การประยุกต์ใช้งาน BLOCKCHAIN และ SMART CONTRACT สำหรับการชาร์จรถยนต์ไฟฟ้า

BLOCKCHAIN AND SMART CONTRACT APPLICATION FOR ELECTRIC VEHICLE CHARGING

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2560

การประยุกต์ใช้งาน BLOCKCHAIN และ SMART CONTRACT สำหรับการชาร์จรถยนต์ไฟฟ้า

BLOCKCHAIN AND SMART CONTRACT APPLICATION FOR ELECTRIC VEHICLE CHARGING

ฐิติวัฒน์ เรืองสาคร
ณัฐนันท์ จันทร์ทอง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2560

ปริญญานิพนธ์ปีการศึกษา 2560

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้งาน BLOCKCHAIN และ SMART CONTRACT สำหรับการชาร์จรถยนต์ไฟฟ้า

BLOCKCHAIN AND SMART CONTRACT APPLICATION FOR ELECTRIC VEHICLE CHARGING

ผู้จัดทำ

1. นายฐิติวัฒน์ เรืองสาคร รหัสนักศึกษา 57010358
2. นายณัฐนันท์ จันทร์ทอง รหัสนักศึกษา 57010430

_____ อาจารย์ที่ปรึกษา

(อาจารย์สรยุทธ กลมกล่อม)

การประยุกต์ใช้งาน BLOCKCHAIN และ SMART CONTRACT สำหรับการชาร์จรถยนต์ไฟฟ้า

นายฐิติวัฒน์ เรืองสาคร 57010358
นายณัฐนันท์ จันทร์ทอง 57010430
อาจารย์สรยุทธ กลมกล่อม อาจารย์ที่ปรึกษา
ปีการศึกษา 2560

บทคัดย่อ

ปริญญานิพนธ์นี้จัดทำขึ้นเพื่อศึกษา ออกแบบ และสร้างระบบจ่ายเงินแบบอิเล็กทรอนิกส์สำหรับรถยนต์ไฟฟ้าโดยใช้เทคโนโลยีของ Blockchain Smart Contract ในการควบคุมและจัดการการจ่ายเงินแทนระบบการจ่ายเงินแบบเก่า เช่น บัตรเครดิต เพื่อที่จะลดข้อจำกัดที่เกิดขึ้นกับระบบแบบศูนย์กลาง เช่น ค่าธรรมเนียมในการทำธุรกรรมสำหรับการทำธุรกรรมขนาดเล็กจำนวนมากๆ นอกจากนั้นระบบนี้จะช่วยลดปัญหาของจำนวนสถานีชาร์จรถยนต์ไฟฟ้าที่ไม่เพียงพอเวลาที่ต้องเดินทางในระยะทางไกล ดังนั้นเราจึงพัฒนาระบบระบบจ่ายเงินแบบอิเล็กทรอนิกส์ขึ้นมาแล้วนำไปติดตั้งไว้กลับสถานีชาร์จไฟส่วนบุคคล โดยการนำโหนดของ Blockchain ไปติดตั้งไว้กับสถานีชาร์จไฟ และ สร้าง Application ที่สามารถเข้าถึง Blockchain ได้ไว้สำหรับให้เจ้าของรถยนต์ไฟฟ้าเข้าใช้งานและชาร์จไฟกับสถานีชาร์จ ซึ่ง Application สามารถแสดงข้อมูลของการชาร์จไฟและสั่งจ่ายเงินและสั่งชาร์จไฟได้ ในส่วนของการจำลองรถยนต์ไฟฟ้านั้นเราจะใช้เป็นการจำลองส่งข้อมูลระหว่างสถานีชาร์จและรถยนต์ไฟฟ้า

BLOCKCHAIN AND SMART CONTRACT

APPLICATION FOR ELECTRIC VEHICLE CHARGING

Mr. Thitiwat Ruangsakorn 57010358

Mr. Natthanan Chanthong 57010430

Mr. Sorayut Glomglome Advisor

Academic Year 2017

ABSTRACT

This thesis is made up for study, design and build the electronic payment system for electric vehicle using Blockchain Smart Contract technology to control and manage payment instead of the present payment system such as credit card for the purpose to reducing central system limitation such as, transaction fee in a large amount of micro transactions. In addition, this system will reduce the problem of number of charging stations that is inadequate for electric vehicle when travel in the long distance, so we build the electronic payment system in the personal charging station by setting up the Blockchain node in the charging station and build the application that can access to Blockchain for electric vehicle's owner to access the charging station. The application can show the charging information and order for payment and charging process. In the electric vehicle part, we will simulate the data transmission between charging station and electric vehicle.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีด้วยความช่วยเหลือจากหลายฝ่ายทั้งในทางตรงและทางอ้อม ปริญญานิพนธ์ฉบับนี้จะสำเร็จลงไม่ได้หากปราศจากความช่วยเหลือของบุคคลเหล่านี้

ขอขอบคุณ อาจารย์ที่ปรึกษา คือ อาจารย์สรยุทธ กลมกล่อม เป็นผู้ให้คำแนะนำ คำปรึกษา และให้ความช่วยเหลือตลอดการทำโครงงาน ซึ่งทำให้การทำงานต่าง ๆ เป็นไปได้อย่างราบรื่นและสำเร็จลุล่วงไปได้ด้วยดี

ขอขอบคุณอาจารย์และบุคลากรต่าง ๆ ในสาขาวิชาวิศวกรรมคอมพิวเตอร์ที่ได้ให้คำแนะนำและสั่งสอนความรู้ต่าง ๆ มาโดยตลอด รวมถึงห้องแล็ป HCRL (The Hybrid Computing Research Laboratory) ที่ได้เอื้อเฟื้อสถานที่ในการทำวิจัยและพัฒนาโครงงาน

ขอขอบคุณรุ่นพี่และเพื่อนหลาย ๆ คนในภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้ให้คำแนะนำ คำปรึกษาและแบ่งปันความรู้ในทุก ๆ ด้าน

ในสุดท้ายนี้ ขอขอบคุณ บิดา มารดา และครอบครัว ที่ได้เลี้ยงดู สั่งสอน และให้การสนับสนุน พร้อมทั้งให้โอกาสในการศึกษาและให้กำลังใจเสมอมา

ฐิติวัฒน์ เรืองสาคร
ณัฐนันท์ จันทร์ทอง

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตาราง หน้า

สารบัญรูปภาพ

รูป หน้า

สารบัญรูปภาพ (ต่อ)

รูป หน้า

สารบัญรูปภาพ (ต่อ)

รูป หน้า

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญ

ปัจจุบันจำนวนการใช้งานรถยนต์ไฟฟ้าในโลกนี้ยังมีจำนวนน้อยเมื่อเทียบกับการใช้งานรถยนต์ทั้งหมด เนื่องจากจำนวนสถานีชาร์จรถยนต์ไฟฟ้าสาธารณะยังมีไม่มาก และ ไม่ครอบคลุมในบางพื้นที่โดยเฉพาะเวลาที่ต้องเดินทางในระยะทางไกลๆ ทำให้ผู้ใช้งานรถยนต์ไฟฟ้าเกิดความวิตกกังวลในการเดินทางว่าแบตเตอรี่จะเพียงพอในการเดินทางหรือไม่

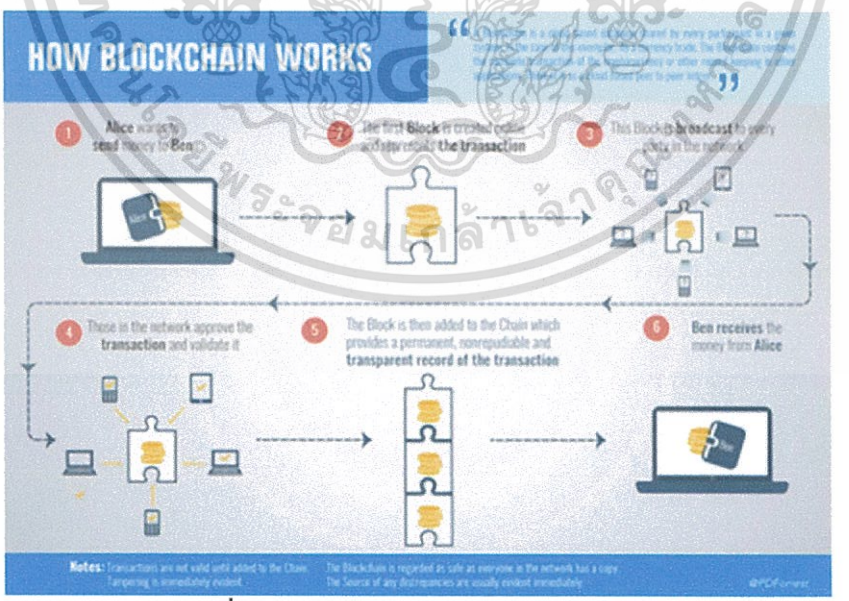
รูปที่ 1.1 Share&Charge[1]

ดังนั้นถ้ามีสถานีชาร์จส่วนบุคคลตามบ้านหรือร้านอาหารที่อยู่ระหว่างทางที่สามารถเปิดให้ผู้ที่ใช้รถยนต์ไฟฟ้าเข้าไปใช้งานได้ก็จะช่วยลดเรื่องของสถานีชาร์จที่มีไม่ครอบคลุมและลดความวิตกกังวลของผู้ใช้งานรถยนต์ไฟฟ้าลงได้และอาจทำให้คนหันมาใช้งานรถยนต์ไฟฟ้ากันมากขึ้น ซึ่งจากตัวอย่างของบริษัท Share&Charge ในรูปที่ 1.1 ได้นำวิธีการดังที่กล่าวมาข้างต้นมาประยุกต์ใช้กับเทคโนโลยี Blockchain และสร้าง Platform ในการจ่ายเงินสำหรับการชาร์จรถยนต์ไฟฟ้า โดยมี Application ไว้ให้ผู้ใช้รถยนต์ไฟฟ้าค้นหาสถานีชาร์จเพื่อเข้าใช้งานและจ่ายเงิน

[1] https://shareandcharge.com/

รูปที่ 1.2 สถานีบริการ EA Anywhere สาขา ครัวบ้านขวัญ อ.เขาย้อย[2]

ในรูปที่ 1.2 เป็นตัวอย่างของสถานีชาร์จไฟฟ้าในประเทศไทยโดยบริษัท EA Anywhereโดยติดตั้งไว้ที่ร้านอาหารครัวบ้านขวัญ อ.เขาย้อย บนเส้นทางหลักวิ่งตรงสู่หัวหิน ซึ่งทำให้ผู้ที่ใช้รถยนต์ไฟฟ้าที่เดินทางผ่านเส้นทางนี้สามารถเข้ามาใช้งานในระหว่างเดินทางได้ นอกจากนี้ทางบริษัท EA Anywhere ยังวางแผนที่จะเพิ่มสถานีชาร์จไฟอีกจำนวนมาก

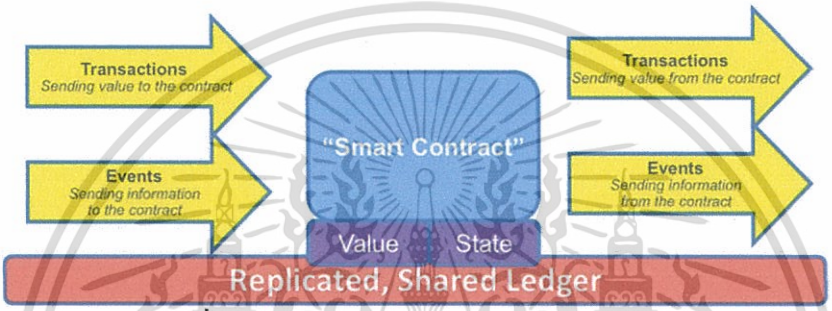


รูปที่ 1.3 การทำงานของ Blockchain[3]

[2] http://www.eaanywhere.com/

[3] https://goo.gl/dUytW5

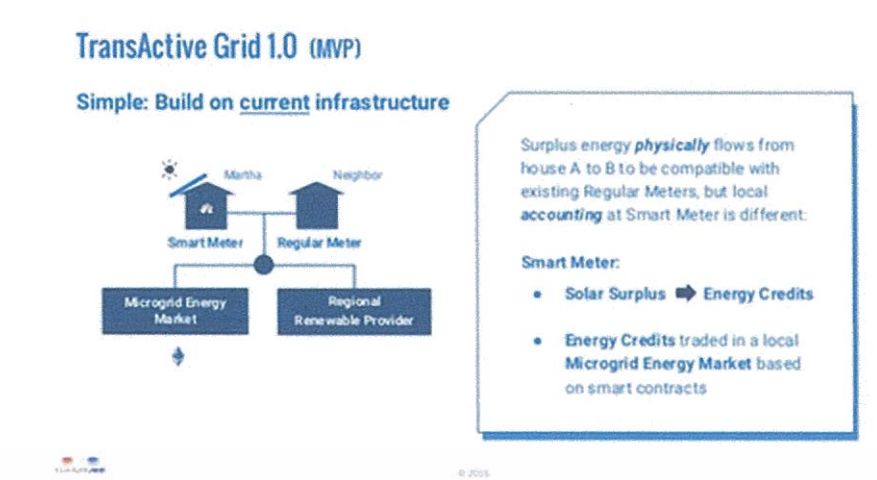
เทคโนโลยี Blockchain ในปัจจุบันเริ่มได้รับความสนใจมากขึ้นในภาคอุตสาหกรรมต่างๆ เช่น การเงิน, การรักษาพยาบาล, อสังหาริมทรัพย์ หรือแม้แต่การนำไปประยุกต์ใช้กับ Internet of Things และจากรูปที่ 1.3 เป็นการทำงานของ Blockchain ซึ่งเป็นระบบที่มีคุณสมบัติเป็น Decentralize fashion คือ การบริหารจัดการที่ไม่ต้องอาศัยตัวกลาง หรือ 3rd Party ทำให้สามารถลดค่าใช้จ่าย เช่น ค่าธรรมเนียม และเพิ่มความรวดเร็วในการทำ Transaction นอกจากนี้ยังมีความปลอดภัยสูง เนื่องจากข้อมูลจะถูกเก็บไว้ใน Blockchain ซึ่งกระจายอยู่ตาม Node ต่างๆใน Network และแต่ละ Data block จะอ้างอิง Data block ก่อนหน้าทำให้สามารถตรวจสอบข้อมูลย้อนกลับเพื่อป้องกันการปลอมแปลงได้



รูปที่ 1.4 การทำงานของ Smart Contract[4]

ใน Blockchain platform ใหม่ๆ เช่น Ethereum จะใช้ Smart Contract ซึ่งเป็น Application ที่ทำงานอยู่บน Blockchain คอยจัดการ Transaction เช่น การ Authentication และ Validation โดยจากรูปที่ 1.4 เป็นการทำงานของ Blockchain ด้วยการรับ Input จาก Node ใน Network จากนั้นจะทำการยืนยัน Transaction ที่ Input เข้ามาแล้วส่ง Output ออกมาบันทึกลง Blockchain

[3] https://goo.gl/dUytW5



รูปที่ 1.5 Transactive Grid[5]

จากรูปที่ 1.4 เป็นโปรเจค TransActive Grid ที่ได้นำ Smart Contract เข้ามาใช้งาน เช่น TransActive Grid ของบริษัท LO3 Energy ซึ่งได้นำ Ethereum Blockchain มาประยุกต์ใช้กับระบบไฟฟ้า Microgrid สำหรับการซื้อขายไฟกันเองภายในเมือง Brooklyn โดยติดตั้ง Smart meter ที่ติดตั้ง Blockchain ไว้ให้กับบ้านที่เข้าร่วมโครงการซึ่งจะมีทั้งบ้านธรรมดาทั่วไป และบ้านที่มี Solar cell โดยบ้านแต่ละหลังสามารถซื้อไฟฟ้าจากบ้านที่มี Solar cell ได้ตามต้องการแบบ Peer-to-Peer โดยไม่ต้องผ่านไฟฟ้า Grid หลัก และบ้านที่ขายไฟฟ้าสามารถกำหนดราคาขึ้นมาเองตามต้องการ โดยระบบการซื้อ-ขายทั้งหมดจะถูกจัดการโดย Smart Contract ของ Ethereum Blockchain

ในโครงงานนี้เราจะประยุกต์ใช้วิธีการของ Share&Charge โดยเราจะสร้างเป็น Community ของสถานีชาร์จไฟและนำ Blockchain มาควบคุมทั้งกระบวนการชาร์จไฟ และการจ่ายเงินทั้งหมดให้เป็นแบบอัตโนมัติ โดยสถานีชาร์จไฟที่เข้าร่วมจะถูกติดตั้ง Blockchain ไว้กับสถานีชาร์จเพื่อจัดการกระบวนการชาร์จไฟ และนอกจากนี้เราจะสร้าง Application สำหรับให้ผู้ใช้รถยนต์ไฟฟ้าเข้ามาใช้งานสถานีชาร์จใน Community ของเรา

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อศึกษาการทำงานของ Blockchain และ Smart Contract และนำไปประยุกต์ใช้กับ Internet of Things ประเภทธุรกิจหลัก
2) เพื่อศึกษากระบวนการในการชาร์จไฟของระบบ EV Charging station
3) เพื่อสร้างระบบควบคุมการซื้อขายไฟของรถยนต์ไฟฟ้า

[5] https://goo.gl/D46XL4

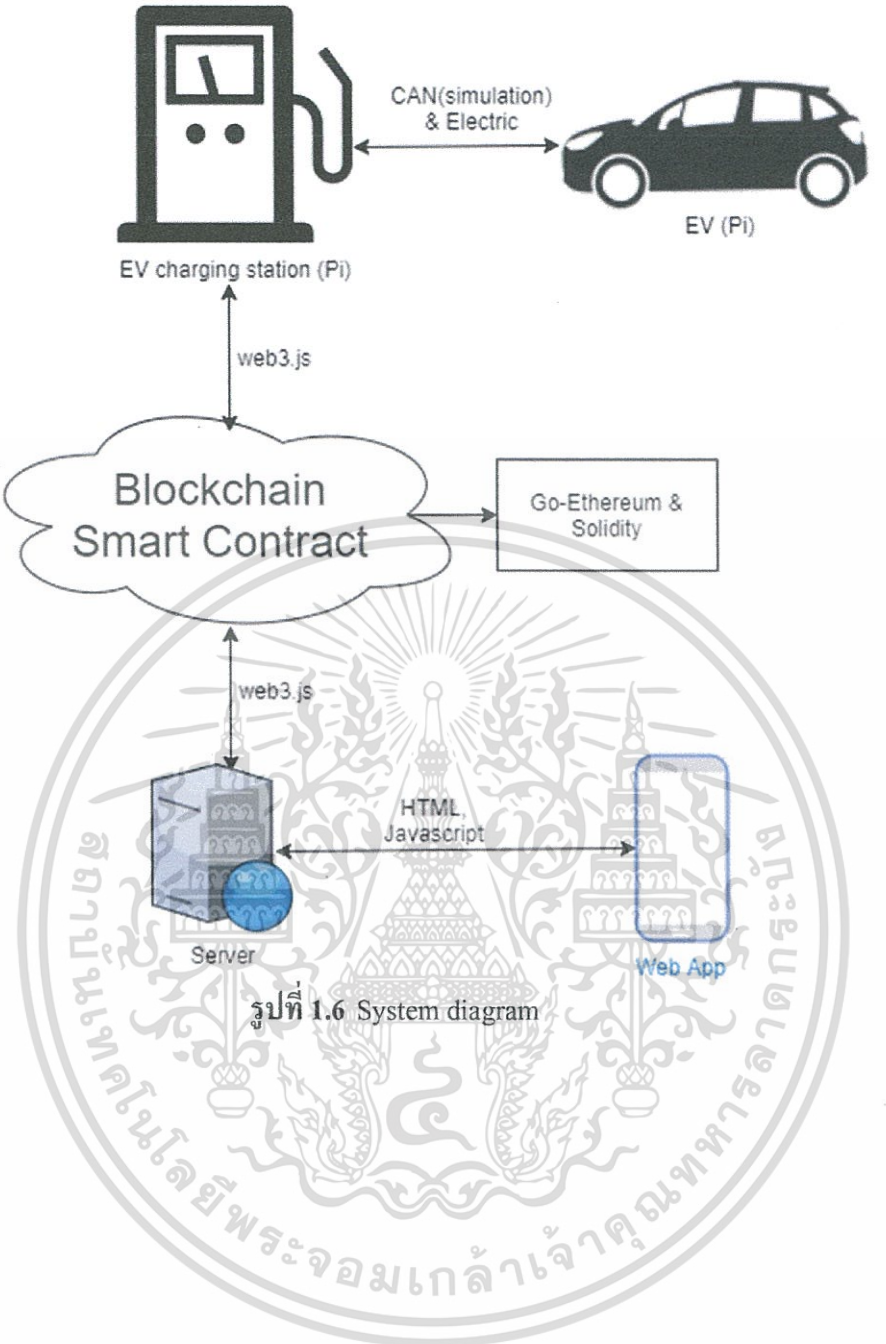
1.3 ประโยชน์ที่ได้รับ

1) เข้าใจการทำงานของ Smart Contract ใน Ethereum Blockchain
2) เข้าใจการประยุกต์ใช้เทคโนโลยี Blockchain ร่วมกับ Internet of Things
3) ระบบซื้อขายไฟรถยนต์แบบอัตโนมัติในรูปแบบ Peer-to-Peer
4) เพื่อลดความกังวลของผู้ขับขี่รถยนต์ไฟฟ้าในการเดินทางไกล

1.4 ขอบเขตของโครงงาน

สร้างระบบควบคุมการชาร์จไฟและจ่ายเงินแบบอิเล็กทรอนิกส์ของรถยนต์ไฟฟ้า โดยเราจะสร้าง Node ของ Blockchain ขึ้นมาดัง System diagram ในรูปที่ 1.6 ซึ่งเราจะแบ่ง Node ออกเป็น 2 ประเภทตามฟังชั่นการทำงานดังนี้

1) Station node เป็น Node ที่ติดตั้งไว้ใน Raspberry Pi แล้วนำไปติดตั้งไว้กับสถานีชาร์จไฟ เพื่อควบคุมกระบวนการชาร์จไฟและการจ่ายเงินผ่าน Blockchain Smart Contract โดย Station node จะอ่านค่าแบตเตอรี่จาก Raspberry Pi อีกบอร์ดหนึ่งที่จำลองการติดต่อสื่อสารกันระหว่างสถานีชาร์จและรถยนต์ไฟฟ้า เมื่อ Station node อ่านค่าแล้วก็จะ Update ข้อมูลไปที่ Blockchain Smart Contract และรอคำสั่งการชาร์จไฟจาก Blockchain Smart Contract
2) Application node เป็น Node ที่ติดตั้งไว้บน Server หรือ Computer ซึ่งจะทำหน้าที่นำข้อมูลการชาร์จไฟทั้งหมดใน Blockchain Smart Contract มาแสดงบน Web application และควบคุมการจ่ายเงิน รวมไปถึงยืนยันความถูกต้องของการจ่ายเงินและการชาร์จไฟด้วยการใช้ฟังก์ชั่นใน Blockchain Smart Contract นอกจากนี้ Application node ยังทำหน้าที่เป็น Mining node ในการยืนยัน Transaction และนำไปบันทึกลง Blockchain



รูปที่ 1.6 System diagram

1.5 แผนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงาน

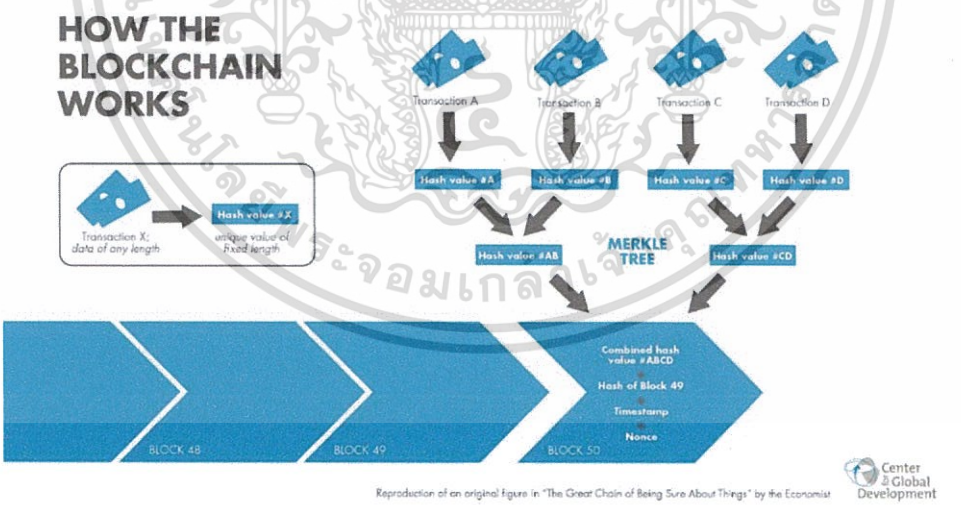
หัวข้อกิจกรรม	เดือน									
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.
1. ค้นหาหัวข้อโครงงานที่น่าสนใจ										
2. เสนอหัวข้อโครงงาน										
3. ออกแบบ Hardware สำหรับการจำลองระบบสถานีชาร์จกับรถยนต์ไฟฟ้า										
4. ติดตั้ง Blockchain ลงบอร์ด Raspberry PI										
5. ทำ Application										
6. นำชิ้นงานแต่ละส่วนมาบูรณาการเป็นระบบ										
7. นำระบบที่ได้มาทดสอบและแก้ไขข้อผิดพลาด										
8. จัดทำเอกสารและนำเสนอโครงงาน										

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในบทนี้กล่าวถึงทฤษฎีต่างๆที่เกี่ยวข้อง ซึ่งถูกนำมาใช้ในการทำโครงงานการประยุกต์ใช้ Blockchain และ Smart Contract สำหรับการชาร์จรถยนต์ไฟฟ้าคือ Blockchain, Electronic vehicle charging และ Ethereum

2.1 Blockchain

Blockchain เป็นรูปแบบการจัดเก็บข้อมูลชนิดหนึ่งที่มีลักษณะเป็นบล็อกเรียงต่อกันเป็นสาย ซึ่งแต่ละบล็อกจะมีการอ้างอิงกับบล็อกก่อนหน้าโดยอ้างอิงจาก Hash number ประจำบล็อก ซึ่งตัวเลข Hash number จะขึ้นอยู่กับข้อมูลที่อยู่ในบล็อกนั้นๆ ข้อมูลที่เก็บอยู่ในบล็อกก็จะประกอบไปด้วยชุดของ Transaction ซึ่งแต่ละ Transaction ก็จะมีเลข Hash number เป็นของตัวเองเพื่อใช้อ้างอิงถึงกัน ภายใน Transaction ก็จะมีข้อมูลการทำ Transaction ต่างๆของแต่ละบัญชีดังรูปที่ 2.1 ซึ่งแต่ละ Node ใน Blockchain network นั้นก็จะเก็บข้อมูลชุดเดียวกันเอาไว้

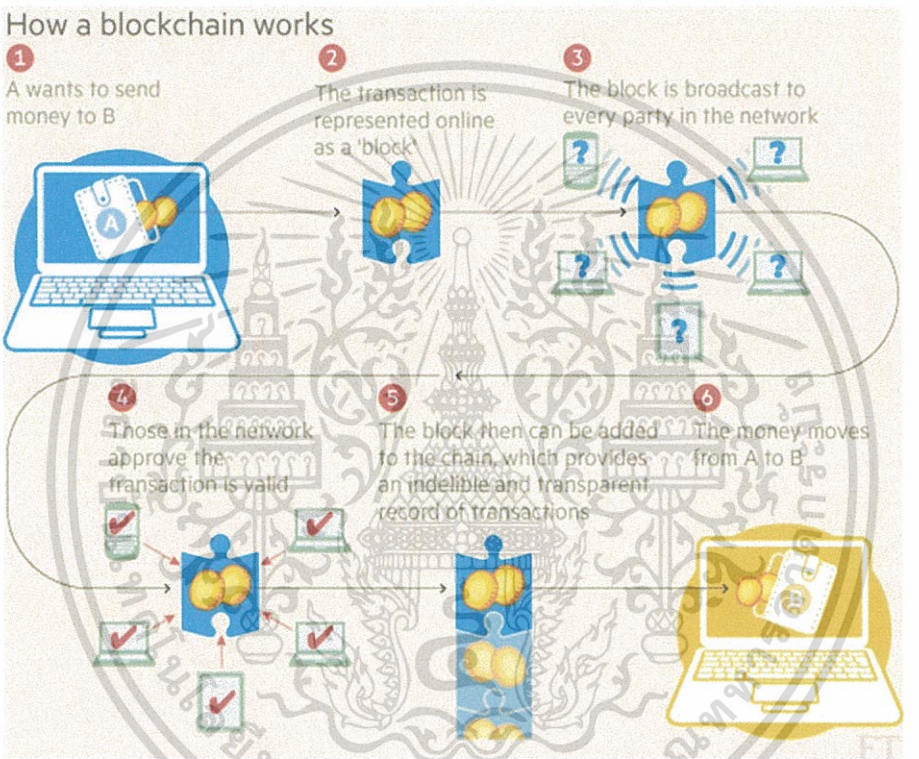


รูปที่ 2.1 How Blockchain works[6]

กระบวนการในการทำ Transaction ใน Blockchain ดังรูปที่ 2.2 จะยกตัวอย่างการโอนเงินจากบัญชี A ไปบัญชี B

[6] https://www.techbriny.com/blockchain-technology-kya-hai/

1) A ต้องการจะโอนเงินไปให้ B เริ่มแรกการจะใช้งานบัญชี A ได้จะต้องมี Private key ของบัญชี A ซึ่งเป็น Public key ก่อนเพื่อยืนยันความเป็นเจ้าของบัญชี
2) สร้าง Transaction ขึ้นโดยระบุว่า บัญชี A โอนเงินให้บัญชี B จำนวนเท่าใด
3) Transaction ดังกล่าวถูกกระจายไปให้ทุก Node ใน Blockchain network
4) แต่ละ Node ใน Network จะทำการยืนยันความถูกต้องของ Transaction นั้น
5) เมื่อ Transaction ถูกยืนยันก็จะถูกบันทึกลง Blockchain ของทุก Node
6) การโอนเงินจากบัญชี A ไปบัญชี B เสร็จสิ้น



รูปที่ 2.2 Transaction in Blockchain[7]

2.2 Ethereum

Ethereum เป็น Platform ของ Blockchain รูปแบบหนึ่ง โดยจะใช้ Token ที่เรียกว่า "Ether" ในการทำ Transaction ต่างๆ เช่น การโอนเงิน, การซื้อ-ขายของต่างๆ เป็นต้น ซึ่ง Ethereum จะมี Smart Contract ซึ่งเป็น Application รูปแบบหนึ่งที่ทำงานอยู่บน Blockchain ทำหน้าที่ในการจัดการ Transaction ต่างๆ

[7] https://www.notebookcheck.net/Microsoft-is-making-it-easier-for-enterprise-to-integrate-block-chain-technology.241094.0.html

รูปที่ 2.3 Ethereum Blockchain Platform[8]

2.2.1 Go-Ethereum (geth)

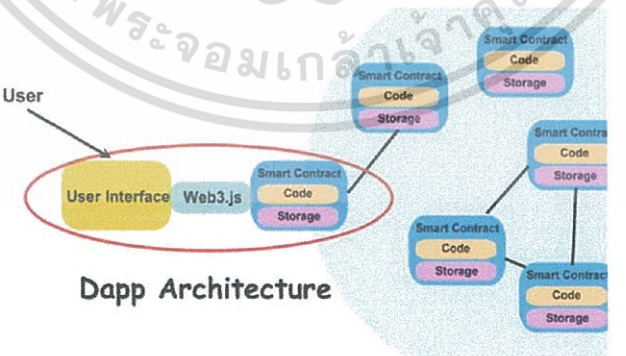
geth คือ Command Line Interface สำหรับรัน Ethereum node ซึ่งจะ Implement ในภาษา Go ซึ่งคุณสมบัติของ geth คือ สามารถ Mine เหรียญ Ether ได้, โอนเงินระหว่าง Account Address, สร้าง Contract และทำ Transactions, ตรวจสอบประวัติของแต่ละ Block ใน Blockchain เป็นต้น

ในส่วนของ Interface สามารถเลือกใช้ได้ 3 รูปแบบดังนี้

1) Javascript Console โดยใช้ Javascript API ของ web3.js ที่ทำงานบน Console
2) JSON-RPC server เป็นการส่งข้อมูลในรูปแบบของ JSON ซึ่งมีขนาดเล็ก
3) Command Line Option เป็นคำสั่งของ geth ที่ใช้กับ Command Line

2.2.2 Web3.js

web3.js เป็น Javascript API ในการ Implement go-ethereum Blockchain ดังรูปที่ 2.4 web3.js จะทำหน้าที่เป็น API ในการเชื่อมระหว่าง User interface กับ Smart Contract ของ go-ethereum Blockchain



รูปที่ 2.4 web3.js[9]

[8] https://goo.gl/RsuuKF

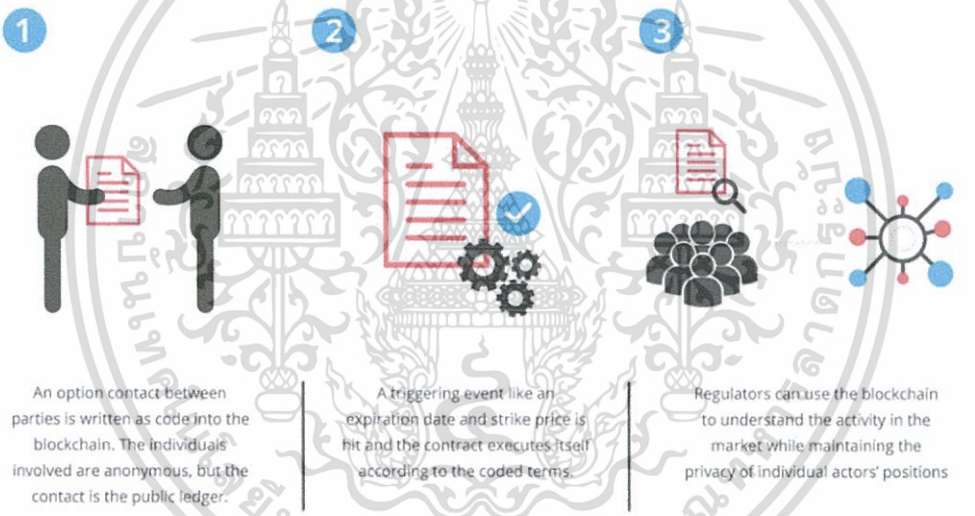
[9] https://medium.com/hci-wvu/how-to-build-your-first-%C3%B0app-fe0c89d8f95f

2.2.3 Smart Contract

Smart Contract คือ “สัญญาอัจฉริยะ” โดย Nick Szabo ได้กล่าวว่า เป็นการทำข้อตกลงในสัญญา โดยไม่อาศัยคนกลาง โดยอาศัยโปรแกรมในการจัดการซึ่งไม่สามารถปลอมแปลงได้ เพราะมีการบรรลุข้อตกลงเป็นเอกฉันท์แล้วภายใน Blockchain

หลักการทำงานของ Smart Contract ดังรูปที่ 2.5 ดังนี้

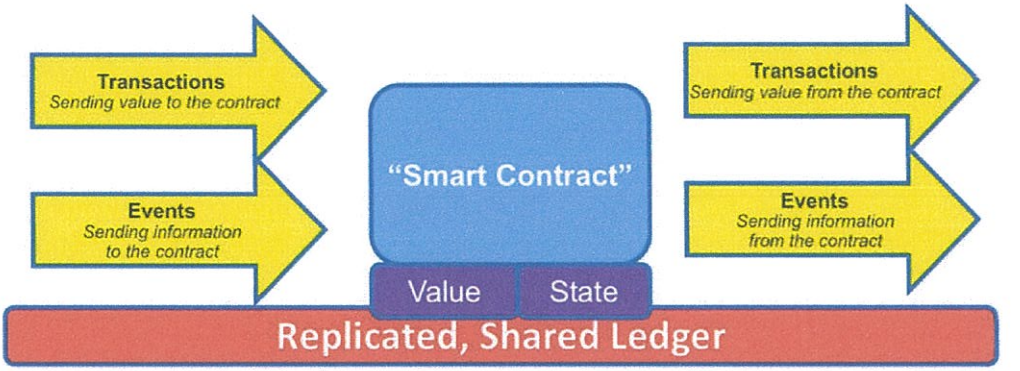
1) มีกลุ่มคนที่ทำสัญญาร่วมกันในรูปแบบของการเขียนโค้ดโดยไม่มีการระบุตัวตน แต่ตัวสัญญาหรือ Contract จะเป็นสาธารณะและถูกบันทึกลง Blockchain
2) เมื่อ Contract ดังกล่าวถูกกระตุ้นให้ทำงานโดย Event ใดๆ Contract นั้นก็จะทำงานตาม Code ที่เขียน
3) Contract ดังกล่าวจะสามารถถูกเรียกใช้งานโดยใครก็ตามผ่าน Blockchain โดยไม่มีการระบุตัวตนของผู้เรียกใช้งาน



รูปที่ 2.5 การทำงานของ Smart Contract[10]

โดยหลักๆแล้วการทำงานของ Smart Contract คือการเป็นตัวโปรแกรมที่ทำงานด้วยตัวเองโดยการรับสินทรัพย์หรือจำนวนเงินเข้ามาแล้วประมวณผลว่าควรจะทำอย่างไรกับสินทรัพย์หรือจำนวนเงินนั้นแล้วส่งเป็น Output ออกมา ซึ่งตัวโปรแกรมนี้จะถูกฝากไว้บน Blockchain และจะถูกเรียกใช้งานโดย Node ต่างๆจาก Address ของแต่ละ Contract ดังรูปที่ 2.6

[10] https://goo.gl/hBXg31



รูปที่ 2.6 Smart Contract Diagram[11]

2.2.4 Solidity

Ethereum นั้นมีภาษาที่ใช้สำหรับเขียน Smart Contract เรียกว่า Solidity ซึ่งมี Syntax พื้นฐานมาจากภาษา Java Scrip ดังรูปที่ 2.7 ซึ่งเป็นตัวอย่างโปรแกรมของ Smart Contract

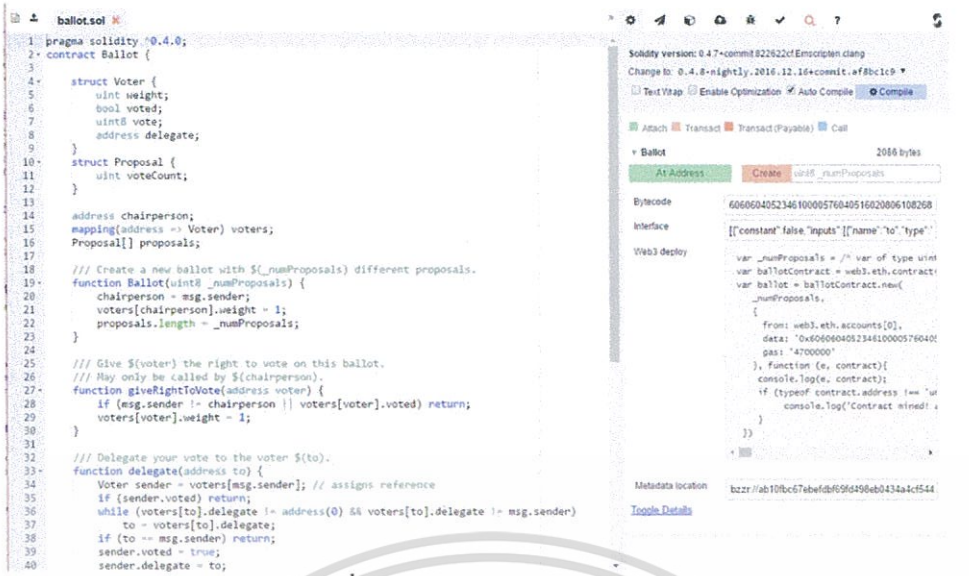
```
1  pragma solidity ^0.4.0;
2
3  contract MyFirstContract {
4      event print_string(string str);
5
6      function HelloWorld(){
7          print_string("Hello world");
8      }
9  }
```

รูปที่ 2.7 ภาษา Solidity[12]

ตัว IDE ที่จะใช้สำหรับเขียน Smart Contract ในที่นี้จะใช้ Solidity Browser ซึ่งสามารถ Compile Code และระบุ Address ของแต่ละ Contract ที่สร้างขึ้นมาดังรูปที่ 2.8

[11] https://goo.gl/V8Pm2u

[12] https://goo.gl/SqUTY4



รูปที่ 2.8 Solidity Browser

2.3 Decentralized and Autonomous System

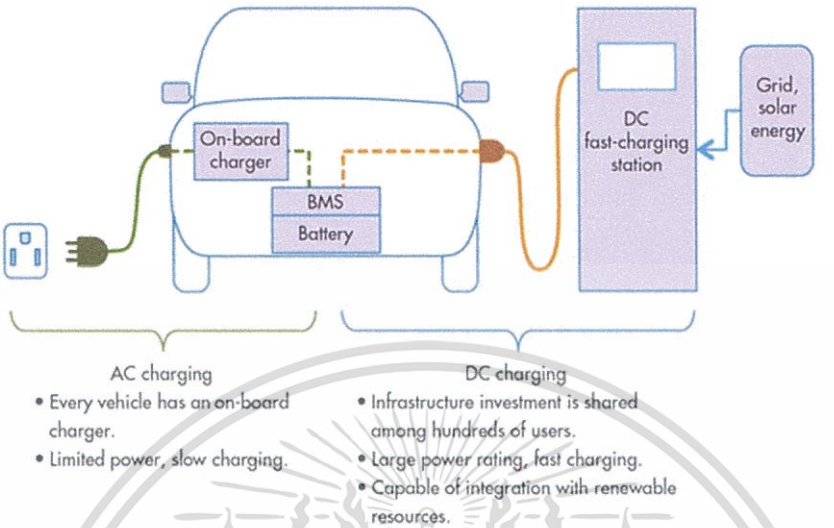
ในระบบการซื้อขายแบบทั่วไปที่เป็นแบบ Centralized นั้น จะต้องมีคนกลางที่เชื่อถือได้ในการเป็นตัวกลางในการแลกเปลี่ยน เนื่องจากความไม่ไว้วางใจกันระหว่างผู้ซื้อ-ขาย ตัวอย่างเช่น ธนาคาร ซึ่งผู้ที่ทำการซื้อขายกันนั้นจำเป็นจะต้องเสียค่าธรรมเนียมในการทำธุรกรรม ดังนั้นถ้านำระบบแบบ Decentralized ที่สามารถสร้างความไว้วางใจระหว่างผู้ซื้อ-ขายได้ ก็จะช่วยลดการเสียค่าธรรมเนียมและต้นทุนของระบบแบบ Centralized ได้การใช้เทคโนโลยี Blockchain และ Smart Contract

ในระบบการทำงานที่เรียกว่า Autonomous นั้นคือ ระบบที่สามารถดำเนินการทุกอย่างได้ด้วยตัวเองโดยไม่ต้องอาศัยการสั่งการ ตัวอย่างเช่น Internet of Things ที่สามารถจัดการทุกอย่างได้เองภายในระบบด้วยการเก็บข้อมูลจาก Sensor แล้วนำข้อมูลที่ได้ไปประมวนผลที่ Cloud Platform แล้วแสดงข้อมูลที่ประมวนผลได้ออกมา หรือใช้ข้อมูลที่ได้ไปสั่งการ Device ให้ทำงาน ซึ่งเมื่อนำแนวคิดของ Internet of Things มารวมกับแนวคิดของ Blockchain ก็จะได้ก็จะได้ระบบที่เป็น Decentralized ซึ่งจะลดข้อจำกัดของ Model แบบ Client-Server ทำให้ได้ระบบที่มีความปลอดภัยมากยิ่งขึ้น ในขณะเดียวกัน Smart Contract ที่อยู่ภายใน Blockchain เองก็มีคุณสมบัติที่เป็น Autonomous เช่นกันเนื่องจากเป็นโปรแกรมที่ทำงานได้ด้วยตัวเอง

2.4 Electronic Vehicle Charging

กระบวนการอัดประจุสำหรับรถยนต์ไฟฟ้านั้นมีทั้งการอัดประจุด้วยไฟฟ้ากระแสสลับและไฟฟ้ากระแสตรง โดยส่วนใหญ่จะใช้การอัดประจุแบบไฟฟ้ากระแสตรงหรือแบบ Fast Charging มี

ระยะเวลาในการอัดประจุประมาณ 20 - 30 นาทีโดยมีความสามารถชาร์จได้สูงสุด 80% จากจำนวนเต็มของแบตเตอรี่

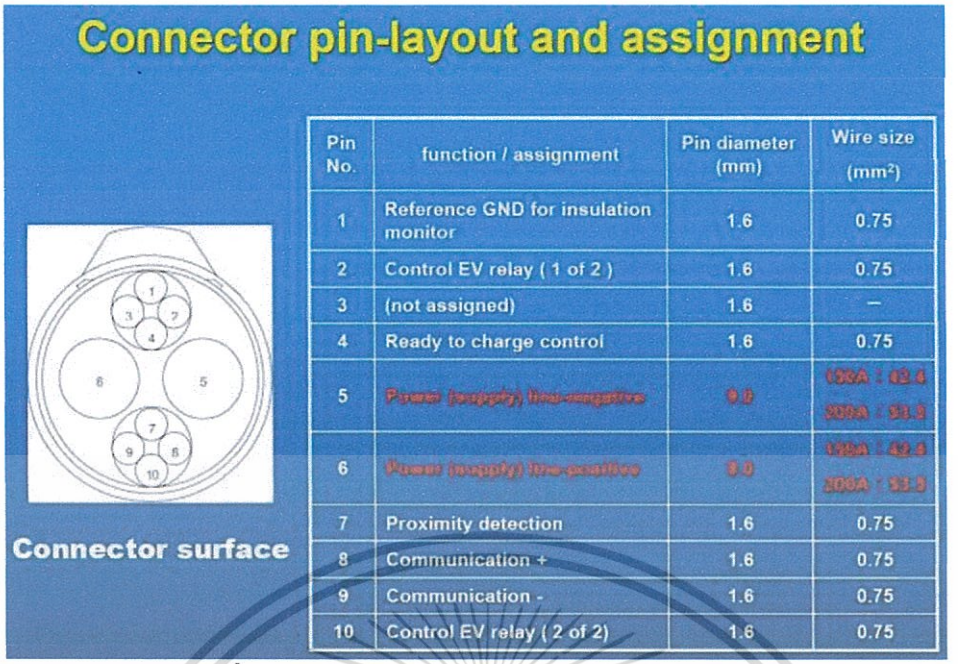


รูปที่ 2.9 Electronic Vehicle Charging[13]

พอร์ตมาตรฐานโปรโตคอลที่ใช้คือ CHAdeMO สำหรับไฟฟ้าแรงดันสูง Fast charging ซึ่งออกแบบโดย TEPCO (Tokyo Electric Power Company) ซึ่งใช้ CAN โดย CAN คือโปรโตคอลสำหรับการส่งข้อมูลติดต่อระหว่าง Components ภายในรถยนต์ไฟฟ้าอย่างเช่นการจัดการกับแบตเตอรี่ของรถยนต์โดยจากรูปที่ 2.10 Pin ที่ใช้ในการติดต่อกับ CAN Bus คือ Pin ที่ 8 กับ 9 สำหรับการชำระค่าบริการปกติจะใช้ระบบการคิดค่าบริการโดยผ่านบัตร RFID ซึ่งเหมือนกับบัตรเติมเงินทั่วไป ซึ่งจะอนุญาตให้ชาร์จไฟฟ้าได้เฉพาะผู้ที่ถือบัตร RFID ได้เท่านั้น

ส่วนในระบบของโครงงานนี้จะเน้นการทำ Transaction เป็นหลักโดยจะใช้การเขียนโปรแกรมจำลองการสื่อสารระหว่างสถานีชาร์จกับแบตเตอรี่ของรถยนต์ไฟฟ้าแทน และใช้เทคโนโลยี Blockchain กับ Smart Contract ในการคิดค่าบริการในการชาร์จประจุไฟฟ้าสำหรับรถยนต์ไฟฟ้าเพื่อบอกกับฝั่ง Application ว่าต้องทำการจ่ายเงินเท่าไหร่เพื่อทำการจ่ายเงินแบบอัตโนมัติ

[13] https://goo.gl/s2FQEE



Pin No.	function / assignment	Pin diameter (mm)	Wire size (mm^2)
1	Reference GND for insulation monitor	1.6	0.75
2	Control EV relay (1 of 2)	1.6	0.75
3	(not assigned)	1.6	–
4	Ready to charge control	1.6	0.75
5	Power (supply) line-negative	8.0	125A : 42.4 200A : 33.6
6	Power (supply) line-positive	8.0	125A : 42.4 200A : 33.6
7	Proximity detection	1.6	0.75
8	Communication +	1.6	0.75
9	Communication -	1.6	0.75
10	Control EV relay (2 of 2)	1.6	0.75

รูปที่ 2.10 พอร์ตชาร์จมาตรฐาน CHAdeMO[14]

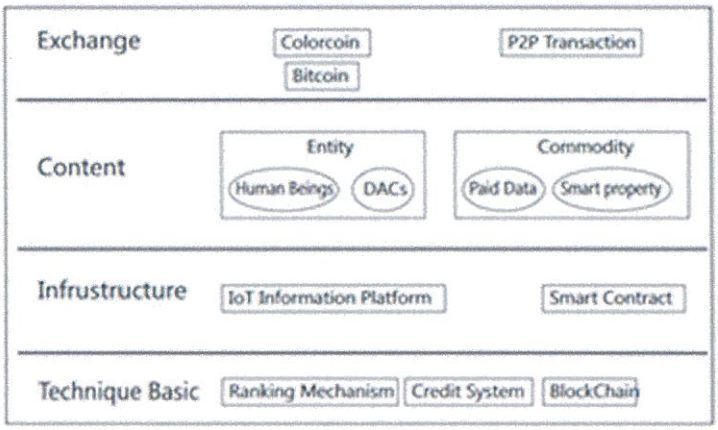
2.5 งานวิจัยที่เกี่ยวข้อง

2.5.1 An IoT Electric Business Model Based on the Protocol of Bitcoin

งานวิจัยนี้กล่าวถึงการการประยุกต์ใช้งาน Internet of Things กับ Electric Business Model ด้วยการใช้ Protocol ของ Bitcoin ซึ่งใช้ระบบของ Blockchain

เนื่องจากการรวม E-business Model ซึ่งเป็นระบบที่เป็นนามธรรม กับ IoT ซึ่งเป็นระบบที่เป็นรูปธรรมอาจทำให้เกิดปัญหาหลายๆอย่าง จึงต้องอาศัย Bitcoin เข้ามาช่วย เช่น การสร้างระบบที่ไม่ต้องอาศัยตัวกลาง และการแปลงข้อมูลของ IoT ให้อยู่ในรูปของ Token ที่มีพื้นฐานมาจาก Bitcoin

[14] https://goo.gl/M19nBv



รูปที่ 2.11 IoT E-business Model

จากรูปที่ 2.11 ระบบจะแบ่งการทำงานเป็น 4 Layer ดังนี้

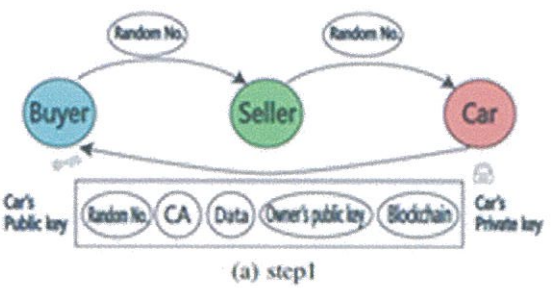
Exchange เป็นส่วนของการแลกเปลี่ยน โดยจะประยุต์ใช้ Bitcoin หรือ Colorcoin ในการแลกเปลี่ยนข้อมูลกันแบบ P2P

Content คือ เนื้อหาในการแลกเปลี่ยน แบ่งเป็น 2 ส่วนคือ

1) Entity จะมีส่วนของ Human Beings คือคำสั่งซื้อจากลูกค้า และ DACs (Decentralized Autonomous Corporations) เป็นรูปแบบของการแลกเปลี่ยนที่ไม่ต้องมีคนกลางคอยจัดการ คือสามารถทำงานได้ด้วยตัวเอง หรือเป็นระบบแบบ Autonomous
2) Commodity จะมีส่วนของ Paid Data คือ ข้อมูลที่ได้จากอุปกรณ์ IoT เช่น ข้อมูลต่างๆจาก Sensor และ Smart property คือ สินทรัพย์อัจฉริยะ เช่น รถยนต์อัจฉริยะที่ต้องใช้ Key เฉพาะในการสตาร์ทรถ

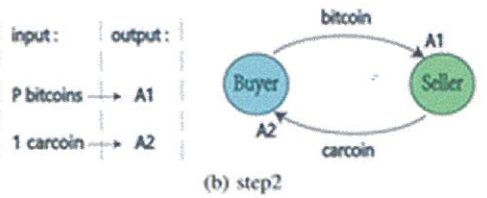
Infrastructure คือโครงสร้างของระบบ เช่น IoT Platform และ Smart Contract

Technique Basic คือ เทคนิคทั่วไปที่นำมาใช้ในระบบ เช่น Ranking Mechanism, Credit System และ Blockchain



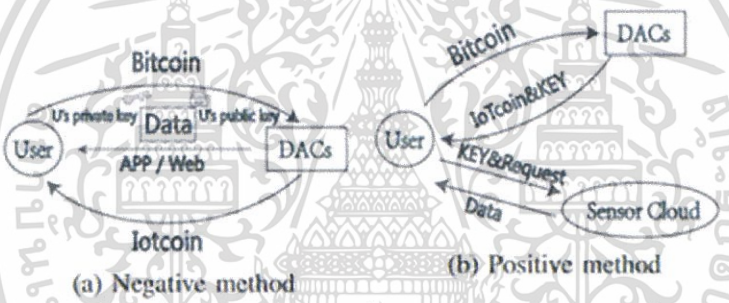
รูปที่ 2.12 การทำการแลกเปลี่ยนกับ Smart Property(1)

จากรูปที่ 2.12 เป็นการที่ Buyer ที่เป็นเจ้าของรถต้องการแลกเปลี่ยน Key ในการสตาร์ทรถ โดยขั้นแรก Seller จะยืนยันตัวตนของ Buyer ก่อน



รูปที่ 2.13 การทำการแลกเปลี่ยนกับ Smart Property(2)

จากนั้นก็ทำการแลกเปลี่ยน Bitcoin กับ Carcoin ซึ่งก็คือข้อมูลของ Key ในการสตาร์ทรถ ในรูปที่ 2.13



รูปที่ 2.14 การแลกเปลี่ยนกับ Paid Data

จากรูปที่ 2.14 เป็นการที่ User ต้องการที่จะแลกเปลี่ยน Paid Data โดยมีสองวิธี วิธีแรกคือการจ่าย Bitcoin ไปแล้วรอรับ Public key สำหรับเข้าถึงข้อมูลนั้นแล้วรับข้อมูลมาเป็น IoTcoin ส่วนวิธีที่สองก็คล้ายกับวิธีแรก แต่การเข้าถึง Data จะต้องใช้ API

ในงานวิจัยนี้เราได้นำการสร้าง Token และการแลกเปลี่ยน Token มาประยุกต์ใช้ ซึ่งวิธีการดังกล่าวเหมาะสมที่จะนำมาใช้ในการแลกเลี่ยนระหว่าง Block ของไฟฟ้า กับ Block ของเงิน เพราะ เป็นการแลกเลี่ยนกันแบบไม่อาศัยตัวกลาง และเป็นแบบอัตโนมัติ

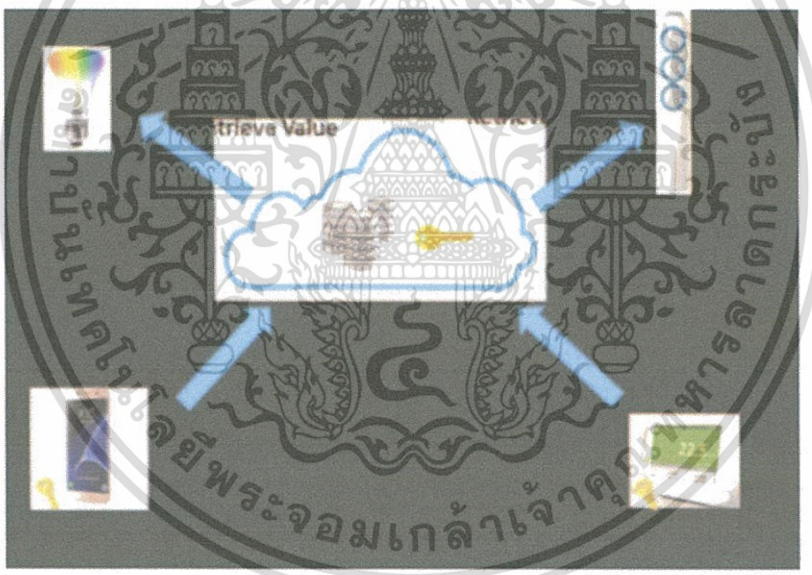
2.5.2 Managing IoT Devices using Blockchain Platform

งานวิจัยนี้กล่าวถึงการประยุกต์การใช้งาน IoT เข้ากับเทคโนโลยี Blockchain เพื่อลดข้อจำกัดของโมเดลรูปแบบ Client-Server โดยการใช้โมเดลรูปแบบ Decentralized ซึ่งไม่จำเป็นต้องมีศูนย์กลางทำให้ป้องกันการปลอมแปลงข้อมูลได้ โดยงานวิจัยนี้ได้เลือกใช้

Ethereum สำหรับเป็น Blockchain Platform เนื่องจากมีการทำ Transaction ที่รวดเร็วกว่า Bitcoin และมีตัว Smart Contract ซึ่งเหมาะสมที่จะนำมา Synchronize กับ IoT Devices

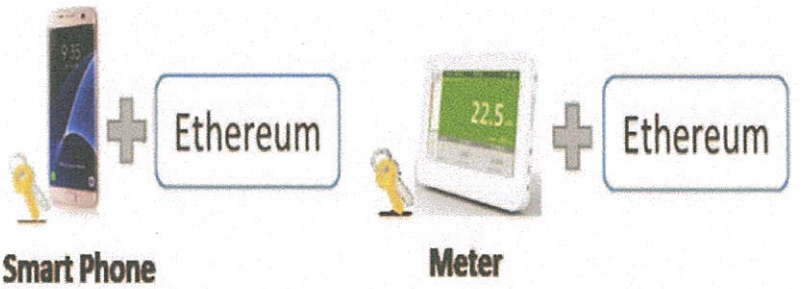
ในการจัดการ IoT devices ด้วยการใช้ Ethereum จะอธิบายโดยแบ่งออกเป็น 3 ส่วน ดังนี้

1) Scenario ในการทดลองนี้จะใช้ Smart Phone 1 เครื่องกับ Raspberry Pi 3 ตัว โดยสำหรับ Raspberry Pi ทั้ง 3 ตัวจะกำหนดให้เป็น Meter สำหรับวัดการใช้ไฟฟ้า, หลอดไฟ และ เครื่องปรับอากาศ ส่วน Smart Phone จะใช้สำหรับตั้งค่า Policy ให้กับแต่ละเครื่องใช้ไฟฟ้า เช่น ถ้าเครื่องปรับอากาศใช้ไฟเกิน 150 KW ให้เปลี่ยนไปใช้งาน Energy Saving ซึ่งค่า Policy ที่ตั้งนั้นจะถูกส่งไปที่ Ethereum Network และเครื่องใช้ไฟฟ้าทั้งสองก็จะได้รับค่า Policy จาก Ethereum Network ส่วน Meter ก็จะอัพเดตค่าจำนวนการใช้ไฟฟ้าไปที่ Ethereum Network เช่นกัน ดังรูปที่ 2.15



รูปที่ 2.15 Scenario Diagram

2) Ethereum Model เนื่องจาก Ethereum เป็นระบบที่ไม่อาศัยศูนย์กลาง ซึ่งจากรูปที่ 2.15 นั้นเป็น Model รูปแบบ Client-Server ดังนั้นระบบของ Ethereum จึงต้องเป็นตามรูปที่ 2.16 ซึ่งแต่ละ Device จะมี Blockchain เป็นของตัวเอง



รูปที่ 2.16 Ethereum Model Diagram

3) Smart Contract เป็น Application ที่ทำงานอยู่บน Ethereum Blockchain ซึ่งในการทดลองนี้จะประกอบไปด้วย 3 Contract ดังนี้

1) Meter Contract จะถูกนำไปใช้กับ Raspberry Pi ที่เป็น Meter เพื่อใช้สำหรับติดตามการใช้ไฟฟ้าของแต่ละเครื่องใช้ไฟฟ้า

```
contract Meter{
  int value;
  bytes publicKey;
  bytes signature;
  update(int _value, bytes _publicKey, bytes _signature){
    value = _value;
    publicKey = _publicKey;
    signature = _signature;
  }
}
```

รูปที่ 2.17 Meter Contract Script

2) Policy Contract เป็น Contract สำหรับตั้งค่า Policy ให้กับ Raspberry Pi ทั้งสองที่เป็นเครื่องใช้ไฟฟ้า โดยจะใช้ Smart Phone เป็นตัวส่งคำสั่ง Policy ไปที่ Policy Contract

```
contract ACPolicy{
  int acLimit;
  bytes publicKey;
  bytes signature;
  update(int _acLimit, bytes _publicKey, bytes _signature){
    acLimit=_acLimit;
    publicKey=_publicKey;
    signature=_signature;
  }
}
```

รูปที่ 2.18 Policy Contract Script

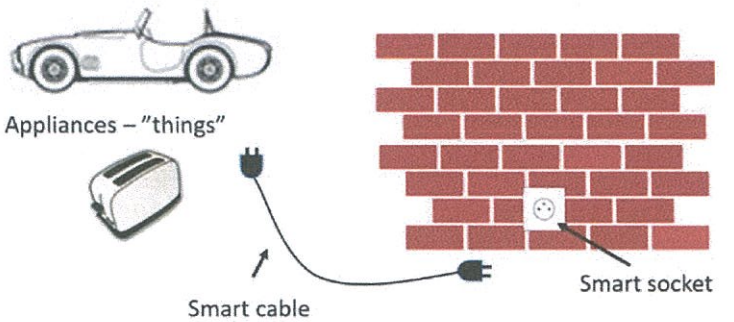
ผลลัพธ์จากการทดลองใช้ Smart Phone ตั้งค่า Policy ให้กับ Raspberry Pi ที่เป็นหลอดไฟ และ เครื่องปรับอากาศคืออุปกรณ์ทั้งสองทำงานตรงตาม Policy ที่กำหนด แต่การทำงานก็ยังมีข้อจำกัดสองประการคือเวลาในการทำ Transaction ที่ค่อนข้างนานประมาณ 12 วินาทีต่อ Transaction จึงอาจไม่เหมาะกำการใช้งานกับบางระบบ และอีกประการหนึ่งคือจำเป็นต้องใช้เนื้อที่ในการเก็บ Blockchain จำนวนมาก

ในงานวิจัยนี้เราได้นำวิธีการที่ติดตั้งโหนดใน Raspberry Pi และ การเขียน Smart Contract เพื่อกำหนด Policy ในการสั่งการอุปกรณ์ไฟฟ้า โดยนำมาประยุกต์ใช้กับการสั่งการโอนเงิน และ การควบคุมการชาร์จไฟ

2.5.3 Thing-to-Thing Electricity Micro Payments Using Blockchain Technology

งานวิจัยนี้กล่าวถึงการทำให้อุปกรณ์ต่างๆ สามารถจ่ายเงินกันเองได้โดยไม่ต้องอาศัยการสั่งการจากมนุษย์ โดยการนำเทคโนโลยีของ Blockchain เข้ามาจัดการการจ่ายเงิน แทนรูปแบบการจ่ายเงินในปัจจุบัน เช่น บัตรเครดิต ซึ่งมีข้อจำกัดในเรื่อง Transaction cost ในการทำ Micro payment ที่ต้องทำ Transaction เป็นจำนวนมาก ซึ่งการใช้ Blockchain นั้นจะทำให้สามารถรองรับการทำ Transaction จำนวนมากๆได้ด้วยการทำงานแบบอัตโนมัติ และด้วยระบบที่เป็น Decentralize ทำให้สามารถสร้างบัญชีให้กับอุปกรณ์ต่างๆได้ง่ายโดยที่ไม่ต้องผ่านระบบศูนย์กลาง

จากรูปที่ 2.19 เป็น Proof-of-Concept ของระบบ ซึ่งจะประกอบไปด้วย Smart cable ที่จะเชื่อมต่อกับ Smart socket เพื่อจ่ายไฟฟ้าให้กับ Things หรืออุปกรณ์ต่างๆตามจำนวนเงินที่จ่าย

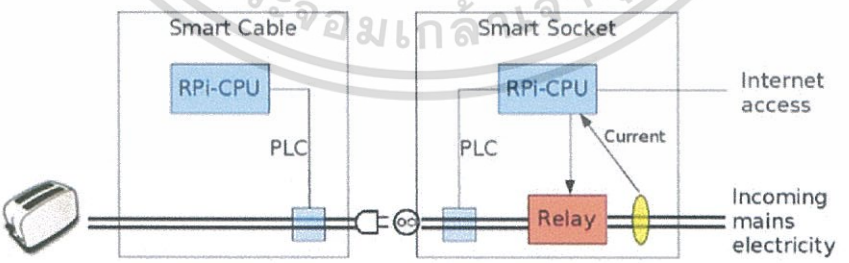


รูปที่ 2.19 The proof-of-concept smart cable and socket

ระบบจะมีการทำงานดังนี้

1) ตอนแรก Smart Socket จะอยู่ใน Standby mode คือจะจ่ายไฟแค่พอเลี้ยงตัว Smart Cable ในเวลาจำกัด
2) เมื่อเสียบสาย Smart Cable เข้ากับ Smart Socket ทั้งคู่จะเริ่มการสื่อสารกัน
3) จากนั้น Smart Cable จะจ่ายไฟฟ้าตามจำนวน Bitcoin ที่จ่ายให้ Smart Socket ถ้า Smart Cable ไม่สามารถจ่ายไฟได้จะเข้าสู่ Lockout mode ชั่วครู่ก่อนเข้าสู่ Standby mode
4) Smart Socket จะตรวจสอบว่า Smart Cable จ่ายไฟครบตามที่กำหนดหรือยัง ก่อนจะตัดสินใจว่าจะเรียกเก็บเงินเพิ่มเพื่อจ่ายไฟต่อไปหรือไม่
5) เมื่อดึง Smart Cable ออก Smart Socket จะกลับสู่ Standby mode

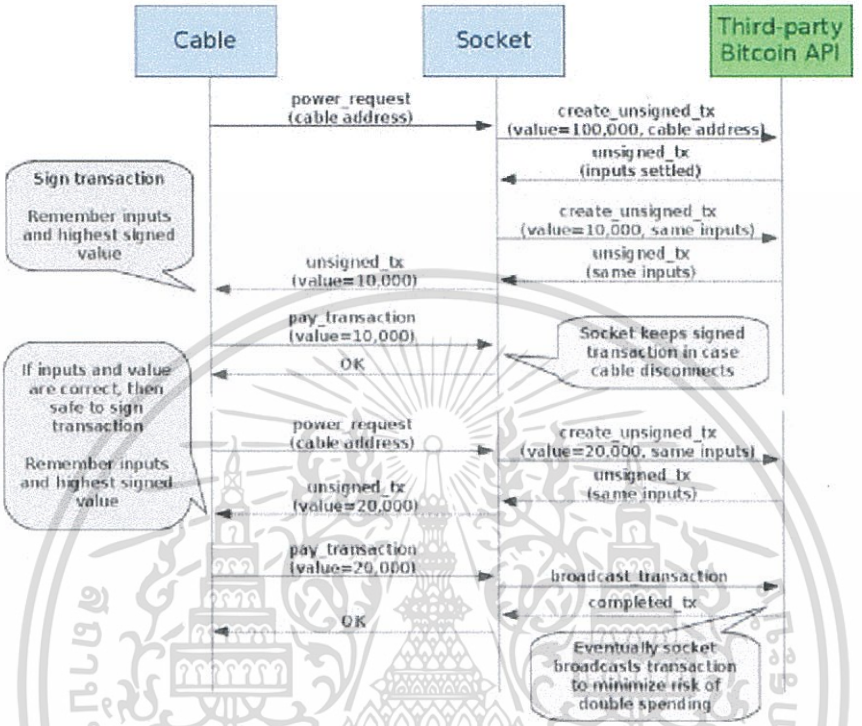
จากรูปที่ 2.20 เป็น Diagram ของ Smart Cable และ Smart Contact ซึ่งจะใช้ Raspberry Pi ในการควบคุม Relay และ วัดกระแสไฟ



รูปที่ 2.20 Diagram of smart cable and socket

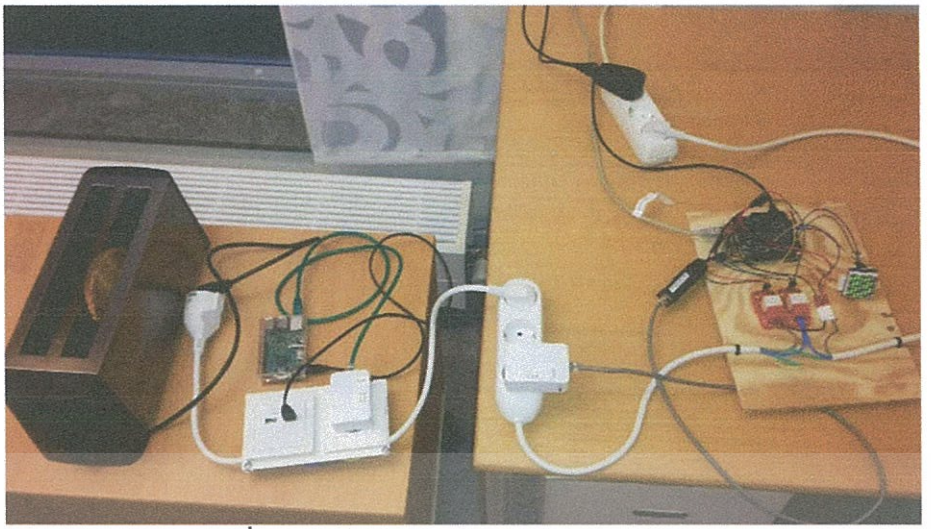
ในรูปที่ 2.21 เป็น Protocol ของการทำ Micro payment โดยเริ่มจาก Cable จะส่ง Request เพื่อขอไฟฟ้าไปที่ Socket จากนั้น Socket จะสร้าง Transaction จาก Bitcoin API เพื่อเรียกเก็บเงินแล้วส่ง Transaction ไปให้ Cable เมื่อ Cable ได้รับ Transaction ก็จะสั่ง

จ่ายเงินกลับไปที่ Socket จากนั้นทางฝั่ง Socket ก็จะเก็บ Transaction นั้นไว้และตัดสินใจว่าจำนวน Transaction มากพอที่จะ Broadcast transaction นั้นไปใน Blockchain ผ่าน Bitcoin API หรือยัง สุดท้าย Socket ก็จะส่ง Acknowledge กลับไปที่ Cable



รูปที่ 2.21 Protocol for micro payments

งานวิจัยนี้ได้ทดลองระบบดังกล่าวกับเครื่องปิ้งขนมปังดังรูปที่ 2.22 ซึ่งผลลัพธ์ที่ได้ก็เป็นไปตาม Concept ที่ออกแบบไว้ ถึงแม้ว่าจะมีปัญหาทางด้านเทคนิคเล็กน้อย เช่น ตัว Prototype ของระบบที่ใหญ่เกินไปสำหรับการใช้งานจริง, การ Boot ตัว Raspberry Pi ที่ค่อนข้างช้า และ เมื่อมีการเชื่อมต่อกับเครื่องใช้ไฟฟ้าที่ใช้พลังงานสูงจะทำให้ Socket เข้า Lockout mode ก่อนที่จะจ่ายเงินเสร็จ นอกจากนี่ยังมีปัญหาด้านความเชื่อใจในตอนที่ Socket ทำการเก็บข้อมูล Transaction ของ Cable เอาไว้ และ ยังมีโอกาสที่จะเกิด Double spending ได้

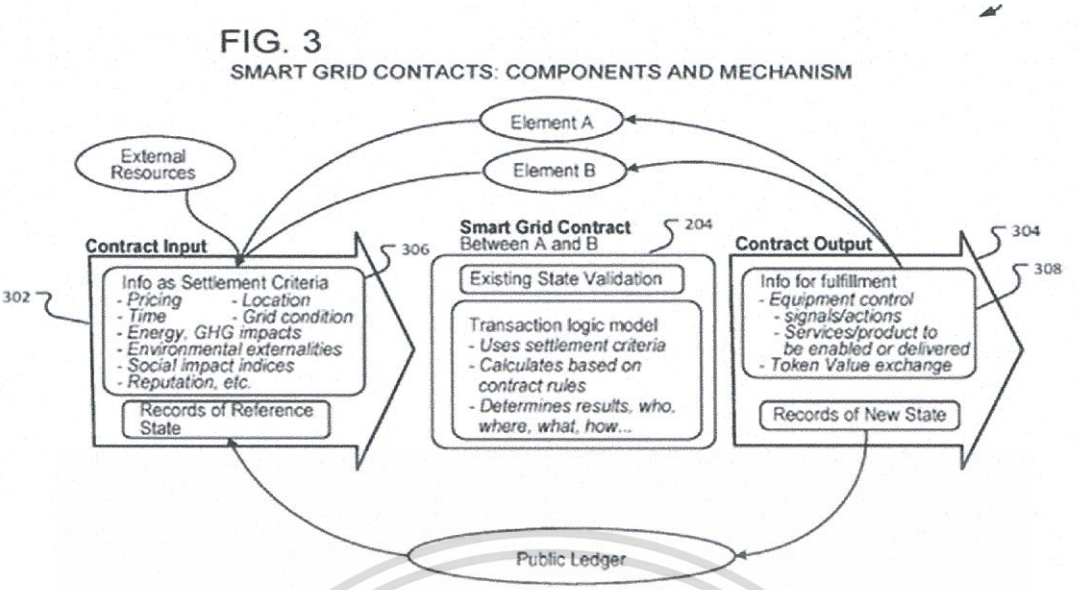


รูปที่ 2.22 The prototype implementation

ในงานวิจัยนี้เราได้นำ Concept ของการที่อุปกรณ์สามารถจ่ายเงินกันเองได้อัตโนมัติ โดยไม่ต้องอาศัยการสั่งการของมนุษย์ มาประยุกต์ใช้กับการจ่ายเงินของสถานีชาร์จไฟ

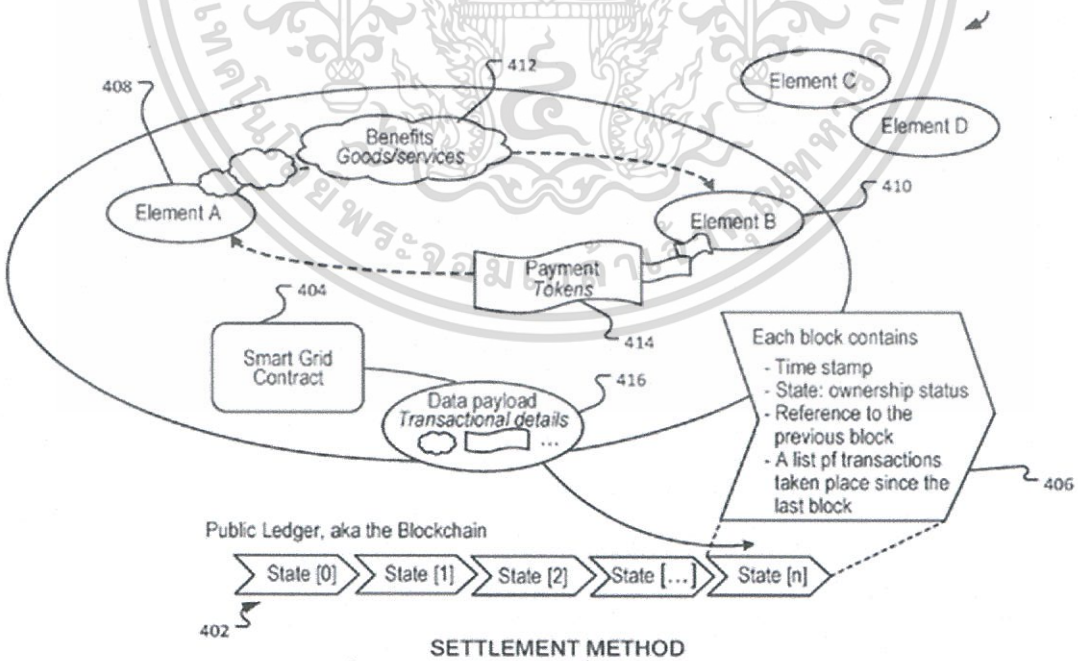
2.5.4 Transactive Grid

Transactive Grid เป็นโครงการซื้อขายไฟภายในเมือง Brooklyn โดยใช้ Blockchain และในรูปที่ 2.23 คือการทำงานของ Smart Grid Contract โดยเริ่มจากแผง Solar Cell จะผลิตไฟฟ้ามาในหน่วย Kilo-Watt แล้วถูก TAGe ซึ่งเป็น Hardware ที่มี Blockchain และ Smart Contract อยู่ภายในแปลงไฟฟ้าเป็น Token ซึ่งประกอบไปด้วย ปริมาณไฟฟ้า, สถานที่, เวลา, TAGe ID, ข้อมูลเกี่ยวกับ Solar Cell, ข้อมูลการติดตั้ง, ขนาดข้อมูล, เจ้าของ Solar Cell, ตัวติดตั้ง, อายุการใช้งาน และ จำนวน TAGe ขั้นต่ำในการทำ Transaction



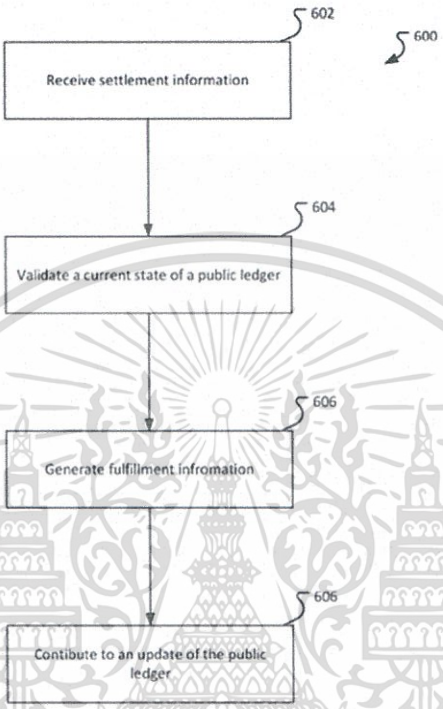
รูปที่ 2.23 ส่วนประกอบและกระบวนการของ Smart Grid Contract

จากนั้น TAGe จะนำข้อมูล Token ดังกล่าวมาสร้าง Smart Contract ซึ่งประกอบไปด้วย เวลา,สถานที่, ชนิดของ Client, ราคาขาย, ราคาขนส่ง และ ตัวแปรที่จำเป็นในการทำ Transaction แล้วส่งขึ้น TAG Network เพื่อนำไปจับคู่กับคู่สัญญาที่ตรงกันแล้วเริ่มทำงาน



รูปที่ 2.24 กระบวนการซื้อขายไฟฟ้า

จากรูปที่ 2.24 Smart Grid Contract ที่อยู่ใน Network รับ Input จาก Element A ที่เป็นผู้ขายและ Element B ที่เป็นผู้ซื้อจากนั้นก็ตัดสินใจว่าจะทำ Transaction อย่างไร เมื่อตัดสินใจได้ก็ส่งผล Output ที่ได้ไปบันทึกลง Blockchain แล้วแลกเปลี่ยนค่า Token กันระหว่าง TAGe ของ Element A และ B เปรียบเสมือนการแลกเปลี่ยนไฟฟ้ากับเงิน



รูปที่ 2.25 Flowchart การทำงานของ Smart Grid Contract

จากรูปที่ 2.25 เป็นขั้นตอนการทำงานของ Smart Grid Contract โดยเริ่มจากรับ Settlement Information เช่น สินค้าและราคาที่จะแลกเปลี่ยนกัน จากอย่างน้อย 2 โหนด แล้วนำไปไป Validate กับข้อมูลใน Blockchain จากนั้นนำไปสร้าง Fulfillment information แล้ว Update ไปที่ Blockchain

ในงานวิจัยนี้ได้อธิบายกระบวนการทำงานของ Smart Contract ภายใน Blockchain ซึ่งมีความเหมาะสมจะนำมาใช้ในระบบการซื้อขายไฟแบบอัตโนมัติ เพราะ ตัว Smart Contract จะทำงานอัตโนมัติด้วยกันรับ Input จากนั้นจะเริ่มการทำงานและส่ง Output ออกมา นอกจากนี้ยังมีการใช้ตัว TAGe ซึ่งเป็นอุปกรณ์ที่ติดตั้ง Blockchain เอาไว้เพื่อให้การซื้อขายไฟเป็นแบบอัตโนมัติ

บทที่ 3
การวิเคราะห์ ออกแบบ และพัฒนาระบบ

ในบทนี้จะอธิบายถึงการวิเคราะห์ ออกแบบและพัฒนาระบบ ซึ่งจะแบ่งออกเป็น 2 ส่วนตามหน้าที่ของ Blockchain node นั้นๆ ได้แก่ Station node ที่ติดตั้งลงใน Raspberry Pi ทำหน้าที่จัดการตัวสถานีชาร์จไฟ และ Application node ที่ติดตั้งลงใน Computer ทำหน้าที่แสดงข้อมูลผ่าน Web application เพื่อสั่งจ่ายเงินและสั่งชาร์จไฟ โดยทั้ง 2 Node จะใช้ Platform ของ go-ethereum และใช้ API ของ web3.js เป็นตัวกลางในการสื่อสารกัน โดยสร้างเป็น Private Blockchain network

3.1 ความต้องการของระบบ

3.1.1 Specification

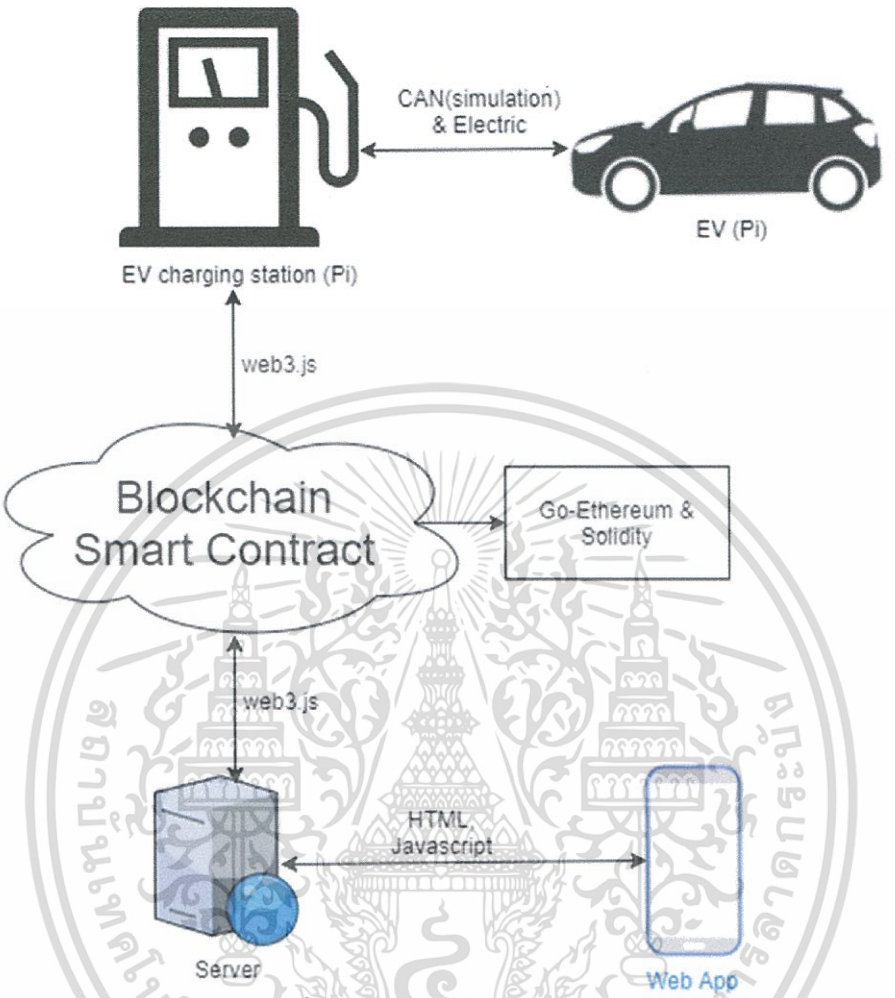
3.1.1.1 Station node

1) นำค่าแบตเตอรี่ที่อ่านได้จากรถยนต์ไฟฟ้าที่จำลองขึ้นจาก Raspberry Pi ไปทำ Transaction เพื่อจัดเก็บลง Blockchain ด้วยการเรียกใช้งานฟังก์ชั่นใน Smart Contract
2) ควบคุมการชาร์จไฟตามคำสั่งใน Smart Contract โดยจะจำลองให้มีการชาร์จไฟทีละ 1 kilowatt
3) ยืนยันการชาร์จไฟด้วยฟังชั่นใน Smart Contract

3.1.1.2 Application node

1) แสดงข้อมูลการชาร์จไฟที่อ่านจาก Smart Contract บน Web application
2) สั่งจ่าย ETH Token ตามจำนวนที่ระบุไว้ใน Smart Contract โดยจะจ่ายตามอัตราค่าชาร์จไฟทีละ 1 kilowatt
3) ยืนยันการจ่ายเงินด้วยฟังชั่นใน Smart Contract
4) สั่งหยุดการชาร์จไฟผ่าน Smart Contract
5) ทำการ Mining เพื่อบันทึก Transaction ลง Blockchain

3.2 โครงสร้างของระบบ



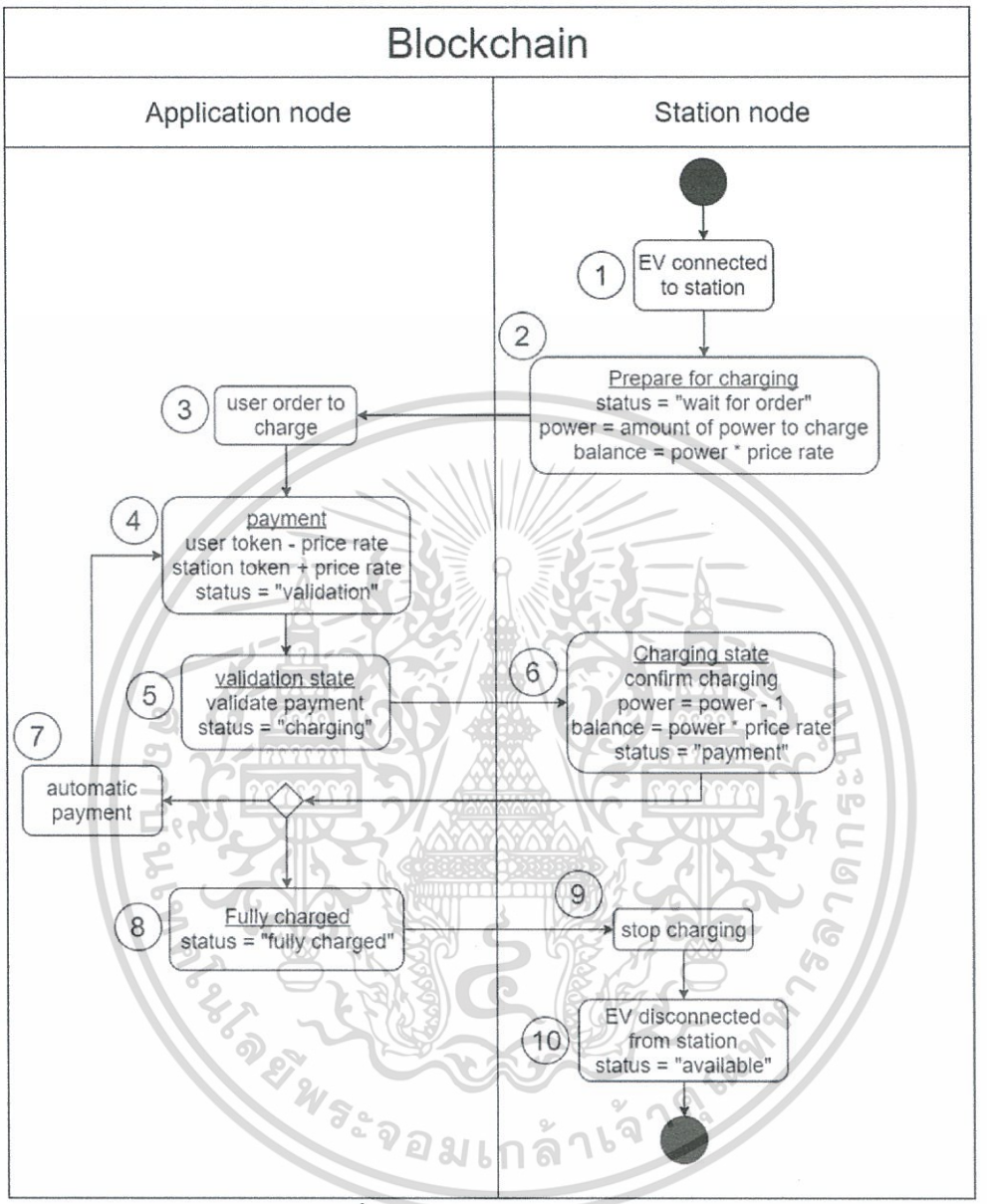
รูปที่ 3.1 System Diagram

จากรูปที่ 3.1 เป็นตัว System Diagram ของระบบซึ่งจะประกอบไปด้วย EV charging station ที่ใช้ Raspberry Pi ที่ติดตั้ง Go-Ethereum Blockchain ไว้เรียกว่า Station node ทำหน้าที่อ่านค่าแบตเตอรี่จาก EV ที่ใช้ Raspberry Pi เขียนโปรแกรมจำลองเป็นรถยนต์ไฟฟ้าแล้วบันทึกลง Blockchain รวมไปถึงอ่านค่าจาก Blockchain

นอกจากนี้ก็จะมี Server ที่ติดตั้ง Go-Ethereum Blockchain ไว้เรียกว่า Application node ทำหน้าที่อ่านค่าการชาร์จไฟจาก Blockchain แล้วส่งไปแสดงผลบน Web Application และทางฝั่ง Web Application สั่งการชาร์จไฟผ่านทาง Server แล้วบันทึกคำสั่งลง Blockchain ได้

ในส่วนของ Blockchain Smart Contract ก็จะใช้ Platform ของ Go-Ethereum และ ใช้ภาษา Solidity ในการเขียน Smart Contract

3.3 Activity Diagram

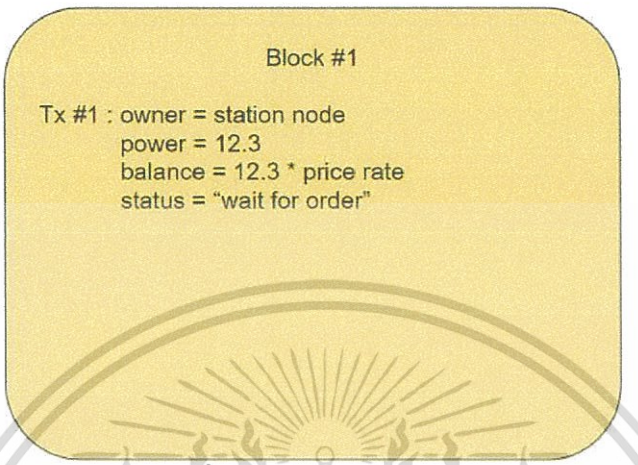


รูปที่ 3.2 Activity Diagram

จากรูปที่ 3.2 เป็น Activity Diagram ที่แสดง State การทำงานของระบบโดยมีการทำงานดังนี้

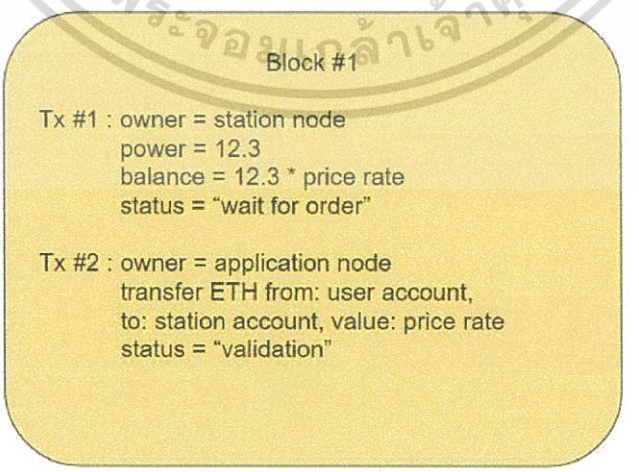
1) จำลองการเชื่อมต่อกันของสถานีชาร์จและรถยนต์ไฟฟ้า โดยให้ Raspberry Pi ที่เป็น Station node อ่านค่า GPIO ว่าถ้ามีค่าเป็น '1' คือมีการเชื่อมต่อกับรถยนต์ไฟฟ้าแล้ว
2) หลังจากที่สถานีชาร์จเชื่อมต่อกับรถยนต์ไฟฟ้าแล้ว Station node จะอ่านค่าแบตเตอรี่ที่จำลองขึ้นจาก Raspberry Pi ที่เป็นจำลองเป็นรถยนต์ไฟฟ้าแล้วสร้าง Transaction เพื่อ Update ตัวแปรใน Smart Contract ดังรูปที่ 3.3 ดังนี้

- status = “wait for order”
- power = จำนวนไฟฟ้าที่ต้องชาร์จ
- balance = จำนวนไฟฟ้าที่ต้องชาร์จ x อัตราค่าชาร์จไฟ



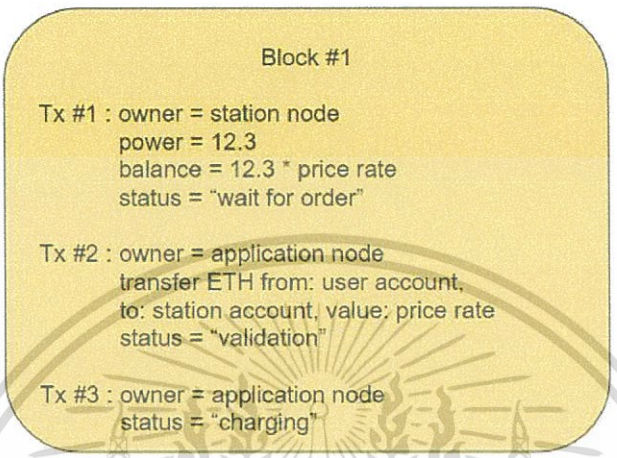
รูปที่ 3.3 Block 1 Tx 1

3) Application node อ่านค่าข้อมูลการชาร์จไฟจาก Smart Contract แล้วนำมาแสดงบน Web application เพื่อให้ User สั่งจ่ายเงินและชาร์จไฟ
4) เมื่อ User สั่งจ่ายเงินและชาร์จไฟแล้ว Application node จะสร้าง Transaction เพื่อโอนเงินและ Update ตัวแปรใน Smart Contract ดังรูปที่ 3.4 ดังนี้
   - โอนเงินจากบัญชีของ User ไปยังบัญชีของสถานีชาร์จไฟนั้นๆ
   - status = “validation”



รูปที่ 3.4 Block 1 Tx 2

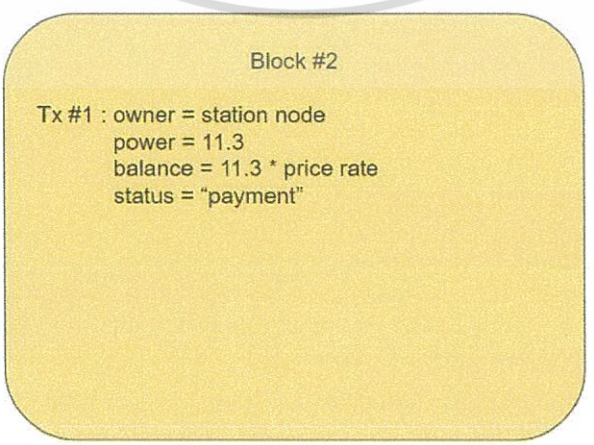
5) เมื่อ status เป็น “validation” แล้ว Application node จะยืนยันการโอนเงินโดยการทำ Transaction เพื่อเรียกใช้งานฟังก์ชั่นสำหรับยืนยันการโอนเงินใน Smart Contract และเมื่อการโอนเงินได้รับการยืนยันแล้วจะ Update ค่า status เป็น “charging” ดังรูปที่ 3.5 แต่ถ้าการโอนเงินไม่ได้รับการยืนยันจะ Update ค่า status เป็น “payment error”



รูปที่ 3.5 Block 1 Tx 3

6) เมื่อ status เป็น “charging” แล้ว Station node จะจำลองสั่งการชาร์จไฟ เมื่อชาร์จเสร็จจะยืนยันการชาร์จไฟ โดยการทำ Transaction ใน Block ต่อไปเพื่อเรียกใช้งานฟังก์ชั่นสำหรับยืนยันการชาร์จไฟใน Smart Contract และเมื่อการชาร์จไฟได้รับการยืนยันแล้วจะ Update ค่าตัวแปรดังรูปที่ 3.6 ดังนี้

- power = จำนวนไฟที่ต้องชาร์จลดลง 1 kilowatt
- balance = จำนวนไฟฟ้าที่ต้องชาร์จ x อัตราค่าชาร์จไฟ
- status = “payment”



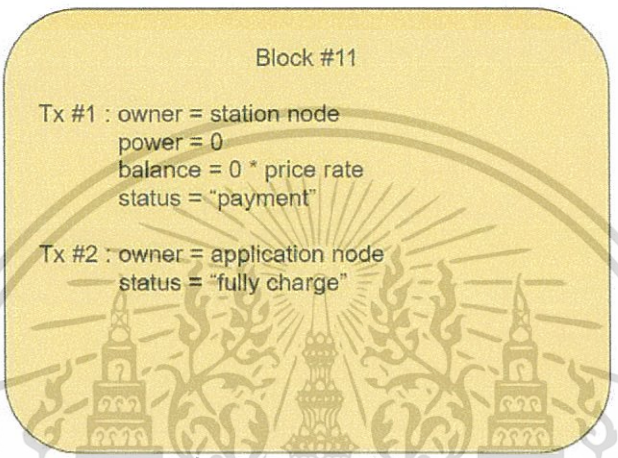
รูปที่ 3.6 Block 2 Tx 1

แต่ถ้าการชาร์จไฟไม่ได้รับการยืนยันจะ update ค่า status เป็น “charging process error”

7) เมื่อ status เป็น “payment” และค่า power มากกว่า 0 Application node จะวนกลับไปทำกระบวนการจ่ายเงินเพื่อชาร์จไฟต่อโดยอัตโนมัติ

8) เมื่อ status เป็น “payment” และค่า power เท่ากับ 0 Application node จะทำ Transaction ใน Block สุดท้ายเพื่อ Update ตัวแปรใน Smart Contract ดังรูปที่ 3.7 ดังนี้

- status = “fully charged”

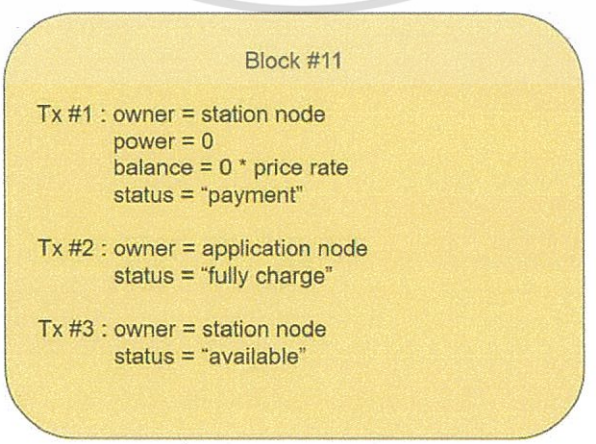


รูปที่ 3.7 Block 11 Tx 2

9) เมื่อ status เป็น “fully charged” แล้ว Station node จะจำลองว่าหยุดการชาร์จไฟแล้ว

10) เมื่อสถานีชาร์จไม่ได้เชื่อมต่อกับรถยนต์ไฟฟ้า คือ Station node อ่านค่า GPIO ว่าถ้ามีค่าเป็น ‘0’ ก็จะทำ Transaction ใน Block สุดท้ายเพื่อ Update ตัวแปรใน Smart Contract ดังรูปที่ 3.8 ดังนี้

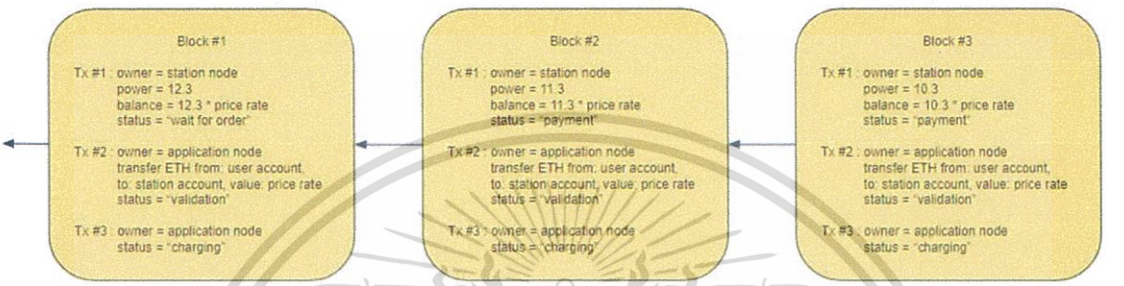
- status = “available”



รูปที่ 3.8 Block 11 Tx 3

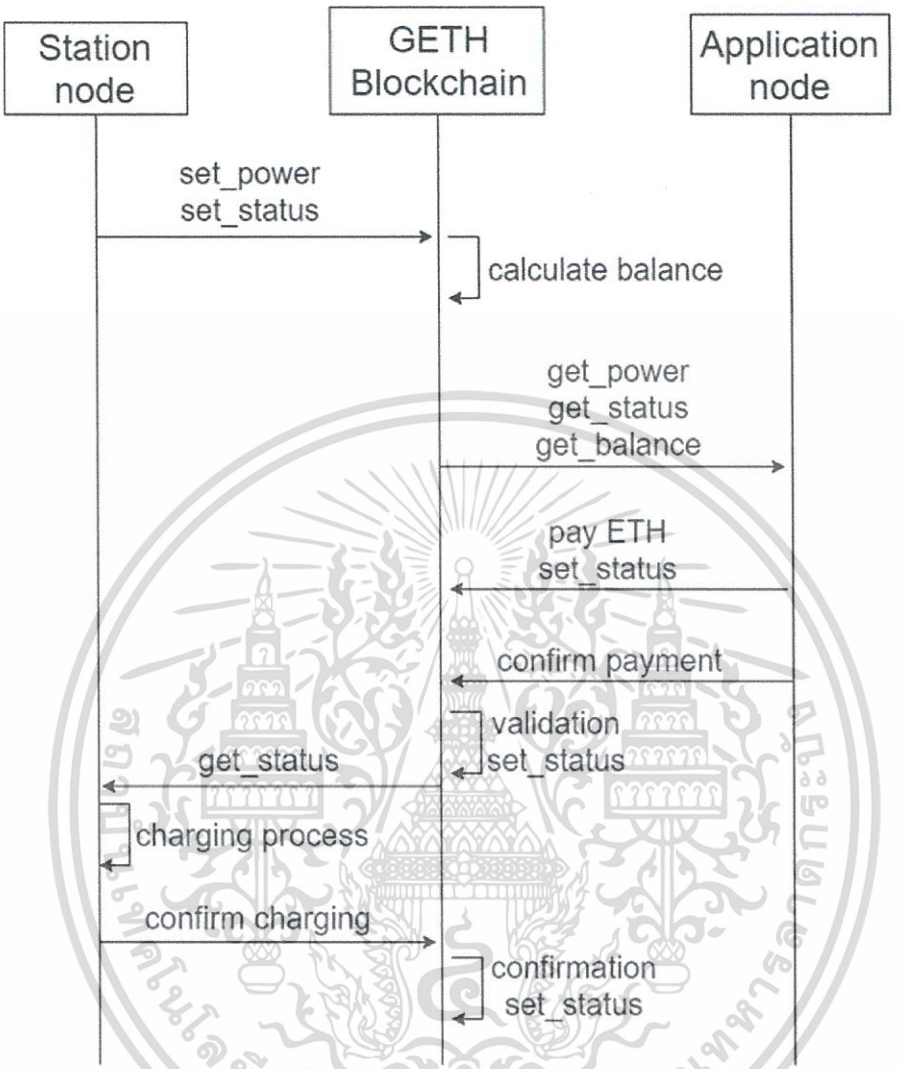
นอกจากนี้ในระหว่างทำกระบวนการชาร์จไฟอยู่ User สามารถสั่งหยุดการชาร์จไฟได้ด้วยตัวเองผ่าน Web application โดย Application node จะทำ Transaction เพื่อ Update ค่า status เป็น “stopping” จากนั้น Station node เมื่อเห็น status เป็น “stopping” ก็จะจำลองว่าหยุดการชาร์จไฟแล้ว ทำ Transaction เพื่อ Update ค่า status เป็น “stopped”

ในรูปที่ 3.9 เป็น Blockchain ที่แสดงกระบวนการชาร์จไฟที่เกิดขึ้นในแต่ละ block ตามกระบวนการที่ได้อธิบายมาข้างต้น



รูปที่ 3.9 EV Charging Blockchain

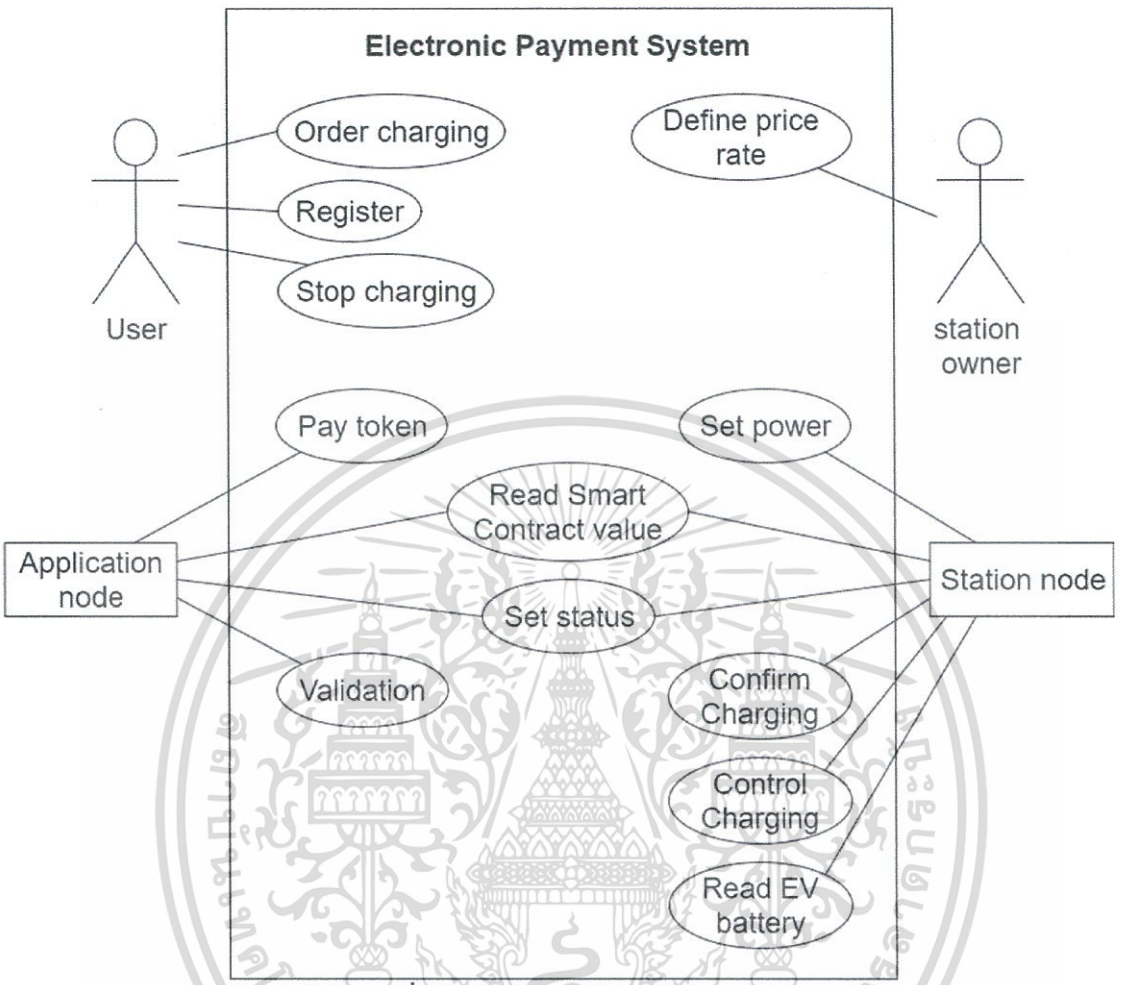
3.4 Behavior Diagram



รูปที่ 3.10 Behavior Diagram

จากรูปที่ 3.10 เป็น Behavior Diagram ที่แสดง state การทำงานของระบบโดยเริ่มต้นจาก Station node เรียกใช้คำสั่ง set_power และ set_status บันทึกค่าแบตเตอรี่ที่อ่านได้และสถานะของการชาร์จลง Blockchain จากนั้น Blockchain จะคำนวนจำนวนเงินที่ต้องจ่ายแล้ว Application node จะเรียกดูข้อมูลการชาร์จไฟด้วยคำสั่ง get_power, get_status และ get_balance ต่อมา Application node จะสั่งจ่ายเงินและยืนยันการจ่ายเงินด้วยคำสั่ง pay ETH และ confirm payment จากนั้น Blockchain จะยืนยันความถูกต้องแล้วอัพเดตค่าสถานะการชาร์จไฟและ Station node จะเรียกดูค่าสถานะแล้วเริ่มการชาร์จไฟจากนั้นก็จะยืนยันการชาร์จไฟไปที่ Blockchain

3.5 Use Case Diagram



รูปที่ 3.11 Use Case Diagram

จากรูปที่ 3.11 เป็น Use Case Diagram ของระบบประกอบไปด้วย actor คือ User, station node, Application node และ Station node นอกจากนี้ก็จะมี Use Case และ คำอธิบาย Use Case ต่างๆดังตารางที่ 3.1 ถึงตารางที่ 3.12

3.6 Use Case Detail

ตารางที่ 3.1 Order Charging Use Case

Use Case Name	Order Charging
Actors	User
Use Case Purpose	User สั่งโอนเงินผ่าน Web application
Pre-conditions	Station node ต้องมีค่า status เป็น "wait for order"
Post-conditions	User สั่งโอนเงินผ่าน Application ได้
Limitations	-
Primary Scenario	1) Web application แสดงข้อมูลการชาร์จไฟที่อ่านจาก Smart Contract 2) User กรอก stationID แล้วเลือกปุ่ม Charge บน Web application 3) User กรอก userID และ Password เพื่อยืนยันความเป็นเจ้าของบัญชี
Alternative Scenario	-

ตารางที่ 3.2 Register Use Case

Use Case Name	Register
Actors	User
Use Case Purpose	User สมัครบัญชีของ Go-ethereum ผ่าน Web application
Pre-conditions	
Post-conditions	User สมัครบัญชีของ Go-ethereum ผ่าน Web application ได้
Limitations	-
Primary Scenario	User เลือกปุ่ม Register แล้วกรอก userID และ Password
Alternative Scenario	-

ตารางที่ 3.3 Stop Charging Use Case

Use Case Name	Stop charging
Actors	User
Use Case Purpose	User สั่งหยุดการชาร์จไฟผ่าน Web application
Pre-conditions	ระบบอยู่ในกระบวนการชาร์จไฟ
Post-conditions	User สั่งหยุดการชาร์จไฟผ่าน Web application ได้

Limitations	-
Primary Scenario	User เลือกปุ่ม Stop เพื่อสั่งหยุดชาร์จไฟ
Alternative Scenario	-

ตารางที่ 3.4 Define Price Rate Use Case

Use Case Name	Define price rate
Actors	Station owner
Use Case Purpose	Station owner กำหนดอัตราค่าชาร์จไฟที่สถานีชาร์จไฟ
Pre-conditions	Station node ต้องมีค่า status เป็น “available”
Post-conditions	Station owner กำหนดอัตราค่าชาร์จไฟที่สถานีชาร์จไฟได้
Limitations	-
Primary Scenario	Station owner กำหนดอัตราค่าชาร์จไฟที่สถานีชาร์จไฟ
Alternative Scenario	-

ตารางที่ 3.5 Pay Token Use Case

Use Case Name	Pay token
Actors	Application node
Use Case Purpose	Application node โอน ETH Token จากบัญชีหนึ่งไปยังอีกบัญชีหนึ่ง
Pre-conditions	บัญชีต้นทางต้องถูก Unlock ด้วย Password
Post-conditions	Application node โอน ETH Token จากบัญชีหนึ่งไปยังอีกบัญชีหนึ่งได้
Limitations	-
Primary Scenario	1) Application node ทำ Transaction เพื่อโอน ETH Token จากบัญชีหนึ่งไปยังอีกบัญชีหนึ่ง 2) เมื่อโอนสร็จสิ้นจะ Update ค่า status เป็น “validation”
Alternative Scenario	-

ตารางที่ 3.6 Validation Use Case

Use Case Name	Validation
Actors	Application node

Use Case Purpose	Application node ยืนยันการโอน ETH Token
Pre-conditions	Application node โอน ETH Token จากบัญชีหนึ่งไปยังอีกบัญชีหนึ่งแล้ว
Post-conditions	Application node ยืนยันการโอน ETH Token ได้
Limitations	-
Primary Scenario	1) Application node เรียกใช้ฟังชั่น confirmPayment จาก Smart Contract 2) เมื่อยืนยันเสร็จสิ้นจะ Update ค่า status เป็น “charging”
Alternative Scenario	-

ตารางที่ 3.7 Read Smart Contract Value Use Case

Use Case Name	Read Smart Contract value
Actors	Application node, Station node
Use Case Purpose	Application node และ Station node อ่านค่าตัวแปรต่างๆจาก Smart Contract
Pre-conditions	Application node และ Station node ต้องเชื่อมต่อกันใน Network
Post-conditions	Application node และ Station node อ่านค่าตัวแปรต่างๆจาก Smart Contract ได้
Limitations	-
Primary Scenario	Application node และ Station node เรียกใช้ฟังก์ชั่นในการเรียกดูตัวแปรแต่ละตัวใน Smart Contract
Alternative Scenario	-

ตารางที่ 3.8 Set Status Use Case

Use Case Name	Set status
Actors	Application node, Station node
Use Case Purpose	Application node และ Station node กำหนดตัวแปร status ใน Smart Contract
Pre-conditions	Application node และ Station node ต้องเชื่อมต่อกันใน Network
Post-conditions	Application node และ Station node กำหนดตัวแปร status ใน Smart

	Contract ได้
Limitations	-
Primary Scenario	Application node และ Station node เรียกใช้ฟังก์ชั่นในการกำหนดตัวแปร status ใน Smart Contract
Alternative Scenario	-

ตารางที่ 3.9 Set Power Use Case

Use Case Name	Set power
Actors	Station node
Use Case Purpose	Station node กำหนดตัวแปร power ใน Smart Contractสำเร็จ
Pre-conditions	Station node อ่านค่า Battery ที่จำลองขึ้น
Post-conditions	Station node กำหนดตัวแปร power ใน Smart Contract ได้
Limitations	-
Primary Scenario	1) Station node อ่านค่า Battery ที่จำลองขึ้น 2) Station node เรียกใช้ฟังก์ชั่นในการกำหนดตัวแปร power ใน Smart Contract แล้วกำหนดค่าที่อ่านได้จาก Battery
Alternative Scenario	-

ตารางที่ 3.10 Confirm Charging Use Case

Use Case Name	Confirm charging
Actors	Station Node
Use Case Purpose	Station node ยืนยันการชาร์จไฟ
Pre-conditions	Station node ชาร์จไฟไปทุกๆ 1 kilowatt
Post-conditions	Station node ยืนยันการชาร์จไฟได้
Limitations	-
Primary Scenario	1) Station node เรียกใช้ฟังก์ชั่น confirmCharging จาก Smart Contract 2) เมื่อยืนยันเสร็จสิ้นจะ Update ค่า status เป็น “payment”
Alternative Scenario	ถ้าไม่ได้รับการยืนยันจะ Update ค่า status เป็น “charging process error”

ตารางที่ 3.11 Control Charging Use Case

Use Case Name	Control charging
Actors	Station Node
Use Case Purpose	Station node จำลองควบคุมกระบวนการชาร์จไฟ
Pre-conditions	-
Post-conditions	Station node จำลองควบคุมกระบวนการชาร์จไฟได้
Limitations	-
Primary Scenario	1) Station node สั่งชาร์จไฟเมื่อมีการยืนยันการโอนเงิน 2) Station node หยุดการชาร์จไฟเมื่อชาร์จเสร็จสิ้น หรือ User สั่งหยุดการชาร์จไฟ
Alternative Scenario	-

ตารางที่ 3.12 Read EV Battery Use Case

Use Case Name	Read EV battery
Actors	Station Node
Use Case Purpose	Station node จำลองการอ่านค่า Battery จาก EV
Pre-conditions	-
Post-conditions	Station node จำลองการอ่านค่า battery จาก EV ได้
Limitations	-
Primary Scenario	Station node อ่านค่าจาก Raspberry Pi ที่จำลองเป็น EV ด้วย UART protocol
Alternative Scenario	-

3.7 Smart Contract Code

ในโปรแกรมที่ 3.1 เป็น Pseudo Code ของ Smart Contract ซึ่งจะประกอบไปด้วยตัวแปรสำคัญๆเช่น ชื่อสถานีชาร์จ, บัญชีของสถานีชาร์จ, บัญชีของผู้ใช้, อัตราค่าชาร์จไฟ, จำนวนเงินและจำนวนไฟที่ต้องชาร์จ นอกจากนี้จะมีฟังก์ชั่นต่างๆดังตารางที่ 3.13

โปรแกรมที่ 3.1 Smart Contract Pseudo Code

```
string stationName;
string stationAccount;
string accountAddress;
uint priceRate;
uint pirce;
uint watt;
uint checkToken;
string status;
function confirmPayment(stationID, amount of paid)
{
     if payment was confirmed; status = "charging"
     else status = "payment error"
}
Function confirmCharge(stationID, amount of paid)
{
     if charging was confirmed; status = "payment"
     else status = "charging process error"
}
```

3.8 Smart Contract Function

ตารางที่ 3.13 Smart Contract Function

Function	Description
confirmPayment	ยืนยันว่าโอน eth ถูกต้องหรือไม่ โดยนำจำนวน eth ที่โอนไปมาตรวดสอบว่าเท่ากับค่า Price rate หรือไม่ ถ้าเท่ากันจะ Update ตัวแปร status เป็น "charging" แต่ถ้าไม่เท่ากันจะ Update ตัวแปร status เป็น "payment error"
confirmCharge	ยืนยันว่าชาร์จไฟถูกต้องหรือไม่ โดยนำค่าแบตเตอรี่ที่อ่านได้มาเปรียบเทียบกับค่าแบตเตอรี่ก่อนชาร์จที่บันทึกไว้ ถ้าค่าต่างกัน 1 kilowatt จะ Update ตัวแปร status เป็น "payment" แต่ถ้าไม่จะ Update ตัวแปร status เป็น "charging process error"
get และ set ตัวแปรแต่ละตัว	กำหนดค่าตัวแปรแต่ละตัว และ เรียกดูค่าตัวแปรแต่ละตัว

บทที่ 4

การทดลอง

การทดลองในบทนี้จะเป็นการทดลองใช้เครื่องมือและอุปกรณ์ต่าง ๆ เช่น การติดตั้งโหนดของ Blockchain ลงใน Raspberry Pi, การใช้งานคำสั่งหรือ API ต่างๆ, การชื่อมต่อแต่ละโหนดเข้าด้วยกัน และ การใช้งาน Smart Contract โดยมีการทดลอง ดังนี้

1) การทดลองเชื่อมต่อแต่ละ Node ของ Blockchain เข้าด้วยกัน
2) การทดลองโอน Ether จาก Account ของ Application node ไปยัง Account ของ Station node
3) การทดลอง Update และ เรียกดูค่าตัวแปรต่างๆใน Smart Contract
4) การทดลองการซื้อขายไฟแบบอัตโนมัติผ่าน Smart Contract

4.1 ทดลองการทดลองเชื่อมต่อแต่ละ Node ของ Blockchain เข้าด้วยกัน

4.1.1 วัตถุประสงค์

เพื่อทดลองการติดตั้ง Go-Ethereum Blockchain ลงใน Raspberry Pi และตั้งค่าเพื่อเชื่อมต่อ Node เข้าด้วยกัน

4.1.2 วิธีการทดลอง

1) ติดตั้งและตั้งค่า Go-Ethereum ลงใน Application node และ Station node ดังรูปที่4.1
2) ใช้คำสั่งดังรูปที่4.2 ใน Station node เพื่อเชื่อมต่อกับ Application node โดยใช้ค่า enode เป็นเลขระบุตัวตน
3) ใช้คำสั่ง admin.peers เพื่อเช็ค Node ที่เชื่อมต่ออยู่

```
Welcome to the Geth JavaScript console!

instance: Geth/node1/v1.7.1-stable-05101641/linux-arm/go1.9
coinbase: 0x5b0cd8030e52416b4eaea6395a0d9495d10ec07f
at block: 13222 (Mon, 20 Nov 2017 17:24:11 +07)
 datadir: /home/pi/node1
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txp
ool:1.0 web3:1.0
```

รูปที่ 4.1 เปิดใช้งาน Go-ethereum node

```
admin.addPeer("enode://881ab6d62d5ea7afa41882df21225cd7e45a65c39f7402d038f538856dc436a0510
ef53f6ec488b84723c9fb514bb03c86ec8a3f17383c089de11f4bbe7bae1a@192.168.1.3:30303");
```

รูปที่ 4.2 คำสั่ง addPeer

4.1.3 ผลการทดลอง

ในรูปที่ 4.3 และรูปที่ 4.4 เป็นการเชื่อมต่อ Node ใน Application node และ Station node ซึ่งจะแสดงข้อมูลต่างๆ เช่น IP address ที่เราเชื่อมต่อ

```
> admin.peers
[{
    caps: ["eth/62", "eth/63"],
    id: "881ab6d62d5ea7afa41882df21225cd7e45a65c39f7402d038f538856dc436a0510ef53
f6ec488b84723c9fb514bb03c86ec8a3f17383c089de11f4bbe7bae1a",
    name: "Geth/miner1/v1.7.0-unstable/windows-amd64/go1.8.3",
    network: {
      localAddress: "192.168.1.4:49462",
      remoteAddress: "192.168.1.3:30303"
    },
    protocols: {
      eth: {
        difficulty: 8092192750,
        head: "0xc096c7c5e75ab32f2e6d91d2a9f9b15983a2866d8b81fa7ba96e6e013ae0e04
7",
        version: 63
      }
    }
}]_
```

รูปที่ 4.3 สถานะของ Peer ใน Station node

```
> admin.peers
[{
    caps: ["eth/62", "eth/63"],
    id: "881ab6d62d5ea7afa41882df21225cd7e45a65c39f7402d038f538856dc436a0510ef53
f6ec488b84723c9fb514bb03c86ec8a3f17383c089de11f4bbe7bae1a",
    name: "Geth/miner1/v1.7.0-unstable/windows-amd64/go1.8.3",
    network: {
      localAddress: "192.168.1.5:56654",
      remoteAddress: "192.168.1.3:30303"
    },
    protocols: {
      eth: {
        difficulty: 8092192750,
        head: "0xc096c7c5e75ab32f2e6d91d2a9f9b15983a2866d8b81fa7ba96e6e013ae0e04
7",
        version: 63
      }
    }
}]_
```

รูปที่ 4.4 สถานะของ Peer ใน Application node

4.2 ทดลองโอน Ether จาก Account ของ Application node ไปยัง Account ของ Station node

4.2.1 วัตถุประสงค์

เพื่อทดลองการทำ Transaction ใน Blockchain โดยการโอน Ether จาก Account ของ Application node ไปยัง Account ของ Station node

4.2.2 วิธีการทดลอง

1) ให้ Application node ทำการ Mining ด้วยคำสั่ง miner.start()
2) ใช้คำสั่งดังรูปที่ 4.5 เพื่อทำ Transaction โอน Ether โดย from: คือบัญชีที่จะทำการโอน, to: คือบัญชีที่จะโอนไป และ value: คือจำนวน Ether ที่จะโอนในหน่วย Wei
3) รอให้ Transaction ถูกยืนยันลง Blockchain

```
> eth.sendTransaction({from: eth.accounts[0], to: 0xc08cd176ce0a4bec3bd4fc0bb9b7
7403344b2428, value: web3.toWei(0.01, "ether")});
```

รูปที่ 4.5 คำสั่งทำ Transaction ในการโอน Ether

4.2.3 ผลการทดลอง

1) เมื่อสั่งโอน Ether จะมี Transaction ที่รอการยืนยันอยู่ใน eth.pendingTransactions ดังรูปที่ 4.6 โดยจะระบุข้อมูลที่จะนำไปบันทึกลง Blockchain เช่น Address ของผู้ทำ Transaction (from:), เลข hash ของ Transaction นั้น (hash:), ค่า gas เป็นค่าในการทำ Transaction (gasPrice:), Address ปลายทาง (to:), จำนวน Ether ที่ส่งในหน่วย Wei (value:) เป็นต้น

```
> eth.pendingTransactions
[{
    blockHash: null,
    blockNumber: null,
    from: "0x5b0cd8030e52416b4eaea6395a0d9495d10ec07f",
    gas: 90000,
    gasPrice: 10000,
    hash: "0xf9d08a60ff7db5f47e0e780e96956c3458c9a493228b1e3dc5cc9ad1b3715f7b",
    input: "0x",
    nonce: 298,
    r: "0x41c2bdc56c02a9594982e8366d6a8db3d650f4a0f9a5ca64c0474ebc3f4cecae",
    s: "0x4072e40348a5f0b6c0e159a4caf3057f757a34958dbb86da6e1418ef4e15d944",
    to: "0xc08cd176ce0a4bec3bd4fc0bb9b77403344b2428",
    transactionIndex: 0,
    v: "0x78",
    value: 10000000000000000
}]
```

รูปที่ 4.6 Pending Transactions

2) เมื่อเช็คจำนวน Ether ใน Station Node จะพบว่ามี Ether เพิ่มมา 0.01 ดังรูปที่ 4.7

```
> web3.fromWei(eth.getBalance(eth.accounts[0]),'ether')
10.920391520924810001
> web3.fromWei(eth.getBalance(eth.accounts[0]),'ether')
10.930391520924810001
```

รูปที่ 4.7 จำนวน Ether ใน Car Node

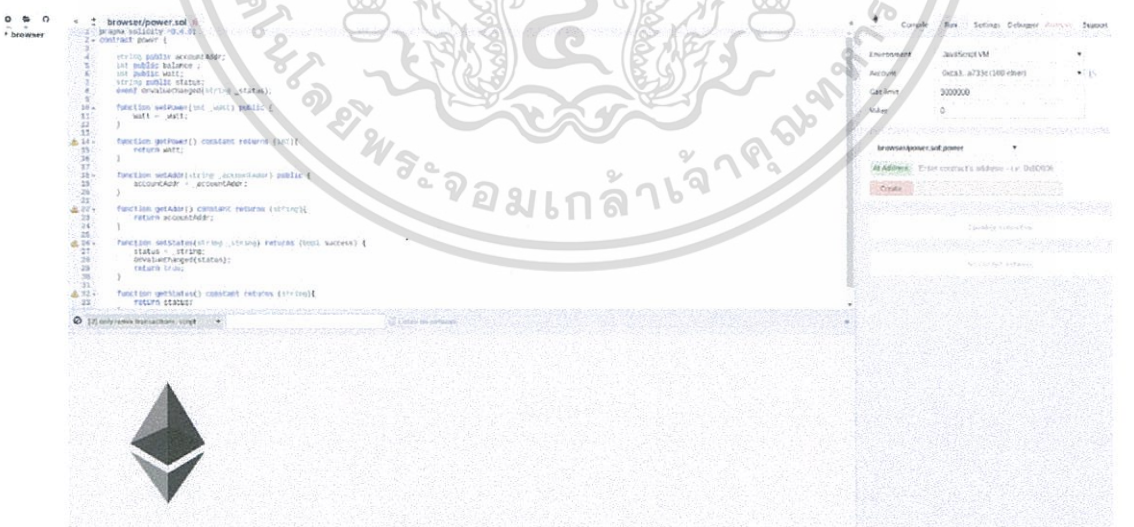
4.3 การทดลอง Update และ เรียกดูค่าตัวแปรต่างๆใน Smart Contract

4.3.1 วัตถุประสงค์

เพื่อทดลองการสร้างและใช้งาน Smart Contract สำหรับการซื้อขายไฟฟ้าแบบอัตโนมัติ

4.3.2 วิธีการทดลอง

1) สร้าง Smart Contract ด้วย Solidity Browser IDE ดังรูปที่ 4.8 และส่งไปใน Go-Ethereum Network เพื่อรอการยืนยัน Transaction แล้วบันทึกลง Blockchain
2) Application Node และ Station Node เรียกใช้งาน Smart Contract
3) เรียกใช้งานฟังก์ชั่นภายใน Smart Contract เพื่อเรียกดูและ Update ตัวแปรต่างๆ ใน Smart Contract



รูปที่ 4.8 Solidity Browser IDE

4.3.3 ผลการทดลอง

ทดลองเรียกใช้งาน Smart Contract และใช้ฟังชั่นในการเรียกดูและ Update ค่าตัวแปรต่างๆ ดังรูปที่ 4.9

```
> contractInstant.getStatus()
"available"
> contractInstant.setStatus("test")
"0xd59cafd10596b7a24fb8b7e0a791dcac4f7baa08c5fea0e1e5eded1d90ad4cc0"
> eth.pendingTransactions
[]
> contractInstant.getStatus()
"test"
> contractInstant.getAddr()
"0xc08cd176ce0a4bec3bd4fc0bb9b77403344b2428"
```

รูปที่ 4.9 เรียกใช้งาน Smart Contract

4.4 การทดลองการซื้อขายไฟแบบอัตโนมัติผ่าน Smart Contract

4.4.1 วัตถุประสงค์

เพื่อทดลองทำการจำลองการซื้อขายไฟแบบอัตโนมัติ ผ่าน Smart Contract

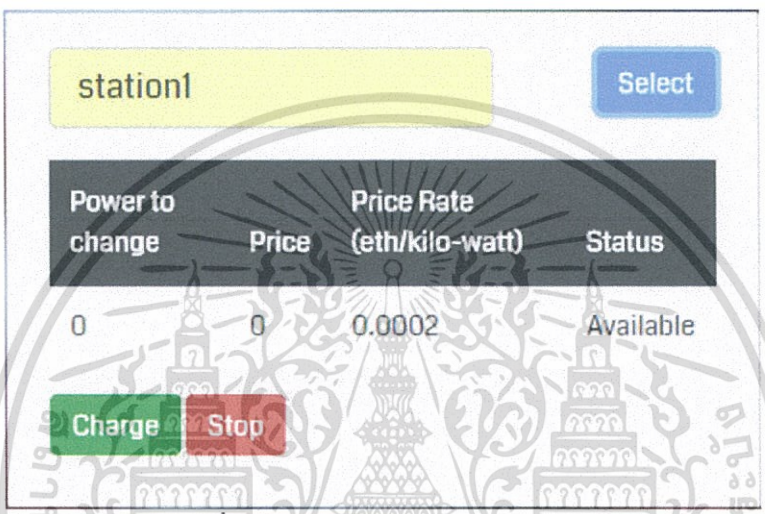
4.4.2 วิธีการทดลอง

1) Station node เชื่อมต่อสาย GPIO กับ Raspberry Pi ที่จำลองเป็น EV เพื่อจำลองการอ่านค่าแบตเตอรี่โดยใช้ UART protocol ในการสื่อสาร
2) Station node นำค่าที่อ่านได้ Update ไปที่ Smart Contract เพื่อคำนวนค่าชาร์จไฟและทำ Transaction เพื่อ Update ค่า status เป็น “Wait for order”
3) Application node อ่านข้อมูลการชาร์จไฟจาก Smart Contract แล้วนำมาแสดงบน Web application
4) User กรอก stationID ของสถานีชาร์จที่เข้าใช้งาน จากนั้นสั่งโอนเงินและชาร์จไฟบน Web application โดย Application node จะทำ Transaction โอนเงินจากบัญชีของ User ไปบัญชีของเจ้าของสถานีชาร์จ และ Update ค่า status เป็น “validation”
5) Application node ทำ Transaction เพื่อยืนยันการจ่ายเงินผ่าน Smart Contract โดยถ้ายืนยันสำเร็จจะ Update ค่า status เป็น “charging”
6) Station node จำลองการชาร์จไฟ จากนั้นทำ Transaction เพื่อยืนยันการชาร์จไฟผ่าน Smart Contract โดยถ้ายืนยันสำเร็จจะ Update ค่า status เป็น “payment”

7) ถ้าค่า power หรือจำนวนไฟที่ต้องชาร์จยังไม่เท่ากับ 0 จะทำกระบวนการจ่ายเงินและชาร์จไฟต่อโดยอัตโนมัติ แต่ถ้า power มีค่าเป็น 0 แล้วจะทำ Transaction เพื่อ Update ค่า status เป็น “Fully Charged” และ Station node จะหยุดการชาร์จไฟ

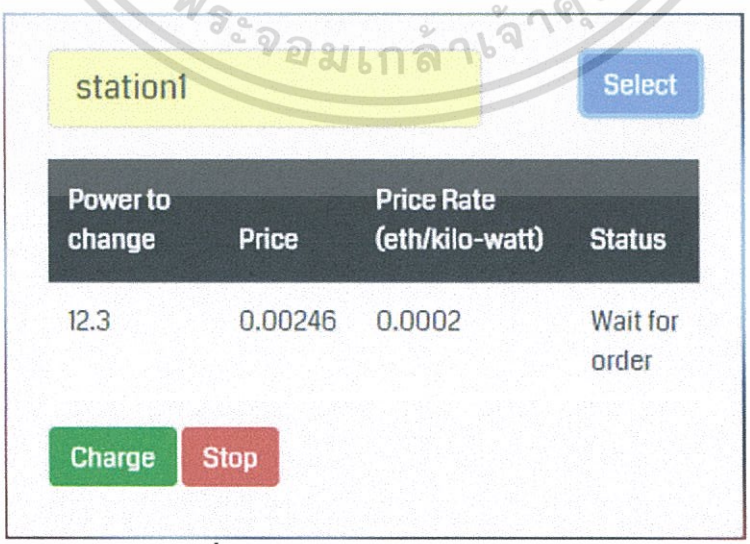
4.4.3 ผลการทดลอง

1) รูปที่ 4.10 เป็นหน้าต่าง Web application ตอนที่ยังไม่ได้เชื่อมต่อสาย GPIO



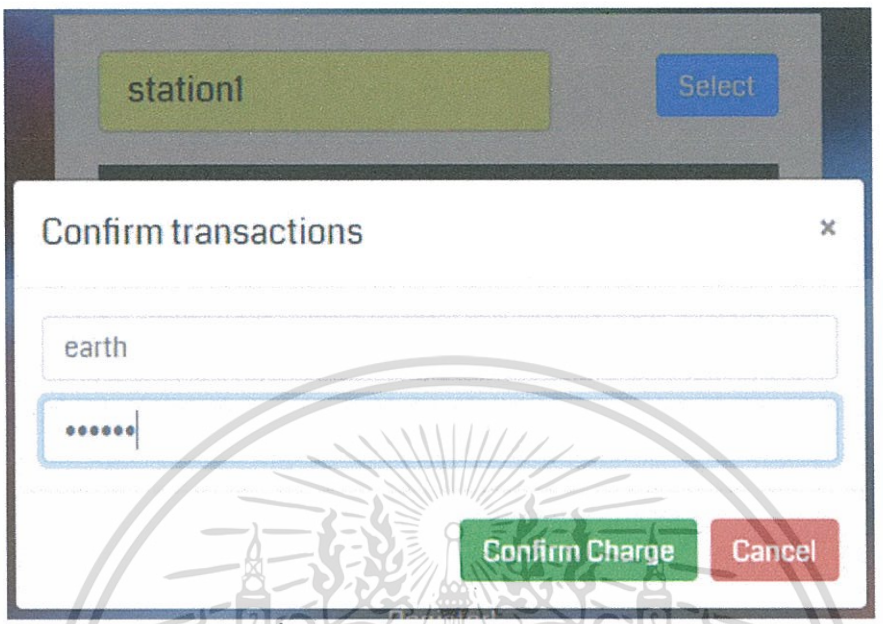
รูปที่ 4.10 หน้าต่าง Application

2) รูปที่ 4.11 เป็นหน้าต่าง Web Application เมื่อเสียบสาย GPIO แล้วรอคำสั่งจาก user



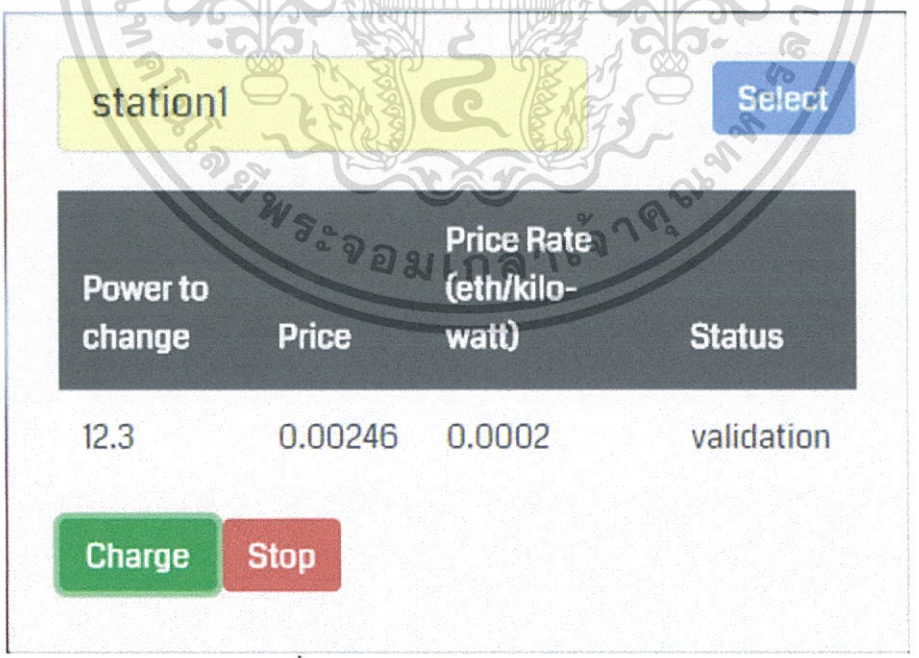
รูปที่ 4.11 หน้าต่าง Application

3) เมื่อ User เลือกปุ่ม Charge ต้องกรอก userID และ Password เพื่อยืนยันความเป็นเจ้าของบัญชีแล้วเลือกปุ่ม Confirm Charge ดังรูปที่ 4.12



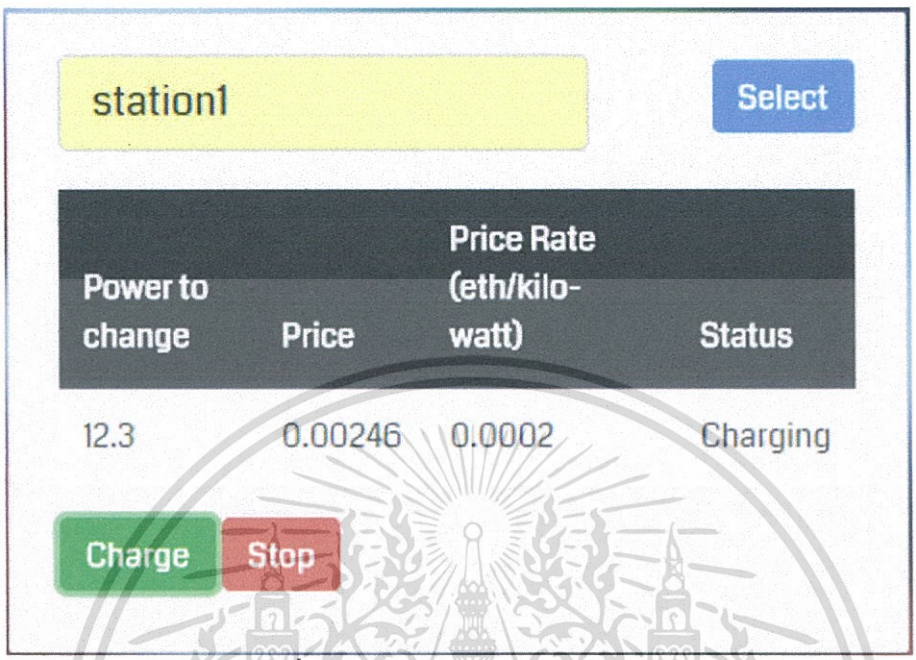
รูปที่ 4.12 หน้าต่าง Application

4) รูปที่ 4.13 เป็น Application node ที่ทำการยืนยันการโอนเงิน



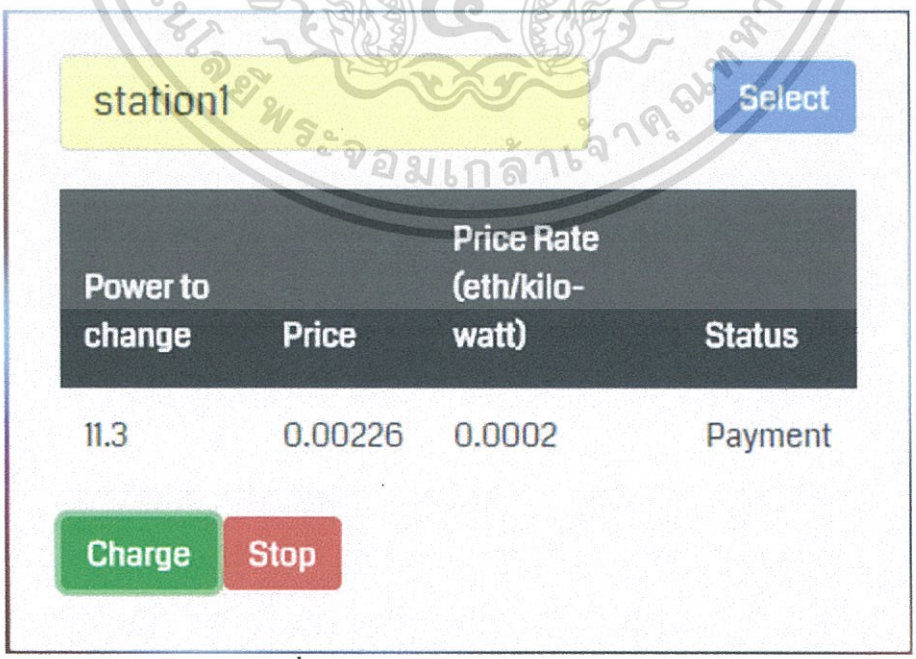
รูปที่ 4.13 หน้าต่าง Application

5) รูปที่ 4.14 เป็นหน้าต่าง Application ตอนที่ Station node จำลองการชาร์จไฟและยืนยันการชาร์จไฟ



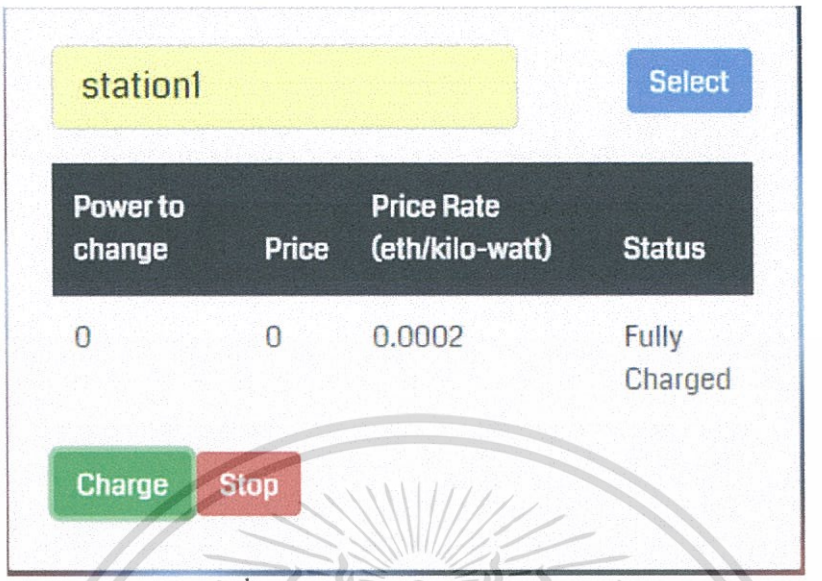
รูปที่ 4.14 หน้าต่าง Application

6) รูปที่ 4.15 เป็นหน้าต่าง Application ตอนที่ ทำกระบวนการจ่ายเงินและชาร์จไฟต่อ โดยอัตโนมัติ



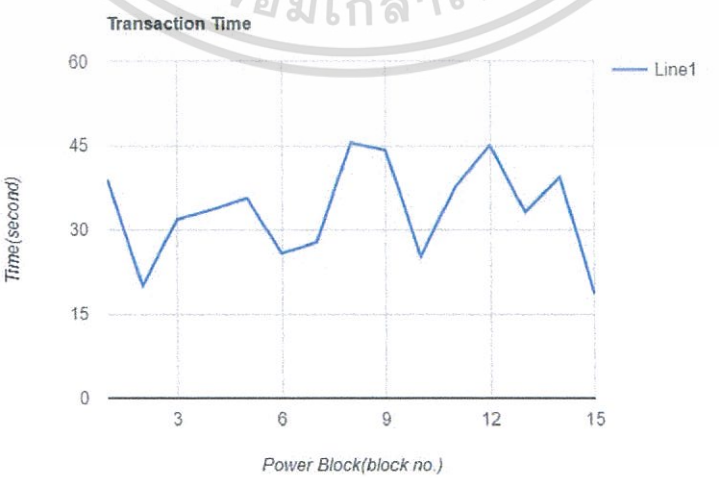
รูปที่ 4.15 Pending Transaction

7) รูปที่ 4.16 หน้าต่าง Application ตอนที่กระบวนการชาร์จไฟเสร็จสิ้น



รูปที่ 4.16 Pending Transaction

ผลการทดลองดังกล่าวแสดงให้เห็นว่าเราสามารถนำการทำ Transaction ของ Blockchain มาใช้ในการควบคุมอุปกรณ์และสร้างเป็นระบบจ่ายเงินแบบอิเล็กทรอนิกส์ได้ ซึ่งเวลาที่ใช้ในการทำ Transaction โดยเฉลี่ยในทุกๆรอบของกระบวนการจ่ายเงินนั้นอยู่ที่ 33.46 วินาทีไม่รวมเวลาในการชาร์จไฟ โดยสเปคของ Hardware ที่ใช้ คือ intel core i7 4710HQ CPU, GTX850M VGA และ RAM 16 GB ซึ่งเราสามารถเวลาของการทำ Transaction ลงได้โดยการเพิ่มประสิทธิภาพของ Mining hardware หรือ ปรับ Mining thread ใน GETH เพิ่มขึ้น โดยในโปรเจคนี้ใช้ 2 Mining thread ดังรูปที่ 4.17



รูปที่ 4.17 กราฟแสดงเวลาในการทำ Transaction ในแต่ละ Power block

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุปของโครงงาน

โครงงานนี้เป็นการสร้างระบบซื้อขายไฟฟ้าอัตโนมัติด้วยเทคโนโลยี Blockchain และ Smart Contract โดยการจำลองการชาร์จไฟของรถยนต์ไฟฟ้ากับสถานีชาร์ไฟ ด้วยการใช้ Raspberry Pi ติดตั้ง Blockchain ไว้เป็น Application node และ Station node ซึ่งแต่ละส่วนมีการทำงานดังนี้

5.1.1 Application node

1) แสดงข้อมูลการชาร์จไฟที่อ่านจาก Smart Contract บน Web application
2) สั่งจ่าย ETH token ตามจำนวนที่ระบุไว้ใน Smart Contract โดยจะจ่ายตามอัตราค่าชาร์จไฟทีละ 1 kilowatt
3) ยืนยันการจ่ายเงินด้วยฟังก์ชั่นใน Smart Contract
4) สั่งหยุดการชาร์จไฟผ่าน Smart Contract
5) ทำการ Mining เพื่อบันทึก Transaction ลง Blockchain

5.1.2 Station node

1) นำค่าแบตเตอรี่ที่อ่านได้จากรถยนต์ไฟฟ้าที่จำลองขึ้นจาก Raspberry Pi ไปทำ Transaction เพื่อจัดเก็บลง Blockchain ด้วยการเรียกใช้งานฟังก์ชั่นใน Smart Contract
2) ควบคุมการชาร์จไฟตามคำสั่งใน Smart Contract โดยจะจำลองให้มีการชาร์จไฟทีละ 1 kilowatt
3) ยืนยันการชาร์จไฟด้วยฟังก์ชั่นใน Smart Contract

5.2 ปัญหาและอุปสรรค

1) การทำ Transaction ภายใน Blockchain ยังค่อนข้างช้า
2) ในอนาคตขนาดของ Blockchain จะมีขนาดใหญ่ขึ้นและอาจไม่พอที่จะจัดเก็บใน Raspberry Pi

5.3 แนวทางในการแก้ปัญหา

1) เปลี่ยนไปใช้ Blockchain platform อื่นเช่น IOTA
2) ลดการทำ Transaction บางส่วนลง

5.4 แนวทางการพัฒนาต่อ

1) ทดลองกับจำนวน Blockchain node ที่มากขึ้น
2) พัฒนาให้สามารถเชื่อมต่อกับ Public Blockchain ได้

บรรณานุกรม

Narayan Prusty. 2017. **Building Blockchain Projects.** Birmingham : Packt Publishing Ltd

Seyoung Huh, Sangrae Cho and Soohyung Kim. 2017. "**Managing IoT Devices using Blockchain Platform.**" Thesis of Electronics and Telecommunications Research Institute.

Kenji Tanaka, Kosuke Nagakubo and Rikiya Abe. 2017. "**Blockchain-based electricity trading with Digitalgrid router.**" Thesis of University of Tokyo.

Yu Zhang and Jiangtao Wen. 2015. "**An IoT Electric Business Model Based on the Protocol of Bitcoin.**" Thesis of Department of Computer Science and Technology, Tsinghua University.

Konstantinos Christidis and Michael Devetsikotis,. 2016. "**Blockchains and Smart Contracts for the Internet of Things.**" Thesis of Department of Electrical and Computer Engineering, North Carolina State University.

Mohammad A. Salahuddin, Ala Al-Fuqaha, Mohsen Guizani, Khaled Shuaib and Farag Sallabi. 2017. "**Softwarization of Internet of Things Infrastructure for Secure and Smart Healthcare.**" Thesis of myCS.

Marco Conoscenti, Antonio Vetro` and Juan Carlos De Martin. 2016. "**Blockchain for the Internet of Things: a Systematic Literature Review.**" Thesis of Nexa Center for Internet & Society DAUIN-Politecnico di Torino.

Kouhei Hayashi, Ryosuke Kato, Ryosuke Torii, Hisao Taoka and Rikiya Abe. 2015. "**Bi-directional power flow through a digital grid router.**" Thesis of Department of Electrical and Electronics Engineering, University of Fukui, Department of Technology Management for Innovation, School of Engineering, The University of Tokyo.

Rikiya Abe, Hisao Taoka and David McQuilkin. 2011. "**Digital Grid: Communicative Electrical Grids of the Future.**" Thesis Of IEEE.

Orsini et al. **Use Of Blockchain Based Distributed Consensus Control.** U.S patent no. 0103468, Apr 2017

Thomas Lundqvist, Andreas de Blanche and H. Robert H. Andersson. "**Thing-to-Thing Electricity Micro Payments Using Blockchain Technology**" Thesis of Department of Engineering Science, University West, Trollhättan, Sweden.

LO3 Energy. 2017. **Microgrid Brooklyn.** [Online]. Available : https://lo3energy.com.

Greentechmedia. 2017. **Blockchain-Enabled Electric Car Charging Comes to California.** [Online]. Available : https://goo.gl/rgdL9H.

Share&Charge. 2017. **What is Share&Charge?.** [Online]. Available : https://shareandcharge.com/.

Medium. 2017. **The first Multipurpose Blockchain enabled EV Charging Station.** [Online]. Available : https://goo.gl/ZgEkkN.

Slock.it. 2017. **Blockchain Energy P2P sharing project Share&Charge going into live Beta.** [Online]. Available : https://goo.gl/g5dRXQ.

Motortrivia. 2012. **PTT Pilot EV Charging Station.** [Online]. Available : https://goo.gl/zG2qkc.

Motortrivia. 2011. **2011 COTY: Nissan LEAF.** [Online]. Available : https://goo.gl/j2Cfv7.

Motortrivia. 2011. **7 บริษัทผู้ผลิตตกลงพัฒนาพอร์ทชาร์จมาตรฐานเดียวกัน.** [Online]. Available : https://goo.gl/kFfzr4.

Eppo. 2015. **โครงการศึกษาการเตรียมความพร้อมรองรับการใช้ยานพาหนะไฟฟ้าในอนาคตสำหรับประเทศไทย.** [Online]. Available : https://goo.gl/xqW1Vv.

Greentransportation. 2017. **EV DC Fast Charging standards – CHAdeMO, CCS, SAE Combo, Tesla Supercharger, etc.** [Online]. Available : https://goo.gl/ciJKF6.

Cryptocoinsnews. 2017. **Hundreds of Charging Stations for Electric Cars Blockchenized with Ethereum in Germany.** [Online]. Available : https://goo.gl/hoQZoE.

Medium. 2017. **Cryptocurrencies with Tim Ferriss, Nick Szabo and Naval Ravikant.** [Online]. Available : https://goo.gl/4Zd6Nb.

Nuuneoi. 2016. **Blockchain คืออะไร? อธิบายแบบละเอียด แต่เข้าใจง่าย.** [Online]. Available : https://goo.gl/PudFJ2.

Nuuneoi. 2016. **Blockchain for Geek ... เบื้องหลังการทำงานฉบับ Technical ตัวอย่างจาก Bitcoin.** [Online]. Available : https://goo.gl/XX5oPu.

Blockchain Fish. 2017. **ว่าด้วยเรื่องการเขียนโค้ด Smart Contract ภาค 3 – Hello world.** [Online]. Available : https://goo.gl/ugWzCc.

Web3.py. 2017. **Web3 API.** [Online]. Available : https://goo.gl/cAsXrM.

Media consensys. 2017. **How to Build a Private Ethereum Blockchain.** [Online]. Available : https://goo.gl/yc2L73.

Github. 2017. **Geth.** [Online]. Available : https://goo.gl/ihtvM9.

Chainskills. 2017. **Create a private Ethereum blockchain with IoT devices.** [Online]. Available : https://goo.gl/zcwiqa.

Github. 2017. **JSON RPC.** [Online]. Available : https://goo.gl/LVoNUd.

Github. 2017. **Browser-Only Solidity IDE and Runtime Environment.** [Online]. Available : https://goo.gl/79GAZr.

Myelifenow. 2017. **CHAdeMO Connector Interface Technical Diagram.** [Online]. Available : https://goo.gl/M19nBv.


สถานีนิเทศโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง