**Implementation of Blockchain in IoT**

By

**Ritwik Ranjan Pathak**

Contact\_details: {

Email : [ritwikranjan26@gmail.com](mailto:ritwikranjan26@gmail.com)

Phone Number: 917488744264

}

**TABLE OF CONTENTS**

1. Abstract
2. List of Figures
3. List of Tables
4. Introduction
5. Theoretical Background
6. Literature Review & Project Objectives
7. Work Done
8. Results and Discussion
9. Conclusion and Future Scope of Work
10. Reference

**ABSTRACT**

The Internet of Things (IoT) refers to a network of interconnected smart devices that enable communication and data exchange. Its significance in society stems from its wide-ranging applications in everyday life, such as in smart homes and automobile sensors. However, one of the major drawbacks of IoT is its centralized network structure, where data passes through multiple nodes, making it susceptible to attacks that compromise these nodes. This vulnerability raises concerns regarding the integrity and trustworthiness of the information transmitted within a centralized network.

To address this challenge and ensure the continuous availability of the IoT network, our project aims to create a secure environment that prevents attacks on IoT devices, thereby preserving the network's availability. The proposed solution involves implementing blockchain technology into IoT applications, which brings about improvements in security. By leveraging blockchain's inherent characteristics, we can establish a tamper-proof infrastructure for IoT.

**LIST OF FIGURES**

###### Fig 1 : Pin Out Diagram of Raspberry Pi 3

###### Fig 2 : Setup For Temperature and Humidity Reading

Fig 3.1 : Code of creating blockchain, generating hash and validating block

Fig 3.2 : Connecting to the server

Fig 3.3 : Code of get and post request in server

Fig 4.1 : Starting menu

Fig 4.2 : Menu of Data Entry

Fig 5 : Output Result

**INTRODUCTION**

In order to practically demonstrate the feasibility of our proposed solution, we will utilize Raspberry Pi to simulate an IoT network. By incorporating blockchain technology into this network, we aim to synchronize data from Raspberry Pi 3.0 and various sensors in a secure and tamper-proof manner by leveraging the blockchain ledger.

The fundamental concept of blockchain revolves around a distributed ledger comprised of interconnected blocks, which are securely linked together using cryptographic hashes. Each block in the chain contains crucial information, including a cryptographic hash of the previous block, a timestamp, and transaction data. To implement this blockchain structure, we will utilize the Python 3 programming language, which offers a robust foundation for building such systems.

Our focus will be on developing a blockchain system that efficiently handles the data received from sensors. This system will securely store sensitive information within hashed blocks, ensuring its protection against malicious attacks. By continuously adding blocks to the chain, we will progressively enhance the security of the IoT network, reinforcing its resistance to potential threats.

Through this approach, we aim to create a tamper-proof environment for IoT data transmission, thereby bolstering the overall security and reliability of the network.

**THEORETICAL BACKGROUND**

**Blockchain is Immutable and Transparent:**

Once a transaction is recorded on the blockchain, it is nearly impossible to alter or delete it. This immutability provides a high level of security and trust in the system. Additionally, blockchain's transparent nature allows all participants to inspect the transaction history, enhancing accountability and reducing fraud.

The DHT11 is a low-cost IoT sensor used for measuring temperature and humidity. It utilizes a resistive humidity sensor and a thermistor. An integrated circuit handles signal acquisition and communication with the microcontroller. The sensor employs a timing-sensitive single-wire digital communication protocol, transmitting data in a fixed 40-bit format that includes temperature and humidity values along with a checksum. The DHT11 operates within a voltage range of 3.3V to 5V and requires a pull-up resistor for proper signal detection.

IoT's theoretical background includes key elements such as connectivity, sensors and actuators, data processing, interoperability, security, scalability, and applications. IoT relies on wireless communication, utilizes sensors for data collection, processes data using ML and AI, ensures interoperability through standardization, prioritizes security and privacy, and focuses on scalability and manageability. IoT applications span diverse domains, enhancing efficiency, decision-making, automation, and creating new services and business models.

**LITERATURE REVIEW**

We have used these resources to study the concepts and taken inspiration from theses resources and implemented them to do our project.

**1] Poster Abstract: Blockchain-based Scalable Authentication for IoT** by Munkenyi Mukhandi, Eduardo Andrade, Francisco Damião, Jorge Granjal and João P. Vilela

Link: <https://dl.acm.org/doi/10.1145/3384419.3430465>

**2] Mastering Bitcoin** by Andreas M. Antonopoulos

The book goes into depth to explain the technology behind bitcoin, the purpose of the virtual currency, and its applications in the real world. We got to learn how the cryptocurrency works, how bitcoin transactions are processed, and how the underlying technology operates.

**PROJECT OBJECTIVES**

**Implementation of IoT using Raspberry Pi 3.0**

The objective of our project is to implement the Internet of Things (IoT) using Raspberry Pi 3.0. Raspberry Pi 3.0 is a versatile and affordable single-board computer that provides the necessary capabilities for connecting and controlling various IoT devices.

**Implementation of Blockchain**

Implementing blockchain involves setting up a distributed ledger system that ensures secure and transparent transactions. The process entails creating a network of nodes that collectively maintain and validate the blockchain. Each node stores a copy of the entire blockchain, and transactions are added to the blockchain through a consensus mechanism. The blockchain utilizes cryptographic techniques to secure and verify the integrity of data. With its decentralized nature and immutability, blockchain implementation offers enhanced security, transparency, and trust in various industries such as finance, supply chain, healthcare, and more.

**Implementation of Blockchain in IoT using Raspberry Pi 3.0**

The objective of our project is to implement blockchain technology in IoT using Raspberry Pi 3.0. By combining these two technologies, we aim to enhance the security and trustworthiness of IoT networks. Raspberry Pi 3.0 will serve as the IoT device, while blockchain will provide a decentralized and tamper-proof ledger for recording and validating IoT transactions

**WORK DONE**

**Implementing IoT using Raspberry Pi 3**

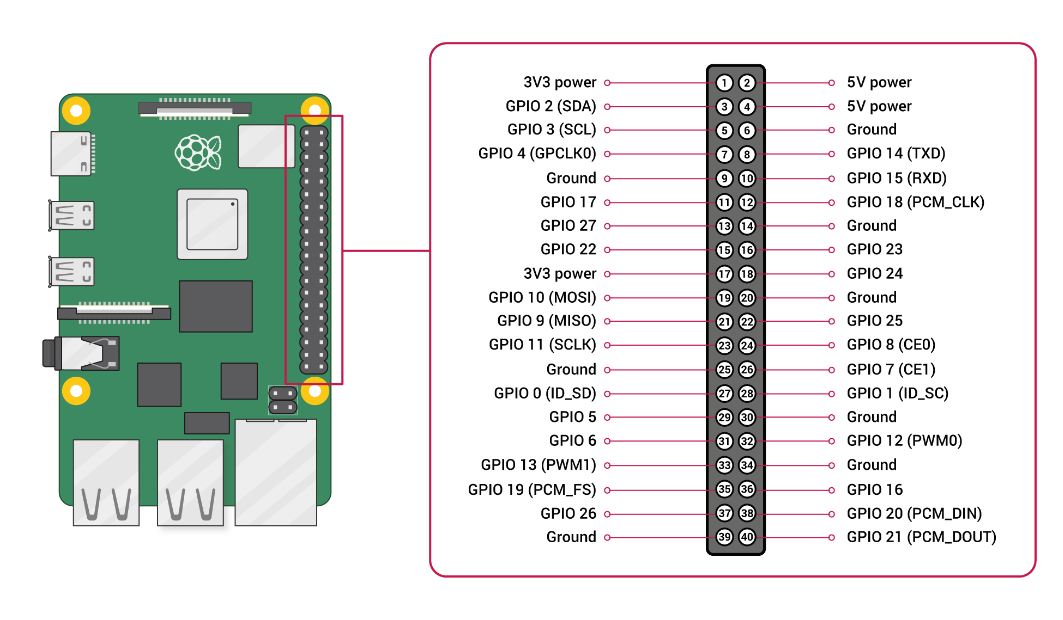
Implementing IoT using Raspberry Pi 3 is a great way to create simple, low-cost, and efficient IoT solutions. Here are the general steps we followed:

**Learning about Raspberry Pi 3 board and its Components**: The Raspberry Pi 3 is equipped with a quad-core 64-bit Broadcom BCM2837 ARM Cortex-A53 SoC processor running at 1.2 GHz, making it about 50% more powerful than the Pi 2.

**1]Installing an Operating System:** We installed an operating system on your Raspberry Pi 4. ‘Raspberry Pi OS’. It is a free operating system based on Debian Linux and comes pre-installed with Python and other software.

**2]Installing IoT software**: There are various IoT software frameworks available for Raspberry Pi 4, including MQTT, Node-RED, and Python. We are using Python to connect to IoT devices.

**3]Connecting IoT devices**: The devices like sensors, cameras, actuators, or any other type of IoT device can be easily connected to the board.



***Figure 1.*** ***Pin Out Diagram of Raspberry Pi 3***

We can use sensors with Raspberry Pi by following these steps-

**1] Connecting the sensor to the Raspberry Pi's GPIO pins.** The GPIO pins are used to connect external devices, such as sensors and actuators, to the Raspberry Pi.

**2] Installing the necessary libraries:** Depending on the type of sensor we need to install additional libraries to read data from the sensor.

**3] Writing code to read sensor data**: We will write code in a Python to read data from the sensor connected to the Raspberry Pi.

**4] Testing the sensor:** The sensor will be tested by running the code.

**Implementation of Blockchain in IoT using Raspberry Pi** .

**1] Realization of exact mechanism of our model**: At core level, this code creates a basic implementation of a blockchain using Python. The blockchain is made up of blocks, where each block contains a set of data, a cryptographic hash of the previous block, and a proof of work that is difficult to calculate but easy to verify. We are using **sha256** hashing algorithm for creating the hash. The first block called the **‘Genesis Block ‘**is made automatically and it has no previous hash. Subsequent blocks are manually created through the mining process. The hash of the previous block is included in the current block, creating a linkage between blocks.

**2] Identification of method for Blockchain Initialization**: We have designed and implemented a Blockchain class using the **Thonny Python IDE**. The class initializes with an empty chain. The first block, known as the Genesis Block, is automatically created and added to the chain. It contains the data "genesis block", a proof value of 1, and a previous hash of "0". The Genesis Block serves as the foundation of our blockchain.

**3] Identification of method for updating the Blockchain Ledger**: In this implementation, the ledger (i.e., the blockchain) is updated by adding new blocks to the chain. To add a new block to the chain, the **mine\_block()** method is called with the data that should be included in the block. This method uses a proof-of-work algorithm to calculate a valid proof for the new block, based on the previous block's proof, the current index, and the new block's data.

**a] Computation of proof of work**: The **proof\_of\_work()** method performs the proof-of-work algorithm by finding a proof that satisfies a specific condition (in this case, a hash starting with four zeros). It starts with a new proof value of 1 and increments it until the condition is met.

**b] Validation of the proof of work**: If the proof of work included in the block satisfies the difficulty level required by the blockchain network. This ensures that a sufficient amount of computational work has been performed to add the block to the chain.

Once a valid proof has been calculated, the mine\_block() method creates a new block dictionary with the relevant information, including the new proof, the previous block's hash, the current index, and the provided data. This new block is then appended to the end of the chain, effectively updating the ledger.

**4] Identification of method for approving the transactions**: The **is\_chain\_valid()** method is used that checks the integrity of the chain by iterating over all blocks in the chain, verifying that the previous hash of the current block matches the hash of its previous block, and that the hash of the current block meets the difficulty level of the proof of work. If any block fails these checks, the method returns False, indicating that the chain is invalid. If all blocks pass the checks, the method returns True, indicating that the chain is valid. In this way if someone wants to tamper the blockchain requires the recalculation of all subsequent block hashes, which is computationally infeasible and therefore ensures the integrity of the blockchain.

The above work forms the core implementation of our blockchain model. It allows for the creation of new blocks, linking them to the previous blocks using cryptographic hashes, and verifying the integrity of the entire chain through proof of work and hash matching. The implementation lays the foundation for a secure and tamper-resistant blockchain ledger.

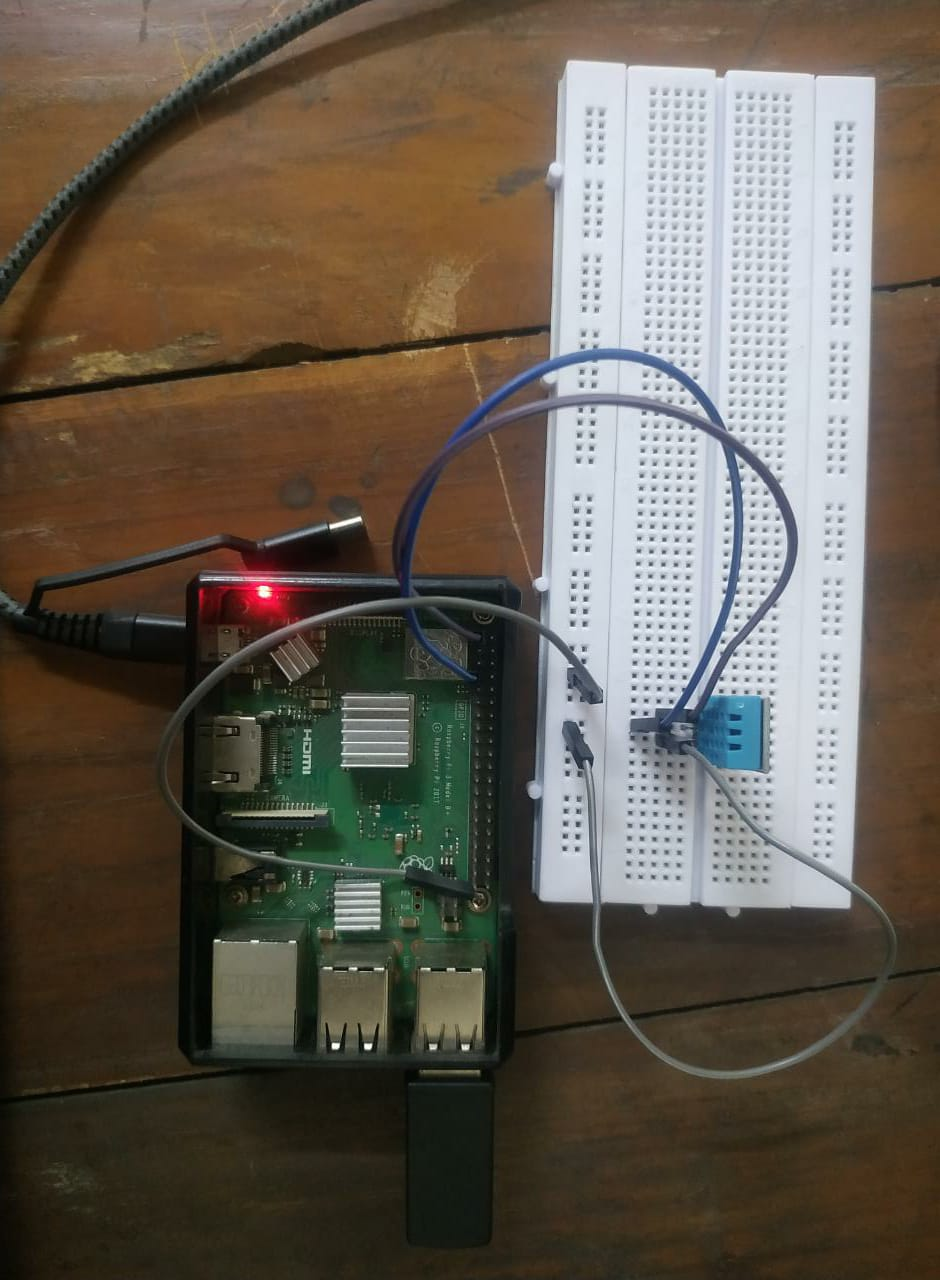
**5] Connecting Code with our server:** We have used base as FASTAPI to create our server. Since many libraries are not compatible with the Raspbian Os so we have to copy the code and then make the library. We are have then integrated our code with and running the server on our local machine. We Mapped our code's functionalities to API endpoints. Each endpoint represents a specific URL that clients can use to access a particular resource or perform an action and also defined the HTTP methods (GET, POST, PUT, DELETE, etc.) associated with each endpoint. For Authentication and authorization, we have used API key which is the unique key generated for our Raspberry Pi by FASTAPI Client.

**6] Testing and Running our Code in the server:** We have configured a localhost server in our Raspberry Pi itself and tested the working of our Project. Since FASTAPI for Raspbian is currently in Pre Alpha stage so its proper UI is not working which enhances the interface but we have made the libraries required for our project and hence all the functionality are working correctly.

## Results and Discussion

**Setup of Raspberry Pi With Sensor**

It reads the data form the sensor and show the data in monitor connected to Raspberry Pi.



***FIG 2:*** ***Setup For Temperature and Humidity Reading***

We have discussed the possibilities of different devices which can be implemented using Raspberry Pi. Raspberry Pi is used in many sectors like research, healthcare and industrial machines. We have used DHT 11 Sensor to take the reading of instantaneous Temperature and Humidity

## 

## Code for implementing the Blockchain

## 

## *FIG 3.1* *Code of creating blockchain, generating hash and validating block*

## The Blockchain class has the following methods:

## \_\_init\_\_(): initializes the blockchain with a single block, known as the "genesis block".

## \_create\_block(): creates a new block with the given data, proof, previous hash, and index.

## get\_previous\_block(): returns the most recent block in the chain.

## \_to\_digest(): encodes the given values as a string and returns it as bytes.

## \_proof\_of\_work(): performs a proof-of-work algorithm to find a valid nonce (proof) that makes the hash of the current block start with four zeros.

## \_hash(): returns the SHA-256 hash of the given block encoded as a JSON string.

## mine\_block(): creates a new block with the given data and adds it to the chain.

## is\_chain\_valid(): checks the validity of the blockchain by verifying that each block's previous hash matches the hash of the previous block, and that the proof-of-work algorithm was properly performed for each block.

## Code for connecting to Server

## 

## *FIG 3.2* *Connecting to the server*

## 

## *FIG 3.3: Code of get and post request in server*

## 1] The code imports necessary libraries and modules, including FastAPI, a blockchain module, uvicorn, threading, and a module named "Humidity."

## 2] An instance of the blockchain class is created.

## 3] We initialized the FASTAPI

## 4] The entry() function is defined, which is responsible for generating data, adding it to the blockchain, and scheduling itself to run every 10 seconds using a timer. It also cancels the timer after every third execution.

## 5] Several API endpoints are defined using decorators and functions:

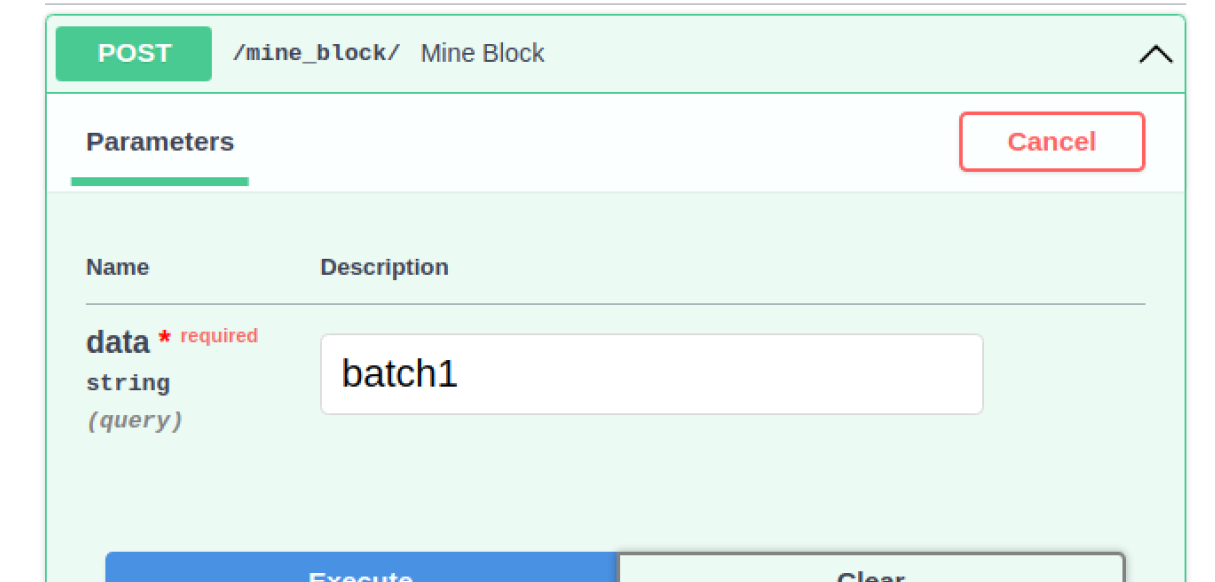
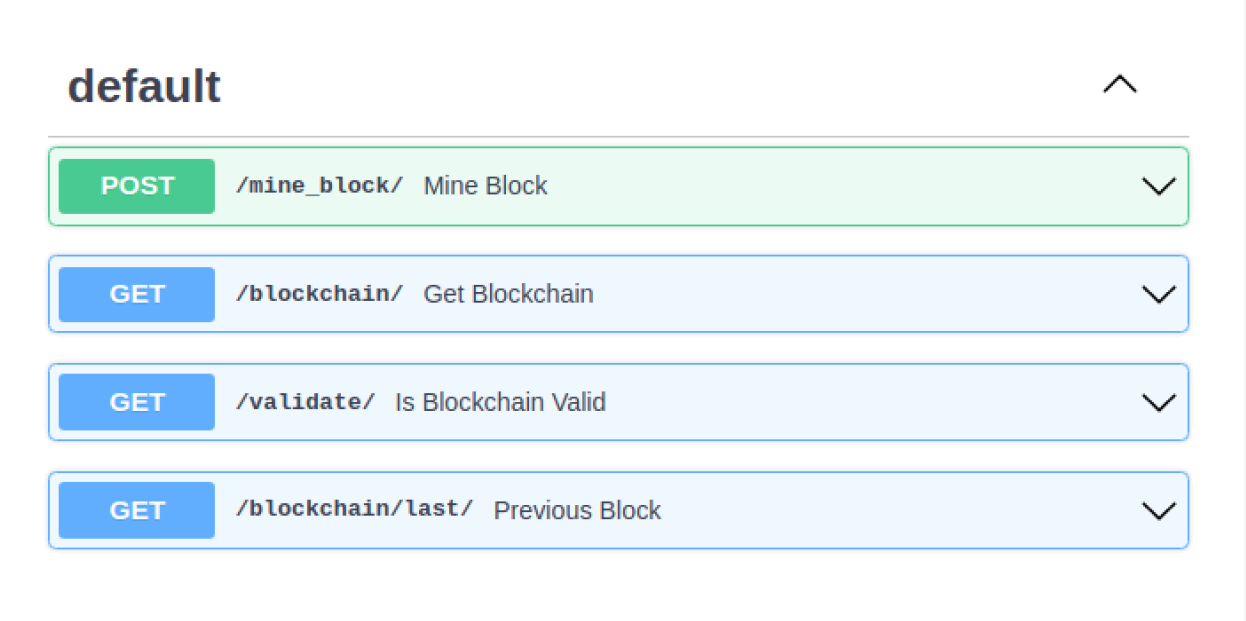
## /mine\_block/ is a POST endpoint that mines a new block by adding the provided data to the blockchain.

## /blockchain/ is a GET endpoint that returns the entire blockchain.

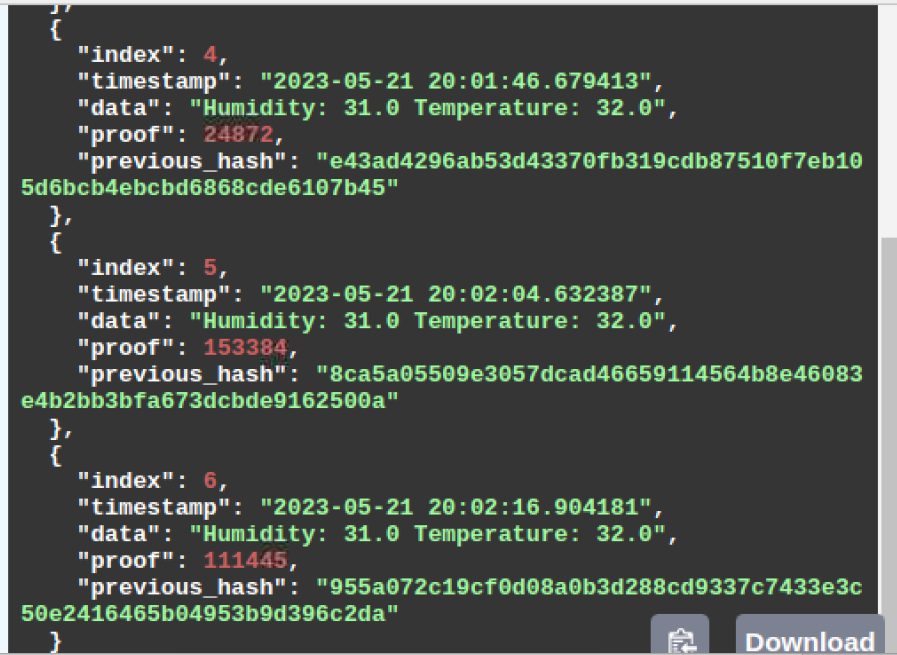
## /validate/ is a GET endpoint that checks if the blockchain is valid.

## /blockchain/last/ is a GET endpoint that returns the last block of the blockchain.

**Interface of our Web App**



***FIG 4.1:*** ***Starting menu FIG 4.2:*** ***Menu of Data Entry***



***FIG 5: Output Result***

**CONCLUSION AND FUTURE SCOPE OF WORK**

We have successfully created a Blockchain class that implements a simple blockchain data structure. This code defines a basic implementation of a blockchain using Python. We have developed a method for updating the blockchain ledger and also mode method for validating our blockchain. This ledger securely stores the information and data collected by the sensors with the help of Raspberry Pi 3. As the data is collected, it is secured in the blockchain network. We have also linked our blockchain with our own API taking the base as Fast API so that we can remotely access our blockchain from anywhere. Thus, IoT security is ensured.

**Future Work:**

We will add some more methods to improve our blockchain

1. Consensus algorithm we can used to ensure that all nodes on the network agree on the validity of a transaction before it is added to the blockchain.

2. Will improve the algorithm efficiency by using Solidity and some more advanced fast and efficient programming language specifically designed for Blockchain

**REFERENCE**

**1] Md Yunus, M. A.** (2017). Internet of Things (IoT) Application in Meliponiculture. International Journal of Integrated Engineering, 9(4).

**2] Zhang, Y., & Wen, J.** (2016). The IoT electric business model: Using blockchain technology for the internet of things. Peer-to-Peer Networking and Applications, 10(4), 983–994.

## 3] Rabiya Abbas, Zainab Sultan, and Shahid Nazir Bhatti. 2017. Comparative analysis of automated load testing tools: Apache JMeter, Microsoft Visual Studio (TFS), LoadRunner, Siege. In 2017 International Conference on Communication Technologies. IEEE, Rawalpindi, Pakistan,

## 4] Bijker, Wiebe E., Thomas P. Hughes, Trevor Pinch, and Deborah G. Douglas. 2012.

## The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology. MIT press.