IoT Blockchain

Smart Meters

Functional Specification

Version 0.1

Ying Lin

(ying.lin@sjsu.edu)

[Version History](#h.gjdgxs)

[Introduction](#h.30j0zll)

[References](#h.1fob9te)

[Requirements](#h.3znysh7)

[Functional Overview](#h.2et92p0)

[Configuration/ External Interfaces](#h.tyjcwt)

[Debug](#h.3dy6vkm)

[Logging](#h.1t3h5sf)

[Counters](#h.4d34og8)

Implementation

[Testing](#h.17dp8vu)

[General Approach](#h.3rdcrjn)

[Unit Tests](#h.26in1rg)

[Appendix](#h.lnxbz9)

# Version History

|  |  |
| --- | --- |
| **Version** | **Changes** |
| 0.1 | Initial version |

# Introduction

Solidity is a development language for developing distributed decentralized applications (Dapps) on the Ethereum Virtual Machine (EVM). Its syntax is very similar to JavaScript but is statically typed.

As we are developing our IoT devices on the blockchain provided by Ethereum, it will be in the form of a Dapp that is running on the EVM. Therefore, it is natural to use solidify as our main development language.

The application will be centered around a smart contract that handles transactions between households who sell excessive solar power or buy them when necessary.

# References

<https://solidity.readthedocs.io/en/develop/>

<http://ethdocs.org/en/latest/introduction/what-is-ethereum.html>

# Requirements

Solidity programming language

Web 3 compiler

Ethereum Virtual Machine (provided by go-ethereum)

# Functional Overview

There are two parts in our system, a smart contract and a few smart meters. I will design and implement the smart meters. A smart meter is connected to a few sensors that reports the power consumption and production at a household. Depending on how much power is being produced vs consumed, it can decide whether to buy or sell power from the smart contract. The meter owner can set up a price range for selling and buying power so that smart meter can send to the smart contract to trade with other smart meters.

## Configuration/ External Interfaces

* No external configuration required
* Dependency  
  - successfully set up Ethereum Virtual Machine on every participating node (computers)  
  - We may not be able to implement the real power sensor. We are looking at using artificial values in our program to simulate the values for now.

## Debug

Debugging will be done as we develop the app. Mostly by logging every transaction to the console.

### Logging

We will regularly log the power consumption and production at any given moment. When a buy or sell request is sent to the smart contract, we will log the request and response to make sure they are behaving as expected.

### Counters

Not used.

# Implementation

1. A smart meter will first accept a buying price range and selling price range from the owner. The function will be:  
   - set\_range(lowest\_buy, highest\_buy, lowest\_sell, highest\_sell)
2. A smart meter will be able to read the power consumption and production values from the sensors and send buy or sell request to the smart contract. This will be handled by a function called update().
3. In update(), two functions, buy() and sell() will be called. They will decide a price and send the request to the smart contract, together with the amount being purchased/sold.

# Testing

## General Approach

Testing the functionalities of our smart contract and smart meters will focus on how the smart meters interact with the smart contract under different circumstances:

* + - 1. When power consumption is higher than production, the meter will send a buy request.
      2. When power consumption Is lower than production, the meter will send a sell request.
      3. The meter will send buy requests starting from the lowest price and raise the price if there is no offer.
      4. The meter will send sell requests starting from the highest price and lower the price if no one is buying
      5. The owner should be able to set the price ranges at any time.

## Unit Tests

Ethereum and Solidity do not support unit test, as far as we know. We are still looking for solutions.

# Appendix

<https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial>

<https://blog.ethereum.org/2016/07/12/build-server-less-applications-mist/>