meterBlock

Supervisors: Mr. Viv Crone
Mr. Mitch Cox

Blockchain facilitated power exchange for microgrid applications

Brandon Verkerk – 875393 **Christopher Maree-1101946**
School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg 2050, South Africa

16-07-2018

Executive summary:

meterBlock is a collaborative project that aims to provide a decentralised platform to facilitate the governance of electrical energy consumption and production, measured and controlled by Distributed Ledger Technology (DLT) connected power meters. The goal of the project is to build a Proof-of-Concept (PoC) prototype device running on the Ethereum blockchain to facilitate this energy governance. The project commenced on the 1st of July 2018 and will run to completion eight weeks later on the 26th of August 2018.

A primary use of this technology is to provide a platform of energy exchange between producers and consumers within self-sustaining micro-grids, trading surplus energy between one another. Full scale implementation involves replacing the current national grid's management system, whereby each individual entity can act as both a consumer and producer, governed by the blockchain, enabling a novel, trustless, decentralised, energy exchange marketplace. These entities could include, but are not limited to: pure producer - such as power utilities (Eskom) - pure consumer - such as manufacturing plants or traditional households - or producer/consumer - such as gated communities, households, hospitals, shopping centres etc. with installed sources of renewable energy.

The project will run on an Ethereum based blockchain (such as Quorum) and will implement the tokenization of an energy unit through an adaptation of the ERC20 token standard.

meterBlock lays the foundations for future collaborative work when taken to its logical extreme at a national grid level. This report discusses the inherent issues that must be considered before full scale implementation of a DLT-based system can be put into production. Some of these issues relate to the practicalities and complexities of implementation when considering application at a national grid level, but can also be extended to the legal and regulatory factors and their broader economic impacts.

Key Words

Below are a collection of application specific key words and jargon used throughout the report.

A. *Application Program Interface (API)*

A collection of functions and procedures that define an interface to access and control communication between various components.

B. *Blockchain*

A Blockchain is a public, peer-to-peer, digital ledger, used to record transactions chronologically. It represents an immutable, persistent record that can never be altered or changed by anyone due to the distributed nature of the consensus algorithm. There is no central authority figure due to the decentralised nature making it highly censorship resistant.

C. *Blockchain Node*

A node is a computer connected to the blockchain that stores a full copy of the historic ledger.

D. *Consensus*

The agreed state of the ledger based on the cumulative communication between the nodes within the blockchain based on a particular protocol, such as Proof-Of-Work.

E. *Decentralized*

System without central governance. There is no centralized authority that makes decision on behalf of all other parties.

F. *Distributed Ledger Technology*

A distributed ledger is a database that is shared and synchronized across a network spread throughout multiple sites with an underlying governance mechanism to ensure continued consistency.

G. *ERC20 Token Standard*

ERC20 is one of the technical standards used for the creation of tokens governed by smart contracts on the Ethereum blockchain.

H. *Ethereum*

Ethereum aims to create the ultimate protocol for building decentralised applications. Ethereum implements a Turing-complete programing language built into its blockchain, enabling anyone to create and deploy smart contracts

I. *Hypertext Transfer Protocol (HTTP)*

HTTP is a Communication and transfer protocol based off the client server model to facilitate direct communication between two entities used within the internet.

J. *Immutable*

Persistent storage of information that is unchanging over time. Records within the blockchain are considered Immutable as it is practically impossible to change them once they are recorded.

K. *Ledger*

An accounting mechanism used to keep a chronological list of transactions and interactions.

L. *Message Queuing Telemetry Transport (MQTT)*

MQTT is a data centric communication protocol that uses a client server publish/subscribe pattern. It is designed to be as lightweight, fast and reliable as possible.

M. *Metamask*

Metamask provides the ability to connect a standard web browser, such as Google Chrome, to the Ethereum blockchain via a predefined RPC endpoint.

N. *Middleware*

Application to connect two different communication protocols over a network.

O. *Off-chain*

Information that is not stored on the blockchain and as such cannot be controlled or accessed by blockchain entities, such as smart contracts.

P. *Oracle*

A mutually agreed upon mechanism used to provide information from the real world (off-chain) to an on-chain asset such as a smart contract. An example could be the oil price.

Q. *Censorship Resistance*

The ability of the system to prevent third party modification to the ledger.

R. *Smart Contract*

Self-contained entity that resides on a blockchain that directly controls the transfer of digital currencies or assets between parties under predefined conditions. They can be used to perform computation in a decentralised fashion, without any central controlling entity.

Abstract— **This paper serves to document the development of meterBlock - a decentralised platform to facilitate the governance of electrical energy consumption and production, measured and controlled by Distributed Ledger Technology (DLT) connected power meters - from conceptualisation through to implementation. As a prototype, meterBlock makes use of a Sonoff POW to monitor and record the energy consumption or production of a typical household, and communicates this information via MQTT to a Raspberry Pi Ethereum node. Each section within this report corresponds to the completion of a weekly milestone over the project duration, starting from the 1st of July 2018 until intended completion on the 26th of August 2018. Included too is the preliminary research and key design decisions - such as the choice of MQTT and the Ethereum blockchain - taken before development began, as well as a critical evaluation of the intended system as a whole with regards to implementation and price discovery.**

I. Introduction

Since the first enquiry from the national power utility (Eskom) in 1989 for the introduction of prepaid power meters, prepaid electricity has become a popular and convenient means of power management in many South African households[1]. meterBlock introduces a novel, scalable alternative energy management system, capable of addressing the consumption and production of energy at a national grid level through the use of DLT. This report covers contextual justification for the implementation of meterBlock, as well as the ideological advantages of the underlying protocol. It then discusses a detailed implementation plan that can be followed to achieve the final product.

II. Implementation Context

In order to fully justify meterBlock's intended use case, a critical analysis of the current system's operations and limitations is required.

A. South African Power Market

There are currently only two ways in which to purchase electricity in South Africa.

1) Pay-as-you-go model: consumers of electricity receive a bill at the end of the month corresponding to their energy usage for that month. This requires a representative of the national utility to visit the household to conduct a reading on the household's power meter. However, these readings are taken on an infrequent basis and as such are inaccurate from month to month as an averaging process is used. This means that on one month you may pay for more than you actually consumed and on another the converse may be true.
2) Pre-paid model: addresses the inaccuracies inherent to the pay-as-you-go model as funds are preloaded on the meter and the user is only billed as-and-when a kWh is consumed.

However, neither of these models facilitate the ability to sell power back to the national grid. This does not provide additional incentives for the installation of renewable energy systems within the household (or other consumers) as there are no mechanisms to profit from surplased energy produced. Additionally, the user has to trust the accounting mechanisms used by the national utility. This includes the manner through which energy consumption is recorded and the associated billing thereof.

This inherently means that there is no platform upon which a microgrid can be created, wherein energy exchange occurs between producers and consumers. Moreover, the current systems have a central point of failure due to being limited to having only one energy producer.

B. How meterBlock Addresses the South African Power Market's Limitations

As identified before, there are two fundamental problems with the current South African power market that meterBlock aims to address: naimly the ability to sell power back to the grid and the requirement to trust a central authority in the accounting and billing of energy. Both of these problems can be addressed through the implementation of a DLT governed power metering system.

The ability to sell power back to the grid is achieved through the use of autonomous smart contracts that govern and control the flow of tokens in accordance with energy consumed and produced. These smart contracts reside on a distributed blockchain and as such have no central controlling authority making the whole system inherently trustless. This results in the ideological notion of trustless governance. Additionally, this governance mechanism provides the ability for anonymous, transparent auditability of the billing ledger.

This fully justifies the exploration into the design of a DLT energy governance system. What follow is a detailed project plan that can be used to fully realise the minimum viable prototype.

III. High Level Proposed System Overview

The proposed system consists of 5 discrete components:

1) Toggle switch and energy measuring device: A device is required to control supply to a house (in consumption mode) or conversely control supply from a house to the grid (in production mode). Additionally, monitor and record energy consumed/produced by the house. In the proposed prototype, this is achieved with a Sonoff POW.

2) Communication protocol to connect toggle switch to Ethereum node: A communication protocol is required to convey all information from the toggle switch energy measuring device (Sonoff POW) to the blockchain as well as commands from the blockchain to the toggle switch. In the proposed prototype, this is achieved through use of a MQTT broker running on a Raspberry Pi Zero W.

3) Ethereum node and middleware: An access point is required to transmit the received information from the communication protocol to the blockchain. For this, a full ethereum node will run on a Raspberry Pi Zero W. Additionally, middleware is required to translate messages received view MQTT into information transmitted to the blockchain. In the proposed prototype, this will also run on the Raspberry Pi Zero W and will be written in Python using `Web3.py`.

4) Smart Contract Management System: A smart contract governance system is required to manage the allocation and deallocation of tokens based on consumption and production for each consumer/producer. In this implementation, this information is received via the middleware running on the Raspberry Pi Zero W.

5) User front end: A simple user front end dashboard is required to inform the user of their consumption/production as well as provide a mechanism to load/withdraw tokens. In the proposed prototype, this is achieved with a `vue.js` front end using `web3.js` to interact with the blockchain.

An overview of this proposed system can be seen in the Figure 1. This diagram shows the communication between households, Sonoff POW's and a node.

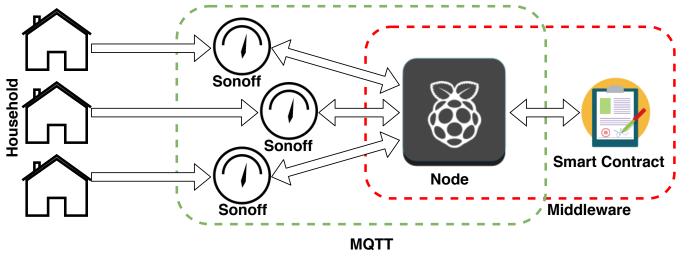


Fig. 1: MQTT System Diagram

This deliverable does not provide the complex economic mechanism to facilitate an energy exchange marketplace but rather provides the underlying governance protocol upon which this mechanism can be built. This economic mechanism is left as a discussion point for future consideration. The final intended prototype deliverable can, in theory, be used to replace the current pre-paid power model without alteration, providing a token based governance mechanism similar to that already used by the national energy market.

IV. Project Plan

The project plan is broken up into eight week-long sub-sections. A holistic overview of the project plan can be viewed in conjunction with the work breakdown structure and associated gantt chart outlined in appendix 2.A and 2.B.

A. Preliminary Research (01-07-2018 to 15-07-2018)

Before any implementation work can begin, a holistic overview of the surrounding technologies is required. This section outlines areas of interest to conduct research regarding: prepaid energy meters, blockchain projects of interest, IoT devices and the blockchain as well as real world applications of this technology.

1) Existing Blockchain related Energy Metering: A number of projects have been implemented using blockchains as auditing and accounting mechanisms for energy consumption and production. One notable project is Grid+, a project run by Consensys[2]. More information on Grid+ can be seen in appendix 1.A.

While Grid+ does have some similarities as the intended use case of meterBlock, they do not offer the ability to sell power back to the grid. Additionally, they do not offer any notion of energy exchange and are simply recognised as another power reseller. More over, their implementation would, strictly speaking, not work in the South African context as Grid+ requires the ability to sell power to consumers over the national grid. At present, one is not able to do this over the South African power grid as South Africa has one national power utility that does not accommodate for individual households to sell back to the National Power Utility (Eskom) [3]. As noted, this is too is a problem for meterBlock when the intended use-case is taken to its logical extreme, and as such the system should not be designed for this scenario alone.

2) Remote Switching Technology: A remote controlled switch, containing a relay or contactor, will be used to simulate the house power control mechanism. Future implementations of meterBlock could involve the design and construction of a high current controller, able to safely control the supply to a full household load.

For the purposes of this project, a Sonoff POW has been selected as it enables remote switching of a load along with remote energy consumption monitoring of said load[4]. The Sonoff POW connects to standard WiFi networks and can be reprogrammed to accommodate custom firmware and functionality. The relays can accommodate up to 15A and the device runs off of 220v mains supply without the need for an external power supply. In addition, the Sonoff POW is compliant with ICASA and the council Radio Equipment Directive (RED) 2014/53/EU making it acceptable to use in a household context[5].

3) Power Consumption Metering Technology: Current Clamps could be used to monitor the power consumed by the load, but for the sake of simplicity the internal power meter within the Sonoff POW will be used to monitor the load draw.

4) Selection of Appropriate Firmware: The stock firmware that comes with the Sonoff POW enables remote control of the application over Hypertext Transfer Protocol (HTTP) and Message Queuing Telemetry Transport (MQTT)[6], [7]. However, this firmware does not offer direct access to the relays over an API and requires the use of a verified application, such as eWeLink[3]. As a result, a custom firmware should be installed on the Sonoff to accommodate additional functionality. For the purposes of this project, Tasmota has been selected as the appropriate firmware. For an extensive justification of this decision, see appendix 1.B.

5) Selection of Appropriate Communication Protocol: A communication protocol is required to interact directly with the Sonoff's. This protocol needs to be lightweight, maintainable and scalable. Two such protocols are HTTP and MQTT, both supported by the selected custom firmware, Tasmota. MQTT offers a clear advantage over HTTP for the application of this project and as such has been selected as the communication protocol of choice. A basic diagram of the publisher subscribe model is shown in the figure 2 below. For the full justification for this choice, see appendix 1.C.

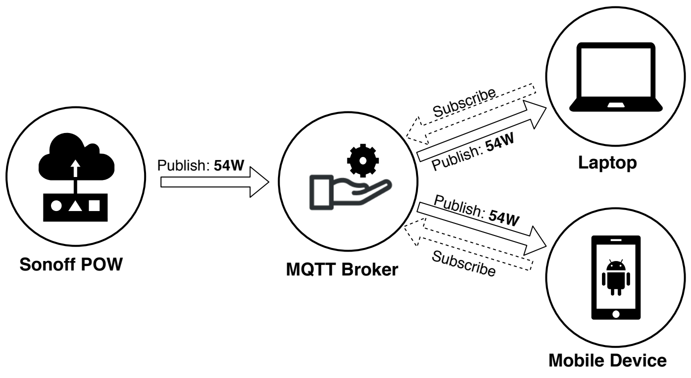


Fig. 2: MQTT System Diagram

6) Blockchain Related Research: A blockchain is required to facilitate decentralised, trustless energy consumption bookkeeping and auditing. A number of public and private blockchains are acceptable for this use case. For the purpose of this project, a private testnet will be used to simulate a consortium between trusted parties to create a permissioned blockchain. For this project, Ethereum will be used as the blockchain of choice. For a full justification, see appendix 1.D.

7) Connection of IoT Device to Blockchain: Next, one needs a way to connect the Sonoff to the blockchain. In an ideal case, with maximum decentralization attaining the highest degree of censorship resistance, each house within the microgrid is provided with a full Ethereum node. These nodes would attest their associated meter's energy consumption to the blockchain. In this implementation, the node itself implements some form of middleware discussed later on in this report, to bridge the off-chain and on-chain data sources of the Sonoff POW and blockchain logics.

8) Monitoring IoT Device From Blockchain: An Ethereum smart contract needs to have access to off-chain information in order to perform correct power consumption and production bookkeeping. Traditionally, this poses a problem as a blockchain is inherently trustless (by design) but an off-chain data source comes from a central entity and as such is not trustless. The normal blockchain approach is to use a centralized Oracle that is agreed by all vested parties to be trustworthy to provide information to the blockchain. The implementation required by meterBlock is, however, slightly different. The individual nodes themselves are the only entities that know about the power consumption of the devices that they connect to. There is no other source of truth, other than the nodes themselves.

This poses a problem such that nodes can be tampered with and as a result can report erroneous power consumption/production readings to the blockchain. There is no way for the other nodes to verify the integrity of the reported readings by the other nodes.

At this point, a fundamental assumption needs to be made: no one can ever modify or change the power meters. They are immutable and always report the correct power readings to the blockchain. This assumption is already made in existing prepaid power meters as there is a risk that individuals can bypass the meter or tamper with readings.

This assumption must also hold when considering the middleware as this can also be seen as a tamperable component of the communication protocol. For the purposes of this project, this will be in the form of a Python application using web3.py to interact directly with the blockchain. This python application will run on each management node and will be responsible for reading information directly from the meters(via MQTT) and transmitting transactions to the blockchain to interact with the management smart contracts.

9) Controlling IoT Device Using Smart Contract Logic: Controlling an off-chain device using an on-chain smart contract involves the nodes monitoring the state of a smart contract and then controlling the Sonoff POW in accordance with the output of the smart contracts. This will occur through the same middleware previously proposed wherein web3.py will be used in conjunction with an MQTT interface to control an off-chain asset with on-chain logic.

B. Preliminary Hardware Experimentation (16-07-2018 to 21-07-2018)

Once the appropriate hardware has been researched and selected, an understanding of the mechanisms through which the remote switch operates must be understood to fully capitalize on its functionality for the intended use case. This includes the purchasing of the intended remote switch and accompanying logic controllers, and experimenting with the stock firmware and user interface of the remote switch. Thereafter, the chosen custom firmware can be flashed onto the device and experimented with to ensure correct operation of the device before implementing the desired communication protocol.

C. Purchasing Selected Hardware

As per section IV.2, the selected remote switch and power measurement device is the Sonoff POW - a wifi enabled 15A relay switch that comes pre-configured with an energy consumption reporting mobile application. Three Sonoff POWs should be purchased to correctly simulate the microgrid application. In addition to the Sonoff POW, a Raspberry Pi Zero W single board computer (SBC) must be purchased to host the MQTT server that will facilitate communication and control between the Sonoff POW and the chosen blockchain. In the event that a custom device will be built to use instead of the Sonoff POW, additional hardware that includes current clamps and contactors must be purchased, however, these will not be used during the proof-of-concept phase of the project and can be neglected for the time being.

Preliminary research suggests that Micro-Robotics is the most cost effective distributer of all desired hardware components[8].

1) Experimentation with remote switch: To understand the operation of the Sonoff POW, a light bulb is connected to the device so that it can be toggled remotely through its preconfigured mobile application. Once connected, the device can be configured to the desired WIFI network by following the instructions outlined in the enclosed leaflet that comes with the device, and downloading their proprietary mobile application eWeLink[5].

Once configured, the state of the relay should be controllable through eWeLink, and the current power consumption, voltage and current draw should be visible through the app. Note, the Sonoff POW registers no energy consumption when connected to a energy saving CFL lamp due to its limited current draw. As such, a standard 50W incandescent lamp should be used to properly monitor and record energy consumption.

2) Flashing of Selected Firmware Onto Remote Switch: The stock firmware of the Sonoff POW and accompanying mobile application does not have a built in web API that can be used to relay the state of switch and power related data back to the blockchain. As a result, custom firmware must be flashed onto the device to provide the functionality necessary for use in the desired application. As per section IV.4, Tasmota is an open source alternative firmware variant for ESP8266 based devices such as the Sonoff POW. To leverage the functionality of Tasmota and the HTTP endpoint or custom MQTT broker, the device must be flashed with the custom firmware. To do this, a standard FTDI flashing module breakout board must be purchased (also available from Micro Robotics), and connected to the Sonoff POW. After downloading or cloning the git repo, the Sonoff POW can be put into flashing mode by connecting inserting the flashing module into the computer with the Sonoff POW toggle button held down. Consulting Tasmota's GitHub page presents the complete set of flashing instructions, including how to edit the config file to connect the device to the chosen network after flashing[9].

3) Experimentation with Custom Firmware: Once flashed, the device can be reconnected to the light and the custom web interface provided by Tasmota can be used to toggle the relay, as well as monitor the power consumption of the device connected to the Sonoff POW.

4) Access through HTTP API endpoint: Through the use of the API endpoint provided via Tasmota, one should be able to exert complete control over the Sonoff pow. Predefined endpoints are outlined in the Tasmota documentation and can be tested for full functionality.
This functionality includes the ability to:

- Control the relay state (toggle or set to a predefined state)
- Read current energy consumption and related measurements (voltage, current, power and time stamp of the reading)
- Change device specific settings and configurations such as power measurement resolution or MQTT settings

Once completed, the system should be capable of toggling the relay and relaying the current energy consumption and related data of the Sonoff POW through use of the API endpoint. However, the Pub/Sub model adopted through use of MQTT is preferred over the client-server model of HTTP in IoT connected devices, and as such, the system should now be configured to make use of this.

D. Preliminary Software Experimentation (23-07-2018 to 28-07-2018)

At this point, the hardware should be able to communicate with the stock applications provided by the hardware manufacturers. The remote switch of choice, such as a Sonoff Pow, should be remote controllable over the internet. Additionally, some form of custom firmware should be configured, enabling access over an HTTP API endpoint.

This milestone aims to explore the usage of the custom firmware installed in the previous milstone. A communication protocol should be selected and implemented to control a number of remote switches. Experimentation with communication protocol As discussed before, a communication protocol is required to interact with the Sonoff Pows. MQTT is the chosen protocol for all communication with the device.

Preliminary MQTT experimentation should involve the use of a publically accessible MQTT broker, such as those provided by [10]. For testing this configuration, a local MQTT client is required to be set up on the testing computer. For simplicity, it is recommended to use MQTT-Spy as this comes pre configured with a number of MQTT brokers[11]. The Sonoff Pows should also be configured to access the same MQTT broker.

Controlling the Sonoff Pow via MQTT involves broadcasting a command on a particular topic within the MQTT channel. The commands available can be found on the Tasmota documentation page.

Monitoring Remote power consumption follows the same principle but instead of broadcasting a command to the topic channel one must now subscribe to a particular channel as defined by the Tasmota documentation.

1) Set up local MQTT server: It is ideal to run your own MQTT server, rather than running on a public, open one as all traffic is visible to all members on a the channel. Clearly, this is not ideal as this enables anyone to read and/or control your devices.

To this end, a local MQTT server should be set up. For this project, a Raspberry Pi zero W has been selected as it is low cost and easy to set up. Raspbian Lite is the operating system of choice as it lightweight, has no desktop environment (not required for our purposes) and easy to configure to automatically connect to wifi networks while enabling SSH.

E. Blockchain Related Software Implementation (30-07-2018 to 04-08-2018)

Next, the blockchain smart contract logic for performing accounting and bookkeeping needs to be implemented. In addition, the middleware required to link to the MQTT controlling network needs to be written. Smart contracts will be written in Solidity.

1) Create basic smart contract: The contract model closely follows that of an ERC20 token wherein a mapping is created between an address and an unsigned integer. The address corresponds to the owner's Ethereum address and the unsigned integer corresponds to the individual's balance of tokens. meterBlock implementation varies from the ERC20 token standard by allowing a verified external contract to add/remove tokens from an account. This enables the meter's node to remove tokens on power consumption and add tokens on power creation.

Apon energy consumption, the meter informs the node which intern informs the smart contract of the consumption of power. This results in proportional the deallocation of tokens.

Next, Logic to control switch state based on power consumed is required. To achieve this, the nodes need to monitor the smart contract and when a particular condition is met, such as depletion of an accounts tokens, the load should be switched off.

Finally, one needs the ability to fund existing accounts with new tokens. An interface should be created to facilitate this by creating a website where an account address can be specified to fund with a particular number of tokens.

It is important that any application that deals with the transfer of value, extensive uniting testing should be written to ensure that the application functions as designed. As a result, all functionality should include a suit of tests that will be written by making use of the Truffle Framework.

F. Create Middleware to Connect Blockchain to MQTT server (06-08-2018 to 11-08-2018)

As discussed before, middleware is required to interconnect the MQTT network to the blockchain. This involves tying two different protocols together in one common interface. To achieve this, a Python application must be created using web3.py to interact with the blockchain and an MQTT client to communicate with the Sonoff pows. The MQTT client will read information from the MQTT topic channel, such as power consumed by different households, and report this information periodically to the blockchain through the publication of an Ethereum Smart contract interaction.

At this point in the implementation, one token (henceforth referred to as a Krag token) corresponds to one Kwh. If the user consumers one Kwh of energy, they lose one Krag token from their wallet. This process occurs via the aforementioned communication process.

Once all Krag tokens have been depleted, the node informs the MQTT broker via the middleware that the load should be switched off and the sonoff responds appropriately.

G. Software User interface Implementation (13-08-2018 to 18-08-2018)

A simple user interface is required to provide households with the ability to interact with the application. This application should have a web interface to allow the user to login and view their token balanced, fund their account with new tokens and gain insight into their current usage. This interface should be completely client side without any server side logic, providing access to `web3.js` via Metamask. Ideally, this application should be built using `Vue.js` for simplistic modular design and utilize `Bootstrap4` for consistent styling.

`Metamask` provides the ability to connect a standard web browser, such as Google Chrome, to the Ethereum blockchain via a predefined RPC endpoint. This RPC endpoint will be the Ethereum node hosted on the Raspberry Pi Zero w to simulate the implementation of an Ethereum consortium. Metamask injects `web3.js` (a Javascript library to provide the ability to interact with Ethereum smart contracts) directly into the web browser.

Vuejs is a progressive javascript framework from making scalable, lightweight, modular, production ready applications. Other Frameworks, such as Angular, React or Meteor could be used but Vue is simple and easy to pick up and is recommended for new developers.

H. Intended Use Case Energy Exchange Conceptualisation (20-08-2018 to 25-08-2018)

As discussed from the onset, full use-case application of this system includes scaling the design to be used by all households, businesses, (all producers and consumers) in a national grid. This implementation requires understanding and conceptualisation of the economic forces that control the price per kWh such that the network can dynamically adjust the price based on the current status of the grid.

1) Modelling end user as both consumer and producer: The most general use case of this technology with respect to its final application would be to have one individual household that can either produce or consume energy. A simple input-output model of this can be used to correctly account for the flow of tokens and their associated value based on whether or not the individual is in surplus or deficit of energy(ie. using more energy than generating, or generating more energy than consuming). This abstraction can then be used to model the interaction of these individual households between themselves, and how these tokens are exchanged for power based on current demand.

There are a number of apparent models that can accommodate dynamic price discovera capable of providing a national marketplace for energy exchange. Implementation of these modes are outside of the scope of this project as they are seen as an economic, application level problem. The underpinning technology, built up to this point, can accommodate both variants. However, to fully realise application of this system, these models should be explored.

2) Energy Exchange Modelled as a Traditional Crypto Asset Marketplace: This implementation sees the use of a traditional two sided exchange between a buyer and seller. The current price is defined by the highest buy price and the lowest sell price. The notion of market depth and volume apply in the traditional sense. The buying/selling of energy differs from that of a traditional crypto asset in that there is an additional dimension to consider: time. This means when you sell energy, you need to commit to providing a certain amount of power, for a certain period of time. As a result, energy needs to be packaged into bundles that can then be bought and sold on the exchange. This ensures that the correct amount of power is provided for the mutually defined period of time. An escrow mechanism can be used wherein the seller of energy locks up a proportional amount of tokens equal to the power being sold for the duration of the energy provision contract. In the even that the seller does not hold up her end of the contract (does not provide the correct power for the stipulated period of time), the escrow is used to compensate the buyer.

3) Non-linear bonded Curation Markets: Other applications could involve the use of non-linear bonding curves used to define the relationship between price and volume. This facilitates the creation of a one sided exchange where in you do not require a buyer and seller to exchange power but rather utilize an autonomise price discovery mechanism that leverages predefined commonly agreed upon logic. These bonding curves require complex financial mathematics and are left as an extension to the project research.

V. Conclusion

This report outlined the full scale implementation of a DLT governed device capable of measuring, monitoring and managing the energy consumed or produced by a typical household. Comparisons to the existing models were drawn to provide a justification for the proposed implementation. An extensive, detailed project plan was also provided outlined weekly steps required to achieve the final prototype.

Appendix 1

A. Grid+

Grid+ leverages the Ethereum blockchain to give consumers direct access to wholesale energy markets. In their implementation, Grid+ buys energy wholesale and resells it to the consumer at a lower price than the competition. They use blockchain as the auditing mechanism, to provide the notion of self sovereign ownership over one's power tokens. Their implementation uses a stable coin, backed by the US dollar, known as the BOLT token.

B. Tasmota Firmware Justification

There are three main firmwares to choose from to accommodate the requirements of the project, naimly: Tasmota, Espeasy and Espurna[6], [5], [9], [12], [13]. Each of these can be flashed onto a Sonoff and provide a selection of additional functionality. The primary functionality required is the ability to control the Sonoffs via a HTTP API or over MQTT with a custom broker. All three firmwares offer this functionality and as such the simplest, most lightweight, supported firmware will be used. To this end, Tasmota has been selected as the easiest firmware to achieve the desired functionality.

C. HTTP vs MQTT

HTTP offers the ability to interact directly with the Sonoffs via API endpoints. These API's enable complete control over the device and could work with some form of middleware to enable the Sonoffs to be connected to a blockchain. HTTP works using the notation of a client-server model. This requires the knowledge of the address of the individual sonoffs in order to interact and control them. HTTP is considered to be a document-centric protocol, for transmitting files with a defined start and end, such as a web page over the internet. Streamed data is required to be broken up into packets before transmision.

MQTT on the other hand is considered data centric and uses a client server publish/subscribe pattern. It is designed to be as lightweight as possible and has shown performance improvements of up to 93 times over that of HTTP when used on a 3G network [14].

The publish/subscribe (pub/sub) pattern decouples a client who is sending a messaged (a publisher) from other clients while receiving messages(a subscriber). As a result, the publisher and subscriber do not require to know about each other in communicating over the channel. The last component of the MQTT protocol is called a broker which facilitates the communication between publishers and subscribers. Both the publisher and subscriber need to know about the broker to facilitate this communication. The diagram below outlines the basics of this protocol.

There are a number of advantages associated with the decoupling of a publisher and subscriber, primarily:

- Space decoupling: neither the publisher nor the subscriber need to know about each other. They dont require IP addresses or ports to communicate in the same way that HTTP does.
- Time decoupling: the subscriber does not need to be online for a publisher to transmit a message facilitating temporal decoupling over the communication channel.
- Synchronization decoupling: if the publisher or subscriber is offline, the communication channel does not halt
- Scalability: pub/sub offers far higher scalability than client server due to its ability to break down messages into parallelized components
- Message Filtering: multiple device types can run on the same broker with the subscriber specifying a particular type of message it is interested in, filtering out the rest

D. Select Blockchain of Choice

1) Bitcoin: is the most popular and well known blockchain. It does however have very limited scripting abilities and does not offer a Turing complete programing language capable of providing dynamic autonomous execution of contract code. It's functionality is limited to the transfer of value from one party to another [15]. As a result, if the Bitcoin blockchain was used, heavy modification to the underlying codebase would be required. Moreover, bitcoin can only facilitate three transactions per second, has high transaction fees and extremely slow transaction processing times (up to twelve hours to process a single transaction). As such, payment channels would also be required to be implemented if Bitcoin was to be used. Lastly, Bitcoin uses Proof-of-work as its consensus algorithm making it inherently inefficient at reaching consensus[15]. This ideology is diametrically opposed to that of meterBlock.

2) IOTA: is designed to be more scalable than traditional blockchains, with no theoretical maximum throughput. Additionally, it has no miner fees as there are no miners within the IOTA network. Rather, each node within the network performs the action that a traditional miner would do when processing a transaction. IOTA does not have blocks and as a result, no chain [16]. It is a stream of interlinked transactions that are distributed and stored across the network through a data structure called a Tangle, a form of a DAG. Despite the apparent advantages of IOTA, it does not offer any platform to create smart contracts and as such would require the use of extensive modification to accommodate any form of complex token exchange settlements.

3) Cardano: is a distributed blockchain computing platform. It uses proof-of-stake, making it inherently more efficient than any other proof-of-work alternative [17]. Cardano does plan to offer smart contracts in the future but in its current implementation it does not have this functionality and so it is not applicable for meterBlock.

4) NEO: formally known as Antshares, is seen as the Chinese version of Ethereum by many. Fundamentally, it offers much of the same functionality as Ethereum with full smart contract support. Moreover, NEO's smart contract platform is language agnostic meaning that any turing complete programming language can compile down to NEO's smart contract bytecode. NEO uses delegated byzantine fault tolerance (dBFT) as it's consensus algorithm[18]. This makes it efficient and fast but this results in the consensus becoming more centralised than other platforms and as such makes it more susceptible to censorship. Additionally, NEO has a limited English speaking developer base making the interaction with the documentation cumbersome at times. Lastly, NEO's platform is still under development and is not considered production ready.

5) Ethereum: is a blockchain platform offering turing complete smart contracts. It has the biggest developer mindshare by an order of magnitude. Ethereum is currently used in more smart contract related projects than any other blockchain, with 96 out of the top 100 token based applications running on the Ethereum blockchain[16]. Ethereum is currently using Proof-Of-Work, but will soon move to Proof-Of-Stake under the Casper update making it far more scalable and efficient. Ethereum offers the ability to easily run a consortium blockchain between trusted nodes, making the setup and operation of a private blockchain advantageous for testing and development. Lastly, Ethereum has extensive documentation and is currently considered production ready.

REFERENCES

[1] Eskom, "Prepayment History." [Online]. Available: http://www.prepayment.eskom.co.za/history.asp

[2] Gridplus, "Grid+ White paper," 2017. [Online]. Available: https://gridplus.io/assets/Gridwhitepaper.pdf

[3] Solaradvice, "Can I Sell Electricity in South Africa? - Solar Advice." [Online]. Available: https://solaradvice.co.za/can-i-sell-electricity-in-south-africa/

[4] Itead.cc, "Sonoff Pow - ITEAD Wiki," 2018. [Online]. Available: https://www.itead.cc/wiki/Sonoff{_}Pow

[5] Sonoff, "eWeLink Apps on Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.coolkit

[6] Mozilla, "HTTP — MDN." [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP

[7] HiveMQ, "HiveMQ - Enterprise MQTT Broker." [Online]. Available: https://www.hivemq.com/

[8] MicroRobotics, "Sonoff POW Wifi Switch + PWR Meter - Micro Robotics." [Online]. Available: https://www.robotics.org.za/IM160810001?search=sonoffpow

[9] Arendst, "Sonoff-Tasmota," 2018. [Online]. Available: https://github.com/arendst/Sonoff-Tasmota

[10] HiveMQ, "MQTT Essentials Part 2: Publish & Subscribe." [Online]. Available: https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe

[11] Kamilfb, "MQTT-Spy," 2016. [Online]. Available: https://github.com/kamilfb/mqtt-spy/wiki/Downloads

[12] Letscontrolit, "ESPEasy - Let's Control It." [Online]. Available: https://www.letscontrolit.com/wiki/index.php/ESPEasy

[13] Xoseperez, "Espurna," 2018. [Online]. Available: https://github.com/xoseperez/espurna

[14] Marina Serozhenko, "MQTT vs. HTTP: which one is the best for IoT? MQTT Buddy Medium." [Online]. Available: https://medium.com/mqtt-buddy/mqtt-vs-http-which-one-is-the-best-for-iot-c868169b3105

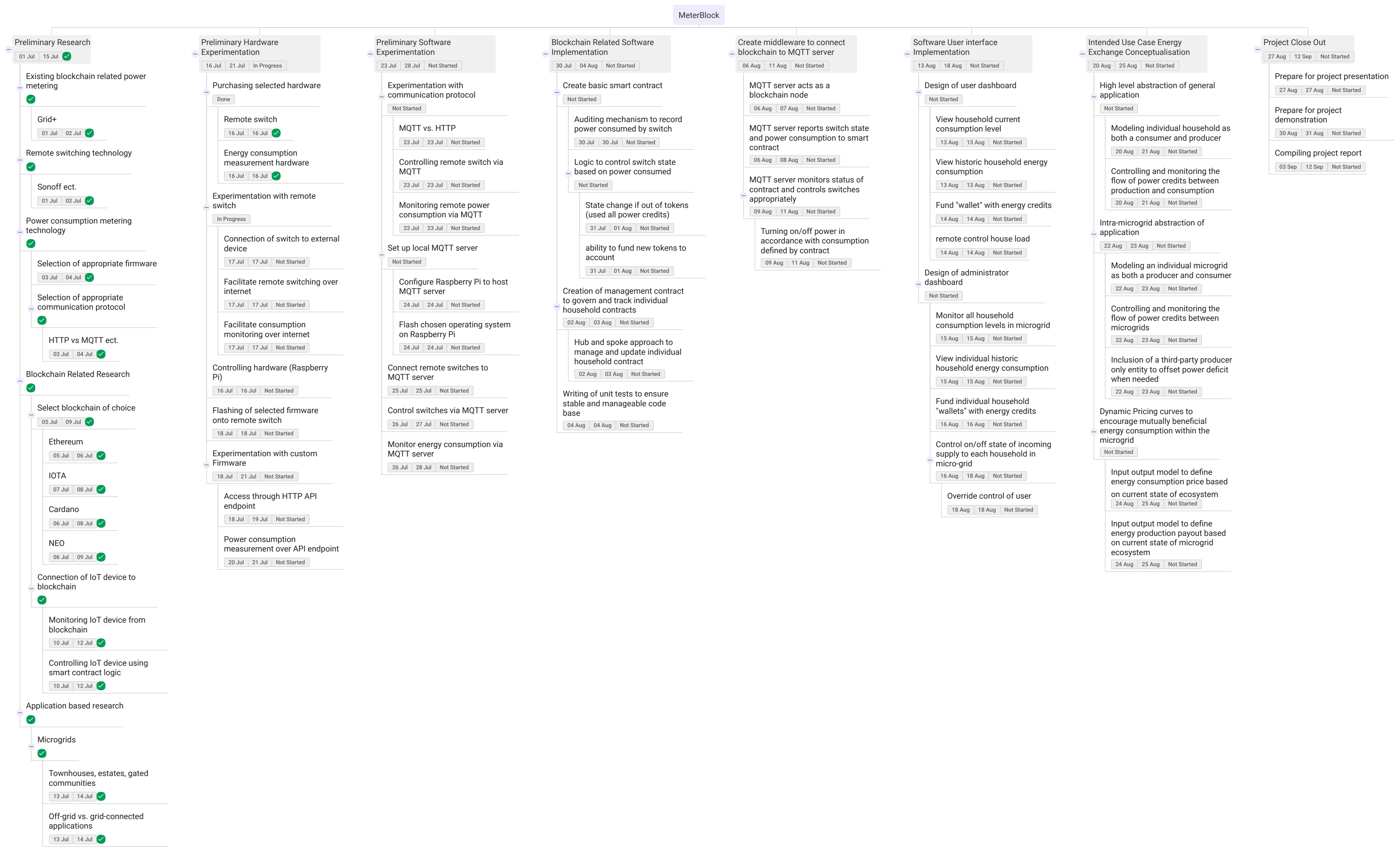
[15] Bitcoin, "Getting Started with Bitcoin Bitcoin.com." [Online]. Available: https://www.bitcoin.com/getting-started

[16] EthereumFoundation, "Ethereum Project," 2018. [Online]. Available: https://www.ethereum.org/

[17] Cardano, "Cardano - Home of the Ada cryptocurrency and technological platform." 2018. [Online]. Available: https://www.cardano.org/en/home/

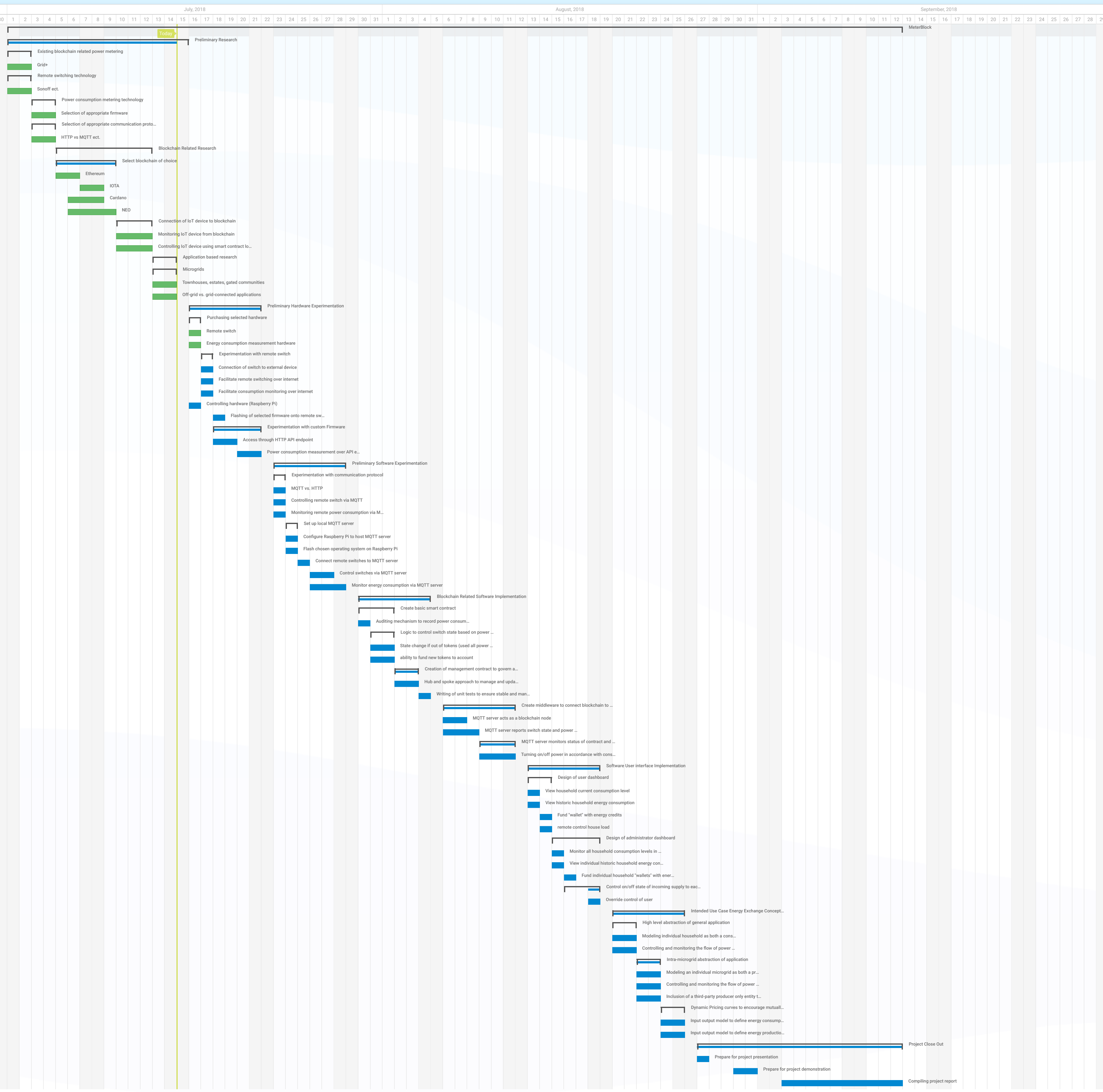
[18] NEO, "NEO Smart Economy," 2018. [Online]. Available: https://neo.org/

Appendix 2.A



- MeterBlock
  - Preliminary Research — 01 Jul | 15 Jul | ✓
    - Existing blockchain related power metering — ✓
      - Grid+ — 01 Jul | 02 Jul | ✓
    - Remote switching technology — ✓
      - Sonoff ect. — 01 Jul | 02 Jul | ✓
    - Power consumption metering technology — ✓
      - Selection of appropriate firmware — 03 Jul | 04 Jul | ✓
      - Selection of appropriate communication protocol — ✓
        - HTTP vs MQTT ect. — 03 Jul | 04 Jul | ✓
    - Blockchain Related Research — ✓
      - Select blockchain of choice — 05 Jul | 09 Jul | ✓
        - Ethereum — 05 Jul | 06 Jul | ✓
        - IOTA — 07 Jul | 08 Jul | ✓
        - Cardano — 06 Jul | 08 Jul | ✓
        - NEO — 06 Jul | 09 Jul | ✓
      - Connection of IoT device to blockchain — ✓
        - Monitoring IoT device from blockchain — 10 Jul | 12 Jul | ✓
        - Controlling IoT device using smart contract logic — 10 Jul | 12 Jul | ✓
    - Application based research — ✓
      - Microgrids — ✓
        - Townhouses, estates, gated communities — 13 Jul | 14 Jul | ✓
        - Off-grid vs. grid-connected applications — 13 Jul | 14 Jul | ✓
  - Preliminary Hardware Experimentation — 16 Jul | 21 Jul | In Progress
    - Purchasing selected hardware — Done
      - Remote switch — 16 Jul | 16 Jul | ✓
      - Energy consumption measurement hardware — 16 Jul | 16 Jul | ✓
    - Experimentation with remote switch — In Progress
      - Connection of switch to external device — 17 Jul | 17 Jul | Not Started
      - Facilitate remote switching over internet — 17 Jul | 17 Jul | Not Started
      - Facilitate consumption monitoring over internet — 17 Jul | 17 Jul | Not Started
    - Controlling hardware (Raspberry Pi) — 16 Jul | 16 Jul | Not Started
    - Flashing of selected firmware onto remote switch — 18 Jul | 18 Jul | Not Started
    - Experimentation with custom Firmware — 18 Jul | 21 Jul | Not Started
      - Access through HTTP API endpoint — 18 Jul | 19 Jul | Not Started
      - Power consumption measurement over API endpoint — 20 Jul | 21 Jul | Not Started
  - Preliminary Software Experimentation — 23 Jul | 28 Jul | Not Started
    - Experimentation with communication protocol — Not Started
      - MQTT vs. HTTP — 23 Jul | 23 Jul | Not Started
      - Controlling remote switch via MQTT — 23 Jul | 23 Jul | Not Started
      - Monitoring remote power consumption via MQTT — 23 Jul | 23 Jul | Not Started
    - Set up local MQTT server — Not Started
      - Configure Raspberry Pi to host MQTT server — 24 Jul | 24 Jul | Not Started
      - Flash chosen operating system on Raspberry Pi — 24 Jul | 24 Jul | Not Started
    - Connect remote switches to MQTT server — 25 Jul | 25 Jul | Not Started
    - Control switches via MQTT server — 26 Jul | 27 Jul | Not Started
    - Monitor energy consumption via MQTT server — 26 Jul | 28 Jul | Not Started
  - Blockchain Related Software Implementation — 30 Jul | 04 Aug | Not Started
    - Create basic smart contract — Not Started
      - Auditing mechanism to record power consumed by switch — 30 Jul | 30 Jul | Not Started
      - Logic to control switch state based on power consumed — Not Started
        - State change if out of tokens (used all power credits) — 31 Jul | 01 Aug | Not Started
        - ability to fund new tokens to account — 31 Jul | 01 Aug | Not Started
    - Creation of management contract to govern and track individual household contracts — 02 Aug | 03 Aug | Not Started
      - Hub and spoke approach to manage and update individual household contract — 02 Aug | 03 Aug | Not Started
    - Writing of unit tests to ensure stable and manageable code base — 04 Aug | 04 Aug | Not Started
  - Create middleware to connect blockchain to MQTT server — 06 Aug | 11 Aug | Not Started
    - MQTT server acts as a blockchain node — 06 Aug | 07 Aug | Not Started
    - MQTT server reports switch state and power consumption to smart contract — 06 Aug | 08 Aug | Not Started
    - MQTT server monitors status of contract and controls switches appropriately — 09 Aug | 11 Aug | Not Started
      - Turning on/off power in accordance with consumption defined by contract — 09 Aug | 11 Aug | Not Started
  - Software User interface Implementation — 13 Aug | 18 Aug | Not Started
    - Design of user dashboard — Not Started
      - View household current consumption level — 13 Aug | 13 Aug | Not Started
      - View historic household energy consumption — 13 Aug | 13 Aug | Not Started
      - Fund "wallet" with energy credits — 14 Aug | 14 Aug | Not Started
      - remote control house load — 14 Aug | 14 Aug | Not Started
    - Design of administrator dashboard — Not Started
      - Monitor all household consumption levels in microgrid — 15 Aug | 15 Aug | Not Started
      - View individual historic household energy consumption — 15 Aug | 15 Aug | Not Started
      - Fund individual household "wallets" with energy credits — 16 Aug | 16 Aug | Not Started
      - Control on/off state of incoming supply to each household in micro-grid — 16 Aug | 18 Aug | Not Started
        - Override control of user — 18 Aug | 18 Aug | Not Started
  - Intended Use Case Energy Exchange Conceptualisation — 20 Aug | 25 Aug | Not Started
    - High level abstraction of general application — Not Started
      - Modeling individual household as both a consumer and producer — 20 Aug | 21 Aug | Not Started
      - Controlling and monitoring the flow of power credits between production and consumption — 20 Aug | 21 Aug | Not Started
    - Intra-microgrid abstraction of application — 22 Aug | 23 Aug | Not Started
      - Modeling an individual microgrid as both a producer and consumer — 22 Aug | 23 Aug | Not Started
      - Controlling and monitoring the flow of power credits between microgrids — 22 Aug | 23 Aug | Not Started
      - Inclusion of a third-party producer only entity to offset power deficit when needed — 22 Aug | 23 Aug | Not Started
    - Dynamic Pricing curves to encourage mutually beneficial energy consumption within the microgrid — Not Started
      - Input output model to define energy consumption price based on current state of ecosystem — 24 Aug | 25 Aug | Not Started
      - Input output model to define energy production payout based on current state of microgrid ecosystem — 24 Aug | 25 Aug | Not Started
  - Project Close Out — 27 Aug | 12 Sep | Not Started
    - Prepare for project presentation — 27 Aug | 27 Aug | Not Started
    - Prepare for project demonstration — 30 Aug | 31 Aug | Not Started
    - Compiling project report — 03 Sep | 12 Sep | Not Started

Appendix 2.B


July, 2018
August, 2018
September, 2018
Today
MeterBlock
Preliminary Research
Existing blockchain related power metering
Grid+
Remote switching technology
Sonoff ect.
Power consumption metering technology
Selection of appropriate firmware
Selection of appropriate communication proto...
HTTP vs MQTT ect.
Blockchain Related Research
Select blockchain of choice
Ethereum
IOTA
Cardano
NEO
Connection of IoT device to blockchain
Monitoring IoT device from blockchain
Controlling IoT device using smart contract lo...
Application based research
Microgrids
Townhouses, estates, gated communities
Off-grid vs. grid-connected applications
Preliminary Hardware Experimentation
Purchasing selected hardware
Remote switch
Energy consumption measurement hardware
Experimentation with remote switch
Connection of switch to external device
Facilitate remote switching over internet
Facilitate consumption monitoring over internet
Controlling hardware (Raspberry Pi)
Flashing of selected firmware onto remote sw...
Experimentation with custom Firmware
Access through HTTP API endpoint
Power consumption measurement over API e...
Preliminary Software Experimentation
Experimentation with communication protocol
MQTT vs. HTTP
Controlling remote switch via MQTT
Monitoring remote power consumption via M...
Set up local MQTT server
Configure Raspberry Pi to host MQTT server
Flash chosen operating system on Raspberry Pi
Connect remote switches to MQTT server
Control switches via MQTT server
Monitor energy consumption via MQTT server
Blockchain Related Software Implementation
Create basic smart contract
Auditing mechanism to record power consum...
Logic to control switch state based on power ...
State change if out of tokens (used all power ...
ability to fund new tokens to account
Creation of management contract to govern a...
Hub and spoke approach to manage and upda...
Writing of unit tests to ensure stable and man...
Create middleware to connect blockchain to ...
MQTT server acts as a blockchain node
MQTT server reports switch state and power ...
MQTT server monitors status of contract and ...
Turning on/off power in accordance with cons...
Software User interface Implementation
Design of user dashboard
View household current consumption level
View historic household energy consumption
Fund "wallet" with energy credits
remote control house load
Design of administrator dashboard
Monitor all household consumption levels in ...
View individual historic household energy con...
Fund individual household "wallets" with ener...
Control on/off state of incoming supply to eac...
Override control of user
Intended Use Case Energy Exchange Concept...
High level abstraction of general application
Modeling individual household as both a cons...
Controlling and monitoring the flow of power ...
Intra-microgrid abstraction of application
Modeling an individual microgrid as both a pr...
Controlling and monitoring the flow of power ...
Inclusion of a third-party producer only entity t...
Dynamic Pricing curves to encourage mutuall...
Input output model to define energy consump...
Input output model to define energy productio...
Project Close Out
Prepare for project presentation
Prepare for project demonstration
Compiling project report