

FMH606 Master's Thesis 2022

Master of Science, Industrial IT and Automation

**Integrating Decentralized Ledger Technologies
with the Internet of Things (IoT) to handle
Operational Maintenance Data**

Rejith Reghunathan

Faculty of Technology, Natural sciences and Maritime Sciences

Campus Porsgrunn

Course: FMH606 Master's Thesis, 2022

Title: Integrating Decentralized Ledger Technologies with the Internet of Things (IoT) to store operational maintenance data.

Number of pages: 63

Keywords: IoT, Preventive Maintenance, Blockchain, Ethereum, IOTA, Smart contracts, Data Storage.

Student: Rejith Reghunathan

Supervisor: Leila Ben Saad

External partner: N/A

Summary:

The number of IoT devices is over several billion as of 2020. The increased number of IoT devices and the generated data introduces the challenge of privacy. There is a need for a safe and trustworthy-based storage model. IoT devices can be used to monitor, predict, and schedule machinery maintenance. The service history and generated IoT data for analysis are critical, and data integrity is vital. The thesis presents a solution for storing such critical IoT operational sensor data in a decentralized ledger. Instead of storing data in a traditional database or in a cloud, the data is stored in both Ethereum blockchain and IOTA tangle. The thesis develops an application that can acquire data from IoT devices, store it in the decentralized ledgers such as Ethereum and IOTA, and stakeholders can view historical operational data in immutable and secure technology. The performance of the proposed solution is evaluated and a comparison between the distributed ledger technologies are conducted.

Preface

The thesis is carried out to partially fulfill the master's degree in Industrial IT and Automation requirement at the University of Southeast Norway (USN).

I wish to thank my supervisor, Ms. Leila Ben Saad, for her guidance and close supervision throughout the project. Her mentorship and expertise aided me in challenging myself to step out of my comfort zone and explore several topics for the thesis.

I thank the discord and community support for IOTA for their guidance during the project.

I wish to thank my good colleague at work, Mr. Jarle Melandsør, for sharing the hardware needed for the thesis implementation. I also thank Mr. Siv S, a software developer in Smart World, for providing a general overview of distributed ledger technologies.

My gratitude goes to my wife Krishna, our family in India, and our two beautiful kids, Anamika, and Archie, for their encouragement throughout the course.

Horten, Norway.

3 September 2022

Contents

1	Introduction	6
2	Background	9
	2.1 IoT Concept.....	9
	2.2 IoT benefits	10
	2.3 IoT requirements	10
	2.4 IoT in operational maintenance systems	10
	2.5 Distributed Ledger Technologies	11
	2.6 Blockchain	11
	2.6.1 <i>How BC works</i>	12
	2.6.2 <i>Types of BCs</i>	12
	2.6.3 <i>Consensus Algorithms</i>	13
	2.7 DAGs.....	15
	2.7.1 <i>IOTA</i>	15
	2.8 Smart Contracts.....	17
	2.9 IoT and DLT integration	18
	2.10 Distributed ledger security	19
	2.11 Literature review.....	19
3	System Concept and Specification	20
	3.1 System overview	20
	3.2 System specification.....	21
	3.3 System requirements and Use Case	21
	3.3.1 <i>Requirements</i>	21
	3.3.2 <i>Use case diagram</i>	22
4	System Design and Implementation.....	23
	4.1 Sequence diagram for use cases.....	23
	4.1.1 <i>Read / Store IoT Data</i>	23
	4.1.2 <i>Access IoT Data</i>	24
	4.1.3 <i>Store Service History</i>	25
	4.1.4 <i>Access Service Data</i>	26
	4.2 System architecture	27
	4.3 System implementation	29
	4.3.1 <i>System Implementation diagram</i>	29
	4.3.2 <i>Algorithm</i>	30
5	Results and Discussion.....	33
	5.1 Testing.....	33
	5.2 Performance Evaluation	39
	5.2.1 <i>CPU and Memory Usage</i>	39
	5.2.2 <i>Gas Cost</i>	40
	5.2.3 <i>IOTA Vs. Blockchain</i>	41
	5.3 Discussion	41
6	Conclusion.....	43
	References	44
	Appendices	47

Nomenclature

IoT	-	Internet of Things
BC	-	Blockchain
DAG	-	Directed Acyclic Graph
HTML	-	Hyper Text Markup Language
DOS	-	Denial of service
AI	-	Artificial Intelligence
DLT	-	Distributed Ledger Technology
IPFS	-	Inter Planetary File System
DApps	-	Distributed Applications
EVM	-	Ethereum Virtual Machine
PoW	-	Proof of Work
PoS	-	Proof of Stake
WSN	-	Wireless Sensor Technology
PdM	-	Predictive maintenance
ISC	-	IOTA Smart Contracts
EOA	-	Externally Owned Account
API	-	Application Programming Interface
HTML	-	Hyper Text Markup Language
CSS	-	Cascading Style Sheets
MQTT	-	Message Queue Telemetry Transport

1 Introduction

Internet of Things, or IoT in its short form, is an emerging technology field where a global network of devices communicates and exchanges data over the Internet [1]. Internet of things can acquire data, analyze it, and make decisions based on the information collected [1]. Such devices are gradually gaining recognition in various industries such as the health sector, supply chain, energy distribution, shipping, and transport sectors. As of 2022, the IoT market is expected to grow to 14.4 billion active connections [2]. It is expected that by 2025, despite the global chip shortage in 2022, the growth accelerated approximately 27 billion connections [2].

The increased data generated from IoT devices introduces the challenge of privacy where there is a need for a safe and trustworthy-based storage model [3]. The traditional techniques use a centralized data server to control data transfer across different nodes. However, it increases maintenance costs and, therefore, network expenses. Moreover, the centralized hubs constrain the overall performance and are a single point of failure [3]. Consequently, it is necessary to have a secure, trustworthy storage model that improves performance and shall not be a single point of failure.

Decentralized Ledger Technologies (DLT) is currently used in financial applications as a solution for secure peer-to-peer transactions as a decentralized solution. DLT technologies include blockchain technologies such as Ethereum and Hash Directed Acyclic Graph (DAG) technologies such as IOTA. Both these technologies have programmable capabilities on distributed ledgers.

The following are essential advantages provided by decentralized technologies based on IoT platforms [4]

- Decentralized technologies introduced an inherent trust as the data stored are immutable [4].
- Costs related to a “trusted middleman” are removed [4].

With these advantages, the technology can be considered a solution for the challenges faced by IoT industries to store data in a safe and trustworthy environment.

One of the applications benefitting IoT and distributed ledger technologies integration is logging operational maintenance data. IoT devices are used to monitor critical equipment and log maintenance data. DLT and IoT integration generate immutable operational data and maintenance records [5]. The thesis presents a use case for storing IoT operational maintenance data in a decentralized ledger environment. The stored data is accessible to the machinery owner and service providers, providing an inherently trusted storage mechanism.

1.1 Objective

Consider a process industry where service companies need IoT operational data and the status of the previous service history of machinery. Service companies use the data to identify trends in the data to plan preventive and periodic maintenance of equipment. A distributed technology can be seen as is a data storage mechanism as a replacement for traditional database or cloud solution to store data.

Figure 1-1 shows the conceptual view of how a piece of machinery is monitored using several IoT sensors to update maintenance data using distributed technology. Maintenance data of equipment is collected using IoT sensors. The data is stored in distributed ledgers such as Directed Acyclic Graph (DAG), IOTA chain, or Blockchain (BC) such as Ethereum chain [3], [6]. Service companies use stored data for analysis and storing maintenance data ledgers. Service companies can write service records to the distributed ledgers after a completed service. Equipment stakeholders can read data from the distributed ledgers to know more about the equipment’s health and service history.

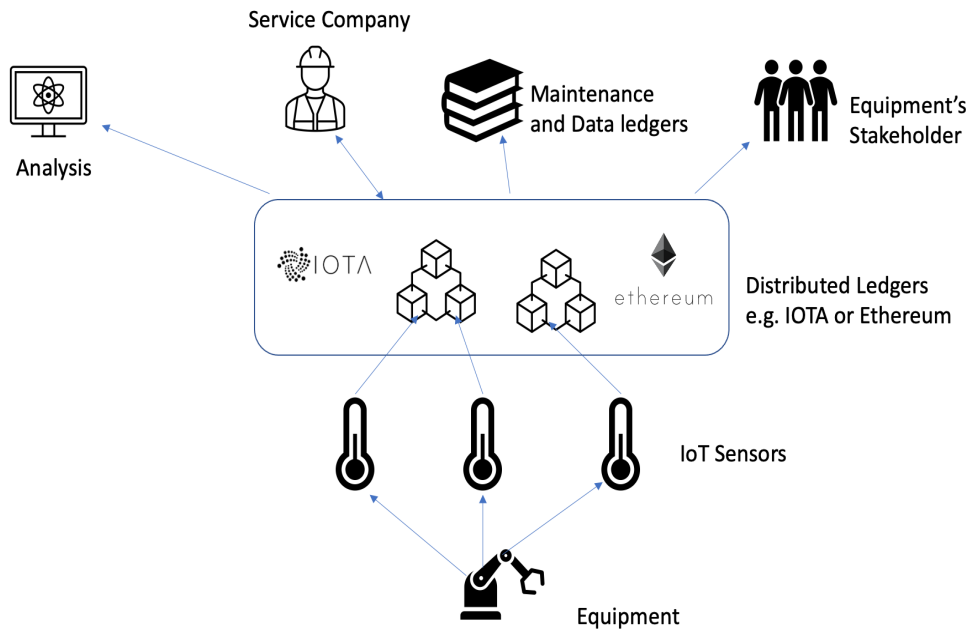


Figure 1-1 Conceptual view of the use case

The following are the core objectives of the project:

- Presenting a review of the use of distributed ledger technology in the internet of things domain.
- Connecting to IoT devices using the python programming language.
- Developing distributed ledger programming to handle decentralized operations.
- Use of python’s libraries for connecting and writing data to decentralized technologies.
- Developing front-end client applications.
- Evaluating and testing decentralized technologies for storing maintenance data, service data, and user interaction.
- For the specific use case, a comparative study of the two types of DLT technologies: Blockchain (BC) and Directed Acyclic Graph (DAG).
- Discussing the limitations and challenges of integrating distributed ledgers in IoT domain and propose possible improvements.

1.2 Outline

This section provides a summary of the chapters of the thesis.

- Chapter 2 presents a general concept of the Internet of Things technology. The section introduces IoT devices in operational maintenance, where the collected data helps in periodic maintenance. The section also discusses the benefits of IoT devices. The chapter also covers an overview of distributed ledger technologies, both blockchain and DAG technology. The general structure of Smart contracts and their applications are presented. The applications and challenges of DLT and IoT integration are also a part of the chapter.
- Chapter 3 discusses the developed system concepts and specifications. The chapter introduces the system overview and system requirements.
- Chapter 4 gives a detailed explanation of the system implementation.
- Chapter 5 presents the results of the work. The chapter also discusses the comparison between BC and DAG technologies and carries out a general discussion of the thesis.
- Chapter 6 discusses some conclusions and proposals for future work.

2 Background

The chapter presents the general concept of the Internet of Things, Distributed Ledger Technologies, and the integration of IoT and DLT. A review of recent research papers in the field of DLT and IoT is also presented.

2.1 IoT Concept

Significant developments in wireless sensor technology and informatics have paved the development of the Internet of things (IoT) technologies. The Internet of Things is present on both a personal and professional level. The IoT is critical in improving living standards through e-health and smart homes. In addition, automation, smart supply chain and transportation, remote monitoring, and shipping industries are all areas where IoT is utilized [11].

Kevin Ashton and his work at Procter and Gamble use RFID tags and introduced the term “IoT”. The concept of IoT has grown from simple RFID tags to a giant ecosystem where it is estimated to have over 1 trillion Internet-connected devices by the year 2030. IoT devices have affected various sectors such as industries, health, and consumer markets [8].

The following are the basic requirements of a device to be considered part of the IoT [8].

- IoT devices should be powerful enough to host an Internet protocol software stack.
- The hardware and power capacity should be ethernet or wireless network connection.
- IoT devices cannot be traditional Internet-connected devices, such as PCs, laptops, servers, etc.
- IoT devices should monitor data from the environment.

In short, IoT devices are compact, intelligent sensors that collect data from their environment; the data is sent to a server or a cloud for storage and processing. The processed information is provided structured via an internet application [9]

Traditionally, the Internet introduced the concept of interconnecting communication devices and enabled end users to communicate using the technology. With technological advancements and Wireless Sensor Networks (WSN), real-time data flow from the sensor devices was possible. The amount of data generated from these devices paved the way for an ample storage cloud-based solution. The specialized cloud services are designed to collect data from WSNs. The results are reverted via the Web by evaluating data precipitously when client requests are received. However, cloud and web-based evaluation introduced security concerns in the IoT applications [9].

RFID tags in factory production materials are an excellent example of IoT solving process management bottlenecks. RFID readers can solve segregation according to the production process and improve the production process’s overall efficiency. Still, it also introduces traceability since the smart sensor devise transfers the information to cloud service, which can be analyzed later [9].

IoT applications include lifestyle, health care, smart cities, industries, and environments. The data collected from the smart sensors are stored in cloud environments, creating a “big data” [9].

2.2 IoT benefits

There are several benefits of IoT [9]:

- Distributed intelligence where the need for a central control unit is removed. The smart device on the network can make decisions based on the environment and situations.
- Communication speed is improved with the development of specific protocols such as OPC UA to facilitate machine-to-machine communication.
- Since the development of IoT devices is cheap, several IoT devices are used to collect information. The collected data is evaluated and analyzed for business decision processes.

2.3 IoT requirements

There are three requirements categories for the IoT devices [10].

- Low resource consumption: IoT devices generate a vast amount of data. Therefore, power consumption is an important criterion in the design phase to increase the lifetime of the device and the life of onboard batteries.
- Widespread interoperability: Due to the integration with web-based technologies and information analysis, it is vital to have IoT devices able to communicate with both the web and other IoT devices.
- Nano transactions: Several transactions are generated by IoT devices, which need to be maintained without sacrificing security or performance. There shall be an infrastructure to handle such micro-transactions.

2.4 IoT in operational maintenance systems

Several sensors can measure temperature, pressure, humidity, etc., directly impacting the equipment life cycle in an industrial environment. Introducing IoT sensors embedded in the machinery makes it possible to identify the condition to diagnose equipment failure. By monitoring these indicators, we can plan preventive maintenance and, to some extent, delay planned maintenance [11]. Predictive maintenance (PdM) helps to reduce maintenance costs and ensure sustainable operational management [12]. PdM aims to predict the next error to perform preventive maintenance before the failure [12]. With several machines in an industrial environment, it becomes challenging to survey these data manually. Therefore, it is necessary to introduce automated data collection and analysis. Industrial giants such as General Electric have developed service-oriented business models using IoT sensors to monitor the performance of their products [11]. Preventive maintenance helps promote sustainable production practices by extending the equipment's life [12].

Classic models used in the modern period primarily use age-dependent failure rates. The models do not consider equipment deterioration [11]. With IoT sensors and the introduction of big data and destructive technologies such as distributed ledgers, failures are detected using a detailed analysis of the equipment characteristics. For example, consider a diesel engine; by using vibration and acoustic sensors, a healthy state of the machine can be plotted. A variation on the sensor patterns can assist in predicting equipment failure characteristics. Temperature sensors are also an excellent indicator of the failure distribution [11].

Using data-driven Artificial Intelligence (AI) applications utilizing data collected from IoT devices can help in the excellent data analysis [12]. In addition, AI analysis helps in preventive maintenance. However, the use of AI in the analysis is not in the thesis scope.

2.5 Distributed Ledger Technologies

Centralized and decentralized systems are the two types of data storage systems based on how data is stored. The centralized systems are vulnerable as a failure in the central authority can cause the collapse of the whole system [3]. There is an additional challenge of privacy where a single source has access to the data. The use of distributed technology-based storage solves the limitations of centralized systems. This section discusses the basic concepts of distributed ledger technologies. Distributed Ledger Technology (DLT) uses several technologies such as private/public-key cryptography, hash functions, databases distributed across various nodes in the network, credible consensus algorithms, and a decentralized processing [13]. DLT is a distributed ledger where the owners of digital goods can transfer from peer to peer, and a register stores the transfer information [14]. The technology is tamper-proof and immutable, where the decentralized network maintains a copy of the whole database, which is synchronized [3].

Figure 2-1 illustrates a server-based centralized network and a blockchain network.

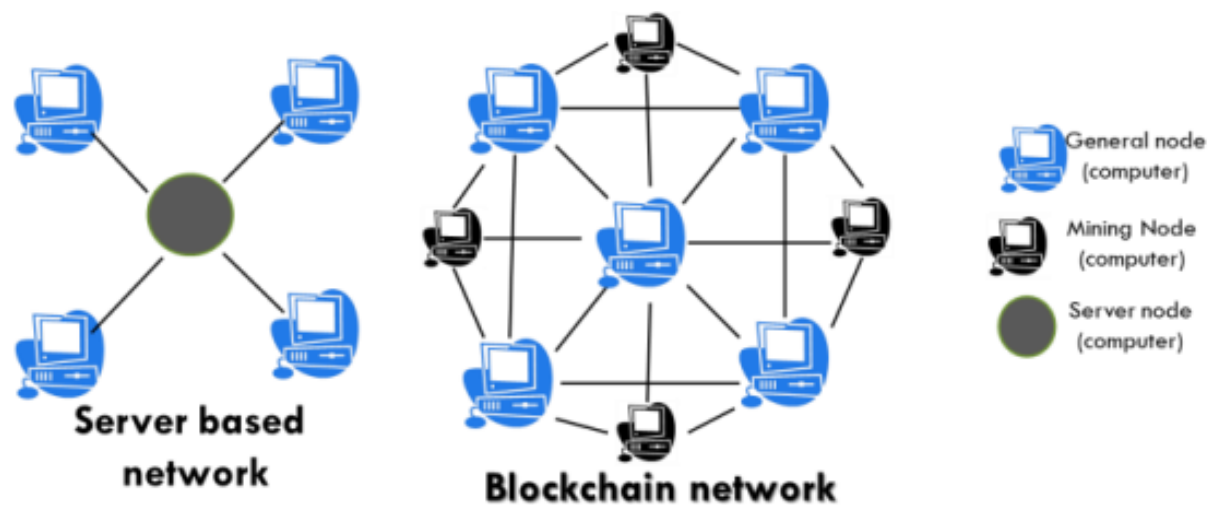


Figure 2-1: Server-based network and Blockchain network [15]

The two main types of distributed ledgers technologies are:

- Blockchain technology.
- Directed Acyclic Graphs (DAGs).

2.6 Blockchain

Blockchain (BC) technology is a fully distributed and trusted ledger technology that Satoshi Nakamoto initially found to store and transfer Bitcoin cryptocurrency from peer to peer [3]. The BC technology removes the need for a centralized authority. The technology is tamper-

proof and makes data modification impossible [2]. All the nodes in the network store a copy of the data in a ledger and are updated simultaneously. The data update method avoids any data loss [3].

Several case studies of the application of BC technology are implemented across different fields. The research studies ranged from developing Blockchain-based IoT solutions to blockchain-based industrial applications [13]. The research work made a way for applications in Blockchain called Distributed Applications (DApps) [13].

2.6.1 How BC works

BC technology uses cryptographic principles. Every node has both public and private keys. A public key is for encryption used by other nodes to encrypt data targeted to a particular node and is the unique identifier for each node. A private key allows you to decrypt and read messages to the node. In addition, the nodes use private keys to authenticate and sign transactions. After approval, the node's peer receives the transaction and broadcasts it through the network. Each node verifies the transaction's validity before retransmission, confirming data integrity. Specific nodes called miners organize the proven transactions into blocks. The Blockchain's consensus algorithm (validation algorithm) determines miners and data on each block. "Genesis block" is the initial block in the chain. Each block contains a header and a transaction list. Each block's header includes the previous block's hash, and the block's transaction list has data transactions. Since the hash depends on the parent, if there is any change in data – it will affect consequent blocks. Therefore, any data change will require a new validation. The network nodes verify that the integrity of previous transactions is intact, and that the new block's hash matches the last block. The ledger now includes a new block when transaction and hash are confirmed [3] [13].

Figure 2-2 shows the basic blockchain structure.

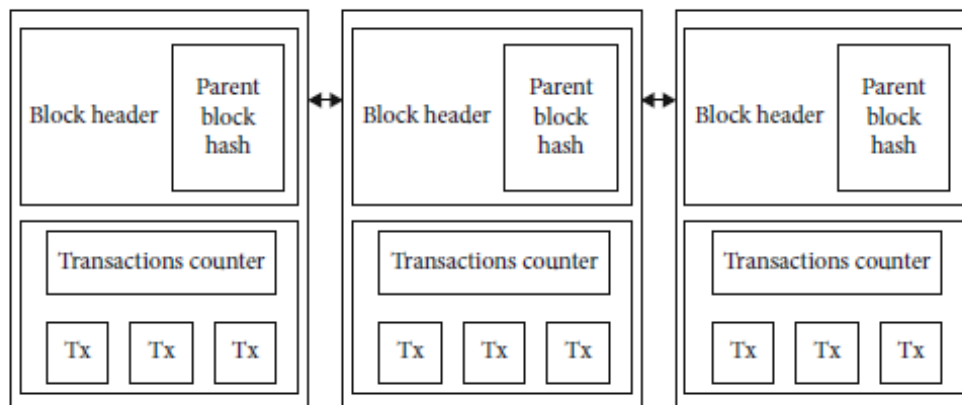


Figure 2-2Blockchain structure [13]

2.6.2 Types of BCs

BCs can be divided into the following three types:

2.6.2.1 Public Blockchain

A public blockchain is a decentralized distributed system where members can share and verify blockchain transactions. The usage of a trustworthy third party is not necessary for authentication. Such chains are usually utilized efficiently in the financial industry for transactions like digital currencies, tokens, international payments, and other areas like crowdfunding. Despite these advantages, there is a disadvantage: writing and processing times may be slowed when there is an increase in the number of participants [15].

2.6.2.2 Private Blockchain

A private chain is related to the public chain, with the difference in that not all can participate in the chain. The service provider limits the individuals who can access the chain. Due to the limited access, the blockchain is seen as a centralized chain with improved security and transaction speed. Only nodes that can take legal responsibility are authorized to approve and verify a transaction. Block creation time is short as only approved nodes participate in the chain. The reliability of the chain is often in question as the nodes depend on the service provider [15].

2.6.2.3 Consortium Blockchain

A consortium blockchain limits users by approving only those with certain participation requirements. Several levels of authorization limit participants from viewing all or part of transactions. These chains are semi-centralized where the participants are companies or organizations [15].

2.6.3 Consensus Algorithms

Safety and liveness are the core principles for designing consensus algorithms. Safety is the ability to handle corrupted or out-of-sync messages so that all non-faulty nodes can reach a valid consensus according to the state machine's rule. The liveness of a system means that despite faults, non-faulty nodes can continue to handle other distributed processes. However, it is possible for a distributed system running a consensus protocol to fall under "Byzantine fault". The fault is where a node sends false messages and misleads other nodes. A node developing such a fault may be due to software bugs or compromised. Several consensus algorithms can handle such defects by considering network performance, uses-cases, and maliciousness of compromised nodes [16].

Publicly deployed blockchains cannot use voting as a consensus mechanism. Therefore, the participants can use multiple accounts in Blockchain and launch a Sybil attack to drive decisions in their favor. In such cases, the lottery-based selection is employed in the consensus algorithm, where a single node publishes new blocks in the Blockchain. Block creation is intentionally set as "expensive" to avoid biasing consensus decisions in a participant's favor [16].

Figure 2-3 shows a typical blockchain process with consensus for transaction validation.

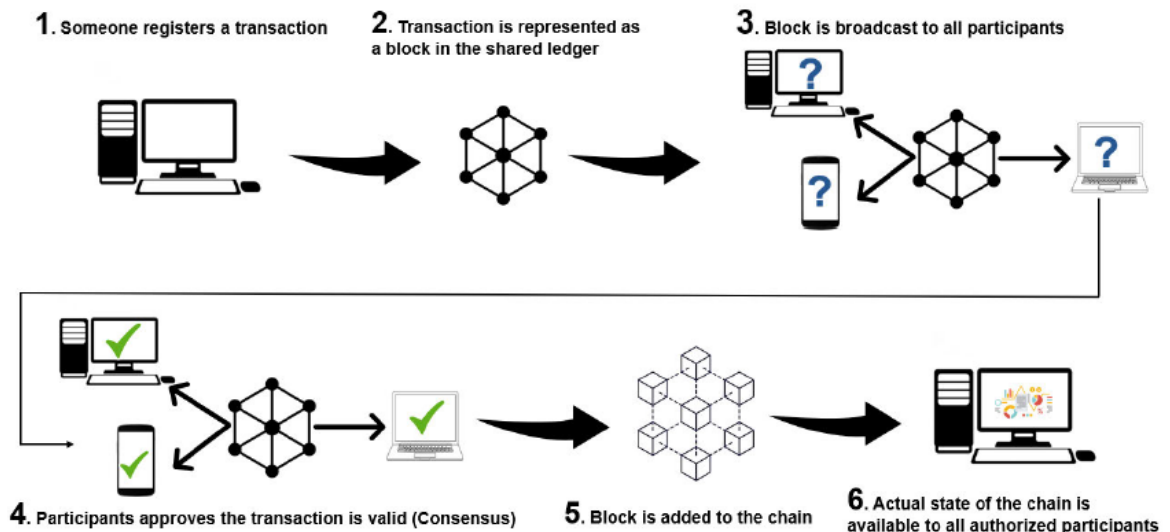


Figure 2-3 Blockchain Process with Consensus Mechanism [3]

2.6.3.1 Proof of work

Proof of Work (PoW) is Bitcoin's first public blockchain consensus algorithm. Any node can publish a new block to the Blockchain when performing computationally intensive work. Posting new blocks is called “mining”, where participants race to find the value of nonce when hashed with the hash of the block results in a resultant smaller than the predefined threshold. As the number of participants rises, the difficulty level of the threshold is changed. The block header records the changes to maintain an average block processing time. After calculating nonce, the miner adds the value to the block header and broadcasts its block to the network. Participating nodes verify this, and the miner receives the processing fee associated with the transaction as a reward. Expensive block creation and transaction fees secure the network from attacks and false block creation [16].

2.6.3.2 Proof of Stake

An environmentally friendly alternative for PoW is Proof of Stake (PoS). The consensus algorithm aims to cut down electricity costs associated with PoW. Here the algorithm seeks to stake the economic share of peers in the network. “Validator” corresponds to “miner” in PoW. One of the validators publishes a new block onto the Blockchain in the concept. The validator selection is in a pseudorandom method where the probability depends on the proportion of the validator’s share in the network [16].

2.6.3.3 Proof of Activity

Proof of Activity combines the features of the proof of work and proof of stake. The concept focuses on rewarding stakeholders who are actively participating in the network. The first step is mining potential block headers, followed by a random group validating the mined block header. The probability of the validator selection is proportional to the peer’s share in the network. Finally, the miner and validators share transaction fees. There are several criticisms of the type of consensus mechanism due to its high energy requirement. Additionally, there can be a possible “nothing at stake” attack [16].

2.7 DAGs

DAGs or Directed Acyclic Graph is an excellent alternative to solve scalability issues of BC technologies. Unlike a blockchain containing blocks, DAG is a data modeling tool with vertices and edges. The transactions are stored on top of one another, known as vertices. The transactions are submitted to the DAG via nodes, where nodes must perform PoW as a consensus mechanism. The model allows for more transactions, and the wait time for transaction completion is considerably reduced compared with BC. Obyte, IOTA, and Nano are a few examples of DAG [17].

2.7.1 IOTA

IOTA means infinitesimally minor in Greek, adequately coined since it enables micropayments [18]. It is an open-source distributed ledger technology known as the “cryptocurrency without a blockchain” [4]. The technology allows data transfer between IoT devices based on Hash directed Acyclic Graph known as the “Tangle” [4]. Unlike miners in the Blockchain, IOTA uses a PoW consensus algorithm. Hence, there are no fees associated.

Figure 2-4 shows the structural difference between a blockchain and IOTA tangle.

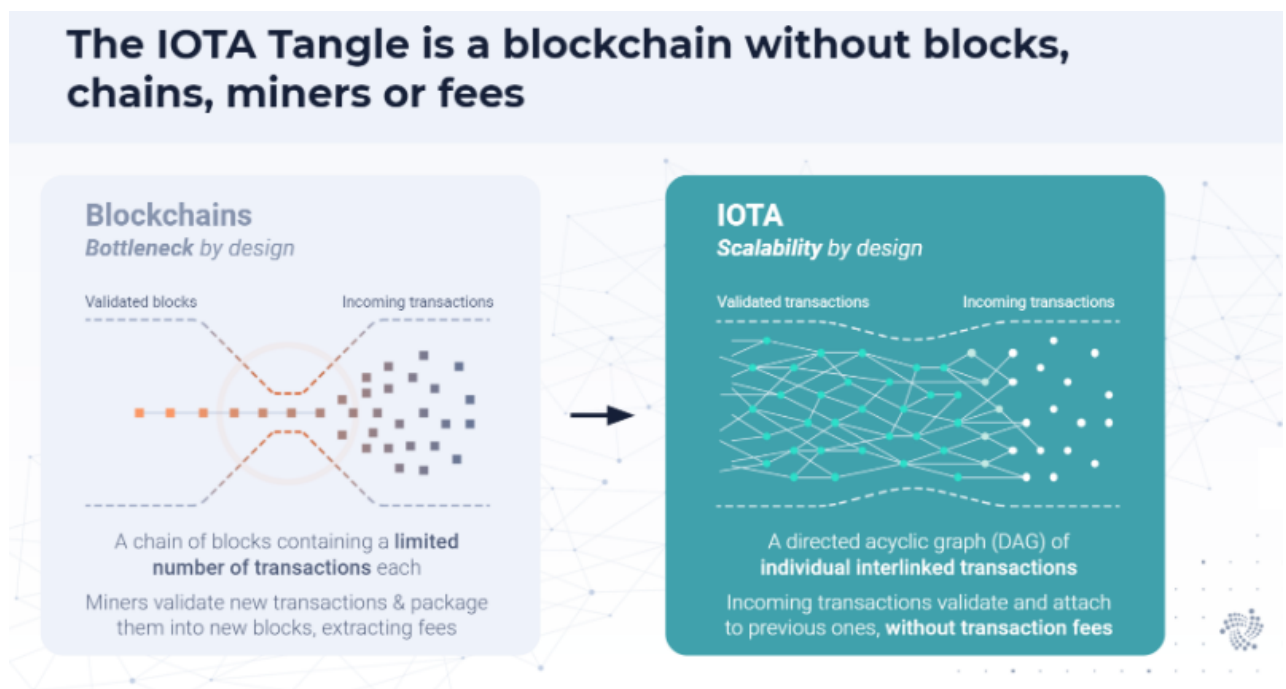


Figure 2-4 BC Vs. IOTA ledger [6]

2.7.1.1 How IOTA works

The tangle is a special type of “DAG”, a lightweight protocol, and a block-less ledger for storing transactions specially designed for IoT devices. A tangle is a collection of vertices (transactions) connected by edges. Each edge is a participant whose consensus mechanism is the Proof of Work (PoW) [10].

Proof of work is carried out in three steps [19]:

1. Construct a transaction bundle that contains individual transactions for the value transfer.
2. Randomly choose unapproved transactions or call tips in IOTA to join the tangle.
3. Calculate the nonce to join the transaction as vertices using PoW.

Figure 2-5 shows the proof of work algorithm and how IoT data is bundled into the IOTA network.

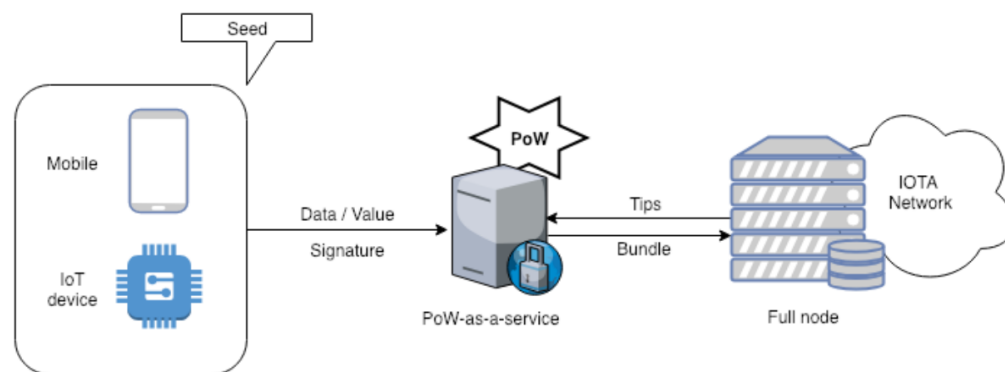


Figure 2-5 IOTA Proof of Work [19]

The following are the main differences between IOTA and blockchain [6]:

- In BC, blocks follow the last block and can only be attached to a single point. Transaction approval occurs once miners include the transaction in a newly issued block. Usually, miners decide which transaction they want to include in the block. Therefore, fees play a major role in BC. Miners favor users willing to pay a higher fee for the transaction. In the IOTA tangle, there are no miners, and all users can issue a new transaction and attach it to the tangle. It is a network of parallel processed transactions called Tips. All transactions are included as long as they do not break the rules of the protocol [6].
- Blockchain must have a single leader, the block producer, who will earn all transaction fees. The situation will introduce a “miner race,” meaning that a miner with the highest computational power or staker with the highest stake is highly likely to be the next block leader. IOTA is a leaderless protocol, and everyone is free to attach transactions. IOTA is a multi-threaded ledger with a high transaction throughput [6].
- IOTA has a coordinator – a temporary solution to send signed messages. There is no coordinator in blockchains. Upcoming IOTA 2.0 will remove the coordinator. A coordinator is a client who can send signed messages called milestones that nodes use to confirm messages. Messages in the tangle are confirmed only when validated nodes directly or indirectly reference the message. To ensure that new messages always have a chance to get validated, the coordinator sends indexed milestones every 10 seconds.

It illustrates how the coordinator monitors new transactions and confirms milestones [6].

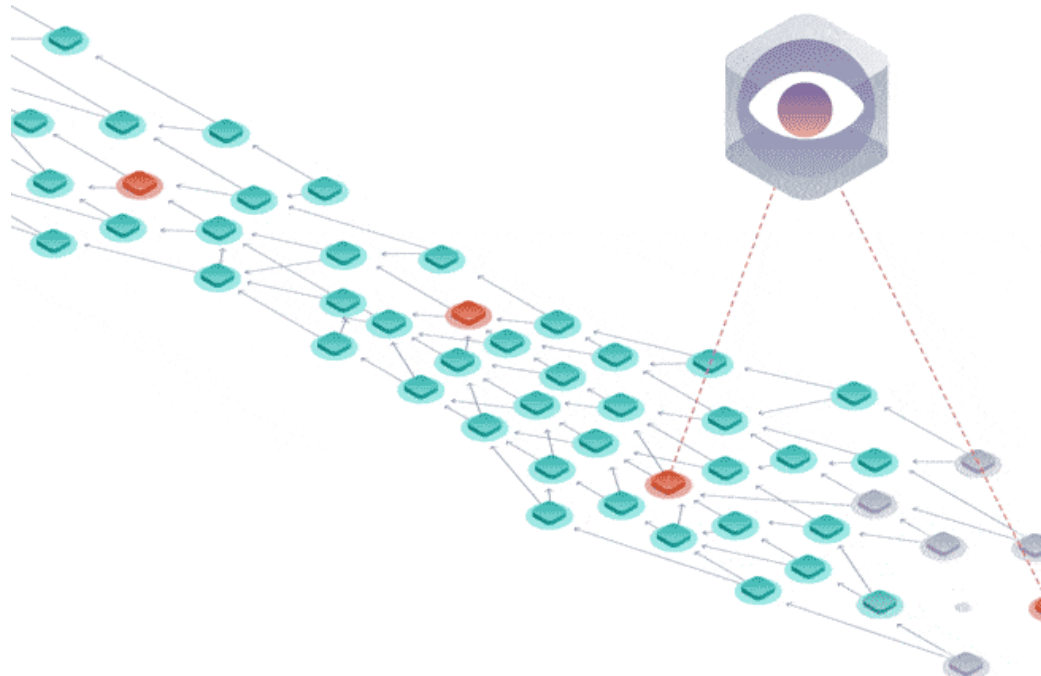


Figure 2-6 Coordinator confirming Milestones [6]

2.8 Smart Contracts

Smart contracts are computer programs deployed and automatically executed on the distributed ledgers. The program is triggered when addressing a transaction after the smart contract is deployed in the chain. Once triggered, the execution is automatic and is carried out by every node on the network. The deployed smart contract cannot be modified; therefore, it is not possible to tamper with the code [10]. In short, smart contracts are programs running on a distributed network. The main requirement is that the ledger state should be agreed upon by all nodes running it [20]. For example, a smart contract can contain some tokens (currency in the distributed ledgers) for handling land ownership. The smart contract accepts both tokens and the land ownership deed. Then the code will predictably exchange them between both parties. It is therefore making it impossible not to deliver the promise. In this case, the smart contract code is law [21].

Solidity is the programming language used by developers in the Ethereum chain to create smart contracts. Ethereum is a transaction-based state machine. The first state is called the genesis state, which is modified with the execution of each transaction. The transaction is validated based on the information in the latest block accepted to the network. Ethereum is made of two accounts – externally owned accounts and contract accounts. Contract accounts are referred to as smart contracts. Both accounts contain a transaction counter called nonce, balance, and the ability to send transactions. Smart contract accounts differ from the externally owned account as it contains contract code and contract data storage [22].

In the case of IOTA, the smart contracts are called IOTA Smart Contracts (ISC). ISC attempts to solve the scalability challenge in the DLT domain. Scalability leads to low throughput and high transaction costs [23]. IOTA’s ISC provides multi-chain DLT, which hosts parallel. IOTA 2.0 is fully decentralized without a coordinator and implements Smart contracts [23]. IOTA 2.0 runs on the development network as of this writing. The structure of the IOTA 2.0 design has several layers which separate messaging, transacting, and validating components from the application layer. The application layer contains smart contracts. Smart contracts run on Wasp nodes on layer two connected to Goshimmer on layer 1. Layer 1 handles transactions and messaging [23]. The current release of IOTA smart contracts has experimental support for Ethereum solidity codes and a native smart contract. Since IOTA fully supports Ethereum code, the thesis attempts to use the same solidity code for developing the system.

Figure 2-7 shows the network layers on IOTA.

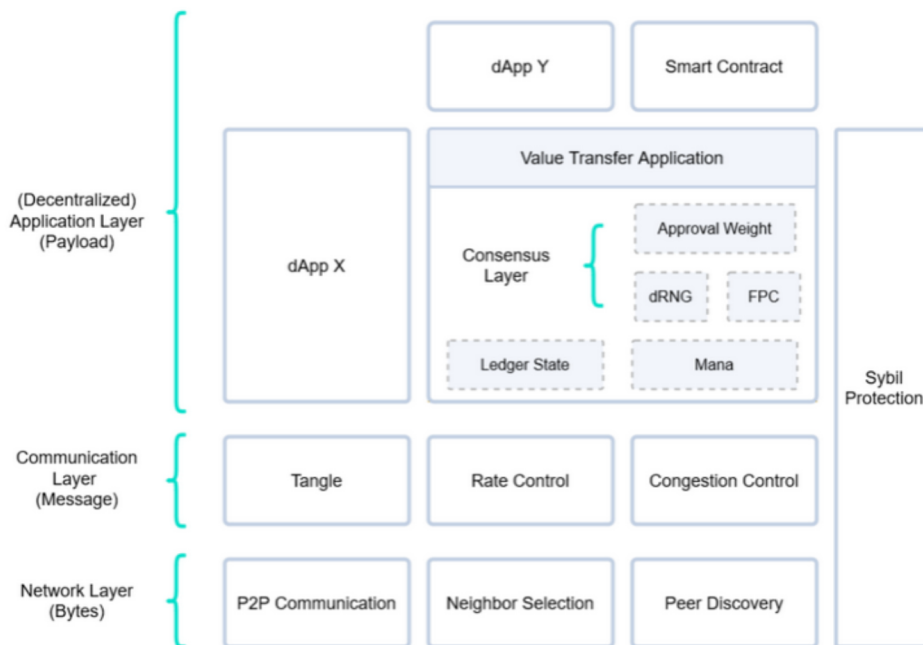


Figure 2-7 IOTA network layers [23]

2.9 IoT and DLT integration

Due to data transparency and immutability, IoT and DLT integration is getting quite popular. The solution now is to store the data locally or in the cloud. As discussed earlier, storing sensitive data in the cloud has inherent privacy issues. In the case of IoT data and especially maintenance data storage, the data needs to be shared with several parties. Data integrity is also vital, particularly since the data estimates maintenance schedules and involves machinery’s health. IoT data handled through the immutable DLT enables classification authorities or a future owner to trace the machine’s history and if the service is done properly at defined time intervals [24].

2.10 Distributed ledger security

Decentralized technologies work without the need for a third party. Therefore, security is the key to ensure trust in the trustless structure. Like traditional computing, DLT also faces attacks such as Denial-of-Service (DoS), code vulnerabilities, endpoint security, and data protection. There are routing attacks that delay the block propagation, eclipse attacks by isolating a victim's network view, and main-in-middle attacks that interfere with network nodes. DLT security is also exposed due to human negligence and insufficient monitoring. The hackers exploit the vulnerability of core software bugs, cryptocurrency exchange platforms, attacks on wallets stealing wallet keys, the vulnerability in smart contract coding, and network attacks such as controlling mining pools. Endpoint attacks inject crypto mining scripts to computers on the web to mine cryptocurrencies [25].

2.11 Literature review

With the emergence of cryptocurrencies and concerns about data privacy, the idea of DLT has become of great interest in recent times. The research paper in the reference [26] discusses IoT networks' cybersecurity and data integrity challenges. Integrating BC helps address these challenges as BC is "security by design" and is immutable, transparent, auditable, encrypted, and resilient. The paper conducts a thorough survey of BC and IoT integration and the challenges faced when integrating BC and IoT. A study of novel uses of BC in machine economy is also a part of the study.

The thesis in the reference [27] proposes a method to gather sensor data from IoT devices and use BC to securely save and recall collected data. They presented a solution of data storage independent of a cloud-based storage solution. The application and performance of Interplanetary File System (IPFS) and Ethereum Swarm are presented. The thesis suggested that since Ethereum requires registration before data transfer and Swarm/IPFS data is stored in an encrypted format; it is not necessary to use a private network. That data can be securely stored in public chains such as Ethereum, IPFS, or Swarm.

The thesis in the reference [22] presents a solution for tracking and tracing products in the supply chain using an application created with Ethereum. The paper develops an Ethereum smart contract system and web-based applications where users can track products. The paper also presents a solution to bridge communication between an Arduino UNO device and a smart contract system. The study presents the benefits and challenges of such a system.

The paper in the reference [28] discusses the use of DLT to reduce security risks in IoT. The paper introduces BC as a technology to eliminate the need for a central node, which at the same time highlights computational costs and limited scalability. The paper presents IOTA as providing unlimited scalability, particularly suitable for the IoT industry. The paper explains the IOTA solution to overcome blockchain's limitations when using it together with IoT.

The current research in distributed technologies focuses on integrating IoT devices into blockchain networks like Ethereum. However, there are limited studies on distributed ledger technology applications storing maintenance data. This thesis aims to extend the IoT- DLT integration for storing operational maintenance data. Additionally, the thesis presents using DAG (IOTA-based) technology as an alternative to blockchain technology. The deployment of Ethereum Solidity code on top of DAG technology is studied and implemented in the thesis.

3 System Concept and Specification

The chapter introduces a high-level overview of the developed application. This is followed by the application's specifications, requirements, and use cases.

3.1 System overview

A typical scenario of machinery monitoring using IoT is presented to understand the solution of storing operational IoT data in a distributed ledger technology.

Consider a factory containing several pieces of machinery. A service company is contracted to carry out maintenance of these machines. Several IoT sensors are installed on the equipment, which will collect information about the state of the machines. The data is stored in distributed ledgers using a program running on an Edge computer. Edge computer is connected to IoT device using USB or communication protocols such as MQTT or Modbus. Once the data is available in the ledger, the service company can process the data and identify patterns to check if the machine needs servicing. The data is presented to the company using a webserver.

The service company can also carry out a periodic service on the machinery. Once the service is completed, the service company will access their client portal and update the last service date of the machinery into the ledger. The data is permanently stored in the distributed ledger.

Stakeholders such as the factory owner, factory operational managers, and authorities can access the ledger using client portals. They can verify if the service is carried out as per the contract. In case of equipment failure, stakeholders and the service company can extract data from the ledger to identify the root cause of the failure.

Figure 3-1 shows the example scenario for storing operational and service data of machinery health.

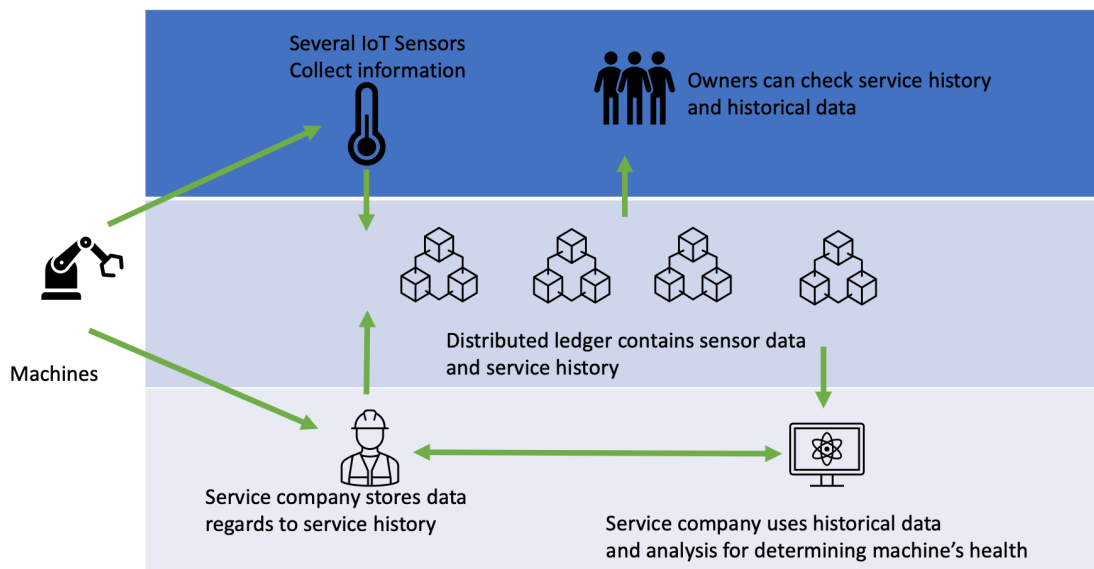


Figure 3-1 Typical Scenario of Machinery Monitoring

3.2 System specification

Considering a typical scenario is explained in section 3.1, the following are the core system specification:

1. The developed system should enable IoT sensors to communicate with the distributed ledgers.
2. The system should permit tracking of sensor name, current sensor value, the record's time stamp, and unit of measurement. The data must be stored as an IoT record in the distributed ledger to track historical data.
3. The system should give service companies, and machinery owners access to read the historical IoT data record.
4. Service companies should be able to write service history records into the distributed ledger and register the state of the machinery health based on their data analysis. The record is called Machinery data record.
5. The developed system should store IoT and Machinery data to Ethereum blockchain and IOTA Tangle using the same smart contract code written in solidity.

3.3 System requirements and Use Case

3.3.1 Requirements

Considering a typical scenario explained in section 3.1, and the system specification explained in 3.2, the following are the system's functional and non-functional requirements. The requirements are classified based on the FURPS+ method.

- **Functionality:**
 - Read/Store IoT data: Read IoT sensor data and write in ledger Ethereum and IOTA chains at a scheduled time interval.
 - Access recent/historical data: Users such as service companies and machinery owners read the latest historical data.
 - Write Service history: Service companies should be able to analyze the IoT data and write the latest service history to the DLT.
 - Access service history data: Users such as service companies and machinery owners read the latest historical service history data.
- **Usability:**
 - Data should be presented in an easily accessible user interface like a web page.
 - Both Ethereum and IOTA chain data shall be available at the same time.
 - New sensors can be added to the code in new sensors/IoT devices are added
- **Reliability:**
 - The script should be running on a continuous loop without any errors.
- **Performance:**
 - None
- **Supportability:**
 - The data shall be available on all web-based devices.
- **+**
 - None

3.3.2 Use case diagram

The use case diagram is developed based on the project's requirement specified in 3.3.1.

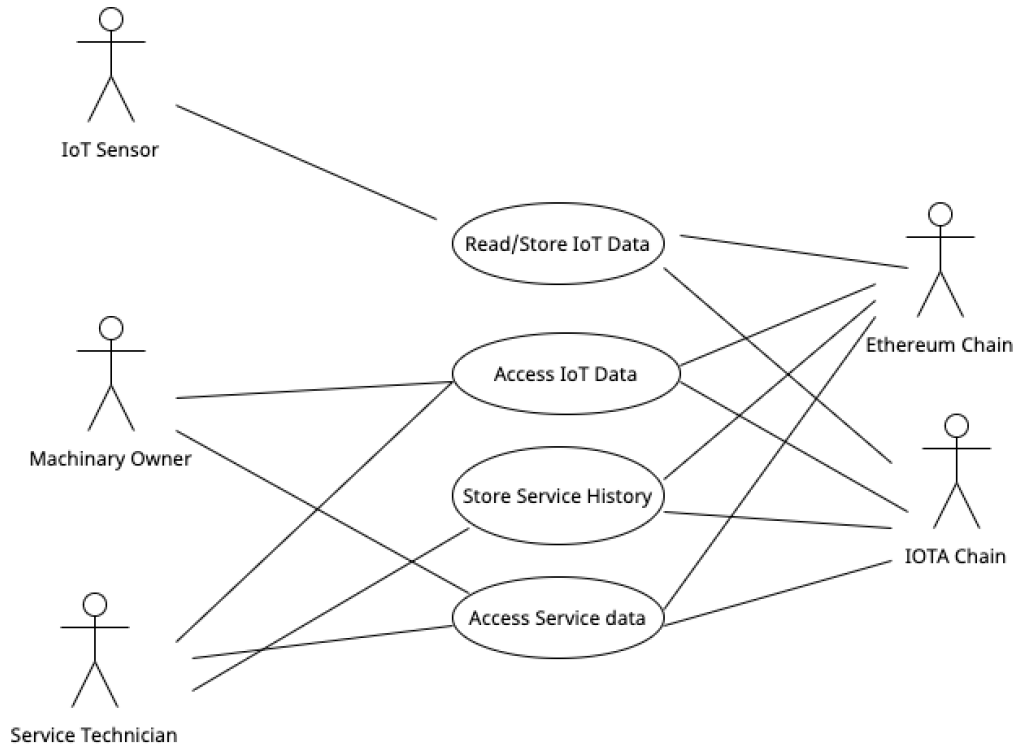


Figure 3-2 Use case diagram of the project

4 System Design and Implementation

The chapter presents the system's design, system architecture, and implementation.

4.1 Sequence diagram for use cases

As identified in the requirements in section 3.3.1, there are four functional requirements. The requirements are converted as a use case in section 3.3.2. Each use case is analyzed and converted to a sequence diagram as a part of the design process.

4.1.1 Read / Store IoT Data

Figure 4-1 shows the sequence diagram for reading sensor data and storing the received data in the DLT. A smart contract is first deployed to the DLT, followed by reading sensor data, reading the smart contract and private key, creating a data bundle with IoT data, and using the smart contract function to write data. The DLT stores the data once the "success" flag is received.

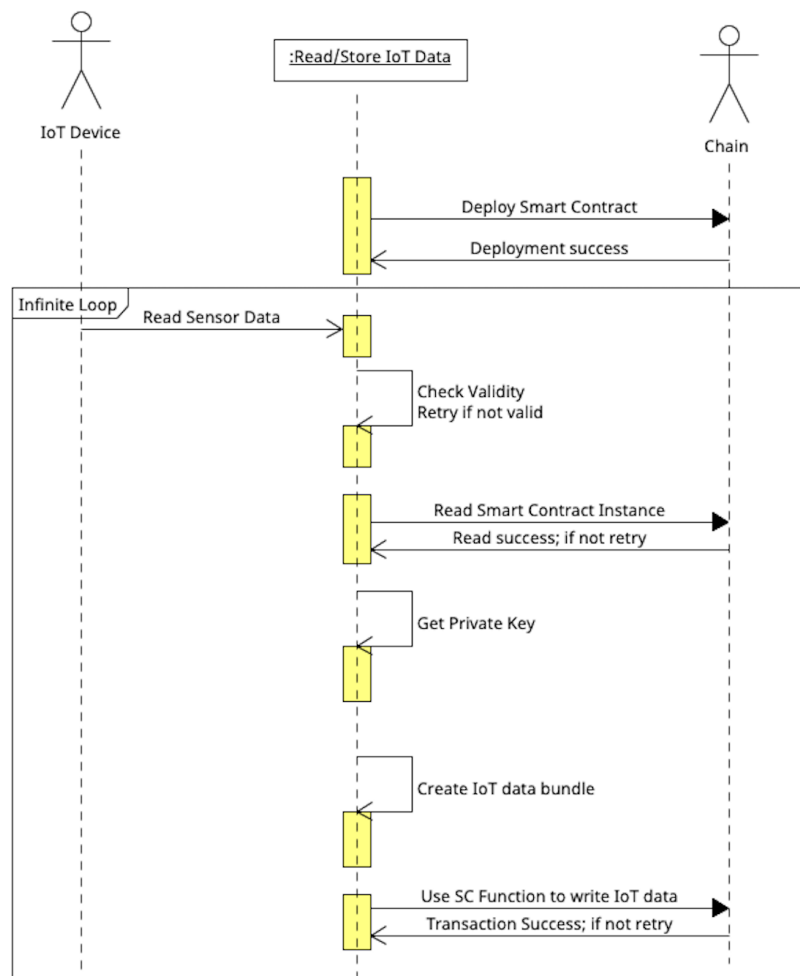


Figure 4-1 Sequence diagram for reading and storing IoT data use case.

4.1.2 Access IoT Data

Figure 4-2 shows the sequence diagram for reading data from the DLT. The securely stored private key is used to initial smart contract functions and communication to the DLT. A smart contract function written to read data is called to get the latest block. The function parameter (block number) can be adjusted to call data from blocks. Once data is received, it is sent to the request initiator.

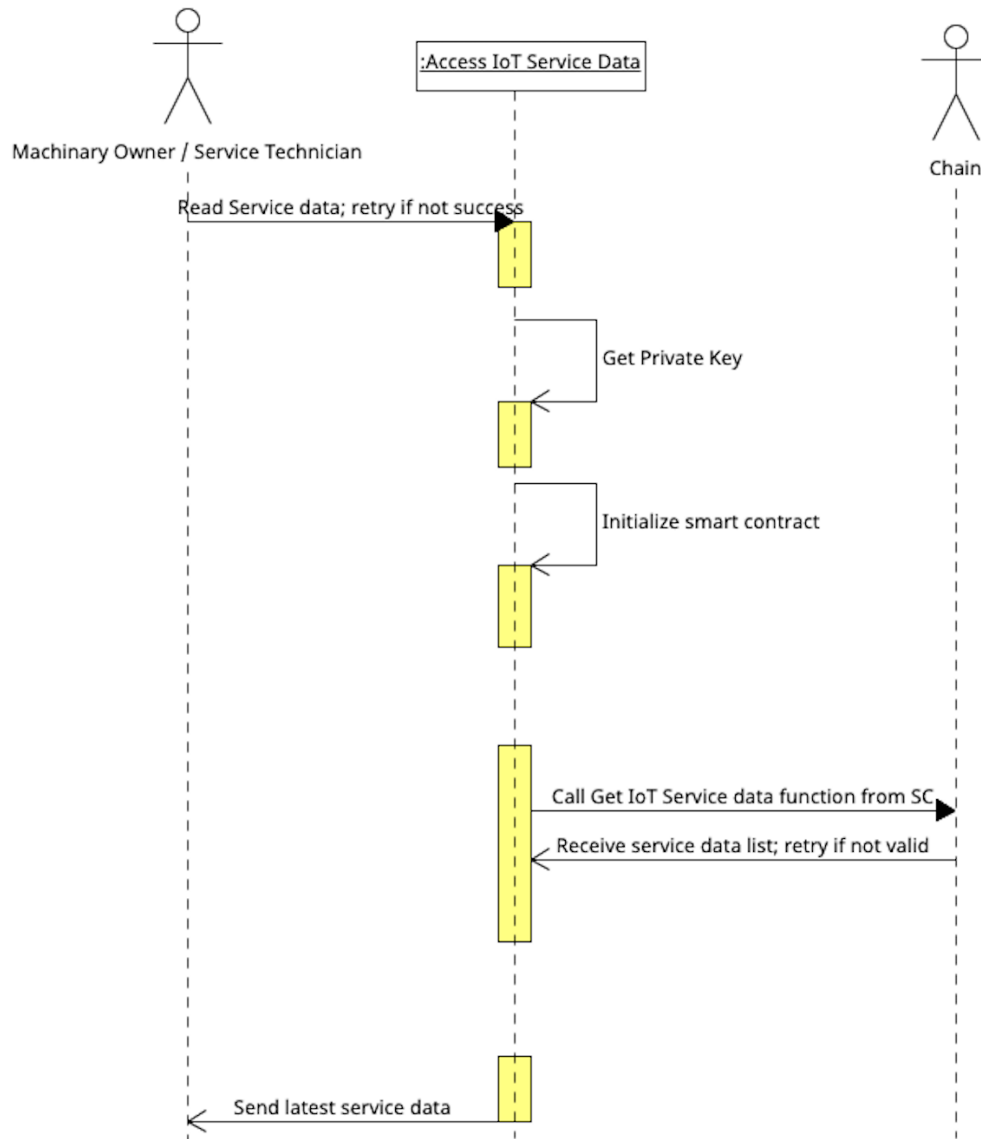


Figure 4-2 Sequence diagram for Accessing IoT data

4.1.3 Store Service History

Figure 4-1 shows the DLT’s sequence diagram for writing in-service data. First, the smart contract and private key are called and using the smart contract function to write service data, the input from the service technician is written to the DLT. The DLT stores the data once the “success” flag is received.

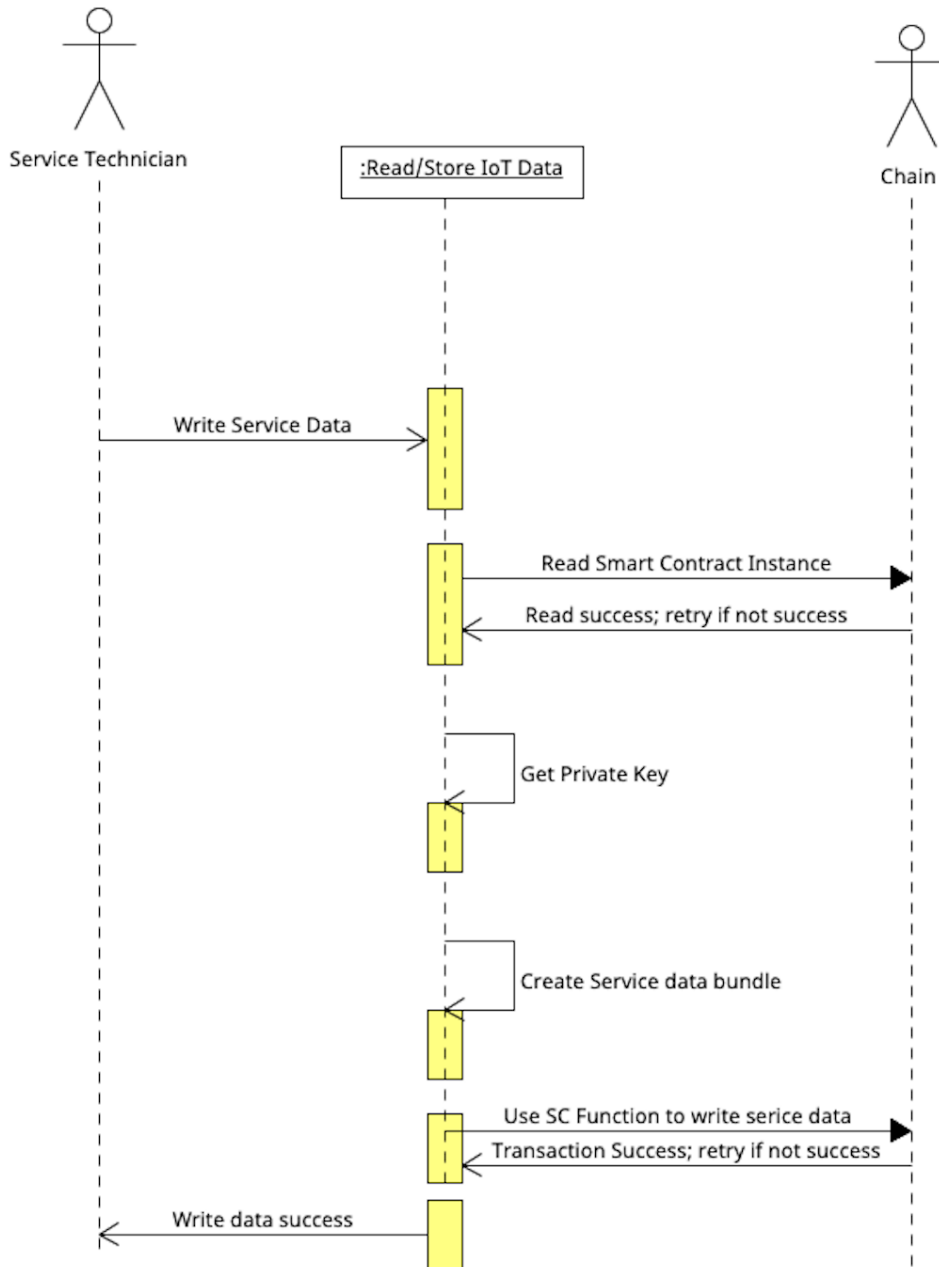


Figure 4-3 Sequence diagram for Writing Service data

4.1.4 Access Service Data

Figure 4-4 shows the sequence diagram for reading service data from the DLT. Like section 4.1.2, securely stored private keys are used to initial smart contract functions and communicate to the DLT. The smart contract function written to read service data will return the value of service records. The latest block is called to get the latest service data records. Once data is received, it is sent to the request initiator.

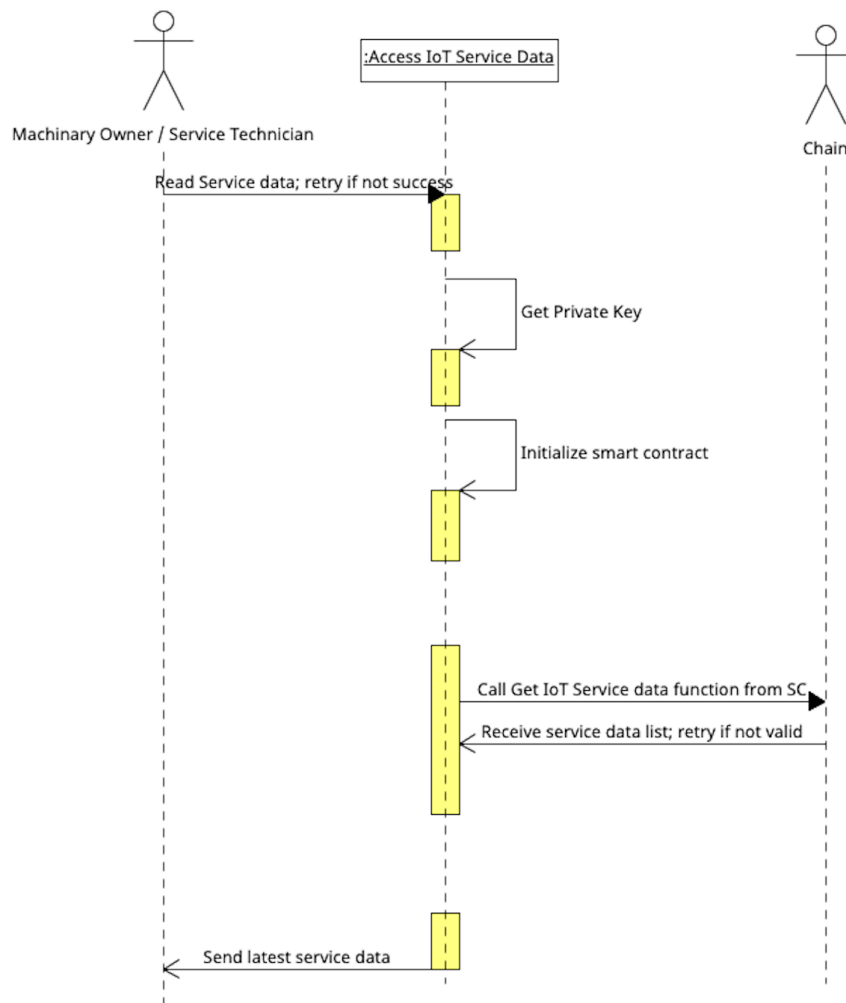


Figure 4-4 Sequence diagram for Accessing Service data

4.2 System architecture

Figure 4-5 shows the architecture of the developed system containing the presentation layer, interface layer, and Distributed Ledger (DLT) layer. System Architecture presents various components used to implement functions in the application. The developed application utilizes all three layers to handle the use cases discussed in section 3.3.

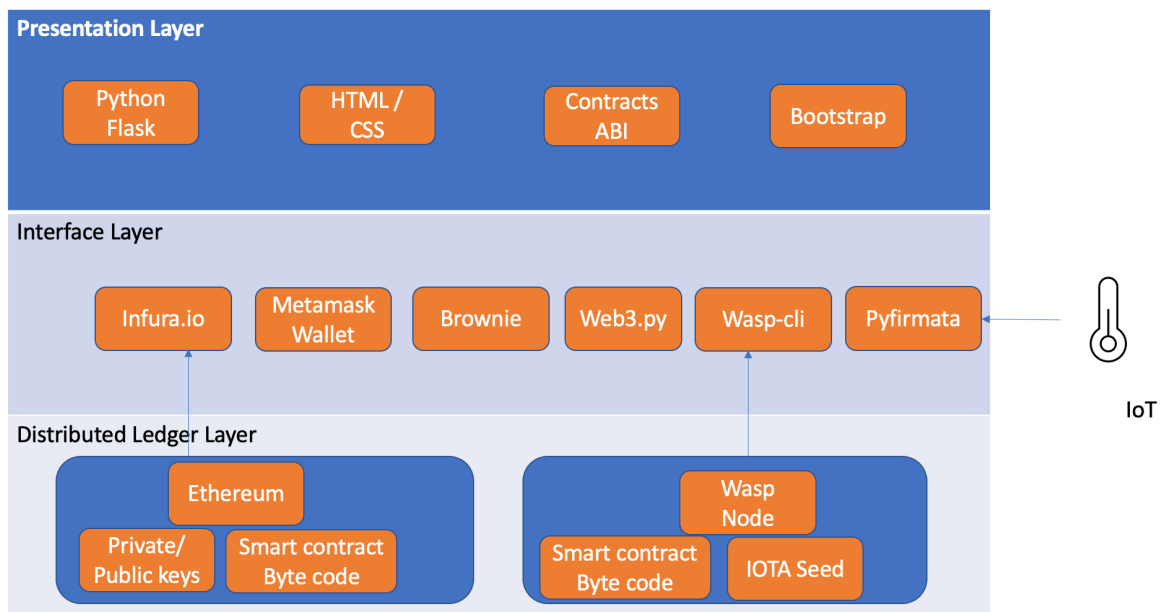


Figure 4-5 System Architecture of developed system [29]

- Distributed Ledger Layer – contains both the Ethereum blockchain and IOTA tangle components. Ethereum blockchain is selected due to its popularity and support of smart contracts. IOTA is selected due to its specific design for IoT applications, fee-less structure, and recent development to migrate codes written in Solidity (Ethereum programming language).

Ethereum nodes are the backbone of the application, containing all the network computers storing the data and the consensus algorithm that confirms the data validity [29]. There are two accounts in the Ethereum blockchain layer, irrespective of the type used. The two accounts are Externally Owned Account (EOA) and Contract accounts [29]. EOA is a regular account that holds the private key, public key, account address, and balance. Contract Accounts has contract-byte-code, which executes on EVM, contract address, balance, and storage [29].

Wasp node is the IOTA version of Ethereum Node. A software node on top of the IOTA tangle validates smart contracts [6].

There are two storage records in both chains. They are IoT Data records and Machinery status data records. IoT Data record stores sensor tag name, the time stamp of the data entry, sensor data and the sensor, and sensor unit. Machinery data record contains the last service date, next service date, the result of the service, service company, and engineer name.

4 System Design and Implementation

- The interface layer manages transactions on the blockchain and IoT devices. The layer is the interfacing component acting as the intermediary between the blockchain and the front end.

The following are the components in the interface layer of the project.

- *Pyfirmata* module in python for interfacing Firmata protocol to communicate with IoT device. Firmata protocol for communicating with microcontrollers implemented in firmware on any microcontroller architecture [30]. For the project, firmata for Arduino is implemented.
- Metamask wallet is a simple crypto wallet that runs a browser extension. For the project, Metamask is installed as a chrome extension. Ethereum test net is used in the implementation. The cryptocurrencies stored in the wallet are test Ethers.
- Infura.io is an Ethereum backend service provider that gives access to the Ethereum chain without running an Ethereum Node. The Infura Ethereum API (application programming interface) enables developers to connect to the Ethereum chain via WebSocket and HTTPS [29]. Figure 46 shows an overview diagram of how Infura connects devices to the Ethereum network.

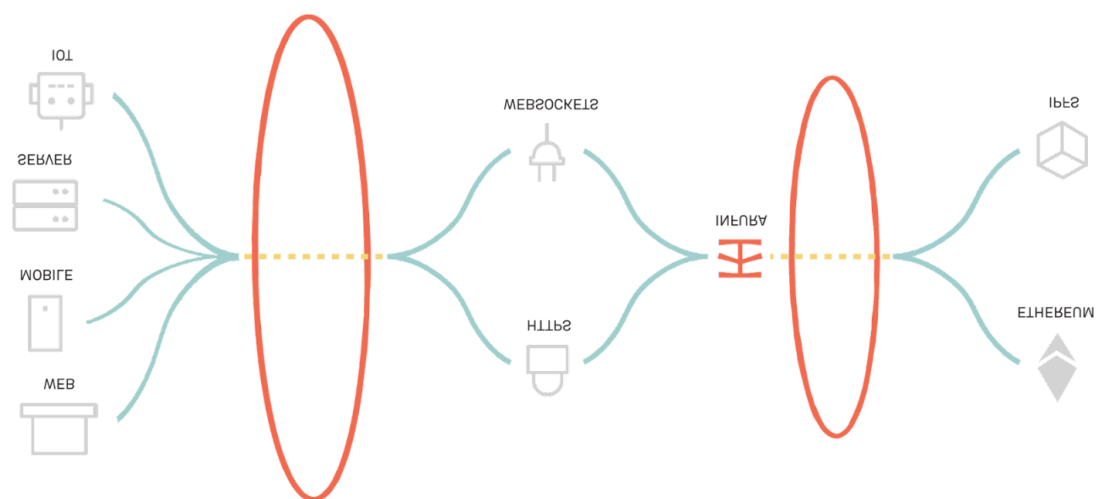


Figure 46 Infura overview diagram[31]-

- *Brownie and Web3.py* are two python libraries used to interact with the Ethereum network via Infura.io. *Web3.py* is a python version of web3.js, a library collection that allows interaction with Ethereum nodes via HTTPS, IPC, or Web Socket [30] [31]. *Brownie* is a smart contract development and testing framework based on the web3.py library [34]. Using *Brownie* compiling smart contracts and testing, the logic is simplified. Front-end interaction via python is limited in *Brownie*. Hence it was necessary to use *web3.py* to interact with the python front end. A brownie handles the use case for storing IoT data on the DLT. Displaying IoT data and service data on the front end to store service data is handled by web3.py. Since IOTA also uses Ethereum support, *brownie* and *web3* libraries are used similarly for the IOTA chain.
- *Wasp-CLI* is the command-line interface used to communicate with the wasp node (IOTA node)

- The presentation layer is the GUI for presenting the data and interacting with the user. Bootstrap framework on top of HTML and CSS is used to develop the front end. Python flask module is used as the web framework. Contracts ABI components are JSON formatted smart contract functions and arguments used in the front-end application for smart contract function calls. Contract ABI does not reside on blockchain but is created when compiling the contract using the brownie framework [29].

4.3 System implementation

The programming language python integrates IoT, DLT interface, and starting the web server. The below sections discuss the integration and algorithm used in the developed application.

4.3.1 System Implementation diagram

Figure 4-7 shows the implementation diagram of the project. Smart contract is compiled using brownie and stored in both IOTA and Ethereum chains. Two Arduino devices are IoT devices that have two sensors each. Using Pyfirmata [30], data is read from IoT and stored to the DLT using brownie. Web3.py [32] library acts as an interface between the front end and chains. Flask is the web framework that runs the front-end code interacting with the user. A script written in bash starts all the python scripts and IOTA node.

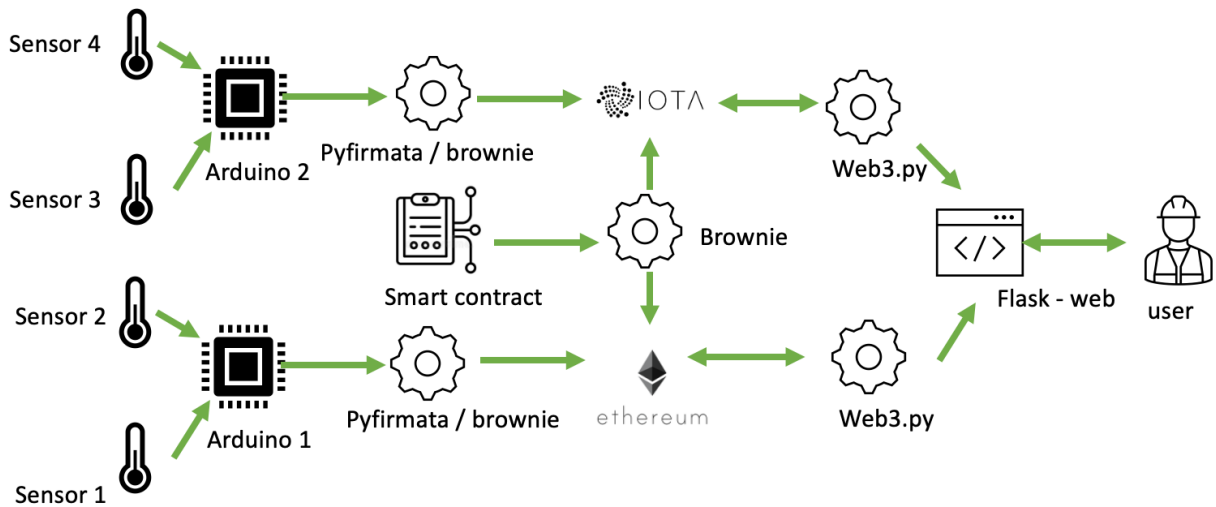


Figure 4-7 System Implementation diagram

The developed application is installed on an Ubuntu server running Ubuntu 20.04.4 LTS, as shown in Figure 4-8. The Ubuntu server has an Intel 2nd generation core i3 processor – four cores, 250 GB hard drive, and 8GB RAM. IoT devices are two Arduino UNO devices (one for the Ethereum chain and the other one for the IOTA chain). Each temperature sensor (LM35)

and potentiometer (as an analog vibration sensor was unavailable) are connected to each Arduino device. The Arduino device is attached to the server via a USB port.

```

Description:      Ubuntu 20.04.4 LTS
rejithr@rejithr-HP-RP5-Retail-System-Model-5810:~$ lsb_release -a
[No LSB modules are available.
Distributor ID:  Ubuntu
[Description:    Ubuntu 20.04.4 LTS
Release:         20.04
Codename:        focal
rejithr@rejithr-HP-RP5-Retail-System-Model-5810:~$ lsb_release -a

```

Figure 4-8 Ubuntu Release Version

4.3.2 Algorithm

The section shows the important algorithms used to implement the system. Detailed developed code for the project is presented in the APPENDIX.

4.3.2.1 Smart Contract Structure

Smart Contract can be seen as a database store procedure. **Algorithm 1** shows the pseudocode of the smart contract written in solidity Ethereum programming language. The code is deployed to both IOTA and Ethereum chains.

Algorithm 1 : Smart Contract

- 1 Declare solidity version
 - 2 Declare contract name
 - 3 Declare IoT data variable as struct with multiple variables
 - 4 Declare machinery data variable as struct with multiple variables
 - 5 Create an array of IoT objects
 - 6 Create an array of machinery objects
 - 7 Define owner of the contract as the creator of smart contract
 - 8 Function get_iot_data -> returns iot array
 - 9 Function get_machine_data -> returns machinery data array
 - 10 Function insert_iot_data (IoT data)
 - 11 Function create_machine_data (machinedata)
-

4.3.2.2 IoT Device Integration and Data Storage to DLT

Algorithm 2 shows the pseudocode to read data from IoT devices and store value in both IOTA and Ethereum chains. The Algorithm uses Brownie Framework [34]. Limited configurations are needed to prepare a smart contract function call, unlike the web3.py module.

Algorithm 2 : Device Integration and Data Storage

- 1 **Data** : Sensor data, Smart Contract, Private Key
- 2 **Result** : Store sensor data in Ethereum and IOTA Chain
- 3 **while** True // Run Indefinitely
- 4 contractInstance = getSmartContract(LatestContract, PrivateKey);
- 5 board = Arduino(PortNumber);

```

6      channel = board(ChannelNumber);
7      voltage = readData(channel);
8      sensorValue = convertVoltToValue(voltage);
9      timestamp = getTimeStamp();
10     sensorUnit = getSensorUnit ();
11     iotDataStorage = setIoTData (sensorValue, timestamp,SensorUnit);
12     waitTransactionToComplete ();
13     print(getIoTDataFromBlockchain);
14 end

```

4.3.2.3 Service Data Storage

Algorithm 3 shows the pseudocode to read data from IoT devices and store value in both IOTA and Ethereum chains. Since the Algorithm uses web3.py – data storage needs to be written explicitly.

Algorithm 3 : Store Service Data

```

1  Data      :    User Input from Web Form, Owner Address, Private Key
2  Result   :    Store service data in Ethereum and IOTA Chain
   // Wait for request method
3  if Request = 'POST'
4      formData = getFormData(UserInput);
5      nonce= getLatestChainTransectionCount();
6      storeServiceData = smartContractStoreFunction(formData, gasPrice,
   OwnerAddress, nonce );
7      signStoreServiceData = signTransection(storeServiceData, PrivateKey);
8      print(getTransactionHash(signStoreSericeData.sendToChain()));
9  end

```

4.3.2.4 Reading Data from Chain and Data presentation

Algorithm 4 shows the pseudocode to read data from Ethereum/IOTA chain and display it on the web page.

Algorithm 4 : Device Integration and Data Storage

```

1  Data      :    Smart Contract Address
2  Result   :    Display Data in the Web Page
   // Wait for request method
3  if Request = 'GET'
4      IoTData = getIoTData(LatestContract);
5      waitTransactionToComplete ();
6      IoTDataList =IoTData.append();
7      ServiceData = getServiceData(LatestContract);
8      waitTransactionToComplete ();
9      ServiceDataList = ServiceData.append();
10     sendDataToFrontEnd(IoTDataList, ServiceDataList);
11 end

```

4.3.2.5 Script for starting all necessary scripts to enable IOTA and Ethereum Integration

The script shown in **Algorithm 5** is the pseudocode of the script prepared to start python scripts and IOTA services without individually starting the scripts. The script written in bash script enables auto startup of applications and services to enable IOTA and Ethereum chain integration.

Algorithm 5 : Bash script pseudocode for starting necessary programs

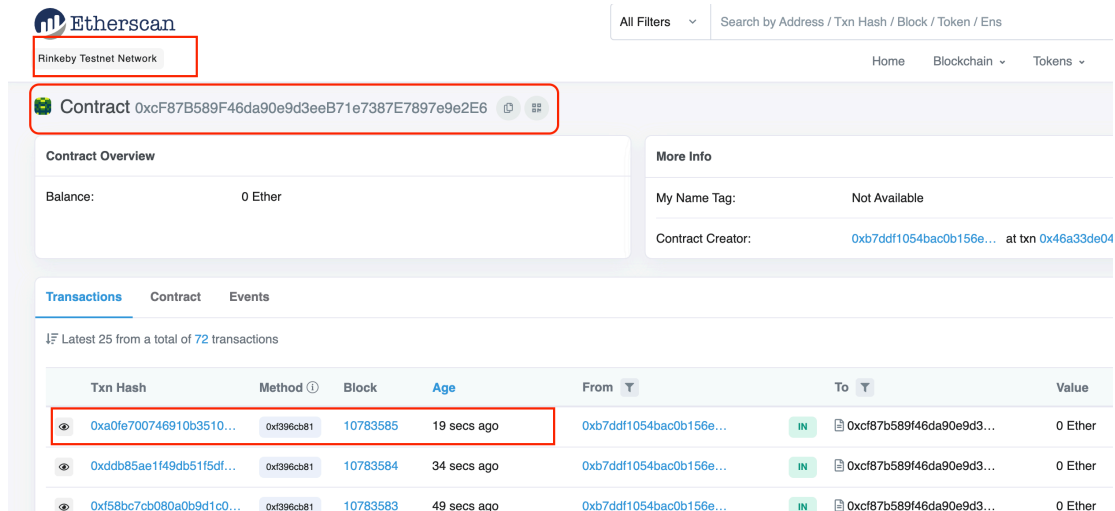
```

stop IOTA node if running
stop the IOTA command-line tool if running
stop all running python scripts if running
change to brownie folder
start script read_write_IOT_data_for_ethereum —select network as ETHEREUM
testnet
start script read_ethereum_data_for_front_end
change IOTA_WASP_Node_folder
start IOTA_WASP_Node select configuration as config.json
start IOTA_Etherum chain
request funds from IOTA Faucet
deposit funds to chain for transaction
start script read_write_IOT_data_for_IOTA_Chain —select network as local IOTA
network
start script read_IOTA_data_for_front_end

```

5 Results and Discussion

A smart contract is the backbone of the application. Therefore, it is important to confirm that smart contract is deployed correctly in the Ethereum test net. Figure 5-2 shows the screenshot of smart contract “0xcF87B589F46da90e9d3eeB71e7387E7897e9e2E6” deployed in Rinkeby Test network. The webpage etherscan.io is used to confirm the smart contract. The address <https://rinkeby.etherscan.io/address/0xcF87B589F46da90e9d3eeB71e7387E7897e9e2E6> confirms the validity of the smart contract.

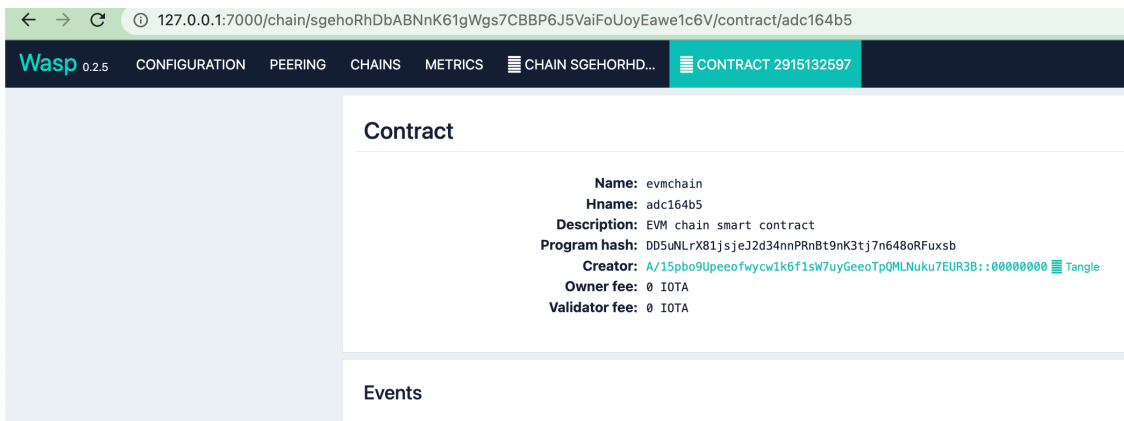


The screenshot shows the Etherscan interface for the Rinkeby Testnet Network. The main heading is "Contract 0xcF87B589F46da90e9d3eeB71e7387E7897e9e2E6". Below this, the "Contract Overview" section shows a balance of 0 Ether. The "More Info" section indicates that the name tag is not available and the contract creator is 0xb7ddf1054bac0b156e... at tx 0x46a33de04. The "Transactions" section shows a list of transactions, with the first one highlighted in red. The transaction details are as follows:

Txn Hash	Method	Block	Age	From	To	Value
0xa0fe700746910b3510...	0xf396cb81	10783585	19 secs ago	0xb7ddf1054bac0b156e...	IN 0xcF87B589F46da90e9d3...	0 Ether
0xddb85ae1f49db51f5df...	0xf396cb81	10783584	34 secs ago	0xb7ddf1054bac0b156e...	IN 0xcF87B589F46da90e9d3...	0 Ether
0xf58bc7cb080a0b9d1c0...	0xf396cb81	10783583	49 secs ago	0xb7ddf1054bac0b156e...	IN 0xcF87B589F46da90e9d3...	0 Ether

Figure 5-2 Smart Contract in Rinkeby Test net

In the case of IOTA, the contract can only be viewed in the localhost address. The smart contract address is “adc164b5“. Figure 5-3 shows the screenshot of the contract in the wasp chain.



The screenshot shows the Wasp configuration page for a smart contract. The URL is 127.0.0.1:7000/chain/sgehoRhDbABNnK61gWgs7CBBP6J5VaiFoUoyEawe1c6V/contract/adc164b5. The page title is "Wasp 0.2.5" and the navigation menu includes CONFIGURATION, PEERING, CHAINS, METRICS, CHAIN SGEHORHD..., and CONTRACT 2915132597. The main content area is titled "Contract" and displays the following details:

- Name: evmchain
- Hname: adc164b5
- Description: EVM chain smart contract
- Program hash: DD5uMLrX81jsjeJ2d34nnPRnBt9nK3tj7n648oRFuxsb
- Creator: A/15pbo9Upeefwycw1k6f1sW7uyGeeoTpQMLNuku7EUR3B: :00000000 Tangle
- Owner fee: 0 IOTA
- Validator fee: 0 IOTA

Below the contract details, there is an "Events" section which is currently empty.

Figure 5-3 Smart Contract view in Wasp configuration page

IoT integration test with Pyfirmata module is tested using simple print statements to confirm that the voltage is read correctly from Arduino devices. An example of such a printout is shown in Figure 5-4. If there is a communication error with the Arduino device, it will be shown in the bash startup script.

```

IOT_Web3.create_iot_data confirmed Block: 10791722 Gas used: 119257 (90.91%)
[45, 45]
[21, 21]
(['22-04-2022', 'tomorrow', 'Not healthy', '1', 'Technician 01'])
192.168.10.120 - - [04/Jun/2022 06:24:00] "GET / HTTP/1.1" 200 -
192.168.10.120 - - [04/Jun/2022 06:24:00] "GET /main.css HTTP/1.1" 404 -
192.168.10.120 - - [04/Jun/2022 06:24:00] "GET /main.js HTTP/1.1" 404 -
192.168.10.120 - - [04/Jun/2022 06:24:00] "GET /static/Chart.min.js HTTP/1.1" 404 -
171
    
```

Figure 5-4 Screenshot of IoT Sensor data

The script stores the data in both Ethereum and IOTA chains. Etherscan.io website is used to confirm the Rinkeby Ethereum test network transaction. The same concept is used to confirm the storage of service data. We can trace the data in the Ethereum test network using the transaction hash printed in the python script.

Figure 5-5 shows the data of successful storage of sensor data. The web address <https://rinkeby.etherscan.io/tx/0xf58bc7cb080a0b9d1c09d5b5ba4fe642df9e7707c5816d9a9fbfa0956bbd1c5> shows the data storage in the test net.

The screenshot shows the transaction details on Etherscan.io for the Rinkeby network. The transaction is successful. The input data field contains the sensor data: '0x Ë Å TEMP_SENSOR_001 06/02/2022,19:59:13 degC'. Other details include a transaction hash, block number (10783583), timestamp (17 mins ago), and gas usage (119,257 / 131,182).

Figure 5-5 Etherscan.io shows the temperature sensor data.

Figure 5-6 shows the data storage in the case of the IOTA chain. Since IOTA 2.0 is still in the testing phase, there are no websites like etherscan.io. The local host address hosted by the wasp chain confirms data storage.

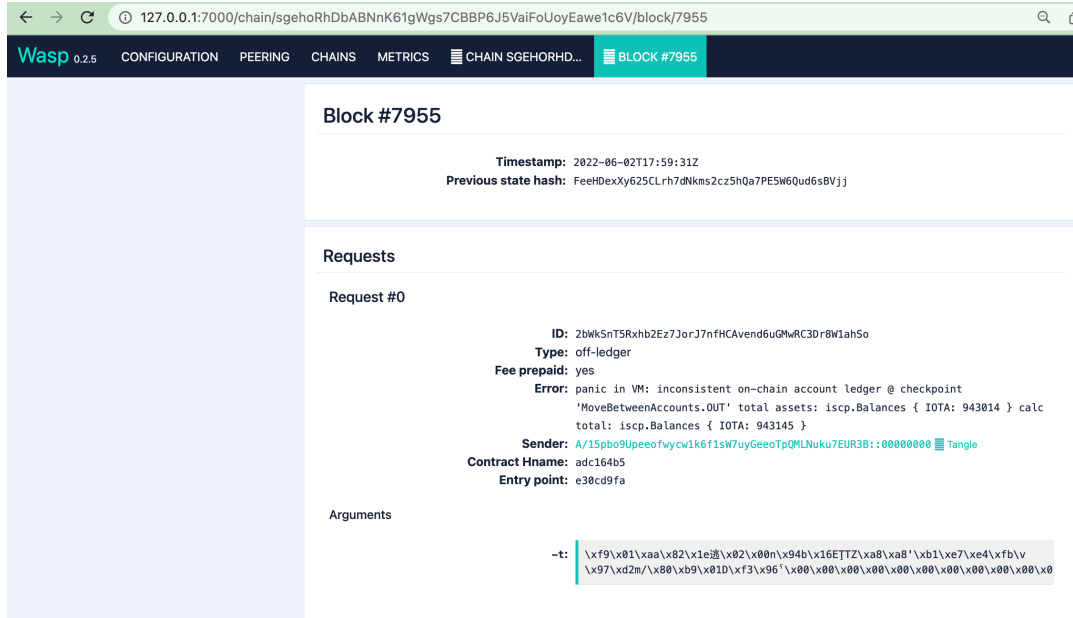


Figure 5-6 Data Storage in IOTA Chain

Historical data for both IOTA and Ethereum chains are stored as a text file to enable users to access data without browsing through etherscan or local wasp sites. Figure 5-7 shows a screenshot of data storage for the Ethereum network. Historical data for the smart contract is stored in the file. The user can easily read the data.

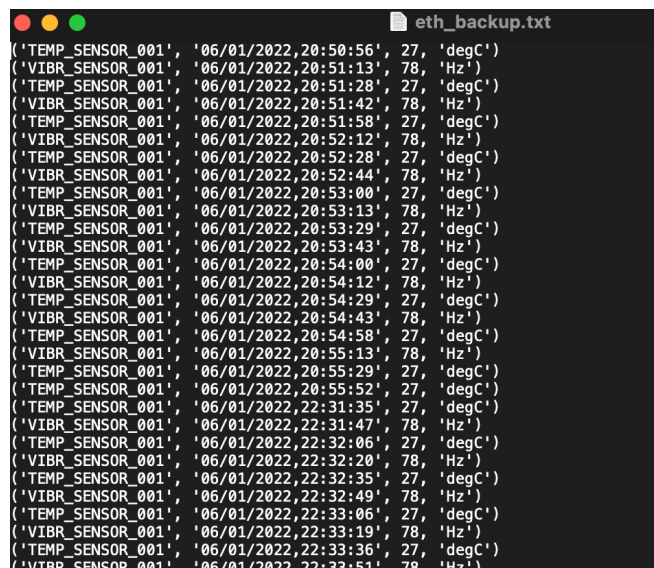


Figure 5-7 Sensor data file

Figure 5-8 shows the developed frontend for presenting the latest data, the last four sensor data stored, and the latest service status. The front end is for maintenance data stored in the Ethereum test network called Rinkeby. The latest data contains the current value of both temperature and vibration sensor, machine health status entered from the latest service status, and the possibility to enter new service data. The latest raw sensor data column contains the sensor tag name, the timestamp when the data was stored, and the sensor value with its unit. The service status block contains the data entered by a service technician. The status contains the serial communication port address, last and next service, machine health status, company name, and service technician's name.

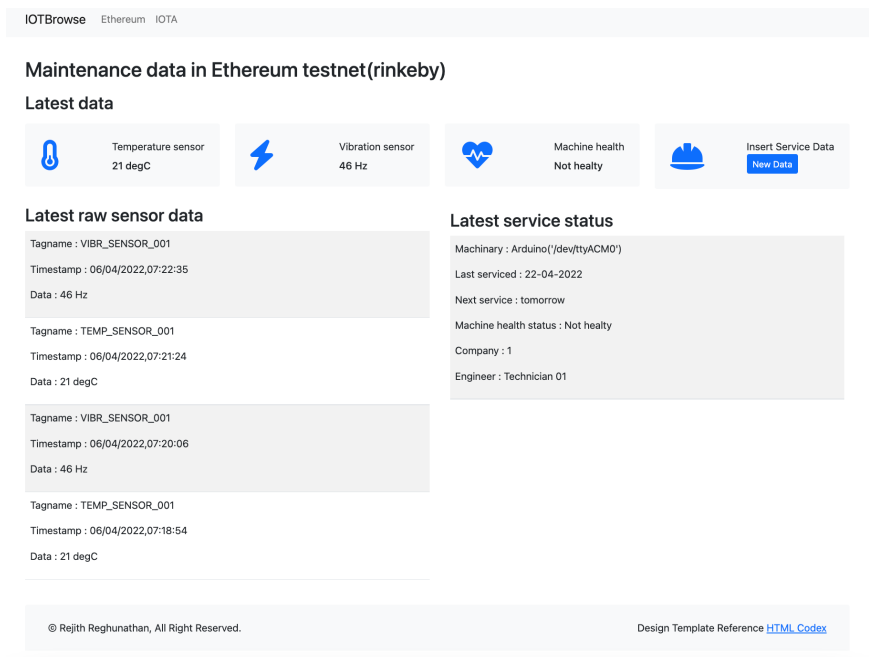


Figure 5-8 Front End for Ethereum Chain

Figure 5-9 shows the data stored in the IOTA chain. The data presentation is like the front end of the Ethereum chain. Data is collected from a separate Arduino device connected to the second USB port.

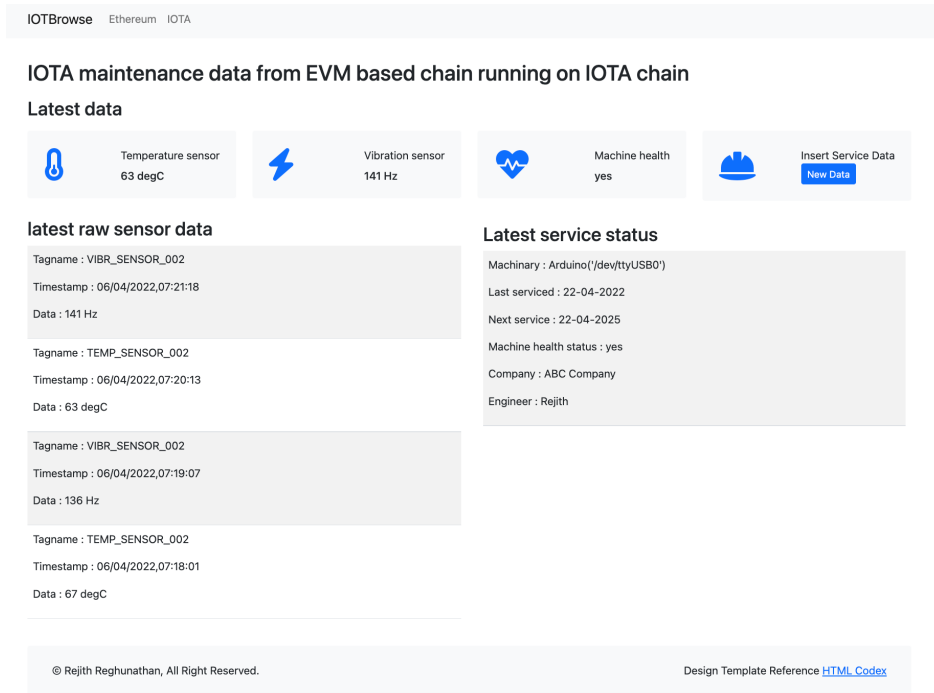


Figure 5-9 Front End for IOTA Chain

Figure 5-10 shows the pop-up form where the service technician enters the service data. The pop-up form allows the service technician to enter when the service is carried out, when the next service due date, the health of machinery based on analysis from the collected historical data, the service company name, and the service technician's name. The data is stored in the distributed ledgers.

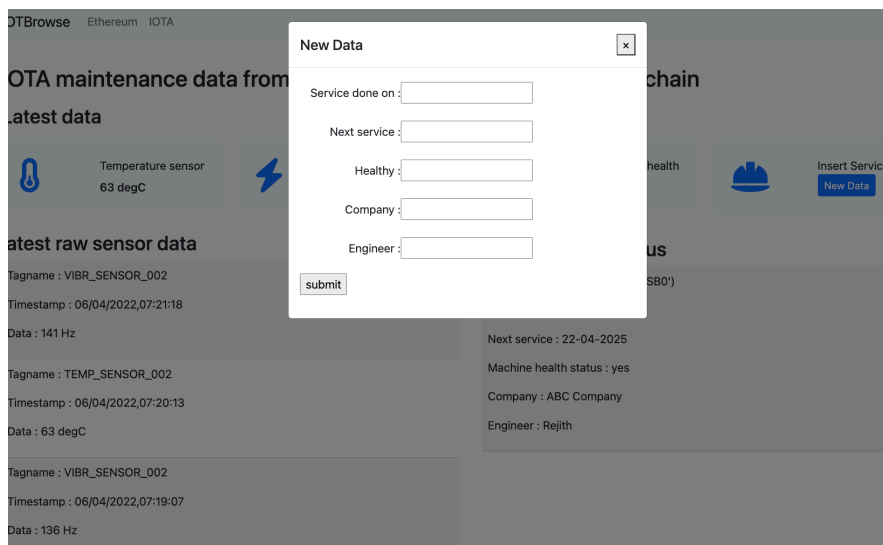


Figure 5-10 Form for entering service data

5.2 Performance Evaluation

A performance evaluation of the developed system is carried out. CPU and memory usage of the server is performed. The analysis is conducted for both Ethereum and IOTA chains together. Gas cost for storing IoT data is also performed during the performance evaluation phase. Gas cost is the fee the sender is ready to pay. One unit of gas is comparable to the execution of a computational step [29]. Gas cost analysis is done only for Ethereum chain as the IOTA chain is still in the development phase with zero cost.

5.2.1 CPU and Memory Usage

NMON or Nigel's performance monitor application is used to check CPU and Memory usage of the developed application. NMON was originally developed for IBM AIX performance monitoring and analysis, is now an open-source [35]. The application is started using the "nmon" command in the terminal. Figure 5-11 and Figure 5-12 show current and long-term CPU usage, respectively. It is observed that the CPU usage is less than 30% in both long-term and short-term cases for both user and system. However, since the processor is Intel 2nd generation core i3, a powerful retail processor process or – the usage of less than 30% of total usage is on a high side, which means that less powerful processors such as Raspberry-Pi-based architecture may struggle to handle the load. Extensive testing is necessary to confirm and verify CPU requirements.

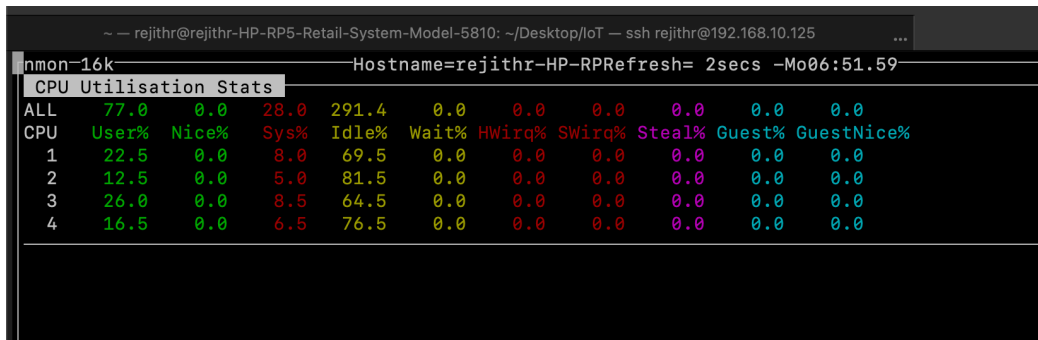


Figure 5-11 CPU Utilization Stats

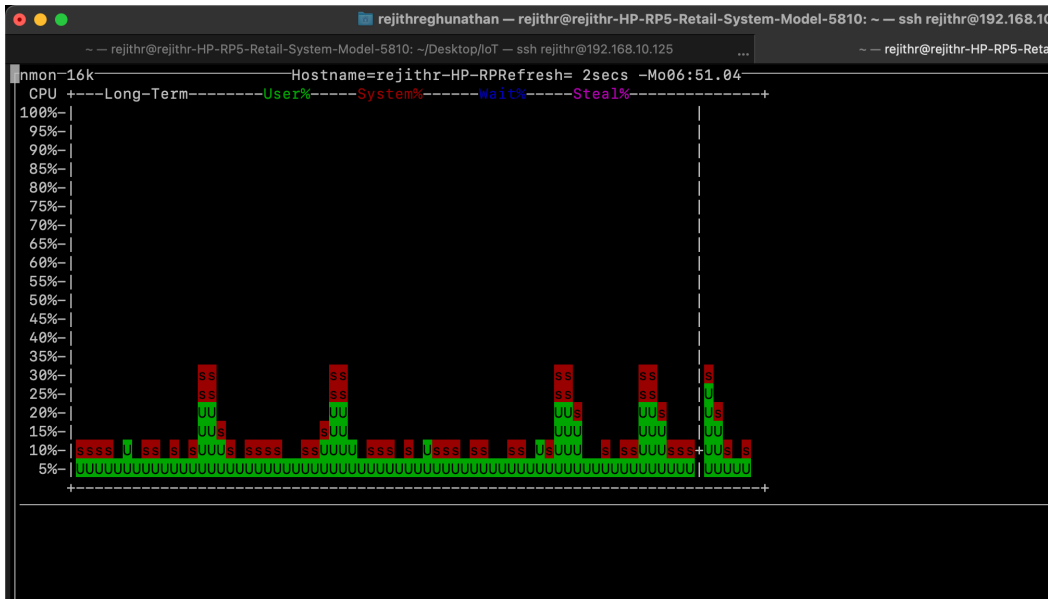


Figure 5-12 CPU Utilization Long Term

Memory usage is around 60% for the server, and only 40% is free, as shown in Figure 5-13. Around 3 GB was free of available 8GB memory. The usage is on the high side. Extensive testing is recommended to confirm and verify memory usage.

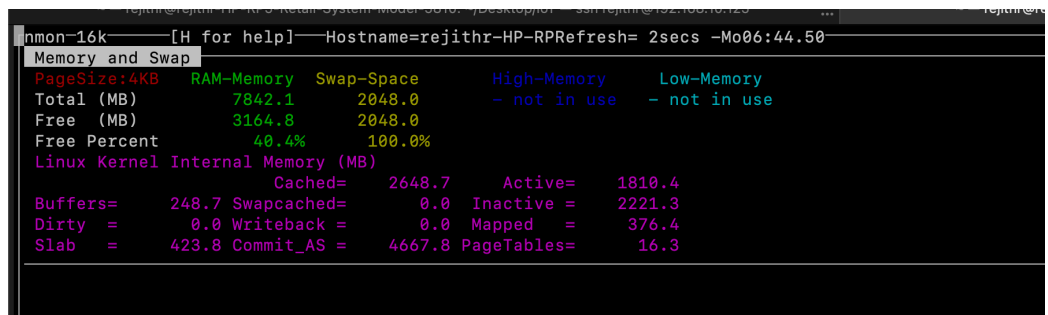


Figure 5-13 Memory Usage

5.2.2 Gas Cost

Gas cost analysis is conducted similar to the article “Blockchain for the Internet of Vehicles: A Decentralized IoT Solution for Vehicles Communication Using Ethereum” [36]. Ethereum test net – Rinkeby is used to deploy Ethereum smart contract, and the IOTA contract is deployed in the wasp node running locally.

The following are the values as of 4th June 2022, 05:58 UTC.

- 1 ETH is \$1767
- The average gas price is 35 GWEI
- 1 GWEI (0.000000001 ETH)
- Price in USD = gas required * gas price in gwei * 0.000000001* ETH to USD rate

Table 5-1 shows the execution cost of various functions in the deployed application. It is observed that the cost of deployment and execution is expensive during the analysis.

Table 5-1 Execution Cost of various functions

Function	Transaction Fees	Price (Approximate)
Deploy Contract	0.027 ETHER	\$49 (One Time / Rinkeby Network)
Store IoT Data (per call)	0.000178 ETHER	\$0,32
Store Service Data (per call)	0.000178 ETHER	\$0,32
Read IoT Data (per call)	0	0
Read Service Data (per call)	0	0

5.2.3 IOTA Vs. Blockchain

Ethereum smart contract is fully developed and supported globally. The challenge with Ethereum is its high gas cost for each transaction and delay in block creation. The arrival of Ethereum 2.0 and the proof of stake mechanism is expected to solve both limitations. On the other hand, IOTA is still in a continuous development phase. IOTA is a scalable decentralized solution that does not require additional transaction fees associated with mining, unlike the traditional BC technology [37]. The system development was entirely relying on the documentation provided by IOTA Organization. The documentation does not cover all scenarios for typical smart contract development. However, there is excellent support provided by the IOTA community.

5.3 Discussion

Distributed Ledger Technology concepts were extensively surveyed in this thesis before application development. Followed by identifying the system specifications, requirements, and use cases. Case diagrams are further developed into sequence diagrams, and system architecture is prepared. Algorithms are based on the DLT technology research and sequence diagrams. Testing and performance analysis revealed that the application is working as expected. However, the computation resource consumption is higher.

The solution presented in the thesis reflects how distributed technology can be used to solve the challenges with IoT data handling and privacy. The system is transparent; this means that both the owners and service companies can read the data from the IoT device and the service history of the machinery. The data is decentralized and immutable. Therefore, the ledger can be presented to any future owner of the equipment and authorities who needs to approve the machinery.

5 Results and Discussion

The solution is not considering a role-based access control. Any user can now write service history into the distributed ledgers. Now, any IoT devices can be connected to the chain or even swapped between ports – the program cannot detect if the connected IoT is from the correct machinery and serial number.

The application is mostly developed in the python programming language. Limited research materials are available to integrate blockchain and IOTA via Python or python modules. The application development was quite time-consuming, especially with IOTA, where the wasp node ran into errors. Several trials and errors were carried out during the application development to integrate the blockchain back end to the front end. Java script programming language is found to be highly supported by decentralized communities. JavaScript-based languages such as react are more supported for front-end integration with blockchain. The suggestion is that similar applications be developed in the Javascript programming language.

The developed application needs to be further optimized to handle more data and devices and simultaneously reduce the gas price. With the development of Ethereum 2.0 soon, it is expected that the gas and transaction fees will go down as the consensus method will be proof of stake instead of existing proof of work. There are also concerns about increasing power consumption regard to the mining process.

IOTA chain has a feeless structure and is effectively designed to handle many IoT devices. IOTA resolved the major blockchain issues, transaction fees, scaling limitations, and centralization [26]. It is worth mentioning that IOTA – Ethereum integration is still in the developing stage. Due to limited knowledge and support available globally, the project development met several changes due to this phase. However, the support of the IOTA developer network is highly appreciated during the product development and challenges faced during the development stage.

6 Conclusion

The increased data generated from IoT devices introduces the challenge of privacy. Integrating IoT and DLT solves the privacy issue. The solution acquires data from IoT devices, which contains maintenance information from machinery. The necessary programs were developed in python and solidity programming language. An Edge Box installed between IoT devices and the blockchains process the data from IoT devices. The Box running Ubuntu operating system will handle interactions with the decentralized ledger. The stakeholders and service companies will use Web-based client applications to view historical operational data recorded on the Blockchain. Maintenance companies can monitor operational data and can decide if service is necessary. If a service on a piece of equipment is carried out, the service will be recorded on the Blockchain using a Web-based client application. The performance analysis showed that there is a requirement for computational resources. Cost analysis proved to be expensive. The release of Ethereum 2.0 and IOTA 2.0 is expected to improve results.

The use of communication protocols such as MQTT and Modbus for data collection from IoT devices and integration into distributed ledgers shall be considered in future studies. The effectiveness of handling large data and stress test in blockchain can be further tested and evaluated. The authentication and authorization aspect of blockchain programming is an area which can be studied further as well. Lastly, future studies can explore the use of collected data to develop machine learning models to recommend or delay periodic maintenance. Extending the developed application by including a data analysis algorithm makes the quite usable in the maintenance of machinery.

References

- [1] Y. Ismail, *Internet of Things (IoT) for automated and smart applications*. 2019. Accessed: Jan. 25, 2022. [Online]. Available: <https://doi.org/10.5772/intechopen.77404>
- [2] Mohammad, “IoT connections market update—May 2022,” *State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally*. <https://iot-analytics.com/number-connected-iot-devices/> (accessed Jul. 11, 2022).
- [3] V. K. Calastray Ramesh, “Storing IOT Data Securely in a Private Ethereum Blockchain”, doi: 10.34917/15778410.
- [4] A. Ahmad, “Integration of IoT devices via a blockchain-based decentralized application,” 2017, doi: 10.18419/OPUS-9466.
- [5] “What is IoT with blockchain? - IBM Blockchain.” <https://www.ibm.com/seen/topics/blockchain-iot> (accessed Jan. 23, 2022).
- [6] “An Introduction to IOTA.” <https://wiki.iota.org/learn/about-iota/an-introduction-to-iota> (accessed May 15, 2022).
- [7] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, “Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios,” *IEEE Access*, vol. 8, pp. 23022–23040, 2020, doi: 10.1109/ACCESS.2020.2970118.
- [8] P. Lea, *IOT and edge computing for architects: implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security*, Second edition. Birmingham: Packt, 2020.
- [9] A. M. Rahmani, S. Bayramov, and B. Kiani Kalejahi, “Internet of Things Applications: Opportunities and Threats,” *Wireless Pers Commun*, vol. 122, no. 1, pp. 451–476, Jan. 2022, doi: 10.1007/s11277-021-08907-0.
- [10] M. Alshaiqli, T. Elfouly, O. Elharrouss, A. Mohamed, and N. Ottakath, “Evolution of Internet of Things From Blockchain to IOTA: A Survey,” *IEEE Access*, vol. 10, pp. 844–866, 2022, doi: 10.1109/ACCESS.2021.3138353.
- [11] S. T. March and G. D. Scudder, “Predictive maintenance: strategic use of IT in manufacturing organizations,” *Inf Syst Front*, vol. 21, no. 2, pp. 327–341, Apr. 2019, doi: 10.1007/s10796-017-9749-z.
- [12] S. Ayvaz and K. Alpay, “Predictive maintenance system for production lines in manufacturing: A machine learning approach using IoT data in real-time,” *Expert Systems with Applications*, vol. 173, p. 114598, Jul. 2021, doi: 10.1016/j.eswa.2021.114598.
- [13] C. Nartey *et al.*, “On Blockchain and IoT Integration Platforms: Current Implementation Challenges and Future Perspectives,” *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–25, Apr. 2021, doi: 10.1155/2021/6672482.
- [14] “Distributed-Ledger-Technology (DLT).” <https://iota-beginners-guide.com/dlt/> (accessed Feb. 07, 2022).
- [15] “Blockchain Technology and Its Applications: Case Studies,” *JSMS*, Mar. 2020, doi: 10.33168/JSMS.2020.0106.

Nomenclature

- [16] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of Blockchains in the Internet of Things: A Comprehensive Survey," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 2, pp. 1676–1717, 2019, doi: 10.1109/COMST.2018.2886932.
- [17] M. Deer, "What is a directed acyclic graph in cryptocurrency? How does DAG work?," Nov. 07, 2021. <https://cointelegraph.com/explained/what-is-a-directed-acyclic-graph-in-cryptocurrency-how-does-dag-work> (accessed Mar. 05, 2022).
- [18] M. Bhandary, M. Parmar, and D. Ambawade, "Securing Logs of a System - An IoTA Tangle Use Case," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2020, pp. 697–702. doi: 10.1109/ICESC48915.2020.9155563.
- [19] Phani, "IOTA Proof of Work: Remote Vs Local explained." <https://medium.com/bytes-io/iota-proof-of-work-remote-vs-local-explained-1cbd89392a79>
- [20] E. Drašutis, "IOTA Smart Contracts." IOTA Foundation, Nov. 10, 2021. [Online]. Available: https://files.iota.org/papers/ISC_WP_Nov_10_2021.pdf
- [21] "Smart Contracts," *Smart Contracts*, May 15, 2022. https://wiki.iota.org/smart-contracts/guide/core_concepts/smart-contracts
- [22] L. M. C. Augusto, "An application of blockchain smart contracts and IoT in logistics," Universidade Nova de Lisboa, Lisboa, 2019. [Online]. Available: https://run.unl.pt/bitstream/10362/75035/1/Augusto_2019.pdf
- [23] Dr. Achim Klein, "Exploring IOTA 2.0 Smart Contracts in a Private Network: Developing a Prediction Market," Sep. 16, 2021. <https://medium.com/5lnodes/exploring-iota-2-0-smart-contracts-in-a-private-network-developing-a-prediction-market-c2d81988f75e> (accessed May 16, 2021).
- [24] M. A. A. Mamun, "Real-time Integration of IoT Sensor and IOTA Tangle for Securing IoT Infrastructure," University of Manitoba, 2022. [Online]. Available: <http://hdl.handle.net/1993/36179>
- [25] H. Guo and X. Yu, "A survey on blockchain technology and its security," *Blockchain: Research and Applications*, vol. 3, no. 2, p. 100067, Jun. 2022, doi: 10.1016/j.bcra.2022.100067.
- [26] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, "Blockchain and IoT Integration: A Systematic Survey," *Sensors*, vol. 18, no. 8, p. 2575, Aug. 2018, doi: 10.3390/s18082575.
- [27] V. K. Calastray Ramesh, "Storing IOT Data Securely in a Private Ethereum Blockchain", doi: 10.34917/15778410.
- [28] M. Alshakhli, T. Elfouly, O. Elharrouss, A. Mohamed, and N. Ottakath, "Evolution of Internet of Things From Blockchain to IOTA: A Survey," *IEEE Access*, vol. 10, pp. 844–866, 2022, doi: 10.1109/ACCESS.2021.3138353.
- [29] A. Ahmad, "Integration of IoT devices via a blockchain-based decentralized application," 2017, doi: 10.18419/OPUS-9466.
- [30] firmata, "Firmata Protocol Documentation." Accessed: May 31, 2022. [Online]. Available: <https://github.com/firmata/protocol>

- [31] IVANONTECH, “Infura Explained – What is Infura?,” Jun. 16, 2021. <https://academy.moralis.io/blog/infura-explained-what-is-infura>
- [32] “Web3.py.” Accessed: May 31, 2022. [Online]. Available: <https://web3py.readthedocs.io/en/stable/index.html>
- [33] “Web3.js.” Accessed: May 31, 2022. [Online]. Available: <https://web3js.readthedocs.io/en/v1.7.3/>
- [34] “Brownie.” Accessed: May 31, 2022. [Online]. Available: <https://eth-brownie.readthedocs.io/en/stable/>
- [35] “About Nmon Performance monitor for Splunk.” Accessed: Jun. 11, 2022. [Online]. Available: <https://nmon-for-splunk.readthedocs.io/en/latest/about.html>
- [36] R. Jabbar, M. Kharbeche, K. Al-Khalifa, M. Krichen, and K. Barkaoui, “Blockchain for the Internet of Vehicles: A Decentralized IoT Solution for Vehicles Communication Using Ethereum,” *Sensors*, vol. 20, no. 14, p. 3928, Jul. 2020, doi: 10.3390/s20143928.
- [37] E. Exposito *et al.*, “Tangle The Blockchain: Toward IOTA and Blockchain integration for IoT Environment,” Sehore, India, Dec. 2019. [Online]. Available: <https://hal-univ-pau.archives-ouvertes.fr/hal-02957070>

Appendices

Appendix A Thesis Description

Appendix B Solidity Smart Contract Code

Appendix C Python - Brownie Code for interacting with IoT and Ethereum Chain (IoT-Ethereum interaction code)

Appendix D Python Code for Ethereum Front End Application

Appendix E Python - Brownie Code for interacting with IoT and IOTA Chain (IoT-IOTA interaction code)

Appendix F Python Code for IOTA Front End Application

Appendix G Bash Script for Starting all applications

Appendix H Complete Code download link