Semantic Web of Things with Blockchain

Abhishek Shrestha, 390055
M.Sc Computer Science,
Technische Universität Berlin
Berlin, Germany
Email: abhishek.shrestha@campus.tu-berlin.de

Sarthak Gosh, 390176
M.Sc Computer Science,
Technische Universität Berlin
Berlin, Germany
Email: s.ghosh@campus.tu-berlin.de

***Abstract*— The Semantic web can basically be defined as a decentralized web which attempts to provide a platform where people and devices can interact constantly via data. And although there are many studies currently going on about the Semantic web, one of the most important concern is security. By its very definition, Semantic web attempts to provide an interconnected system to which anyone may contribute. So, it becomes hard to verify the trustworthiness of each data source. And this is where a fairly new technology, Blockchain, comes in. The transparency and security in data transaction that a Blockchain offers and its decentralized and immutable nature makes it ideal for data transactions within a decentralized network. Through our project we tried to understand how Semantic web, Blockchains and Internet of Things (IoT) can be correlated and how a transaction actually works. We created an application that gathers the IoT data, which in our case is the environment data namely pressure, temperature and humidity. Then we implemented a decentralized application (dApp) that runs on an Ethereum Virtual Machine (EVM). A web interface provides the user an option to select which data he wants to upload. The application then uploads the data to the Blockchain. Users within the network can then transact this data with Ether as virtual currency.**

Keywords—Semantic web, Blockchain, Internet of Things (IoT), Decentralized Application (dApp), Ethereum Virtual Client (EVM), Ether

I. INTRODUCTION

In the creators' own words: "Web 2.0 is the business revolution in the computer industry caused by the move to the internet as a platform, and an attempt to understand the rules for success on that new platform. Chief among those rules is: Build applications that harness network effects to get better the more people use them" [1]. There is no denying that Web 2.0 has changed the way we interact with the web. The more users and data a source could generate, the more effective it becomes [2]. Various web hosting services provide us the ease of not having to own our very own servers to host data. However, these services have now become so massive that the whole web is now a handful of nodes (Google, Amazon, Facebook and so on). All the data/ queries are routed through these central nodes. Which is not good in terms of privacy, security and economy. Moreover, this also means that there is a single point of failure. Semantic web tries to move away from these issues by introducing the concept of decentralizing the web thereby removing the central authority and distributing control over the network.

Blockchains on the other hand are a type of linked lists that are immutable and are organised based on time stamp. The inherent characteristics of Blockchains such as proof by consensus, immutability, secure transaction recordings and their distributed nature makes them applicable for more than just Cryptocurrencies [3]. Platforms such as Ethereum offers features to store code snippets (smart contracts) in the Blockchain allowing us to build decentralized applications with our own rules of ownership, transaction formats and state transition functions.

Thus, the idea of semantic web together with Blockchain technology can be implemented to construct decentralized applications that can support data or file transactions in a secure and transparent way. Moreover, the removal of single point of failure and the risk of a single authority having too much information makes Semantic web with Blockchain ideal for transaction of IoT data.

Our objective with this project was to implement the most basics of these technologies to create a decentralized application with Ethereum Blockchain capable of uploading files and also retrieving specified files from the Blockchain and thus performing a transaction which should cost some defined virtual currency.

The remainder of this report is organized as follows. In section II, a brief discussion on the related works which we studied during and before development. Section III describes in detail about our workflow and approach. Section IV presents our evaluation and outcomes. Section V discusses our conclusion and future work.

II. RELATED WORK

Since this is a very interesting topic and promises a broad area of application, we found a lot of works which we could take reference from. Some of the most notable ones are referenced in [4], [5], [6].

[4] is a voting application deployed in Ethereum Blockchain. The author uses Truffle framework [7] to manage smart contract and Solidity [8] to write the contract. Ganache [9] is used to create a local block chain to test the application and Chrome with Metamask [10] extension is used to interact with the Blockchain. He uses boilerplate code (Pet Shop box) [11] offered by Truffle for development and later updates the code as per requirement.

The contract itself is pretty simple and straight forward. He simply creates a mapping between the vote count and individual candidates. In the constructor of the contract he creates two candidates. So, whenever the contract is deployed two candidates are created. A separate function is used to count the votes each candidates received. For the client-side application to interact with the smart contract, the HTML and JavaScript files that came with the "Pet Shop box" are modified.

This project is quite similar to how we ended up writing the IPFS [12] file hash from user to Blockchain. There is no logic for retrieval of data or transaction of data but still the write logic to the Blockchain and the interaction with client-side application is similar with our implementation.

In [5] the author creates an application that allows the user to select the image file he wants to upload to IPFS. The uploaded image can be then be viewed on the UI. For the

front end he uses react.js and a separate api to communicate with IPFS. He uses IPFS node provided by Infura [13]. We used the similar concept to upload text files to IPFS. The api he uses to connect to Infura is similar to what we ended up implementing since it does not require us to start our own instance of IPFS.

[6] is another voting app. This is quite similar to the voting app in [5] but is a bit simpler. The author here uses Solidity to write the smart contract. The client side application again uses HTML and Javas script. The 3 candidates whose votes have to be counted are hardcoded at the front end. The user then types the name of the candidate he want to vote for and the application calls the smart contract and writes the value to Blockchain. This app was a really good reference for us to build the part where the app writes the data to the Blockchain. Our initial smoke test was done with an application similar to this.

III. Our Approach

In this section we describe in details how we implemented different modules of application. Along with a control flow diagram and flowcharts for better representation.

For better overview we can divide our workflow into 2 parts.

A. *Gather IoT data.*

We used Raspberry Pi with sense Hat to gather the environment data. This in our case includes temperature, pressure and humidity. The data is collected every 10 minutes and stored in the memory in a list. After every hour all the data is moved to a physical file and the memory is cleared. However, for all the data of a single day, the same file is appended. And when a new day begins, a new file is created. For storing the three different types of data (temperature, humidity and pressure), three separate files are created.

So each day we have a new file with 144 entries and 3 new files. The files were named using the following convention:

HData_datestamp, PData_datestamp, TData_datestamp for Humidity, pressure and temperature respectively.

So for instance, for April 18, 2018 all the collected temperature will be stored in a file with name PData_2018-4-18. The data is stored in file in the format as shown in fig (1).

A

```
08:43:00 PM Wed 18 April, 2018    38.3 %
08:48:00 PM Wed 18 April, 2018    38.3 %
08:53:00 PM Wed 18 April, 2018    38.3 %
08:58:00 PM Wed 18 April, 2018    38.3 %
09:00:00 PM Wed 18 April, 2018    37.54 %
09:05:00 PM Wed 18 April, 2018    37.54 %
09:10:00 PM Wed 18 April, 2018    37.54 %
09:15:00 PM Wed 18 April, 2018    37.77 %
```

B

```
08:43:00 PM Wed 18 April, 2018    1007.67 Millibars
08:48:00 PM Wed 18 April, 2018    1007.67 Millibars
08:53:00 PM Wed 18 April, 2018    1007.9 Millibars
08:58:00 PM Wed 18 April, 2018    1007.9 Millibars
09:00:00 PM Wed 18 April, 2018    1007.87 Millibars
09:05:00 PM Wed 18 April, 2018    1007.73 Millibars
09:10:00 PM Wed 18 April, 2018    1007.62 Millibars
09:15:00 PM Wed 18 April, 2018    1007.56 Millibars
```

C

```
08:43:00 PM Wed 18 April, 2018    25.67°C
08:48:00 PM Wed 18 April, 2018    25.79°C
08:53:00 PM Wed 18 April, 2018    25.77°C
08:58:00 PM Wed 18 April, 2018    25.37°C
09:00:00 PM Wed 18 April, 2018    26.05°C
09:05:00 PM Wed 18 April, 2018    25.56°C
09:10:00 PM Wed 18 April, 2018    25.43°C
09:15:00 PM Wed 18 April, 2018    25.19°C
```

Fig. 1. Data storage format for A: Humidity. B: Pressure. C: Temperature.

B. *Decentralized Application with Ethereum for transacting IoT data.*

We used Truffle framework [7] and Solidity [8] for Smart Contract development. For front end development we are using Reactjs [14] and Node.js [15] framework for backend. To create a local Blockchain for testing we used Ganache [9] and Metamask [10] in the Chrome browser to connect with the local Blockchain. A predefined ipfs api was used that connects to Infura [13] for IPFS nodes [12]. We used the truffle-box boilerplate code for getting started [16].

The dApp provides an interface through which users are able to upload files to Blockchain and also retrieve specified files at the cost of some virtual currency. It is not a good practice to upload entire files to the Blockchain due to efficiency issues. So instead we upload the files to IPFS and get a unique file hash for the uploaded file and then upload that file hash to Blockchain.

The user can select the data file he wants to upload from the UI. Once user submits the file to upload, the file is first uploaded to IPFS and a unique file hash corresponding to the file is generated. This file hash along with the filename and the user who uploaded it (account number) is then written into the Blockchain using the smart contract. So the Blockchain contains a unique mapping of filename and the file details which in this case is the uploader and the file hash. The list of uploaded files can be seen in the UI. And the list is same for all the users across the network since the grid uses IPFS link as source.

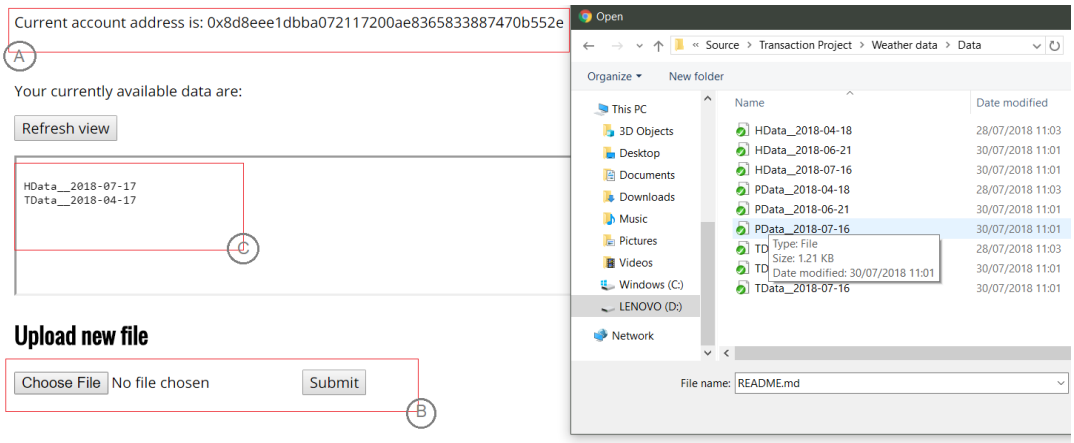

Fig. 2. File upload UI showing A: Current user account address. B: File selection. C: Currently uploaded files

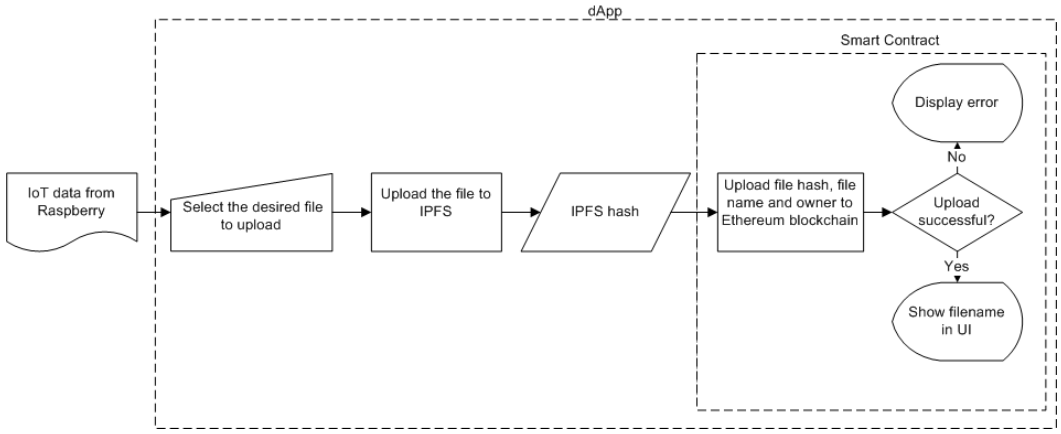

Fig. 3. Steps to upload file

Any other user who sees the file list in the UI and wants to buy the data of a particular day, can enter the file name in the UI. When the user submits, the smart contract checks if the user is able to buy the file via a "payable" function which verifies if the user has enough Ether and whether or not the file exists in the Blockchain. The minimum Ether requirement for buying a file is set to 10 (~ 37000 Gas).

Once all validation is successfully completed, the Ether is transferred to the owner of the file and the file hash associated with the file is then provided to the buyer. The buyer can see the file hash along with transaction details on the UI. The buyer can then access the file with the file hash. The transaction details is also shown which contains the account number of buyer and the seller.

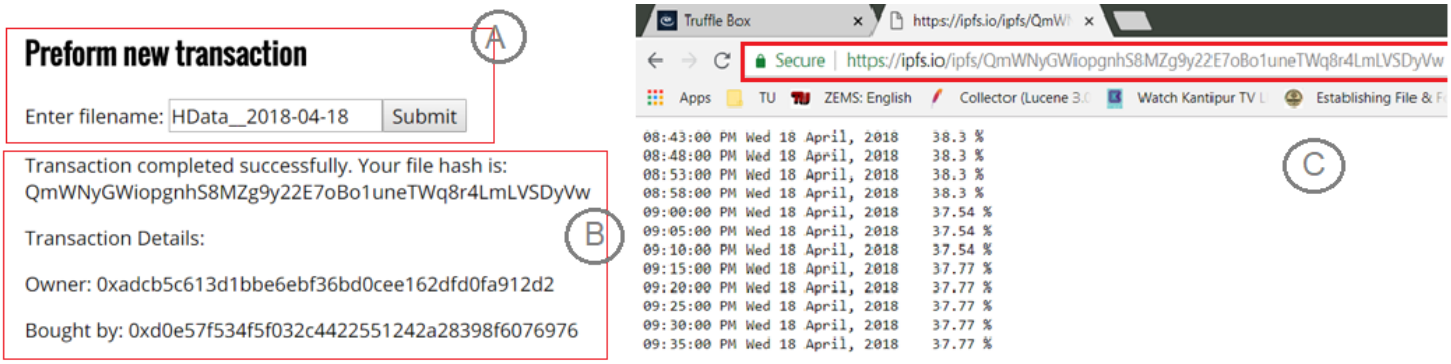

Fig. 4. A: User enters the file he wants to buy. B: result is displayed after successful transaction. C: File can be access through the file hash

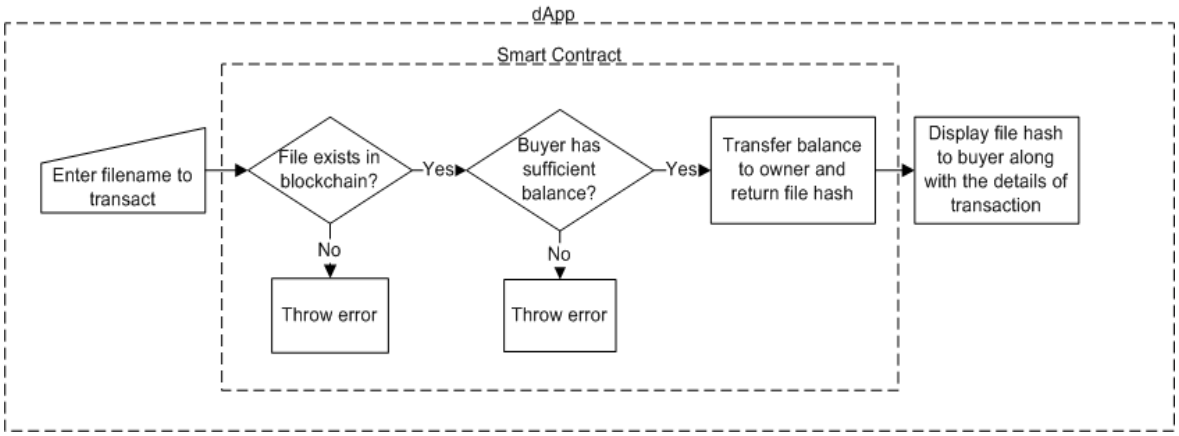

Fig. 5. Steps to retrieve/ buy file

The entire control flow can thus be summarized in Fig (6)

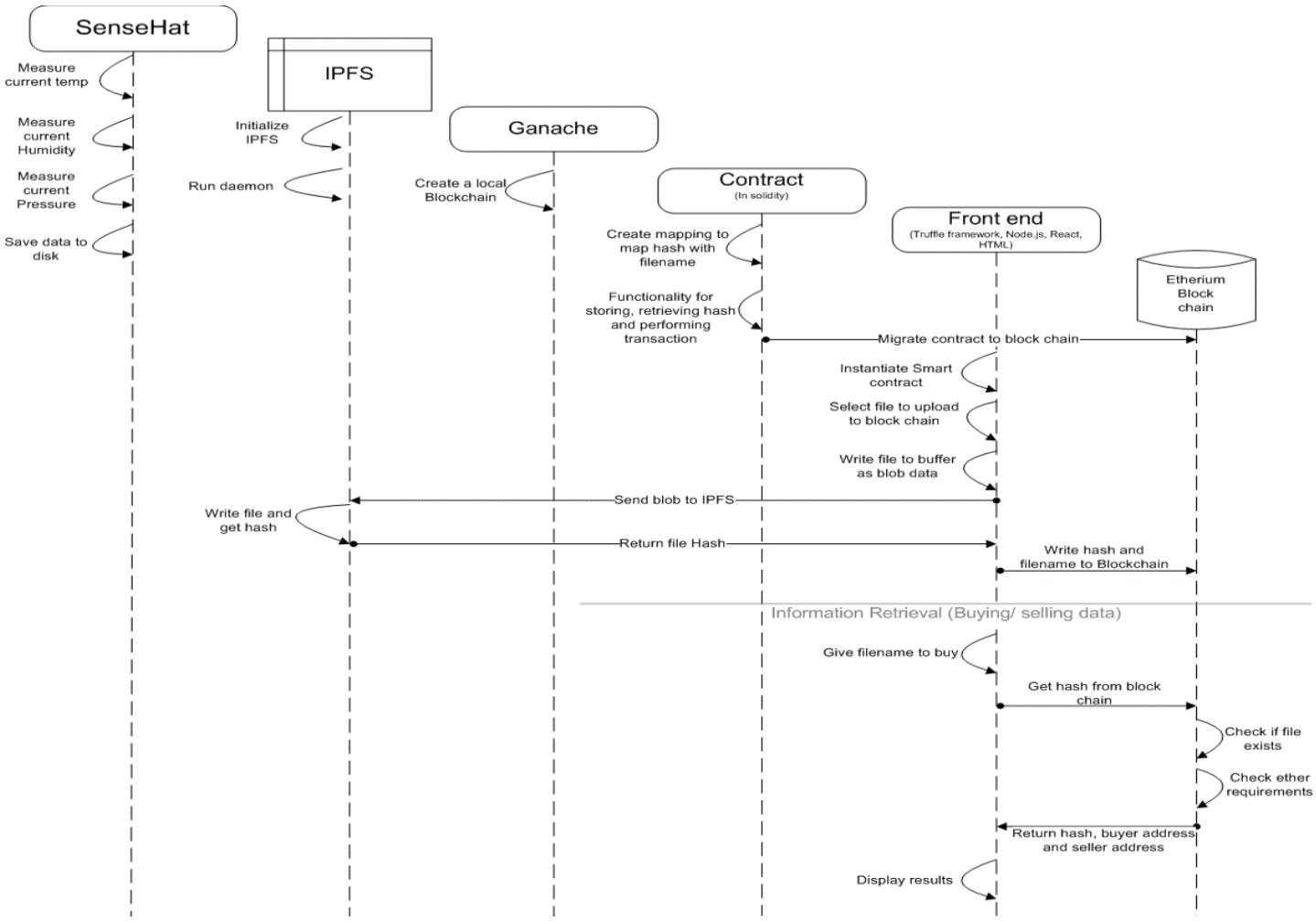

Fig. 6. Control flow diagram demonstrating the whole workflow

IV. EVALUATION

Concerning the data collection from Raspberry Pi, we monitored and collected data for 24 hours. All the data seems to align except the temperature data. One of the problems with the temperature is that the sensor is too close to the CPU [17]. So the CPU heating up was causing the readings to be always constant and not accurate at all. So we had to introduce a correction for getting the temperature data.

The corrected temperature was calculated with relation as in (1).

$$t\_corr = t - ((t\_cpu - t) / 1.5) \quad (1)$$

Where, t_corr = Corrected temperature, t is the average of temperature computed from humidity and temperature computed from pressure [18] and t_cpu is the CPU temperature. Though the temperature readings are better but not completely accurate.

The file upload to IPFS seems to be a bit slow even for small files. This might be because we are using Infura. For the smoke test we hosted local IPFS node and it was working fine. Further, the UI refresh is also a bit clunky as it takes some time to refresh the list showing the currently uploaded files.

From fig (6) we can see that every module of the system are pretty closely interacting except for Ganache and Sense Hat. This is because Ganache is optional, we can create our own local Blockchain without Ganache (may be with multiple Raspberries). Further, the data in our case was IoT data from the Sense Hat but in reality this data could be anything e.g. Images, text files, music files etc. Thus, this module, like Ganache, is implementation specific and can be replaced by anything else without needing to change or update any other module.

Moreover, the payable function in the smart contract sometimes does not work. It is intermittent and we could not find the possible cause of the issue. May be it is an issue with the local Blockchain provider we are using. But we were unable to fix it.

V. Conclusion and Future Work

Our project was a simple implementation of the logics involving Semantic web, Blockchain and web technologies like Node.js, Reactjs and HTML. It does not intend to solve any new or major problem or present any complexities with the currently used technologies. The goal of our project was to familiarize ourselves with the basics of these technologies so that we would be able to understand and handle more advanced problems and we believe that we fulfilled this goal.

This project could be a nice reference point for anyone who wants to get started with any of the technologies we used. Since, we were also complete beginners while we started coding this, and we took references from multiple sources and had a lot of problems solving simple issues especially with the smart contract and UI issues with Reactjs. So we intend to create some sort of tutorial video on Youtube and link our Github there so that people can get started as we did.

Further, we would like to run the application in an actual Blockchain rather than the one created from Ganache. We did not have enough time to test this but this is something we would definitely like to do.

Acknowledgment

Although hard work and proper management are the key to success of any project, there are several other crucial factors like guidance, motivation, support and feedbacks that are significant for any kind of endeavour. The success and final outcome of our project would not have been possible without the proper supervision and assistance of our supervisors Dr. Danh Le Phuoc and Dr. Qian Liu. By the end of the project we have developed confidence in using various web development tools, developing smart contracts and learned a lot about Web 3.0 which were completely unknown technology for us previously and for this we are really grateful to the reputed supervisors for providing us this wonderful opportunity.

We would also like to thank Mrs. Heike Fischer for providing all the necessary information during the enrolment process. We are really thankful that she responded and helped us in every possible way during enrolment and responded to our numerous e-mails regarding the process. We are deeply obliged for it.

Finally, we would like to thank Fraunhofer FOKUS and TU Berlin, Department of Open Distributed Systems for organizing this course.

References

[1] O'Reilly, Time (2006, Dec.) "Web 2.0 Compact Definition: Trying Again". [Online]. Available: http://radar.oreilly.com/2006/12/web-20-compact-definition-tryi.html

[2] Josh Swiss (2017, Jul.) "Blockchain, Bitcoin and the Semantic Web: A Synopsis of all the Buzzwords you Were too Scared to ask About!". [Online]. Available: https://medium.com/@joshswiss/blockchain-bitcoin-and-the-semantic-web-a-synopsis-of-all-the-buzzwords-you-were-too-scared-to-5b275c4501fc

[3] Héctor Ugarte (2017, Jun.) "Semantic Blockchain: A more Realistic Web3.0". [Online]. Available: https://medium.com/@heeduugar/semantic-blockchain-a-more-realistic-web-3-0-9eda20867645

[4] Gregory McCubbin (2018, May.) "A Decentralized Ethereum Voting Application Tutorial". [Online]. Available: https://github.com/dappuniversity/election

[5] Gregory McCubbin (2018, Jun.) "IPFS Image Uploads dApp With Ethereum Smart Contracts". [Online]. Available: https://github.com/dappuniversity/ipfs_image_uploader

[6] Siraj Raval (2018, Jun.) "Your First Decentralized Application". [Online]. Available: https://github.com/llSourcell/Your_First_Decentralized_Application

[7] (2018) Truffle Official site. [Online]. Available: https://truffleframework.com/

[8] (2018) Solidity Official site. [Online]. Available: http://solidity.readthedocs.io/en/v0.4.24/

[9] (2018) Truffle Official site. [Online]. Available: https://truffleframework.com/docs/ganache/using

[10] (2018) Metamask official site. [Online]. Available: https://metamask.io/#how-it-works

[11] (2018) "Pet Shop Truffle Box". [Online]. Available: https://github.com/truffle-box/pet-shop-box

[12] (2018 Nov.) An introduction to IPFS. [Online]. Available: https://github.com/INFURA/tutorials/wiki/Introduction-to-IPFS

[13] (2018) Infura Documentation. [Online]. Available: https://infura.io/docs

[14] (2018) Reactjs Official site. [Online]. Available: https://reactjs.org/

[15] (2018) Node.js Official site. [Online]. Available: https://nodejs.org/en/

[16] (2018) Boilerplate code available from Truffle-box. [Online]. Available: https://github.com/truffle-box/webpack-box

[17] (2016, Jun.) The official Raspberry Pi forum, Sense Hat Rubbish temperature accuracy. [Online]. Available: https://www.raspberrypi.org/forums/viewtopic.php?t=152498

[18] (2016 Aug.) Accurate temperature reading from Raspberry PI Sense HAT. [Online]. Available: http://yaab-arduino.blogspot.com/2016/08/accurate-temperature-reading-sensehat.html