รายงานผลการเข้าฝึกอบรม

Blockchain Programming

จัดโดย

Software Park Thailand

ณ อาคาร Software Park Building จังหวัด นนทบุรี

วันที่ 24-26 พฤษภาคม 2560

ผู้จัดทำ

รองศาสตราจารย์ ณัฐพร เห็นเจริญเลิศ

สาขาวิชาวิทยาศาสตร์และเทคโนโลยี

โครงการนี้ได้รับการสนับสนุนจากทุนพัฒนาบุคลากรประจำปีงบประมาณ 2560

1. ชื่อ น.ส.ณัฐพร **นามสกุล** เห็นเจริญเลิศ **อายุ** 52 **ปี**
ตำแหน่ง รองศาสตราจารย์ **ระดับ** 9 **สังกัด** สาขาวิชาวิทยาศาสตร์และเทคโนโลยี
ไปเข้าร่วมฝึกอบรม เรื่อง Blockchain Programming
วันที่ 24-26 พฤษภาคม 2560 รวมระยะเวลา 3 วัน

2.วัตถุประสงค์ของการประชุม

2.1 เพื่อศึกษาการเขียนโปรแกรม ใน Blockchain Technology

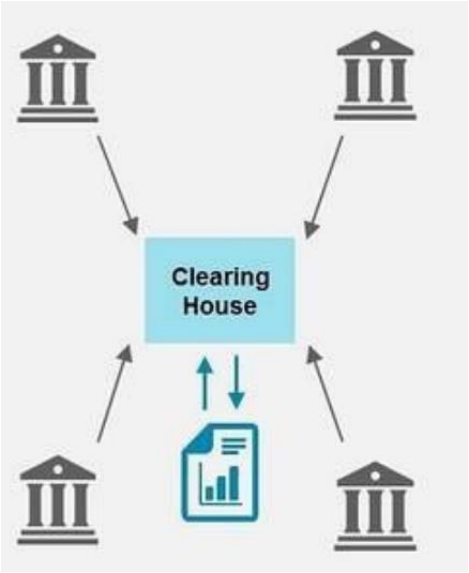
2.2 เพื่อนำความรู้ไปประยุกต์ในการทำวิจัย

3.สรุปเนื้อหา จากการเข้าฝึกอบรม ดังนี้

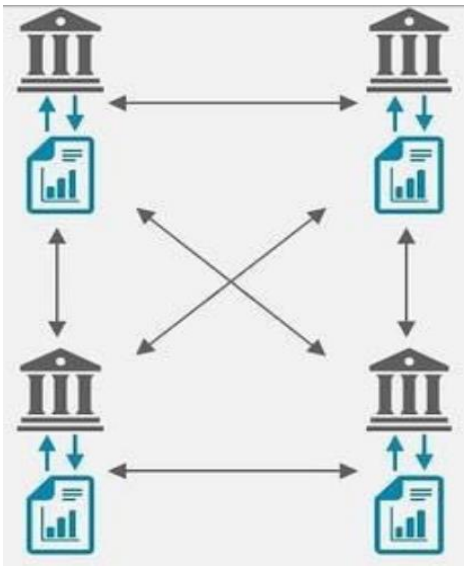
3.1 ความหมายของบล็อกเชน

บล็อกเชน (Blockchain) มีความหมายหลายอย่างจากหลายมุมมองที่ต่างกัน เช่น Blockchain เป็นฐานข้อมูลแบบกระจาย หรือเป็นการบันทึกรายการแบบกระจาย เป็นสกุลเงินอิเล็กทรอนิกส์ เช่นเดียวกับบิทคอยท์ (Bitcoin) เป็นแพล็ตฟอร์มหนึ่งของการกระจายการประมวลผลและการรักษาความมั่นคงปลอดภัย โดยการเข้ารหัสของรายการธุรกรรมต่างๆ ซึ่งมีลักษณะ Peer-to-peer Network

ในธุรกิจแล้ว Blockchain เป็นเทคโนโลยีใหม่ ที่มีประโยชน์ที่นำมาซึ่งความปลอดภัย น่าเชื่อถือ โดยไม่ต้องอาศัยคนกลาง ในการทำธุรกรรมทางการเงิน ดังภาพที่ 1



ก.



ข.

ภาพที่ 1 การทำธุรกรรมทางการเงินในรูปแบบเดิมและ การทำธุรกรรมที่อาศัยเทคโนโลยี Blockchain

การทำธุรกรรมทางการเงินโดยปกติจะเป็นแบบภาพที่ 1 ก คือต้องอาศัยองค์กรกลางเช่น ธนาคาร เป็นศูนย์กลางในการทำธุรกรรมทางการเงิน แต่ถ้าอาศัยเทคโนโลยีบล็อกเชนแล้วจะทำให้ธุรกิจสามารถทำธุรกรรมการเงินได้โดยตรง ไม่ต้องผ่านคนกลาง ดังภาพที่ 1 ข

พัฒนาการของเทคโนโลยี Blockchain แบ่งออกเป็น 3 ช่วงใหญ่ๆ คือ Blockchain1.0, Blockchain2.0, และ Blockchain3.0

- Blockchain 1.0 การพัฒนาในระยะที่ 1.0 ประกอบด้วยสกุลเงินแบบเสมือน (เงินดิจิทัล) เช่น Bitcoin ซึ่งสามารถใช้แทนสกุลเงินจริงได้
- Blockchain 2.0 พัฒนาการใช้รูปแบบ smart contract โดย “smart contract” หมายถึง กระบวนการทางดิจิทัล ที่กำหนดขั้นตอนการทำธุรกรรมโดยอัตโนมัติไว้ล่วงหน้า โดยไม่ต้องอาศัยตัวกลาง อย่างเช่น ธนาคาร ซึ่งการสร้าง smart contract ที่เป็นระบบอัตโนมัติอย่างเต็มรูปแบบ โดยคู่สัญญาทั้งสองฝ่ายจะมีการตกลงกันก่อนหน้านี้ ถึงขั้นตอน กลไกในการทำรายการธุรกรรมดังกล่าว ซึ่งการพัฒนานี้ส่งผลกระทบต่อรูปแบบธุรกิจแบบดั้งเดิมของธนาคาร นักพัฒนาสามารถสร้างแอปพลิเคชันบน Blockchain สาธารณะ หรือ Blockchain ส่วนตัวได้
- Blockchain 3.0 เป็นการพัฒนาแนวคิดเกี่ยวกับ smart contract เพื่อสร้างกระบวนการแบบกระจายศูนย์ที่เป็นอิสระ ที่ต้องมีการกำหนดกฎการทำธุรกรรมของกลุ่มกันเองและดำเนินการด้วยความเป็นอิสระ ในรูปแบบธุรกรรมอัตโนมัติ สามารถนำไปประยุกต์แอปพลิเคชันต่างๆได้หลากหลายมากขึ้นทั้งทางธุรกิจ การเงิน และเศรษฐกิจ

3.2 การทำงานของ Blockchain

การทำงานของ Blockchain พัฒนามาจากลักษณะการบันทึกรายการในสมุดบัญชี ซึ่งจะต้องมีข้อมูลต่างๆ เช่น ยอดเงินที่เกิดขึ้น ใครเป็นเจ้าของรายการ วันเวลาที่เกิดรายการ เป็นต้น ซึ่งลักษณะการทำงานของ Blockchain เป็นแบบ Public Ledger คือ สมุดจดบัญชีที่เปิดเผยสู่สาธารณะ การบันทึกรายการเดินบัญชีก็จะบันทึกต่อกันมาเรื่อยๆ เนื่องจากเป็นระบบเครือข่ายแบบกระจาย จึงต้องมีการสำเนาบัญชีนี้ให้ทุกคนในเครือข่าย ซึ่งก็คือโหนด แต่ละโหนดในเครือข่ายคอมพิวเตอร์ ลักษณะเฉพาะในการทำงานมีดังนี้

1. ทุกๆ ข้อมูลที่มีการบันทึกลงไปใน Blockchain นั้นจะไม่สามารถถูกลบออกไปได้ และสามารถติดตามลำดับการบันทึกข้อมูลย้อนหลังทั้งหมดได้อย่างโปร่งใส

2. ข้อมูลภายใน Blockchain นี้จะถูกกระจายไปจัดเก็บบน Hardware หลายๆ เครื่องซึ่งเราจะเรียก Hardware แต่ละชุดนี้ว่า Node โดยจะมีการรับประกันว่าข้อมูลเหล่านั้นจะเหมือนกันทั้งหมด ซึ่ง Node เหล่านี้จะเก็บเอาไว้ในองค์กรเดียวกัน หรือกระจายช่วยกันเก็บในหลายองค์กรก็ได้เช่นกัน

3. การบันทึกข้อมูลใดๆ ลงไปใน Blockchain นั้นจะต้องได้รับการตรวจสอบและยืนยันจาก Node อื่นๆ ตามเงื่อนไขการตรวจสอบที่กำหนด ก่อนที่จะมีการบันทึกข้อมูลเหล่านั้นเข้าระบบและกระจายให้ Node ต่างๆ บันทึกข้อมูลชุดเดียวกันลงไป เพื่อให้สามารถปรับใช้งานได้ตามความต้องการ

4. รองรับการเข้ารหัสสำหรับข้อมูลแต่ละชุดได้ ดังนั้นถึงแม้ข้อมูลของเราจะถูกกระจายไปยัง Node อื่นๆ และอาจถูกบางคนมองเห็น แต่คนอื่นๆ ก็จะไม่สามารถถอดรหัสข้อความของเราได้ นอกจากตัวเราเองและผู้ที่เกี่ยวข้องกับข้อมูลเหล่านี้ที่เราอนุญาตให้เข้าถึงได้เท่านั้น

การเข้ารหัสจะใช้ Function Hash คือ การเข้ารหัสทางเดียว แล้วจะได้ชุดตัวเลขสั้น ๆ โดยจะสร้าง Digital Signature ของข้อมูล Digital ที่ไม่สามารถถอดรหัสกลับมาได้ และใช้เป็นตัวแทนของข้อมูลนั้นๆ โดยใช้หลักการของ Private key และ Public key

1. ทำการเข้ารหัสด้วย Function Hash โดยใช้ Private Key ของผู้ส่ง ออกมาเป็น Digital Signature

2. เมื่อได้ Digital Signature มาก็จะส่งให้กับผู้รับ พร้อมกับ Public Key

3. ผู้รับตรวจสอบ Digital Signature ที่ได้โดยใช้ Public Key ของผู้ส่งถ้าได้ค่า Hash ที่ตรงกัน ก็ตรวจสอบได้ว่าเป็นข้อความที่ถูกต้อง เชื่อถือได้

3.3 ประเภทของ Blockchain

แบ่งออกเป็น 3 ประเภท คือ Public Private และ Consortium ซึ่งเหมาะสมกับการใช้งานแตกต่างกันไป

- Public Blockchain เป็นการใช้งานบนอินเทอร์เน็ตแบบเปิด ทุกคนที่เข้าสู่เครือข่ายอินเทอร์เน็ตสามารถเข้าใช้งานได้ทั่วโลก เป็นการใช้งานจริงของ Bitcoin และ Ethereum ในฐานะที่เป็นสกุลเงินดิจิทัล ข้อดีของ Public Blockchain คือองค์กรไม่ต้องลงทุนทางด้าน Infrastructure ผู้ใช้เพียงแค่จ่ายค่าการรับส่งบน Public Blockchain จะกลายเป็นว่าข้อมูลนั้นโดนเปิดเผยแก่สาธารณชน ซึ่งจะต้องหาวิธีการหรือโปรแกรมเพื่อห่อหุ้มข้อมูลขององค์กรอีกที

- Private Blockchain เป็นการใช้งานบนเครือข่ายเฉพาะขององค์กรเอง หรือภายในเครือข่ายขององค์กรที่ได้ทำข้อตกลงกันไว้จึงจะเข้าถึงข้อมูลในวงได้ ซึ่งจะทำให้มีข้อจำกัดที่ต้องมีการลงทุนในการสร้าง

ระบบ Infrastructure เอง ข้อดีอีกแบบของ Private Blockchain คือ องค์กรสามารถปรับกฎเกณฑ์ต่างๆของ Blockchain Network ให้ทำงานได้ตามที่ต้องการ ไม่จำเป็นต้องออกแบบระบบให้เป็นไปตามกฎของ Public Public Blockchain เช่น ถ้า Ethereum เวลามีการส่งเงินนั้นก็ต้องออกแบบระบบให้มีการรอ Confirm ธุรกรรม 10-15 นาทีตามกฎของ แต่กลับกันถ้าองค์กรตั้งวง Private Blockchain เองสามารถออกแบบให้การ Confirm ธุรกรรมแล้วเสร็จภายใน 2 วินาทีก็เป็นไปได้

- Consortium Blockchain การรวมกันของ 2 แนวคิดแรกเข้าด้วยกันเป็นการผสาน Public – Private Blockchain เข้าด้วยกัน เป็นการที่องค์กรต่างๆ ที่มีลักษณะธุรกิจเหมือนกันและต้องรับส่งแลกเปลี่ยนข้อมูลกันอยู่แล้ว มารวมกันตั้งสิ่งที่เรียกว่า Consortium Blockchain เช่น Consortium Blockchain สำหรับธนาคารใช้ในการแลกเปลี่ยนข้อมูลการโอนเงินกันภายในสมาคมธนาคารด้วยกัน และธนาคารที่จะเข้ามาร่วมในวงได้ต้องได้รับการอนุญาตจากตัวแทนเสียก่อนถึงจะมีสิทธิเข้าถึงการใช้งานร่วมกันได้ ข้อดีคือ ธนาคารไม่ต้องกลัวว่าข้อมูลสำคัญขององค์กรและลูกค้าจะกลายเป็นข้อมูล Public และในเรื่องการลงทุน Infrastructure ก็ลดลง แต่มีข้อจำกัดด้านความไม่คล่องตัวในการปรับปรุงเปลี่ยนแปลงเงื่อนไขการใช้งานต่างๆ เพราะอาจจะต้องผ่านมติเห็นชอบในสมาคมเสียก่อน เป็นต้น

3.4 การประยุกต์ใช้ Blcokchain

ด้วยคุณสมบัติของ Blockchain สามารถนำไปประยุกต์ทางการเงิน เช่น เป็นสกุลเงินดิจิทัล หรือทางด้านอื่นๆ เช่น บัตรประชาชนดิจิทัลในประเทศ Astonia บันทึกข้อมูลสุขภาพ ทรัพย์สิน โดยเฉพาะอย่างยิ่งใน Blockchain 2.0 และ 3.0 ที่มี smart contact ที่สามารถโปรแกรมเงื่อนไข หรือข้อกำหนดในการทำงานได้ ทำให้ประยุกต์ในด้านต่างๆ ได้มากขึ้น เช่น การตรวจสอบหรือยืนยันทรัพย์สิน การประยุกต์ใน Internet of Things (IoT) ในยุคของ Blockchain จะกลายเป็น Ledger of Things คือมี “บัญชีธุรกรรมอิเล็กทรอนิกส์” เพื่อจะ track อุปกรณ์ที่นำมาเชื่อมต่อ เช่น ในการขนส่ง พาหนะต่างๆ ที่เรานั่งไปนั้นจะช่วยเลือกเส้นทางที่เร็วที่สุดให้ หลีกเลี่ยงจุดก่อสร้าง จัดการเรื่องค่าใช้จ่ายทางด่วน ที่จอดรถ และสื่อสารกับสัญญาณจราจรได้ การขนส่งสินค้าต่างๆ จะจัดการเรื่องการตรวจสอบภาษีได้ไวขึ้น เป็นต้น

3.5 การเขียนโปรแกรมใน Blockchain

การเขียนโปรแกรมใน Blockchain โดยใช้แพล็ตฟอร์มของ Ethereum ซึ่งเป็นสกุลเงินดิจิทัลเช่นเดียวกับ Bitcoin ความนิยมใน Bitcoin ส่งผลให้มีคนจำนวนมากสร้างสกุลเงินดิจิทัลลักษณะเดียวกันขึ้นมา และหนึ่งในนั้นคือ Ethereum เริ่มพัฒนาโดย Vitalik Buterin ชาวรัสเซีย Vitalik เป็นหนึ่งในนักพัฒนา Bitcoin มาก่อน แต่ก็เห็นข้อบกพร่องและข้อจำกัดของ Bitcoin ทำให้เขาหันมาเริ่มสร้าง Ethereum ขึ้นมาใช้แทน ปัจจุบัน Ethereum เป็นซอฟต์แวร์โอเพนซอร์สที่ดูแลโดยหน่วยงานไม่หวังผลกำไร Ethereum Foundation ในสวิตเซอร์แลนด์ ความสามารถของ Ethereum ที่เพิ่มเข้ามาคือฟีเจอร์ที่เรียกว่า Smart

Contracts ที่อนุญาตให้สามารถเขียนโปรแกรมลงไปในข้อมูลของสกุลเงิน Ether ให้ทำงานอัตโนมัติเมื่อเจอเงื่อนไขที่กำหนด ความสามารถนี้ทำให้สามารถสร้างแอพพลิเคชันต่างๆ ขึ้นมาบนเครือข่าย Ethereum อีกชั้นหนึ่ง

3.6 การโปรแกรม Blockchain

1. ติดตั้ง Virtual Machine ของ Ethereum ซึ่งอยู่บน VM ของ Ubuntu โดยเข้าสู่โหมดของ server และสร้างโหนดต่างๆ ขึ้นบน VM เพื่อจำลองว่าเป็นโหนดที่อยู่ในเครือข่าย Ethereum ที่เป็น private network และมีการรับส่งข้อความ หรือข้อมูลต่างๆ ในเครือข่าย สามารถใช้คำสั่งของ Java Script ,Solidity และคำสั่งระบบของ VM ในการโปรแกรม ตัวอย่างคำสั่ง บน console mode

-## Starting Ethereum

-Starting Ethereum node and connect to public test network

-cd eth-intro

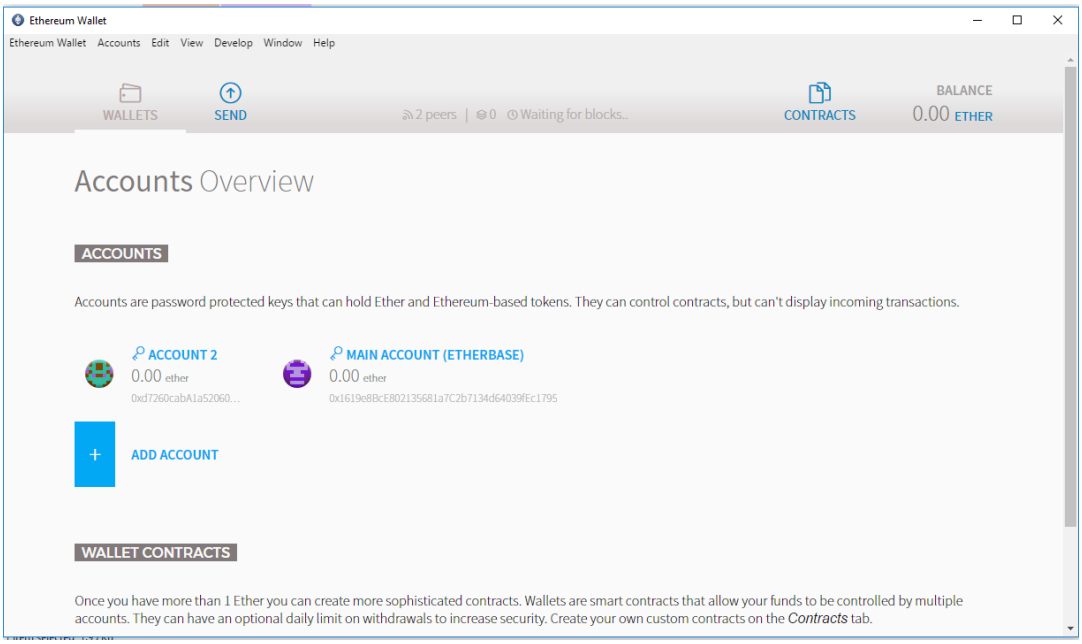
-./ge.sh console

-Connect to the local node. Do this on a separate ssh prompt.

-geth attach http://localhost:8545

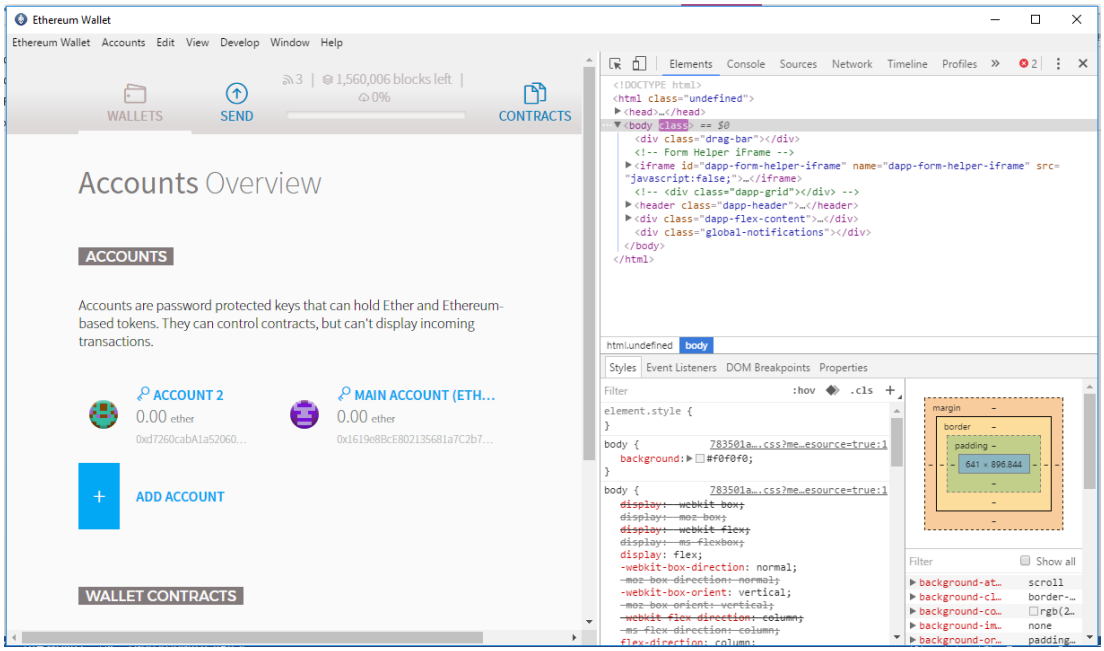
-geth attach ipc://$HOME/.ethereum/testnet/geth.ipc

2. ติดตั้ง Ethereum โดยการดาวน์โหลด Mist/Eteherum v0.8.10 ตัว Mist เป็น browser ในการทำ decentralized applications on the Ethereum blockchain ดังภาพ



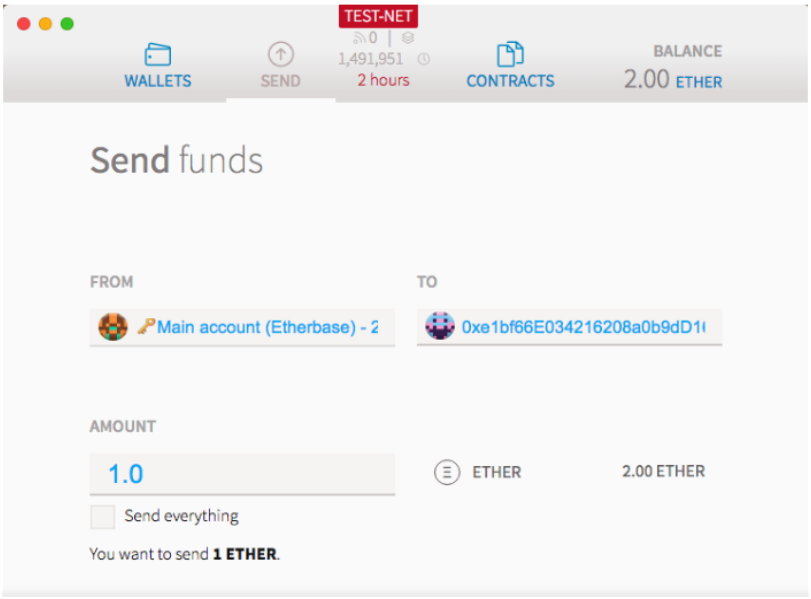
ซึ่งในการสั่งให้ Ethereum ประมวลผลใดๆ ต้องมีการใช้ gas จึงต้องมีการ miner ที่ตัว console เพื่อให้มี gas เพียงพอในการประมวลผลแต่ละครั้ง หน่วยของ gas ที่ใช้จะเป็น ether

ใน Mist/Ethereum จะมี User Interface (UI) ช่วยในการโปรแกรมได้ดังภาพ



เมื่อได้เข้าสู่ Ethereum wallet แล้วสามารถสร้างบัญชีและรหัสผ่านของผู้ใช้ ซึ่งจะเป็น Main Account และจะได้ Account Address ซึ่งเป็นเลขฐาน 16 เพื่อเป็นการอ้างอิงในการส่งข้อความ และ ether ระหว่าง Account ได้

จะต้องมี ether จึงจะประมวลผลได้ จึงต้องมีการส่ง ether ระหว่าง account ดังภาพ



ซึ่งในการส่ง ether เข้าสู่ Account หรือ Wallet ถือว่าเป็น transaction และจะมีการรวม transaction เป็น block และจะถูกเข้ารหัสด้วย hash function ใน blockchain

ตัวอย่างคำสั่งในการส่ง ether ภายในเครือข่าย ผ่าน console ของ VM

```
$ ./ge1.sh --mine --minerthreads=1 console
```

หรือจะใช้คำสั่ง java script เพื่อส่ง ether ไปที่ main account ได้

```
balance = web3.fromWei(eth.getBalance(eth.accounts[0]), "ether")
```

ซึ่งเป็นการส่ง ether หรือ miner เพื่อให้ได้จำนวน ether เก็บไว้ในการประมวลผลต่างๆ

เมื่อต้องการหยุดการ miner ใช้คำสั่ง

```
$ miner.stop()
```

คำสั่งในการส่ง ether ไปยัง Account จากหน้า console

-From geth console

-sender = eth.accounts[0];

-receiver = eth.accounts[1];

-amount = web3.toWei(0.01, "ether")

-eth.sendTransaction({from:sender, to:receiver, value: amount})

ในการโปรแกรมเพื่อส่ง ether ระหว่าง Account ตรวจสอบ transaction ค่า Balance

ตัวอย่างการเขียนโปรแกรมเพื่อส่งข้อด้วยภาษา java script ที่สมบูรณ์ขึ้น

```
contract MessageStorage {
        string storedData;
                event Set(
                 address indexed _from,
                string s
                );
         function set(string s) {
                storedData = s;
                 Set(msg.sender, s);
        }
         function get() constant returns (string retVal) {
        return storedData;
         }
}
```

ตัวอย่างคำสั่งในการ check Balances

```
function checkAllBalances() {
  var i =0;
  eth.accounts.forEach( function(e){
+   console.log("  eth.accounts["+i+"]: " +  e + " \tbalance: " + web3.fromWei(eth.getBalance(e), "ether")
" ether");
       i++;
  })
}
checkAllBalances()
```

ตัวอย่างโปรแกรมภาษา Solidity ในการสร้าง contact

```
pragma solidity ^0.4.0;
pragma solidity ^0.4.10;
  contract MessageStorage {
    string storedData;
```

ตัวอย่างโปรแกรมภาษา Solidity ในการ tweet ข้อความ

```
pragma solidity ^0.4.10;
 // "class" TweetStorage
 contract TweetStorage {

 @@ -13,7 +15,7 @@ contract TweetStorage {
   event Tweet(
     address indexed _from,
     string s,   int _tweetCount
     uint256 _tweetCount
   );
    // constructor
```

ตัวอย่างโปรแกรมภาษา Solidity เพื่อส่งข้อความ tweet

```
// "class" TweetStorage
 contract TweetStorage {
// "array" of all tweets of this account: maps the tweet id to the actual tweetmapping (uint => string)
_tweets;
// total number of tweets in the above _tweets mapping
            uint _tweetCount;
// "owner" of this account
            address _adminAddress;
// "array" of all tweets of this account: maps the tweet id to the actual tweet
              mapping (uint => string) _tweets;
  // total number of tweets in the above _tweets mapping
              uint _tweetCount;
// "owner" of this account
  address _adminAddress;
            event Tweet(
             event Tweet(
              address indexed _from,
               string s,
              int _tweetCount  );
```

```
// constructor
        function TweetStorage() {
                _tweetCount = 0;
                _adminAddress = msg.sender;
        }
// returns true if caller of function ("sender") is admin
        function isAdmin() constant returns (bool isAdmin) {
                return msg.sender == _adminAddress;
        }

        // create new tweet
        function tweet(string tweetString) returns (int result) {
                if (bytes(tweetString).length > 256) {
                        // limit to 256 bytes
                        result = -3;
                } else {
                        _tweets[_tweetCount] = tweetString;
                        _tweetCount++;
                        result = 0; // success
                }
 // constructor
         function TweetStorage() {
          _tweetCount = 0;
          _adminAddress = msg.sender;
                  }
 // returns true if caller of function ("sender") is admin
         function isAdmin() constant returns (bool isAdmin) {
         return msg.sender == _adminAddress;
                  }
 // create new tweet
          function tweet(string tweetString) returns (int result) {
```

```
        if (bytes(tweetString).length > 256) {
// limit to 256 bytes
        result = -3;
        } else {
        _tweets[_tweetCount] = tweetString;
        _tweetCount++;
        result = 0; // success
                }
        Tweet(msg.sender, tweetString, _tweetCount)
        }
        function getTweet(uint tweetId) constant returns (string tweetString) {
                tweetString = _tweets[tweetId];
        }

        function getLatestTweet() constant returns (string tweetString, uint numberOfTweets) {
                tweetString = _tweets[_tweetCount - 1];
                numberOfTweets = _tweetCount;
        }
        function getOwnerAddress() constant returns (address adminAddress) {
                return _adminAddress;
        }
        function getNumberOfTweets() constant returns (uint numberOfTweets) {
                return _tweetCount;
                }
         }
        function getTweet(uint tweetId) constant returns (string tweetString) {
        tweetString = _tweets[tweetId];
        }
        function getLatestTweet() constant returns (string tweetString, uint numberOfTweets) {
        tweetString = _tweets[_tweetCount - 1];
        numberOfTweets = _tweetCount;
        }
```

```
function getOwnerAddress() constant returns (address adminAddress) {
 return _adminAddress;
 }
 function getNumberOfTweets() constant returns (uint numberOfTweets) {
return _tweetCount;
}
function adminDeleteAccount() {
        if (isAdmin()) {
function adminDeleteAccount() {
  if (isAdmin()) {
// Deletes the contract and returns all funds to the owner's address
suicide(_adminAddress);
        }
}
 suicide(_adminAddress);
 }
}
 }
```

นอกจากนี้ยังมีการเขียนโปรแกรมในการส่งข้อความ และให้ปรากฏชื่อผู้ส่ง การนับจำนวนข้อความ รายละเอียดโปรแกรมสามารถสืบค้นได้ที่ https://github.com/skanakakorn/eth-intro

4. ประโยชน์ที่ได้รับ

1) นำความรู้ไปใช้ในการทำวิจัย เรื่อง การประยุกต์ใช้เทคโนโลยีบล็อกเชนในการพัฒนาระบบการตรวจสอบวุฒิการศึกษา

2) นำความรู้ไปปรับปรุงเนื้อหาเอกสารการสอนทางด้านเทคโนโลยีสารสนเทศและการสื่อสารให้ทันสมัย

3) นำความรู้ไปแนะนำเทคโนโลยีใหม่ในการสัมมนาเข้ม สัมมนาเสริม และหัวข้อในการทำวิทยานิพนธ์ การศึกษาค้นคว้าอิสระทางด้านเทคโนโลยีสารสนเทศและการสื่อสาร

คำชี้แจงการใช้เอกสาร

ขอขอบคุณที่ท่านให้ความสนใจศึกษาเอกสารเผยแพร่ความรู้ (KM) ของสาขาวิชาวิทยาศาสตร์และเทคโนโลยี มสธ. ซึ่งจัดทำขึ้นเพื่อเผยแพร่ให้เกิดประโยชน์เชิงวิชาการในวงกว้าง ทั้งนี้หากท่านนำข้อมูลจากเอกสารนี้ไปใช้ ขอให้อ้างอิงแหล่งที่มาจากเราด้วย พร้อมทั้งแจ้งให้เราทราบแหล่งที่ท่านนำไปอ้างอิง โดยแจ้งทางอีเมล์มาที่ stoffice@stou.ac.th เพื่อประโยชน์ในการบูรณาการข้อมูลร่วมกัน