Decentralized IoT Gateway Platform

A PROJECT REPORT

Submitted by

FABIAN FERNO (311119205014)
GIRISH VJ (311119205501)
JOHN BRITE (311119205024)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

LOYOLA - ICAM
COLLEGE OF ENGINEERING AND
TECHNOLOGY CHENNAI - 600034

ANNA UNIVERSITY: CHENNAI - 600025

AUGUST 2021

ANNA UNIVERSITY: CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this project **"DECENTRALIZED IOT GATEWAY PLATFORM"** is the bonafide work of **FABIAN FERNO (311119205014), GIRISH VJ (311119205501)** and **JOHN BRITE (311119205024)** who carried out the project work under my supervision.

SIGNATURE
Dr Sherril Sophie Maria Vincent,

B.E., M.E., SUPERVISOR

Assistant Professor
Information Technology
Loyola-ICAM College of
Engineering and Technology
Loyola Campus, Nungambakkam,
Chennai-34

SIGNATURE
Dr. R Juliana, B.E, M.E, Ph.D

HEAD OF DEPARTMENT

Professor
Information Technology
Loyola-ICAM College of
Engineering and Technology
Loyola Campus, Nungambakkam,
Chennai-34

Submitted for the Project Viva Voce examination held on 21/06/22.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First of all, we are grateful to God for granting us this opportunity and capability to proceed successfully. Working on this project has been a rewarding experience.

We would like to express our sincere thanks and gratitude to our Director, **Rev. Dr Maria Wenisch SJ, Ph.D.**, the Dean of Students **Dr Caleb Chanthi Raj**, **Ph.D.,** and the Dean of Engineering Education, **Mr Nicholas Juhel, M.E.,** for their constant support and encouragement; We thank them for making the right resources available at the right time.

We would like to extend our gratitude to our Principal, **Dr L Antony Michael Raj, M.E., Ph.D.,** for his motivating, constructive criticism and valuable guidance during the course of the project. We would like to express our sincere thanks to the Head of the Department **Dr R Juliana, B.E., M.E., Ph.D.**, for her comments and suggestions.

We also would like to thank the project coordinators **Ms Shobana G, M.Tech.**, and **Ms V Kavitha, B.Tech. M.Tech.**, the faculty and technical staff of our department for their critical advice and guidance which was helpful in completion of the project.

We extend our sincere gratitude to our project guide **Ms Sherril Sophie Maria Vincent, B.E., M.E.,** for providing valuable insights and resources leading to the successful completion of our project. We would also like to thank the faculty members of our department for their critical advice and guidance which was indispensable for our work. Last but not the least, we place a deep sense of gratitude to our friends and families who have been a constant source of inspiration during the preparation of this project.

ABSTRACT

IoT is facing identity, security and interoperability problems. Current systems rely on a centralized client-server model that will soon be unsatisfactory due to the rapid increase in the number of devices connected to the Internet. With the advent of more than 30 billion connected devices making up the “Internet of Things”. Networks are highly prone to trojan devices that piggyback into the so called smart devices and steal sensitive data and hack into our lives. This is more cynical when it comes to industry automations, where high volume sensor data can easily get into the wrong hands. Tampered devices with corrupt firmwares, unknown manufacturers’ metadata and the lack of ownership of these devices are problems that are often overlooked. These problems can be solved by making use of the blockchain technology to add a validation layer hence providing an IAM for the same. We can effectively solve the data, communication and security pitfalls associated with IoT by effectively utilizing the blockchain technologies. Hash validation on every connection that verifies the firmware, metadata and the device owner. Also the data relayed by these devices can be stored on a decentralized storage solution called the IPFS, and a socket connection to listen to the data changes can eventually be used to add a communication layer for these devices thereby the rIOT platform ~ a decentralized IoT gateway platform helps in solving the data, security and communications downfalls of edges devices that make up the “Internet of Things” through a secure ledger of things.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.No.	ACRONYM	ABBREVIATION
1	DAPP	DECENTRALIZED APPLICATION
2	IOT	INTERNET OF THINGS
3	IAM	IDENTITY ACCESS MANAGEMENT
4	IPFS	INTER - PLANETARY FILE SYSTEM
5	ERC	ETHEREUM REQUEST FOR COMMENT
6	ETH	ETHEREUM
7	SDK	SOFTWARE DEVELOPMENT KIT
8	P2P	PEER TO PEER
9	SHA	SECURE HASH ALGORITHM
10	IP	INTERNET PROTOCOL
11	NFT	NON FUNGIBLE TOKEN
12	TCP	TRANSMISSION CONTROL PROTOCOL
13	PaaS	PLATFORM AS A SERVICE
14	DFD	DATA FLOW DIAGRAM
15	ESP	ESPRESSIF SYSTEMS
16	WIFI	WIRELESS FIDELITY

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The Internet of Things and its growing network of connected devices are facing problems with security, data and communication prospects. A system that validates these devices and also thereby provides means to enable a smart network that works based on decentralized technologies can effectively solve these issues of authenticating these devices and allowing decentralized data storage and communication can be implemented using the blockchain ledger technology. Each device is minted (registered) into the rIOT platform by computing the device's rIOT hash (a merkle hash containing the hashes of the device owner address, device metadata, device firmware and its identification number). This rIOT hash is validated on every connection request to the platform using the smart contract methods that can be called on the blockchain to verify and hence allow connections to the data layer and the communication layers. This can very conveniently stop unknown / tampered devices from connecting to any smart network of IoT devices). The platform can be implemented in smart homes, smart industry facilities, connected metropolitan infrastructures and anywhere where IoT has a growing use-case. The rIOT platform gives an dApp intuitive panel to manage the connected devices and to register new devices.

1.1 INTRODUCTION TO THE DOMAIN

- **INTERNET OF THINGS [IoT]**

The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. The definition of the Internet of Things has evolved due to the convergence of multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems. Traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), and others all contribute to enabling the IoT. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the "monitoring systems", covering devices and appliances that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and smart speakers.

- **WEB3 ~ BLOCKCHAIN**

Web3 is the name some technologists have given to the idea of a new kind of internet service that is built using decentralized blockchains — the shared ledger systems used by cryptocurrencies like Bitcoin and Ether. It’s about the permissionless and dynamic characteristics of the internet. NFTs, crypto tokens and crypto-currencies are the recent hype wagon with a lot of these technologies providing utility to what they promise - essentially - data, security and decentralization of the same.

1.2 OVERVIEW OF THE PROJECT

The rIOT platform is blockchain enabled decentralized IoT gateway that aims to utilize the blockchain technology to register and validate the legitimacy of the connected devices in a smart infrastructure also providing means to store data on a decentralized storage medium that can solve the data, security and the communication problems of “Internet of Things”. Devices connected to the platform are verified based on their firmware, owner details, manufacturer metadata and a device identifier. This merkle hash is used to call a smart contract on a blockchain to check if the device was already minted (registered) on the platform. The project consists of three layers - a front end dApp, a device firmware and a smart contract that is deployed on the blockchain network, “Polygon”. The decentralized data storage is provided by IPFS.

1.3 PURPOSE OF THE PROJECT

IoT devices / edge computing devices have little to zero security where smart infrastructures can be tampered with if left un-audited. This can let any tampered device with its tampered firmware to allow data theft. Moreover, they can effectively control the infrastructure at ease. The data aspects of the rIOT platform can provide byzantine fault tolerance of data, i.e unlike traditional server client implementations that have a single point of failure whereas decentralized technologies can prevent that. In IoT, the volume of data is evergrowing and the such “big data” that could involve sensor data can be stored into these IPFS nodes and further be used by other connected devices to fetch the same. The dApp (front end) will also serve as a panel to view these data collected by the edge device that efficiently makes use of the rIOT platform. The authentication is the entry point for connections of these devices - meaning the devices are verified of their firmware, manufacturer details, owner details and device id based on their hash before letting them use the data layer and hence preventing tampered devices to access the infrastructure.

1.4 PROJECT PLAN

The project planning stage requires several inputs, including conceptual proposals, project schedules. The project schedule begins from identifying the problem statements to finally deploying the project on the testnet. The development of this project is not successfully done without proper planning and scheduling.

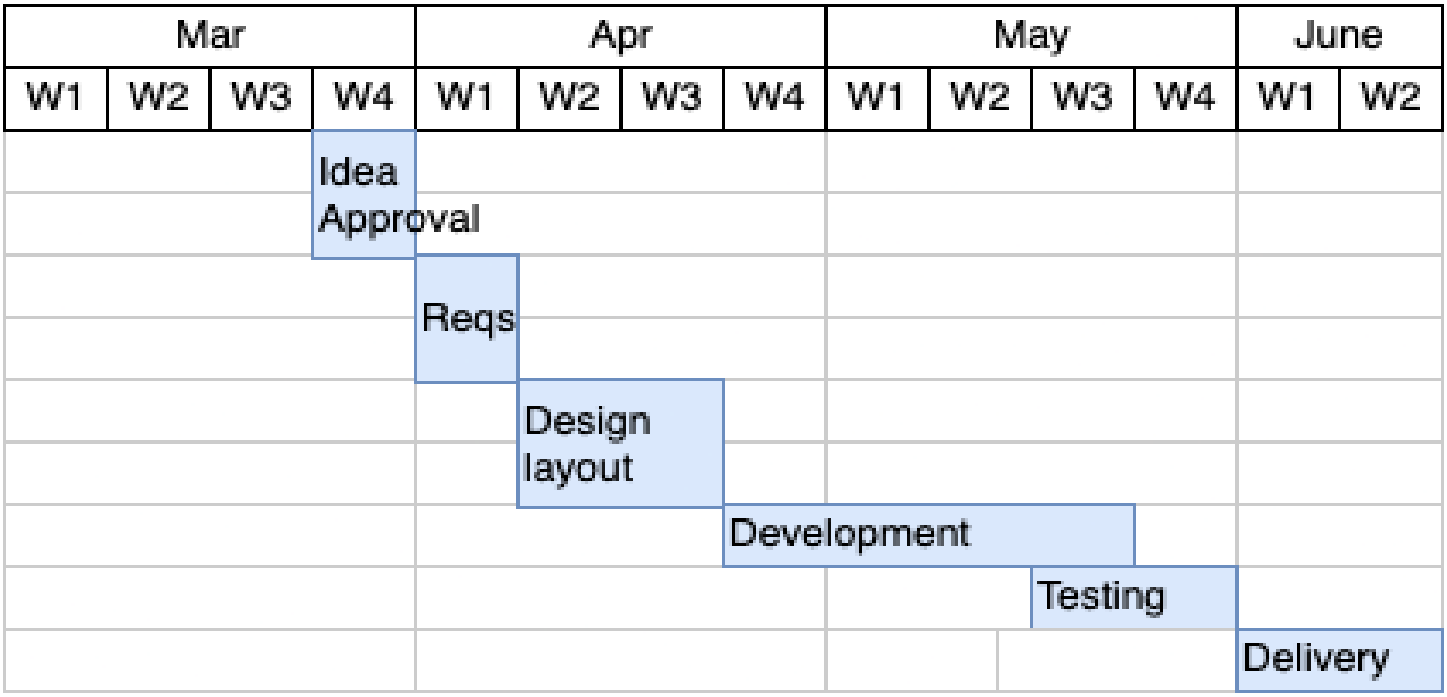

Figure 1.1 – Gantt Chart

Idea Approval: Maximum number of days for idea approval for 7 days

Requirements: Maximum number of days for idea approval for 7 days

Design Layout: Maximum number of days for idea approval for 14 days

Development: Maximum number of days for idea approval for 28 days

Testing: Maximum number of days for idea approval for 14 days

Delivery: Maximum number of days for idea approval for 7 days.

1.5 SCOPE OF THE PROJECT

The scope of this project is extensive starting from the security aspects to the data warehousing of the IoT data. It is not limited to just those aspects where the rIOT platform can be serving as one-stop PaaS. Current machine to machine data transactions occur with a static token based authentication or a session based implementation, where the rIOT platform can become a pre-emptive solution not letting the tampered devices to connect in the first place. Also the data warehousing in the platform can enable high volume data storage. The future scopes of the project include incentivizing the validation layer by allowing tokenomics and rewarding the nodes with a token for each device validated by their merkle hash. Also the data platform can have an API built to interact with the data warehouse and effectively get data for analytics or other purposes. Maintenance of such platforms are autonomous given the fact that all of the business logic is done by the smart contract deployed on the blockchain meaning that it cannot be hindered with. This also solves the problems with human errors and other problems with centralized data.

1.6 SUMMARY

Blockchain encryption makes it virtually impossible for anyone to overwrite existing data records. And using blockchain to store IoT data adds another layer of security to prevent malicious attackers from gaining access to the network. The rIOT gateway platform used the cryptographic hashes - one of the important characteristics of blockchain to leverage security in the IOT devices. Simply, a device is first minted (registered) on the blockchain similar to the NFTs on their smart contract. The mint occurs by creating the rIOT hash based on firmware, metadata, owner address and the

device identifier. The next time the device tries to authenticate using the rIOT platform, this rIOT hash is compared with the on-chain data and allows further access to the IPFS API to transact the sensor data. The admin can use the front end dApp to mint, view devices and look at the sensor data collected by the rIOT platform. This is done completely decentralized from end to end, meaning there's no single point of failure and hence byzantine fault tolerance is achieved.

CHAPTER 2

LITERATURE SURVEY

This chapter briefly discusses the literature survey done on various projects based on blockchain and IoT. Their performance, uses and limitations are analyzed.

2.1 Design and Development of Blockchain Based Decentralized IoT Platform:

Authors:Ali Dorri; Salil S. Kanhere; Raja Jurdak; Praveen Gauravaram
Published in: 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)
Date of Conference: 13-17 March 2017

Internet of Things (IoT) security and privacy remain a major challenge, mainly due to the massive scale and distributed nature of IoT networks. Blockchain-based approaches provide decentralized security and privacy, yet they involve significant energy, delay, and computational overhead that is not suitable for most resource-constrained IoT devices. In our previous work, we presented a lightweight instantiation of a BC particularly geared for use in IoT by eliminating the Proof of Work (POW) and the concept of coins. Our approach was exemplified in a smart home setting and consists of three main tiers namely: cloud storage, overlay, and smart home. In this paper we delve deeper and outline the various core components and functions of the smart home tier. Each smart home is equipped with an always online, high resource device, known as “miner” that is responsible for handling all communication within and external to the home. The miner also preserves a private and secure BC, used for controlling and auditing communications. We show that our proposed BC-based smart home framework is secure by thoroughly analysing its security with respect to the fundamental security goals of confidentiality.

2.2 A Storage architecture of Blockchain for Time-Series Data

Authors: Ziru Yu; Yefan Cai; Weibin Hong
Published in: 2019 2nd International Conference on Hot Information-Centric Networking (HotICN)
2019 Date of Conference: 13-15 December 2019

Blockchain technology is becoming more and more important in different areas. However, it is not suitable for storing time-series data because of its rapid growth. This paper starts from the traditional storage model of the blockchain and proposes a novel storage architecture intended for time-series data. Some implementation details are given as well. The system mainly includes three modules: block storage, index storage and time-series data storage. The time-series database is used as the basic storage tool so it can fully utilize the corresponding features such as elasticity, automation, efficiency, etc. By proper parameter configuration, the system can retain data within a certain time window, automatically clear expired blocks, free up storage space for dynamic growth, and provide data aggregation statistics that can be directly applied to data analysis systems. Therefore, it's more suitable for time-series data in some areas.

Cons: *Usage of RFID limits the range of implementation.ca*

2.3 Multi-Layer Aggregate Verification for IoT Blockchain

Authors: Jingze Wu;Ming-Fong Sie;Seth Austin Harding;Chien-Lung Lin;San-Tai Wang;Shih-wei Liao
Published in: IEEE Internet of Things Journal (Volume: 9)
Date of Conference: 27-30 September 2021

We design a Multi-Layer Aggregate Verification (MLAV) solution to improve supply chain management with IoT Blockchain devices. We apply MLAV to IoT Blockchain applications in Agriculture 4.0 to demonstrate the feasibility of our solutions and models. In the current Agriculture 4.0 structure, large companies have successfully applied blockchain solutions and ecosystems for tracking and tracing agricultural produce, achieving transparency, traceability, and digitalization. However, these existing blockchain solutions are not comprehensive. First, the upstream nodes they serve are all large-scale production suppliers, and smallholders are not taken into consideration. In order to solve this problem, we use a multi-layer architecture that serves three purposes: facilitating smallholders in joining the agricultural blockchain as equal-opportunity nodes, uploading of production activity data, and reducing costs (ex. Ethereum gas fee). Second, the majority of IoT blockchains adopt an ID-based signature scheme in IoT devices, which frequently has lower efficiency. In applying aggregate verification, we may effectively increase ID-based verification efficiency while processing large clusters of data transferred by IoT devices. Finally, we design a blockchain management framework using smart contracts to facilitate the financing of upstream producers.

2.4 SUMMARY

In this fast pace of life, it is difficult for people to be constantly available near IoT Devices. To overcome these problems a Security System is introduced, which can be used to monitor the Access level, Data, Hash Data followed by a Decentralized data store that helps in storing all the data in a failsafe manner. All of these values will be updated in the mobile app using a real-time cloud service which makes the system more efficient than the existing ones.

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

3.1 PROBLEM DEFINITION

Today's IoT networks and edge devices have deficient low security layers which makes them vulnerable to attacks and hackers piggybacking critical data.

A single protocol that ensures device authentication, authorization and manages the data layer is a much needed technology of the hour.

3.2 NEED ANALYSIS

- Major companies have faced multiple data breaches in their home / industry IoT smart enablements.
- As more conventions get into smart electronics in the industrial / domestic sector, data security becomes a necessity rather than a luxury.
- Smart security cameras, Industrial Sensors, Fancy household IoT enablements, & more have been victims of breaches in the past.
- Kaspersky Article - Most of the attacks used the telnet protocol to access IoT devices, and researchers recorded over 872 million (58%) of the total using this protocol.
- Forbes - Annual spending on IoT security measures will increase to $631 million in 2021.

3.2.1 BULL DIAGRAM

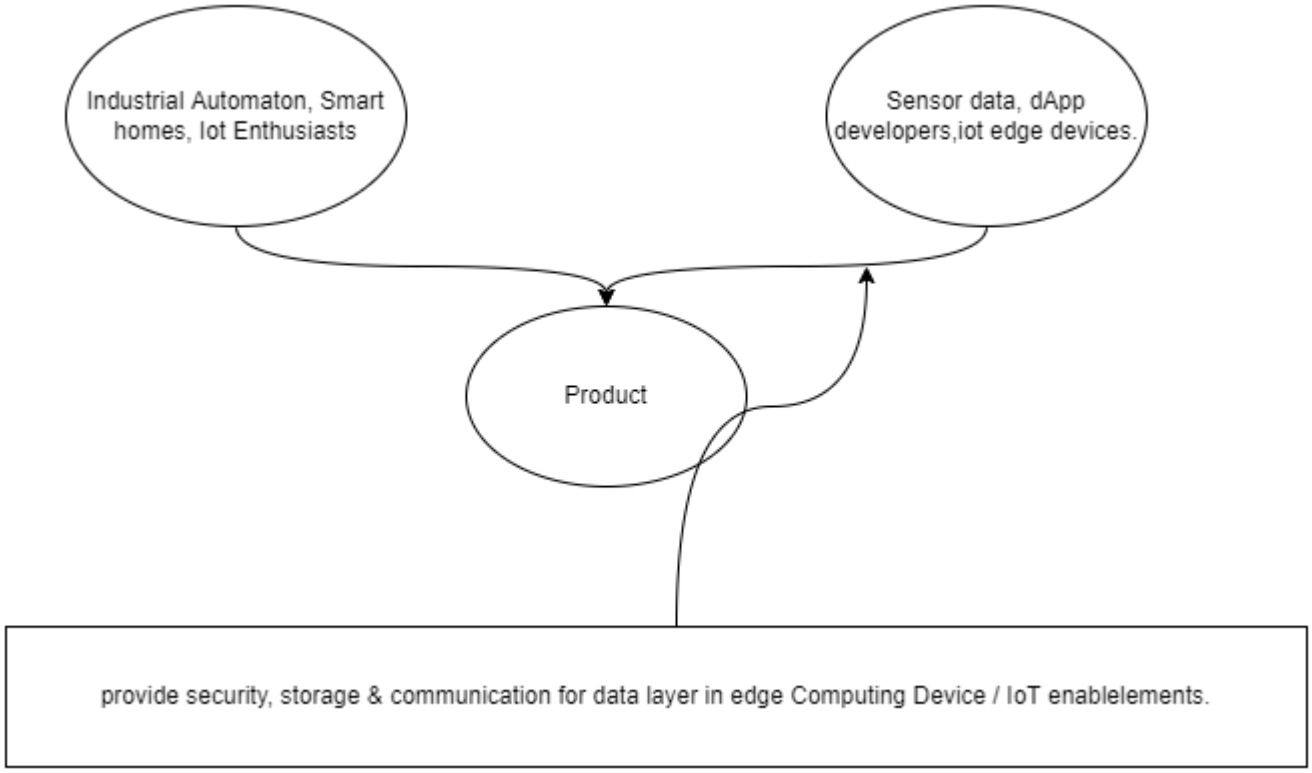

Figure 3.1 – Bull Diagram

3.3 FUNCTIONAL ANALYSIS

The Octopus Diagram describes the functionalities involved in a system. The principal functions specify the primary functionalities that must be provided by the system. The constraint functions are those, which on satisfying the users will provide additional credits to the system or product developed.

3.3.1 OCTOPUS DIAGRAM

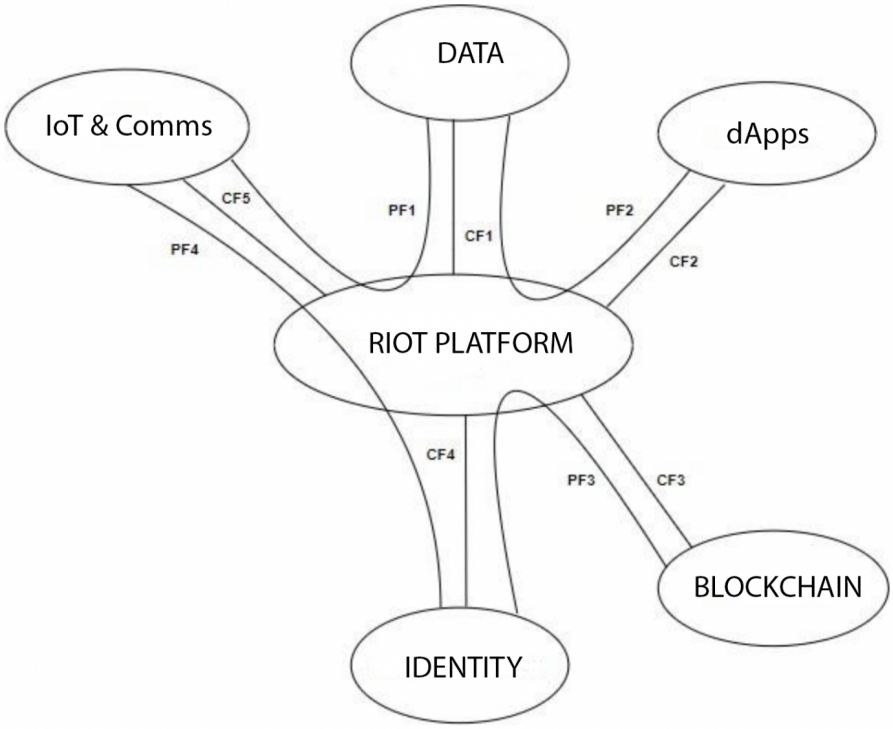

Figure 3.2 – Octopus Diagram

Constraint Functions

CF1	Need to provide Byzantine Fault Tolerance & handle big data
CF2	Ensuring 100% uptime for edge IOT security
CF3	Having an efficient data communication protocol

Principle Functions

PF1	Provide an IoT dApp integration platform
PF2	Mint unique identity for each device on blockchain
PF3	Provide an micro-framework for implementing the decentralized network
PF4	Implement a ledger for enrolling devices
PF5	Provide a decentralized data storage solution via IPFS.

3.4 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. DFDs can also be used for the visualization of data processing. A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored.

3.4.1 DFD-0

DFD 0 is designed to be an abstraction view, showing the system as a single process with its relationship to external entities.

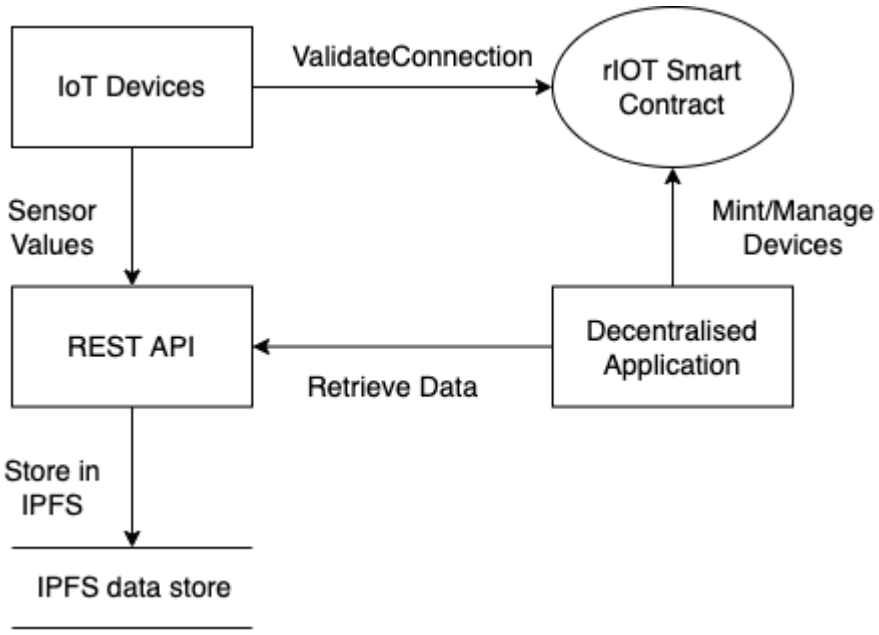

Figure 3.3 – DFD 0

3.4.2 DFD-1

DFD 1 is used to highlight the main functions of the system and break down the high-level process of 0-level DFD into subprocesses.

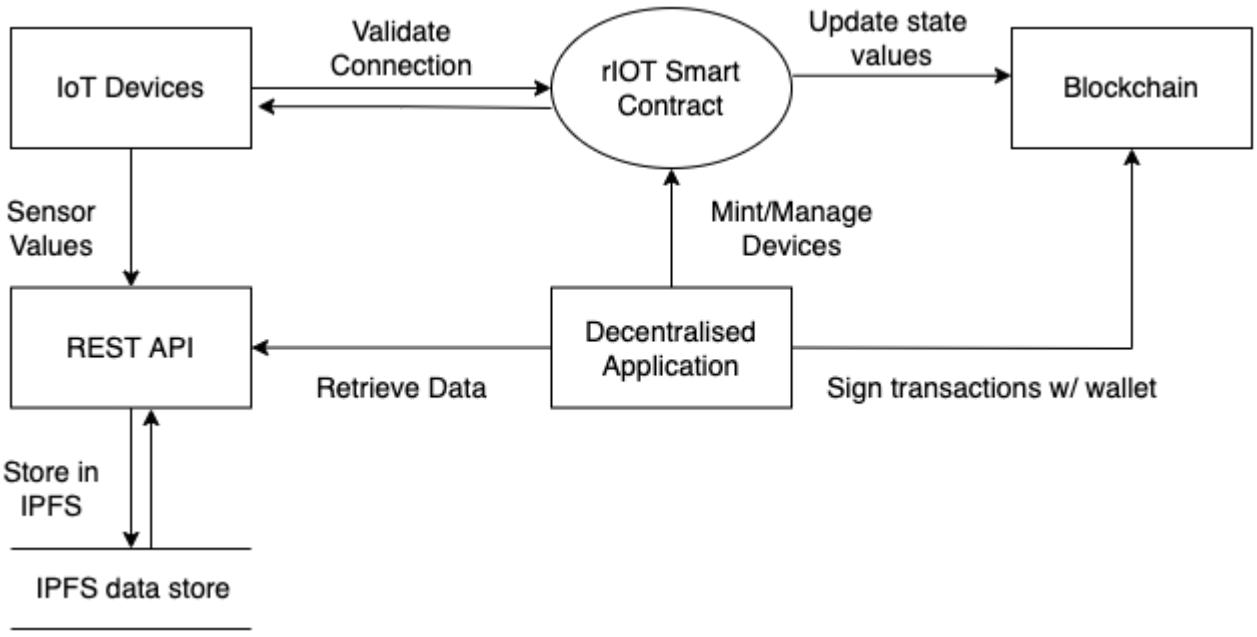

Figure 3.4 – DFD 1

3.4.3 DFD-2

DFD 2 is used to plan or record the specific/necessary detail about the system's functioning.

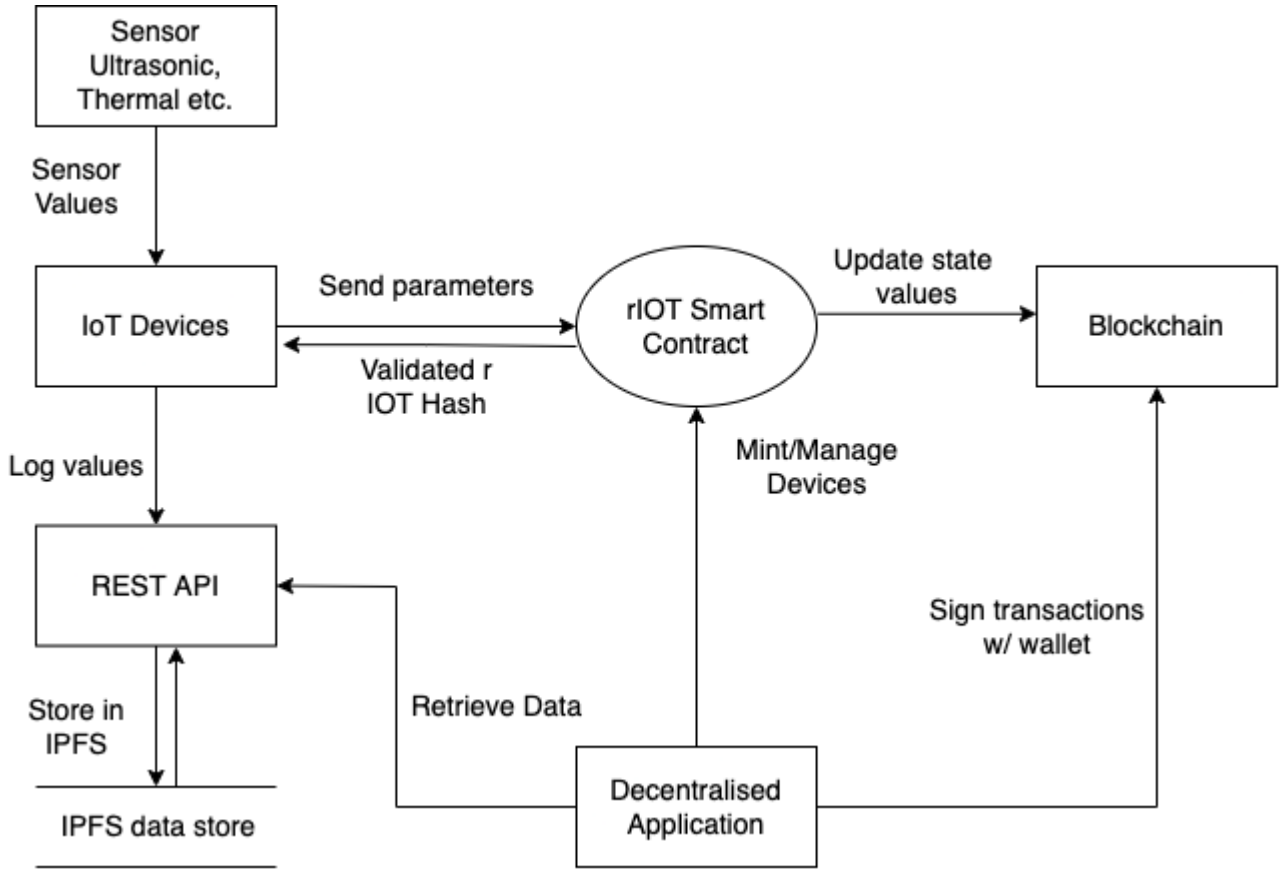

Figure 3.5 – DFD 2

3.5 UML DIAGRAM

UML diagrams are drafted below with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the 'RIOT - DECENTRALIZED IOT GATEWAY PLATFORM'.

3.5.1 USE CASE DIAGRAM

A Use Case Diagram is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

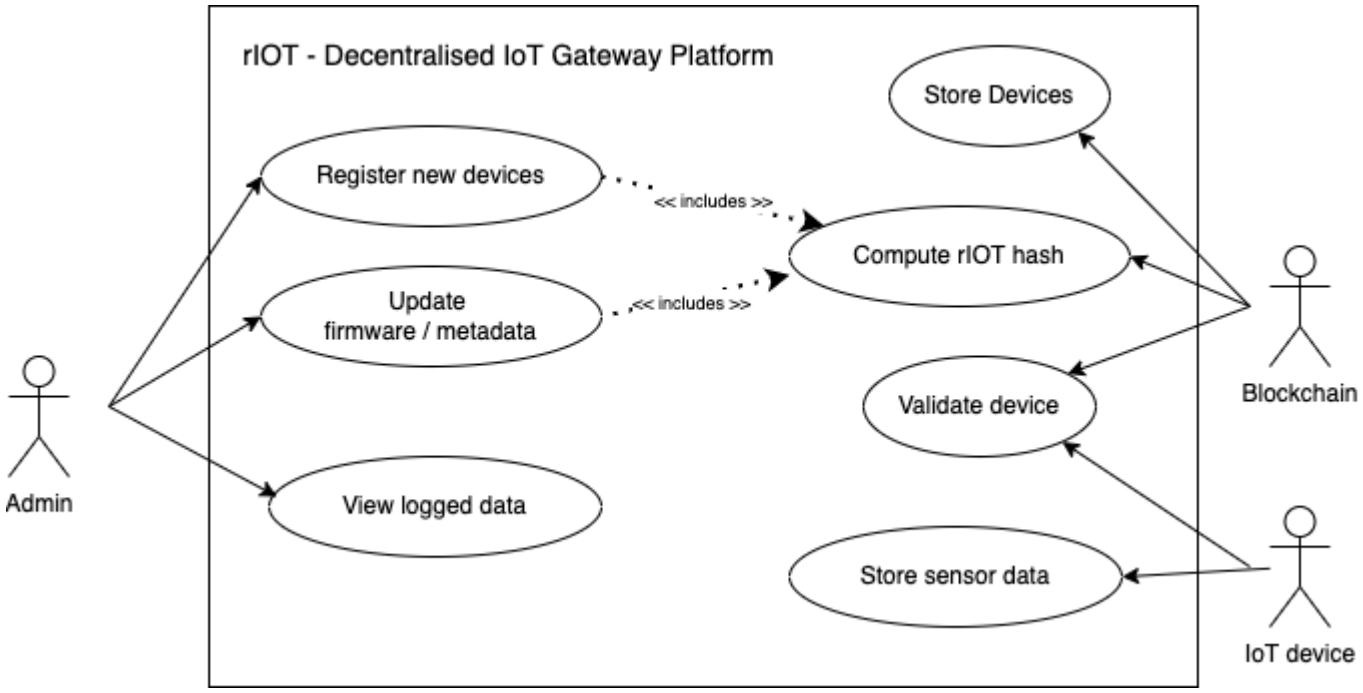

Figure 3.6 – Use Case Diagram

3.5.2 CLASS DIAGRAM

A Class Diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

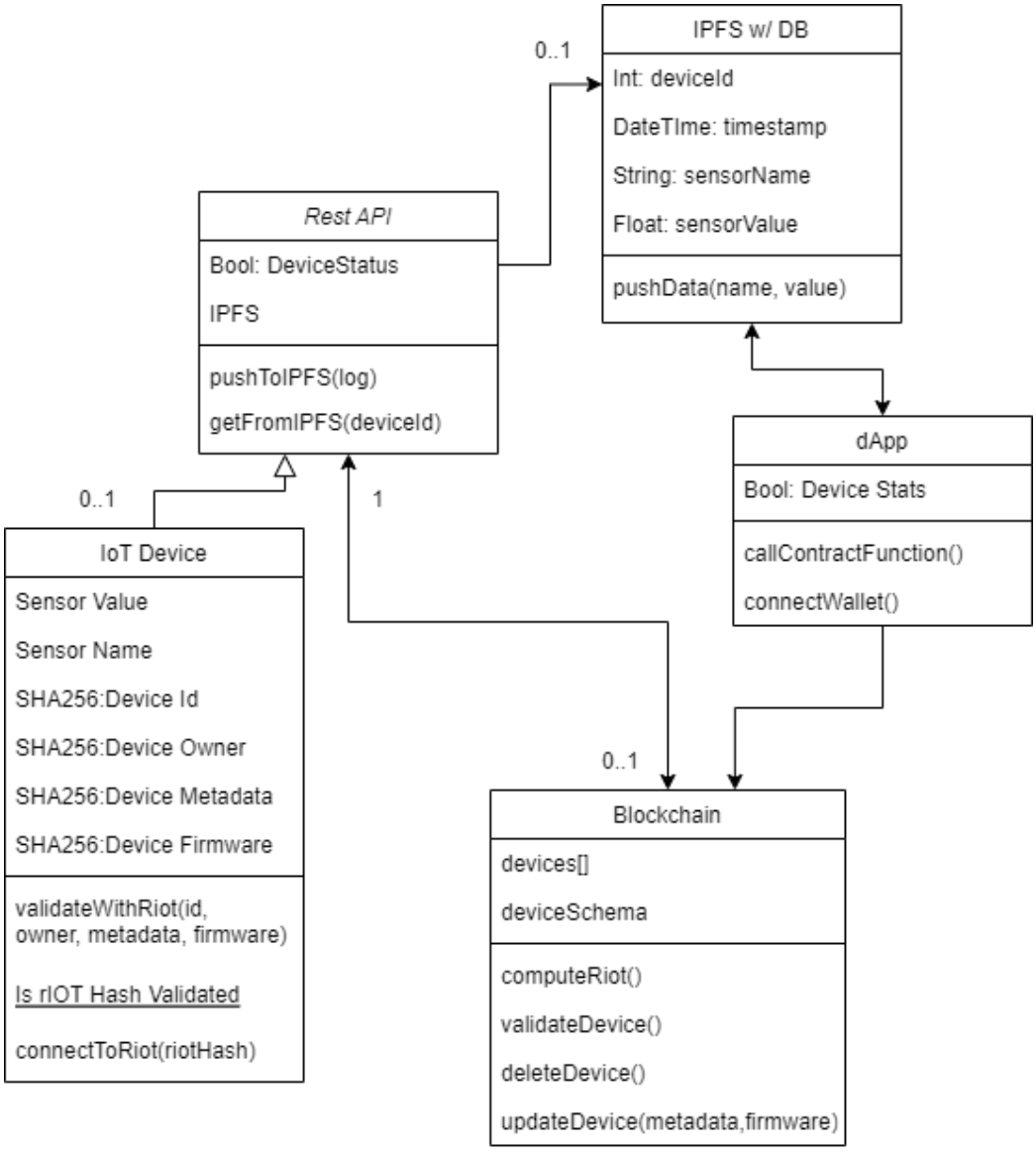

Figure 3.7 – Class Diagram

3.5.3 SEQUENCE DIAGRAM

A Sequence Diagram shows object interactions arranged in time sequence. It depicts the sequence of messages exchanged between the objects to carry out the functionality of the scenario.

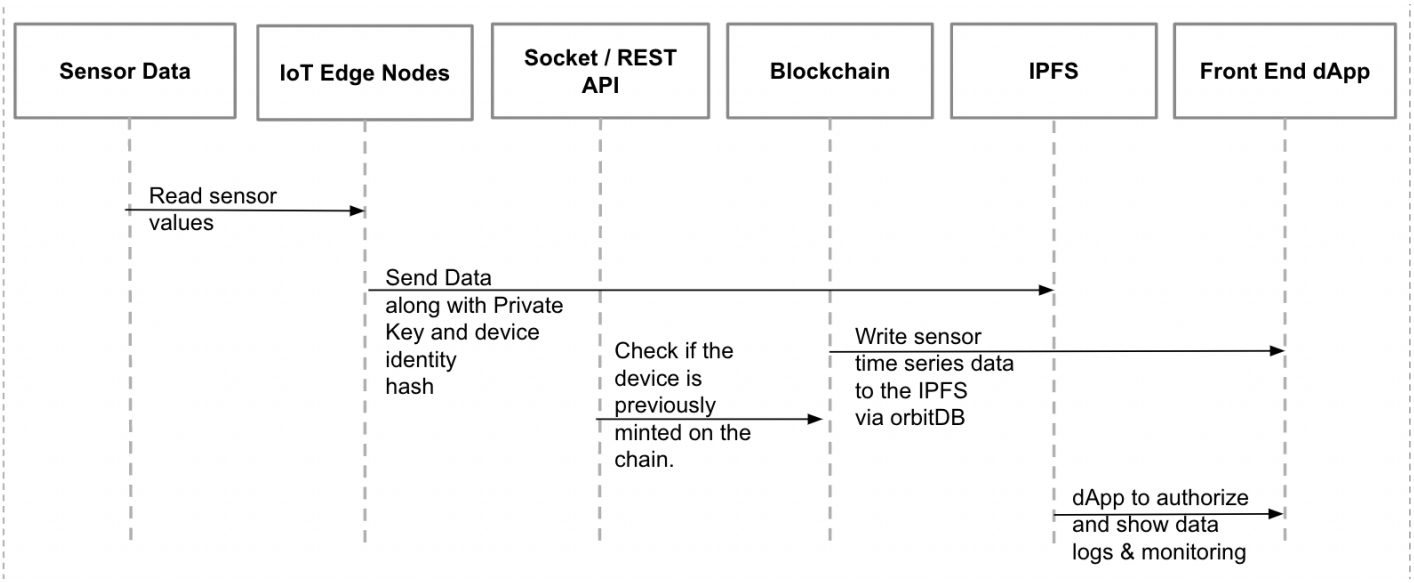

Figure 3.8 – Sequence Diagram

3.5.4 COLLABORATION DIAGRAM

A Collaboration Diagram models the interactions between objects or parts in terms of sequenced messages.

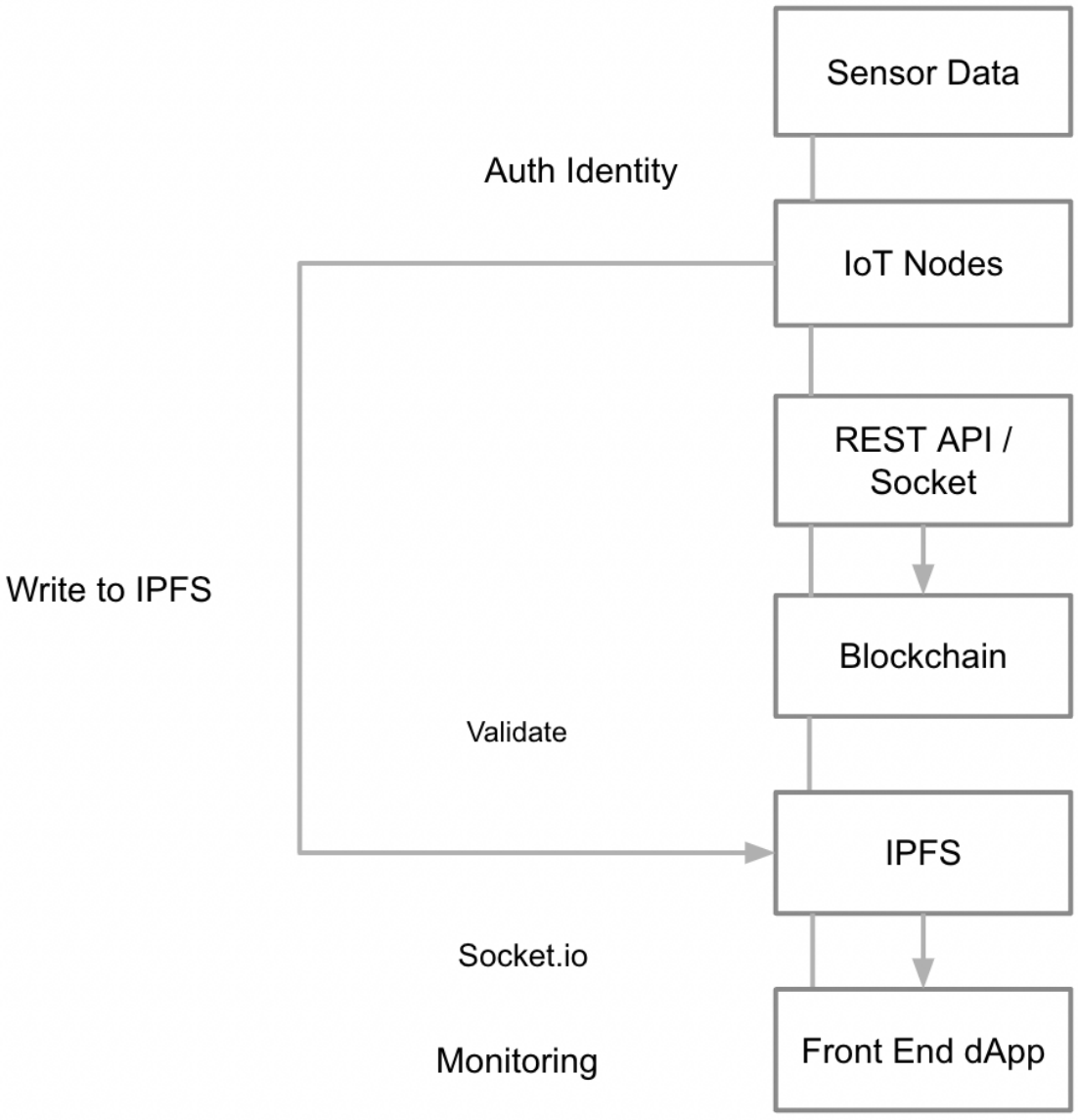

Figure 3.9 – Collaboration Diagram

3.5.5 STATE CHART DIAGRAM

A State Diagram shows the behavior of classes in response to external stimuli. Specifically, it describes the behavior of a single object in response to a series of events in a system.

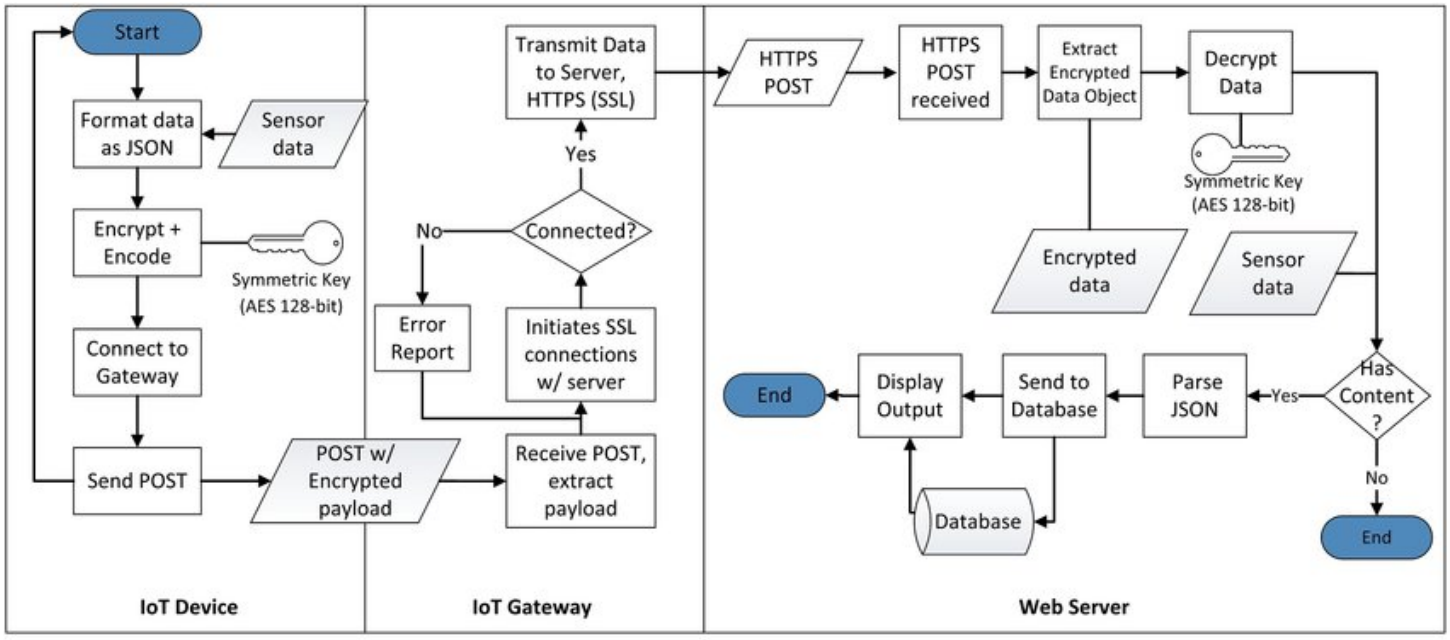

Figure 3.10 – State Chart Diagram

3.5.6 ACTIVITY DIAGRAM

Activity Diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency.

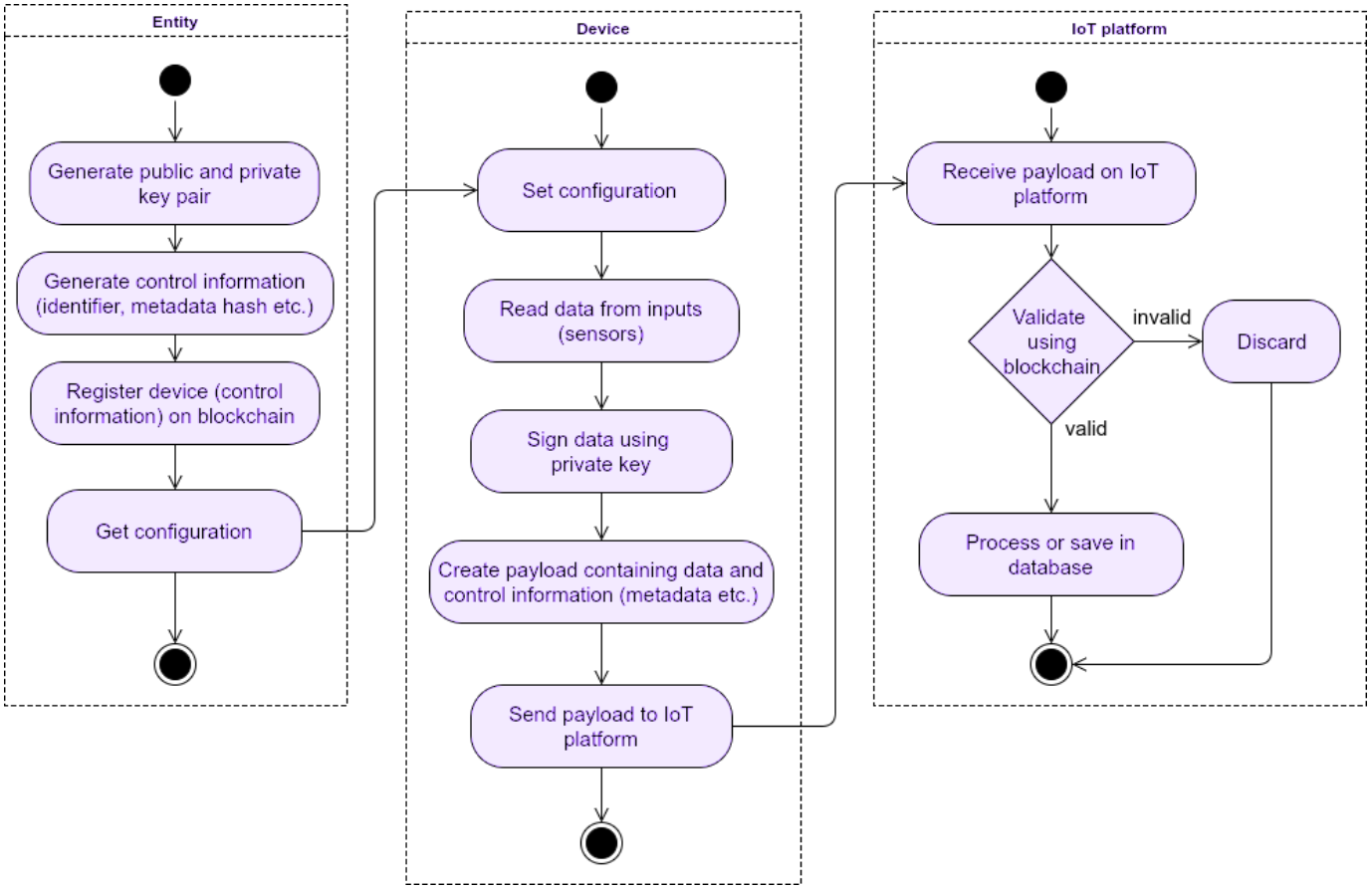

Figure 3.11 – Activity Diagram

3.5.7 COMPONENT DIAGRAM

A Component Diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

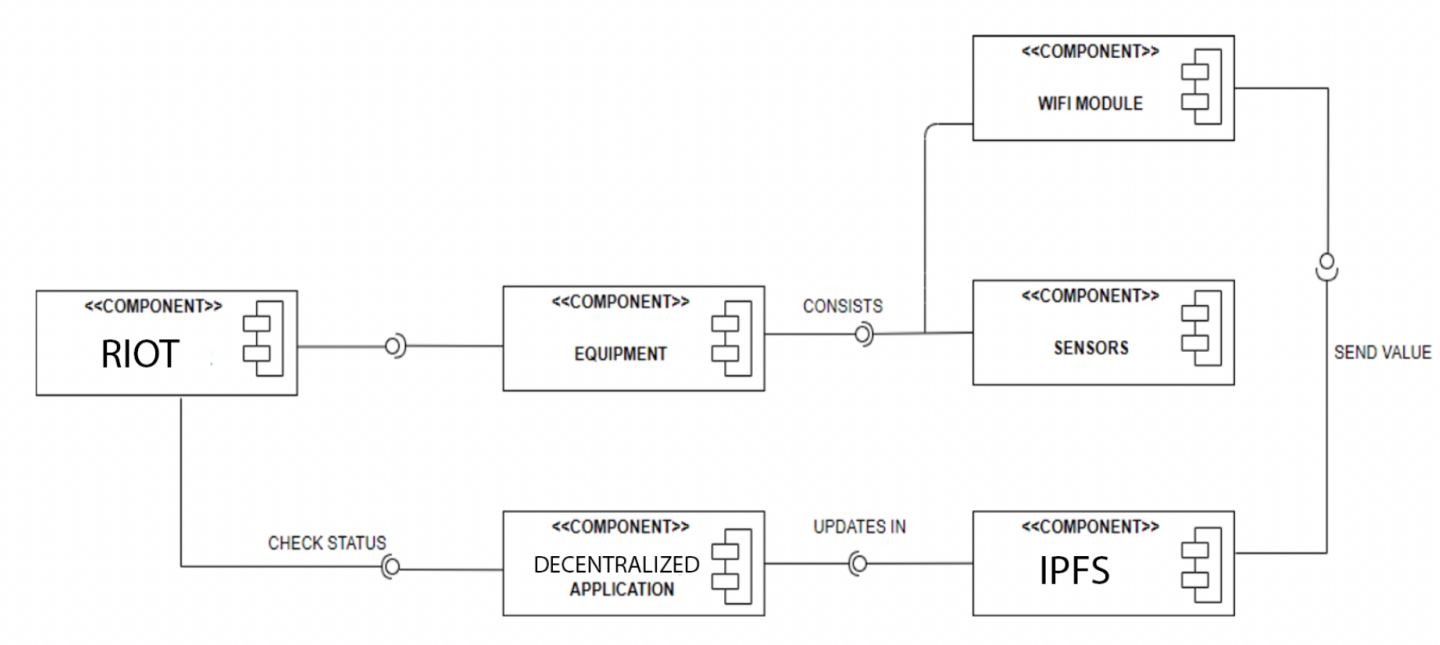

Figure 3.12 – Component Diagram

3.5.8 DEPLOYMENT DIAGRAM

A Deployment Diagram models the physical deployment of artifacts on nodes. It shows what hardware components exist, what software components run on each node, and how the different pieces are connected.

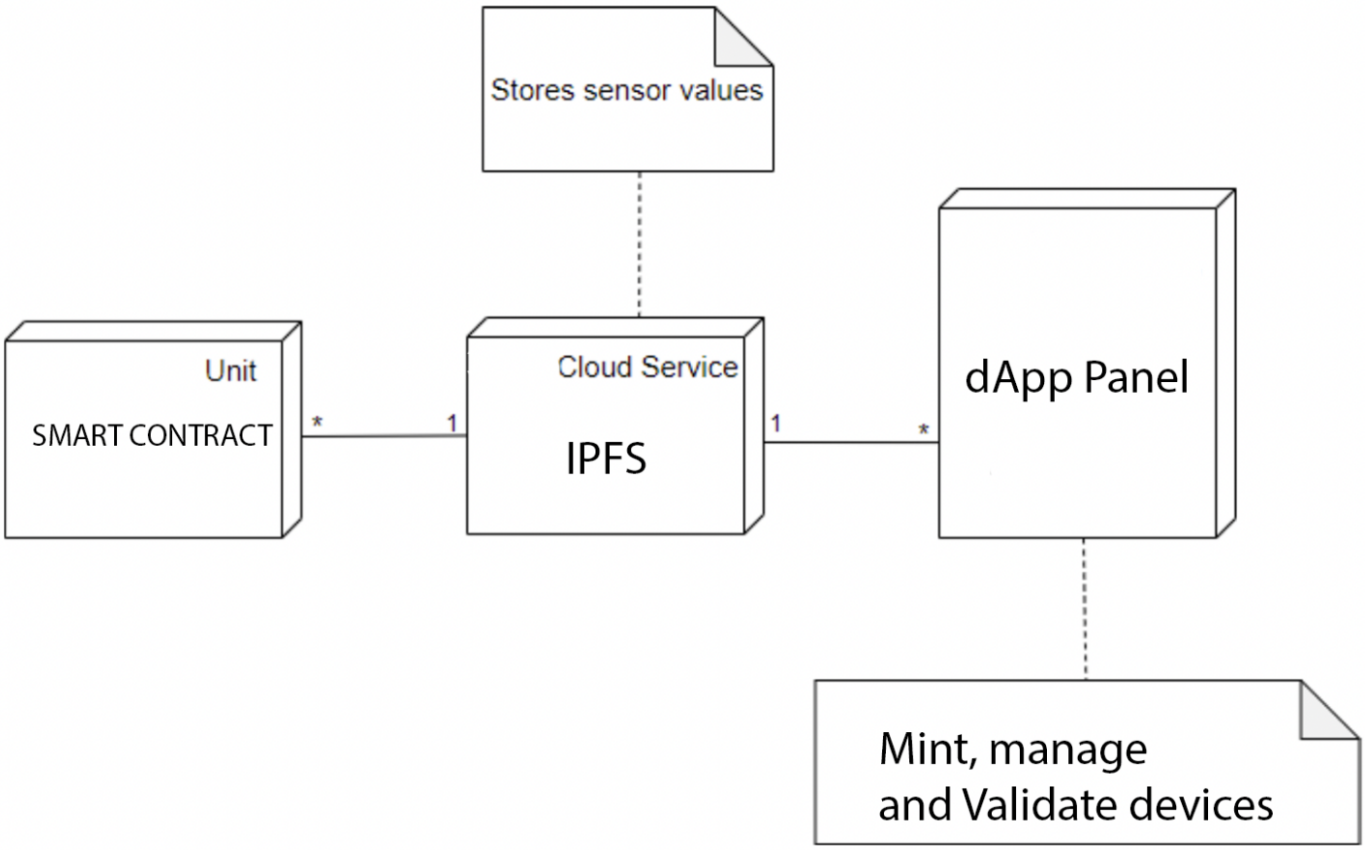

Figure 3.13 – Deployment Diagram

3.5.9 PACKAGE DIAGRAM

Package Diagram can be used to simplify complex class diagrams, it can group classes into packages. A package is a collection of logically related UML elements.

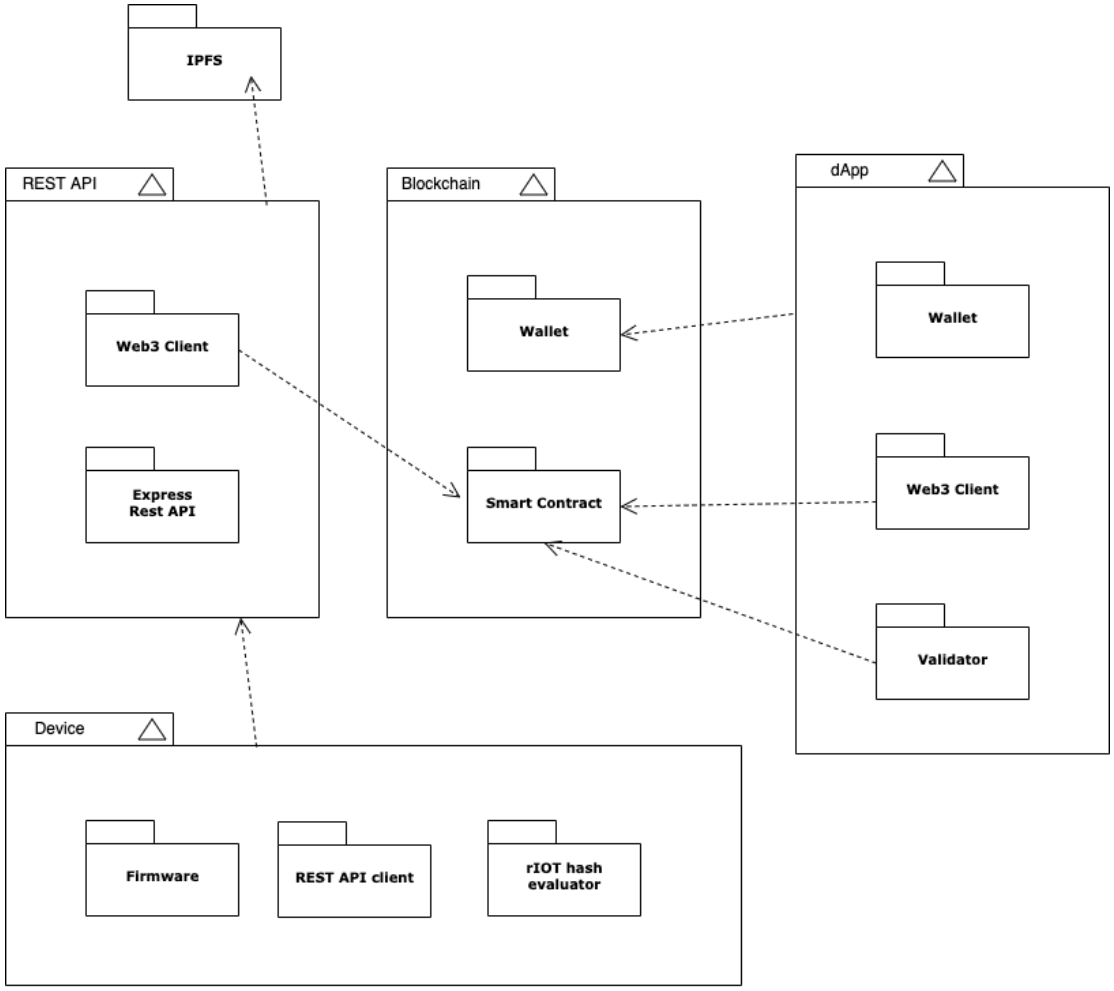

Figure 3.14 – Package Diagram

3.6 SUMMARY

All the UML diagrams for the system are drafted for a better understanding of the system's objects, functions and classes.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 FUNCTIONAL REQUIREMENTS

4.1.1 Software Requirements

4.1.1.1 Polygon

Polygon is a layer 2 blockchain that is built on top of the Ethereum Virtual Machine. It is becoming the widest used blockchain technology owing to its cheaper gas fees. The smart contracts are deployed on this chain and for development purposes - the mumbai testnet is used.

4.1.1.2 dApp

A decentralized application is an application that can operate autonomously, typically through the use of smart contracts, that runs on a decentralized computing blockchain system. Like traditional applications, DApps provide some function or utility to its users. The riot platform built its dapp using moralis SDK and the react library. The contract integrations are done using web3.js , Wallet integrations are also done in such dApps.

4.1.1.3 IPFS

IPFS allows users to host and receive content in a manner similar to BitTorrent. As opposed to a centrally located server, IPFS is built around a decentralized system[5] of user-operators who hold a portion of the overall data, creating a resilient system of file

storage and sharing. Any user in the network can serve a file by its content address, and other peers in the network can find and request that content from any node who has it using a distributed hash table (DHT).

4.1.1.4 REST API - Express

Express JS on nodeJS is used to build the REST API to which the IoT devices can connect to. This Backend handles all validations both on-chain verification and IPFS data logging functionalities.

4.1.2 Hardware Requirements

4.1.2.1 ESP 8266

The ESP-01S ESP8266 WiFi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all WiFi networking functions from another application processor. These device will be running the firmware that is capable of connecting with the RioT Rest API for relaying data to the platform.

4.2 NON-FUNCTIONAL REQUIREMENTS

4.2.1 Performance Requirements

Dynamic numerical requirements:

- 80% of the operations shall be processed in less than

10s. Static numerical requirements:

- The number of simultaneous users: Many
- The number of Kit per unit: 1

4.3 SUMMARY

The proposed blockchain enabled decentralized IoT gateway utilizes the blockchain technology to register and validate the legitimacy of the connected devices in a smart infrastructure also providing means to store data on a decentralized storage medium that can solve the data, security and the communication problems of “Internet of Things

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 SYSTEM ARCHITECTURE

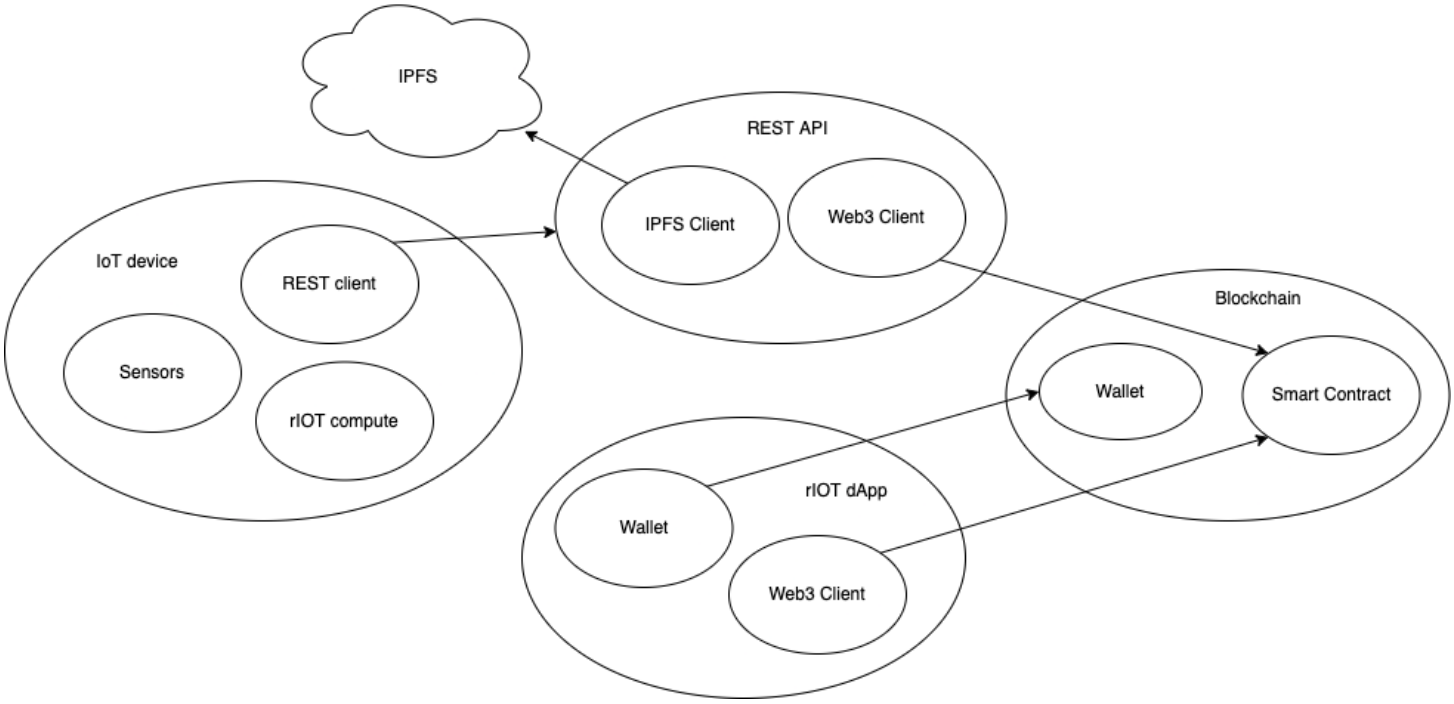

Figure 5.1 – System Architecture

The proposed system platform used the cryptographic hashes - one of the important characteristics of blockchain to leverage security in the IOT devices. Simply, a device is first minted (registered) on the blockchain similar to the NFTs on their smart contract. The mint occurs by creating the rIOT hash based on firmware, metadata, owner address and the device identifier.

The system's gateway is implemented through the Express JS Rest API framework and the dApp is built using the React library.

The web3 client is used to run the on-chain smart contract methods and a wallet connecter is needed for that purpose - could be metamask, coinbase, etc.

And IPFS client is required to interact with the IPFS nodes and hence store and retrieve the logged sensor data.

The next time the device tries to authenticate using the rIOT platform, this rIOT hash is compared with the on-chain data and allows further access to the IPFS client API to transact the sensor data. The admin can use the front end dApp to mint, view devices and look at the sensor data collected by the rIOT platform. This is done completely decentralized from end to end, meaning there's no single point of failure and hence byzantine fault tolerance is achieved.

MODULE DESCRIPTION

5.2.1 rIOT device firmware module

This module acts as the edge device authentication module with a modified bootloader that connects via a TCP socket to the rIOT authentication server.

The computed device hash (merkle root hash of firmware content, device metadata, device owner and identifier) is then validated with the blockchain contract state to allow inbound connections to the platform.

5.2.2 rIOT device management module

The device management module is used to register / mint a new device into the blockchain. For each new mint - a new merkle hash is computed and inserted into the blockchain. The device status can also be monitoring to know if the device firmware has been tampered. A blockchain wallet can be connected to to map the device owner and authenticate the platform

5.2.3 rIOT on-chain module - smart contract

This module is the blockchain module that works on the smart contract deployed on the polygon network. The contract consists of :device to :owner mapping that has the merkle root hash that validates whenever the contract method is called.
The on-chain data may also have the IPFS metadata to index the data associated with the devices.

5.2 SUMMARY

The proposed system is to secure connections via an IoT gateway which can let us log sensor data into the data platform after validating the connected device by the rIOT hash with the on-chain smart contract on the polygon blockchain. On verification they connection is allowed for logging data into the IPFS. The dapp lets us create more devices into the system while ensuring and protecting the devices with an option to burn/delete the device or update the firmware and metadata hash.

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 PERFORMANCE EVALUATION

The available systems come up with a functionality to provide security based on token authentication or session based authentication which still has points of failure that can effectively open backdoors for hackers.

The data store provision via IPFS might be slower than most databases but it provides a much more significant characteristic - i.e byzantine fault tolerance and data permanence.

PARAMETER	EXPERIMENTAL VALUE
Data Push	>1500 ms
Data Pull	>2000ms
Points of failure	100-10000 nodes
Data Replication	True
Data Permanence	True

Table 1 – ***Parameter Vs Experimental Value for decentralized solution***

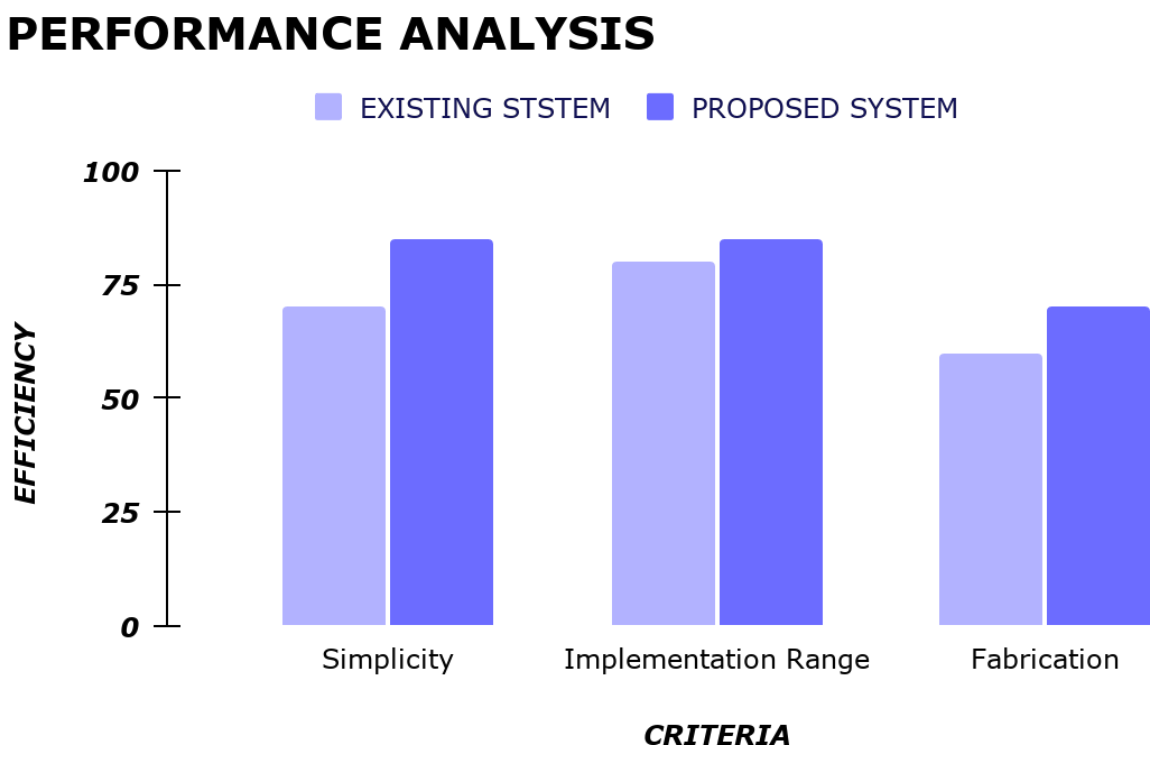

Figure 6.1 – Performance Analysis

The above graph is an output of the logical study conducted between the existing systems and the proposed system. The X-axis denotes the criteria of evaluation and Y-axis denotes the efficiency. The existing systems possess certain disadvantages such as higher points of failure, low scalability and less secure gateways. The on-chain data validation gives an upper hand in security and scalability.

The fabrication of such platforms is also decentralized hence not dependent on a single component. The implementation of the rIOT platform has proven to be effective data store solution and an IOT gateway that effectively manages the IOT infrastructure.

6.2 TIME FACTOR

1. The module takes less than 10 seconds to come online and to be connected with the mobile application.
2. The latency of the change in values is the same as that of the bandwidth of the WiFi.
3. The time interval between any two successive pulls is 5 seconds.

6.3 SUMMARY

The proposed equipment is more efficient than the other existing systems in terms of simplicity and the security that are used.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The Riot - Decentralized IOT Gateway platform serves the purpose of making smart infrastructures safer. Utilizing the secure characteristics of the blockchain to verify and validate the connecting devices gives an additional layer of security to the Internet of Things. The platform's native IPFS data layer and the REST API to store data from the devices increases the credibility of the software. Also the wallet integrations and the ability to collect gas fees from the user accounts to mint devices on the blockchain is an end to end feature. All of these features make the rIOT platform - a one stop solution for IoT.

7.2 FUTURE ENHANCEMENT

The next level of this project scope involves tokenomics where we can set up device nodes that can be incentivized to compute more advanced hashes. This can by itself become its own blockchain of IoT.

APPENDIX I

SAMPLE CODE

SMART CONTRACT

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.2;

contract Riot {
 uint256 constant NULL = 0;

 address private _owner;
 uint256 public entityId = 0;

 struct Device {
   uint256 deviceId;
   bytes firmwareHash;
   bytes manufacturerHash;
   bytes32 riotHash;
   bool owned;
 }

 mapping(address => Device[]) ownersToDevices;
 Device[] devices;

 constructor() {
   _owner = msg.sender;
 }

 function createDevice(
   uint256 deviceId,
   string calldata manufacturerHash,
   string calldata firmwareHash
 ) public payable {
   require(msg.value >= 0.01 ether, "INSUFFICIENT_FUNDS");
   bytes32 _riotHash = keccak256(
```

```
    abi.encodePacked(deviceId, manufacturerHash, firmwareHash)
  );
  Device memory _device = Device(
    deviceId,
    bytes(manufacturerHash),
    bytes(firmwareHash),
    _riotHash,
    true
  );
  ownersToDevices[msg.sender][entityId] = _device;
  devices[entityId] = _device;
  entityId += 1;
}

function updateFirmware(uint256 _entityId, string calldata _firmwareHash)
  public
  validate(_entityId)
{
  Device storage _device = ownersToDevices[msg.sender][_entityId];
  _device.firmwareHash = bytes(_firmwareHash);
  _device.riotHash = hashingHashes(
    _device.deviceId,
    _device.manufacturerHash,
    _device.firmwareHash
  );
}

function updateManufacturerId(
  uint256 _entityId,
  string calldata _manufacturerHash
) public validate(_entityId) {
  Device storage _device = ownersToDevices[msg.sender][_entityId];
  _device.manufacturerHash = bytes(_manufacturerHash);
  _device.riotHash = hashingHashes(
    _device.deviceId,
    _device.manufacturerHash,
    _device.firmwareHash
  );
}
```

```
function deleteDevice(uint256 _entityId) public validate(_entityId) {
  Device storage _device = ownersToDevices[msg.sender][_entityId];
  _device.owned = false;
  devices[_entityId].owned = false;
}

modifier validate(uint256 _entityId) {
  require(_entityId < entityId, "DEVICE_UNAVAILABLE");
  require(ownersToDevices[msg.sender][_entityId].owned, "NO_ACESS");
  Device memory _device = ownersToDevices[msg.sender][entityId];
  require(
    keccak256(
      abi.encodePacked(
        _device.deviceId,
        _device.manufacturerHash,
        _device.firmwareHash
      )
    ) == _device.riotHash,
    "MALICIOUS"
  );
  _;
}

function withdraw() public {
  require(msg.sender == _owner, "YOU_ARE_NOT_THE_OWNER");
  (bool callSuccess, ) = payable(msg.sender).call{
    value: address(this).balance
  }("");
  require(callSuccess, "TRANSACTION_FAILED");
}

function hashingHashes(
  uint256 _deviceId,
  bytes memory _manufacturerHash,
  bytes memory _firmwareHash
) internal pure returns (bytes32) {
  return
    keccak256(abi.encodePacked(_deviceId, _manufacturerHash, _firmwareHash));
}
}
```

FIRMWARE CODE

```
include <WiFi.h>
#include <Web3.h>
#include <Contract.h>
#define USE_SERIAL Serial
#define ENV_SSID "rIOTPhantomNode"
#define ENV_WIFI_KEY "8754462663"
#define MY_ADDRESS "0x<MY_ADDRESS>"
#define CONTRACT_ADDRESS "0x64574ddbe98813b23364704e0b00e2e71fc5ad17>"
#define INFURA_HOST "rinkeby.infura.io"
#define INFURA_PATH "dausihco8sahd8ailuhsc87asghcaluisdbiuscgas"

const char PRIVATE_KEY[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

Web3 web3(INFURA_HOST, INFURA_PATH);

void eth_send_example();

void setup()
{

  USE_SERIAL.begin(115200);
  for (uint8_t t = 4; t > 0; t--)
  {
      USE_SERIAL.printf("[SETUP] WAIT %d...\n", t);
      USE_SERIAL.flush();
     delay(1000);
  }

  WiFi.begin(ENV_SSID, ENV_WIFI_KEY);
  // attempt to connect to Wifi network:
  while (WiFi.status() != WL_CONNECTED)
  {

     Serial.print(".");
```

```
    // wait 1 second for re-trying
    delay(1000);
  }

  USE_SERIAL.println("Connected");
  eth_send_example();
}

void loop()
{
}

void eth_send_example()
{

  Contract contract(&web3, CONTRACT_ADDRESS);
  contract.SetPrivateKey((uint8_t*) PRIVATE_KEY);
  uint32_t nonceVal = (uint32_t) web3.EthGetTransactionCount((char*) MY_ADDRESS);
  uint32_t gasPriceVal = 141006540;
  uint32_t gasLimitVal = 3000000;
  uint8_t toStr[] = CONTRACT_ADDRESS;
  uint8_t valueStr[] = "0x00";
  uint8_t dataStr[100];
  memset(dataStr, 0, 100);
  string func = "set(uint256)";
  string p = contract.SetupContractData(&func, 123);
  string result = contract.SendTransaction(nonceVal, gasPriceVal, gasLimitVal, &toStr,
&valueStr, &p);
  USE_SERIAL.println(result);
}
```

APPENDIX II

SNAPSHOTS

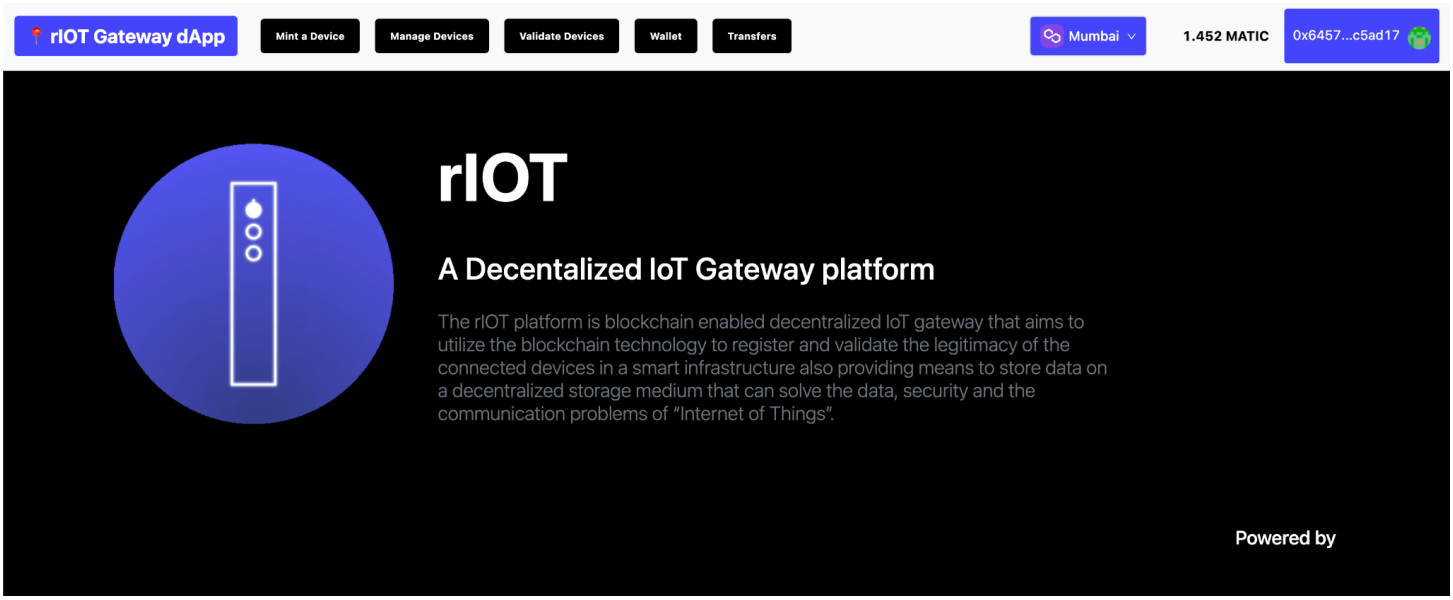

FIGURE A

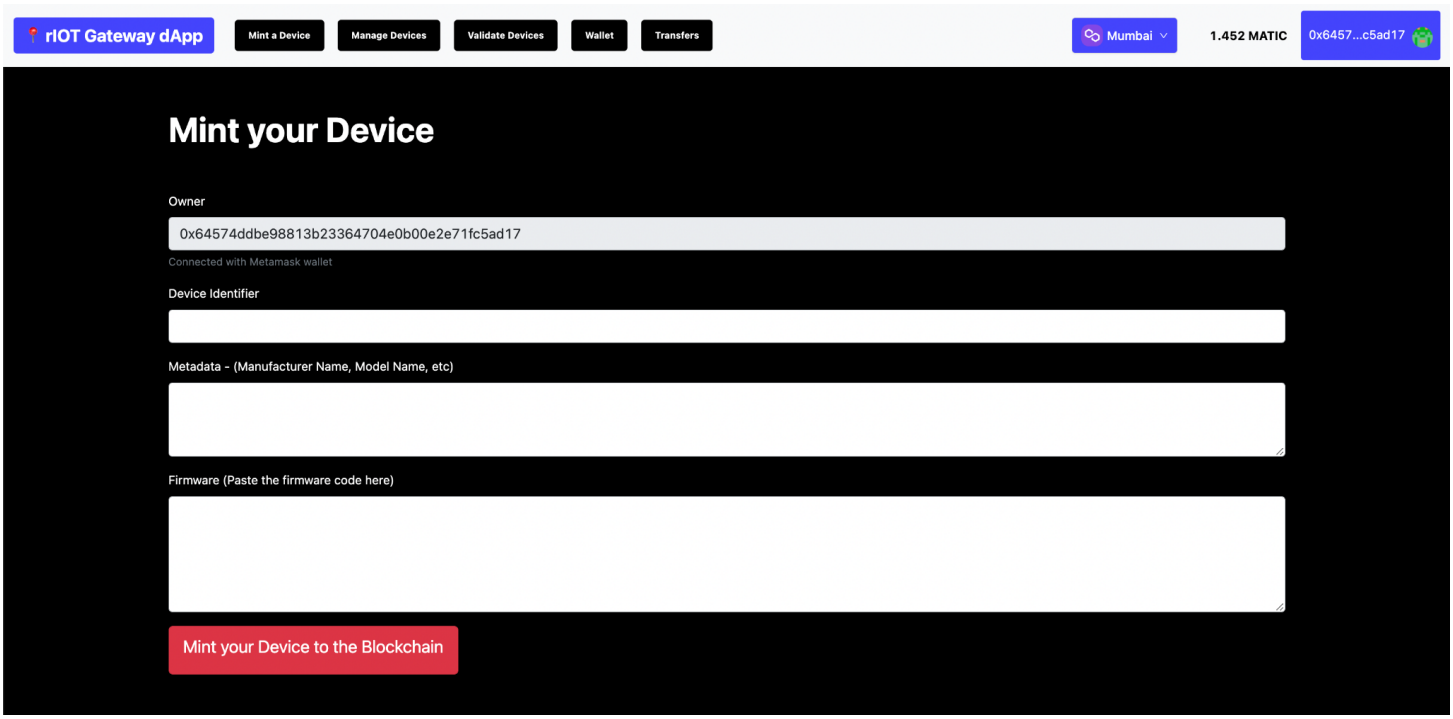

FIGURE B

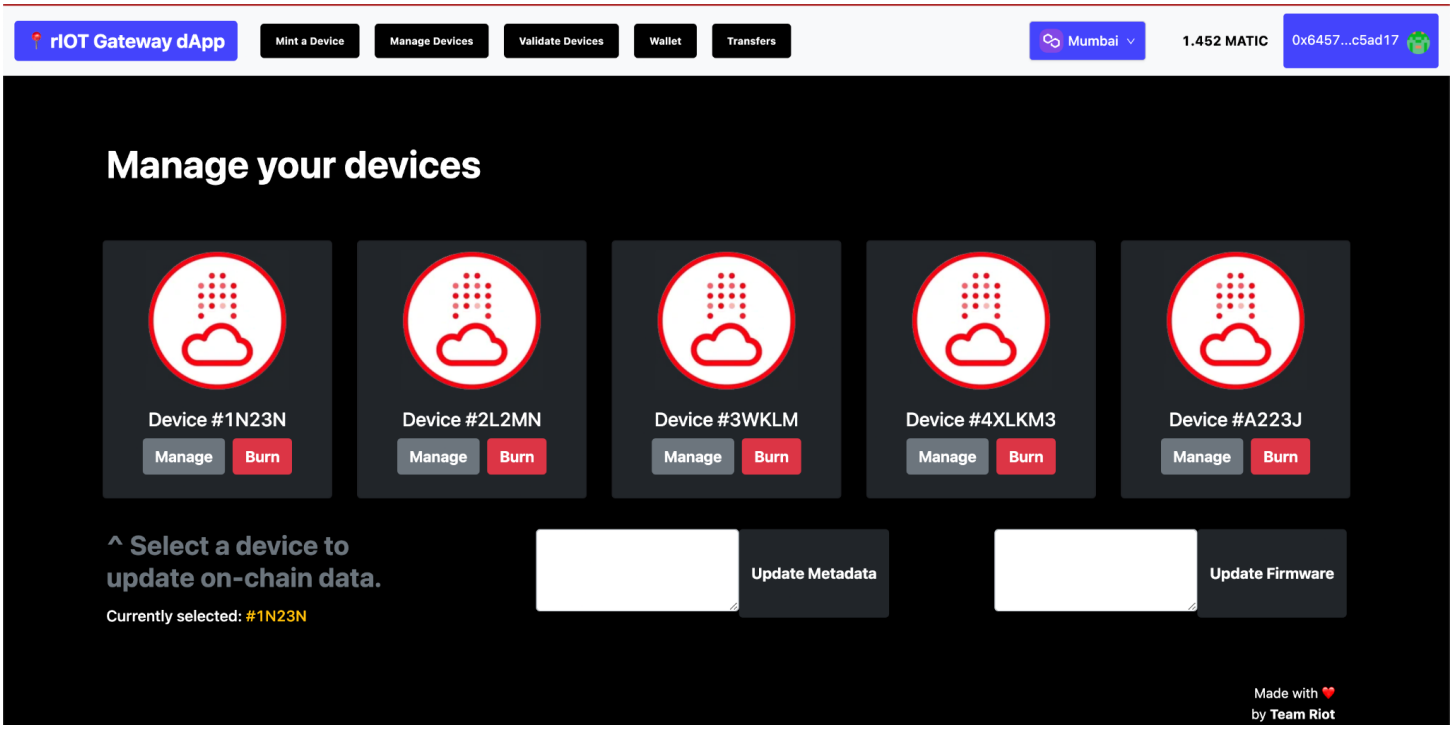

FIGURE C

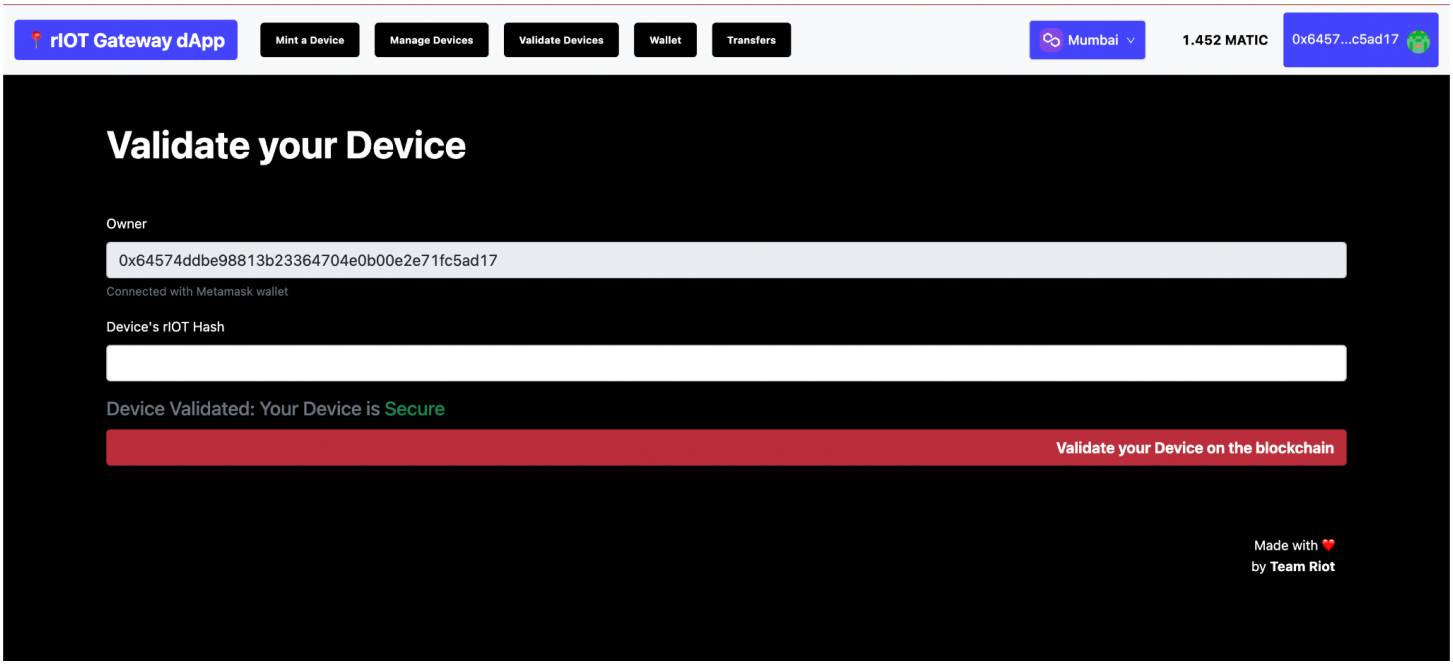

FIGURE D

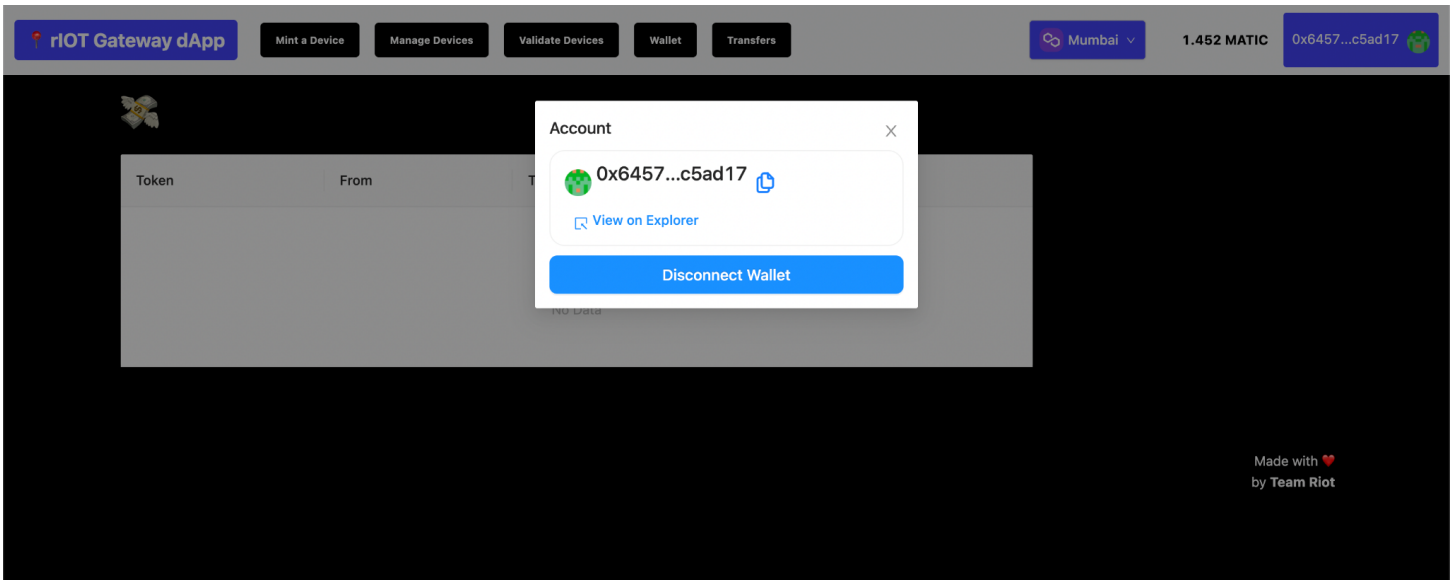

FIGURE E

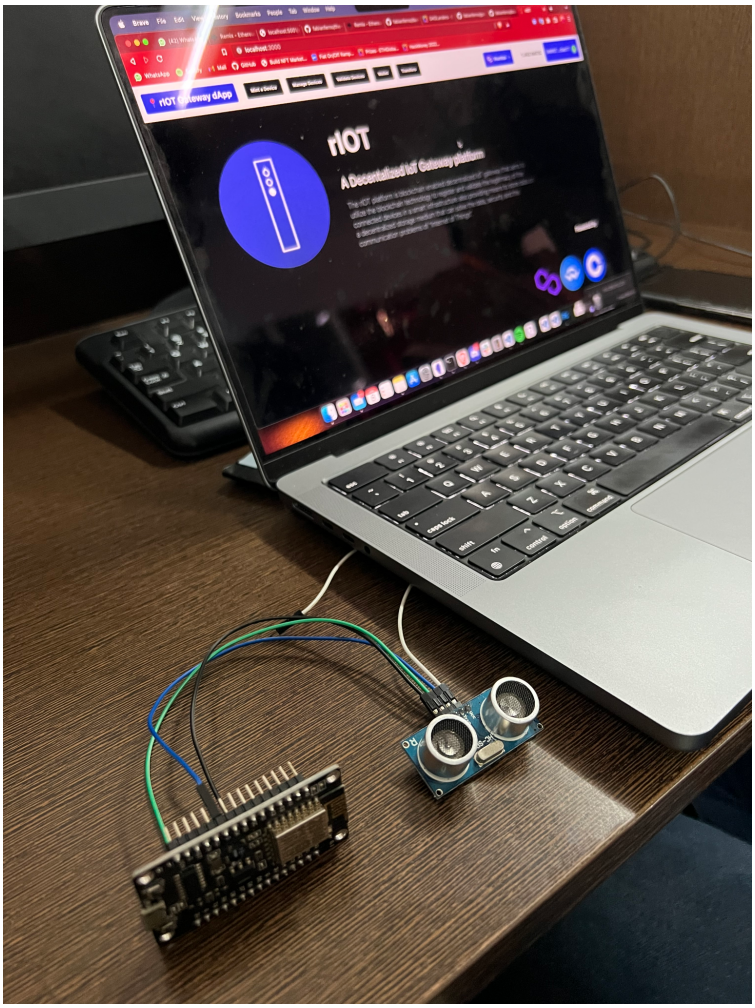

FIGURE F

APPENDIX III

REFERENCE

[1] Balakrishnan S,S Sheeba Rani , K C Ramya “Design and Development of IoT Based Smart Aquaculture System in a Cloud Environment”. In International Journal of Oceans and Oceanography (11 January 2017)

[2] Cesar Encinas, Erica Ruiz, Joaquin Cortez, Adolfo Espinoza ”Design and implementation of a distributed IoT system for the monitoring of water quality in aquaculture”. Wireless Telecommunications Symposium (WTS) 2017

[3] De Silva SS, Soto D, 2009, “Climate change and aquaculture: Potential impacts, adaptation and mitigation. In Climate change implications for fisheries and aquaculture: overview of current scientific knowledge”, eds. K. Cochrane, C. De Young, D. Soto, and T. Bahri. FAO Fisheries and Aquaculture Technical Paper. No. 530, 151–212.

[4] Duy, N. T. K., Tu, N. D., Son, T. H., & Khanh, L. H. D. (2015, March). Automated monitoring and control system for shrimp farms based on embedded system and wireless sensor network. In Electrical, Computer and Communication Technologies (ICECCT), 2015 IEEE International Conference on (pp. 1-5). IEEE.

[5] Espinosa-Faller, F.J.; Rendón-Rodríguez, G.E. 2012, “A ZigBee Wireless Sensor Network for Monitoring an Aquaculture Recirculating System”. J.Appl. Res. Technol. 2012, 10, 380–387.

[6] Kamarul Hafiz Kamaludin, W. Ismail “Water quality monitoring with internet of things (IoT) “. In IEEE Conference on Systems, Process and Control (ICSPC) Dec 2017

[7] P. Y. Yang, J. T. Tsai, J. H. Chou, W. H. Ho and Y. Y. Lai, "Prediction of water quality evaluation for fish ponds of aquaculture," 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, 2017, pp. 545-546.

[8] S. K. Vaddadi, Development of Embedded Wireless Network and Water Quality Measurement Systems for Aquaculture, Sixth Int. Conf. Sens.Technol. Dev., pp. 637641, 2012.

[9] Lee, P. G. (1995). A review of automated control systems for aquaculture and design criteria for their implementation. Aquacultural Engineering, 14(3), 205-227.

[10] Hu, S. (2015, July). Dynamic monitoring based on wireless sensor networks of IoT. In Logistics, Informatics and Service Sciences (LISS), 2015 International Conference on (pp. 1-4). IEEE.

[11] Duy, N. T. K., Tu, N. D., Son, T. H., & Khanh, L. H. D. (2015, March). Automated monitoring and control system for shrimp farms based on embedded system and wireless sensor network. In Electrical, Computer and Communication Technologies (ICECCT), 2015 IEEE International Conference on (pp. 1-5). IEEE.