Blockchain for IOT

Bachelor of Technology

in

Information Technology

Submitted by

S. No	Name	Enrollment No
1	Prateek Durgapal	IIT2019014

Under the supervision of

Dr. J. Kokila

Indian Institute of Information Technology, Allahabad, Uttar Pradesh, India

Content

1. Introduction

The present world we're living in has more IoT connected devices than humans. There is a whole variety of them from wearable devices to inventory tracking chips. With IoT devices proliferating, these devices often lack the authentication standards necessary to keep user data safe. Critical infrastructure will be damaged if hackers penetrate through the broad range of IoT devices. To ensure trust, authentication, and standardization across all elements of IoT are essential for widespread adoption.

2. Problem Identification

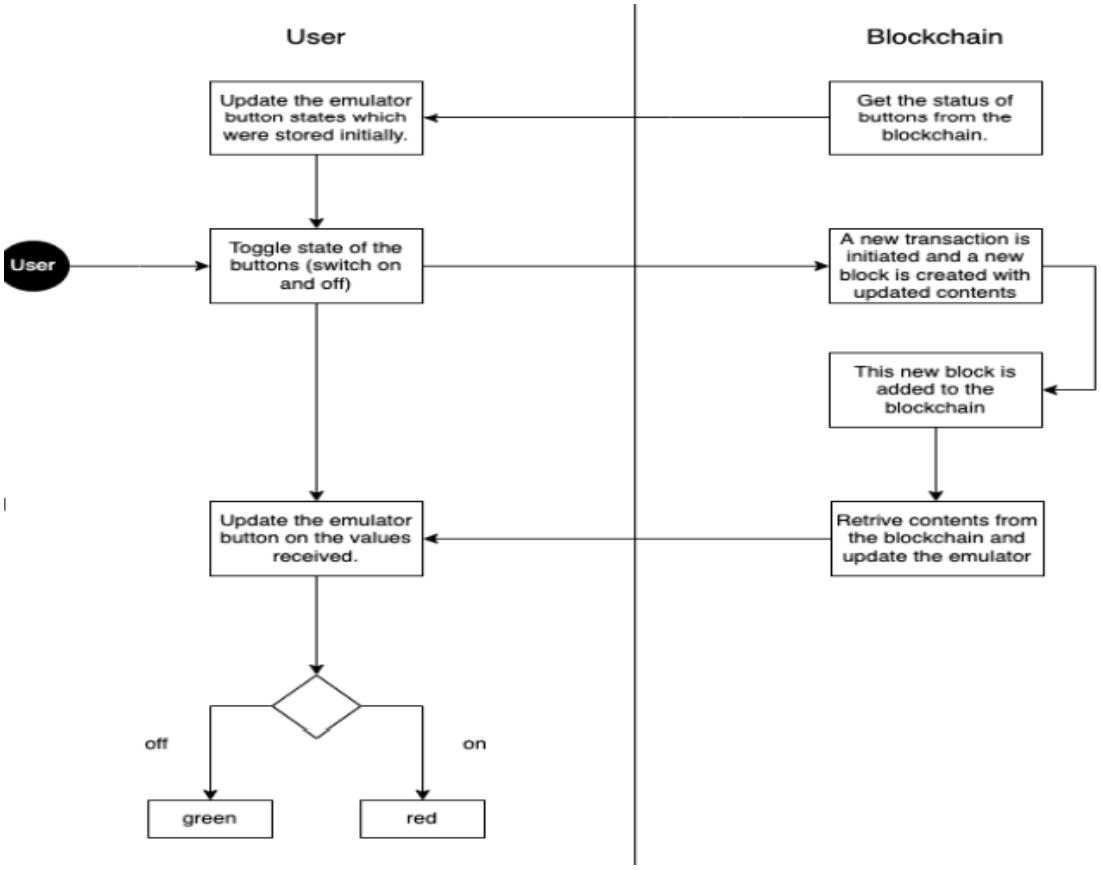
The very main objective for IoT players is to protect the information in the ecosystem. In this report, we have discussed how blockchain can be used to store information of IOT devices, specifically the status of GPIO Pins (on / off) Raspberry Pi, and temperature readings from thermal sensor emulator.

3. Need for Blockchain

- There are three major advantages of the blockchain, which are enhanced security, reduced costs, and speed of transactions.
- Blockchain can be used to track the sensor data measurements and prevent duplication with any other malicious data. The deployment and operation costs of IoT can be reduced through blockchain since there is no Intermediary.
- Deployments of IoT devices can be complex, and a distributed ledger is well suited to provide IoT device identification, authentication and seamless secure data transfer. IoT devices are directly addressable with blockchain, providing a history of connected devices for troubleshooting purposes.

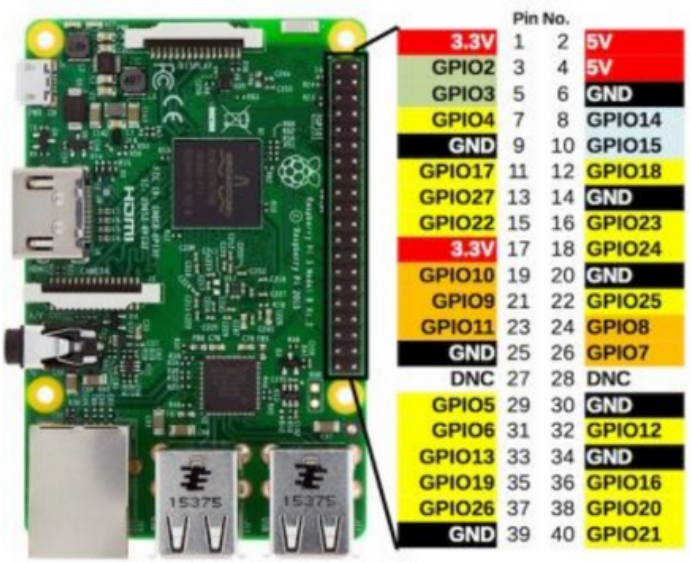
- Once the data stored in the blockchain can not be altered, this plays a very important role for IoT devices.
- In our case, The correctness of information of IoT devices (Raspberry Pi, Thermal sensor) is very important if there is software which depends on this information. Using blockchain technology, We can ensure the correctness of this information since it can not be altered by any malicious third party.

4. Activity Diagram



5. Entities and Functionalities

- User: The user directly communicates with the GUI and can control the endpoints of the GPIO pins/Raspberry Pi with the click of a button. Any device / sensor connected to the endpoint i.e. an LED or a sensor will be triggered and this transactional data would be stored in blockchain. Users can also see the transaction history of the blockchain with the GUI.
- GPIO Pins: GPIOSim is an emulator used for simulation of Raspberry Pi. Raspberry Pi has 40 pins which are represented internally in the code as a list which allows us to give a HIGH or LOW signal to any pin needed.



Raspberry Pi

- Virtual Sensor: We have implemented a temperature sensor emulator which gives custom temperature readings every 2 seconds. This is assumed to be connected with one of the pins of Raspberry Pi. Once the pin is on, the temperature readings start to get stored locally as well as in the blockchain.

6. Implementation Details and Smart Contract

- *myContract.sol*

  This is the one and only smart contract for our project. It defines a structure called `pin` which comprises of three things:

  - `isTempSens`: whether the sensor attached is a temperature sensor or not.
  - `status`: whether the pin is on (1) or off (0).
  - `temp`: current temperature reading if `isTempSens` is true.

  Next, a mapping between the pin and this structure called `pinStatus` is defined which would store the above data for each pin. Finally a function named `control` is defined which controls the pin by changing the values in the structure of a pin accordingly.

- *globals.py*

  Defines 3 variables for future usage. The variables are:

  - `PATH_TO_DEVICES`: In a real Raspberry Pi, any sensor connected to it logs its values continuously in a file stored in a folder called *devices*. So, we created a *devices* folder of our own in the project directory where our virtual temperature sensor logs its values.
  - `TEMP_PIN`: the pin number to which our temperature sensor is supposed to be connected. In our case, we have chosen `TEMP_PIN = 5`.
  - `PROCESS`: it stores the process variable which is responsible for running *virtual_temp.py* in the background. This process is created when the temperature sensor is turned on and terminated when it is turned off.

- *virtual_temp.py*

  It first creates and opens a file called *temp* in the *devices* folder. Then a random initial temperature value is chosen in the range [-100, 200]. Now, while the process is active, the temperature is written in the *temp* file in every 2s interval. The temperature value is varied by ± 3 in this interval. While the temperature is being logged, simultaneous transactions are also being performed in the blockchain storing the temperature information.

- *app.py*

  This file has the following tasks:

  - set up every pin (40 in total) of GPIO Simulator for output mode.
  - use the compiled smart contract to extract the ABI and bytecode of the contract.
  - instantiate the contract using the ABI and bytecode.
  - run the Flask app and manage routing of users to the home page and transactions page.
  - change the status (and temperature if possible) of each pin. If a temperature sensor is attached to the pin and action is *on,* then run *virtual_temp.py* in the background. If action is *off,* then terminate this script.
  - output HIGH / LOW in the GPIO Simulator accordingly.

7. Improvements

We had proposed 2 improvements:

- Our current model only stores 0/1 i.e. status of each pin. We could also store the information of the sensor connected to that pin, for example, temperature readings from a thermal sensor or coordinates from a GPS sensor. Since we did not have an actual temperature sensor, we wrote a Python script to simulate one as explained in the description of *virtual_temp.py*.

- Instead of simulating Raspberry Pi using python libraries, we could use a real Raspberry Pi / Arduino and connect it to blockchain to extend the practical use case of our project. We indeed extended our project to use a Raspberry Pi to switch on / off an LED sensor, the demonstration for which can be found below.

8. Frontend Development

We have 2 files which handle the frontend:

- *index.html*

  The home page of our webapp. Displays the ON / OFF buttons corresponding to each of the pins. Also contains a link to view transaction history. We have also handled the case if a button is clicked ON when the pin is already HIGH (or vice versa).

- *trans.html*

  This page shows the transactions in the blockchain. But first we need to browse where the *logs* file is stored (in the project folder) which would contain all the transactions. This is done for security reasons. Otherwise you would be able to create a JavaScript that automatically uploads a specific file from the client's computer and thereby leaking sensitive information.

9. Demonstration

The emulator demo with a virtual temperature sensor can be viewed here. The demo with a real Raspberry Pi connected to an LED can be viewed here.

10. References

1. https://github.com/Salmandabbakuti/IoT-and-Blockchain
2. https://faun.pub/demo-on-iot-and-blockchain-integration-3337b60db81e
3. https://innovationatwork.ieee.org/blockchain-iot-security/
4. https://www.trendmicro.com/vinfo/mx/security/news/internet-of-things/blockchain-the-missing-link-between-security-and-the-iot
5. https://www.chakray.com/blockchain-iot-security/