BLOCKCHAIN BASED SECURE METERING SYSTEM

Final Year Project

Report by

Haseeb Saeed

In Partial Fulfillment

Of the Requirements for the degree

Bachelor of Science in Computer Science (BSCS)

School of Electrical Engineering and Computer Science

National University of Sciences and Technology

Islamabad, Pakistan(2021)

ACKNOWLEDGEMENTS

I am thankful to Almighty Allah for blessing me with guidance, courage, and mindfulness throughout our lives.

I acknowledge with special thanks and appreciation to my project advisor, Dr. Syed Taha Ali, who has been a great source of inspiration for me throughout my tenure of final year. His knowledge and guidance made it easier for me to accomplish my goal even in such tough times of a global pandemic, COVID-19.

Thank you.

TABLE OF CONTENT

LIST OF FIGURES

ABSTRACT

Like other commodities, electricity is now a part of our day-to-day life and we cannot neglect its importance. The conventional methods of electricity production such as using coal, oil and natural gas for steam run generators have been proved very harmful for the environment. Use of such non-renewable energy resources for energy generation is responsible for green gas emissions and air and water pollution. This has led mankind to move towards the other environment friendly alternatives, renewable energy resources like solar and wind energy. Professionals around the world are busy designing and manufacturing the equipment and tools for energy generation using such resources. Generation is now considered a task we can claim our command on. But energy trading is still a problem. The energy regulation departments require a smart network for peer-to-peer transactions that requires no centralized authority. Learning from the growth pattern in last decade, blockchain has proved to be the most suitable and revolutionized concept. Setting aside its scope in cryptocurrencies, the idea of smart contracts has proved helpful in connecting blockchain to other tracking systems such as drugs and medicine tracking, insurance systems, real estate, food tracking, drugs and medicine tracking etc. Acknowledging the pattern of energy transactions via regulatory authorities, a blockchain based secure metering system is introduced. This solution ensures that no central authority is required to regulate the energy transactions between peers. A secure and completely transparent system is introduced for consumers and prosumers with an active microcontroller that monitors and regulates the consumed and produced energy units.

Chapter 01

INTRODUCTION

With each technological breakthrough in modern era, we encounter various problems each day. Some problems are so common and iterative that people have adapted to them but it still requires a solution. With energy thefts being a recursive and most common problem these days, it cannot be ignored. Solving such problems effectively is what paves our way towards a safe, smart and digital world. One such problem is being dealt with through our project, ‘blockchain based secure metering system’.

Electricity is a modern-day necessity to survive and thus is available to everyone at both domestic and industrial level. After its invention and transmission through cables, the next milestone was generation and transmission of electricity at macro level that could be supplied to an entire city. The engineers, researchers and physicists put their lives in discovering the most reliable resources of electricity production. They came up with today’s most commonly used resources i.e., fossil fuels. Fossil fuels include petroleum, coal and natural gas. Combustion of fossil fuel runs the generators responsible for energy production. This mode of production has proved to be hazardous for the environment in several ways. It has mainly caused air pollution and production of green gases contributing to global

warming and sudden climate changes. Other notable factor is water pollution that is contaminating oceans and aquatic life. Another alternative for energy production is via nuclear energy. Nuclear is world's second largest resource producing nearly 11% of total electricity preceded by coal and natural gas that contributes about 60% of total electricity production sources (fig 1).

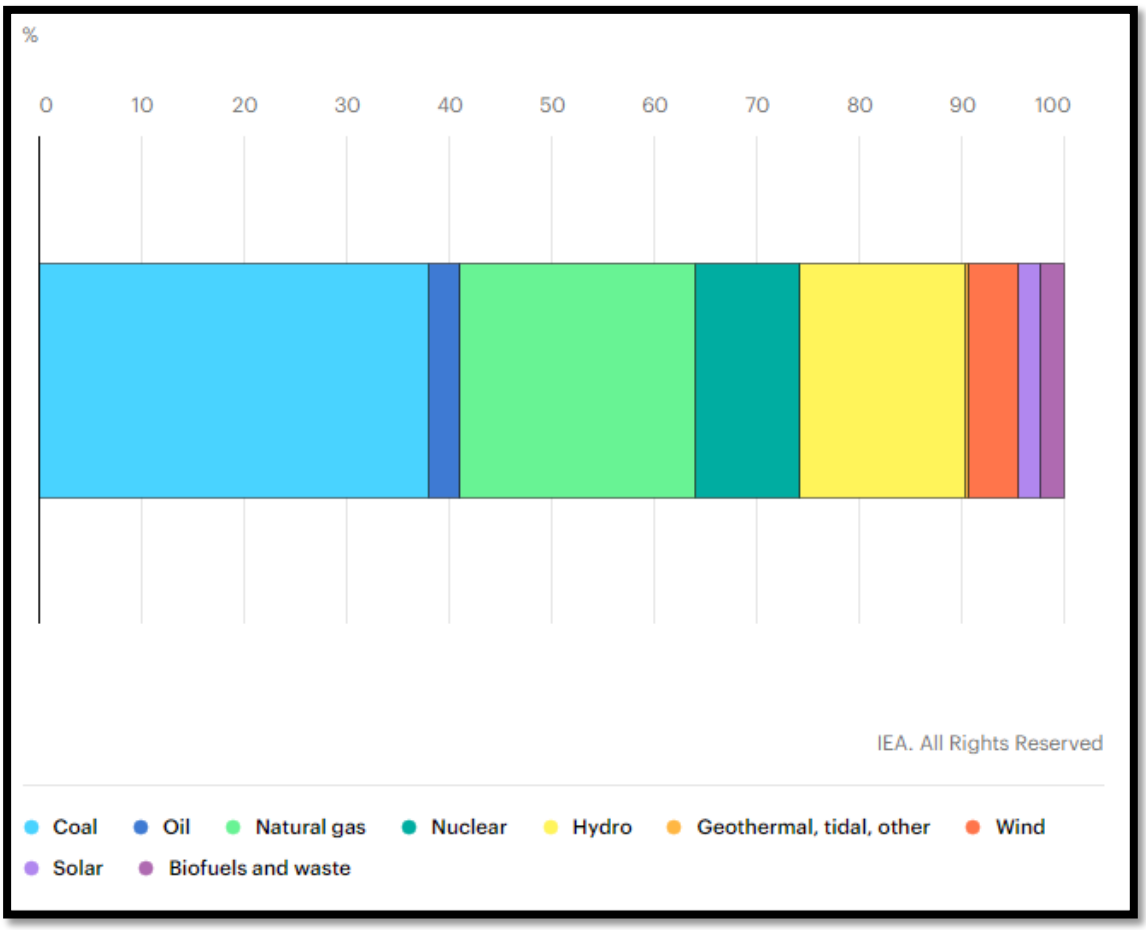

Fig 1: World gross electricity production, by source, 2018

Source: IEA.org

1.1 BLOCKCHAIN

Blockchain is a fairly new word in this world of technology. It has

existed since 1991 but there was no acceptance for it back in the days and it had failed miserably. But later on with the passage of time, the need of hour realized its importance and it was relaunched by the owners again in 2009. Introduction of bitcoin and other cryptocurrencies paved their way through the economic system in the same decade that made blockchain more important. Blockchain has made transactions easy, secure and easier to verify. Notable features are anonymity, reliability and non-repudiation. It allows the prosumer and consumer to communicate in an authentic way without any violations in privacy protocols.

1.2 TRANSACTIVE ENERGY SYSTEM

TES is a newly established mechanism in electric power system that allows different control agents to use distributed generation units in order to engage in energy transactions and provide necessary services. It improves the reliability of a certain energy system. With the increasing trend of distributed energy resources such as rooftop photovoltaic solar units and wind turbines, TES is the best possible solution. Such a system requires a secure, foolproof and reliable platform for its implementation. And blockchain is one such technology that handles the energy transaction between consumers and prosumers in best way. Hence blockchain based secure energy metering system is a well-defined and comprehensive solution that provides complete authority of transactions to prosumers and consumers. The proposed project encourages the prepaid cryptocurrency based billing system and also promotes the green energy sources.

This technology enables the consumer to become the prosumer and vice versa. Consumer is defined as the primary user of the generated energy. Prosumer is a consumer that both generates and consumes energy and has

its own energy generation units installed. Blockchain makes it easier for the prosumer to trade its locally generated energy with other consumers or a distribution company at a central/ national level.

In our project report, we will explain solution for energy thefts and centralized, unauthorized systems using blockchain based metering and transactive system in detail. This introduction is followed by a comprehensive literature review of some research papers we studied to get a better insight of blockchain world. This report extends by explaining the methodology, solution and possible future work.

Chapter 02

LITERATURE REVIEW

OVERVIEW

This chapter explains the precap of research work done before proceeding with the practical implementation of project. The chapter starts with a general and concise problem statement followed by a comprehensive problem description. It includes a proposed solution to the explained problem in a bird's eye view. At the end of this chapter, the main objectives and goals of the project are mentioned.

2.1 PROBLEM STATEMENT

With the increased use of green energy sources, arises the need of a decentralized network instead of in practice centralized transaction network which encourages consumer and prosumer to trade among themselves.
Our target is to provide a platform that promotes peer to peer transactions between consumers and prosumers and encourage smart, secure and digital solutions to energy crisis we are facing globally and locally.

2.2 PROBLEM DESCRIPTION

Blockchain based secure metering system aims at addressing following issues in detail.

2.2.1 Renewable energy sources

Globally, owing to air pollution and other air borne issues caused by energy production using fossil fuel burning, a general trend of shifting towards RES started. This encouraged people to install solar panels and wind turbines (at coastal areas) to contribute for their share in conservation of environment. The growing trend of using solar energy to its full potential is quite evident through a figure explaining net electricity generation in Germany in a week. This says almost 25 % of total electricity produced was via solar panels (fig 2). There are numerous cases where individuals have installed Solar panels in their own capacity and are able to produce enough electricity for their own use, sometimes more than the

required amount. This surplus is either stored in the batteries or sent back to the grid. It may occur to some individuals that they want to trade this extra energy in their neighborhood for money. This is where arises the need for peer-to-peer energy trading systems. The continual usage of such technologies raised the need for a proper system that can give the two trading parties complete autonomous control of trading between them. Currently, taking an example from Pakistan, NEPRA is the most active regulatory authority which generates licenses for such traders and other DISCOs act as a central body.

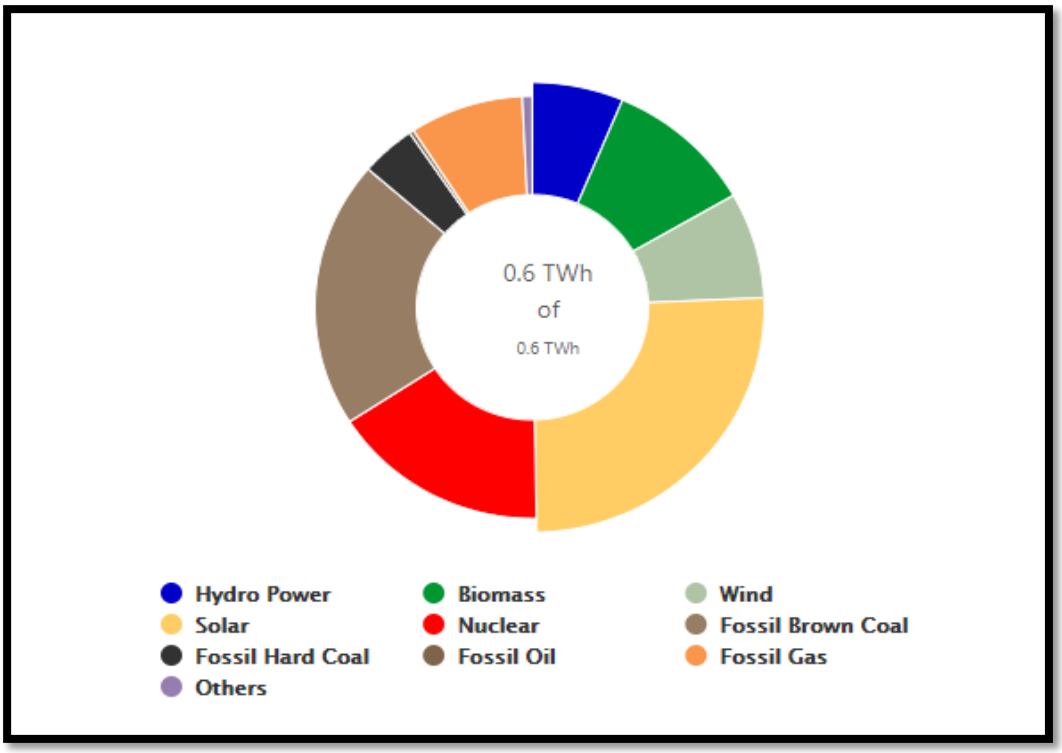

Fig 2: Net public electricity generation in Germany in a week

Source; Fraunhofer institute for solar energy systems ISE

2.2.2 Peer to peer trading

With smart and continual usage of RES, the number of solar panel users is increasing. Such users generating their own energy are called prosumers

who both consume and produce the energy. Sometimes, even after the required consumption of energy, some surplus amount is produced. This energy can go to waste if not handled properly. One option is to store energy in high power batteries. This option is reliable yet less practiced as the batteries cost way too much. A more beneficial and yielding alternative is peer-to-peer transactions via smart metering systems.

According to this solution, the surplus energy is transferred back to the central authority or grid. Prosumer produces its own energy in daytime when solar panels are functional and at night, energy from the distributing companies is consumed. The surplus sent to grid is net metered with the energy consumed during night hours and an electricity bill is generated. This electricity bill has negative or positive balance as per energy consumption. This entire process is controlled and monitored by NEPRA in Pakistan and is called net-metering [1]. Many people have benefited through this system so far.

With such advancements, people started to realize the need of a decentralized network that allows them to transact energy among their own neighborhood without involving any centralized third party or grids. During last decade, many solutions have been proposed using a microgrid.

2.2.3 Smart meter vs regular meter

A smart meter can be defined as an electricity meter that measures electricity units and displays them digitally. It provides the user with a precise and accurate energy consumption reading with regular intervals of thirty minutes or one hour.

A regular meter is the one in which unlike a smart meter, energy readings are noted manually and hence are less accurate. Such readings are taken every month or after 3 months. This reduces the reliability of metering

system.

These days, due to advancements in energy and regulatory sectors, a newer concept of two-way meters is introduced. Such meters came into existence after net-metering became a common practice.

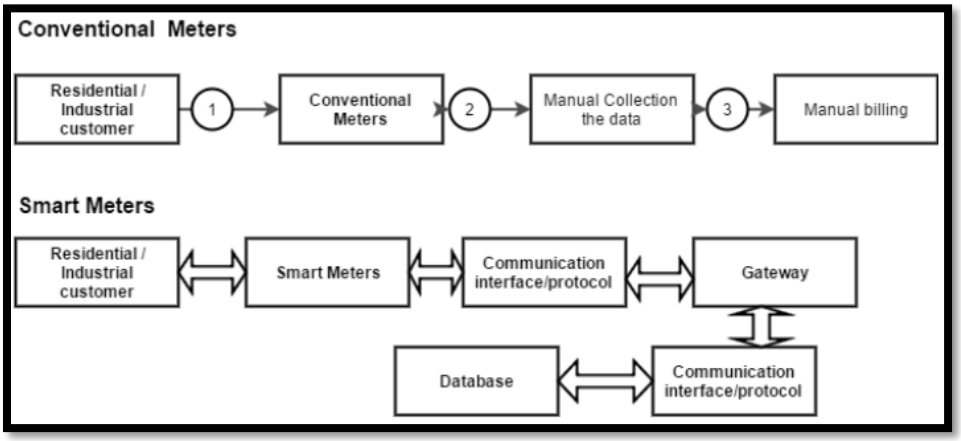

Fig 3: comparison between regular meters and smart energy meters

2.2.4 Micro grid

A microgrid is shrunk version of larger and conventional grids. It works for a certain small geographic area. A microgrid requires its own energy production resources such as solar panels. Generally, a microgrid is connected to a larger grid where the energy produced by microgrid is fed, or it is transferred to certain allocated consumption points using main grid's power lines. When the macro grid is non-functional due to repairing purposes, microgrid can work on its own. Such a process is called the Islanded mode.

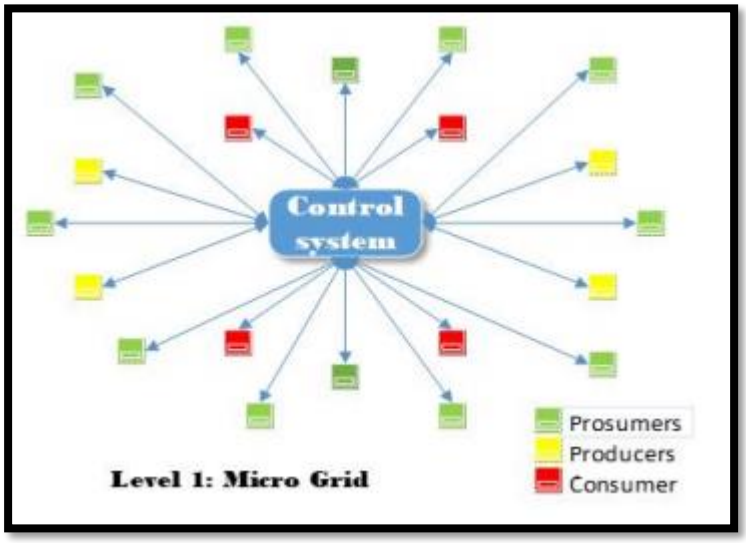

Fig 4: Micro grid with a central control system

2.3 SOLUTION

Keeping in view the advancements in digital world and energy sector, we have proposed a smart and secure solution to above mentioned problem. This not only encourages the people to shift towards the RES but also contributes to a modern and digitized world. The proposed metering system keeps a check of available energy units and cryptocurrency in a digital wallet. During a certain transaction, only the two trading parties know about the transactions and details ensuring a secure and transparent platform. After the transaction is held, a detailed receipt /history of transaction can be seen on wallet. The distributed energy network is a system that is adopted by the energy distribution companies to minimize the energy losses hence improving energy efficiency.

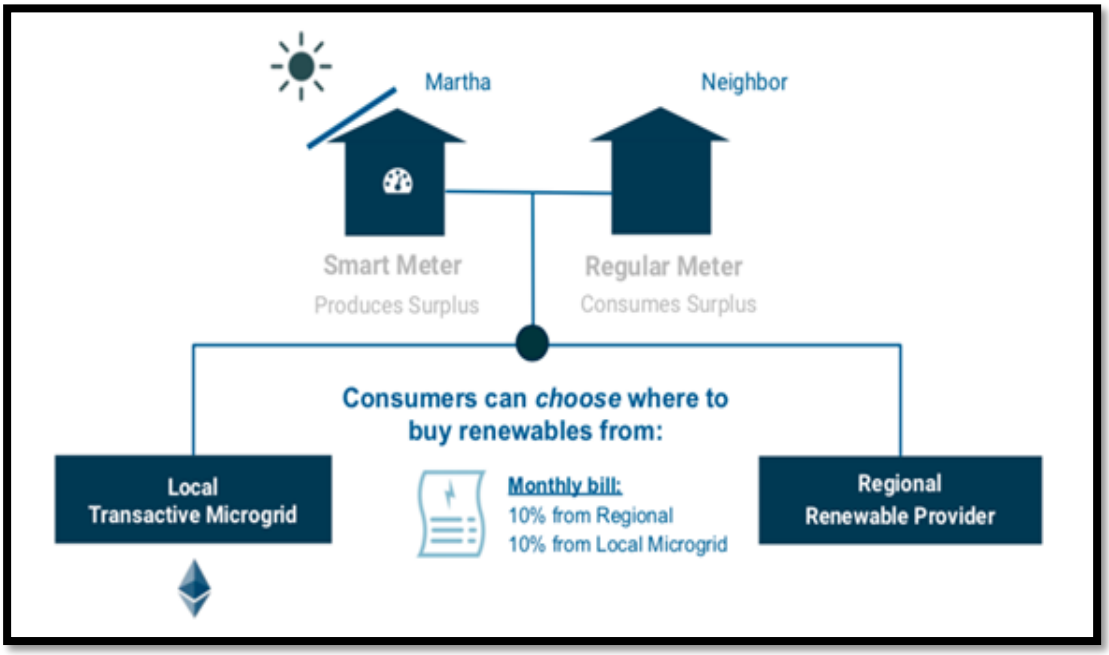

Fig 5: transaction of energy among local neighbors

2.4 PREVIOUS WORK

To find an appropriate solution to the above mentioned problem, we went through a lot of scholarly articles and research papers in addition to less mentioned blog posts and GitHub commits. This technology is not a traditionally practiced method in billing world hence not much of work was done previously. Although some of the developed countries as Germany, new Zealand and Australia have this technology being used to its full potential but it may take sometime for developing country like ours to deploy such efficient mechanisms. A blockchain based metering and billing system like this was proposed already which paved an easier path for our work. [2]

2.5 GOALS AND OBJECTIVES

A blockchain based secure metering system designed in this project that aims to fulfil following objectives,

1- Encourage the masses towards RES for energy production.
2- Provide a complete and secure solution for transactions.
3- Establish a blockchain network.
4- Establish a cryptocurrency (ether in this case)
5- Design a Power Grid
6- Interfacing PowerGrid with blockchain network

Chapter 03

METHODOLOGY

OVERVIEW

This chapter proceeds by explaining the complete methodology and steps followed to achieve the desired goal. All the steps involved are clearly explained in detail and the result achieved after every step is also mentioned. The first step is creating a microgrid. Then the transaction between peers takes place. The chapter proceeds by explaining the overall methodology through a block diagram.

3.1- CREATION OF MICROGRID

A microgrid is a basic level implementation on micro scale that allows large number of prosumers and consumers to carry out transactions with a smart control that monitors all the transactions. In this project only one prosumer and one consumer is created for the demo purpose. The produced and consumed number of energy units and analogous amount of cryptocurrency is monitored by the microcontroller. The blockchain network used in this project is 'Ethereum' and the cryptocurrency use is 'ether'.

3.2- DEPLOYMENT OF SMART CONTRACTS

A smart contract is deployed using remix (a solidity IDE). This provides a mutual platform for both the transacting parties with a certain set of terms and conditions. Smart contract is a transaction protocol which is automatically executed and controls / documents the events which are legally relevant. This step is discussed in further detail in upcoming chapters. This is the most basic and important part of our project.

3.3- TRANSACTION MECHANICS

In this project, a microgrid level prototype is created to give a better and clear insight of a blockchain based transactive energy system. This can be explained figuratively in a better way (refer to fig.). Such a power microgrid can be referred as a cyber-physical system. A network layer is created that is basically Ethereum (a private block chain system). Generally, a power micro grid consists of a source of energy (sun), solar panel, battery, microcontroller, load and a two way meter. These two-way meters are a common practice these days with the introduction of net-metering. The network layer consists of a complete Ethereum node

that would be run on laptop and four wallets, one for producer one consumer, and one for each regulator and owner.

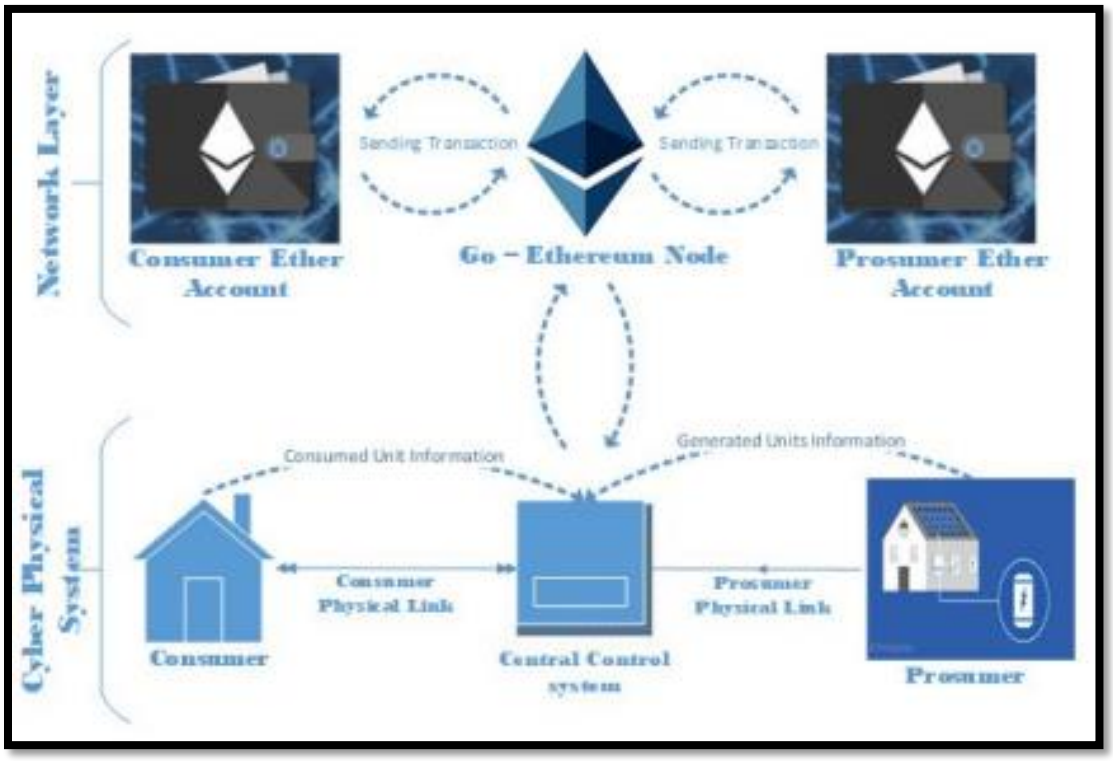

Fig 6: Blockchain based transactive energy system

Chapter 04

DETAILED DESIGN AND ARCHITECTURE

OVERVIEW

This chapter discusses in detail the design and architecture of the project. The project is divided into substages for ease of understanding. This is a four step process explained in detail. As this project is completely software based, this chapter explains all the software resources. Each software interface has its own uses, advantages and disadvantages that are explained below.

4.1 BREIF SUMMARY OF SOFTWARE DESIGN

Our software design is divided into 4 basic parts. All of these contributing to the final product.

1- Blockchain network (Ethereum)

2- Smart contracts

3- Transactive energy system

4- Testing

As a very initial task, main node of blockchain was set up. For this purpose, we used web3.js .

Once after the node is up and running, smart contracts were deployed using meta mask (crypto wallet).

The next step was using these smart contracts to run a transaction and get a transactive system running. Public nodes were accessed through public API using Infura.

All of the created smart contracts and information is deployed in a network via Ropsten. **Ropsten** is a proof-of-work testnet that most closely resembles the current Ethereum blockchain.

The above explanation gives a very brief gist of entire project which will be explained in detail later on in this report.

4.2 DETAILED SYSTEM DESIGN

The software methodology already mentioned in section 4.1 will be discussed in detail here. Following section will explain all the software resources used in detail.

4.2.1 Blockchain Network

4.2.1.1 web3.js

Web3.js is a collection of libraries that allows the user to interact with some local or remote Ethereum node using IPC, HTTP or WebSocket. It enables the user to employ the clients who interact with Ethereum blockchain. It aids the user to perform certain actions like transferring ether from one account to another, write and read the data from any smart contract, and most importantly it allows the user to create the smart contracts. Web3.js provides JavaScript bindings to Ethereum, which can then be used to build intuitive user interfaces using the web stack. (fig 5)

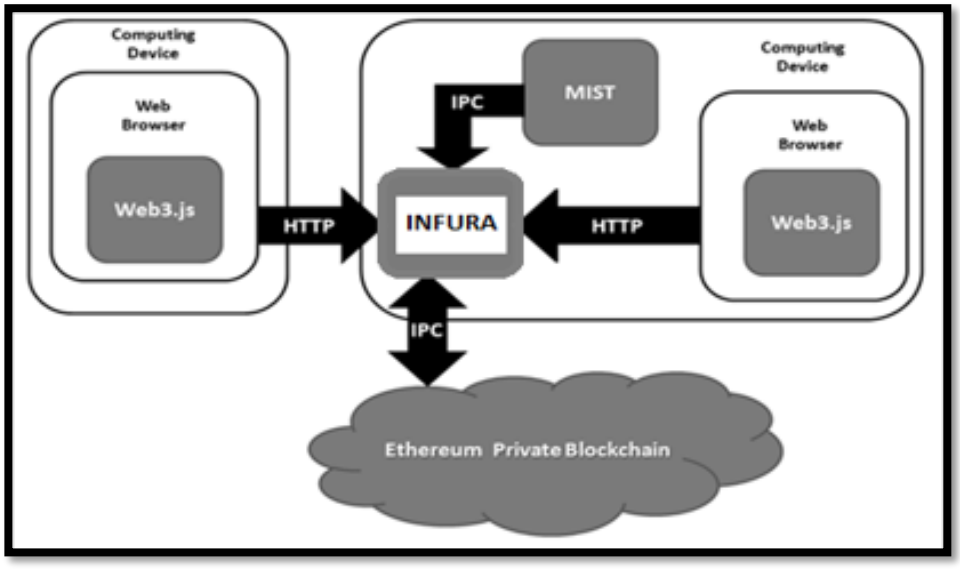

Fig 7: web3.js interacting with a private blockchain via INFURA.

4.2.2 Meta Mask

Meta Mask is a cryptocurrency wallet that interacts with the Ethereum network through software. Users can utilize a browser extension or a mobile app to access their Ethereum wallet, which can further be helpful in interacting with various other decentralized applications. **Meta Mask** is responsible for managing the Ethereum wallet, this wallet contains the user's Ethers (or money) and allows the user to send and receive Ethers through a DApp of interest.

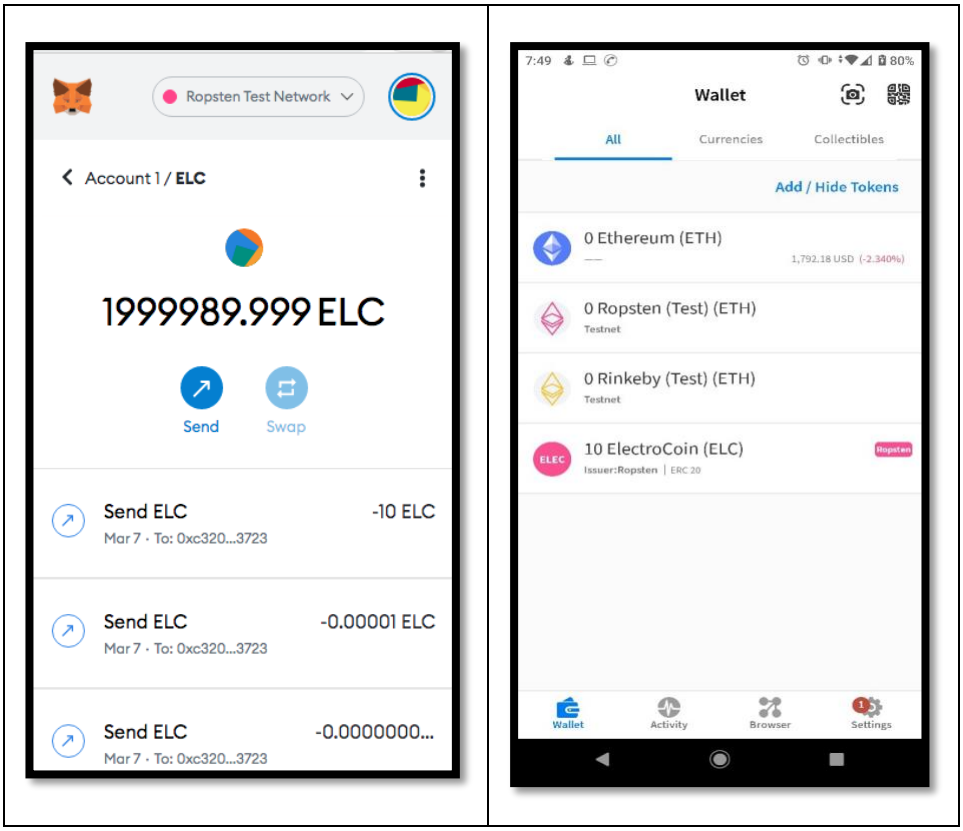

Fig 8: meta mask user wallets

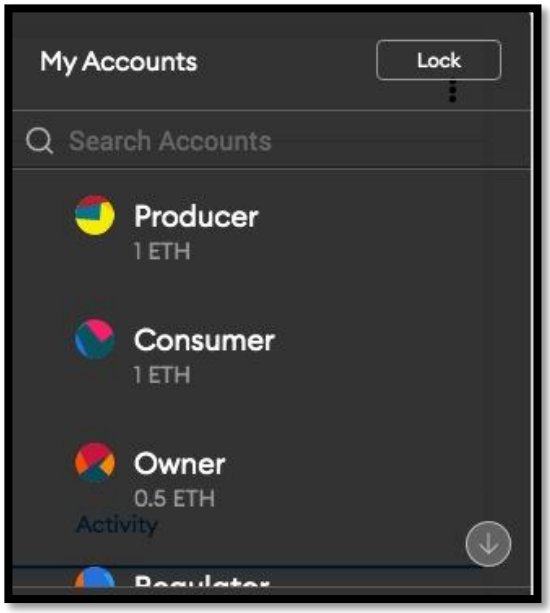

Fig 9: initial account balance before transaction

4.2.3 Infura

Infura is an Ethereum node cluster that allows the users to run an app without having to set up an Ethereum node or wallet of their own. Infura does not manage user's private keys for security reasons, thus it cannot sign transactions on user's behalf. With simple, dependable access to Ethereum and IPFS, Infura delivers the tools and infrastructure that allow developers to simply move their blockchain application from testing to scaled deployment. Infura is most preferred platform by the developer due to its user-friendly features such as instant availability (no syncing or complicated setups required.), IPFS and Ethereum interface.

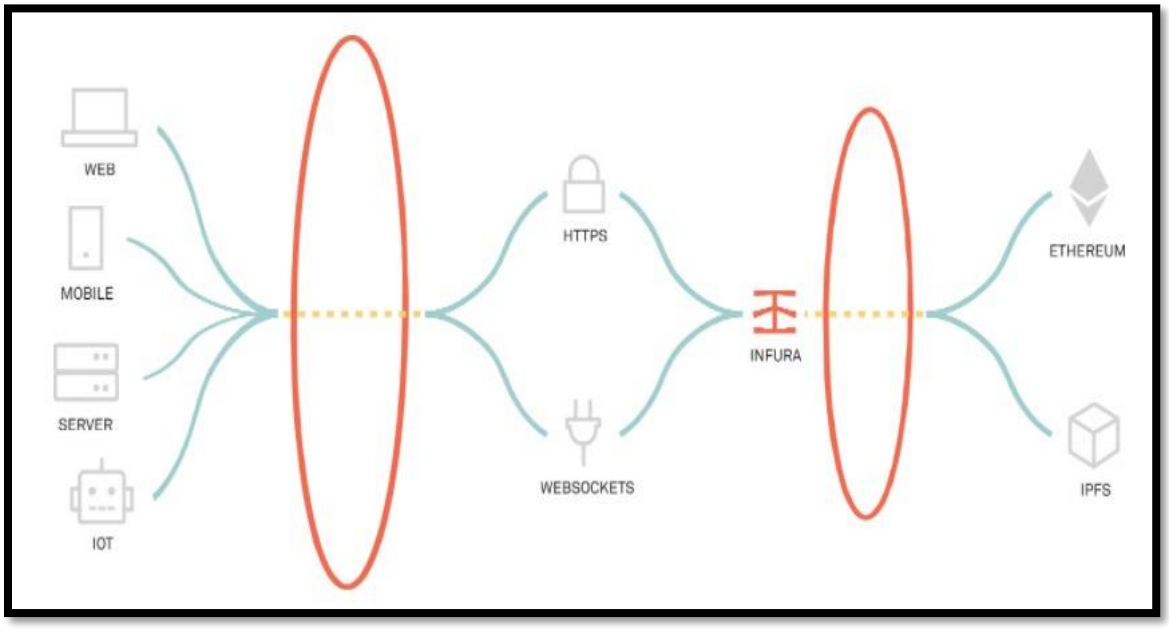

Fig 10: infura as a developer friendly platform.

4.2.4 Ropsten

Ropsten ETHs are **used** for testing purposes. When developers are building DApps, or experimenting on the network, to avoid losing money paying real ETH for transaction fees and smart contract deployments, it is better to use the **Ropsten** Network. Ropsten Ethereum, also known as "Ethereum Testnet", are as the name implies, a testing network that runs the same protocol as Ethereum does and is used to testing purposes before deploying on the main network (Mainnet).

4.2.5 Truffle And Ganache

Truffle is an Ethereum Blockchain Development Environment, Testing Framework, and Asset Pipeline. While Ganache is a personal Ethereum Blockchain for testing smart contracts. It allows the developer to deploy contracts, construct applications, run tests,

and execute other operations for free.

Using **Truffle**, a developer can introduce the Smart Contracts into web apps, and develop front end for decentralized apps. These days, Truffle is one of the most frequently used IDEs for Ethereum Blockchain.

4.2.6 Arduino IDE

An Arduino IDE or Arduino software has a text editor to write code, a text console that usually indicates possible errors or work progress, a toolbar with several commands and buttons that performs specific functions and offer a specific menu. It is usually interfaced with a hardware component called Arduino UNO. This is basically a microcontroller.

For our project we did not use Arduino UNO because only specific libraries from Arduino IDE were needed for integration with ESP32.

4.2.7 Remix IDE

It is an integrated development environment (IDE) used for writing, compiling and debugging of a certain solidity code. Remix provides a platform for deployment, development and administration of smart contracts for blockchains such as Ethereum. It requires solidity code. Where solidity is a programming language that is a high-level and contract oriented. It is a modernized version of various other programming languages such as C++, Python and Javascript.

4.2.8 ESP32 module

ESP32 module is the only hardware component we have used in our project so far. It is a microcontroller with an integrated Wi-Fi

module and a Bluetooth module. ESP32 can be interfaced with a larger system (laptop in our case) to provide for wi-fi and Bluetooth functionality with the help of SPI/SDIO and I2C / UART interface. It is a fairly cheap microcontroller considering its functionalities.

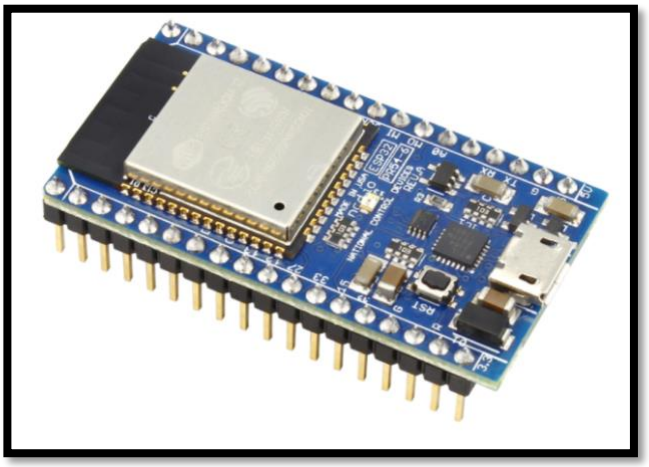

Fig 11: ESP32 wi-fi module

Chapter 05

IMPLEMENTATION OF SMART CONTRACT

OVERVIEW

In this chapter, we will go in detail of our suggested system for performing peer-to-peer energy transactions via blockchain, which eliminates the need for a central authority figure. This part will take readers through all of the stakeholders, their roles and duties, as well as the system's design and technical specifics. We begin by providing an overview of the problem and a detailed solution in section 5.1. Section 5.2 delves into the stakeholders, their obligations and rights, as well as the system's ecosystem. Then, in section 5.3, various technical specifics about the blockchain and other modules are discussed. The system's architecture, smart contract functionality, access control rights, microgrid, smart meters are all described in Section 5.4.

5.1 GENERAL DESCRIPTION OF PROBLEM AND SOLUTION

We have used Infura to implement the solution we propose for a peer-to-peer energy distribution system. Infura is an Ethereum client that assists in the setup of a fully functional Ethereum node for private systems. Infura provides Web3 tools and infrastructure to developers and businesses in a simple and dependable manner.

. In our system, we must address the following issues:

1. The entire system is controlled by a single central authority.
2. Payment mechanisms that are transparent to avoid demurrage and power hoarding.
3. The system's load balancing.

To resolve "i", we created a permissioned blockchain that is governed by a group of regulators. Section 5.3.1 outlines the duties and obligations of regulators. To ensure "ii," we would employ a micro-grid for energy storage after a producer advertises it, making it impossible for the producer to back out of the arrangement. Checks for maximum power storage in the grid at a certain time will be added for "iii."

5.2 STAKEHOLDERS AND ECOSYSTEM

This section outlines who the system's stakeholders are and what their roles are. The system's users include:

- Regulators
- Producers
- Consumers

5.2.1 regulators

Regulators are the entities that would aid in the system's stearing. Our system will have numerous regulators rather than simply one. Regulators' roles

include: • establishing market regulations.

- Assisting and overseeing the transactions
- Changing the number of producers and consumers.
- Adding new blocks to the network by mining.
- Smart contracts can be modified to improve the rules and trade methods.

The regulators are not in charge of facilitating commerce between parties or determining prices. They attempt to eliminate disagreements by automating the trade process.

5.2.2 producers

In our system, a producer is someone who generates electricity using their own solar panels. The producers would use the smart contract to make " availableEnergyAdvert" transactions.

5.2.3 consumers

Energy produced and advertised by producers can only be purchased by consumers through " RequestForEnergy" transactions.
These roles are mutually exclusive, meaning that a person can only be a member of one of them. For physical energy transfer, all of these entities are connected to the micro-grid. IInfura's entire nodes will be used by regulators to mine puposes. Consumers and producers do not require full nodes; instead, they will use the ESP32 protocol.

5.3 ENERGY TRANSACTION

Users should be able to make sell and buy transactions using their smart metres without having to deal with haggling or late transmissions as a result of the desired outcome. Certain components must be in place for this to happen. The system is described in full in this part, along with all of its needs.

5.3.1 Permissioned Blockchain and PoW

Instead of setting up an node on the personal computer we used 'Infura' an online service to which runs your virtual node and facilitates requests using an API. After signing up an api key is denoted to the user like this one 4f1102ebebfc4001990187303598a4e3 . User can send curl request to the infura node using this url. "https://mainnet.infura.io/v3/4f1102ebebfc4001990187303598a4e3". Infura can be connected to Ethereum mainnet, Ropsen, Rinkeby, Kovan, Gorli, Polygon Mainnet, and each net having different mainnet id and curl request url. "https://network.infura.io/v3/INFURA_KEY" Many each network works on different different mechanism like Rinkeby uses 'Proof of Authority' while many others uses 'Proof-of-Authority'. Proof-of-authority (PoA) is a consensus technique that relies on trusted and well-known validators to generate blocks and hence give computational power to a network. It uses a Byzantine Fault Tolerance (BFT) algorithm with identity as a stake to enable comparably speedier transactions.

We're using Proof-of-Work here with Ropsten network. Proof of work (PoW) is a type of zero-knowledge cryptographic proof in which one party (the prover) establishes to others (the verifiers) that a specified amount of computational effort has been invested. Following that, with no effort on their part, verifiers can authenticate this spend.

5.3.2 Smart Contracts

We need to move forward with electricity trading after our private Infura node is up and operating, mining is started, and nodes are connected to each other. We'll need smart contracts for this. A smart contract is similar to a legal contract in that it automatically executes when specific circumstances are satisfied. One of the parties must initially commit to the deal. The other parties then evaluate the offer and respond if they want to participate in the transaction. The terms and conditions are handled by the smart contract code.

The parties involved just need to provide a few specifications as part of their offer.

We created smart contracts with multiple functions for our system. Each function has its own set of access rights constraints. Role-based access rights have been incorporated in our system. The smart contract specifies three roles:

- Regulators
- Producers
- Consumers

5.4 ARCHITECTURE AND DATA FLOW

This section illustrates the data flow that occurs when energy trade on the blockchain begins. At this stage, we're assuming that anyone interested in becoming a Producer or a consumer will provide their blockchain private address, as well as their physical address, to one of the regulators. Regulators will then append the participant's key to their individual physical address for the purpose of power distribution. The following block diagram shows the hierarchy of users with the functions they can perform.

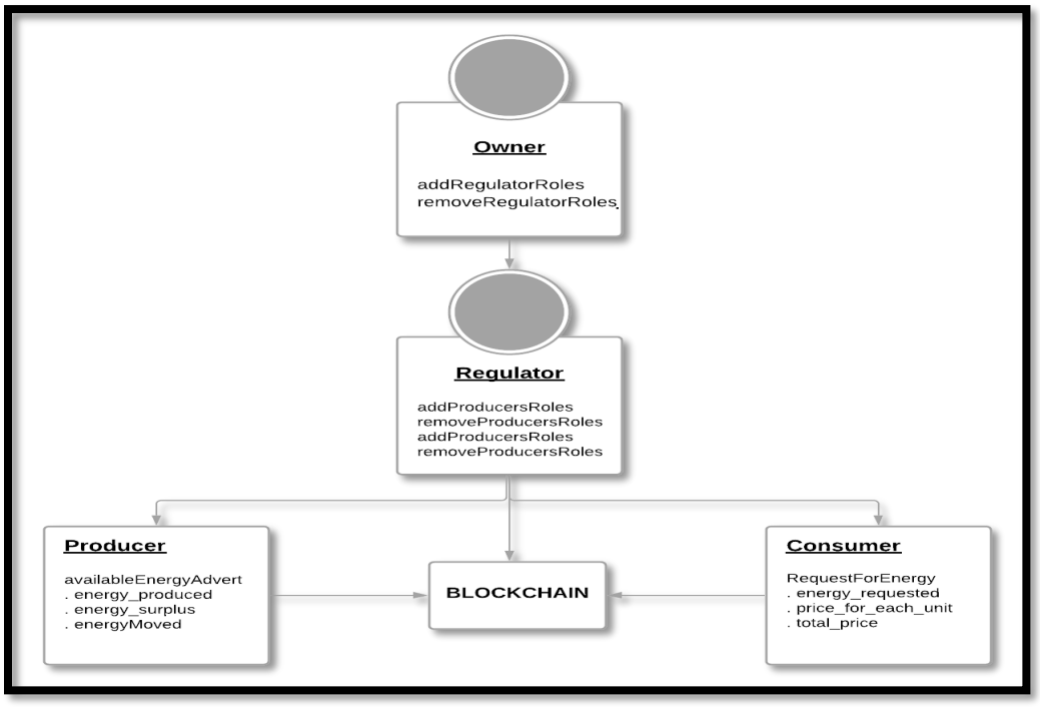

Fig 12: block diagram of users' function

5.4.1 Adding and Removing Users

Only the smart contract's owner, i.e. the address that deployed the smart contract over the blockchain, has the right to add and remove regulators. Any of the addresses mentioned in the regulator's role can add or remove producers and consumers. These addresses lists are dynamic and can be changed at any time. Prior to the commencement of trade, these roles do not need to be statically initialized.

5.4.2 Sell Advertisements

The contract's AvailableEnergyAdvert() function correlates to the it's SellAdvert transactions. This function is only available to the addresses assigned to the "Producers" role. This function accepts two arguments: the advertised energy units and the producer's suggested minimum price per unit.

AvailableEnergyAdvert() energy_produced proposedprice

The seller will send the advertised energy to the grid after invoking this function. As a result, if a buyer replies to this SellAdvert, the manufacturer will be unable to back out of the contract. This is how the problems of power abuse and dishonesty would be addressed.

Furthermore, the seller is required to pay a tax to the authorities based on the following formula:

tax = total_price - producer_share; *(i)*

where

*producer_share = energy_requested*producer_share_per_unit;* *(ii)*

producer_share_per_unit = 0.5 gwei

*total_price = energy_requested*price_for_each_unit* *(iii)*

while price_for_each_unit varies according to the no of requested units of energy

To assist the seller, we designed the contract so that the price per unit increases by ten percent of the seller's minimum price proposal every hour. A SellAdvert

has a three-hour lifespan after which the electricity is returned to the supplier. If the producer so desires, they can re-advertise those units.

5.4.3 Buy Offers

The smart contract's RequestForEnergy() function refers to the BuyOffer transactions. Only the addresses defined in the "Consumers" role have access to this function. The consumer chooses a producer from whom they want to acquire energy after looking through the transactions pool and viewing the live SellAdverts. This function requires four parameters: the quantity of energy units the customer wishes to purchase, the date SellAdvert was launched, the producer's minimum proposed price, and the producer's address.

RequestForEnergy() energy_requested adverttime proposedprice p_address

This function computes the price per unit based on the number of days since SellAdvert was introduced. This function can only be used by the consumer if their current balance is more than the amount due to the producer and the amount of tax due to regulators. The regulators would then allow electricity to be transferred from the microgrid to the consumer after the tokens were transferred.

5.4.4 Micro Grid

The microgrid would be physically connected to all of the members' consumers and producers. Energy trading would take place on a blockchain that is connected to all peers, but an electric connection would be made between the microgrid and the clients. Depending on the capacity of the grid, there is a limit on how much energy can be stored at any particular time.
We assumed that the grid's minimum storage capacity was 1 KWh and its maximum storage capacity was 1000 KWh at the time of development.
If the microgrid's storage capacity is already 990 KWh, the producers will not be able to place a SellAdvert. This would help to avoid grid storage being

overloaded for an extended period of time. The following flow chart shows the working mechanics of the system.

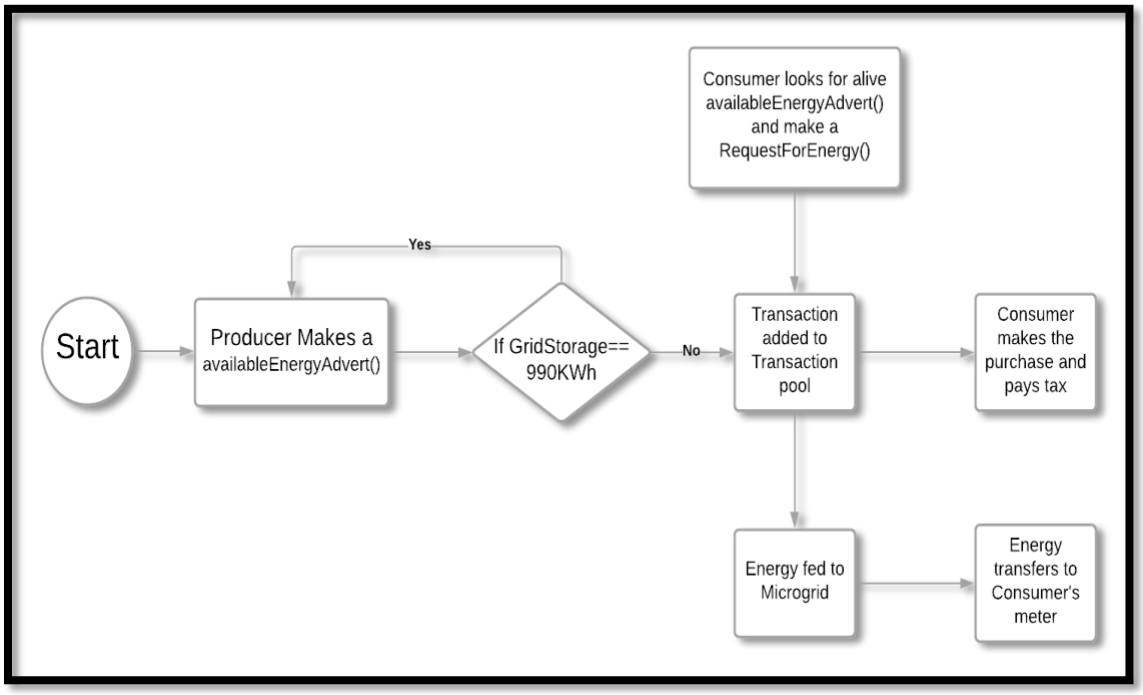

Fig 13: Flowchart of the Micro grid system

5.5 PAY-AS-YOU-GO PEER TO PEER PAYMENT

Another smart contract that allows micropayments has been built to allow the consumer to make repeated purchases. This agreement establishes a one-way payment channel between the consumer and the producer. There are three steps to it:

1. The customer finances and activates the smart contract, which includes the recipient's address, the contract's validity period, and the total money to be escrowed.
2. The consumer seals the messages with his private key and sends them to the producer after each purchase, indicating the contract's address, the total amount due up to this point, and his signature.
3. At the end of the transaction, the producer closes the payment channel by presenting the consumer's last signed message. The contract checks the signature and transfers the funds to the producer, with the remainder going back to the customer.

On the blockchain, only the transactions required for Steps 1 and 3 would be recorded. The rest of the communication will take place off-chain. It can be done through emails or through social media. A specific consumer's contract would only work between him and the identified producer. It is taken care of by the contract's access controls, and no other address can intervene.

When the producer believes the exchange is complete, they can control the channel. If a producer fails to terminate the channel before the validity period expires, the consumer can close the channel themselves, reclaiming any escrowed funds.

Chapter 06

DEPLOYMENT OF SMART CONTRACTS VIA ESP32

Overview

Nobody can deny that blockchain and the Internet of Things are complex technologies. However, due to the growing reputation of managed platform services, they are becoming easier to implement in recent years. The technical components of blockchain and IoT have become far less scary thanks to these managed services. While many of the technical obstacles have been overcome, there are still some significant obstacles to overcome. In our system, ESP32 module provides a bridge between Blockchain world and physical world.

6.1 ESP32 MODULE

ESP32 with it compact, simple and easy to program design has opened doors to new innovations with minimum effort and lesser external circuitry.

6.1.1 Hardware Specifications

The ESP32 is a line of low-cost, low-power system-on-a-chip microcontrollers that include built-in Wi-Fi and dual-mode Bluetooth. The ESP32 series contains built-in antenna switches, RF baluns, power amplifiers, low-noise receive amplifiers, filters, and power-management modules, as well as a Tensilica Xtensa LX6 CPU in dual-core and single-core versions.

6.1.2 Why ESP32

The ESP32 is a low-cost Wi-Fi module that is ideal for a variety of Internet of Things (IoT) projects. These modules provide GPIOs and support for a number of protocols, including SPI, I2C, UART, and others. They also come with wireless networks, which distinguishes them from other microcontrollers such as Arduino. That means you can quickly operate and monitor gadgets through the internet at a reasonable cost.

6.2 BLOCKCHAIN WITH IoT

The adoption of Blockchain in conjunction with IoT could be the answer to IoT challenges. Data is saved in a distributed ledger that is spread across multiple devices connected by peer-to-peer networks. No device can alter this data, hence it is unalterable. It will be easier to track billions of IoT devices using blockchain, which will allow for distributed processing and coordination. Blockchain's decentralized method will eliminate single points of failure, which is a problem with IoT's present centralized approach, the cloud. Cryptographic techniques used in blockchain help preserve private data generated by Iot devices.

6.3 EXECUTION OF PROPOSED SYSTEM

We need to move forward with electricity trading after our private Infura node is up and operating, mining is started, and nodes are connected to each other. With everything ready at its place, we are ready to perform our execution over ESP32. Each party involved has to perform its part of deal on an automatic system. This execution consists of the following steps:

6.3.1 Addition and Removal of Users

In the very start owner is the only user. This system follows a kind of hierarchy where only owner has the ability to add or remove a regulator and only regulator has the ability to add or remove a producer and/or a consumer. The owner has no authority over addition or removal of customer or producer. However owner gets a part of Customer-Producer trade as tax.

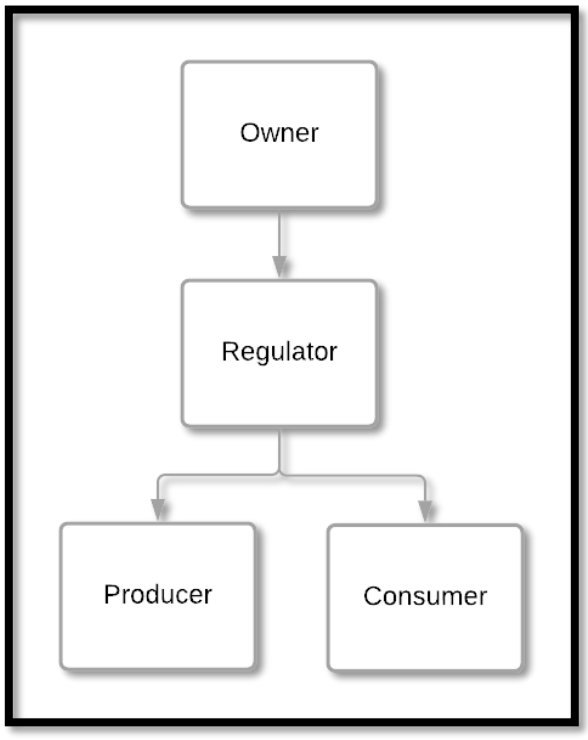

Fig 14: hierarchy of Users

Every transaction, either it is smart contract deployment, request for asserts or SellAdvert, comes with a cost. This cost is known as gas fee. Gas fee contribute to the Ethereum network's security. We prevent actors from spamming the network by requiring a price for each computation performed on it. Each transaction is required to specify a limit on how many computing steps of code execution it can consume in order to prevent unintentional or hostile endless loops or other computational waste in code. Prices are expressed in gwei, with 1 ETH equaling 1* 109 (1,000,000,000) gwei. A 21,000 gas transaction would cost 21,000 * 5 = 105,000 gwei at a gwei pricing of 5. (0.000105 ETH). The following figure shows a Ropsten Test Network window for generating a new contract. Here the Gas fee is 0.00271 gwei. The owner has to pay this amount to proceed with the transaction.

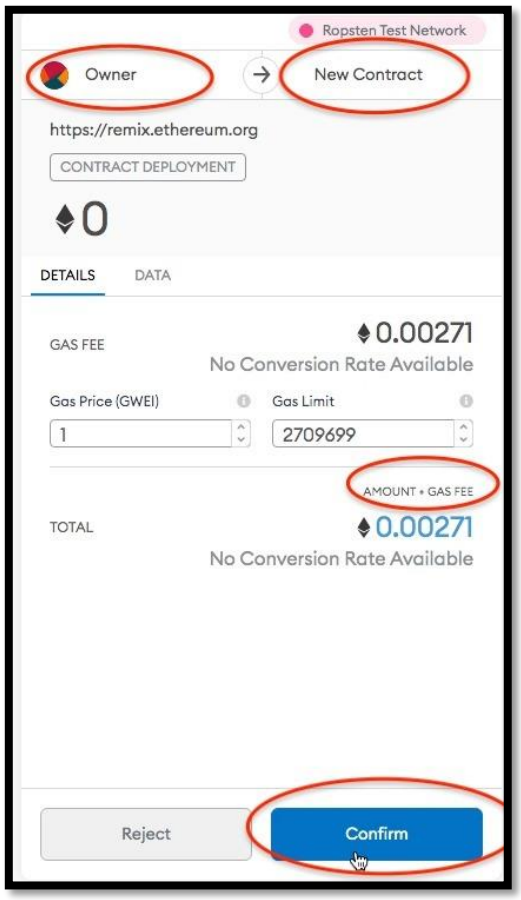

Fig 15: Ropsten Test network; new contract window

6.3.2 Deployment of smart contract

Once the addresses are assigned for producer, consumer and regulator, our smart contract is ready for deployment.

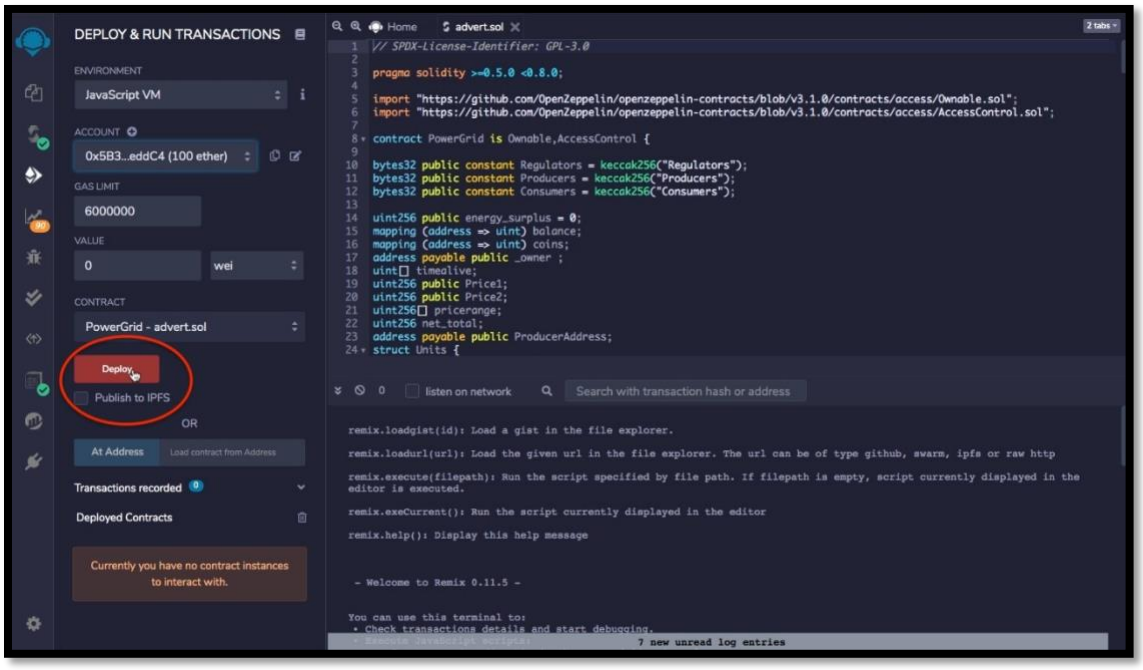

Fig 16: Remix: Deployment of smart contract

6.3.3. User hash

Once the smart contract is deployed, the producers and consumers come into action. For this particular instance, we have pre-coded values for transaction. Once the transactions are carried out we can see them in the form of hash functions (as seen in the serial monitor capture below, *Fig 17*).

6.4.4. Transfer Completion

Once the transaction is completed, the said amount is taken away from the Customer and transferred to the producer in exchange for the energy. Owner gets a part of Customer-Producer trade as tax (as described in equation iii). The following figure dhows the increase in the account balance of Producer and Owner while decrease in Consumer account balance after the deployment of

the smart contract. These are seen through Meta Mask Google Chrome browser extension.(*Fig 18*)

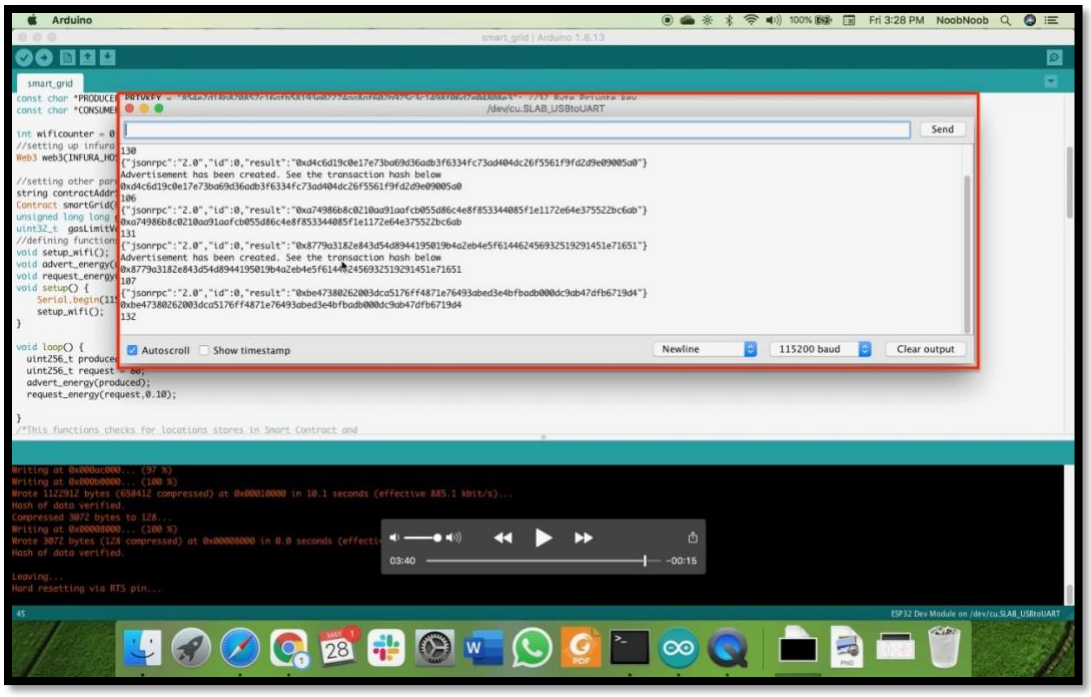

Fig 17: Hash Functions of Producer and Consumer

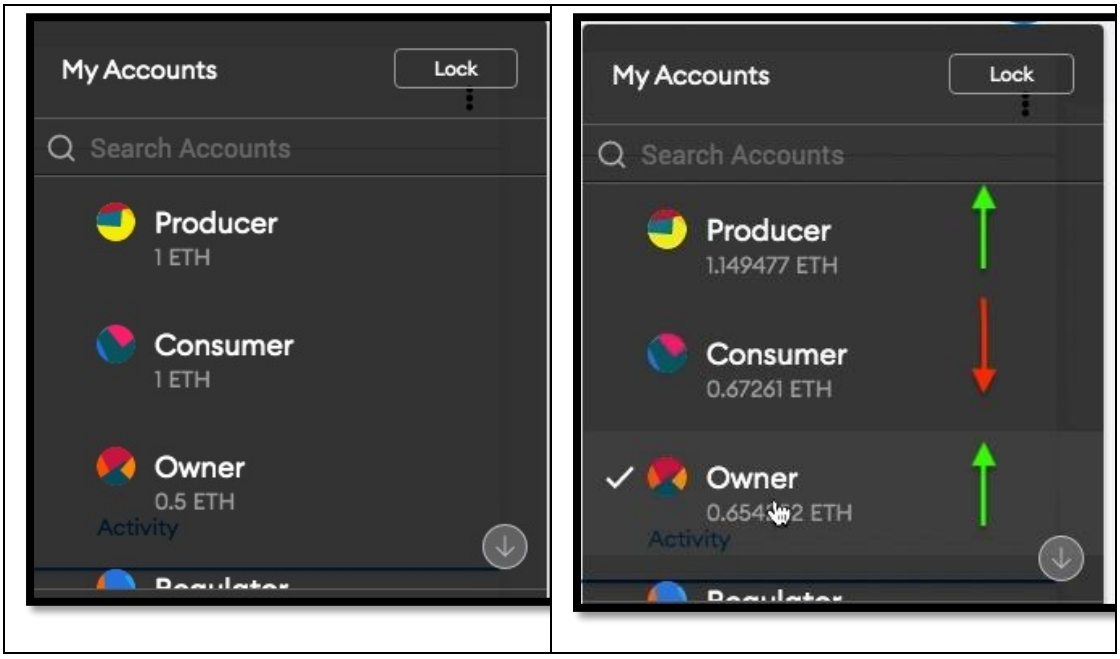

Fig 18: Change in Account balances after the deployment of the smart contract

6.4 Verification method

JSON RPC API

Each Ethereum node exposes a JSON RPC API which can be used to interact with blockchain. JSON-RPC is remote procedure call (RPC) protocol. It can be used over http. We can make http requests to remote Ethereum node for different function calls. A simple http request through curl for retrieving a block is as follows

```
curl -H "Content-Type: application/json" --data
"{\"jsonrpc\":\"2.0\",\"method\":\"eth_getBlockByNumber\",\"params\":
[\"latest\", true],\"id\":1}"
https://ropsten.infura.io/v3/828e910904b04c86b10cdbc01ed11dbf
```

In order to call different function, we change the value of "`method`". JSON RPC Documentation lists all the available methods and relevant examples. The interesting thing here is the "`params`" variable. The value of params is used to control different parameters. For example, in above case it is used to obtain "`latest`" block.

The example below calls a smart contract function.

```
curl -H "Content-Type: application/json" --data
"{\"jsonrpc\":\"2.0\",\"method\":\"eth_call\",\"params\":["{\"from\":
\"0xc320aB81E4cb3BC91DAaC417e3EB0db9Ae5A3723\",\"to\":\"0x82157c33a5B
6D4E34b72C13Cb913F5db21f9704B\",\"data\":\"0x8771e19a0000000000000000
00000000c320ab81e4cb3bc91daac417e3eb0db9ae5a372300000000000000000000000
0000000000000000000000000000000000000000000004d0000000000000000000000000
0000000000000000000000000000000000003f\"}",\"latest\"],\"id\":1}"
https://ropsten.infura.io/v3/828e910904b04c86b10cdbc01ed11dbf
```

In this case `params` contains another JSON object inside it. The role of each pair inside `params` is described below:

`to`: The address of smart contract

`from`: the address which is calling the function. In simple calls which don't change the state of blockchain it is optional.

`data`: Data field is most important field. It contains information about the function name and function arguments. This information is encoded using Contract ABI Specification.

Contract ABI Specification

Suppose we want to call the following function which is defined inside a smart contract

```
function baz(uint32 x, bool y) public pure returns (bool r) { r = x > 32 ||
y; }
```

The data field will be obtained as follows

- **0xcdcd77c0**: The Method ID. This is derived as the first 4 bytes of the Keccak hash of the ASCII form of the signature baz(uint32,bool).
- **0x0000000000000000000000000000000000000000000000000000000000000045**: the first parameter, a uint32 value 69 padded to 32 bytes
- **0x0000000000000000000000000000000000000000000000000000000000000001**: the second parameter - boolean true, padded to 32 bytes

In this way we can call different smart contract functions.

However, things are different when we want to call functions which modify the state of blockchain or in simply which change values of variable. Instead of `eth_call` method we now need `eth_sendRawTransaction`. This method takes signed transaction hash as value inside `params` and broadcasts it to the network.

Transaction Signatures

In order to sign a transaction, we need to generate transaction data. The transaction data comprises of the following entries

nonce: It is an integer value which serves as a counter. It keeps track of number of transactions sent from an address.

gasPrice: Gas is the fee required to perform a transaction on the network. gasPrice tells the amount the sender is willing to pay for the transaction.

gasLimit: It is maximum amount of gas the sender is willing to pay.

to: The address where transaction is being sent.

from: The address of sender

value: Hexadecimal value. It is needed when we want to send eth to an address.

data: the data field is calculated using the process explained in above section.

This data is encoded using RLP encoding which is standard in Ethereum which is signed using private key and sent to the network using `eth_sendRawTransaction`. This function returns the transaction hash.

Generating Transaction Signatures

The signing and signature verification process in Ethereum uses Elliptic Curve Cryptography. The signature is generated by following equation

$$Sig = F_{sig}(F_{keccak256}(m), k)$$

Here

F_{sig} is signing algorithm

m is RLP encoded transaction data

$F_{keccak256}$ is the keccak256 hash function

k is the private key

Sig is the resulting signature. The resulting signature has two pars r and s.

$$Sig = (r, s)$$

The process of generating signature works as follows

- An ephemeral private key ***q*** is generated so that actual private key is protected. This is done to ensure that actual private key cannot be calculated by attackers.
- From q ephemeral public key Q is generated using G and q. r value is then the x coordinate of Q. From there s value is calculated as follows

$$s \equiv q^{-1}(Keccak256(m) + r * k) \ (mod\ p)$$

Verifying Transaction Signatures

In order to verify the signature, we need r and s, serialized transaction data and public key corresponding to the private key that signed the transaction. The process of verification works as follows

- Check all inputs are valid
- Calculate $w = s^{-1} mod\ p$
- Calculate $u_1 = Keccak256(m) * w\ mod\ p$
- Calculate $u_2 = r * w\ mod\ p$
- Finally calculate the point on the elliptic curve $Q = u_1 * G + u_2 * K\ (mod\ p)$
- If x coordinate of point Q is equal to r we can conclude that the transaction was signed by authorized person.

Here
p is prime order of elliptic curve
K is the signer's public key
G is the elliptic curve generator point
m is the transaction data that was signed.

ESP32

Esp32 is a low-cost microcontroller with integrated Wi-Fi and dual mode Bluetooth. Esp32 is available at much lower cost than a Raspberry Pi. Most of the work done so far used Raspberry Pi. Very little work is done on Esp32. There are few libraries available for interfacing blockchain with Esp32 and they also lack proper documentation. In this work we have this library. It had few bugs which were fixed. The entire process explained above – making http request, signing transactions, sending transaction- is handled by the library under the hood. For instance, the following code snippet calls a smart contract function(checkLocation) using esp32:

```
void checkLocation(uint256_t latitude, uint256_t longitude,const
char *address)
{
  string contractAddrStr = LOCATIONCHECK;
  Contract contract(&web3, LOCATIONCHECK);
  string addr = address;
  string param =
contract.SetupContractData("checkLocation(address,uint256,uint256)"
, &addr,latitude,longitude);
  string result = contract.ViewCall(&param);
  Serial.println(result.c_str()); //prints JSON RPC Response
  int permission = web3.getInt(&result); //Obtaining integer result
i.e either 1 or 0
  Serial.println(permission); //displaying result on screen
}
```

The code below calls a smart contract function which modifies a value on blockchain.

```
void changeOwner(const char *newOwner)
{
  string contractAddrStr = LOCATIONCHECK;
  Contract contract(&web3, LOCATIONCHECK);
  contract.SetPrivateKey(PRIVATE_KEY);
  string addr = MY_ADDRESS;
  unsigned long long gasPriceVal = 24000000000ULL;
  uint32_t  gasLimitVal = 3000000;
  uint32_t nonceVal = (uint32_t)web3.EthGetTransactionCount(&addr);
  Serial.println(nonceVal);
  uint256_t valueStr = "0x00";
```

```
  string transferAddr = newOwner;
  string p =
contract.SetupContractData("transferToOriginPCustoms(address)",
&transferAddr);

  string result = contract.SendTransaction(nonceVal, gasPriceVal,
gasLimitVal, &contractAddrStr, &valueStr, &p);
  Serial.println(result.c_str());
  string transactionHash = web3.getString(&result);
  Serial.println(transactionHash.c_str());
}
```

The following function in the sketch uploaded to Esp32 takes all the relevant data and sends the transaction to Ethereum network.

```
string result = contract.SendTransaction(nonceVal, gasPriceVal,
gasLimitVal, &contractAddrStr, &valueStr, &p);
```

Inside the source code for the function SendTransaction, first a signature is generated as shown

```
string Contract::SendTransaction(uint32_t nonceVal, unsigned long
long gasPriceVal, uint32_t gasLimitVal,
                                 string *toStr, uint256_t
*valueStr, string *dataStr)
{
    uint8_t signature[SIGNATURE_LENGTH];
    memset(signature, 0, SIGNATURE_LENGTH);
    int recid[1] = {0};
    GenerateSignature(signature, recid, nonceVal, gasPriceVal,
gasLimitVal,
                      toStr, valueStr, dataStr);

    vector<uint8_t> param = RlpEncodeForRawTransaction(nonceVal,
gasPriceVal, gasLimitVal,
                                                       toStr,
valueStr, dataStr,
                                                       signature,
recid[0]);

    string paramStr = Util::VectorToString(&param);
    return web3->EthSendSignedTransaction(&paramStr, param.size());
}
```

Currently the issue with this library is that it requires private key in plain text. This problem needs to be fixed in later works.

Chapter 07

EVALUATION OF PROPOSED SOLUTION

Chapter 08

CONCLUSION AND FUTURE WORK

Overview

This is final chapter of this report that sums up the entire project in minimum amount of words. It gives a complete overview of the achieved goals. The chapter grows further by mentioning the possible future works of the project and its scope in coming years. It discusses the future of blockchain in power sector and its impact on pre-existing technologies.

8.1 CONCLUSION

Electricity is the most essential need in our daily lives in today's world. We require energy to charge our cell phones and laptop computers, without which we would be completely shut off from the rest of the world. Electricity is required to power hospital machines, air conditioning, factories, and other facilities. Because of the ever-increasing need for electricity, natural resources that can be used to generate energy are in constant demand. After almost a century of utilizing fossil fuels to generate electricity, humanity has now begun to recognize the massive devastation that fossil fuel use has wreaked on the planet.

Renewable energy, unlike fossil fuels, can be used at the individual level, even if authorities are falling behind in their grid set-ups.

Trading, on the other hand, is a major issue in peer-to-peer scenarios. Fortunately, we can use blockchain to create a distributed trading system.

This project is a comprehensive implementation of the block chain-based smart energy system. The Ethereum network has produced a thorough prototype consisting of a single consumer and a single producer with a control system and a detailed block chain. This can easily be used for multiple producers and consumers. The ether cryptocurrency is used to pay for the energy units purchased by the consumer from the producer.

After a successful ether transaction on the Ethereum private network, the system sends a signal to the controller of the electricity grid where both the consumer and the producer are physically linked. When the controller receives a signal indicating that the transaction was completed and the number of units to be transferred, it opens the consumer load's physical connection.

The units consumed are continuously measured by the controller. After the energy units have been successfully transferred, the controller will turn on the consumer link.

8.2 LIMITATIONS

1)**Problem:** ESP32 cannot run an entire node on its own. To send transaction to

network we have to become one of the full node.

Solution: Send and receive transactions through an API. Which is infura in this case.

2) **Problem:** If we use an API like infura, we need to have to share our private key for the node to sign transaction and send it ahead.

Solution: Instead of sharing private key with infura over http request. We sign the transaction ourselves. It means that the signature provides the authentication that the transaction is send by the owner this particular public address without actually having the private key

3) **Problem:** Geth is more secure option since new we can use it to run a private network but does not allow any RPC calls on the node because if geth do allow such RPC calls, user can even read all the private keys. This can be counted as a breach of privacy.

Solution: To solve such a problem, Infura was used instead of geth

8.3 FUTURE WORK

Future development on this subject could go in a number of different areas.

- Bidding for the energy units that a seller has offered. By adjusting prices in accordance with market changes, all parties concerned can gain more benefits.
- Infura's web3.js addon allows for automatic transactions. This would aid in the automatic sale and purchase of units by both prosumer and consumer.

To complete the end-to-end trade, the electric meter and solar panel must currently be merged with the system. To better optimize the system, a script and accompanying database can be used to track transactions from the transaction pool from the geth console.

REFERENCES:

World wide web

[1] how net metering works in Pakistan, **https://zerocarbon.com.pk/how-net-metering-works-in-pakistan/#:~:text=Net%20metering%20Pakistan%20is%20an,hours%20or%20at%20times%20when**

Research Papers

[2] A. Onder Gur, S. Oksuzer and E. Karaarslan, 'Blockchain Based Metering and Billing System Proposal with Privacy Protection for the Electric Network', 2019 7th International Istanbul Smart Grids and Cities Congress and Fait (ICSG) IV page 206.

[3] Kvaternik, K., Laszka, A., Walker, M., Schmidt, D., Sturm, M., lehofer, M., & Dubey, A. (2017). *Privacy-Preserving Platform for Transactive Energy Systems*. *c*. http://arxiv.org/abs/1709.09597

[4] Rahman, M. A., Rashid, M. M., Shamim Hossain, M., Hassanain, E., Alhamid, M. F., & Guizani, M. (2019). Blockchain and IoT-Based Cognitive Edge Framework for Sharing Economy Services in a Smart City. *IEEE Access*, *7*, 18611–18621. https://doi.org/10.1109/ACCESS.2019.2896065

[5] Xu, Q., He, Z., Li, Z., & Xiao, M. (2019). Building an Ethereum-Based Decentralized Smart Home System. *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, *2018-December*, 1004–1009. https://doi.org/10.1109/PADSW.2018.8644880

[6] Zhang, C., Wu, J., Zhou, Y., Cheng, M., & Long, C. (2018). Peer-to-Peer energy trading in a Microgrid. *Applied Energy*, *220*(June), 1–12. https://doi.org/10.1016/j.apenergy.2018.03.010