**Blockchain and Smart Contract Payment System for EV Charging**

**Share&Charge**

Built the blockchain platform to share personal EV charging stations to other EV owners. Share&Charge combines the blockchain and EV charging by using the blockchain to manage the charging process. It has a blockchain base application that allows EV owners getting access to personal EV charging stations and then making payments for the charging in a peer-to-peer method.

The purpose of building an electronic payment system using blockchain and smart contracts is as follows: it allows one to manage transactions, such as validating and automating payments.

**Smart contract:** Controls the charging process in the charging station

**Benefits of using blockchain and smart contract:** It helps getting rid of the central payment system because every transaction is open to the public so the EV can charge and pay in a peer-to-peer manner.

Blockchain used in the paper was **Ethereum.**

* Ethereum has a smart contract feature to create programs to manage and control hardware for the charging process
* It has a small transaction time of about 15-20 seconds

Blockchain nodes are separated into two types:

* Application node: The node for the customer to charge their electric vehicle
* Station node: The node for the charging station to manage the charging process

**Managing IoT Devices using Blockchain Platform**

The following paper: “Managing IoT Devices using Blockchain Platform.” presents a method to manage hardware resources such as a Raspberry Pi, using Ethereum blockchain by programming a smart contract to control the privacy of electronic equipment and deploy smart contract on Ethereum blockchain, then set up the Ethereum node on the RPi.

As a result, the Raspberry Pi could control the electronic equipment policy as programmed into the smart contract. This solution can be applied to the smart contract in this project to manage control charging station with RPi.

**Thing-to-Thing Electricity Micro Payments using Blockchain Technology**

The following research paper: Thing-to-Thing Electricity Micro Payments Using Blockchain Technology” presents a method to enable micropayments between devices without human interaction using blockchain technology by using RPi with blockchain node to manage payment and control electricity supply with a relay.

**Blockchain**

Blockchain is a decentralised database that stores data in several local nodes. Each local nodes has the same data, called “Blockchain”. Each block in a blockchain refers to the previous block, and hence creating a “chain” of blocks that contain data. Inside each block, there is a set of transactions that were created.

**Go-Ethereum**

Go-Ethereum is the command line interface (CLI) for running a full Ethereum node, implemented in Go. The functions of Go-Ethereum include:

* Mining real Ether
* Transferring funds between addresses
* Creating contracts
* Sending transactions
* Exploring block history

Go-Ethereum has 3 interfaces:

* Javascript console including web3 javascript API
* JSON-RPC server
* Command Line Options

**Smart Contract**

It can be defined as a protocol that computerises transactions such as payment terms, liens, confidentiality and even enforcement.

**Objective of Smart Contracts:** Manage a common contractual condition and to minimise the need for trusted intermediaries.

In the context of blockchain, smart contracts are programs that are deployed into blockchain and mined as transactions, so the contracts can’t be edited or deleted.

This program is invoked from each node in the blockchain by receiving code from the node, executing the given code, and storing the output in the blockchain. **Solidity** is used to write the smart contract to manage and control the charging process.

**Proposed System Architecture**

In the proposed system architecture, the nodes are separated into two: Application nodes and station nodes.

Application node - set up on the server, this node is used by the user web application to order the charging. This node is also used as the mining node to mine all the transactions in the system network. Each transaction created in this node is verified by the owner of each account wallet. This node is also used to transfer ETH tokens from EV owner to the charging station owner, and also validates the ETH transfer and command to start or stop charging.

The station node is set up in the charging station using a RPi to control the charging process. The transactions created in this node calculate the amount of ETH tokens from the data read from the EV, confirms charging and controls charging.

The web3 library has some very important functions, such as:

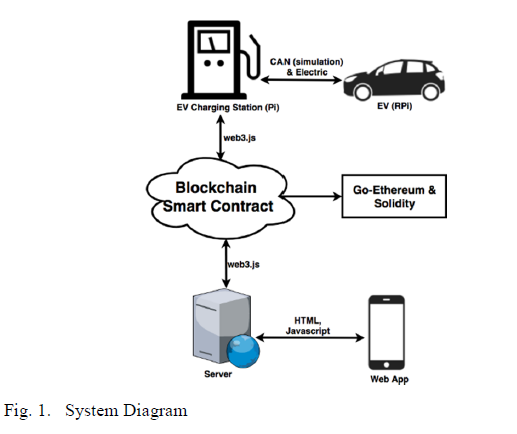
* Account management
* ETH transferring
* Transaction reading

It also enables the creation of customised smart contracts, which can be deployed in blockchains for the application and station nodes to invoke, to execute functions in the smart contract.

Some of the important variables found inside the smart contract are:

* Electric power
* ETH to pay
* Price rate
* Status
* Account name
* Account address

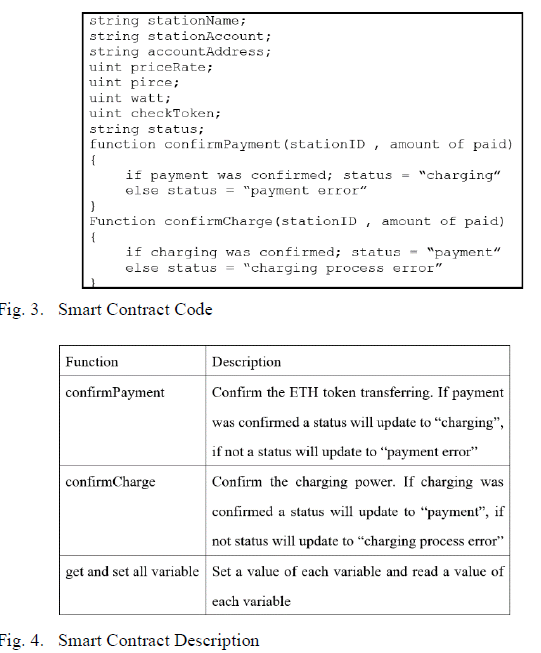
Smart contracts have getter and setter methods to read and edit these variables, as well as functions for payment validation and charging confirmation.

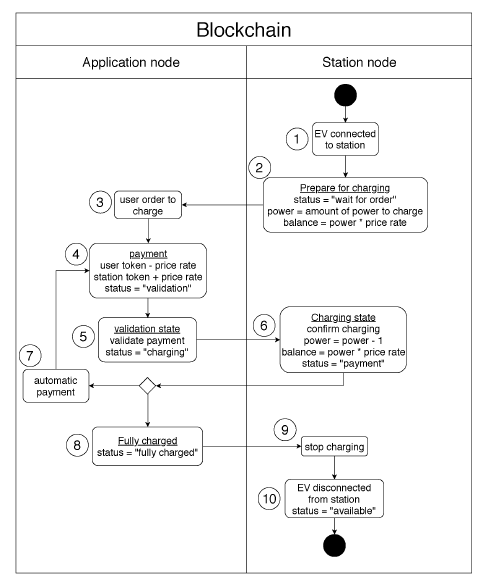


**System workflow**

1. The EV connects to the charging station
2. Prepare for charging:
   1. EV sends charging data to the charging station
   2. Station node creates transaction as follows:
      1. Status = “wait for paid”
      2. Power = amount of power to charge
      3. Balance = power\*price rate
   3. The variables that were created by the station node will be read by the application node and will be displayed in the web application.
3. User orders to charge
4. Application creates a node as follows:
   1. Transfer ETH from user account to station account
   2. Status = “validation”
5. Once the status reads “validation”, the application node creates a transaction by calling the smart contract function to validate the payment. Once the payment validation is successful, the status changes to “charging”
6. When the status reads “charging”, the charging of the EV will begin. Once 1kW of charge has been transferred, the station node reads the battery value and creates a transaction to call the smart contract and confirm charging. Once the confirmation is successful, the status changes to “payment”, and the following variables will be set:
   1. Power = power – 1
   2. Balance = power\*price rate
7. If the power variable is above 0, the application node continues to the payment state.
8. If the power variable is equal to 0, the application node will create a transaction: status = “fully charged”
9. Once the status is “fully charged”, the charging station will stop charging
10. The charging station disconnects from the EV, and station node creates a transaction as follows:
    1. Status = “available”
    2. When the status is “available”, if the user wishes to stop charging, the application node will create a transaction:
       1. Status = “stopping”
    3. The charging station will then stop charging and station node creates a node as follows:
       1. Status = “stopped”

**Source code and description of smart contract**

****



**System Behaviour**

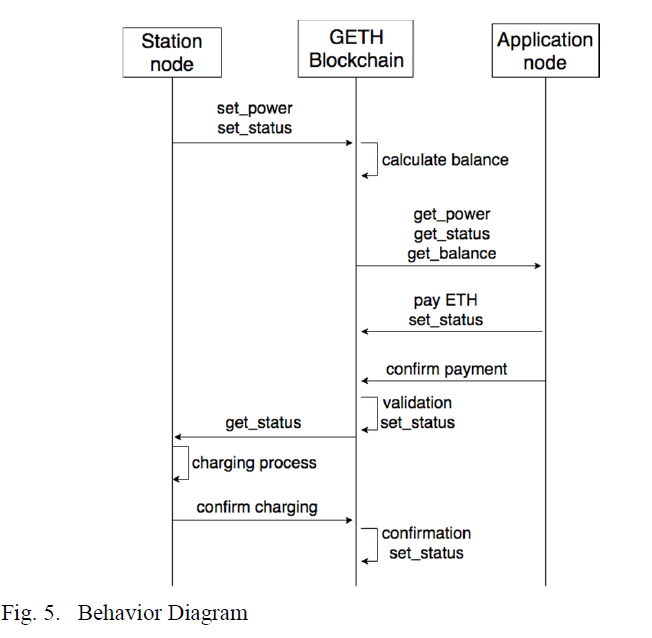
The system has 3 parts that interact with each other to establish smart contracts:

* Station Node
* Application Node
* GETH Blockchain

The sequence of behaviour among these 3 parts are as follows:

1. The station node sets a power and status variable in the blockchain. (Station Node -> GETH Blockchain)
2. The blockchain will calculate a price depending on the amount of power and price rate. (GETH Blockchain -> GETH Blockchain)
3. The application node gets the power, status and balance variables from the blockchain. (GETH Blockchain -> Application Node)
4. The application node then pays the ETH token and sets the status variable in the blockchain. (Application Node -> GETH Blockchain)
5. The application node then confirms the payment by calling the confirmPayment function in the blockchain. (Application Node -> GETH Blockchain)
6. The blockchain then validates the payment and sets a status variable (GETH Blockchain -> GETH Blockchain)
7. The station node gets the status variable from the blockchain. (GETH Blockchain -> Station Node)
8. The station node continues the charging process, based on the status variable received. (Station Node -> Station Node)
9. Once the charging process is complete, the station node confirms that the charging has finished by calling a confirmCharging function in the blockchain. (Station Node -> GETH Blockchain)
10. Finally, the blockchain confirms the charging completion and sets the status, before continuing the cycle.

The behaviour diagram is shown below:

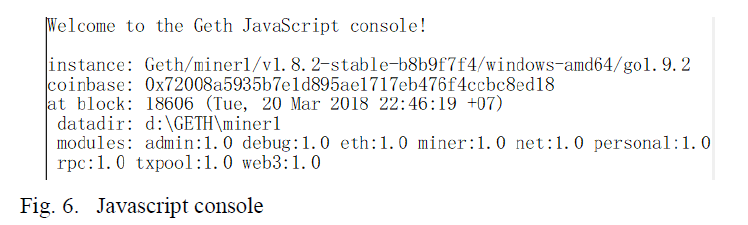
****

**Experiments**

**Experimental Procedure**

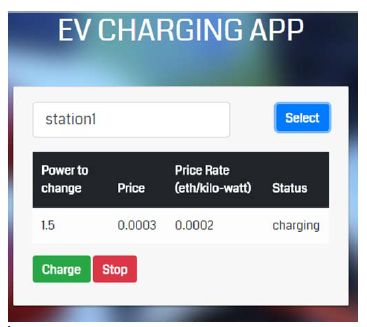
1. Set up Go-Ethereum blockchain on computer
2. Set up Go-Ethereum blockchain on Raspberry Pi
3. Connect all nodes via Wi-Fi

The GETH Javascript Console used as an interface to connect the nodes is shown below:

****

**Access Web Application**

The web application can order to charge, as well as display the current charging status, as shown below:



* Power to charge – Shows the remaining power that the charging station has to charge
* Price – Shows the remaining ETH token that the application node has to pay for charging
* Price Rate – Shows the price rate of the charging station
* Status – Displays the current charging status

**Limitations of the system**

The transaction time is quite long for each working state.

The storage space of the blockchain increases as the program continues running.