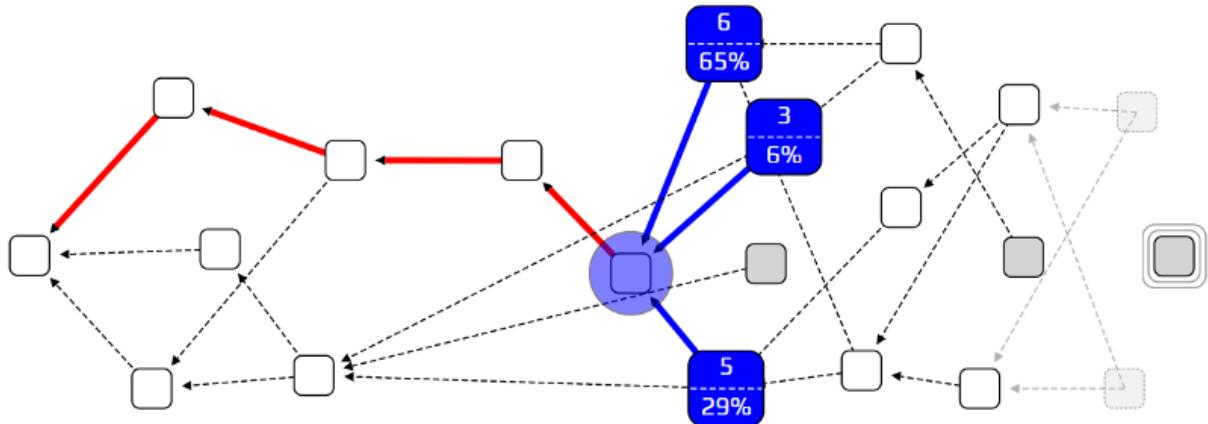


Figura 3.30: Weighted Random Walk, cada transacción azul muestra su peso acumulado y la probabilidad de ser escogido como el siguiente paso en el recorrido

Si se establece un $\alpha = 0$, se obtiene de nuevo un recorrido sin peso. Si se establece con un valor muy alto, se obtiene el un recorrido super-pesado o Super-Weighted Walk. Se debe encontrar un punto intermedio adecuado entre la penalización a los tips vagos y no dejar demasiado tips atrás. Determinar el valor ideal es una importante área de investigación actualmente para los desarrolladores de IOTA.

Este método de establecer una regla que determine la probabilidad de cada paso del recorrido aleatorio se llama *Método de Markov para Cadenas Monte Carlo* ó MCMC. En una cadena de Markov cada paso no depende del anterior, sino que sigue una regla que se decide por adelantado.

3.5.3.12.2 Validadores, Balances y Dobles Gastos:

Como se ha mencionado anteriormente, cada transacción tiene información del tipo “Alicia da a Bernardo 100 IOTAs”. Es trabajo del validador asegurarse de que Alicia realmente posee esos 100 IOTAs antes de enviarlos.

¿De dónde surgen estos IOTAs? La cantidad total de IOTAs que existen y que existirán, fueron creados en el momento del *génesis* o la transacción 0. Desde ese momento se transfirieron a las cuentas de los inversores originales del proyecto, en proporción a la cantidad aportada al mismo. Posteriormente, algunos de dichos IOTAs se fueron vendiendo a otros y así se estableció la red de intercambio.

Volviendo a Alicia y Bernardo, la siguiente figura ilustra las situaciones antes y después de que se produzca la transacción:

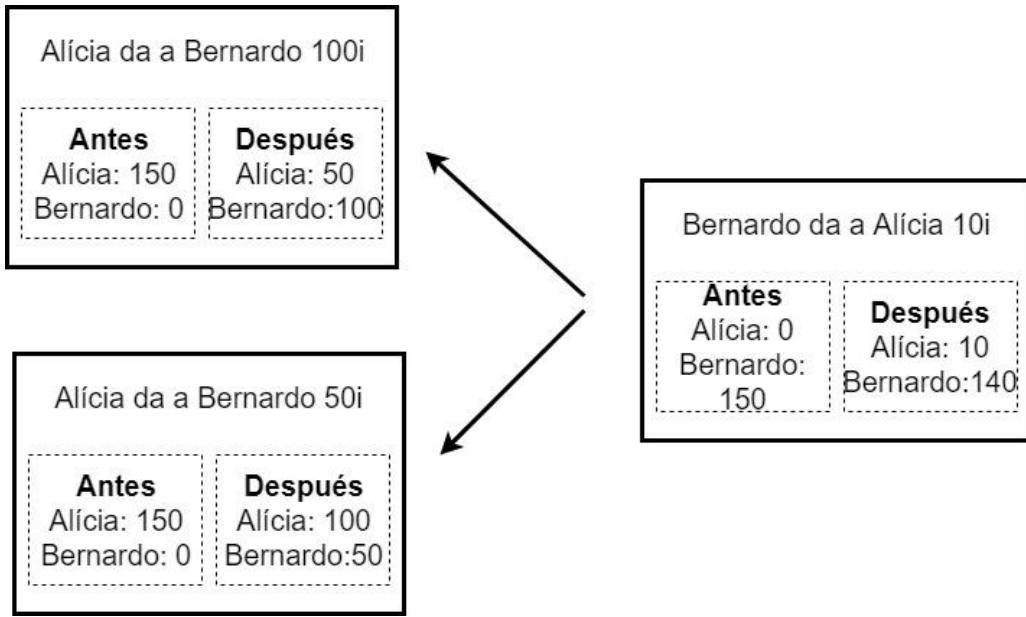
Alicia da a Bernardo 100i	
Antes Alícia: 100 Bernardo: 0	Después Alícia: 0 Bernardo:100

En otro momento, Carlos quiere hacer su propio pago. Ejecuta el algoritmo de selección de *tips*, y debe aprobar la transacción de Alicia para poder realizar la suya. Para ello, debe verificar que Alicia realmente tiene los 100i que está gastando. Para estar completamente seguro, debe registrar todas las transacciones aprobadas directa e indirectamente por la de Alicia, siguiendo el proceso hasta el *génesis*, quedando una lista similar a la siguiente:

1. Génesis crea 200i
2. Génesis da a Bernardo 20i
3. Génesis da a Alicia 80i
4. Génesis da a Carlos 5i
5. Bernardo da a Alicia 20i

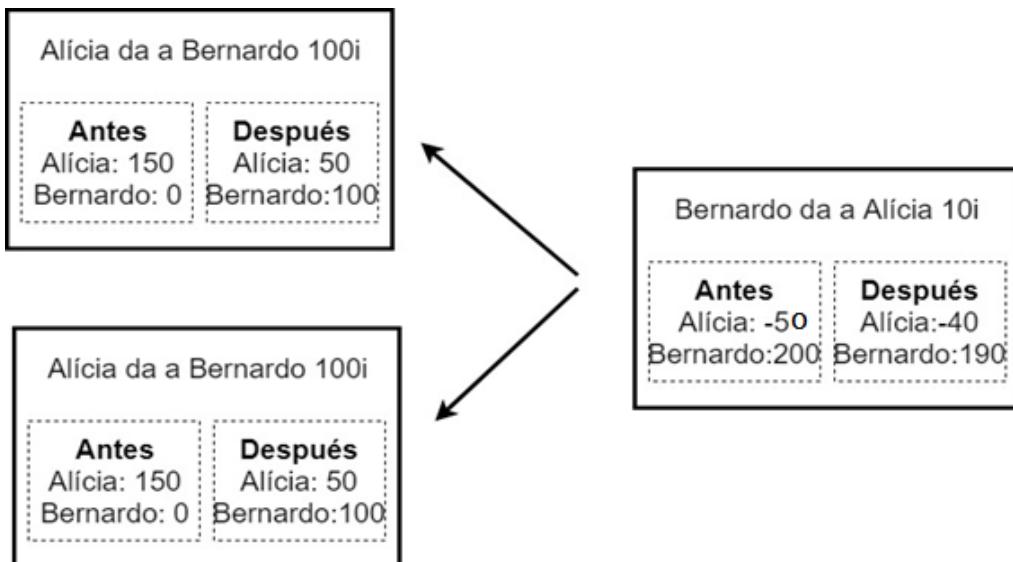
Cualquier lista que termina con 100i en la cuenta de Alicia y o en la de Bernardo es aceptable. Carlos ha de tener en cuenta el resto de los balances del sistema, para asegurarse de que no sean negativas: si cualquiera de las cuentas en las secciones “antes” o “después” son negativas, las transacciones son inválidas.

La situación se vuelve aún más interesante cuando se aprueban dos transacciones:



Este es un ejemplo de dos transacciones validas, correctamente verificadas por Bernardo en la suya propia al enviar 10i a Alicia.

A continuación, se muestra una situación “*illegal*”, Bernardo no puede validar las transacciones de Alicia porque resultan en un balance negativo en la cuenta de la misma. Si lo hace, estaría rompiendo las reglas del protocolo de IOTA y nadie aprobaría su nueva transacción.



Ésta última situación se denomina **doble gasto**, ya que Alicia gasta su dinero 2 veces. No ha roto el protocolo en ningún momento dado que tenía suficiente dinero para realizar cada transacción por separado. La solución a este problema es de nuevo el *Weighted Walk*. Eventualmente, uno de los dos pagos acumulará más peso que el

otro, y este será abandonado. Esto también implica que una transacción no puede ser considerada para ser validada inmediatamente después de ser lanzada, aunque pueda tener varios validadores potenciales, ya que puede ser parte de una serie de transacciones que serán abandonadas. Para asegurar que la transacción mandada es confirmada, se debe esperar a que su *confianza de confirmación* sea suficientemente alta.

La *confianza de confirmación* es una medida de una transacción sobre el nivel de aceptación del resto de la *Tangle*. Se computa de la siguiente manera:

1. Se ejecuta el algoritmo de selección de *tip* 100 veces.
2. Se cuentan cuántos de esos *tips* validan la transacción y se le llama A.
3. La confianza de confirmación de nuestra transacción es “A por ciento”

En otras palabras, es el porcentaje de *tips* que aprueban una transacción determinada. No todos los *tips* se consideran de la misma manera: a los *tips* más probables (de mayor

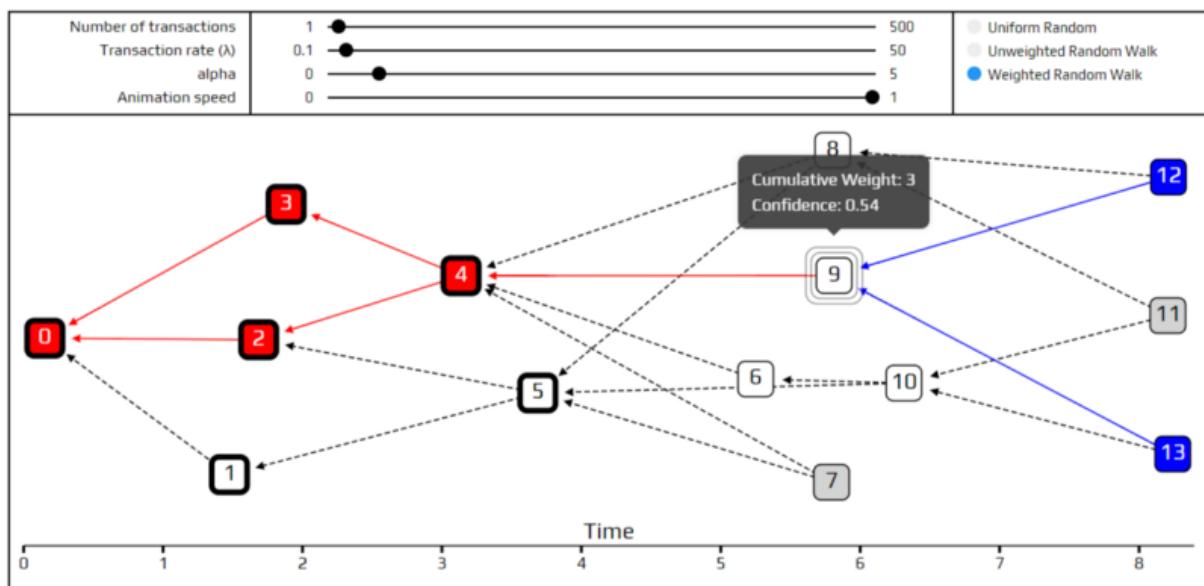


Figura 3.31: Preferencia de los tips con mayor peso acumulado en el Tangle

peso acumulado), se les da más importancia.

En la red anterior, la transacción 9 se aprueba por dos de los cuatro *tips* disponibles, lo que le daría un porcentaje de confianza del 50%. Pero dado que dichos *tips*, tienen aparentemente una mayor probabilidad de ser elegidos, eleva la confianza un poco por encima hasta llegar a un 54%.

Una vez que una transacción alcanza un valor límite de confianza, digamos 95% resulta muy difícil que dicha transacción sea expulsada del consenso. Se ha de especificar que, aunque muy improbable, es posible que esto se dé: si el realizador de la transacción tiene suficiente poder de computación, puede intentar un doble gasto.

Para ello, se debería referir a todas las transacciones anteriores que se puedan, tratando de aumentar al máximo el peso acumulado de la nueva transacción. Si tiene suficiente poder de computación, puede utilizar toda la red de IOTA para creerle y seguir su recorrido, reescribiendo de esta manera la historia de transacciones y teniendo éxito en el doble gasto.

Este escenario es posible solo si se pueden mandar más transacciones que todo el resto de la red combinada. Mientras que para redes maduras y activas esto no supone un riesgo considerable, es un problema real para la actual red de IOTA. No hay suficientes transacciones en el sistema para que este sea seguro ante un ataque concentrado de doble gasto.

Ante esta situación y dado que uno de los pilares en la filosofía de creación de IOTA es su escalabilidad, se ha implementado un sistema de consenso temporal y voluntario por razones de seguridad. Este mecanismo gira en torno a la figura del Coordinador, cada dos minutos una transacción milestone o *hito* se manda desde la Fundación IOTA, y todas las transacciones aprobadas por ella se consideran que tienen una confianza de confirmación del 100%. De esta manera, transacciones que intenten llevar a cabo doble gastos nunca serán validadas.

Actualmente, la utilización de dicho sistema es necesario como mecanismo de protección, mientras que la red de IOTA sigue creciendo, hasta alcanzar la actividad necesaria para que la red pueda ser totalmente segura actuando autónomamente en una manera 100% descentralizada. En este punto la Fundación de IOTA acabará con la figura del Coordinador y dejará que la red se gestione autónomamente, aumentando en varios órdenes de magnitud su eficiencia.

Capítulo 4 Solución Adoptada

En los capítulos anteriores se han detallado los diferentes componentes del sistema, la Raspberry Pi, el sensor SenseHat, el ordenador Linux, la base de datos MariaDB y se ha hecho una introducción a IOTA y a BigchainDB. En el capítulo actual se va a desarrollar como se implementan y configuran estos elementos para llevar a cabo la consecución del proyecto, así como un detalle de la metodología llevada a cabo en las distintas pruebas realizadas para comprobar la funcionalidad del sistema global.

4.1 Introducción

La arquitectura general del sistema propuesto sigue la siguiente distribución:

1) Captación de los datos:

A partir del sensor SenseHat unido a través de los pines de conexión con la placa Raspberry Pi. Para ello se desarrollará un programa en el lenguaje de programación Python 3.5 que recoja datos ambientales de presión, temperatura, humedad, aceleraciones, giroscopio, orientación y campo magnético.

2) Envío de los datos a máquina Linux:

A través del protocolo de comunicación M2M de MQTT. Se establecerá una conexión servidor-cliente y se procederá al envío de los datos de forma periódica.

3) Recepción de los datos:

En la máquina Linux, previamente suscrita al canal MQTT correspondiente. El programa cliente, una vez establecida satisfactoriamente la conexión con el bróker y recibidos los datos, realizará 3 acciones simultáneas por cada mensaje recibido (pasos 4.5 y 6).

4) Inserción en base de datos MariaDB

5) Inserción en base de datos Blockchain BigchainDB

6) Calculo de resumen estadístico temporal periódico y publicación en el Tangle de IOTA

Un esquema de la arquitectura general del sistema:

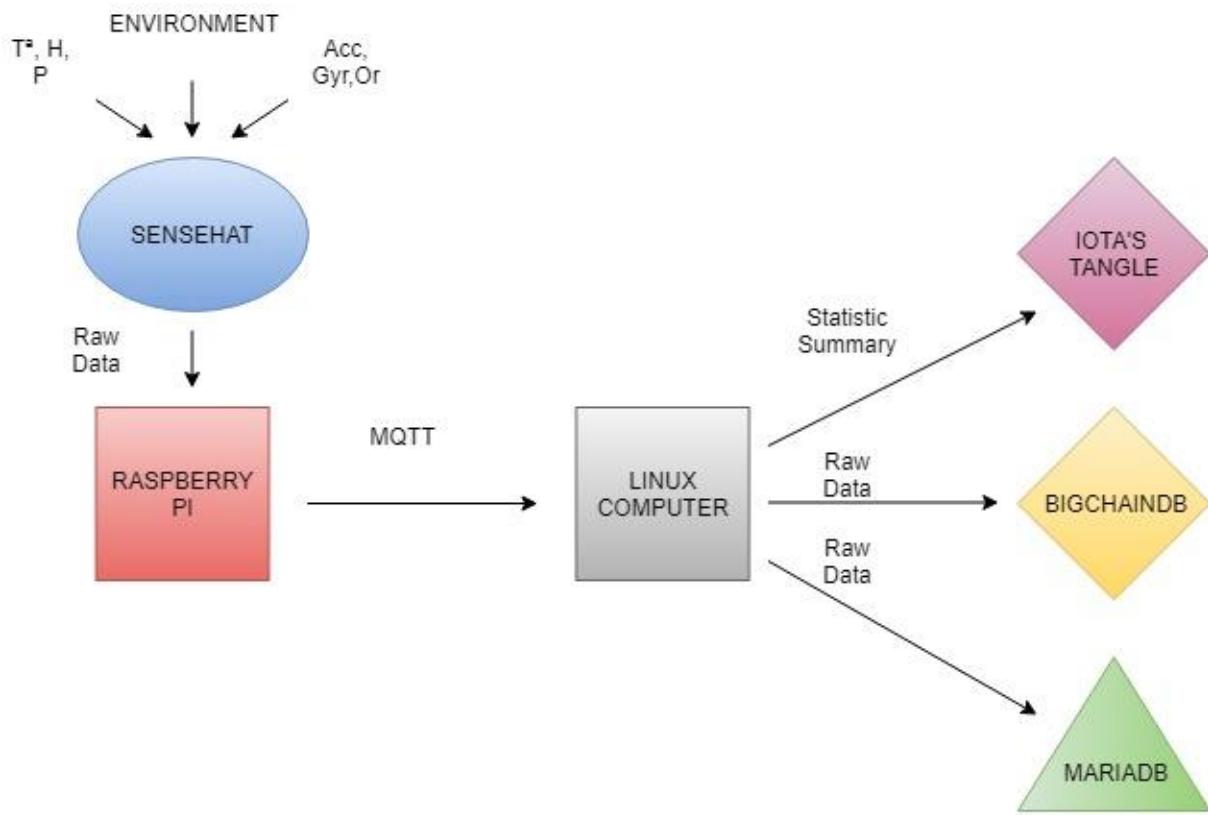


Figura 4.1: Arquitectura general de la solución propuesta

4.2 Configuración del Sistema

Para poder realizar el proceso anterior es necesario instalar y configurar una serie de programas, librerías y drivers para lograr la interacción satisfactoria entre las diversas partes del sistema.

4.2.1 Requisitos lado servidor

Con la Raspberry Pi encendida y conectada a internet vía ethernet o WIFI, se abre el comando del sistema y se procede a instalar las librerías y dependencias necesarias para Python 3.5, en concreto, se instalarán las correspondientes a SenseHat y los drivers para conectarse al servidor Paho MQTT, ya que el resto de las librerías vienen preinstaladas con la versión de Python utilizada.

4.2.1.1 Actualización del sistema y los paquetes del sistema

En primer lugar, se verifica que el sistema está en la última versión y no existen versiones nuevas de los paquetes instalados y preinstalados. La ejecución de los siguientes comandos servirá para tal propósito y, en caso de necesidad se aplicarán las actualizaciones correspondientes.

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

4.2.1.2 Librería de SenseHat:

Con la ventana de comando abierta, se escribe y ejecuta la siguiente línea de código:

```
sudo apt-get install sense-hat
```

4.2.1.3 librería Paho MQTT

```
pip install paho-mqtt
```

4.2.2 Requisitos lado Cliente

El lado cliente corresponde a la maquina Linux, cuyo sistema operativo ha sido previamente instalado y configurado. Como en el caso del lado servidor, habrá que instalar una serie de programas, librerías y drivers, pero antes de proceder con ello se comprueba que el sistema y los paquetes se encuentran actualizados, abriendo el comando del sistema y ejecutando las siguientes líneas:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

4.2.2.1 Librería Paho MQTT:

```
pip install paho-mqtt
```

4.2.2.2 MariaDB

4.2.2.2.1 Instalación de MariaDB

Para instalar MySQL en el ordenador, se ejecuta la siguiente línea de código en el terminal:

```
sudo apt-get install mysql-server
```

Este comando instalará el servidor MySQL y otros paquetes. Durante la instalación habrá que configurar una contraseña para la cuenta Root de MySQL.

A continuación, se instalará el módulo MySQLdb con interfaz Python para la base de datos:

```
sudo apt-get install python-mysqldb
```

4.2.2.2.2 Configuración de MariaDB

En primer lugar, se configurará MariaDB con un usuario y contraseña específico, después se accederá al servidor a través de la dirección <http://localhost/phpmyadmin> y se creará la base de datos a utilizar en el proyecto.

1) Configuración:

A continuación, se creará un usuario nuevo y una base de datos, a través del cliente mysql, ejecutando el siguiente código en el terminal:

```
mysql -u root -p
```

Se pedirá la contraseña y una vez introducida correctamente, se conectará a la base de datos a través de la cuenta root:

```
mysql> CREATE DATABASE iota;
```

Query OK, 1 row affected (0.02 sec)

```
mysql> CREATE USER 'iotauser'@'localhost' IDENTIFIED BY 'root';
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> USE iota;
```

Database changed

```
mysql> GRANT ALL ON iota.* TO 'iotauser'@'localhost';
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> quit;
```

Bye

A partir de estas líneas de código se ha creado un nuevo usuario “iotauser” para la base de datos y se le han concedido todos los privilegios al mismo sobre la base de datos “iota”. De esta manera, ya está lista para ser usada.

2) Creación

Una vez accedida a la base de datos a través de <http://localhost/phpmyadmin> e introduciendo el usuario y la contraseña correspondientes, se procederá a crear la tabla que se utilizará en el proyecto, que se llamará “Environment_Parameters”.

Dado que la información recibida es constante, se configurarán las columnas de cada tipo de dato recibido previamente, para después inyectar cada dato en su lugar correspondiente. Por tanto, se crearán las siguientes columnas, estableciendo el tipo de dato que se inyectará:

- “Data_Storage_Time_Stamp”: Marca temporal correspondiente a la inyección de la nueva cadena de datos.
- “Data_Collection_Time_Stamp”: Marca temporal correspondiente a la recogida de la cadena de datos inyectada
- “Temperature”
- “Pressure”
- “Humidity”
- “Pitch”
- “Roll”
- “Yaw”
- “Magnetic_Field_X”
- “Magnetic_Field_Y”
- “Magnetic_Field_Z”
- “Gyroscope_X”
- “Gyroscope_Y”
- “Gyroscope_Z”
- “Acceleration_X”
- “Acceleration_Y”
- “Acceleration_Z”

The screenshot shows the phpMyAdmin interface for the 'iota' database. The left sidebar lists databases (information_schema, iota, New, Environment_Parameters) and tables (Environment_Parameters). The main area displays the 'Structure' tab for the 'Environment_Parameters' table. The table has 17 columns:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	Data_Storage_Time_Stamp	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP		ON UPDATE CURRENT_TIMESTAMP
2	Data_Collection_Time_Stamp	timestamp			No	0000-00-00 00:00:00		
3	Temperature	float			No	None		
4	Pressure	float			No	None		
5	Humidity	float			No	None		
6	Pitch	float			No	None		
7	Roll	float			No	None		
8	Yaw	float			No	None		
9	Magnetic_Field_X	float			No	None		
10	Magnetic_Field_Y	float			No	None		
11	Magnetic_Field_Z	float			No	None		
12	Acceleration_X	float			No	None		
13	Acceleration_Y	float			No	None		
14	Acceleration_Z	float			No	None		
15	Gyroscope_X	float			No	None		
16	Gyroscope_Y	float			No	None		
17	Gyroscope_Z	float			No	None		

Figura 4.2: Tabla de inserción de datos ambientales en MariaDB

4.2.2.3 BigchainDB

4.2.2.3.1 Instalación de los drivers de Python para BigchainDB

1) Prerrequisitos:

- Versión de Python 3.5+
- Versión reciente del gestor de paquetes de Python pip
- Versión reciente de la dependencia *setuptools*:

pip3 install –upgrade setuptools

- Versión reciente de criptografía y cripto-condiciones:

Sudo apt-get install python3-dev libssl-dev libffi-dev

3) Instalando el Driver

pip3 install bigchaindb-driver

4.2.2.3.2 Levantar nodo de BigchainDB a través de Docker-Compose

1) Prerequisites:

- Versión actualizada de Docker:

```
sudo apt install docker.io
```

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

- Versión actualizada de Docker-compose, para instalarla y aplicar permisos de ejecución:

```
Sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

2) Levantar nodo:

```
docker-compose build bigchaindb
```

```
docker-compose up -d bdb
```

Estas líneas descargarán e instalarán los paquetes y dependencias de MongoDB, Tendermint y BigchainDB necesarias para ejecutar y utilizar apropiadamente un nodo BigchainDB.

Una vez correctamente instaladas, para correr el nodo BigchainDB en primer plano, ejecutar en el directorio creado:

```
make run
```

Si todo ha salido bien, se verá lo siguiente en el terminal(imagen):

```

bigchaindb_1  | [2018-06-05 08:01:30] [INFO] (bigchaindb.backend.localmongodb.schema) Create `blocks` table. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:30] [INFO] (bigchaindb.backend.localmongodb.schema) Create `metadata` table. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:30] [INFO] (bigchaindb.backend.localmongodb.schema) Create `validators` table. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:30] [INFO] (bigchaindb.backend.localmongodb.schema) Create `pre_commit` table. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:31] [INFO] (bigchaindb.backend.localmongodb.schema) Create `transactions` secondary index. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:32] [INFO] (bigchaindb.backend.localmongodb.schema) Create `assets` secondary index. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:32] [INFO] (bigchaindb.backend.localmongodb.schema) Create `assets` secondary index. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:32] [INFO] (bigchaindb.backend.localmongodb.schema) Create `utxos` secondary index. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:32] [INFO] (bigchaindb.backend.localmongodb.schema) Create `pre_commit` secondary index. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:33] [INFO] (bigchaindb.backend.localmongodb.schema) Create `validators` secondary index. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:33] [INFO] (bigchaindb.commands.bigchaindb) Starting BigchainDB main process. (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:33] [INFO] (bigchaindb.tendermint.commands)
*****
*      *      *      *
*      BIGCHAINDB 2.0.1ER
*      *      *      *
* codename "fluffy cat"
* Initialization complete. BigchainDB Server is ready and waiting.
* 
* You can send HTTP requests via the HTTP API documented in the
* BigchainDB Server docs at:
*   https://bigchaindb.com/http-api
* 
* Listening to client connections on: 0.0.0.0:9984
* 
*****
(MainProcess - pid: 7)
[2018-06-05 08:01:33] [DEBUG] (asyncio) Using selector: EpollSelector (ws      - pid: 26)
[2018-06-05 08:01:33] [DEBUG] (bigchaindb.config_utils) System already configured, skipping autoconfiguration (MainProcess - pid
: 7)
bigchaindb_1  | [2018-06-05 08:01:33] [DEBUG] (bigchaindb.backend.connection) Connection: <class 'bigchaindb.backend.localmongodb.connection.LocalMongoDBConnection'> (MainProcess - pid: 7)
bigchaindb_1  | [2018-06-05 08:01:33] [DEBUG] (gunicorn.error) Current configuration:
bigchaindb_1  | config: None
bigchaindb_1  | bind: ['0.0.0.0:9984']
bigchaindb_1  | backlog: 2048
bigchaindb_1  | workers: 9
bigchaindb_1  | worker_class: sync
bigchaindb_1  | threads: 1
bigchaindb_1  | worker_connections: 1000
bigchaindb_1  | max_requests: 0

```

Figura 4.3: Prueba de operatividad de nodo BigchainDB

Lo que indicará que el nodo BigchainDB se encuentra corriendo y operativo. Y se podrá acceder a la página HTTP correspondiente al puerto configurado por defecto 9984, escribiendo en el navegador: localhost:9984

Además, se puede comprobar que tanto el nodo BigchainDB, como Tendermint y MongoDB se encuentran operativos a través del comando:

Docker-compose ps

Que devuelve lo siguiente por pantalla:

Name	Command	State	Ports
bigchaindb_bigchaindb_b_1	.ci/entrypoint.sh	Up (healthy)	0.0.0.0:32769->4665 8/tcp, 0.0.0.0:9984->9984/tcp, 0.0.0.0:9985->9985/tcp
bigchaindb_mongodb_1	docker-entrypoint.sh mongod	Up	0.0.0.0:32768->2701 7/tcp
bigchaindb_tendermin_t_1	sh -c tendermint init && t ...	Up	0.0.0.0:46656->4665 6/tcp, 0.0.0.0:46657->46657/tcp

Figura 4.4: Imágenes de BigchainDB, Tendermint y MongoDB levantadas

4.2.2.3 Generar Claves Criptográficas

Para conectar con el nodo local y poder realizar transacciones, es necesario contar con claves criptográficas pública y privada. Estas se pueden generar fácilmente a través de la función `generate_key_pair()`, disponible en los drivers de Python de BigchainDB, y que devuelve una lista de dos cadenas de caracteres alfanuméricos que corresponden a las claves mencionadas.

Dichas claves, deben ser almacenadas en un lugar seguro pues son la identificación del usuario en el ambiente BigchainDB y resultan imprescindibles para enviar y recibir transacciones.

4.2.2.4 IOTA

4.2.2.4.1 Generar Seed

La Seed en IOTA es la manera de identificar a cada participante del Tangle, uniendo en una única cadena de 81 caracteres alfanuméricos el usuario y la contraseña, por lo tanto, es crucial generarla y almacenarla de manera segura, ya que es la llave a toda la información y tokens del usuario. Hay una gran cantidad de generadores online, pero se han dado muchos casos de robo de dichas sedes y, por tanto, el método recomendado para generar la Seed en dispositivos Linux, es través de la terminal.

Se correrán los siguientes comandos:

```
cat /dev/urandom |tr -dc A-Z9|head -c${1:-81}
```

4.2.2.4.2 Instalar Drivers de Python para IOTA, PyOTA

Hay disponible una versión de PyOTA, con extensión del lenguaje C, que mejora el desempeño de las características criptográficas hasta en 60 veces, por tanto, esa será la que se instalará:

```
pip install pyota[ccurl]
```

4.2.2.5 Instalar modulo Pandas

Para calcular el resumen estadístico temporal que se publicará al Tangle, se inyectará cada recogida de datos en un dataframe de pandas y en el tiempo especificado, se computará el resumen y reiniciará el dataframe. Por ello, se instalará pandas con el siguiente comando:

```
pip3 install pandas
```

4.2.3 Captación de Datos – Lado Servidor

En primer lugar, se realizarán todos los pasos necesarios para captar satisfactoriamente los datos buscados. Para ello, una vez instalado y configurado el sistema operativo en la Raspberry Pi, según los pasos marcados en (insertar localización en el documento), se instalarán las librerías necesarias, se conectará adecuadamente el sensor SenseHat a la Raspberry Pi, se creará el programa y se ejecutará para comprobar su correcto funcionamiento.

4.2.3.1 Conectar sensor SenseHat

La conexión entre el sensor y la Raspberry Pi resulta extremadamente sencilla, pues SenseHat está creado específicamente para ser usado en dicha placa y los pasos a seguir se describen a continuación:

- 1) Sacar sensor de la caja:

Y el resto de los elementos que vienen con ella, 4 tornillos y correspondientes roscas, y el conector de pines GPIO.

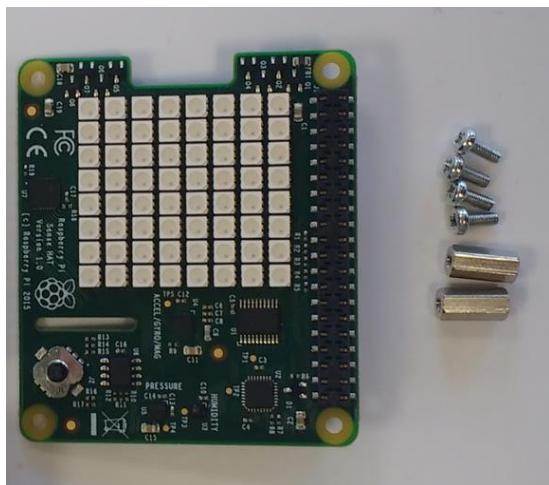
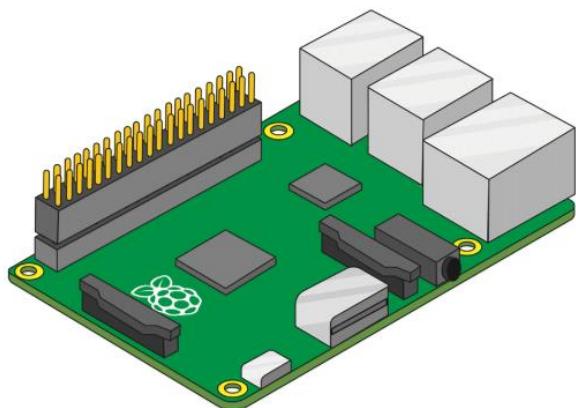


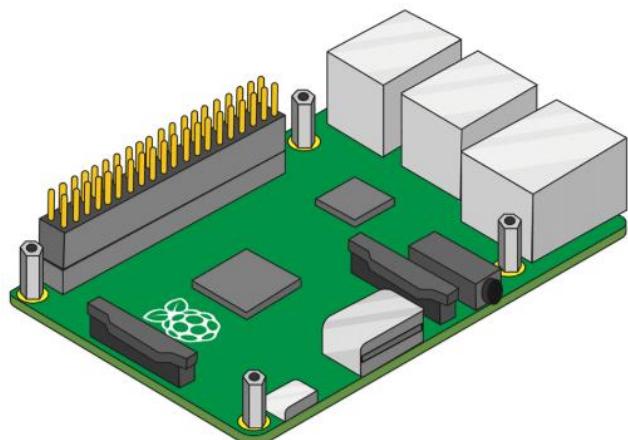
Figura 4.5: SenseHat y tornillos

- 2) Colocar el puerto de pines GPIO:

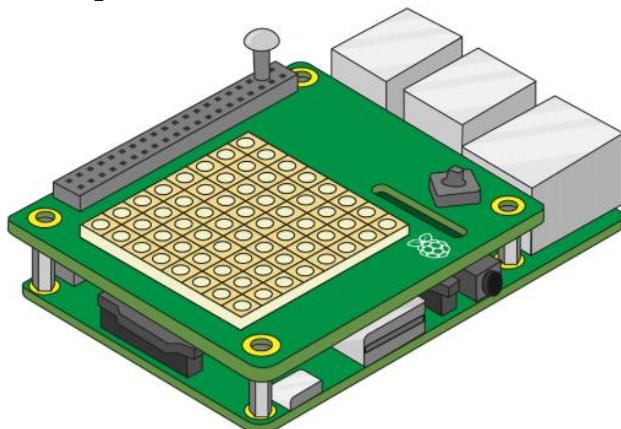
En la Raspberry pi, según la siguiente ilustración.



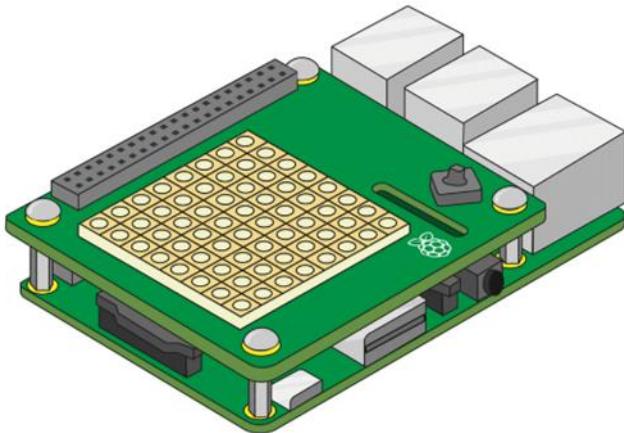
- 3) Colocar roscas en sus respectivos huecos:
Y ajustar tornillos por debajo.



- 4) Conectar placa:



5) Ajustar tornillos finales:



4.2.4 Desarrollo del programa

Los programas desarrollados se pueden encontrar en el repositorio de GitHub:

[“rromanss23/Store-Sensor-Data-in-BigchainDB-MariaDB-and-Publish-Statistic-Summary-to-IOTA-s-Tangle”](https://github.com/rromanss23/Store-Sensor-Data-in-BigchainDB-MariaDB-and-Publish-Statistic-Summary-to-IOTA-s-Tangle).

El programa correspondiente a esta parte del proyecto es `pubsensordata.py` y el código desarrollado es el siguiente:

```
from datetime import datetime

from sense_hat import SenseHat

from time import sleep

import paho.mqtt.client as mqtt

import paho.mqtt.publish as publish

import time

import json

sense = SenseHat()
```

```

##### Logging Settings #####
TEMP_H=True
TEMP_P=False
HUMIDITY=True
PRESSURE=True
ORIENTATION=True
ACCELERATION=True
MAG=True
GYRO=True
DELAY=2

##### MQTT Settings #####
Broker = "localhost"
sub_topic = "test/instructions"      # receive messages on this topic
pub_topic = "test/data"             # send messages to this topic

##### FUNCTIONS #####
def get_sense_data():
    sense_data={}
    if TEMP_H:
        sense_data["Temperature"] = sense.get_temperature_from_humidity()
    if TEMP_P:
        sense_data["Temperature"] = sense.get_temperature_from_pressure()
    if HUMIDITY:
        sense_data["Humidity"] = sense.get_humidity()

```

```

if PRESSURE:

    sense_data["Pressure"] = sense.get_pressure()

if ORIENTATION:

    o = sense.get_orientation()

    sense_data["Yaw"] = o["yaw"]

    sense_data["Pitch"] = o["pitch"]

    sense_data["Roll"] = o["roll"]

if MAG:

    mag = sense.get_compass_raw()

    sense_data["Magnetic_Field_x"] = mag["x"]

    sense_data["Magnetic_Field_y"] = mag["y"]

    sense_data["Magnetic_Field_z"] = mag["z"]

if ACCELERATION:

    acc = sense.get_accelerometer_raw()

    sense_data["Acceleration_x"] = acc["x"]

    sense_data["Acceleration_y"] = acc["y"]

    sense_data["Acceleration_z"] = acc["z"]

if GYRO:

    gyro = sense.get_gyroscope_raw()

    sense_data["Gyroscope_x"] = gyro["x"]

    sense_data["Gyroscope_y"] = gyro["y"]

    sense_data["Gyroscope_z"] = gyro["z"]

```

```

sense_data[ "Data_Collection_Time_Stamp" ] = time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime())

return sense_data

##### MQTT Section #####
# when connecting to mqtt do this

def on_connect(client, userdata, flags, rc):

    print("Connected with result code "+str(rc))

    client = mqtt.Client()

    client.on_connect = on_connect

    client.connect(Broker, 1883, 60)

    client.loop_start()

while True:

    client.publish(pub_topic, json.dumps(get_sense_data()))

    time.sleep(DELAY)

```

1) Configuración de MQTT:

Estableciendo adecuadamente los siguientes parámetros:

- Broker: con la dirección IP del bróker o localhost
- pub_topic: nombre del topic de publicación de los datos
- sub_topic: nombre del topic de subscripción

sub_topic se configurará en caso de querer suscribirse a un topic determinado una vez realizada la conexión bróker-cliente, esto puede ser útil para el intercambio de mensajes que comprueben un flujo satisfactorio de información.

En el caso del proyecto, se configura:

- Broker = 'localhost'
- pub_topic = 'sensordata'

2) Establecer datos a recoger:

Se configuran las siguientes variables booleanas y si su valor es True, se recogerán los datos correspondientes a:

- TEMP_H: Temperatura (°C) recogida por el sensor de humedad
- TEMP_P: Temperatura (°C) recogida por el sensor de presión.
- HUMIDITY: Porcentaje de humedad en el ambiente.
- PRESSURE: Presión en milibares.
- ORIENTATION: Orientación de la placa con respecto a los ejes espaciales X, Y, y Z
- ACCELERATION: Aceleración en m/s² la placa con respecto a los ejes espaciales X, Y y Z.
- MAG: Orientación del campo magnético terrestre respecto a los ejes espaciales X, Y y Z.
- DELAY: Frecuencia de recogida de los datos en segundos

En el caso del presente proyecto, se establecen todos los parámetros con valor True, lo que implica la recogida de todos los datos. Y el valor de DELAY = 5, para hacer una captación cada dos segundos.

Una vez realizados estos pasos, se accede al directorio donde se encuentra guardado el programa y se ejecuta la siguiente línea en el comando del sistema:

```
python3 pubsensor.py
```

4.2.5 Recepción de Datos – Lado Cliente

Una vez configurado adecuadamente el ambiente en el lado cliente según el punto 5.4, el sistema está listo para recibir y gestionar la información recibida del servidor de manera adecuada, a través de los correspondientes programas.

En primer lugar, se desarrollará un programa que se suscriba al topic MQTT por el que se publican los datos del sensor. Posteriormente se desarrollará un programa para cada servidor en el que se inyectarán los datos: MariaDB, BigchainDB e IOTA.

4.2.5.1 Cliente MQTT

```
import paho.mqtt.client as mqtt

from tobdb import send_to_bdb

from tomysql import store_mysql

from toiotia import send_to_tangle

import json
```

```

### MQTT SETTINGS ###

mqtt_broker = "192.168.1.108" # MQTT broker IP Address

mqtt_port = 1883 # MQTT port

sub_topic = "test/data" # receive messages on this topic

pub_topic = "test/instrutions" # send messages to this topic

### WHEN CONNECTION IS DONE ###

def on_connect(client, userdata, flags, rc):

    print("Connected with result code " + str(rc))

    client.subscribe(sub_topic)

### WHEN RECEIVING A MESSAGE ###

def on_message(client, userdata, msg):

    global first_message

    print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))

    sensor_data = json.loads(msg.payload)

    # Send data to BigchainDB Client

    send_to_bdb(sensor_data)

    # Send data to MariaDB client

    store_mysql(sensor_data)

    # Send data to IOTA's Tangle

    send_to_tangle(sensor_data)

client = mqtt.Client()

client.on_connect = on_connect

client.on_message = on_message

```

```

client.connect(mqtt_broker, mqtt_port, 60)

### Start the MQTT forever loop

client.loop_forever()

```

Al ejecutar el programa mediante el comando:

python3 mqttclient.py

Se realizará la conexión con MQTT y se empezarán a recibir los datos enviados desde la Raspberry Pi. Dichos datos se mandaron codificados en JSON para que fuesen aún más ligeros y, por tanto, rápidos de transmitir. El programa también se encargará de decodificarlos en un diccionario clave-valor tradicional y pasarlo a cada función que inyectará los datos en los distintos servidores.

4.2.5.2 MariaDB

```

import time

import MySQLdb as mdb

### MYSQL SETTINGS ###

data_base_hostname = ""          # MySQL host ip address or name

data_base_database = ""          # MySQL database name

data_base_username = ""          # MySQL database user name

data_base_password = ""          # MySQL database password

### MYSQL SETUP ###

data_base = mdb.connect(data_base_hostname, # connect with MySQL database

                       data_base_username,

                       data_base_password,

                       data_base_database)

data_base_cursor = data_base.cursor()      # prepare a cursor object

```

```

def store_mysql(sensor_data):

    vars_to_sql = []
    keys_to_sql = []
    data_list = []
    data_list = sensor_data

    for key, value in data_list.items():

        if key == 'Data_Storage_Time_Stamp':

            value = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())

            vars_to_sql.append(value)

            keys_to_sql.append(key)

    keys_to_sql = ', '.join(keys_to_sql)

    try:

        # Execute the SQL command

        queryText = "INSERT INTO Environment_Parameters(%s) VALUES %r"

        queryArgs = (keys_to_sql, tuple(vars_to_sql))

        data_base_cursor.execute(queryText % queryArgs)

        print('Successfully Added record to mysql')

        data_base.commit()

    except mdb.Error as e:

        try:

            print ("MySQL Error [%d]: %s" % (e.args[0], e.args[1]))

        except IndexError:

            print ("MySQL Error: %s" % str(e))

```

```

# Rollback in case there is any error

data_base.rollback()

print('ERROR adding record to MYSQL')

```

Dado que al codificarse los datos en formato JSON, se desordenan las parejas clave-valor, este programa se encarga de conectar con la base de datos creada y configurada en el punto 5.2.2.1, y de iterar cada diccionario recibido e insertar cada valor de cada parámetro en la columna correspondiente, previamente configurada en el punto 5.2.2.2.

4.2.5.3 BigchainDB

```

from bigchaindb_driver import BigchainDB

import time

### BigchainDB Connection

bdb_root_url = 'http://localhost:9984'

bdb = BigchainDB(bdb_root_url)

### BigchainDB Keys

victor_priv = 'YOURPRIVATEKEYHERE'

victor_pub = 'YOURPUBLICKEYHERE'

### Raspberry Data

raspberry_asset = {

    'data': {

        'raspberry pi': {

            'serial_number': 'RPI01',

            'owner': 'UPM'

        },

    }

},

```

```

    },
}

raspberry_asset_metadata = {

    'Environment Parameters': 'Here goes sensor data'

}

### Sending Data to BigchainDB

def send_to_bdb(sensor_data):

    raspberry_asset['data']['raspberry pi']['Storage TimeStamp'] = str(time.strftime('%Y-%m-%d %H:%M:%S', time.localtime()))

    raspberry_asset_metadata = sensor_data

    prepared_creation_tx = bdb.transactions.prepare(
        operation='CREATE',
        signers=victor_pub,
        asset=raspberry_asset,
        metadata=raspberry_asset_metadata
    )

    fulfilled_creation_tx = bdb.transactions.fulfill(
        prepared_creation_tx,
        private_keys=victor_priv
    )

    print(fulfilled_creation_tx)

    bdb.transactions.send(fulfilled_creation_tx)

```

Este programa se encarga de recibir cada diccionario de datos, actualizar los datos correspondientes añadiendo una marca temporal del momento en el que se almacenan dichos datos y crear un nuevo activo en la base de datos de BigchainDB, cuyos datos son la Raspberry con su número de serie y el propietario; y los metadatos serán los datos ambientales recibidos en cada mensaje.

4.2.5.4 IOTA

```
from iota import Address, ProposedTransaction, Tag, Transaction, Iota, TryteString, json
import pandas as pd
import json
import time
import iota
import threading
##### IOTA SETTINGS #####
SEED = 'PUTYOURSEEDHERE999999999999999999999999999999999999999999999999999999999'
api = Iota('http://173.212.218.8:14265/', SEED) # POW node, SEED IS NEEDED
TO DO THE SIGNING
tag = iota.Tag('PUTYOURTAGHERE') # Tag for in the message.
address = api.get_new_addresses(0, 1)['addresses'] # Generates a list of 1
address
##### PANDAS DATA FRAME SET #####
data_frame = pd.DataFrame()
#####
def send_to_iota(sensor_data, message):
    global data_frame
    data_decoded = sensor_data
    df1 = pd.DataFrame([data_decoded], columns=data_decoded.keys())
```

```

data_frame = pd.concat([data_frame, df1])

print(data_frame)

if message:

    thread.start()

    message = False

return message

def wait_time():

    while True:

        time.sleep(10)

        send_summary()

def send_summary():

    global data_frame

    new_data = {}

    ## IOTA'S IMPLEMENTATION

    data_summary = data_frame.describe([]).round(4)

    print(data_summary)

    dict_summary = data_summary.to_dict()

    for key, value in dict_summary.items():

        new_data["Num"] = int(value.pop("count"))

        new_data[key] = list(value.values())

    data_frame = pd.DataFrame()

    json_summary = json.dumps(new_data)

    print(json_summary)

    print(len(json_summary))

```

```

trytes = TryteString.from_string(json_summary) # Code Json data in trytes

print('Message sent to tangle:', trytes.decode()) # shows decoded msg sent to tangle

print('sleep...')

time.sleep(1)

print(api)

t0 = time.time()

api.send_transfer(
    depth=3,
    transfers=[ProposedTransaction(address=Address(Address(address[0], )),
                                   
                                   value=0,
                                   tag=tag,
                                   message=trytes, ), ], )

t1 = time.time()

print(
    'Time:', time.strftime('%H:%M:%S', time.localtime()), ' data transferred! in:',
    round((t1 - t0), 1), 'seconds ')
    
### THREAD FOR COMPUTING AND SENDING DATA SUMMARY ###

thread = threading.Thread(target=wait_time)

```

Por último, este programa se encarga de crear un dataframe en pandas, e inyectar cada diccionario de datos, pasado un tiempo especificado, calculará el resumen estadístico temporal constituido por: número de inyecciones desde el último resumen, media, mediana, desviación típica, máximos y mínimos. Además, una vez calculado dicho resumen, conectará con un nodo completo de IOTA y lo publicará, mostrando por pantalla el tiempo que se ha tardado en publicar la transacción, que corresponderá al tiempo que ha tardado el nodo completo en realizar el PoW.

4.2.6 Vida de una Recogida de Datos

Una vez configurados correctamente tanto la Raspberry (el servidor del sistema) y la maquina Linux (el cliente del sistema), y ejecutados los programas: pubsql.py en el primero y mqttclient.py en el segundo, da comienzo la recogida y flujo de datos. En este punto se mostrará el recorrido de una colección de datos desde su recogida pasando por todos los estadios del sistema.

- 1) Colección y envío de los datos ambientales:

Accediendo al directorio correspondiente y escribiendo en el terminal de la Raspberry Pi:

```
python3 pubsql.py
```

Se ejecuta el programa desarrollado y en periodos de 5 segundos, se toma una muestra de datos ambientales y se crea un diccionario clave-valor en el que se asocia la valoración cuantitativa que realiza el sensor SenseHat a la clave correspondiente.

```
root@UDPRBP02:~/Escritorio/HAT/Data# python pubsql.py
Connected with result code 0
Data recorded: {'Data_Collection_Time_Stamp': '2018-06-20 11:30:08', 'Temperature': 38.109092712402344, 'Acceleration_y': 0.6851606369018555, 'Acceleration_x': 0.015274092555046082, 'Gyroscope_x': 0.004145707935694833, 'Yaw': 151.26422437817544, 'Magnetic_Field_y': -9.62348747253418, 'Humidity': 29.86866569519043, 'Pressure': 944.28466796875, 'Magnetic_Field_z': 4.333179473876953, 'Magnetic_Field_x': -18.230669021606445, 'Pitch': 359.1934691180509, 'Gyroscope_z': 0.0025741183198988438, 'Gyroscope_y': -0.0034309150651097298, 'Roll': 43.42333745970627, 'Acceleration_z': 0.7215863466262817}
Data recorded: {'Data_Collection_Time_Stamp': '2018-06-20 11:30:13', 'Temperature': 38.21818161010742, 'Acceleration_y': 0.6841894388198853, 'Acceleration_x': 0.014061863534152508, 'Gyroscope_x': 0.002639155834913254, 'Yaw': 151.5494530779152, 'Magnetic_Field_y': -23.936538696289062, 'Humidity': 29.234420776367188, 'Pressure': 944.28271484375, 'Magnetic_Field_z': 9.98161506652832, 'Magnetic_Field_x': -45.10332489013672, 'Pitch': 358.3445476275136, 'Gyroscope_z': 0.0014231132809072733, 'Gyroscope_y': -0.002239307388663292, 'Roll': 44.31297301419501, 'Acceleration_z': 0.7240233421325684}
Data recorded: {'Data_Collection_Time_Stamp': '2018-06-20 11:30:18', 'Temperature': 38.21818161010742, 'Acceleration_y': 0.6863746047019958, 'Acceleration_x': 0.015274092555046082, 'Gyroscope_x': 0.0027222931385640283, 'Yaw': 151.66381776537787, 'Magnetic_Field_y': -26.566672760009766, 'Humidity': 29.375364303588867, 'Pressure': 944.298095703125, 'Magnetic_Field_z': 10.941874504089355, 'Magnetic_Field_x': -49.98472595214844, 'Pitch': 357.67050528662895, 'Gyroscope_z': 0.0019204993732273579, 'Gyroscope_y': -0.0011720238253474236, 'Roll': 45.413302829105724, 'Acceleration_z': 0.72353595495224}
Data recorded: {'Data_Collection_Time_Stamp': '2018-06-20 11:30:24', 'Temperature': 38.18181610107422, 'Acceleration_y': 0.6856462359428406, 'Acceleration_x': 0.014546754769980907, 'Gyroscope_x': 0.002293005585670471, 'Yaw': 151.80811234178356, 'Magnetic_Field_y': -26.727863311767578, 'Humidity': 30.044845581054688, 'Pressure': 944.290771484375, 'Magnetic_Field_z': 11.923676490783691, 'Magnetic_Field_x': -50.779727935791016, 'Pitch': 357.589716315022, 'Gyroscope_z': 0.0018245556857436895, 'Gyroscope_y': -0.00018000509589910507, 'Roll': 45.18624343543241, 'Acceleration_z': 0.722804844379425}
```

Figura 4.6: Visualización en la terminal de la Raspberry Pi de los datos recogidos

Después, se codifica el diccionario creado en formato JSON, que es un formato estándar para el intercambio de información en entornos IoT, y se publica vía MQTT a través del topic establecido, el formato se muestra en la siguiente imagen:

- 2) Recepción en el cliente:

El cliente se encuentra escuchando por el topic de subscripción y cuando recibe el mensaje lleva a cabo una serie de acciones consecutivas:

- Muestra el mensaje recibido (msg.payload, imagen) y el nivel de servicio de calidad de la transmisión de los datos.
- Decodifica el mensaje JSON en un diccionario.

- Pasa el valor de dicha variable a las funciones importadas de los programas auxiliares que se encargan de inyectar la información en los servidores correspondientes, en sus formatos específicos.

```
vroman@pcudp05:~/HAT/programas$ python3 mqtt.py
Connected with result code 0
test/data 0 b'{"Data_Collection_Time_Stamp": "2018-06-20 11:31:00", "Temperature": 38.21818161010742, "Acceleration_y": 0.6837038993835449, "Acceleration_x": 0.014304309152066708, "Gyroscope_x": 0.004105187952518463, "Yaw": 151.5492208514834, "Magnetic_Field_y": -27.209548950195312, "Humidity": 31.060279846191406, "Pressure": 944.274658203125, "Magnetic_Field_z": 13.4045991897583, "Magnetic_Field_x": -50.68513488769531, "Pitch": 357.405392561212, "Gyroscope_z": 0.0008658710867166519, "Gyroscope_y": -0.0009315479546785355, "Roll": 43.636521324080704, "Acceleration_z": 0.7225611209869385}'
```

Figura 4.7: Datos recibidos en el cliente Linux

3) Inyección en BigchainDB

Aquí se lleva a cabo todo el proceso de inyección en la base de datos Blockchain BigchainDB, desde la conexión con el nodo local, hasta la creación del activo que supone la nueva muestra de datos.

- Primero se conecta con el nodo local, con URL: http/localhost:9984, es decir, el nodo se encuentra corriendo en la máquina utilizada, de ahí la IP= localhost y se realiza la comunicación a través del puerto 9984, que es el configurado por defecto y el recomendado por los desarrolladores.

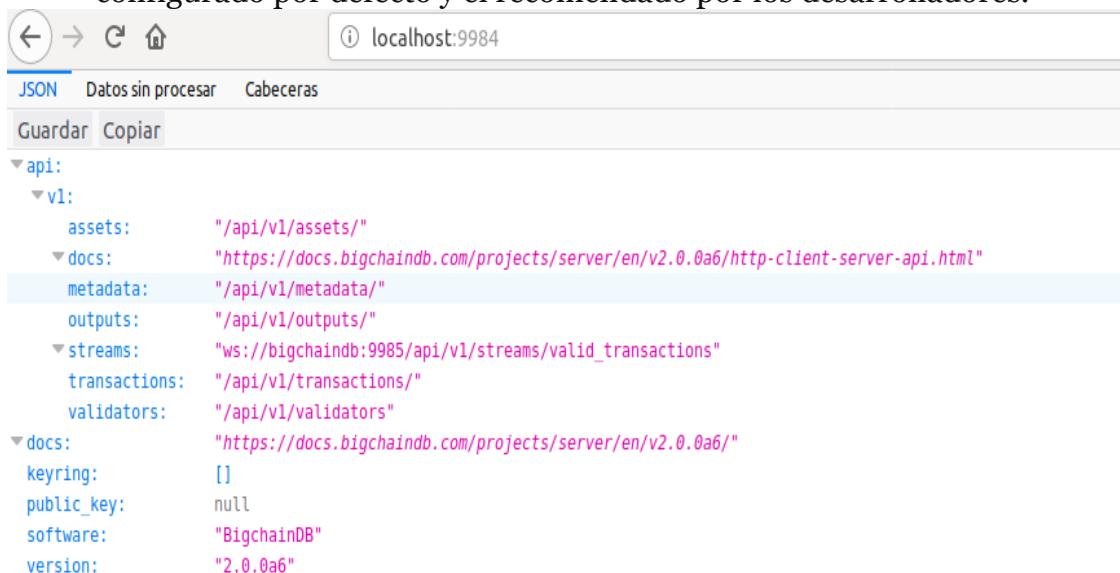


Figura 4.8: Visualización a través de HTTP del nodo levantado en el cliente Linux

- El mensaje plantilla que se inyectará en la base de datos está definido en la estructura del programa desarrollado. Por cada mensaje recibido, se actualizan los datos de dicho activo, y el contenido del mensaje, es decir, los valores de las magnitudes ambientales medidas se almacenan en forma de metadatos. El resultado de la actualización, con los nuevos metadatos, será lo que se registre en BigchainDB.

- Se prepara la transacción a enviar, definiéndose la operación (operation = CREATE), firmándose la misma con la clave pública (signers=victor_pub), estableciendo el activo a transferir(asset=raspberry_asset) y configurando los metadatos como los datos ambientales recibidos (metadata=raspberry_asset_metadata).
- Se completa la transacción preparada, firmándola con la clave privada (victor_priv).
- Se envía la transacción, y esta puede rastrearse a partir de su ID. Una de las formas es a través de la página HTTP, con la siguiente dirección: http://localhost:9984/api/v1/transactions/tx_ID. La correspondiente a la colecta de datos que se sigue en este punto se muestra a continuación.

```

{
  "id": "c9aa71f7f60a2314a7b9eb0787b3578fcffd4acc0df96cd1959f5abb7",
  "asset": {
    "data": {
      "raspberry pi": {
        "serial_number": "RPI01",
        "owner": "UPM",
        "StorageTimeStamp": "2018-06-20 11:31:00"
      }
    },
    "version": "2.0"
  },
  "inputs": [
    {
      "0": {
        "owners_before": "9Dw9Pc3P6zA74Dtt9uu0aDnUpHWEsxJoc7gX8Tfqrx8W",
        "fulfills": null,
        "fulfillment": "pGSAIHatr8mun9r-Wsxau0q7we_XhTZ_zGUxViEsDpZNga7XgUAKKz-emUjgRAM6cPtpVWP31xY1H6aMlsBha0HJYU59i2B4MB8YTquhrUe6vtmBT5ffsGPfxdCZU_layV8U1sP"
      }
    }
  ],
  "outputs": [
    {
      "0": {
        "public_keys": "9Dw9Pc3P6zA74Dtt9uu0aDnUpHWEsxJoc7gX8Tfqrx8W",
        "condition": {
          "details": {
            "type": "ed25519-sha-256",
            "public_key": "9Dw9Pc3P6zA74Dtt9uu0aDnUpHWEsxJoc7gX8Tfqrx8W",
            "uri": "ni://sha-256;qQD3knoLi16iFUtKI0-6NUn0xmJKKENN4eEDyrmP2FK?fpt=ed25519-sha-256&cost=131072",
            "amount": "1"
          }
        },
        "operation": "CREATE"
      }
    }
  ],
  "metadata": {
    "Data_Collection_Time_Stamp": "2018-06-20 11:31:00",
    "Temperature": 38.21818161010742,
    "Acceleration_y": 0.6837038993835449,
    "Acceleration_x": 0.014304309152066708,
    "Gyroscope_x": 0.004105187952518463,
    "Yaw": 151.5492208514834,
    "Magnetic_Field_y": -27.209548950195312,
    "Humidity": 31.060279846191406,
    "Pressure": 944.274658203125,
    "Magnetic_Field_z": 13.4045991897583,
    "Magnetic_Field_x": -50.68513488769531,
    "Pitch": 357.405392561212,
    "Gyroscope_z": 0.0008658710867166519,
    "Gyroscope_y": -0.0009315479546785355,
    "Roll": 43.636521324080704,
    "Acceleration_z": 0.7225611209869385
  }
}

```

Figura 4.9: Activo creado e insertado en BigchainDB, cuyos metadatos constituyen los datos ambientales recogidos y transmitidos al cliente Linux

4) Inyección en MariaDB

- Se conecta a la base de datos creada en MariaDB, a partir del nombre (iota), el puerto (3364), el usuario y la contraseña determinados.
- Se almacena el mensaje en una variable, y dado que las claves-valor se encuentran desordenadas con respecto al orden de las columnas de la tabla creada en MariaDB, se itera el diccionario y se va introduciendo cada valor en su columna correspondiente. Además, se escribe la marca temporal de registro.
- Si todo sale bien, se muestra un mensaje de éxito por pantalla.
- Se puede visualizar la inyección de los datos a través del buscador en la dirección: <http://localhost:3364/phpmyadmin>

```
vroman@pcudp05:~/HAT/programas$ python3 mqtt.py
Connected with result code 0
test/data 0 b'{"Data_Collection_Time_Stamp": "2018-06-20 11:31:00", "Temperature": 38.21818161010742, "Acceleration_y": 0.6837038993835449, "Acc
on_x": 0.014304309152066708, "Gyroscope_x": 0.004105187952518463, "Yaw": 151.5492208514834, "Magnetic_Field_y": -27.209548950195312, "Humidity": 279846191406, "Pressure": 944.274658203125, "Magnetic_Field_z": 13.4045991897583, "Magnetic_Field_x": -50.68513488769531, "Pitch": 357.405392561
yroscope_z": 0.0008658710867166519, "Gyroscope_y": -0.0009315479546785355, "Roll": 43.636521324080704, "Acceleration_z": 0.7225611209869385}''
Successfully Added record to BichainDB!
Successfully Added record to MySQL
```

Figura 4.10: Datos

	Data_Storage_Time_Stamp	Data_Collection_Time_Stamp	Temperature	Pressure	Humidity	Pitch	Roll	Yaw	Magnetic_Field_X	Magnetic_Field_Y	Magnetic_Field_Z
2018-06-20 11:32:30	2018-06-20 11:32:22	38.2545	944.291	27.4022	357.879	44.2795	150.323	-50.4894	-26.9957		
2018-06-20 11:32:30	2018-06-20 11:32:28	38.1455	944.309	31.0411	357.353	44.0495	150.206	-50.3316	-27.5808		
2018-06-20 11:32:33	2018-06-20 11:32:33	38.2182	944.376	30.977	357.771	43.1817	150.052	-50.0781	-27.2729		
2018-06-20 11:32:38	2018-06-20 11:32:38	38.2364	944.285	29.7053	357.894	43.4764	149.901	-49.9835	-26.871		
2018-06-20 11:32:43	2018-06-20 11:32:43	38.2364	944.313	31.5376	358.143	43.9368	149.872	-50.2384	-27.4523		
2018-06-20 11:32:48	2018-06-20 11:32:48	38.2364	944.314	31.6881	358.443	43.6487	149.969	-50.3694	-26.9097		
2018-06-20 11:32:53	2018-06-20 11:32:53	38.2727	944.293	28.046	359.273	43.958	150.301	-50.675	-26.9751		
2018-06-20 11:32:59	2018-06-20 11:32:59	38.4	944.348	30.4869	359.786	43.9025	150.256	-50.593	-26.6368		
2018-06-20 11:33:04	2018-06-20 11:33:04	38.3091	944.344	29.7533	359.679	43.8337	150.28	-50.4442	-26.6006		
2018-06-20 11:33:09	2018-06-20 11:33:09	38.3091	944.351	27.3125	359.427	43.1139	150.165	-50.5163	-26.7395		
2018-06-20 11:33:14	2018-06-20 11:33:14	38.3091	944.317	29.4426	358.963	44.5062	149.877	-50.4316	-27.1386		
2018-06-20 11:33:19	2018-06-20 11:33:19	38.4545	944.347	29.1287	359.066	43.1016	149.823	-50.4123	-26.7115		
2018-06-20 11:33:24	2018-06-20 11:33:24	38.4	944.296	30.0448	359.204	42.7407	150.142	-50.6261	-27.3723		
2018-06-20 11:33:30	2018-06-20 11:33:30	38.3455	944.277	29.1159	358.95	44.6084	150.077	-50.0542	-26.8047		
2018-06-20 11:33:35	2018-06-20 11:33:35	38.4364	944.294	33.0143	359.582	42.3014	150.655	-49.9126	-27.2648		
2018-06-20 11:33:40	2018-06-20 11:33:40	38.3455	944.254	30.2947	359.396	42.7267	150.381	-50.5278	-27.1696		
2018-06-20 11:33:45	2018-06-20 11:33:45	38.3818	944.319	30.8713	358.909	43.5774	149.659	-50.7131	-26.8484		
2018-06-20 11:33:50	2018-06-20 11:33:50	38.4545	944.328	28.6899	359.889	41.6374	150.292	-50.6715	-26.8961		
2018-06-20 11:33:55	2018-06-20 11:33:55	38.4545	944.304	29.1159	359.561	41.5748	149.831	-50.6961	-27.059		
2018-06-20 11:34:01	2018-06-20 11:34:01	38.4182	944.263	27.7962	359.76	43.0247	150.365	-50.192	-27.3408		
2018-06-20 11:34:06	2018-06-20 11:34:06	38.3636	944.301	28.5842	359.603	42.0033	150.623	-50.401	-26.6123		
2018-06-20 11:34:11	2018-06-20 11:34:11	38.4364	944.293	30.1602	359.81	41.5662	151.071	-50.2245	-26.4221		
2018-06-20 11:34:16	2018-06-20 11:34:16	38.4	944.216	29.1191	0.204684	42.0357	151.21	-50.4628	-26.7339		
2018-06-20 11:34:21	2018-06-20 11:34:21	38.5091	944.261	29.8366	359.913	42.3397	151.411	-50.7838	-26.4904		
2018-06-20 11:34:26	2018-06-20 11:34:26	38.5455	944.212	29.952	0.291969	42.2899	151.695	-50.6688	-26.5125		

Figura 4.11: Datos registrados en MariaDB

Roll	Yaw	Magnetic_Field_X	Magnetic_Field_Y	Magnetic_Field_Z	Acceleration_X	Acceleration_Y	Acceleration_Z	Gyroscope_X	Gyroscope_Y	Gyroscope_Z
2795	150.323	-50.4894	-26.9957	13.1078	0.0130921	0.684675	0.722805	0.00167599	-0.000760255	0.000107202
0495	150.206	-50.3316	-27.5808	12.6177	0.0147892	0.685161	0.721586	-0.001256475	-0.00126273	0.00124405
1817	150.052	-50.0781	-27.2729	11.5205	0.0152741	0.621792	0.720124	0.00149227	-0.0000697533	0.000338743
4764	149.901	-49.9835	-26.871	11.2033	0.0145468	0.684189	0.722805	0.00272156	-0.00168999	-0.00138315
9368	149.872	-50.2384	-27.4523	11.8761	0.0138194	0.683947	0.72378	-0.00125515	-0.000469845	0.000169374
5487	149.969	-50.3694	-26.9097	12.7794	0.0147892	0.683461	0.721343	-0.000680815	-0.00132415	-0.00132889
.95.	150.301	-50.675	-26.9751	11.8714	0.0143043	0.684432	0.721343	-0.00206031	-0.00078023	0.000833391
9025	150.256	-50.593	-26.6368	13.9786	0.0150316	0.684675	0.721586	0.000916533	0.00168862	-0.000379976
3337	150.128	-50.4442	-26.6006	14.276	0.0155165	0.682976	0.723049	-0.00175219	0.000174287	0.000194257
1139	150.165	-50.5163	-26.7395	12.92	0.0147892	0.684918	0.725973	-0.00137448	-0.000640114	0.000748646
5062	149.877	-50.4316	-27.1386	13.5971	0.0140619	0.68686	0.724998	-0.00439128	0.00146845	0.000133796
1016	149.823	-50.4123	-26.7115	13.9646	0.0145468	0.684918	0.726948	-0.00529471	0.00226637	0.00160345
7407	150.142	-50.6261	-27.3723	13.1995	0.0162439	0.684432	0.725242	0.00225617	0.000178031	-0.000597098
5084	150.077	-50.0542	-26.8047	14.6125	0.0145468	0.68686	0.726948	-0.00536348	0.00256867	0.000548527
3014	150.655	-49.9126	-27.2648	13.7666	0.0140619	0.682733	0.725486	-0.000135433	-0.00225336	0.000482681
7267	150.381	-50.5278	-27.1696	13.8363	0.0150316	0.684675	0.7172	-0.0023835	0.000262763	0.00138188
5774	149.659	-50.7131	-26.8484	12.9813	0.013577	0.688074	0.726704	-0.00433455	0.00241967	0.00102034
5374	150.292	-50.6715	-26.8961	13.2697	0.013577	0.686132	0.727191	0.00176436	-0.000937544	0.00074457
5748	149.831	-50.6961	-27.059	13.5667	0.0155165	0.682976	0.71988	0.00429059	0.00236493	0.000731924
0247	150.365	-50.192	-27.3408	12.4096	0.0140619	0.683704	0.725729	-0.00267415	-0.00217035	-0.000536447
0333	150.623	-50.401	-26.6123	12.0977	0.0152741	0.684432	0.7172	0.00247857	0.000570475	-0.000842627
5662	151.071	-50.2245	-26.4221	12.6966	0.0147892	0.74513	0.720368	0.00006276	0.000615421	0.000711408
0357	151.21	-50.4628	-26.7339	12.5067	0.0145468	0.685403	0.723536	-0.00271475	-0.000290981	-0.000112297
3397	151.411	-50.7838	-26.4904	12.6867	0.0152741	0.62252	0.722561	0.0011409	0.000304937	-0.00136281
2899	151.695	-50.6688	-26.5125	12.7163	0.0152741	0.682247	0.722317	0.00310326	-0.0000105137	-0.000191379

Figura 4.12: Datos registrados en MariaDB

5) Compuتو del resumen estadístico temporal y publicación en IOTA

- A través de la Seed creada previamente y la dirección del nodo completo, se establece la conexión vía API a IOTA.
- Se generan las direcciones a las que se realizarán las transacciones.
- Se crea un dataframe de pandas (imagen) y se inyecta el contenido del mensaje recibido a través de MQTT.
- A la vez, se inicia un contador que determina cuando se realizará el cálculo del resumen estadístico y la publicación al Tangle.

- Se calcula el resumen estadístico.

```

Temperature Acceleration_y Acceleration_x Gyroscope_x   Yaw \
count      2.0000      2.0000      2.0000      2.0000      2.0000
mean       37.3909     0.1657     -0.0761     0.0006  144.7547
std        0.0643     0.0002     0.0005     0.0009    0.0526
min        37.3455     0.1656     -0.0765     0.0008  144.7175
50%       37.3909     0.1657     -0.0761     0.0006  144.7547
max        37.4364     0.1658     -0.0758     0.0012  144.7919

Magnetic_Field_y Humidity Pressure Magnetic_Field_z \
count      2.0000      2.0000      2.0000      2.0000
mean      -42.8126    26.9249    942.9504    12.3581
std       0.0970     0.4711     0.0435     0.2890
min      -42.8811    26.5917    942.9197    12.1537
50%       -42.8126    26.9249    942.9504    12.3581
max      -42.7440    27.2580    942.9812    12.5624

Magnetic_Field_x Pitch Gyroscope_z Gyroscope_y   Roll \
count      2.0000      2.0000      2.0000      2.0000      2.0000
mean      -60.1859     3.7309     0.0007     -0.0002    7.8209
std        0.0039     0.9633     0.0004     0.0013    0.2380
min      -60.1886     3.0497     0.0004     -0.0011    7.6527
50%       -60.1859     3.7309     0.0007     -0.0002    7.8209
max      -60.1831     4.4121     0.0010     0.0008    7.9892

Acceleration_z
count      2.0000
mean       0.9782
std        0.0007
min        0.9777
50%       0.9782

```

Figura 4.13: Resumen estadístico periódico de los datos recolectados

- Dado que los mensajes enviados a IOTA tienen una limitación de 81 trytes (2187 caracteres) y el formato del resumen estadístico que devuelve pandas no es compatible con IOTA, es necesario tratar la información. En primer lugar, se convierte a diccionario y se redondean los decimales. Además, se omiten los valores de los percentiles del 25% y del 75% por su menor relevancia.
- Como la información del resumen es posicional, se transforma al nuevo formato.

```

Message sent to tangle: {"Num": 6, "Temperature": [35.9, 0.0707, 35.8182, 35.8818, 36.0], "Acceleration_y": [0.4499, 0.0011, 0.4487, 0.4495, 0.4516], "Acceleration_x": [0.0196, 0.0019, 0.0179, 0.019, 0.0233], "Gyroscope_X": [-0.0016, 0.0021, -0.0042, -0.0017, 0.0016], "Yaw": [144.6136, 0.2877, 144.3218, 144.5763, 145.0923], "Magnetic_Field_y": [-38.42, 0.1462, -38.686, -38.4137, -38.2796], "Humidity": [27.1518, 0.2357, 26.7327, 27.1571, 27.4342], "Pressure": [938.0862, 0.0649, 938.0007, 938.0945, 938.145], "Magnetic_Field_z": [15.8246, 0.8381, 14.7016, 15.8784, 16.8803], "Magnetic_Field_x": [-56.2405, 0.2446, -56.5078, -56.2651, -55.9386], "Pitch": [359.3264, 0.43, 358.8901, 359.2653, 359.9761], "Gyroscope_z": [0.0006, 0.0008, -0.0007, 0.0007, 0.0017], "Gyroscope_y": [-0.0009, 0.0019, -0.0042, -0.0007, 0.0013], "Roll": [26.1115, 0.5811, 24.9579, 26.307, 26.5397], "Acceleration_z": [0.8899, 0.026, 0.8856, 0.8905, 0.8924]}

```

Figura 4.14: Mensaje publicado al Tangle de IOTA

- Se codifica el resumen en trytes.
- Se envía la transacción y se muestra el mensaje enviado y el tiempo que tarda en inyectarse en el Tangle.

```
Time: 11:31:25 data transferred! in: 13.5 seconds
```

Figura 4.15: Tiempo que se ha tardado en realizar el PoW necesario para publicar el mensaje en el Tangle de IOTA

- El mensaje publicado puede rastrearse a través de diversos medios, el escogido por mayor comodidad es la página web www.thetangle.org. Se pueden rastrear las transacciones escribiendo en el buscador que tiene integrado las direcciones a las que se envían o se han recibido, por el ID de las mismas, o por el TAG. Una vez encontrada la transacción se visualiza de la siguiente manera:

① <https://thetangle.org/transaction/WQFFRFZNZYGWNYWCZABHKIO99MOCKGXNFVXPUGTSVCGAAAYTEREGWEXZ9HAFBXURSPIRAFJPAOYZ9999>

Transaction

WQFFRFZNZYGWNYWCZABHKIO99MOCKGXNFVXPUGTSVCGAAAYTEREGWEXZ9HAFBXURSPIRAFJPAOYZ9999 </> ⓘ

June 20, 2018 11:31:38 - 5 minutes and 49 seconds ago

Value	0 I	Pending
Conversion	0 USD	Index in bundle 0 / 0
Tag	RBRRYPISNRSMRYFRSTVSN99999	Weight magnitude 14
Address	EABEHEOVIFMSKSZUWKZNWDHWSUDVAMGSFNEYFODWPIDHTXCGNHKAAWKUVIRANAHTQFCHQOUUX9NKVMLAXUDQJSBCA	
Bundle	ANJYHVNPYZEEBBULQYDYJLFFXZNGAVXJVXGNYRKFLCUJHLTNNBBLEYMVTBYGBRZJ9RBDZCZIGJVZEL9	
Nonce	P9MKOGHLEPEEF9TBIKXHQTGUBJG	
Message	{"Num": 1, "Temperature": [38.2545, NaN, 38.2545, 38.2545, 38.2545], "Acceleration_y": [0.6847, NaN, 0.6847, 0.6847, 0.6847], "Acceleration_x": [0.0141, NaN, 0.0141, 0.0141, 0.0141], "Gyroscope_x": [0.0018, NaN, 0.0018, 0.0018, 0.0018], "Yaw": [151.534, NaN, 151.534, 151.534, 151.534], "Magnetic_Field_y": [-26.88, NaN, -26.88, -26.88, -26.88], "Humidity": [30.032, NaN, 30.032, 30.032, 30.032], "Pressure": [944.2878, NaN, 944.2878, 944.2878, 944.2878], "Magnetic_Field_z": [13.0227, NaN, 13.0227, 13.0227, 13.0227], "Magnetic_Field_x": [-50.4104, NaN, -50.4104, -50.4104, -50.4104], "Pitch": [357.4617, NaN, 357.4617, 357.4617, 357.4617], "Gyroscope_z": [0.0009, NaN, 0.0009, 0.0009, 0.0009], "Gyroscope_y": [-0.0008, NaN, -0.0008, -0.0008, -0.0008], "Roll": [44.6103, NaN, 44.6103, 44.6103, 44.6103], "Acceleration_z": [0.7213, NaN, 0.7213, 0.7213, 0.7213]}}	

Parent transactions

Trunk	Branch
KWOHLRNXBHDAMHDJKQ9MB9JAECM9NEWKZWIAOXRIKGWPYK	GYGQSQTEHFAKXQHLLTFGUETIQGXKHHWDOF9YNMUTX9YXNSFWL9H
WRJIMIQWRIRINZ9ACRVPEMV99999	MBGHVFSRTUWSKOOXUPUAVGZ9999

Figura 4.16: mensaje publicado al Tangle de IOTA, visualizado en texto

Capítulo 5 Conclusiones, Líneas Futuras y Difusión

En este capítulo se hará una resumen, descripción y análisis de los resultados de las pruebas realizadas en el proyecto, y se destilarán unas conclusiones de ellas. Además, se hablará de posibles líneas futuras de desarrollo del proyecto con implementación de nuevas funcionalidades, y se acabará detallado las principales plataformas de difusión del proyecto.

5.1 Conclusiones generales

La primera conclusión obtenida a partir de los resultados observados en la transmisión de datos vía el protocolo de comunicación MQTT, es que se trata de un sistema sumamente fiable, útil y gratuito para transmitir información de manera prácticamente instantánea entre dispositivos que se encuentren conectados vía internet. Por lo que su elección resultó un acierto.

Por otro lado, se observan ciertas discrepancias entre los datos obtenidos y los reales, al utilizar el sensor para medir la temperatura del ambiente, pues al encontrarse tan cerca del dispositivo Raspberry Pi, hay un cierto error en la medición provocado por el calor que genera la propia Raspberry. Por ello, los datos de temperatura obtenidos no son fiables y habría que tenerlo en cuenta a la hora de implementar el sistema en un escenario en el que se busque monitorizar la temperatura de forma real. Como la temperatura de la Raspberry se mantiene prácticamente constante a lo largo del tiempo en torno a unos 38 °C, se podría tomar esa medida como base y aplicar un error para obtener las medidas reales.

La segunda conclusión, con respecto a la inserción de los datos en el servidor MariaDB es similar a la primera, ambas son tecnologías ampliamente utilizadas, probadas y que funcionan exactamente como se supone que deben hacerlo.

La tercera conclusión obtenida, es que, gracias a la sinergia de una serie de tecnologías que se pueden encontrar y utilizar de forma prácticamente gratuita, a pesar de encontrarse en la vanguardia del desarrollo, se puede construir con ellas un sistema fiable, seguro, eficaz y eficiente para la monitorización y seguimiento de una cualquier objeto o área de interés para varias partes interesadas que no tienen por qué confiar entre ellas. Este sistema permite el flujo de información íntegra e inmutable, de manera que cada parte pueda tomar las mejores decisiones pertinentes a cada situación y, de esa forma, optimizar y maximizar la eficacia de sus operaciones y, por tanto, su beneficio. Resumiendo, con una inversión muy baja se pueden obtener grandes retornos.

La cuarta conclusión que se puede derivar es que la principal barrera para la implementación de este tipo de soluciones en la actualidad, es la falta de información y de tutoriales al respecto, lo que limita el campo de conocimiento a unos pocos desarrolladores que, a pesar de realizar una actividad muy intensa en foros y páginas especializados, en ocasiones se queda corta.

La tercera conclusión obtenida con respecto a los resultados obtenidos en la publicación de mensajes en IOTA es que existe una dispersión muy grande en los resultados obtenidos en cuanto al tiempo de inserción en la Tangle, esta dispersión se debe a que dicho tiempo depende del nivel de actividad de la red en ese momento, pues cuantas más transacciones halla, mayor será la probabilidad de encontrar transacciones disponibles para ser validadas. Además, este tiempo también depende de la capacidad de computación del nodo asociado, pues esta capacidad determinará el tiempo que le tomará en realizar el PoW necesario para validar las transacciones. En las pruebas se han obtenido tiempos comprendidos entre: Máximos de 35 segundos y mínimos de 1,8 segundos. Se ha observado que los tiempos siguen una distribución

```
Time: 10:27:00  data transferred! ln: 1.8 seconds
test/data 0 b'{ "Data": "Collection Time Stamp": "2017-06-05 10:27:03", "Temperature": 35.9636344909668, "Acceleration_y": 0.45159417390823364, "Acceleration_x": 0.019395673647522926, "Gyroscope_x": -0.0069330669939517975, "Yaw": 144.1095739018304, "Magnetic_Field_y": -38.236305236816406, "Humidity": 27.158706665039062, "Pressure": 938.05224609375, "Magnetic_Field_z": 16.38738441467285, "Magnetic_Field_x": -56.48558044433594, "Pitch": 357.7164573052609, "Gyroscope_z": -0.0063939439775422155, "Gyroscope_y": 0.002840142697095871, "Roll": 25.36490981310305, "Acceleration_z": 0.8843758702278137}', "Sensor Data sent to BigchainDB:
Successfully Added record to mysql
Message sent to tangle: {"Num": 1, "Temperature": [35.9818, NaN, 35.9818, 35.9818, 35.9818], "Acceleration_y": [0.4526, NaN, 0.4526, 0.4526, 0.4526], "Acceleration_x": [0.0158, NaN, 0.0158, 0.0158, 0.0158], "Gyroscope_x": [0.0108, NaN, 0.0108, 0.0108, 0.0108], "Yaw": [144.2746, NaN, 144.2746, 144.2746, 144.2746], "Magnetic_Field_y": [-38.3065, NaN, -38.3065, -38.3065, -38.3065], "Humidity": [27.0082, NaN, 27.0082, 27.0082, 27.0082], "Pressure": [938.1191, NaN, 938.1191, 938.1191, 938.1191], "Magnetic_Field_z": [16.4387, NaN, 16.4387, 16.4387, 16.4387], "Magnetic_Field_x": [-56.3281, NaN, -56.3281, -56.3281, -56.3281], "Pitch": [358.0374, NaN, 358.0374, 358.0374, 358.0374], "Gyroscope_z": [0.0005, NaN, 0.0005, 0.0005, 0.0005], "Gyroscope_y": [0.0068, NaN, 0.0068, 0.0068, 0.0068], "Roll": [24.2707, NaN, 24.2707, 24.2707, 24.2707], "Acceleration_z": [0.8832, NaN, 0.8832, 0.8832, 0.8832]}
sleep...
^ctime: 10:27:16  data transferred! ln: 4.3 seconds
```

Figura 5.1: Mínimo tiempo obtenido en la publicación de un mensaje en el Tangle

normal, quedando la mayoría de ellos entre los 5 y los 20 segundos.

Otro factor a tener en cuenta, son los Snapshots que se llevan a cabo una vez al mes aproximadamente en el Tangle de IOTA que elimina todas las transacciones que tienen un balance de 0 miotas con el objetivo de “limpiar el Tangle” y mejorar su eficiencia, y que por tanto eliminarán todas las publicaciones realizadas a través del sistema desarrollado en dicho Tangle.

Con respecto a los resultados obtenidos en la utilización de BigchainDB, la primera conclusión que se obtiene es que resulta muy laborioso trabajar con la estructura de las transacciones y que, es necesario una mejora en la simplicidad de utilización del sistema para que la tecnología pueda ser adoptada por un público mas extenso. Por otro lado, el sistema funciona de manera superlativa, siendo la inserción de los datos un proceso prácticamente instantáneo y quedando registrados de forma extremadamente fiable, pues resultan incorruptibles.

Como conclusión final, y a nivel personal, puedo afirmar que la realización del proyecto me ha supuesto el aprendizaje de una gran cantidad de tecnologías que desconocía y me ha ayudado a ver la gran cantidad de oportunidades de aplicación de las mismas en los problemas los que se enfrenta la nueva industria.

5.2 Líneas Futuras

A pesar del buen funcionamiento del proyecto, este no es perfecto, y algunas de las posibles implementaciones futuras que podrían mejorar la versatilidad y utilidad del mismo son:

- **Integrar un sistema de seguimiento GPS:** Esta posible implementación sería de especial interés para el caso comentado en la introducción del proyecto de la empresa distribuidora, pues conectar un chip de posicionamiento por GPS a la Raspberry Pi resulta relativamente sencillo y permitiría un mayor control en la monitorización y seguimiento de la mercancía a lo largo de su ruta.
- **Implementar la funcionalidad de IOTA de MAM:** MAM corresponde a las siglas de Masked Authenticated Messaging y una traducción al español podría ser Mensajería Autenticada Enmascarada. Es una comunicación a través de canales en los que el publicador configura un nivel de privacidad de entre tres opciones: público (cualquiera puede ver los mensajes), privado (solo el publicador puede ver los mensajes) y restringido (se limitan los que pueden ver los mensajes mediante la compartición de una clave específica). Este sistema es más conveniente la implementación en corporaciones que el utilizado para el proyecto pues, permite la privacidad de las publicaciones y las mismas no están sujetas al borrado ocurrido durante las Snapshots.
- **Utilizar Single Board Computer (SBC) con mayores especificaciones técnicas:** La principal ventaja de la Raspberry Pi, es su contenido precio, pero ello también conlleva ciertas limitaciones en cuanto a características técnicas, que impiden la realización de tareas exigentes. Si se utilizase una placa más potente se podrían llevar a cabo varias acciones de manera autónoma desde la propia placa base, evitando la necesidad de enviar los datos a la máquina Linux.
 - 1) Una de ellas sería la utilización de dicha placa como nodo completo de IOTA, de manera que no habría que depender de otro externo, y se podría realizar de manera autónoma el PoW necesario para insertar los datos en el Tangle. Es decir, sustituir la Raspberry Pi que se utiliza actualmente por otra placa de mayores capacidades técnicas que pueda realizar el PoW necesario para la inserción de datos en el Tangle. La RAM mínima necesaria son 4 Gb en el momento de escritura del documento.
 - 2) Otra posible aplicación, sería la inserción autónoma de los datos recogidos en las bases de datos de MariaDB y BigchainDB. Ya que, en el momento de la escritura del documento, existen ciertas incompatibilidades en los drivers que impiden su utilización en la Raspberry Pi.
- **Utilización de un sensor de temperatura externo:** Capaz de tomar medidas más cercanas a la realidad y que no se vean influidas por la temperatura de la propia placa.

- **Complementar el sistema con otra Blockchain que ejecute Smart Contracts:** Como por ejemplo Ethereum, ello permitiría que el sistema ejecutase automáticamente contratos inteligentes en función de si se cumplen o no las condiciones estipuladas por las distintas partes interesadas e involucradas en la monitorización del sistema. En el caso de la empresa distribuidora, se podrían ejecutar pagos automáticamente cuando la mercancía llegase al lugar acordado en las condiciones adecuadas.

5.3 Difusión:

Debido al carácter práctico y funcional de la solución adoptada, de la cantidad y usos y aplicaciones posibles en la industria y la novedad de las tecnologías utilizadas, se considera que resulta un proyecto interesante y merecedor de ser difundido. Por ello, se realizará la difusión de dicha solución por varias vías y canales que permitirán su utilización y mejora por parte de cualquier persona interesada.

En concreto, las principales vías de difusión serán:

- 1) **GitHub:** GitHub es el repositorio de proyectos de programación por excelencia y todo el código desarrollado para la realización del proyecto se encuentra subido en el mismo. Concretamente en la siguiente dirección:
<https://github.com/rromanss23/Store-Sensor-Data-in-BigchainDB-MariaDB-and-Publish-Statistic-Summary-to-IOTA-s-Tangle>
- 2) **Discord:** Discord es un chat originalmente creado para conversaciones en tiempo real entre gamers pero que fue evolucionando y se fue adoptando por numerosas comunidades de desarrolladores de proyectos de programación. Actualmente es uno de los foros de este tipo más activos y el presente proyecto se ha compartido a través del canal oficial de desarrolladores de IOTA.
- 3) **Gitter:** Gitter es un foro perteneciente a GitHub por el que los desarrolladores intercambian ideas y se poyan en la resolución de dudas y problemas. El presente proyecto también se ha compartido por esta vía.
- 4) **Conferencia SC2 IoV SOCA:** Conferencia de tecnologías punteras en la industria en la que se proporcionan tutoriales y presentaciones sobre soluciones novedosas a los problemas de la industria 4.0. Los temas principales son: Servicios Cloud, IoT aplicado a vehículos y computación orientada a aplicaciones. Dicha conferencia se celebrará entre los días 19-22/11/2018 en MSH Paris Nord y en ella se presentará un paper en el que se explicará la solución adoptada en el presente documento. Dicho paper se ha realizado de manera colaborativa entre el autor del presente documento y su tutor Joaquín Ordieres. <http://lipn.univ-paris13.fr/~cerin/sc2iovsoc2018.html>

Capítulo 6 Gestión del Proyecto

En este capítulo se hará una descripción del proceso seguido a lo largo del proyecto y como se han gestionado los recursos para la consecución del mismo.

6.1 Introducción

Para llevar a cabo una realización lo más eficaz y eficiente posible, el proyecto se dividió en diferentes segmentos de trabajo, con el objetivo de ir consiguiendo determinados hitos en unos plazos preestablecidos.

Debido a que el proyecto exigía el trabajo con tecnologías desconocidas por el autor, hubo primero un tiempo de toma de contacto y aprendizaje con la maquina Linux y la Raspberry Pi, que fue clave para ganar familiaridad con el entorno Linux y para poder trabajar eficientemente con ellos a través del terminal de comandos. Esta, fue una tarea imprescindible pues, acciones de instalación de programas y librerías es necesario hacerlo a través de dicho terminal, así como para solucionar los distintos problemas de incompatibilidades y errores que han ido surgiendo a lo largo de la elaboración del proyecto.

Por otro lado, también fue necesario realizar un aprendizaje previo del lenguaje de programación Python, pues es el utilizado a lo largo del proyecto para desarrollar los programas que le dan forma. Este aprendizaje se realizó a través de varias vías, siendo las principales:

- Cursos MOOC Introduction to Interactive Programming in Python I y II, y Fundamentals of Computing I y II, ofertados en la página web Coursera.
- Libro Python Crash Course
- Numerosos foros y blogs como: StackOverflow, GitHub, Discord...

En particular, los últimos foros y blogs mencionados también resultaron de especial importancia y ayuda para poder desarrollar apropiadamente los programas necesarios para alcanzar los objetivos del proyecto, obteniendo un gran feedback e intercambio de ideas con las comunidades de programadores.

6.2 Planificación Temporal

A continuación, se presenta el diagrama EDP, en el que se muestran las distintas fases y subfases en las que se ha dividido la realización del proyecto.

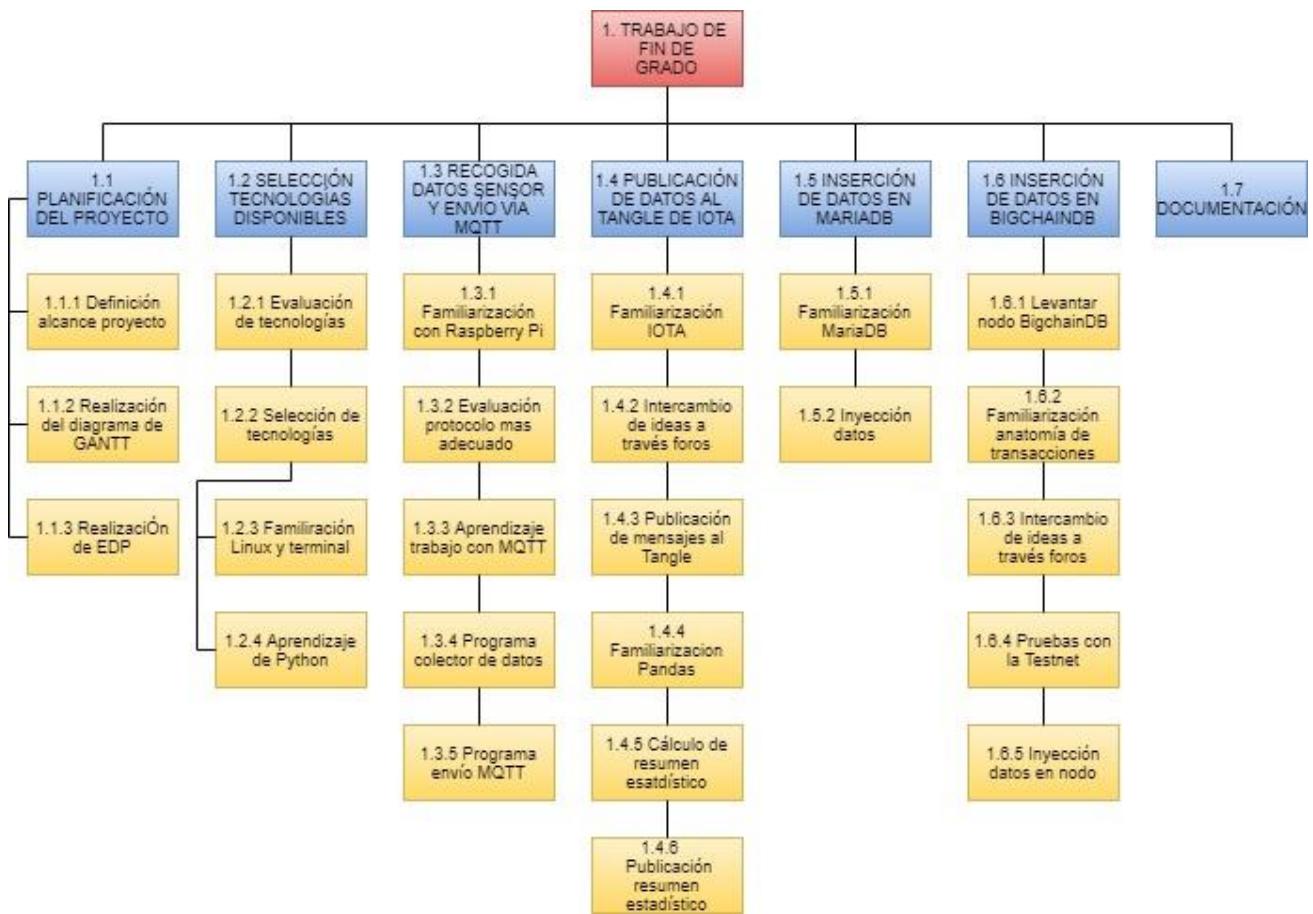


Figura 6.1: Diagrama EDP de las fases del proyecto

Las distintas fases han sido las siguientes:

- **Planificación del Proyecto:** Fase en la que se definió el alcance del proyecto, las diferentes tareas necesarias para la consecución de sus objetivos y se establecieron los horizontes temporales de dichas tareas.
- **Selección de Tecnologías:** Fase en la que se realizó una evaluación y selección de las tecnologías utilizadas en el proyecto. Además, se realizó una primera introducción y aprendizaje de las diferentes tecnologías y lenguajes de programación utilizados a lo largo del proyecto.
- **Recogida de datos y envío a través de MQTT:** En esta fase, se desarrolló el programa de recogida de datos a través del sensor SenseHat conectado a la Raspberry Pi, se realizó la evaluación y selección del protocolo de comunicación entre máquinas más adecuado, se realizó el aprendizaje de MQTT y se desarrollaron los programas de envío y recepción de los datos a través de dicho canal, así como las pruebas necesarias para la demostración del correcto funcionamiento.

- **Publicación de datos al Tangle:** En esta fase se configuraron los sistemas para poder trabajar con IOTA, se realizó un aprendizaje de utilización de la DLT, se realizó el aprendizaje de utilización del módulo de tratamiento de datos de Python Pandas, se desarrolló el programa de cálculo de resumen estadístico y por último se desarrolló el programa de publicación de dicho resumen y las pruebas necesarias para la demostración del correcto funcionamiento.
- **Inserción de datos en MariaDB:** En esta fase se instaló y configuró la base de datos MariaDB, así como la tabla que se utilizó para registrar los datos ambientales. Además, se desarrolló el programa para la inserción de los datos y las pruebas del correcto funcionamiento.
- **Inserción de datos en BigchainDB:** En esta fase se levantó y configuró el nodo de BigchainDB, se realizó un aprendizaje y familiarización con la anatomía de las transacciones en la DLT, se realizaron pruebas de funcionamiento en su testnet y por último se implementaron en la red real.
- **Documentación:** Paralelamente a la realización del proyecto se fue realizando la elaboración del presente documento.

A continuación se muestra el diagrama de Gantt en el que se detalla de manera gráfica los plazos temporales que llevaron cada una de las fases, desde el inicio del proyecto a principios de Febrero del 2018, hasta la entrega del mismo, a finales de Julio de 2018. Siendo la duración total de la elaboración del mismo de unos 6 meses, que coincide con el plazo estimado cuando se definieron el alcance y los objetivos del proyecto.

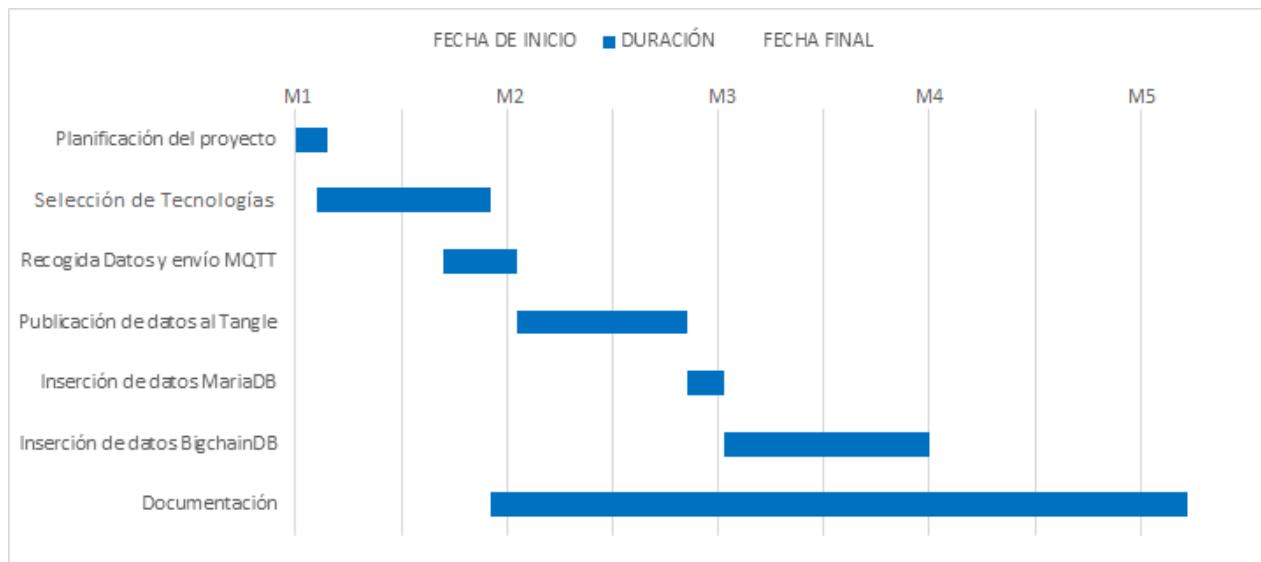


Figura 6.2: Diagrama de Gantt del proyecto

Tabla 6.1: Tabla de desglose de las fechas de inicio y final de las fases del proyecto

TAREA	FECHA DE INICIO	DURACIÓN	FECHA FINAL
Planificación del proyecto	01/02/2018	6	07/02/2018
Selección de tecnologías	05/02/2018	33	10/03/2018
Recogida Datos y envío MQTT	01/03/2018	14	15/03/2018
Publicación de datos al Tangle	15/03/2018	32	16/04/2018
Inserción de datos MariaDB	16/04/2018	7	23/04/2018
Inserción de datos BigchainDB	23/04/2018	39	01/06/2018
Documentación	10/03/2018	132	20/07/2018

6.3 Presupuesto

En este punto se desarrolla como se acometería la realización de un encargo profesional para la realización del proyecto, cuanto tardaría en implementarse para que resultase completamente funcional y el coste económico que supondría. La implementación supondría realizar las fases desglosadas en el EDP. El proyecto se ha realizado con un ordenador de sobremesa Linux con dos núcleos simétricos GPU y 8 Gb de RAM con un precio original de 2450€. Además, se ha utilizado una Raspberry Pi y un sensor, para medir parámetros ambientales, de la marca SenseHat y una serie de softwares de código libre que no tienen coste de utilización. Se considera una amortización a cuatro años del coste del ordenador y la placa sensor.

Tabla 6.2: Desglose de Coste Material del proyecto

Elemento	Vida Útil (años)	Precio (€)	Tiempo de uso (años)	Amortización (€)
Ordenador	4	2450	0.5	306,25
Microsoft Office	-	-	0.5	-
Raspbian	-	-	0.5	-
Ubuntu	-	-	0.5	-
MQTT	-	-	0.5	-
IOTA	-	-	0.5	-
MariaDB	-	-	0.5	-
BigchainDB	-	-	0.5	-
Raspberry Pi	4	36,19	0.5	4,52
SenseHat	4	38,18	0.5	4,77
	Presupuesto Ejecución Material			315,54

Tabla 6.3: Honorarios por la realización del proyecto

Elemento	Salario (€/h)	Trabajo (h)	Total (€)
Honorarios Ingeniero Técnico	30	300	9000
Honorarios Documentalista	30	60	1800
Coste Mano de Obra Total			10800

Se considera una mano de obra por valor de 30€ la hora para el trabajo realizado como Ingeniero Técnico, realizando la investigación instalación y configuración de las tecnologías y sistemas, llevando este trabajo 300 horas. Por otro lado, el trabajo de documentalista se estima en 20€ la hora y el mismo ha llevado 60 horas de trabajo. Ascendiendo el Coste de Mano de Obra a la cantidad de diez mil ochocientos euros, **10800€**.

El Presupuesto por Ejecución Material (PEM), corresponde a:

$$\text{PEM} = \text{Coste Material} + \text{Coste Mano de Obra} = \mathbf{11115,54\text{€}}$$

Asciende el Presupuesto de Ejecución Material, a la citada cantidad de once mil ciento quince euros con cincuenta y cuatro céntimos.

Para calcular el Presupuesto de Ejecución por Contrata, al PEM se le deben sumar los Costes Generales (CG) del proyecto, que se estiman en un 8%, y el Beneficio Industrial (BI), que se estima en un 5%. A dicha suma se le aplicará un 21% de IVA.

Tabla 6.4: Presupuesto de Ejecución por Contrata

Elemento	Total (€)
PEM	11115,54
BI	555,78
CG	889,24
IVA (21%)	2637,72
PEC	15198,28

Ascendiendo la cifra del **Presupuesto Total** (que coincide con el PEC) a la cantidad de quince mil ciento noventa y ocho con veintiocho céntimos, **15198,28€**.

Capítulo 7 Bibliografía

- [1] K. Rose, S. Eldridge, and L. Chapin, “LA INTERNET DE LAS COSAS— UNA BREVE RESEÑA.”
- [2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.
- [3] M. N. O. Sadiku, Y. Wang, S. Cui, S. M. Musa, and R. G. Perry, “INDUSTRIAL INTERNET OF THINGS,” *Int. J. Adv. Sci. Res. Eng.*, vol. 3, no. 11, 2017.
- [4] O. Vermesan and P. Friess, “Internet of Things – From Research and Innovation to Market Deployment River Publishers Series in Communication,” 2014.
- [5] L. Columbus, “2017 Roundup Of Internet Of Things Forecasts.” [Online]. Available: <https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/#613ef2f81480>. [Accessed: 18-Jun-2018].
- [6] D. M. Ángel and M. Sánchez, “Desarrollo de proyectos IoT utilizando Raspberry Pi como plataforma,” p. 140.
- [7] Nicky Woolf, “DDoS attack that disrupted internet was largest of its kind in history, experts say | Technology | The Guardian.” [Online]. Available: <https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet>. [Accessed: 18-Jun-2018].
- [8] F. Labs, “Swarming IoT Attacks, Cryptojacking, and Ransomware Drive Dramatic Spike in Malware.” [Online]. Available: <https://www.fortinet.com/blog/threat-research/swarming-iot-attacks--cryptojacking--and-ransomware-drive-dramat.html>. [Accessed: 22-May-2018].
- [9] L. J. Aguilar, “Capítulo primero Ciberseguridad : la colaboración público-privada en la era de la cuarta revolución industrial (Industria 4 . o Luis Joyanes Aguilar Abstract,” pp. 19–64, 2017.
- [10] “Bitcoin mining is swallowing up power resources in Iceland.” [Online]. Available: <https://www.siliconrepublic.com/enterprise/iceland-bitcoin-cryptocurrency-energy>. [Accessed: 27-Apr-2018].
- [11] “MQTT.” [Online]. Available: <http://mqtt.org/>. [Accessed: 08-Jul-2018].
- [12] A. Preukschat, *Blockchain: La revolución industrial de Internet*. 2016.
- [13] Jaime Nuñez Miller, “Consenso - Libro Blockchain.” [Online]. Available: <http://libroblockchain.com/consenso/>. [Accessed: 18-Jun-2018].

- [14] Tom Groenfeldt, “IBM And Maersk Apply Blockchain To Container Shipping.” [Online]. Available: <https://www.forbes.com/sites/tomgroenfeldt/2017/03/05/ibm-and-maersk-apply-blockchain-to-container-shipping/#905b36f3f05e>. [Accessed: 19-Jun-2018].
- [15] *1129409 - Raspberry Pi Foundation*. Charity Commission for England and Wales, 2011.
- [16] *FAQs*. Raspberry Pi Foundation.
- [17] R. Cellan-Jones and R. Cellan-Jones, *A £15 computer to inspire young programmers*. BBC News, 2011.
- [18] P. Price and P. Price, *Can a £15 computer solve the programming gap?* BBC Click, 2011.
- [19] *Made in the UK!* Raspberrypi.org, 2012.
- [20] C. Domínguez, “Aplicaciones orientadas a la domótica con Raspberry Pi,” p. 82, 2015.
- [21] *YakketyYak/ReleaseSchedule - Ubuntu Wiki*. .
- [22] *Ubuntu MATE ya es un “sabor” oficial de Ubuntu*. .
- [23] *UbuntuFlavours - Ubuntu Wiki*. .
- [24] *Preguntas frecuentes Página oficial de Ubuntu MATE*. .
- [25] Raspberry Foundation, “Sense HAT - Raspberry Pi.” [Online]. Available: <https://www.raspberrypi.org/products/sense-hat/>. [Accessed: 19-Jun-2018].
- [26] J. C. Mu and D. Guti, “Autor: Juan Cruz Muñoz Tutor: Daniel Gutiérrez Reina,” 2017.
- [27] B. GmbH, “BigchainDB: The blockchain database,” no. May, pp. 1–14, 2018.
- [28] T. McConaghy, “BigchainDB Transaction Specs.” .
- [29] IOTA Foundation, “What is IOTA?” [Online]. Available: <https://iota.readme.io/docs/what-is-iota>. [Accessed: 19-Jun-2018].
- [30] Konfid.io, “IOTA Report: Decoding the Tangle — Part 1 – Konfid.io Blockchain Reports – Medium.” [Online]. Available: <https://medium.com/konfid-io-blockchain-reports/iota-report-decoding-the-tangle-part-1-a7705c458583>. [Accessed: 19-Jun-2018].
- [31] IOTA Foundation, “IOTA Glossary.” [Online]. Available: <https://iota.readme.io/docs/glossary>. [Accessed: 19-Jun-2018].
- [32] IOTA Foundation, “The Anatomy of a Transaction.” [Online]. Available:

<https://iota.readme.io/docs/the-anatomy-of-a-transaction>. [Accessed: 19-Jun-2018].

- [33] Louie Lu, “In-depth explanation of how IOTA making a transaction (with picture).” [Online]. Available: <https://medium.com/@louielu/in-depth-explanation-of-how-iota-making-a-transaction-with-picture-8a638805f905>. [Accessed: 19-Jun-2018].
- [34] Alon Gal, “The Tangle: an illustrated introduction – IOTA.” [Online]. Available: <https://blog.iota.org/the-tangle-an-illustrated-introduction-79f537boa455>. [Accessed: 19-Jun-2018].