รายงานสหกิจศึกษาฉบับสมบูรณ์

# เครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกล

# Modbus-Based Data Logger Using ASUS Tinker Board for Power Telemetering

นางสาวจันทรัช ภักดีวงศ์

หลักสูตรวิศวกรรมอัตโนมัติ
ภาควิชาวิศวกรรมการวัดและควบคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

รายงานสหกิจศึกษาฉบับสมบูรณ์

# เครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกล

# Modbus-Based Data Logger Using ASUS Tinker Board for Power Telemetering


นางสาวจันทรัช ภักดีวงศ์

หลักสูตรวิศวกรรมอัตโนมัติ

ภาควิชาวิศวกรรมการวัดและควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง


ปีการศึกษา 2561

**ชื่อโครงงานสหกิจศึกษา** เครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกล

**ชื่อ-สกุล นักศึกษา** นางสาวจันทรัช ภักดีวงศ์ รหัสนักศึกษา 58010166

**ภาควิชา** วิศวกรรมอัตโนมัติ

**คณะ** วิศวกรรมศาสตร์

**ชื่อ-สกุล อาจารย์นิเทศ** รศ.ดร.ไสว พงศ์สวัสดิ์

ผศ.ดร.ธีรวัฒน์ เทพมณี

**ชื่อ-สกุล ผู้นิเทศงาน** นายสิริชัย วงศ์ยุทธนาพงศ์

**สถานประกอบการ** บริษัท สเกด้า ออโตเมชั่น จำกัด

## บทคัดย่อ

โครงงานสหกิจนี้นำเสนอเทคนิคการสร้างเครื่องบันทึกข้อมูลโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกลที่ใช้เพาเวอร์มิเตอร์รุ่น EASTRON X96-3 ในการวัดค่าพารามิเตอร์ทางไฟฟ้า 1 เฟส และ 3 เฟส โดยเครื่องบันทึกข้อมูลนี้มีการส่งคำสั่งร้องขอค่าพารามิเตอร์ที่วัดได้จากเพาเวอร์มิเตอร์ด้วยโปรโตคอลมอดบัส-อาร์ทียูทุก ๆ 5 วินาที เพื่อบันทึกลงในฐานข้อมูลที่สร้างขึ้นสำหรับการแสดงผลด้วยซอฟต์แวร์ LabVIEW และการแสดงผลบนเว็บ จากผลการทดลองเพื่อทดสอบฟังก์ชันการอ่านค่าพารามิเตอร์ทางไฟฟ้า ฟังก์ชันการบันทึกข้อมูล และฟังก์ชันการแชร์ข้อมูลสามารถยืนยันได้ว่าเครื่องบันทึกข้อมูลที่นำเสนอสามารถทำงานได้อย่างถูกต้องได้

**คำสำคัญ:** โปรโตคอลมอดบัส อาร์ทียู, บอร์ด ASUS Tinker, เครื่องบันทึกข้อมูล, เพาเวอร์มิเตอร์, การวัดระยะไกล

| | |
|---|---|
| **Cooperative Project Title:** | Modbus-Based Data Logger Using ASUS Tinker Board for Power Telemetering |
| **Student:** | Ms. Chantharat Phakdeewong Student ID 58010166 |
| **Program:** | Automation Engineering |
| **Faculty:** | Engineering |
| **Advisors:** | Assoc.Prof.Dr. Sawai Pongsawat |
| | Asst.Prof.Dr. Teerawat Thepmanee |
| **Mentor:** | Mr. Sirichai Wongyutthanapong |
| **Company:** | Scada Automation Company Limited |


## ABSTRACT

This cooperative education project presents a technique to implement data logger by using ASUS Tinker board for power telemetering, that utilizes a power meter modeled EASTRON SMART X96-3 to measure the 1-phase and 3-phases electrical parameters. The implemented data logger sends the Modbus-RTU-based command every 5 seconds to read the measured values from the power meter to store in the created database for sharing the recorded data to further display on the LabVIEW-based and Web-based monitoring. The experimental results from testing the functions to read the electrical parameters, to write the received data, and to share the recorded data confirm that the proposed data logger can operate correctly.



**Keyword:** Modbus RTU Protocol, Tinker board ASUS, Data Logger, Power Meter, Telemetering

# กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี ผู้จัดทำขอขอบคุณบุคลากร บริษัท สเกด้า ออโตเมชั่น จำกัด ที่ได้ให้ความรู้ ประสบการณ์ในการทางาน คำปรึกษาและคำชี้แนะแนวทางในการแก้ไขปัญหาต่าง ๆ และอุปกรณ์ต่าง ๆ ตลอดระยะเวลาการศึกษาจัดทาปริญญานิพนธ์

ขอขอบคุณ คุณรณน สถิตปัญญาพันธ์ ประธานบริษัท ที่คอยให้คำปรึกษา และคำแนะนำต่าง ๆ เกี่ยวกับงานของบริษัท

ขอขอบคุณ คุณวัชระ ป้องกัน และ คุณสิริชัย วงศ์ยุทธนาพงศ์ พนักงานที่ปรึกษา ที่คอยกำกับดูแลให้คำปรึกษาและชี้แนะแนวทางการแก้ไขปัญหา รวมทั้งควบคุมงานให้สำเร็จลุล่วงได้ด้วยดี

ขอขอบคุณ ผศ.ดร. ธีรวัฒน์ เทพมณี และ รศ.ดร. ไสว พงศ์สวัสดิ์ ที่ให้คำแนะนำและกำกับดูแลนักศึกษาชั้นปีที่ 4 ในโครงการ สหกิจศึกษาของสาขาวิศวกรรมอัตโนมัติ

ขอขอบคุณ รศ.ดร. อัมพวัน จุลเสรีวงศ์ และคณาจารย์ทุกท่าน ที่ให้คำแนะนำและปรับแก้ไขปริญญานิพนธ์ฉบับนี้ให้สำเร็จลุล่วงได้ด้วยดี

ขอขอบคุณเอกสารอ้างอิงต่าง ๆ ที่คณะผู้จัดทำได้นำมาใช้อ้างอิงเพื่อทำปริญญานิพนธ์ฉบับนี้ด้วย

นางสาวจันทรัช ภักดีวงศ์

# สารบัญ

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญ (ต่อ)

หน้า

# สารบัญตาราง

**ตารางที่** **หน้า**

# สารบัญภาพ

**ภาพที่** **หน้า**

# สารบัญภาพ (ต่อ)

**ภาพที่** **หน้า**

# บทที่ 1
# บทนำ

## 1.1 ที่มาและความสำคัญ

ในปัจจุบันถ้าจะพูดถึง Internet of Things (IoT) คือ "อินเตอร์เน็ตในทุกสิ่ง" หมายถึง การที่อุปกรณ์ต่าง ๆ สิ่งต่าง ๆ ได้ถูกเชื่อมโยงทุกสิ่งทุกอย่างสู่โลกอินเตอร์เน็ต ทำให้มนุษย์สามารถสั่งการควบคุมการใช้งานอุปกรณ์ต่าง ๆ ผ่านทางเครือข่ายอินเตอร์เน็ต เช่น การเปิด-ปิด อุปกรณ์เครื่องใช้ไฟฟ้า (การสั่งการเปิดไฟฟ้าภายในบ้านด้วยการเชื่อมต่ออุปกรณ์ควบคุม เช่น มือถือ ผ่านทางอินเตอร์เน็ต) รถยนต์ โทรศัพท์มือถือ เครื่องมือสื่อสาร เครื่องมือทางการเกษตร อาคาร บ้านเรือน เครื่องใช้ในชีวิตประจำวันต่าง ๆ ผ่านเครือข่ายอินเตอร์เน็ต เป็นต้น

ในยุค Thailand 4.0 เทคโนโลยีถูกนำมาพัฒนาต่อยอดเพื่อลดบทบาทของมนุษย์ และเพิ่มศักยภาพของมนุษย์ในการใช้ความคิดเพื่อข้ามขีดจำกัด สร้างสรรค์พัฒนาสิ่งใหม่ ๆ โดยจะใช้ชื่อยุคนี้ว่าเป็นยุค Machine-to-Machine เช่น การเปิด-ปิด หรือสั่งงานอื่น ๆ กับเครื่องใช้ไฟฟ้าในบ้านตัวเองผ่านแอพพลิเคชันโดยไม่ต้องเดินไปกดสวิตช์

ในส่วนของบริษัทสเกด้า ออโตเมชันได้เห็นถึงความสำคัญของเทคโนโลยีในยุค Thailand 4.0 นี้ จึงได้มีโครงการมากมายที่พัฒนาให้ก้าวทันสมัย ที่มีวัตถุประสงค์หลักในการดำเนินการคือ ประกอบกิจการงานระบบควบคุม งานออโตเมชัน งานโปรแกรมเพื่อใช้ในระบบควบคุม ซึ่งบริษัทฯ มีความชำนาญทั้งงานติดตั้ง, งานระบบ รวมถึงงานไฟฟ้า เป็นต้น

ดังนั้นทางบริษัทสเกด้า ออโตเมชันจึงได้จัดทำเครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกลโดยจะเก็บค่าพลังงานจากเพาเวอร์มิเตอร์ (Power Meter) ซึ่งนำเทคโนโลยี IoT มาประยุกต์ใช้ในงานดังกล่าวนี้ โดยมีหลักการทำงานคือนำข้อมูลจากเพาเวอร์มิเตอร์มาเก็บลงใน Micro SD Card บนบอร์ด ASUS Tinker ซึ่งอ่านค่าผ่านโปรโตคอลมอดบัส (Modbus Protocol) และจะนำไปเก็บลงฐานข้อมูล (Database) แล้วนำไปใช้สำหรับการวัดพลังงานระยะไกลที่จะแสดงผลผ่านทางซอฟต์แวร์ LabVIEW และหน้าเว็บ เพื่อทราบสถานะได้ตลอดทุกช่วงเวลาและสามารถแสดงค่าแบบเรียลไทม์ (Real-Time) และแบบย้อนหลัง (Historian) ได้โดยผ่านทางสัญญาณ Wi-Fi

## 1.2 วัตถุประสงค์ของโครงงาน

สร้างเครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกล

**1.3 ขอบเขตของโครงงาน**

1. รุ่นเพาเวอร์มิเตอร์ที่นำมาศึกษาคือ EASTRON SMART X96-3

2. เครื่องบันทึกข้อมูลสร้างขึ้นเพื่อเก็บค่าพารามิเตอร์กลุ่มพื้นฐานระบบไฟฟ้า 3 เฟส เช่น ค่าแรงดันไฟฟ้า (Voltage) ค่ากระแสไฟฟ้า (Current) กลุ่มพลังงานระบบไฟฟ้า 3 เฟส เช่น ค่าตัวประกอบกำลัง (Power factor) และกลุ่มพลังงานรวมทั้ง 3 เฟส และเก็บค่าเวลาที่อ่านได้จากเพาเวอร์มิเตอร์ (Date Time)

3. มีระยะเวลาจัดเก็บข้อมูลทุก ๆ 5 วินาที และสามารถจัดเก็บข้อมูลไว้ได้นานที่สุดเป็นระยะเวลา 1 ปี เพื่อการร้องขอข้อมูลจากการแสดงผลลัพธ์บนซอฟต์แวร์ LabVIEW และการแสดงผลลัพธ์บนหน้าเว็บ

**1.4 แผนการดำเนินงาน**

เครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกลโดยจะเก็บค่าพลังงานจากเพาเวอร์มิเตอร์มีขั้นตอนการศึกษาดังนี้

1. ศึกษาการใช้งานของบอร์ด ASUS Tinker และเพาเวอร์มิเตอร์รุ่น EASTRON SMART X96-3
2. ศึกษาแนวคิดการวัดพลังงานระยะไกล
3. ศึกษาการใช้งานโปรแกรมที่เกี่ยวข้องได้แก่ Visual Studio Code และ SQLite เพื่อใช้ในการเขียนโปรแกรมเพื่ออ่านค่าโปรโตคอลมอดบัสพร้อมกับแชร์ข้อมูลบนเว็บเซอร์วิส (Web Service) สำหรับการร้องขอข้อมูลจากการแสดงผลลัพธ์บนซอฟต์แวร์ LabVIEW และเว็บ และทำฐานข้อมูลตามลำดับ

จากขั้นตอนการศึกษาข้างต้นสามารถแสดงได้ดังตารางต่อไปนี้

**ตารางที่ 1.1** หัวข้องาน และระยะเวลาดำเนินงาน

| หัวข้องาน | ส.ค. 2561 | | | | ก.ย. 2561 | | | | ต.ค. 2561 | | | | พ.ย. 2561 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.ศึกษาการใช้งานบอร์ด ASUS Tinker และเพาเวอร์มิเตอร์ | | | | | | | | | | | | | | | | |
| 2.ศึกษาการเขียนโปรแกรมโดยใช้ภาษา Python เพื่ออ่านค่าโปรโตคอลมอดบัส-อาร์ทียู | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3.สร้างฐานข้อมูลโดยใช้โปรแกรม SQLite และแชร์ข้อมูลบนเว็บเซอร์วิส | | | | | | | | | | | | | | | | |
| 4.Integrated System และตรวจสอบระบบ | | | | | | | | | | | | | | | | |
| 5.จัดทำและแก้ไขรูปเล่ม | | | | | | | | | | | | | | | | |

**1.5 ประโยชน์ที่คาดว่าจะได้รับ**

1. สามารถนำค่าพารามิเตอร์ที่อ่านได้นั้นมาเก็บลงฐานข้อมูล และสามารถนำค่าที่อ่านได้นั้นแชร์ข้อมูลบนเว็บเซอร์วิสได้

2. เครื่องบันทึกข้อมูลในส่วนของฐานข้อมูลสามารถดูค่าพารามิเตอร์พลังงานย้อนหลังแล้วนำผลลัพธ์มาวิเคราะห์วางแผนการใช้พลังงานได้

3. ผู้ใช้งานไม่ต้องเสียเวลาในการเดินทางไปยังหน้างาน โดยระบบจะออกแบบมาให้สะดวกกับการเข้าถึงข้อมูลผ่านเครือข่ายอินเตอร์เน็ตได้

4. ทางบริษัทสามารถนำเครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกลไปนำเสนอกับกลุ่มลูกค้าที่มีความสนใจในเรื่องดังกล่าวได้

# บทที่ 2
# แนวคิดและหลักการที่เกี่ยวข้อง

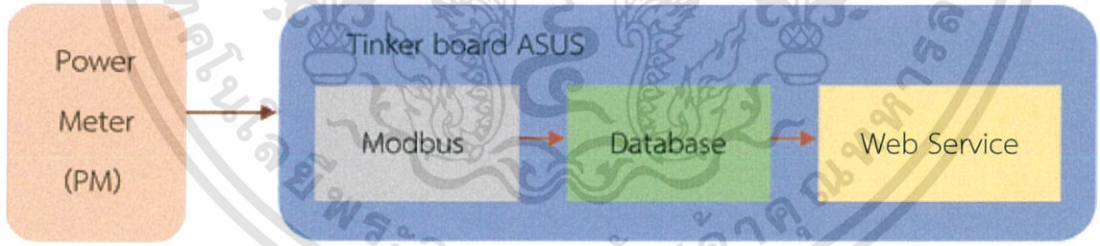
## 2.1 กล่าวนำ

ในบทที่ 2 จะกล่าวถึงแนวคิดการวัดพลังงานระยะไกล อุปกรณ์ที่ใช้ในการวัดค่าพลังงาน บอร์ด ASUS Tinker และซอฟท์แวร์ที่ใช้ในการทำเครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกล

ขั้นตอนของการทำงานในการทำเครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกล ได้แบ่งเป็น 3 ส่วนในการทำงานหลัก ๆ คือ

1. การอ่านค่าโปรโตคอลมอดบัสจากเพาเวอร์มิเตอร์ แล้วบันทึกค่าพารามิเตอร์ลงในฐานข้อมูลและสามารถแชร์ผ่านเว็บเซอร์วิสสำหรับการร้องขอข้อมูลบนซอฟต์แวร์ LabVIEW และเว็บ
2. การแสดงค่าพารามิเตอร์ผ่านโปรแกรมซอฟต์แวร์ LabVIEW
3. การแสดงค่าพารามิเตอร์ผ่านหน้าเว็บ

ซึ่งในรายงานฉบับนี้จะพูดถึงขั้นตอนส่วนแรกก็คือ การอ่านค่าโปรโตคอลมอดบัสจากเพาเวอร์มิเตอร์ แล้วเก็บค่าพารามิเตอร์ลงในฐานข้อมูล ซึ่งในส่วนนี้จะแบ่งได้อีก 3 ส่วน ได้แก่ ส่วนที่หนึ่ง คือ ส่วนที่อ่านค่าโปรโตคอลมอดบัสจากเพาเวอร์มิเตอร์ ส่วนที่สอง คือ เก็บค่าลงฐานข้อมูล และส่วนที่สาม คือ การส่งผ่านข้อมูลผ่านเว็บเซอร์วิสซึ่งผ่าน HTTP Protocol โดยใช้สัญญาณ Wi-Fi



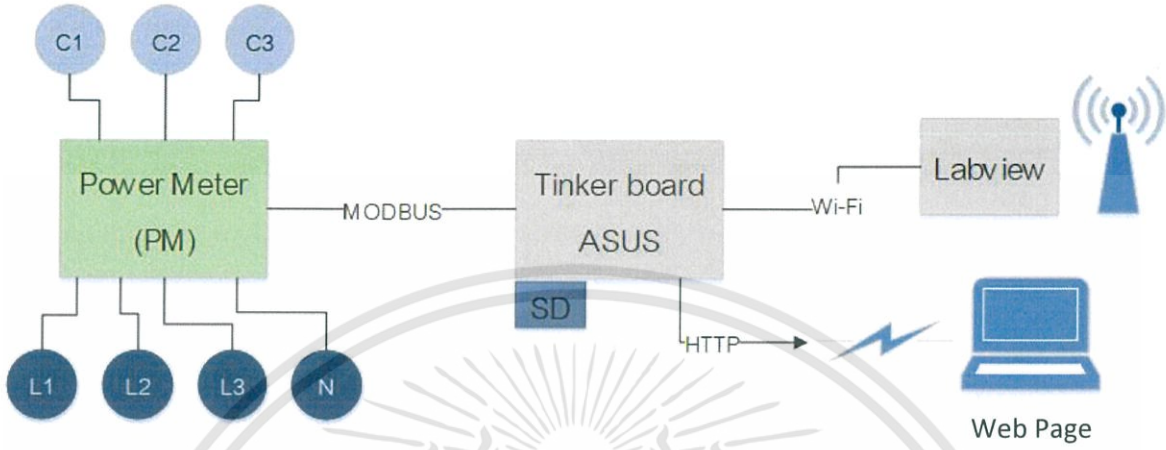

**ภาพที่ 2.1** System Diagram ของโปรแกรมในบอร์ด ASUS Tinker เพื่ออ่านพารามิเตอร์จากเพาเวอร์มิเตอร์

## 2.2 แนวคิดการวัดพลังงานระยะไกล

แนวความคิดการวัดพลังงานระยะไกลสามารถส่งผ่านข้อมูลได้ทั้งแบบใช้สายและแบบไม่ใช้สาย แนวคิดการสร้างระบบซึ่งทำหน้าที่วัดค่าการใช้พลังงานไฟฟ้าด้วยอุปกรณ์อิเล็กทรอนิกส์แทนมิเตอร์วัดแบบเดิมได้มีขึ้นมาก่อนหน้าแล้วและได้มีพัฒนาการมาเป็นลำดับ

สำหรับโครงงานเรื่องนี้จะนำเสนอในรูปแบบการนำข้อมูลที่อ่านได้จากเพาเวอร์มิเตอร์มาเก็บลงเครื่องบันทึกข้อมูล (Data Logger) แล้วนำเข้าฐานข้อมูลเพื่อที่จะส่งผ่านข้อมูลโดยใช้เว็บเซอร์วิสเพื่อการ

ร้องขอข้อมูลของ LabVIEW และเว็บที่สามารถดูผลลัพธ์ได้ทั้งแบบค่าปัจจุบันและค่าย้อนหลังได้เพื่อประโยชน์ในส่วนของผู้ใช้งานที่ไม่ต้องเสียเวลาในการเดินทางไปยังหน้างาน หรือสามารถวางแผนการใช้พลังงานได้ เป็นต้น




**ภาพที่ 2.2** ภาพรวม System Diagram ทั้งระบบ

## 2.3 อุปกรณ์ที่ใช้ในการวัดค่าพลังงาน

2.3.1 เพาเวอร์มิเตอร์รุ่น EASTRON SMART X96-3


**ภาพที่ 2.3** เพาเวอร์มิเตอร์รุ่น EASTRON SMART X96-3

อุปกรณ์แสดง " ค่าพารามิเตอร์และปริมาณพลังงานไฟฟ้า " เช่น แรงดัน กระแส กำลังงานไฟฟ้าจริง กำลังงานไฟฟ้ารีแอคทีฟ และ Harmonic เป็นต้น เพื่อให้ทราบถึงค่าทางไฟฟ้าในกระบวนการผลิตและการใช้พลังงานไฟฟ้าได้ โดยส่วนใหญ่แล้วในภาคอุตสาหกรรม จะนำเพาเวอร์มิเตอร์ไปใช้ในการควบคุมหรือปรับปรุงการใช้พลังงานไฟฟ้า เพื่อให้เกิดประสิทธิภาพในการทำงานได้อย่างเต็มที่ อีกทั้งยังเป็นการช่วยจัดการพลังงาน ซึ่งเป็นไปตามมาตรฐานของ ISO 50001 โดยเพาเวอร์มิเตอร์นั้นสามารถแบ่งออก

ได้เป็น 2 แบบ คือ แบบเข็ม (Analog Power Meter) และแบบหน้าจอดิจิตอล (Digital Power Meter) ซึ่งในที่นี้ได้ใช้แบบหน้าจอดิจิตอล โดยจะมีคุณสมบัติดังตารางต่อไปนี้

**ตารางที่ 2.1** คุณสมบัติทางเทคนิคเพาเวอร์มิเตอร์รุ่น EASTRON SDM630-MT

| Specification | |
|---|---|
| Nominal voltage(Un) | 3x230/400 V ac |
| Operational voltage | 60% ~ 120% of Un |
| Insulation capabilities | |
| - AC voltage withstand | 4KV for 1 minute |
| - Impulse voltage withstand | 6KV-1.2μS |
| Rated current (Ib) | 100mA or 333mV CT input |
| Operational current range | 0.4%Ib-Imax |
| Over current withstand | 20 Imax for 0.01s |
| Operational frequency range | 50 or 60Hz |
| Power consumption per phase | ≤ 2W/10VA |
| Pulse output 1 | Configurable |
| Pulse output 2 | 3200 imp/kWh |
| Display | LCD |
| Max reading | 9999999.9 kWh/kVarh |

*หมายเหตุ: เนื่องจากคุณสมบัติทางเทคนิคของเพาเวอร์มิเตอร์รุ่น EASTRON SMART X96-3 ไม่ปรากฏ จึงใช้คุณสมบัติทางเทคนิคของเพาเวอร์มิเตอร์รุ่น EASTRON SDM630-MT แทนเนื่องจากเป็นรุ่นที่ใช้อ้างอิงในการอ่านโปรโตคอลมอดบัสของพารามิเตอร์แต่ละค่าในเพาเวอร์มิเตอร์

2.3.2 หม้อแปลงกระแสรุ่น EASTRON ESCT-TU36


**ภาพที่ 2.4** EASTRON ESCT-TU36

หม้อแปลงกระแส (Current Transformer) หรือ CT เป็นอุปกรณ์ไฟฟ้าที่ใช้ประกอบการวัดกระแสไฟฟ้าโดยต่อร่วมกับเครื่องวัดกระแส หรือเพาเวอร์มิเตอร์ โดยทำหน้าที่แปลงกระแสไฟฟ้า หรือลดทอนกระแสไฟฟ้า (Step down) ที่จะวัดนั้นให้เหมาะสมกับพิกัดกระแสไฟฟ้าที่ขดลวดกระแสของเครื่องมือวัดรับได้ เช่น อัตราส่วน15/5A, 50/5A, 150/5A, 500/5A หรือรวมไปจนถึงกระแสที่สูง ๆ เช่น 10,000/5A, 15,000/5A เป็นต้น

1. ประเภทของ Current Transformer หรือ CT แบ่งตามการใช้งานได้เป็น 2 รูปแบบ ดังนี้

1) ตัวแปลงกระแสแบบถอดห่วงไม่ได้ (Current Transformer: CT)

2) ตัวแปลงกระแสแบบถอดห่วงได้ (Split Core Current Transformer: CT)

ใส่วนของ Split Core Current Transformer: CT ก็จะมีหลากหลายขนาดมีทั้งเล็กและใหญ่

- CT แบบ Split Core ขนาดใหญ่ เอาต์พุตที่ออกมาจะเป็นกระแส AC เช่น 5A หรือ 1A
- CT แบบ Split Core ขนาดเล็ก เอาต์พุตที่ออกมาจะเป็นแรงดัน 333mV

2. ค่า Burden

ค่า Burden หมายถึง กำลังไฟฟ้าที่สูญเสีย (VA) ของขดลวดแปลงกระแสไฟฟ้า หลักการ คือ เมื่อมีอุปกรณ์ไฟฟ้าทุกตัวที่ต่ออนุกรมกับขดลวดด้าน secondary มีกำลังไฟฟ้าสูญเสียในเครื่องวัดรวมกันกับกำลังไฟฟ้าสูญเสียในสายวัดมีค่าภายใต้เบอร์เดนของขดลวดแปลงกระแสแล้ว ความผิดพลาดจากการวัดจะเป็นไปตามคลาส (Class) ของขดลวดแปลงกระแสไฟฟ้านั้นเช่น CT 100/5 Class 0.5 จะมีค่าของ Burden 1.5VA , Class3 จะมีค่าของ Burden 2.25VA ตามตารางด้านล่าง ถามว่าแล้วจะเลือกใช้งานแบบไหนดี ระหว่าง Class 0.5 กับ Class3 ให้เลือก Class ที่มีตัวเลขต่ำไว้ก่อนครับ คือ 0.5 เพราะจะบ่งบอกถึงความแม่นยำในส่วนอื่นตามมา

3. อัตราส่วน (Ratio) ของ CT

อัตราส่วน CT หมายถึง อัตราส่วนของการแปลงกระแส ทางด้านอินพุต primary ต่อ เอาต์พุต Secondary เช่น Ratio ของ CT 400/5A คือ 80ไม่มีหน่วย อธิบายได้ คือ“CT มีPrimary Current เท่ากับ 400A และ Secondary Current 5A มีอัตราส่วนตัวคูณคือ 80”

4. ข้อควรระวังเกี่ยวกับ CT

เมื่อมีการใช้งานกระแสด้าน อินพุต primaryอยู่จะมีกระแสทางด้านเอาต์พุต Secondary ออกมาด้วย ไม่ควรเปิดลูปวงจรด้านนี้ควร Short ลูป Secondary เอาไว้เพื่อหลีกเหลี่ยงการพังเสียหายของ CTได้ ถ้าระดับกระแสอินพุตมีสูงมากจะทำให้ร้อนและเกิดเพลิงไหม้ได้ แนะนำให้งดจ่ายกระแส อินพุต primary ทุกครั้งที่มีการแก้ไขปรับปรุง

5. คุณสมบัติทางเทคนิคของหม้อแปลงกระแสรุ่น EASTRON ESCT-TU36

**ตารางที่ 2.2** คุณสมบัติทางเทคนิคของหม้อแปลงกระแสรุ่น EASTRON ESCT-TU36

| | |
|---|---|
| Frequency | 50-60Hz |
| Rated current | 5A to 600A loads |
| Rated output | 0.333V / 100mV( AC) |
| Accuracy | ±1% from 20% to 120% of rated current |
| Phase angle | less than 2 degrees at 50% of rated current |
| Insulation voltage | 600Vac |
| Dielectric strength | 2.5KV/ 1mA/1min |
| Operating temperature | -15°C to 60°C |
| Operating humidity | < 85% |
| Case material | PC/UL94-V0 |
| Bobbin | PBT |
| Core | Perm alloy |
| Internal structure | Epoxy |
| Leads | UL 1015, Twisted Pair, 22AWG |

6. Application

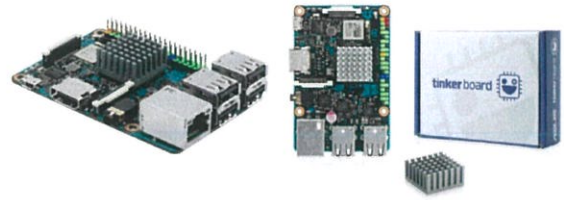
1) Current measurement, monitoring and protection for electrical wiring and equipment

2) Current and power measurement for electric motors, lighting, air compressor, heating and ventilation system, air-condition equipment and automation.

3) Current, power and energy monitoring device

4) Relay protection device

## 2.4 บอร์ด ASUS Tinker




**ภาพที่ 2.5** บอร์ด ASUS Tinker [1]

บอร์ด ASUS Tinker เป็น Single Board Computer (SBC) ในรูปแบบฟอร์มแฟคเตอร์แบบ Ultra-Small ในขณะที่ยังให้ความเข้ากันได้สูงสุด Tinker Board ยังให้เมกเกอร์ ผู้สนใจใน IoT ผู้ชื่นชอบงานอดิเรก ผู้ประกอบคอมพิวเตอร์ และผู้คนทั่วไปได้รับแพลทฟอร์มที่ไว้วางใจได้และความสามารถสูงเพื่อสร้างและประกอบไอเดียต่าง ๆ ให้เป็นความจริง

2.4.1 IoT Connectivity

Tinker Board มาพร้อมกับตัวเลือกการเชื่อมต่อสำหรับเมคเกอร์และผู้ชื่นชอบงานอดิเรก รวมถึงอินเตอร์เฟส 40-pin GPIO นอกจากนี้ยังมาพร้อมกับการเชื่อมต่อ HD MIPI 2 พอร์ต เพื่อการแสดงผล HD และกล้อง HD Tinker board ยังมีพอร์ต LAN Gbit เพื่อการโอนถ่ายข้อมูลที่เหนือกว่าเหมาะแก่การเป็นศูนย์กลางระบบเครือข่ายและเป็นอุปกรณ์จัดเก็บข้อมูลบนระบบเครือข่าย อีกทั้งพอร์ต LAN บน Tinker Board ยังได้รับทรัพยากรบัสพิเศษเพื่อให้สามารถโอนถ่ายข้อมูลที่มีประสิทธิภาพ ภาคควบคุม Wi-Fi และ Bluetooth ได้รับการป้องกันสัญญาณรบกวนด้วยแผ่นโลหะ เพื่อให้ได้รับประสิทธิภาพสูงสุด และยังมีช่องเชื่อมต่อสายอากาศ IPEX อีกด้วย

นอกจากการเชื่อมต่อข้างต้นแล้ว Tinker Board ยังมีพอร์ต HDMI แบบมาตรฐานเพื่อเชื่อมต่อกับ TV จอภาพ และการแสดงผลรูปแบบอื่น พร้อมพอร์ต USB 2.0 4 พอร์ตเพื่อการเชื่อมต่ออุปกรณ์เสริมที่ครบครัน

2.4.2 TinkerOS Supported OS Applications

Debian-based ช่วยเพิ่มการใช้งานที่ราบลื่นและรวดเร็ว นอกจากนั้นยังสามารถนำไปประยุกต์ใช้งานให้ตอบสนองกับการทำงาน ไม่ว่าจะเป็นการใช้งานในรูปแบบเอ็นเตอร์เทนเม้น หรือใช้งานด้านการเขียนโปรแกรมในการสั่งการหรือควบคุมเครื่องจักร TinkerOS ถือเป็นอีกหนึ่งทางเลือกที่เหมาะสำหรับการใช้งานในรูปแบบต่าง ๆ ที่ตอบสนองการทำงาน

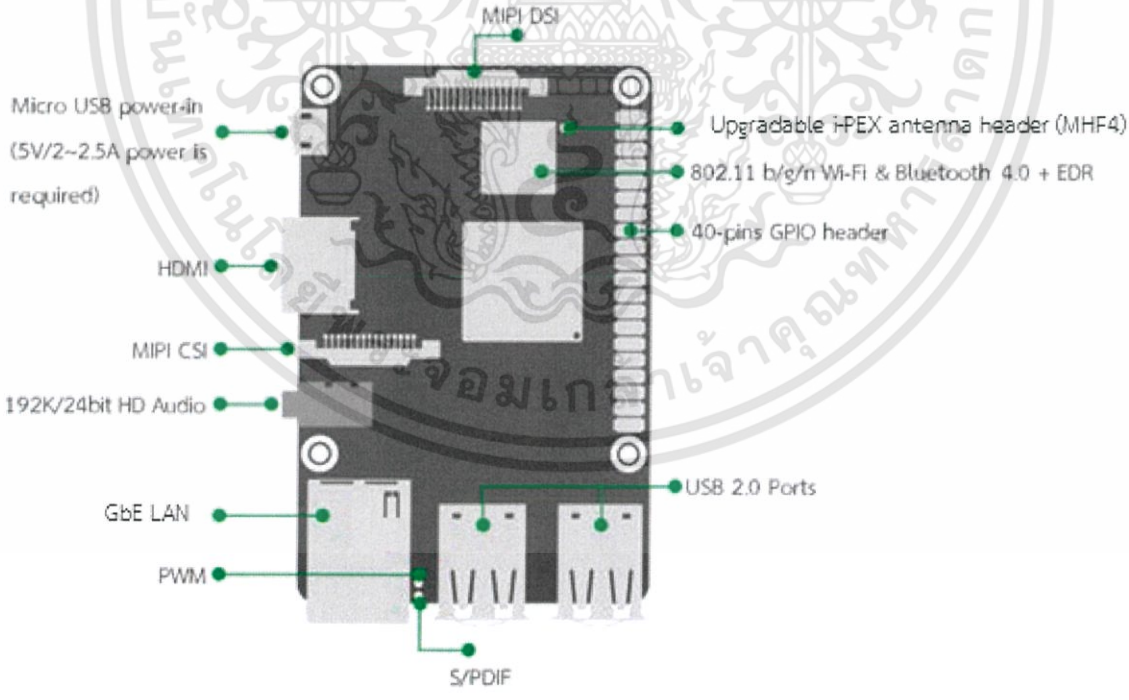
นอกจากนั้น TinkerOS ได้รับการออกแบบมาให้มีน้ำหนักเบาเพื่อเพิ่มความยืดหยุ่นในการใช้งาน โดยจะทำงานบน Debian 9 GUI เหมาะสำหรับบอร์ด SBC ที่รองรับการเชื่อมต่อจากอุปกรณ์จัดเก็บข้อมูลภายนอกแบบ Plug & Play NTFS เว็บเบราเซอร์ Chromium ที่ผ่านการปรับแต่งมาให้เหมาะสำหรับการใช้งานที่เหมาะสมจะช่วยเพิ่มประสิทธิภาพในการใช้งานที่รวดเร็วและเสถียรและ ได้พัฒนาให้เว็บเบราเซอร์ทำงานได้อย่างรวดเร็วโดยสามารถแสดงผลวิดีโอบนความละเอียดระดับ HD ผ่านทาง Youtube ได้อย่างราบลื่น

2.4.3 สิ่งที่จำเป็นสำหรับ Tinker Board

1. 1 x Micro SD Card ความจุอย่างน้อย 8GB
2. 1 x Micro สาย USB และอแดปเตอร์ 5V/2~2.5A ที่มี LPS
3. 1 x จอภาพ และสาย HDMI
4. 1 x คีย์บอร์ด และ เมาส์

*หมายเหตุ: ควรใช้ SD Card ความเร็วสูง (Class 10 หรือสูงกว่า) เพื่อเสถียรภาพระบบสูงสุด

2.4.4 Hardware




**ภาพที่ 2.6** Port เชื่อมต่อต่าง ๆ ของบอร์ด ASUS Tinker

1. Power Supply

Tinker Board ใช้ไฟฟ้า 5V/2~2.5A จากพอร์ต micro-USB กระแสไฟฟ้าที่ Tinker Board ใช้ (mA) จะขึ้นอยู่กับอุปกรณ์ที่เชื่อมต่อ สำหรับการใช้งานทั่วไป ภาคจ่ายกระแสไฟฟ้า 2A จากผู้ผลิตชั้นนำนั้นเพียงพอสำหรับการใช้งาน Tinker Board

โดยทั่วไป Tinker Board ใช้กระแสไฟฟ้า 700 - 1000mA ขึ้นอยู่กับอุปกรณ์เชื่อมต่อที่ใช้ มันอาจใช้กระแสเพียง 500mA เมื่อไม่ได้เชื่อมต่ออุปกรณ์เสริม กระแสไฟฟ้าสูงสุดสำหรับ Tinker Board คือ 1 A หากต้องการเชื่อมต่ออุปกรณ์ USB ที่ใช้กระแสไฟฟ้าสูงกว่า 0.5 A ต้องเชื่อมต่อผ่าน USB Hub ที่มีแหล่งพลังงานของตัวเอง

2. USB

Tinker Board มาพร้อมกับพอร์ต USB 2.0 4 พอร์ตซึ่งเชื่อมต่อกับ GL852G USB Hub จากพอร์ตอัพสตรีมทาง RK3288

พอร์ต USB ทำให้สามารถเชื่อมต่ออุปกรณ์เสริม เช่น คีบอร์ด เมาส์และเว็บแคม ให้คุณสามารถใช้งานได้อย่างหลากหลาย

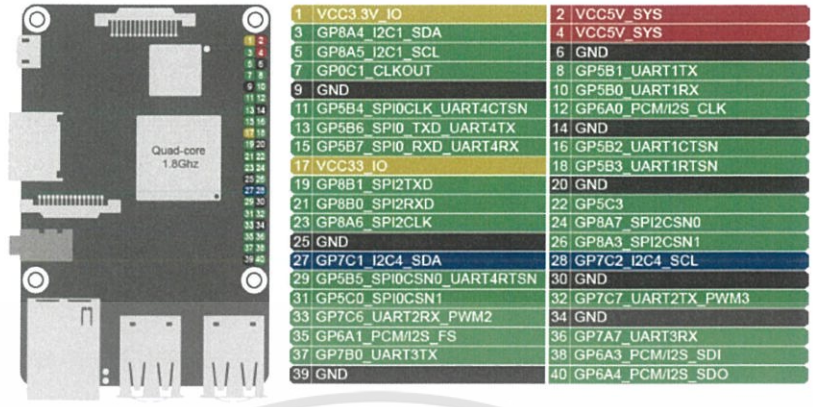
ฮาร์ดแวร์ USB บน Tinker Board มีความแตกต่างกับฮาร์ดแวร์ USB บนคอมพิวเตอร์เดสก์ท็อป โน้ตบุ๊กและแท็บเล็ต

โฮสต์พอร์ต USB ภายใน Tinker Board ใช้สำหรับกระแสไฟฟ้าเท่านั้น จุดประสงค์เดิมของ RK3288 คือ การใช้สำหรับตลาดอุปกรณ์พกพา เช่น พอร์ตUSB เดี่ยวบนโทรศัพท์สำหรับการเชื่อมต่อกับ PC หรือการเชื่อมต่อกับอุปกรณ์ชิ้นเดียว นั่นหมายถึงฮาร์ดแวร์ OTG นั้นเรียบง่ายกว่าฮาร์ดแวร์บน PC

โดยทั่วไป OTG รองรับการสื่อสารกับอุปกรณ์ USB ทุกชนิด แต่สำหรับการใช้งานที่เหมาะสมกับอุปกรณ์ USB ที่อาจใช้งานกับ Tinker Board ดังนั้นซอฟท์แวร์ระบบจะต้องทำงานมากขึ้น

Port Power Limit โดยทั่วไปอุปกรณ์ส่วนใหญ่จะรองรับระบบปฏิบัติการ Linux ซึ่งสามารถใช้ในการปรับแต่งบอร์ด ASUS Tinkerได้ (ข้อที่ถูกยกเว้นจะมีรายละเอียดตามด้านล่าง) Linux มีฐานข้อมูลไดร์เวอร์ที่รองรับการทำงานของอุปกรณ์รุ่นเก่าได้ TinkerOS เป็นระบบ Debian Kernel ซึ่งรองรับการเชื่อมต่อกับอุปกรณ์ภายนอกได้เป็นจำนวนมากหากมีอุปกรณ์อื่น ๆ และต้องการใช้งานร่วมกับบอร์ด ASUS Tinker สามารถทดลองการเชื่อมต่ออุปกรณ์เหล่านั้น เข้ากับ Tinker Board เพื่อทดสอบว่าอุปกรณ์เหล่านั้นสามารถทำงานร่วมกับ Tinker Board ได้หรือไม่ หากใช้อินเทอร์เฟซแบบกราฟิกที่รองรับงานเฉพาะทาง มีความเป็นไปได้ว่าอุปกรณ์เหล่านั้นจะปรากฏหน้าไอคอนหลังจากทำการติดตั้งไดร์เวอร์

3. GPIO




**ภาพที่ 2.7** พิน GPIO ของบอร์ด ASUS Tinker

พิน GPIO (general purpose input/output) ที่อยู่ด้านขอบของบอร์ดพิน เหล่านี้เป็นอินเตอร์เฟสระหว่าง Tinker Board และโลกภายนอก สามารถมองพินเหล่านั้นเป็นสวิตช์ที่สามารถเปิด-ปิดได้ จากทั้งหมด 40 พิน 28 พินเป็น GPIO (แชร์กับพิน SPI/UART/I2C) Tinker Board ได้รับการติดตั้ง SPI bus ที่สามารถเลือกได้ 2 ชิพ SPI Bus มีอยู่บนเฮดเดอร์ 40-พิน

2.4.5 คุณสมบัติทางเทคนิค

**ตารางที่ 2.3** คุณสมบัติทางเทคนิคของบอร์ด ASUS Tinker

| | |
|---|---|
| CPU | Rockchip Quad-Core RK3288 processor |
| หน่วยความจำ | 2GB Dual Channel DDR3 |
| กราฟฟิก | Integrated Graphics Processor<br>ARM® Mali™-T764 GPU*1 |
| สตอเรจ | Micro SD(TF) card slot |
| LAN | RTL GB LAN |
| Wireless Data Network | 802.11 b/g/n, Bluetooth V4.0 + EDR |
| Audio | RTL ALC4040 CODEC |
| พอร์ต ยูเอสบี | 4 x USB 2.0 |
| พอร์ต I/O ภายใน | 1 x 40-pin header :<br>- up to 28 x GPIO pins |

| | |
|---|---|
| | - up to 2 x SPI bus<br>- up to 2 x I2C bus<br>- up to 4 x UART<br>- up to 2 x PWM<br>- up to 1 x PCM/I2S<br>- 2 x 5V power pins<br>- 2 x 3.3V power pins<br>- 8 x ground pins<br>1 x 2-pin contact pin :<br>- 1 x PWM<br>- 1 x S/PDIF<br>1 x 15-pin MIPI DSI<br>1 x 15-pin MIPI CSI |
| อุปกรณ์เสริม | Passive heat sink*2<br>User manual |
| น้ำหนัก | 55g |
| หมายเหตุ | *1. The ARM® Quad-Core Mali GPU supports max. Resolution of 4K@30hz (up-scaled from 1080P) with H.264/H.265 hardware decoder (update coming soon).<br>*2. Do not touch the SoC or heat sink surface directly to prevent possible burn injury. |

2.4.6 Monitor

**ภาพที่ 2.8** Monitor

มอนิเตอร์ คือ จอภาพ ปัจจุบันมีให้เลือกใช้งานหลายขนาด หน่วยวัดหน้าจอมอนิเตอร์ของคอมพิวเตอร์ ไม่แตกต่างจากจอทีวีคือมีหน่วยเป็นนิ้ว (inch) และการวัดก็ใช้วิธีการวัดแนวทแยง จอมอนิเตอร์ที่ได้รับความนิยมในปัจจุบันได้แก่จอประเภท LCD หากต้องการนำจอมอนิเตอร์ไปใช้งานงานกราฟฟิกส์ เช่น การปรับแต่งรูปภาพด้วยโปรแกรมกราฟฟิกส์จะมีส่วนเสริมที่เรียกว่า IPS อีกด้วย

การเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์ กับจอมอนิเตอร์ จะเชื่อมต่อด้วยสายซึ่งมีหัวสำหรับเชื่อมต่ออยู่หลายชนิด เช่น RGB, DVI, HDMI เป็นต้น ปัจจุบันหัวต่อที่ได้รับความนิยมได้แก่หัวต่อแบบ HDMI สำหรับเครื่องคอมพิวเตอร์ที่ยังคงเป็นสเปคเดิม (รุ่นเก่า) จะเป็นหัวต่อแบบ RGB หรือ DVI

การเลือกซื้อจอมอนิเตอร์นั้น เราจะให้ความสำคัญกับประเภทของจอภาพ เช่น เป็นจอชนิด LED และมี IPS มีความละเอียดเท่าไหร่ หากต้องการให้จอมอนิเตอร์สามารถรองรับกับไฟล์วีดีโอที่มีความละเอียดสูงก็ต้องเป็นจอภาพแบบ Full HD ที่มีความละเอียด 1920 x 1080 ส่วนฟังก์ชั่นการทำงานอื่น ๆ ก็ต้องเลือกให้เหมาะสมกับตนเอง ตามอุปกรณ์ที่มีอยู่ หรือคาดการณ์ว่าจะมีในอนาคต

2.4.7 Wireless Mouse

**ภาพที่ 2.9** Wireless Mouse

เป็น Mouse ที่มีการทำงานเหมือน Mouse ทั่วไปเพียงแต่ไม่มีการใช้สายไฟต่อออกมาจากตัว Mouse ซึ่ง Mouse ชนิดนี้จะมีตัวรับและตัวส่งสัญญาณซึ่งทางด้านตัวรับสัญญาณอาจจะเป็นหัวต่อแบบ PS/2 หรือ แบบ USB ที่เรียกว่า Thumb USB receiver ซึ่งใช้ความถี่วิทยุอยู่ที่ 27MHz

2.4.8 สาย HDMI

**ภาพที่ 2.10** สาย HDMI

HDMI เป็นระบบการเชื่อมต่อ (สายสัญญาณและช่องต่อ) ภาพและเสียงแบบใหม่ครับ ย่อมาจากคำว่า High Definition Multimedia Interface โดย HDMI จะเชื่อมต่อทั้งสัญญาณภาพและเสียงระบบดิจิตอลไว้ในสายสัญญาณเพียงเส้นเดียว ไม่ต้องต่อหลายสายหลายเส้นให้ยุ่งยาก ให้ความคมชัดของภาพ มีความละเอียด และให้เสียงที่สมบูรณ์แบบที่สุดเท่าที่เคยมีมา ทุกวันนี้ HDMI ถูกนำมาใช้อย่างแพร่หลายกับ พลาสม่าทีวี แอลซีดีทีวี Home Theatre เครื่องเล่นดีวีดี ฯลฯ

## 2.5 ซอฟท์แวร์ที่เกี่ยวข้อง

2.5.1 Visual Studio Code หรือ VS Code


**ภาพที่ 2.11** โปรแกรมที่ใช้เขียนภาษา Python และภาษา JSON [14]

Visual Studio Code หรือ VSCode เป็นโปรแกรม Code Editor ที่ใช้ในการแก้ไขและปรับแต่งโค้ด จาก Microsoft มีการพัฒนาออกมาในรูปแบบของ Open Source จึงสามารถนำมาใช้งานได้แบบฟรี ๆ ตามที่ต้องการ

ซึ่ง Visual Studio Code นั้น เหมาะสำหรับนักพัฒนาโปรแกรมที่ต้องการใช้งานข้ามแพลตฟอร์ม รองรับการใช้งานทั้งบน Windows, macOS และ Linux สนับสนุนทั้งภาษา JavaScript, TypeScript และ Node.js สามารถเชื่อมต่อกับ Git ได้ นำมาใช้งานได้ง่ายไม่ซับซ้อน มีเครื่องมือส่วนขยายต่าง ๆ ให้เลือกใช้อย่างมากมาย

ในโครงงานนี้ได้นำมาเขียนโปรแกรมภาษา Python สื่อสารกับเพาเวอร์มิเตอร์โดยใช้

โปรโตคอลมอดบัส-อาร์ทียูผ่านตัวแปลง RS-485 และเก็บข้อมูลพารามิเตอร์ลงฐานข้อมูลซึ่งโปรแกรมที่ใช้ในการเก็บคือ SQLite Studio รวมไปถึงใช้แชร์ข้อมูลบนเว็บเซอร์วิสสำหรับการร้องขอข้อมูลของซอฟต์แวร์ LabVIEW และใช้ภาษา JSON ในการแชร์บนเว็บเซอร์วิสสำหรับการร้องขอข้อมูลของเว็บ เนื่องจากในการแชร์เว็บเซอร์วิสจะใช้ Flask ซึ่งเป็น Web Framework

1. ภาษา Python

Python คือชื่อภาษาที่ใช้ในการเขียนโปรแกรมภาษาหนึ่ง ซึ่งถูกพัฒนาขึ้นมาโดยไม่ยึดติดกับแพลตฟอร์ม กล่าวคือสามารถรันภาษา Python ได้ทั้งบนระบบ Unix, Linux, Windows NT, Windows 2000, Windows XP หรือแม้แต่ระบบ FreeBSD อีกอย่างหนึ่งภาษาตัว นี้เป็นเหมือนอย่าง PHP ทำให้ทุกคนสามารถที่จะนำ Python มาพัฒนาโปรแกรมของเราได้ฟรี ๆ โดยไม่ต้องเสียค่าใช้จ่าย และความเป็น Open Source ทำให้มีคนเข้ามาช่วยกันพัฒนาให้ Python มีความสามารถสูงขึ้นและใช้งานได้ครอบคลุมกับทุกลักษณะงาน

2. ภาษา JSON

JSON หรือ Java Script Object Notation เป็นวิธีการที่ทำให้ JavaScript แลกเปลี่ยนข้อมูลกับ Server ได้อย่างง่าย

ถ้าจะให้อธิบายรูปแบบเป็นประโยคก็คือ JSON ถูกสร้างขึ้นจากชุดข้อมูลของ literal object notation ใน javascript JSON จะใช้ [] แทน array และใช้ {} แทน hash (หรือ associate array) แต่ละสมาชิกคั่นด้วย comma (,) และแต่ละ ชื่อสมาชิกคั่นด้วย colon (:)

3. Flask


**ภาพที่ 2.12** Web Framework [16]

Flask คือ Web Framework ที่เขียนขึ้นมาสำหรับ Python เพื่อใช้ร่วมกัน Web Server เช่น Apache และได้รับการยอมรับจาก Community We Pages ชั้นนำเช่น Pinterest, LinkedIn เป็นต้น โดย Flask ถูกเรียกว่า Micro Framework เพราะว่า มันไม่ต้องการเครื่องมือหรือ Library อะไรมาก อีกทั้ง ไม่จำเป็นต้องมีฐานข้อมูลด้วย แต่อย่างไรก็ตาม Flask ก็ยังรองรับการเพิ่ม Extensions พิเศษได้ ถ้ามันรองรับ Flask

2.5.2 SQLite Studio


**ภาพที่ 2.13** โปรแกรมที่ใช้ทำฐานข้อมูล [9]

SQLite เป็นโปรแกรมฐานข้อมูลที่มีขนาดเล็กมาก (ไม่ถึง 1MB) เก็บฐานข้อมูลเป็นไฟล์ โดยไม่จำเป็นต้องมีเซิร์ฟเวอร์ ทำให้ถูกใช้ในหลาย ๆ โปรแกรมหรือถูกติดตั้งลงไปในอุปกรณ์พกพาหลาย ๆ ชนิด เช่น iPhone, Android เพื่อใช้ในการเก็บข้อมูล

ในส่วนนี้ขอแสดงตัวอย่างวิธีใช้ SQLite แบบรันเป็นคำสั่ง (Command Line) คือรันคำสั่ง sqlite3 เพื่อเข้าใช้งาน และหลาย ๆ OS เช่น Linux, MacOS จะติดตั้ง SQLite มาเป็นดีฟอลต์เลย เราสามารถรันคำสั่ง sqlite3 เพื่อใช้งานได้เลยแต่ถ้าบน Windows ต้องดาวน์โหลดไฟล์โปรแกรมมาก่อน โดยเข้าเว็บไซต์ SQLite Download (http://www.sqlite.org/download.html)

เข้า SQLite ด้วยคำสั่ง sqlite3

```
[user1@cent6-dev ~]$ sqlite3
SQLite version 3.6.20
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
```

คำสั่งใน SQLite (ที่ไม่ใช่ SQL) จะขึ้นต้นด้วยเครื่องหมายจุด “.” หากต้องการดูคำสั่ง และวิธีการใช้ ให้พิมพ์คำสั่ง .help

```
sqlite> .help
.backup ?DB? FILE      Backup DB (default "main") to FILE
.bail ON|OFF           Stop after hitting an error.  Default OFF
.databases             List names and files of attached databases
.dump ?TABLE? ...      Dump the database in an SQL text format
                         If TABLE specified, only dump tables
matching
                         LIKE pattern TABLE.
.echo ON|OFF           Turn command echo on or off
.exit                  Exit this program
...
```

ลองสร้างตารางแล้วลองใส่ข้อมูล ด้วย SQL

```
sqlite> CREATE TABLE users (id int, name varchar(255));

sqlite> INSERT INTO users VALUES (1, 'Alice');
sqlite> INSERT INTO users VALUES (2, 'Ben');

sqlite> SELECT * FROM users;
1|Alice
2|Ben
sqlite>
```

ถ้ารันคำสั่ง sqlite3 อย่างเดียวโดยไม่ได้ระบุไฟล์ฐานข้อมูลแบบนี้ หลังจากสร้าง table ใส่ข้อมูลแล้ว ฐานข้อมูลที่สร้างจะถูกเก็บไว้ในหน่วยความจำชั่วคราว

ถ้าพิมพ์คำสั่ง .exit ออกจาก Sqlite เลย ข้อมูล table ที่สร้างนี้จะหายไป ต้องบันทึกฐานข้อมูลลงเป็นไฟล์ด้วยคำสั่ง .backup

```
sqlite> .backup test1.db
```

พิมพ์คำสั่ง .exit ออกจาก Sqlite

```
sqlite> .exit
[user1@cent6-dev ~]$
```

ลองใช้คำสั่ง ls ดู จะมีไฟล์ test1.db ถูกสร้างขึ้นมา

```
[user1@cent6-dev ~]$ ls -l test1.db
-rw-r--r--. 1 user1 users 2048 Jun 29 15:22 test1.db
```

ใช้คำสั่ง file เพื่อดูชนิดของไฟล์ที่บันทึกได้

```
[user1@cent6-dev ~]$ file test1.db
test1.db: SQLite 3.x database
```

ไฟล์นี้เป็นไฟล์ที่ SQLite ใช้เก็บเป็นฐานข้อมูล ขึ้นอยู่กับว่าจะเก็บไฟล์นี้ไว้ที่ไหน ตั้งชื่อไฟล์ว่าอะไร แล้วเรียกคำสั่ง sqlite3 และตามด้วยชื่อไฟล์นี้ ก็สามารถดึงข้อมูลที่เก็บได้ ตัวอย่างเช่น

```
[user1@cent6-dev ~]$ sqlite3 test1.db
SQLite version 3.6.20
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> SELECT * FROM users;
1|Alice
2|Ben
```

คำสั่ง .tables แสดงชื่อตารางที่มีอยู่

```
sqlite> .tables
users
```

คำสั่ง .schema ตามด้วยชื่อตาราง เพื่อแสดงคำสั่ง SQL ที่ใช้สร้างตารางนั้น

```
sqlite> .schema users
CREATE TABLE users (id int, name varchar(255));
```

เปลี่ยนวิธีการแสดงผล Default การแสดงผลจะเป็นแบบนี้

```
sqlite> SELECT * FROM users;
1|Alice
2|Ben
3|Cat
```

ใช้คำสั่ง .mode ตามด้วยชนิดของการแสดงผลเช่น column

```
sqlite> .mode column
sqlite> SELECT * FROM users;
1           Alice
2           Ben
3           Cat
```

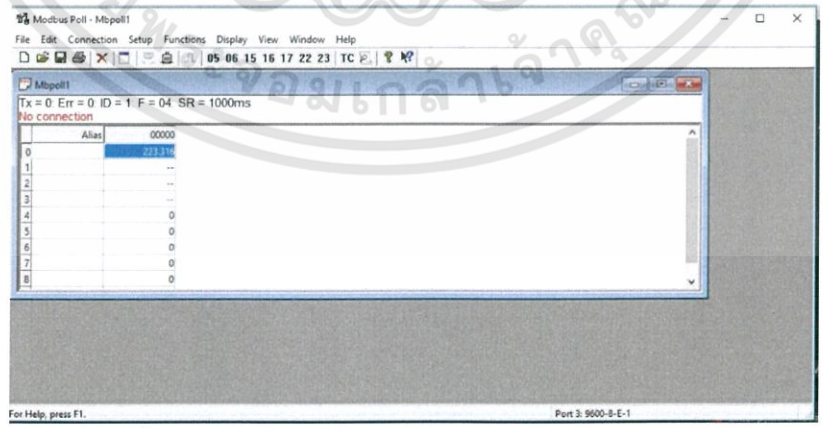
หากต้องการให้แสดงชื่อคอลัมน์ด้วย ให้ใช้คำสั่ง .headers on

```
sqlite> .header ON
sqlite> SELECT * FROM users;
id          name
----------  ----------
1           Alice
2           Ben
3           Cat
```

หากต้องการจัดเก็บเป็นคอนฟิกเก็บไว้ ไม่ต้องมาเปลี่ยนโหมดทุกครั้ง สามารถทำได้โดยสร้างเป็นคอนฟิกไฟล์ .sqliterc อยู่ในไดเร็คทอรี Home

```
[user1@cent6-dev ~]$ cat .sqliterc
.headers on
.mode column

[user1@cent6-dev ~]$ sqlite3 test1.db
-- Loading resources from /home/user1/.sqliterc
SQLite version 3.6.20
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> SELECT * FROM users;
id          name
----------  ----------
1           Alice
2           Ben
3           Cat
sqlite>
```

2.5.3 Modbus Poll



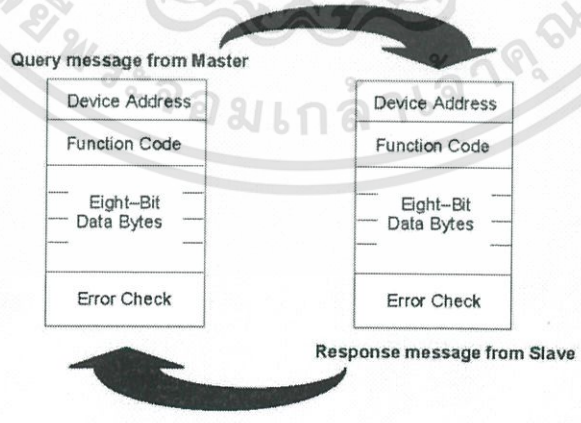

**ภาพที่ 2.14** โปรแกรมสำหรับอ่านค่าโปรโตคอลมอดบัส

## 2.6 โปรโตคอลมอดบัส-อาร์ทียู

โปรโตคอลมอดบัส เป็นโปรโตคอลเพื่อสื่อสารข้อมูลอินพุต/เอาต์พุตและรีจีสเตอร์ภายใน PLC ซึ่งถูกคิดค้นโดย Modicon (ปัจจุบันคือบริษัท Schneider Electric) โปรโตคอลมอดบัสได้เป็นที่ยอมรับกันอย่างกว้างขวางในการติดต่อสื่อสารที่เป็นแบบ Network Protocol อันเนื่องมาจากโปรโตคอลมอดบัสเป็นระบบเปิด ไม่มีค่าใช้จ่าย เชื่อมต่อและพัฒนาง่าย พร้อมทั้งยังสามารถนำโปรโตรคอลนี้ไปใช้งานในอุปกรณ์อื่น ๆ เช่น Digital Power Meter, RTU (Remote Terminal Unit), Remote I/O, PLC เป็นต้น นอกจากนี้โปรโตคอลมอดบัสยังสามารถรองรับและใช้งานร่วมกับ Application จำพวก SCADA และ HMI Software ได้อีกด้วย

โปรโตคอลมอดบัสเป็นการสื่อสารข้อมูลในลักษณะ Master/Slave ซึ่งเป็นการสื่อสารจากอุปกรณ์แม่ (Master) เครื่องเดียว ส่วนใหญ่มักเป็นซอฟต์แวร์คอมพิวเตอร์หรืออุปกรณ์แสดงผล HMI ไปยังอุปกรณ์ลูก (Slave) ได้หลาย ๆ เครื่อง โดยสามารถกำหนดหมายเลขอุปกรณ์ได้สูงสุด 255 เครื่อง โดยมีลักษณะการส่งข้อมูล 2 แบบ คือ ข้อมูลแบบแอสกี (ASCII) และข้อมูลแบบเลขฐานสอง (Binary) ในโปรโตคอลมอดบัสที่สื่อสารข้อมูลแบบ ASCII จะเรียกมอดบัสแอสกี (Modbus ASCII) และโปรโตคอลมอดบัสที่สื่อสารข้อมูลแบบเลขฐานสอง จะเรียกมอดบัส-อาร์ทียู (Modbus-RTU) ทำให้มีความแตกต่างในการกำหนดค่าพอร์ตสื่อสาร

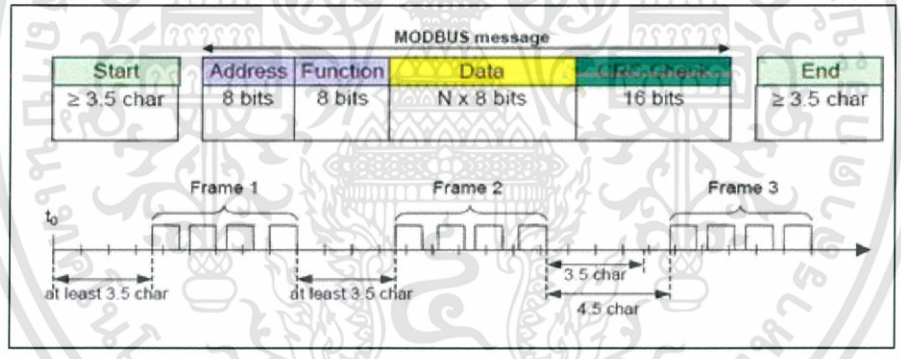
การรับส่งข้อมูลด้วยโปรโตคอลมอดบัสสามารถเลือกได้ 2 โหมด คือ โหมดแอสกี (ASCII) และโหมดอาร์ทียู (RTU) ซึ่งทั้ง 2 โหมด นี้มีความแตกต่างกันที่การกำหนดรูปแบบของชุดข้อมูลภายในเฟรม จะเลือกโหมดใดก็ได้แต่มีเงื่อนไขว่า อุปกรณ์ทุกตัวที่ต่อร่วมกันอยู่ในบัสหรือเครือข่ายเดียวกัน จะต้องตั้งให้เลือกใช้โหมดเดียวกันทั้งหมด ซึ่งในโครงงานที่ทางกลุ่มจัดทำขึ้นมานี้จะเกี่ยวข้องกับโปรโตคอลมอดบัส-อาร์ทียู



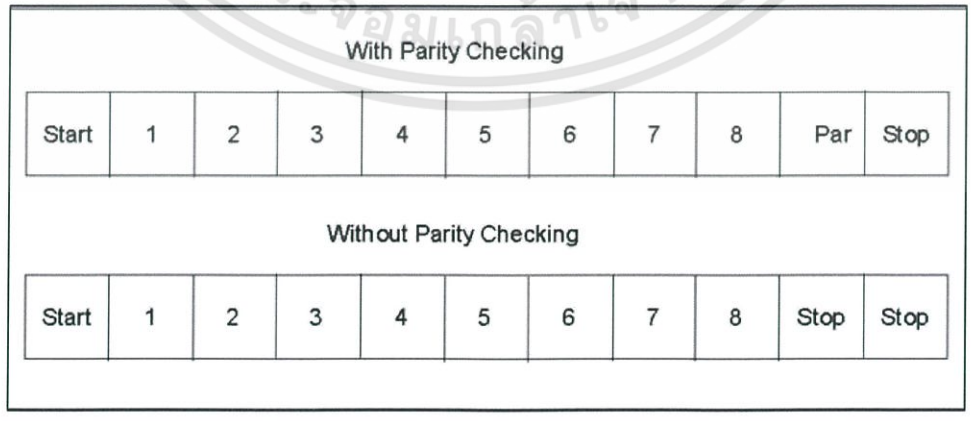

**ภาพที่ 2.15** การติดต่อสื่อสารแบบ Master/Slave [12]

2.6.1 คุณสมบัติของโปรโตคอลมอดบัส-อาร์ทียู

เฟรมข้อมูลในโหมดอาร์ทียู ประกอบด้วยข้อมูลแสดงตำแหน่งแอดเดรส 1 ไบต์ หมายเลขฟังก์ชัน 1 ไบต์ ข้อมูลที่ทำการรับส่งจำนวนมากสุดไม่เกิน 252 ไบต์ และรหัสตรวจสอบความถูกต้องของข้อมูลแบบ CRC (Cyclical Redundancy Checking) ขนาด 2 ไบต์ ค่า CRC นี้เป็นค่าที่คำนวณมาจากข้อมูลทุกไบต์ ไม่รวมบิต Start, Stop และ Parity Check โดยที่ตัว Slave ตัวที่ส่งข้อมูลออกมาจะสร้างรหัส CRC แล้วส่งตามท้ายไบต์ข้อมูลออกมา หลังจากนั้นเมื่อ Master ได้รับเฟรมข้อมูลและถอดข้อมูลออกจากเฟรมแล้วจะทำการคำนวณค่า CRC ตามสูตรเดียวกับ Slave เพื่อทำการเปรียบเทียบค่า CRC ทั้ง 2 ค่าว่าตรงกันหรือไม่ หากไม่ตรงกันแสดงว่าเกิดความผิดพลาดในการรับส่งข้อมูลในโหมดอาร์ทียู การรับส่งข้อมูล 1 ไบต์ ไม่ว่าจะเป็นข้อมูลส่วนใดภายในเฟรมจะต้องทำการส่งบิตข้อมูลรวม 11 บิต คือ บิตเริ่มต้น (Start) 1 บิต บิตข้อมูล 8 บิต บิตตรวจสอบ Parity ของข้อมูล 1 บิตและบิตหยุด 1 บิต (Stop) 1 บิต หรือหากเลือกแบบไม่มีบิต Parity ก็จะเป็นแบบ Stop แทน 2 บิต สำหรับการกำหนดให้มีบิต Parity นั้นสามารถเลือกเป็นแบบคู่ (Even Parity) หรือคี่ (Odd Parity) ก็ได้ และหากต้องการออกแบบให้สอดคล้องกับอุปกรณ์ที่มีใช้กันทั่วไปมากที่สุด ควรเลือกแบบคู่โดยที่สามารถปรับเปลี่ยนเป็นแบบคี่หรือไม่มีการตรวจสอบ Parity (No Parity) ได้ด้วย




**ภาพที่ 2.16** ลักษณะเฟรมข้อมูลของโปรโตคอลมอดบัส-อาร์ทียู [12]




**ภาพที่ 2.17** ลักษณะข้อมูลแต่ละไบต์ของโปรโตคอลมอดบัส-อาร์ทียู [12]

2.6.2 Commonly Use Function Code

01-Read Coil Status

02-Read Input Status

03-Read Holding Register

04-Read Input Register

05-Write Single Coil Status

06-Write Single Register

15-Multiple Coil Write

16-Multiple Register Write

2.6.3 Modbus Data Type and Address Space

**ตารางที่ 2.2** ชนิดข้อมูลโปรโตคอลมอดบัส และตำแหน่งข้อมูล

| Object type | Access | Size | Address |
|---|---|---|---|
| Coil | Read/Write | 1-bit | 00001-09999 |
| Discrete Input | Read Only | 1-bit | 10001-19999 |
| Input Register | Read Only | 16-bits | 30001-39999 |
| Holding Register | Read-Write | 16-bits | 40001-49999 |

## 2.7 การติดต่อสื่อสาร (Communication)

2.7.1 HTTP Protocol

HTTP ย่อมาจาก Hypertext Transfer Protocol คือ โปรโตคอลสื่อสารสำหรับการแลกเปลี่ยนสารสนเทศผ่านอินเทอร์เน็ต โดยหลักแล้วใช้ในการรับเอกสารข้อความหลายมิติที่นำไปสู่การเชื่อมต่อกับ World Wide Web (WWW ) จะใช้เมื่อเรียกโปรแกรมweb browser เช่น Firefox, Google Chrome, Safari, Opera และ IE Microsoft Internet Explorer เรียกดูข้อมูลหรือเว็บเพจ โปรแกรมบราวเซอร์ดังกล่าวจะใช้โปรโตคอล HTTP ซึ่งโปรโตคอลนี้ทำให้เซิร์ฟเวอร์ส่งข้อมูลมาให้บราวเซอร์ตามต้องการ และบราวเซอร์จะนำข้อมูลมาแสดงผลบนจอภาพได้อย่างถูกต้อง

ในการแลกเปลี่ยนข้อมูลกันระหว่าง Server และ Client ของ World Wide Web (Server) โดยส่งข้อมูลแบบ Clear text คือ ข้อมูลที่ทำการส่งไปนั้น ไม่ได้ทำการเข้ารหัส ทำให้สามารถถูกดักจับและอ่านข้อมูลได้ง่าย

2.7.2 Wi-Fi


**ภาพที่ 2.18** ตรารับรองอุปกรณ์ที่รองรับมาตรฐาน Wi-Fi ประเภทต่าง ๆ [19]

Wi-Fi (ย่อมาจาก Wireless Fidelity) คือ องค์กรหนึ่ง ที่ทำหน้าที่ทดสอบผลิตภัณฑ์ Wireless LAN หรือระบบ Network แบบไร้สายภายใต้เทคโนโลยีการสื่อสาร ภายใต้มาตรฐาน IEEE 802.11 ว่าอุปกรณ์ทุกตัวซึ่งต่างยี่ห้อกันนั้นมันสามารถติดต่อสื่อสารกันได้โดยไม่ มีปัญหา หากว่าอุปกรณ์ตัวนั้นผ่านตามมาตรฐานเขาก็จะปั๊ม ตรา Wi-Fi Certified ซึ่งเป็นอันรู้กันว่าอุปกรณ์ชิ้นนั้นสามารถติดต่อสื่อสารกับอุปกรณ์ตัวอื่น ที่มีตรา Wi-Fi Certified นี้ได้เช่นกัน แต่ทำไปทำมามันกลายเป็นคำศัพท์สำหรับอุปกรณ์ LAN ไร้สายไปโดยปริยาย จนบางคนก็เรียกกันจนติดปาก

ประโยชน์ของ Wi-Fi การเชื่อมต่อเครือข่ายไรสายหรือ Wireless LAN นั้นเป็นการเชื่อมต่ออุปกรณ์ต่าง ๆ ให้สามารถใช้งานอินเตอร์เน็ตและแลกเปลี่ยนข้อมูลซึ่งกันและกันมาก โดยประโยชน์ของ Wi-Fi นั้นมีอยู่มากมายอาทิเช่น

1. ช่วยลดค่าใช้จ่ายในการเดินระบบเครือข่ายซึ่งปกติแล้วการเชื่อมโยงเครือข่ายนั้นจำเป็นต้องใช้สายนำสัญญาณในการเชื่อมโยงเครือข่าย และต้องเสียค่าใช้จ่ายในการเดินสายสัญญาณแต่สำหรับระบบ Wi-Fi ไม่จำเป็นเพราะระบบ Wi-Fi จะส่งคลื่นวิทยุผ่านอากาศไปยังเครื่องรับ

2. มีความยืดหยุ่นในการใช้งานเพราะการใช้งานไวไฟนั้นไม่จำเป็นต้องอยู่กับที่สามารถเคลื่อนย้ายไปไหนก็ได้ภายในรัศมีของการกระจายสัญญาณ

3. ใช้มาตรฐาน IEEE 802 ซึ่งเป็นมาตรฐานที่ยอมรับกันทั่วไปพร้อมกันนั้นอุปกรณ์ต่าง ๆ ที่เกี่ยวกับ Wi-Fi ก็มีราคาถูกลงและมีให้เลือกซื้อหาหลายยี่ห้อ

4. ช่วยส่งเสริมธุรกิจและธุรกรรมทางการเงิน อาทิเช่นการซื้อขายผ่านอินเตอร์เน็ต การทำธุรกรรมผ่านอินเตอร์เน็ต หรือแม้กระทั่งเป็นจุดเด่นของการดำเนินธุรกิจด้านบริการได้อีกด้วย

สิ่งที่กล่าวมานี้เป็นเพียงประโยชน์บางส่วนของการเชื่อมต่อเครือข่ายไร้สายหรือไวไฟ ซึ่งการใช้งาน Wi-Fi นั้นมีความสะดวกสบายมากแต่ถึงกระนั้นถ้าไม่อยู่ในรัศมีของเครื่องกระจายสัญญาณไวไฟเราก็จะไม่สามารถใช้งานอินเตอร์ได้เหมือนกัน

# บทที่ 3
# เครื่องบันทึกข้อมูลสำหรับการวัดพลังงานระยะไกล

## 3.1 กล่าวนำ

สำหรับบทที่ 3 เป็นขั้นตอนการสร้างเครื่องบันทึกข้อมูลสำหรับการวัดพลังงานระยะไกล โดยจะกล่าวถึงการส่งผ่านข้อมูลในการวัดพลังงานระยะไกล การติดตั้งบอร์ด ASUS Tinker สำหรับการใช้งาน การเขียนโปรแกรมเพื่อสร้างเครื่องบันทึกข้อมูล และฟังก์ชันการทำงานของเครื่องบันทึกข้อมูลโดยใช้เว็บเซอร์วิสผ่าน HTTP Protocol

## 3.2 การส่งผ่านข้อมูลในการวัดพลังงานระยะไกล

3.2.1 การอ่านข้อมูลจากเพาเวอร์มิเตอร์ด้วยโปรโตคอลมอดบัส

ศึกษาการอ่านค่าโปรโตคอลมอดบัสของพารามิเตอร์ในแต่ละตัวที่ต้องการใช้แสดงผล โดยมีรูปแบบการอ่านดังนี้

> **The format for each byte in RTU mode is:**
>
> **Coding System:** 8-bit per byte
>
> **Data Format:** 4 bytes (2 registers) per parameter.Floating point format (to IEEE 754).Most significant register first (Default). The default may be changed if required -See Holding Register "Register Order" parameter.
>
> **Error Check Field:** 2 byte Cyclical Redundancy Check (CRC)
>
> **Framing:** 1 start bit
>
> 8 data bits, least significant bit sent first
>
> 1 bit for even/odd parity (or no parity)
>
> 1 stop bit if parity is used; 1 or 2 bits if no parity

ในส่วนของพารามิเตอร์นั้นจะมีจำนวนหลายค่า ทางผู้จัดทำจึงเลือกมาอ่านค่า Input Registers เพียง 34 ค่า ซึ่งจะแสดงดังตารางต่อไปนี้

ตารางที่ 3.1 Input Registers

| Address (Register) | SDM630MT Input Register Parameter | | Modbus Protocol Start Address Hex | | 3 Ø | 3 Ø | 1 Ø |
|---|---|---|---|---|---|---|---|
| | Description | Units | Hi Byte | Lo Byte | 4 W | 3 W | 2 W |
| 30001 | Phase 1 line to neutral volts. | Volts | 00 | 00 | √ | X | √ |
| 30003 | Phase 2 line to neutral volts. | Volts | 00 | 02 | √ | X | X |
| 30005 | Phase 3 line to neutral volts. | Volts | 00 | 04 | √ | X | X |
| 30007 | Phase 1 current. | Amps | 00 | 06 | √ | √ | √ |
| 30009 | Phase 2 current. | Amps | 00 | 08 | √ | √ | X |
| 30011 | Phase 3 current. | Amps | 00 | 0A | √ | √ | X |
| 30013 | Phase 1 power. | Watts | 00 | 0C | √ | X | √ |
| 30015 | Phase 2 power. | Watts | 00 | 0E | √ | X | √ |
| 30017 | Phase 3 power. | Watts | 00 | 10 | √ | X | X |
| 30019 | Phase 1 volt amps. | VA | 00 | 12 | √ | X | √ |
| 30021 | Phase 2 volt amps. | VA | 00 | 14 | √ | X | X |
| 30023 | Phase 3 volt amps. | VA | 00 | 16 | √ | X | X |
| 30025 | Phase 1 volt amps reactive. | VAr | 00 | 18 | √ | X | √ |
| 30027 | Phase 2 volt amps reactive. | VAr | 00 | 1A | √ | X | X |
| 30029 | Phase 3 volt amps reactive. | VAr | 00 | 1C | √ | X | X |
| 30031 | Phase 1 power factor (1). | None | 00 | 1E | √ | X | √ |
| 30033 | Phase 2 power factor (1). | None | 00 | 20 | √ | X | X |
| 30035 | Phase 3 power factor (1). | None | 00 | 22 | √ | X | X |
| 30053 | Total system power. | Watts | 00 | 34 | √ | √ | √ |
| 30057 | Total system volt amps. | VA | 00 | 38 | √ | √ | √ |
| 30061 | Total system VAr. | VAr | 00 | 3C | √ | √ | √ |
| 30063 | Total system power factor (1). | None | 00 | 3E | √ | √ | √ |
| 30071 | Frequency of supply voltages. | Hz | 00 | 46 | √ | √ | √ |
| 30073 | Import Wh since last reset (2). | kWh/M | 00 | 48 | √ | √ | √ |
| 30075 | Export Wh since last reset (2). | kWH/M | 00 | 4A | √ | √ | √ |
| 30077 | Import VArh since last reset (2). | kVArh/ | 00 | 4C | √ | √ | √ |
| 30079 | Export VArh since last reset (2). | kVArh/ | 00 | 4E | √ | √ | √ |
| 30081 | VAh since last reset (2). | kVAh/ | 00 | 50 | √ | √ | √ |
| 30201 | Line 1 to Line 2 volts. | Volts | 00 | C8 | √ | √ | X |
| 30203 | Line 2 to Line 3 volts. | Volts | 00 | CA | √ | √ | X |
| 30205 | Line 3 to Line 1 volts. | Volts | 00 | CC | √ | √ | X |
| 30235 | Phase 1 L/N volts THD | % | 00 | EA | √ | X | √ |
| 30237 | Phase 2 L/N volts THD | % | 00 | EC | √ | X | X |
| 30239 | Phase 3 L/N volts THD | % | 00 | EE | √ | X | X |

3.2.2 การอ้างอิงพารามิเตอร์สำหรับโปรแกรม SQLite Studio

ซึ่งในบทที่2ได้กล่าวไว้แล้วว่าโปรแกรม SQLite Studio ได้ใช้ในการเก็บบันทึกฐานข้อมูล ในส่วนของบทที่ 3 นั้นจะกล่าวถึงในส่วนประกอบของตาราง โดยมีค่าพารามิเตอร์ ดังต่อไปนี้

**ตารางที่ 3.2** ชื่อพารามิเตอร์ และชื่อคอลัมน์ในตาราง

| ชื่อ Parameter | ชื่อคอลัมน์ใน table |
|---|---|
| Date_time | date_time |
| Phase 1 line to neutral volts. | V1 |
| Phase 2 line to neutral volts. | V2 |
| Phase 3 line to neutral volts. | V3 |
| Phase 1 current. | Current1 |
| Phase 2 current. | Current2 |
| Phase 3 current. | Current3 |
| Phase 1 power. | Power1 |
| Phase 2 power. | Power2 |
| Phase 3 power. | Power3 |
| Phase 1 volt amps. | Volt_Amps1 |
| Phase 2 volt amps. | Volt_Amps2 |
| Phase 3 volt amps. | Volt_Amps3 |
| Phase 1 volt amps reactive. | Volt_Amps_React1 |
| Phase 2 volt amps reactive. | Volt_Amps_React2 |
| Phase 3 volt amps reactive. | Volt_Amps_React3 |
| Phase 1 power factor (1). | Power_Factor1 |

| | |
|---|---|
| Phase 2 power factor (1). | Power_Factor2 |
| Phase 3 power factor (1). | Power_Factor3 |
| Total system power. | Tol_power |
| Total system volt amps. | Tol_VoltAmp |
| Total system VAr. | Tol_VAr |
| Total system power factor (1). | Tol_power_factor |
| Frequency of supply voltages. | Frequency |
| Import Wh since last reset (2). | Import_Wh |
| Export Wh since last reset (2). | Export_Wh |
| Import VArh since last reset (2). | Import_VArh |
| Export VArh since last reset (2). | Export_VArh |
| VAh since last reset (2). | VAh_last_reset |
| Line 1 to Line 2 volts. | L1_to_L2 |
| Line 2 to Line 3 volts. | L2_to_L3 |
| Line 3 to Line 1 volts. | L3_to_L1 |
| Phase 1 L/N volts THD | P1_LN_THD |
| Phase 2 L/N volts THD | P2_LN_THD |
| Phase 3 L/N volts THD | P3_LN_THD |

ชื่อคอลัมน์ในตารางจะถูกนำไปใช้เขียนโค้ด Python ในส่วนของการประกาศตัวแปรเพื่อที่จะบันทึกค่าที่อ่านได้จากโปรโตคอลมอดบัสเข้าฐานข้อมูล

3.2.3 รูปแบบพารามิเตอร์ในการจัดเก็บข้อมูล

ในการบันทึกข้อมูลลงฐานข้อมูลนั้นได้มีการจัดเตรียมพื้นที่สำหรับเก็บข้อมูลไว้ ซึ่งจะพูด

ถึงในส่วนของจำนวน Rows ที่มากที่สุดใน 1 ตารางคือ $2^{64}$ (18446744073709551616 or about 1.8e+19) แต่ในความเป็นจริงแล้วเราไม่สามารถใช้ได้ถึงขนาดนั้น โดยจะมีการคำนวณพื้นที่จัดเก็บข้อมูลดังต่อไปนี้

1. Date Time มี Format เป็น YYYY-MM-DD HH:MM:SS มีค่าอยู่ระหว่าง 1000-01-01 00:00:00' to '9999-12-31 23:59:59 ซึ่งจะมีขนาด 8 Bytes

2. Parameter แต่ละค่าจะมีขนาด 2 registers ซึ่งจะมีขนาดเท่ากับ 4 Bytes

<u>วิธีการคำนวณ</u>

Parameter 34 ตัว จะมีขนาด 34 x 4 Bytes = 136 Bytes

Date Time จะมีขนาด 8 Bytes

ดังนั้นขนาดข้อมูล 1 row = 136 + 8 = 144 Bytes

*จะบันทึกค่าเป็นเรียลไทม์ทุก ๆ 5 วินาที*

จะคิดจาก 1 ชั่วโมง จะมีทั้งหมด 3,600 วินาที/5 วินาที = 720 rows

แสดงว่า 1 ชั่วโมง จะมีขนาดข้อมูลเป็น 144 Bytes x 720 rows = 103,680 Bytes

ในระยะเวลา 1 วัน จะมีขนาดข้อมูลเป็น

103,680 Bytes x 24 hrs. = 2,488,320 Bytes

ในระยะเวลา 1 เดือน จะมีขนาดข้อมูลเป็น

2,488,320 Bytes x 30 days = 74,649,600 Bytes

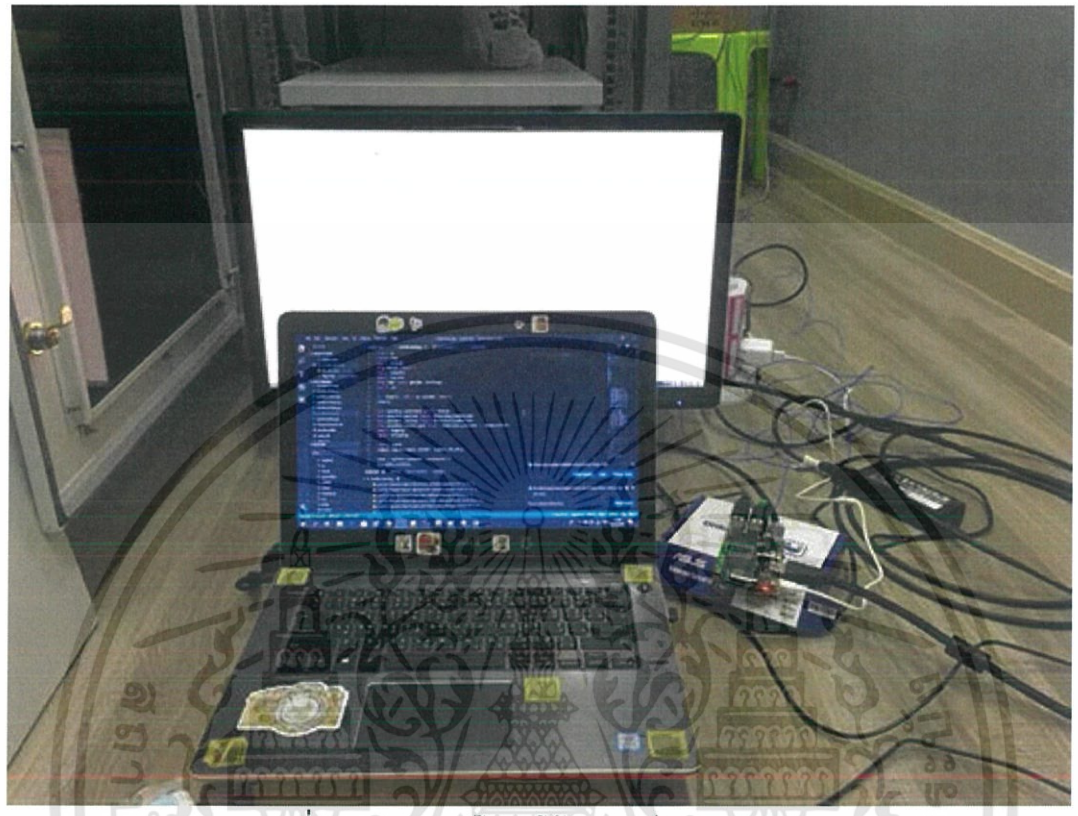
ในระยะเวลา 1 ปี จะมีขนาดข้อมูลเป็น

74,649,600 Bytes x 12 months = 895,795,200 Bytes หรือเท่ากับ 895.795 MB

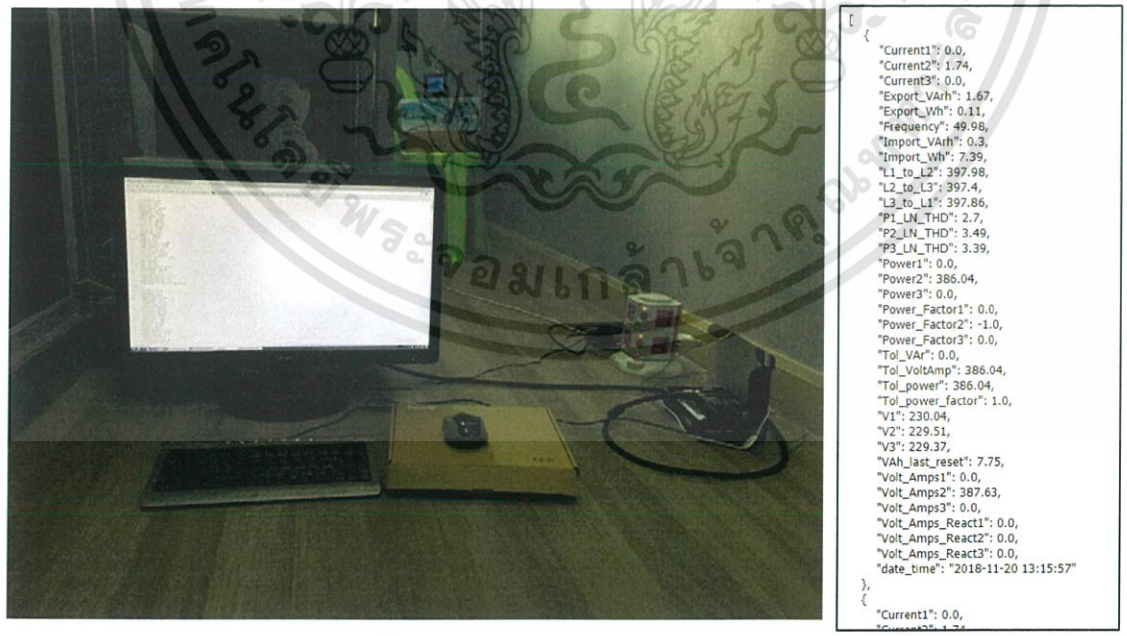
**3.3 การติดตั้งบอร์ดสำหรับการใช้งาน**

นำ Micro SD Card ใส่ไปในช่องสำหรับใส่ Micro SD Card ที่อยู่ใต้บอร์ด ASUS Tinker แล้วนำสาย HDMI ต่อระหว่างบอร์ดและ Monitor ต่อเมาส์และคีย์บอร์ดเข้ากับบอร์ดที่ช่องUSB แล้วต่อตัวแปลง RS-485 ระหว่างบอร์ดที่ช่อง USB กับเพาเวอร์มิเตอร์ หลังจากนั้นต่อสาย LAN เข้าระหว่างบอร์ดกับ

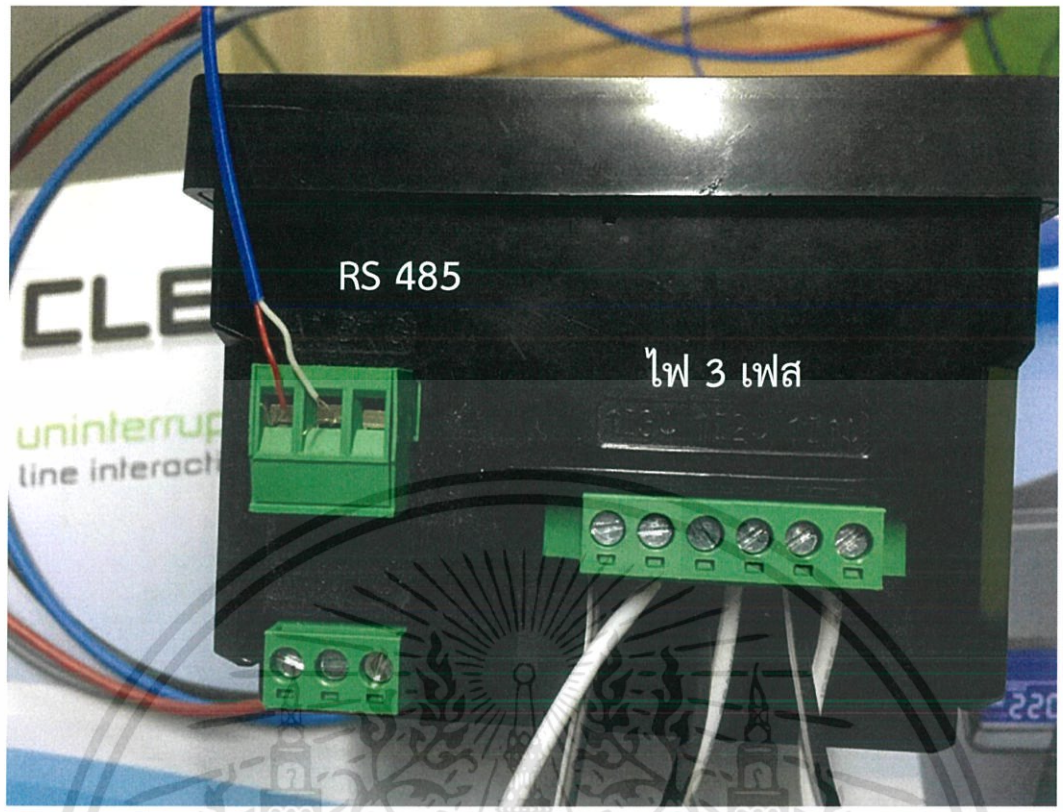
คอมพิวเตอร์เพื่อดาวน์โหลดโค้ดที่ได้ทำการเขียนโปรแกรมไปยังบอร์ด แล้วต่อไฟเลี้ยงโดยใช้สาย USB ที่มี Adaptor จ่ายไฟให้ 2 A ที่ช่อง Micro USB Power-In



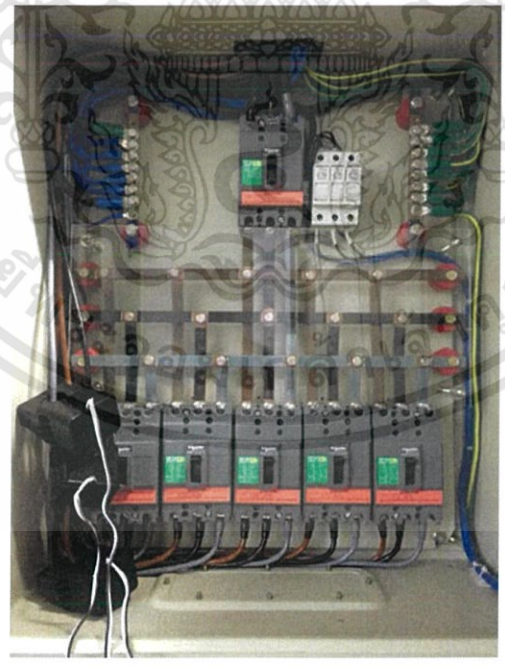
**ภาพที่ 3.1** การดาวน์โหลดโค้ดลงบอร์ด ASUS Tinker



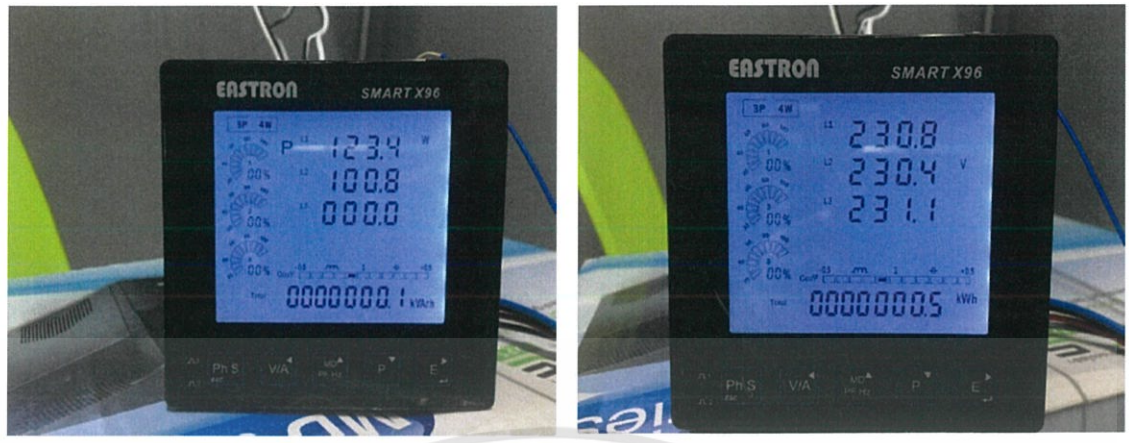

**ภาพที่ 3.2** การควบคุมผ่านบอร์ด ASUS Tinker




**ภาพที่ 3.3** การต่อตัวแปลงสัญญาณ RS-485 และระบบไฟฟ้า 3 เฟส



**ภาพที่ 3.4** การต่อหม้อแปลงกระแสทั้ง 3 เฟส




**ภาพที่ 3.5** ตัวอย่างผลลัพธ์ที่แสดงบนหน้าจอเพาเวอร์มิเตอร์

## 3.4 การเขียนโปรแกรมเพื่อสร้างเครื่องบันทึกข้อมูล

ในขั้นตอนนี้จะแบ่งออกเป็น 2 หัวข้อ คือ

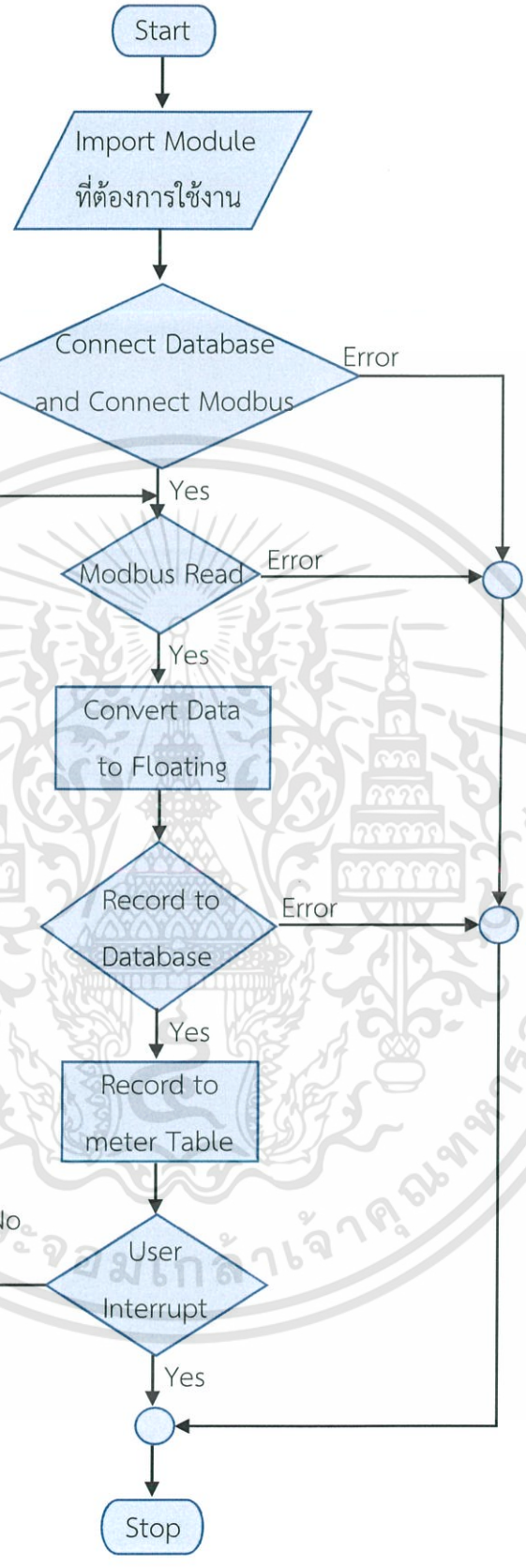
3.4.1 การอ่านค่าโปรโตคอลมอดบัส-อาร์ทียู แล้วบันทึกลงฐานข้อมูล

3.4.2 การแชร์ฐานข้อมูลโดยใช้เว็บเซอร์วิสผ่าน HTTP Protocol

จะมีรายละเอียดดังต่อไปนี้

3.4.1 การอ่านค่าโปรโตคอลมอดบัส-อาร์ทียู แล้วบันทึกลงฐานข้อมูล

ในการอ่านค่าโปรโตคอลมอดบัส-อาร์ทียู แล้วบันทึกลงตารางฐานข้อมูลนั้นสามารถนำมาเขียนเป็นFlowchart ได้ดังนี้

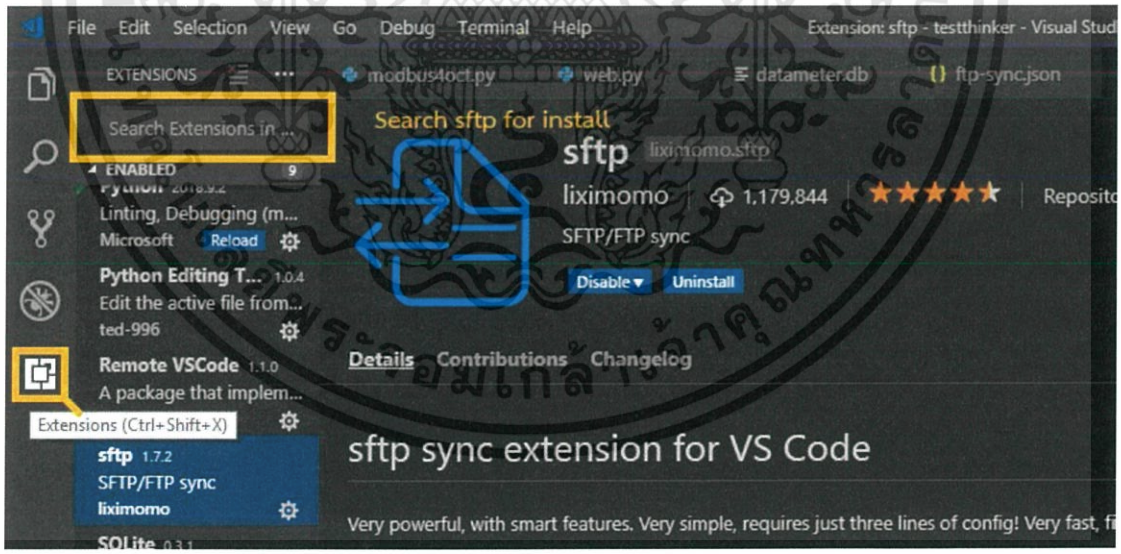



**ภาพที่ 3.6** Flowchart การอ่านค่าโปรโตคอลมอดบัส-อาร์ทียูแล้วบันทึกลงฐานข้อมูล

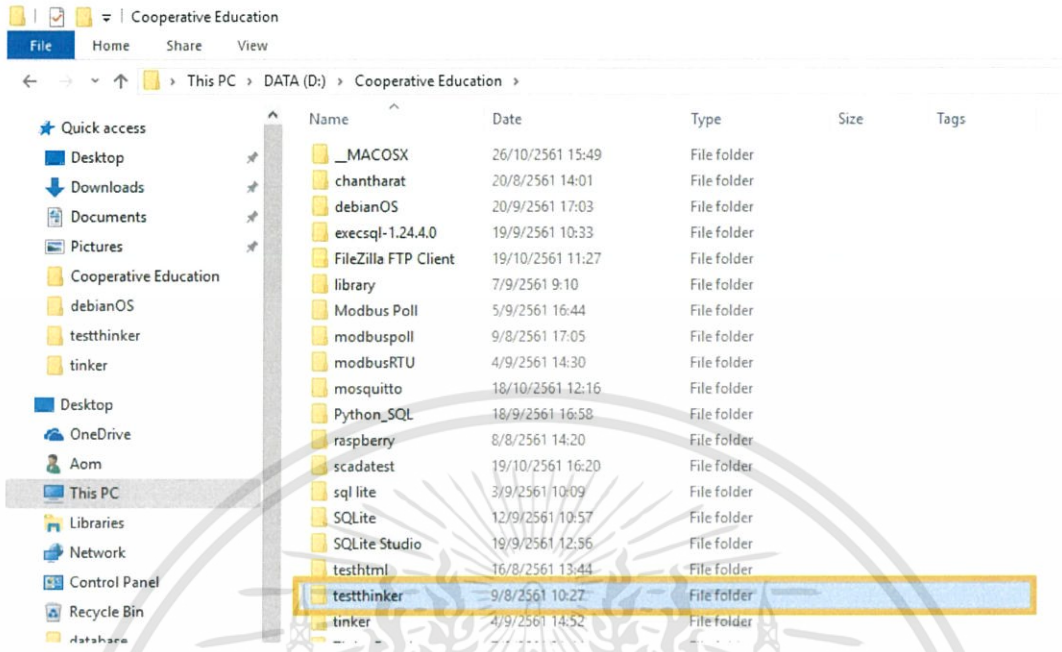
ส่วนโปรแกรม Visual Studio Code (VS Code) ที่ใช้เขียนโปรแกรมภาษา Python จะมีขั้นตอนในการเขียนโค้ดดังนี้

1. ดาวน์โหลดโปรแกรม Visual Studio Code ได้จาก Link https://code.visualstudio.com/download แล้ว ทำการติดตั้ง Python ในโปรแกรม


2. ติดตั้ง Extensions ในส่วนของ sftp Protocol ซึ่งเป็นโปรโตคอลการถ่ายโอนไฟล์ ในที่นี้จะถ่ายโอนไฟล์ที่เขียนในโน้ตบุ๊กไปยังบอร์ด ASUS Tinker

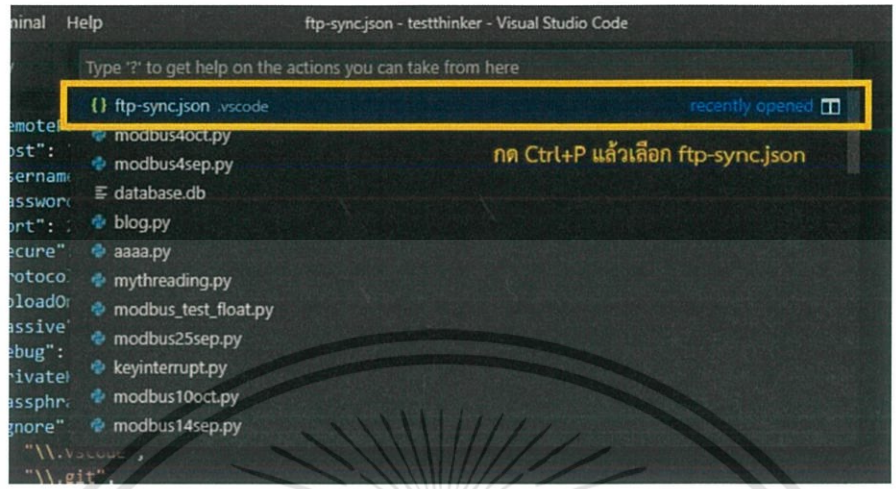


3. ทำการสร้าง Folder เพื่อที่จะทำการเก็บ File Code Python

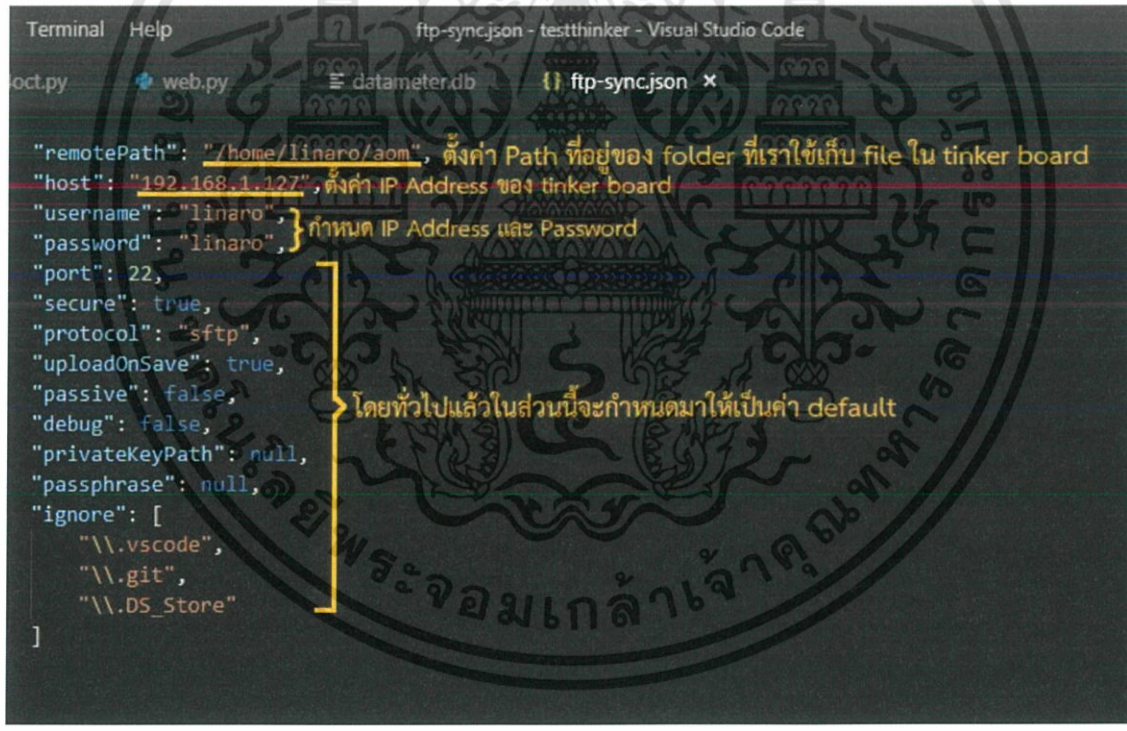



4. กลับมาที่โปรแกรม Visual Studio Code จะทำการสร้าง File (Fileที่สร้างขึ้นจะต้องใส่ .py ให้ด้วยเพื่อเป็นการระบุชนิดของไฟล์ในการเขียนโค้ด Python) โดยจะสร้างเก็บไว้ใน Folder ที่ถูกสร้างไว้ในขั้นตอนที่3 มีขั้นตอนคือ Explorer >New File > ตั้งชื่อ File เพื่อสร้าง

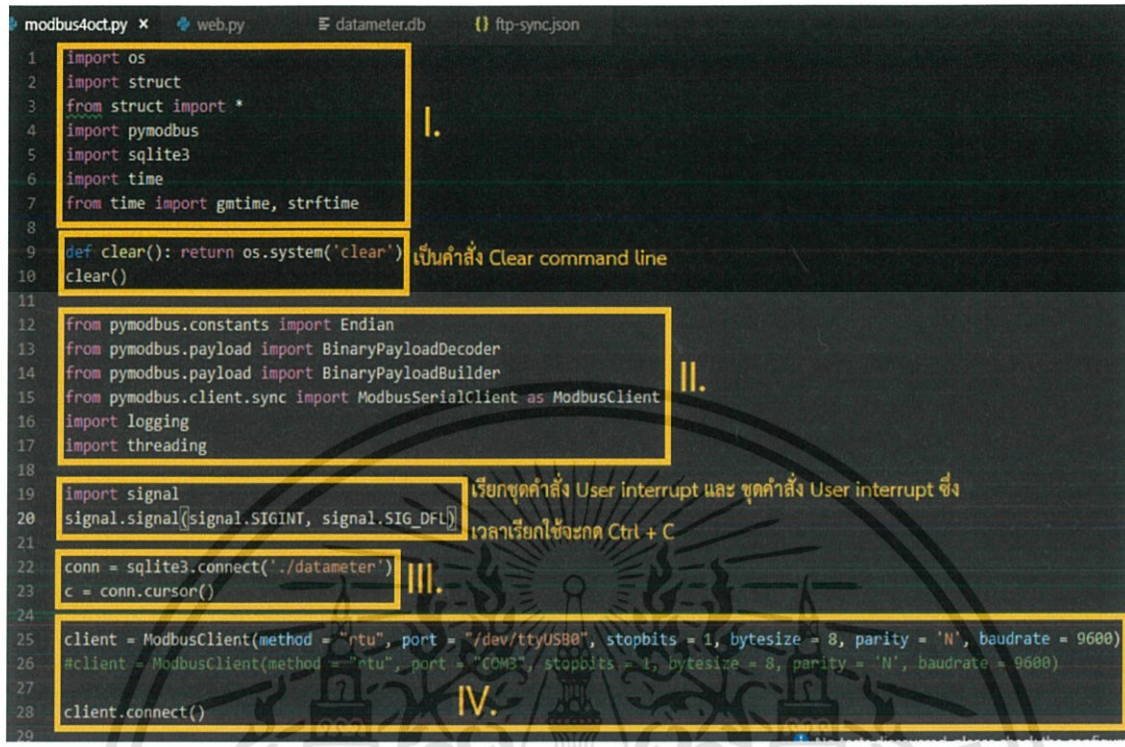
5. ตั้งค่า sftp Protocol โดยการกด Ctrl+P > พิมพ์ sftp ในช่อง Search > เลือก ftp-sync.json



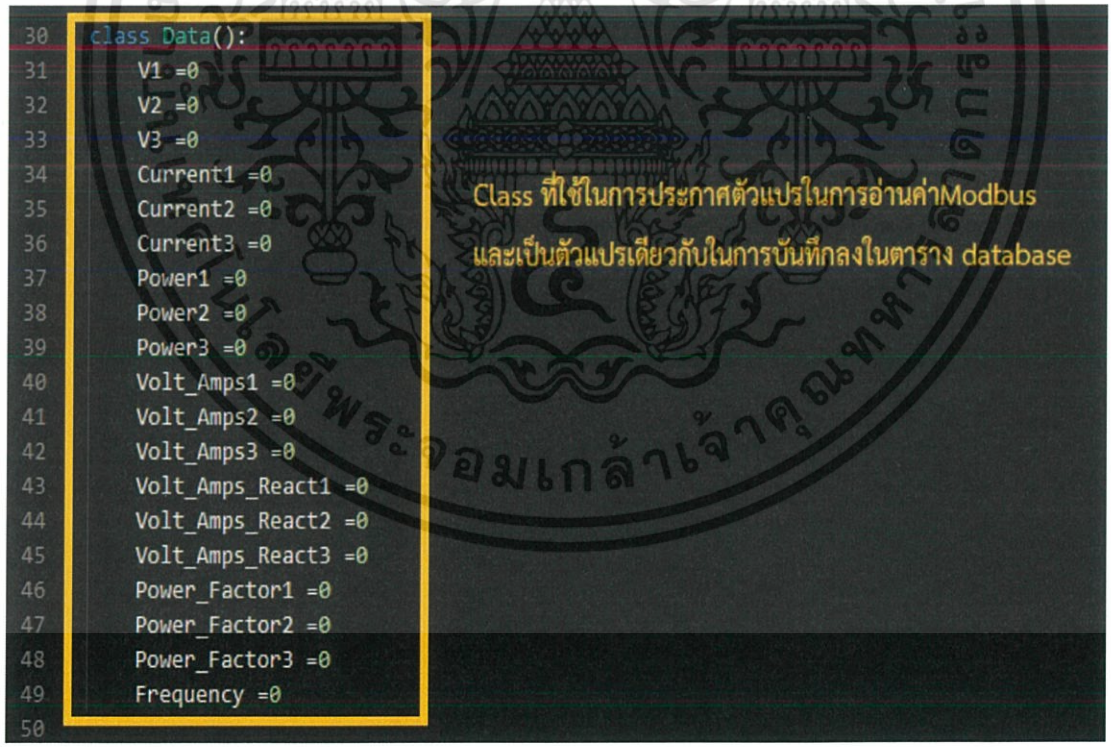

แล้วตั้งค่าตามภาพดังนี้



6. ส่วนนี้จะเป็นโค้ดที่ใช้อ่านโปรโตคอลมอดบัส-อาร์ทียู และบันทึกลงฐานข้อมูล




```
import os
import struct
from struct import *
import pymodbus
import sqlite3
import time
from time import gmtime, strftime

def clear(): return os.system('clear')
clear()

from pymodbus.constants import Endian
from pymodbus.payload import BinaryPayloadDecoder
from pymodbus.payload import BinaryPayloadBuilder
from pymodbus.client.sync import ModbusSerialClient as ModbusClient
import logging
import threading

import signal
signal.signal(signal.SIGINT, signal.SIG_DFL)

conn = sqlite3.connect('./datameter')
c = conn.cursor()

client = ModbusClient(method = "rtu", port = "/dev/ttyUSB0", stopbits = 1, bytesize = 8, parity = 'N', baudrate = 9600)
#client = ModbusClient(method = "rtu", port = "COM3", stopbits = 1, bytesize = 8, parity = 'N', baudrate = 9600)

client.connect()
```



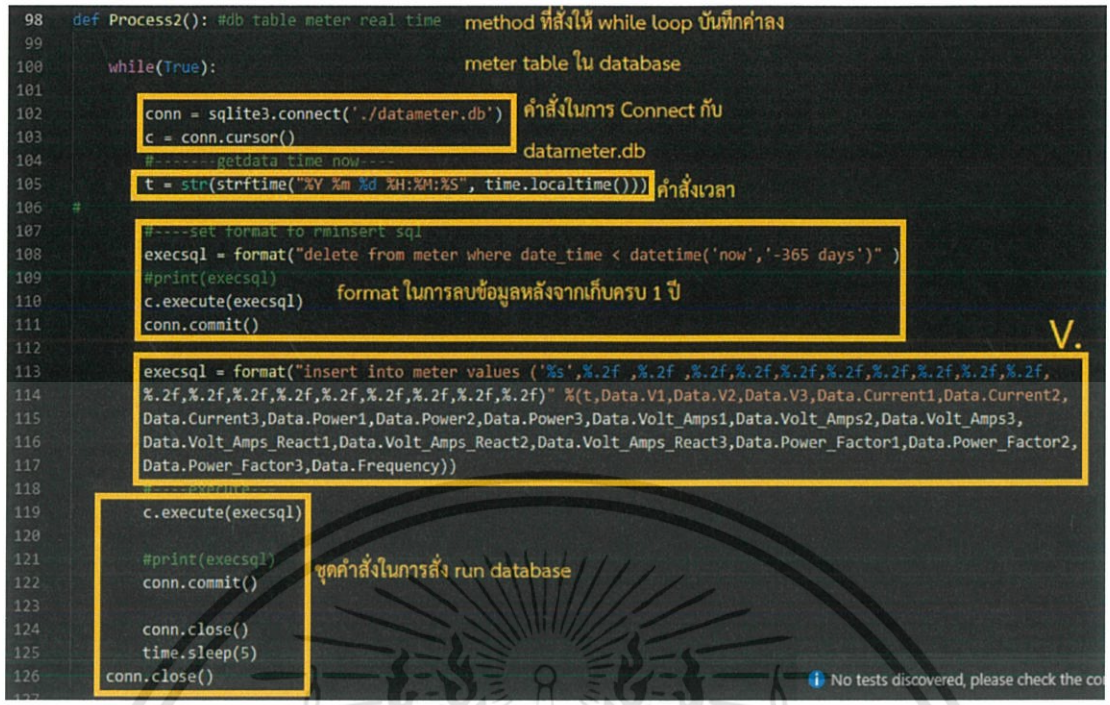

```
class Data():
    V1 =0
    V2 =0
    V3 =0
    Current1 =0
    Current2 =0
    Current3 =0
    Power1 =0
    Power2 =0
    Power3 =0
    Volt_Amps1 =0
    Volt_Amps2 =0
    Volt_Amps3 =0
    Volt_Amps_React1 =0
    Volt_Amps_React2 =0
    Volt_Amps_React3 =0
    Power_Factor1 =0
    Power_Factor2 =0
    Power_Factor3 =0
    Frequency =0
```

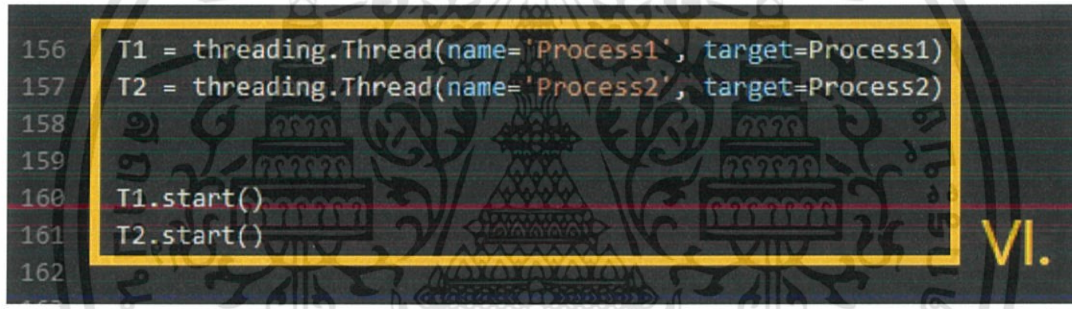
```python
def Process1(): #read_modbus

    while(True):
        result = client.read_input_registers(0x0000,60,unit= 0x01)
        r = result.registers
        Data.V1 = round(unpack('>f', pack('>HH', r[0], r[1]))[0],2)
        Data.V2 = round(unpack('>f', pack('>HH', r[2], r[3]))[0],2)
        Data.V3 = round(unpack('>f', pack('>HH', r[4], r[5]))[0],2)
        Data.Current1 = round(unpack('>f', pack('>HH', r[6], r[7]))[0],2)
        Data.Current2 = round(unpack('>f', pack('>HH', r[8], r[9]))[0],2)
        Data.Current3 = round(unpack('>f', pack('>HH', r[10], r[11]))[0],2)
        Data.Power1 = round(unpack('>f', pack('>HH', r[12], r[13]))[0],2)
        Data.Power2 = round(unpack('>f', pack('>HH', r[14], r[15]))[0],2)
        Data.Power3 = round(unpack('>f', pack('>HH', r[16], r[17]))[0],2)
        Data.Volt_Amps1 = round(unpack('>f', pack('>HH', r[18], r[19]))[0],2)
        Data.Volt_Amps2 = round(unpack('>f', pack('>HH', r[20], r[21]))[0],2)
        Data.Volt_Amps3 = round(unpack('>f', pack('>HH', r[22], r[23]))[0],2)
        Data.Volt_Amps_React1 = round(unpack('>f', pack('>HH', r[24], r[25]))[0],2)
        Data.Volt_Amps_React2 = round(unpack('>f', pack('>HH', r[26], r[27]))[0],2)
        Data.Volt_Amps_React3 = round(unpack('>f', pack('>HH', r[28], r[29]))[0],2)
        Data.Power_Factor1 = round(unpack('>f', pack('>HH', r[30], r[31]))[0],2)
        Data.Power_Factor2 = round(unpack('>f', pack('>HH', r[32], r[33]))[0],2)
        Data.Power_Factor3 = round(unpack('>f', pack('>HH', r[34], r[35]))[0],2)
        Data.Frequency = round(unpack('>f', pack('>HH', r[58], r[59]))[0],2)
        clear()
```

method ที่ใช้สั่ง while loop ให้อ่านค่า Modbus แล้วแปลงเป็น floating

```python
        clear()
        print('Voltage Phase1 : %f Vac' % Data.V1)
        print('Voltage Phase2 : %f Vac' % Data.V2)
        print('Voltage Phase3 : %f Vac' % Data.V3)
        print('Current Phase1 : %f Amps' % Data.Current1)
        print('Current Phase2 : %f Amps' % Data.Current2)
        print('Current Phase3 : %f Amps' % Data.Current3)
        print('Power Phase1 : %f Watts' % Data.Power1)
        print('Power Phase2 : %f Watts' % Data.Power2)
        print('Power Phase3 : %f Watts' % Data.Power3)
        print('Volt Amps Phase1 : %f VA' % Data.Volt_Amps1)
        print('Volt Amps Phase2 : %f VA' % Data.Volt_Amps2)
        print('Volt Amps Phase3 : %f VA' % Data.Volt_Amps3)
        print('Volt Amps Reactive Phase1 : %f VAr' % Data.Volt_Amps_React1)
        print('Volt Amps Reactive Phase2 : %f VAr' % Data.Volt_Amps_React2)
        print('Volt Amps Reactive Phase3 : %f VAr' % Data.Volt_Amps_React3)
        print('Power Factor Phase1 : %f None' % Data.Power_Factor1)
        print('Power Factor Phase2 : %f None' % Data.Power_Factor2)
        print('Power Factor Phase3 : %f None' % Data.Power_Factor3)
        print('Frequency of supply voltages : %f Hz' % Data.Frequency)
        time.sleep(5)
    client.close()
```

คำสั่งแสดงผลลัพธ์ที่อ่านค่าModbus ผ่าน command line




I. เป็นส่วนที่โหลด Library ที่จะนำมาใช้ในชุดคำสั่งต่าง ๆ มีดังนี้

1) 
```
import os
import struct
from struct import *
```

เป็น Library ของชุดคำสั่งในการแปลงข้อมูลเป็น String

2) 
```
import pymodbus
```

เป็น Library ของชุดคำสั่งในการอ่านค่าโปรโตคอลมอดบัส

3) `import sqlite3`

เป็น Library ของชุดคำสั่ง SQLite Studio ซึ่งใช้ทำฐานข้อมูล

4) `import time`

`from time import gmtime, strftime`

เป็น Library ของชุดคำสั่งในการอ่านค่าเวลา

II. จะประกอบด้วย Library 2 ชุดคือ

1)
```
from pymodbus.constants import Endian
from pymodbus.payload import BinaryPayloadDecoder
from pymodbus.payload import BinaryPayloadBuilder
from pymodbus.client.sync import ModbusSerialClient as ModbusClient
import logging
```

เป็น Library ในการใช้ชุดคำสั่งแปลงค่ามอดบัสเป็น Floating

2) `import threading`

เป็น Library ในการใช้ชุดคำสั่งการทำ Threading ซึ่งจะประกอบด้วยชุดคำสั่งต่อไปนี้

```
def Process1():
    .
    .
    .
T1 = threading.Thread(name='Process1', target=Process1)
```

III. เป็นคำสั่งในการ Connect กับฐานข้อมูลโดยจะประกอบไปด้วยชุดคำสั่งดังนี้

```
conn = sqlite3.connect('./datameter')
c = conn.cursor()
    .
    .
    .
execsql = format("insert into meter values ('%s',%.2f)" %(t,Data.V1,...))
c.execute(execsql)
conn.commit()
conn.close()
```

IV. เป็นชุดคำสั่ง Connect กับโมดบัส อาร์ทียู ต้องตั้งค่าให้ตรงกับในเพาเวอร์มิเตอร์

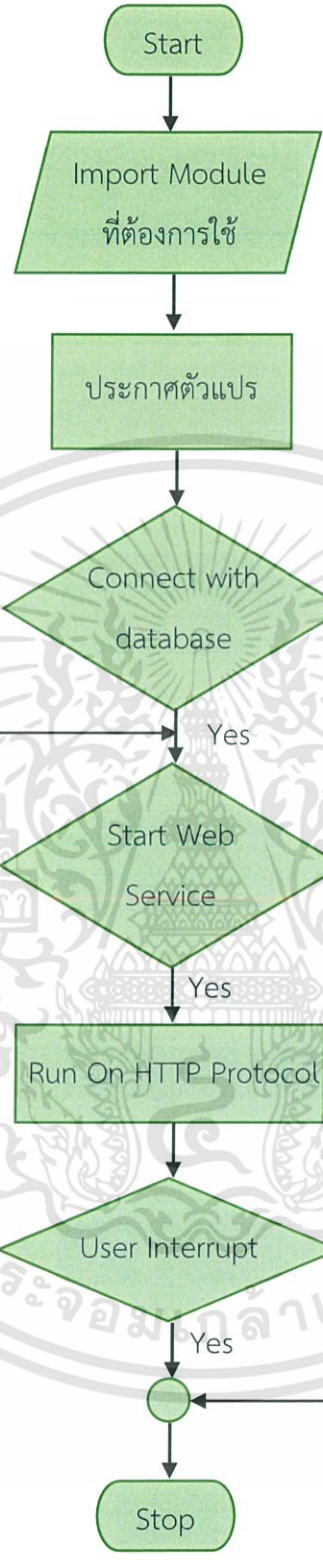
V. Format ในการ Insert ค่ามอดบัสที่แปลงเป็น Floating แล้วลงในตารางของฐานข้อมูล

VI. เป็นชุดคำสั่งที่ใช้ในการสั่งงานแบบ Threading

3.4.2 การส่งผ่านฐานข้อมูลโดยใช้เว็บเซอร์วิสผ่าน HTTP Protocol

เว็บเซอร์วิส คือ ระบบซอฟต์แวร์ที่ออกแบบมา เพื่อสนับสนุนการแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์ผ่านระบบเครือข่าย โดยที่ภาษาที่ใช้ในการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ คือ XML เว็บเซอร์วิสมีอินเทอร์เฟส ที่ใช้อธิบายรูปแบบข้อมูลที่เครื่องคอมพิวเตอร์ประมวลผลได้ ลักษณะการให้บริการของเว็บเซอร์วิสนั้นจะถูกเรียกใช้งานจาก Application อื่น ๆ ในรูปแบบ RPC (Remote Procedure Call) ซึ่งการให้บริการจะมีเอกสารที่อธิบายคุณสมบัติของบริการกำกับไว้ โดยภาษาที่ถูกใช้เป็นสื่อในการแลกเปลี่ยนคือ XML ทำให้เราสามารถเรียกใช้ Component ใด ๆ ก็ได้ในระบบ หรือ Platform ใด ๆ ก็ได้บน HTTP Protocol ซึ่งเป็นโปรโตคอลสำหรับ World Wide Web หรืออินเทอร์เน็ต อันเป็นช่องทางที่ได้รับการยอมรับทั่วโลกในการติดต่อสื่อสารกันระหว่าง Application กับ Application ในปัจจุบัน

การแชร์ข้อมูลโดยใช้เว็บเซอร์วิสสามารถนำมาเขียนเป็น Flowchart ได้ดังนี้

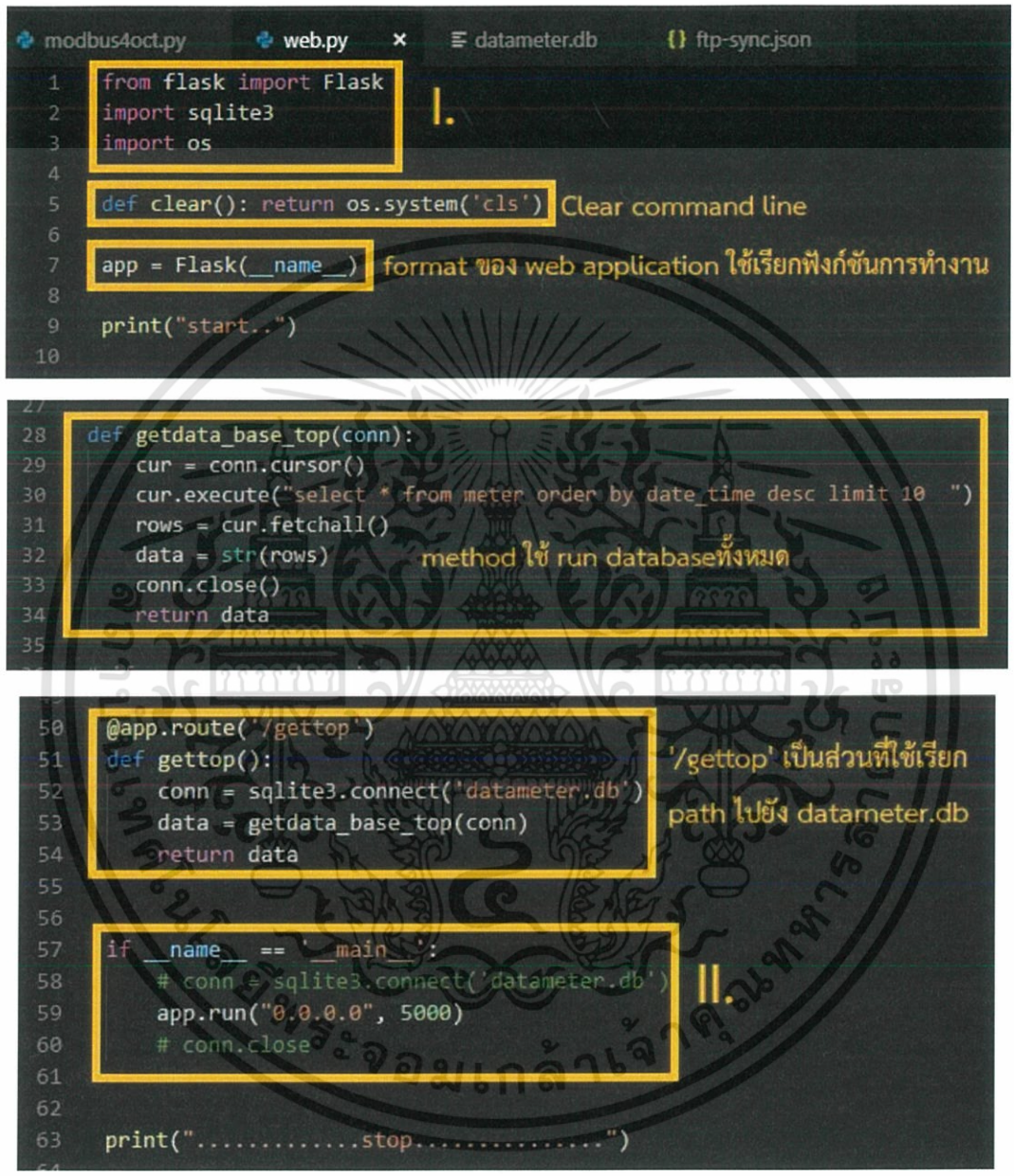



**ภาพที่ 3.7** Flowchart การแชร์ฐานข้อมูลโดยใช้เว็บเซอร์วิส

ส่วนโปรแกรม Visual Studio Code ที่เขียนโค้ดเว็บเซอร์วิสมีขั้นตอนดังต่อไปนี้

1. สร้าง File ใช้ชื่อว่า web.py ไว้ใน Folder เดียวกับ File ที่เขียนโค้ดมอดบัส

2. ส่วนนี้จะเป็นโค้ดที่แชร์ฐานข้อมูลโดยใช้เว็บเซอร์วิส




```
from flask import Flask
import sqlite3
import os

def clear(): return os.system('cls')

app = Flask(__name__)

print("start..")
```

```
def getdata_base_top(conn):
    cur = conn.cursor()
    cur.execute("select * from meter order by date_time desc limit 10  ")
    rows = cur.fetchall()
    data = str(rows)
    conn.close()
    return data
```

```
@app.route('/gettop')
def gettop():
    conn = sqlite3.connect('datameter.db')
    data = getdata_base_top(conn)
    return data


if __name__ == '__main__':
    # conn = sqlite3.connect('datameter.db')
    app.run("0.0.0.0", 5000)
    # conn.close


print(".............stop.............")
```

I. เป็นส่วนที่โหลด Library ที่ใช้ในชุดคำสั่งต่าง ๆ มีดังนี้

1) from flask import Flask

เป็น Library ของชุดคำสั่ง Framework ทำ Web Application โดยจะมี Format ดังนี้

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
return "Hello World!"
```

2) import sqlite3

เป็น Library ของชุดคำสั่งของฐานข้อมูล

3) import os

เป็น Library ของชุดคำสั่งในการแปลงค่าเป็น String

II. เป็นส่วนที่ใช้ในการสั่งงาน Application

**3.5 ฟังก์ชันการทำงานของเครื่องบันทึกข้อมูลสำหรับการวัดพลังงานระยะไกล**

ฟังก์ชันการทำงานของโปรแกรมจะแบ่งเป็น 3 ส่วนด้วยกัน คือ

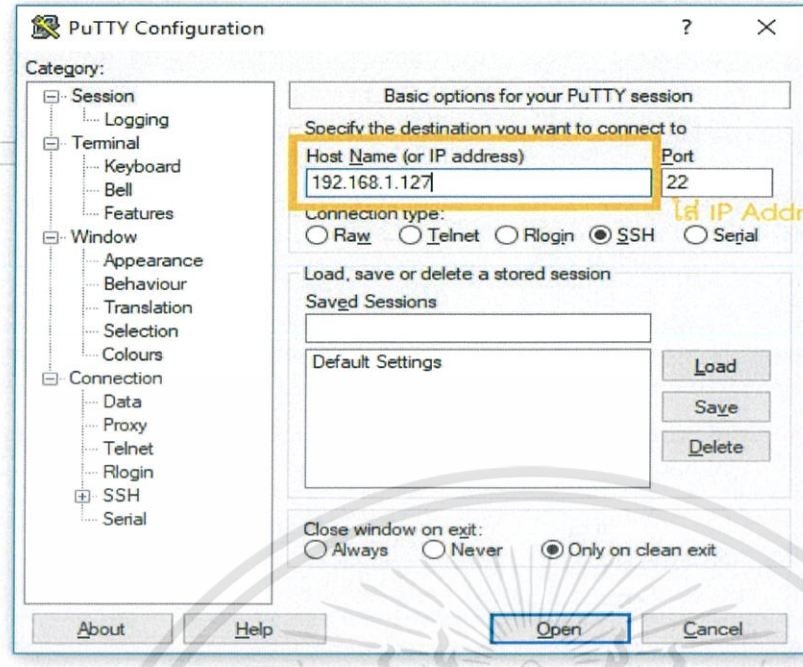
3.5.1 Run Modbus RTU Function

3.5.2 Run Database Function

3.5.3 Run Web Service Function

ขั้นแรกจะต้องโหลดโปรแกรม Putty (เป็นโปรแกรม Remote Server หรือ SSH (Secure Shell) พูดให้เข้าใจง่าย ๆ ก็คือ เราสามารถใช้โปรแกรมนี้ในการ สั่งงาน Server ด้วย Command Line โดยส่วนใหญ่แล้วจะใช้เชื่อมต่อไปยัง server ที่เป็น Linux) เพื่อที่จะใช้ป้อนคำสั่งบน Command Line ในการสั่งงานโปรแกรม ซึ่งสามารถโหลดได้ที่ https://www.putty.org/

เปิดโปรแกรม Putty แล้วใส่ IP Address ของบอร์ด ASUS Tinker แล้วกดปุ่ม Open ดังรูป

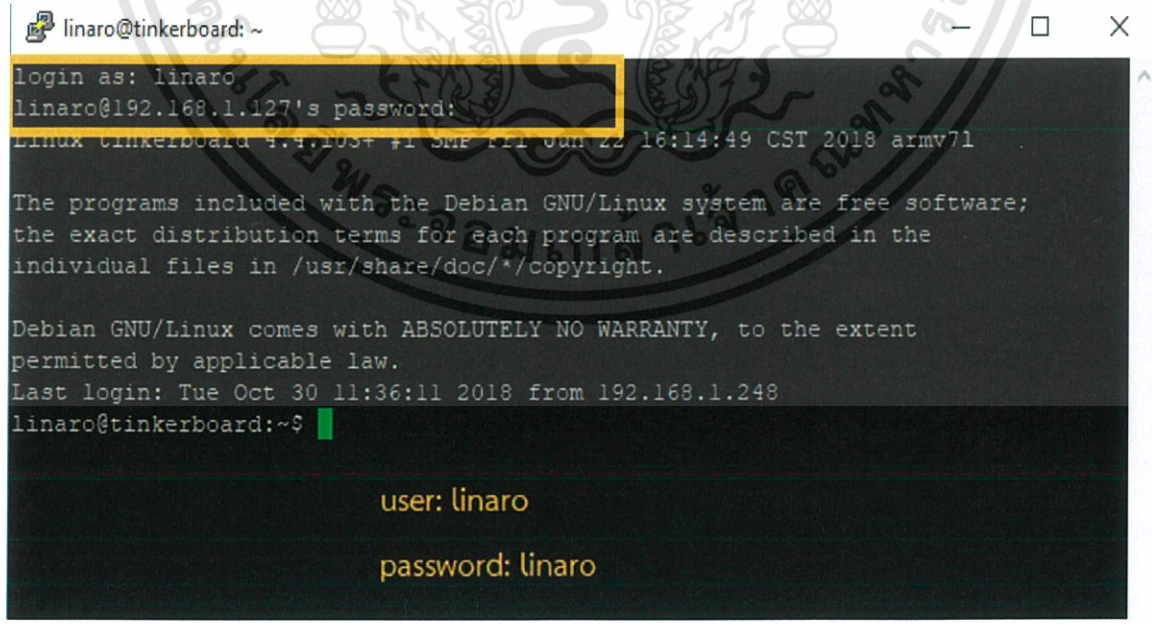



**ภาพที่ 3.8** โปรแกรมที่ใช้ Remote ไปยังบอร์ด ASUS Tinker

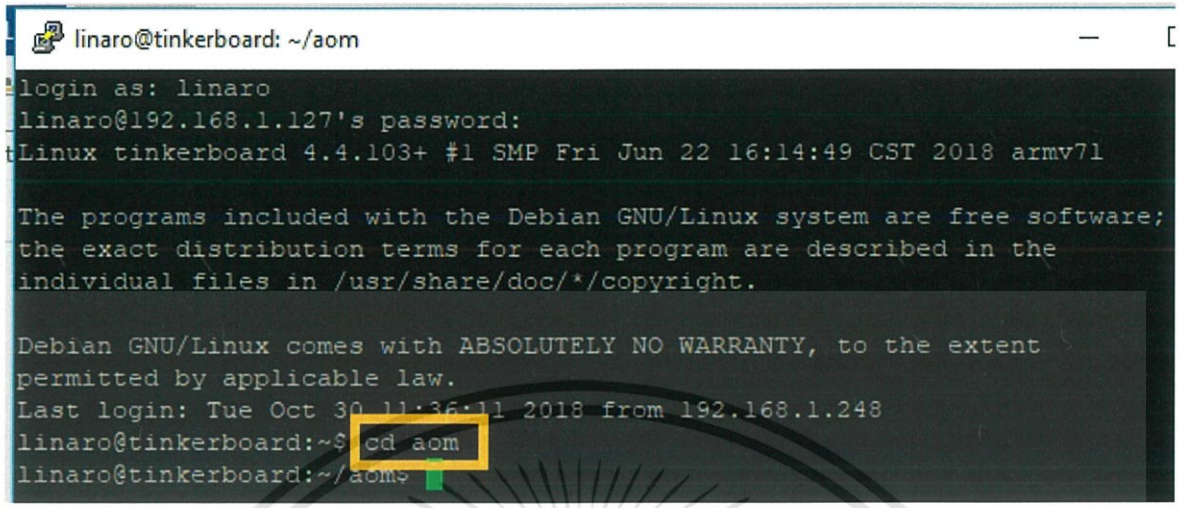
3.5.1 Run Modbus RTU Function

เมื่อ ใส่ IP Address ในโปรแกรม Putty แล้วกด Open จะเปิดหน้า Command Line ให้อัตโนมัติ หลังจากนั้นทำตามขั้นตอนต่อไปนี้

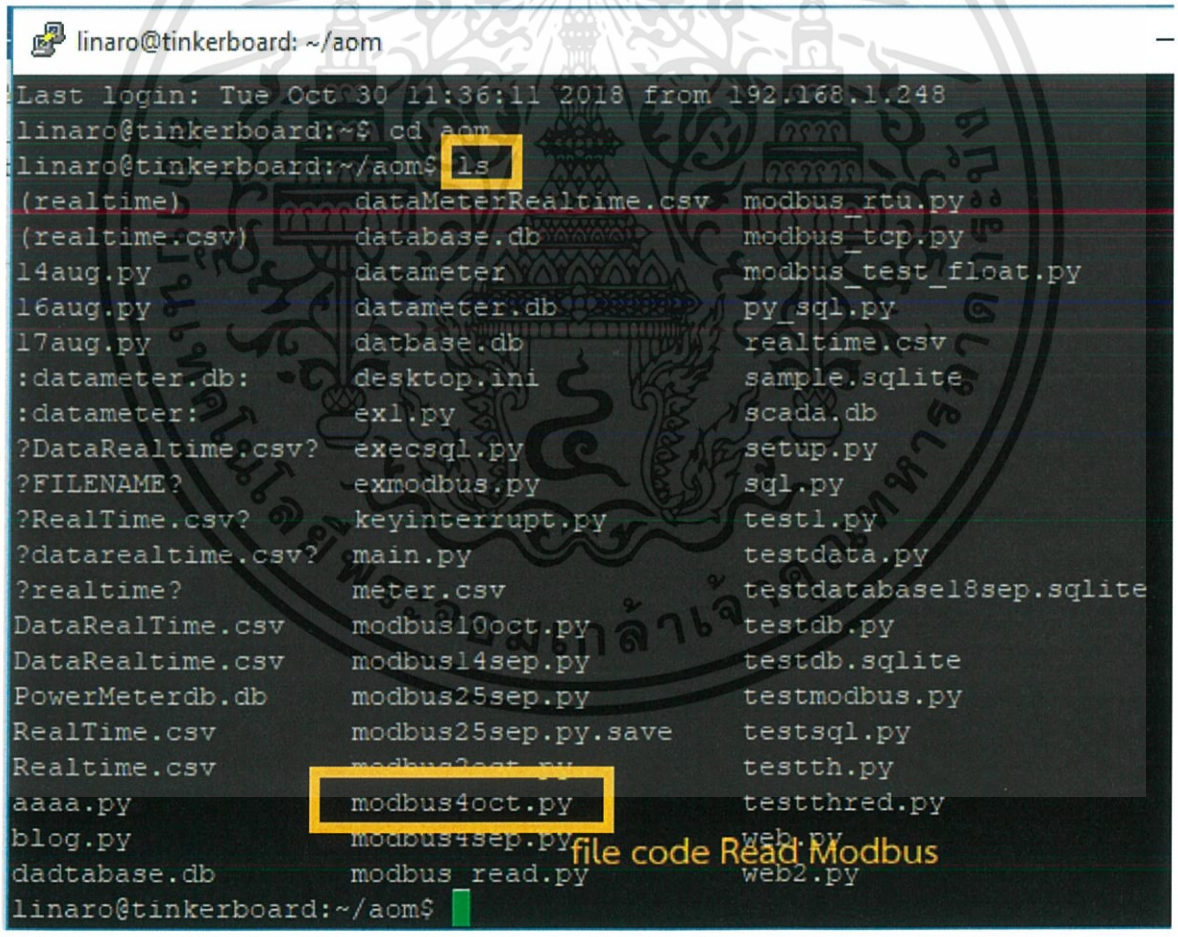
1. Login ใช้ User: linaro และ Password : linaro



2. พิมพ์คำสั่ง cd aom (ชื่อ Folder) เพื่อเปิด Folder ที่ได้เก็บไฟล์งานเอาไว้



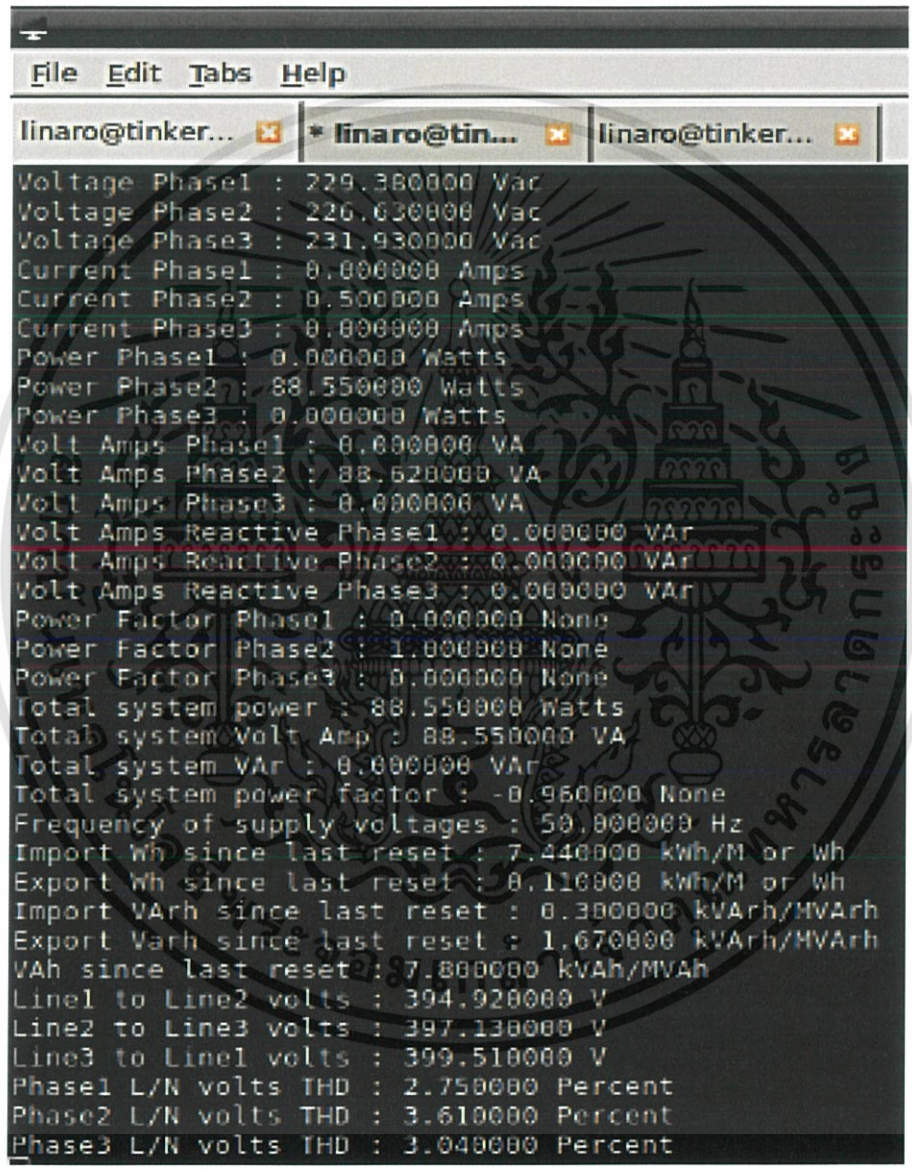

3. พิมพ์คำสั่ง ls เพื่อเปิด List File งานที่มีทั้งหมด



4. พิมพ์คำสั่ง sudo python modbus4oct.py (ชื่อไฟล์ที่เขียนโค้ด) เพื่อใช้ Run Modbus Function

```
modbus4sep.py          web.py
modbus_read.py         web2.py
ard:~/aom$ sudo python modbus4oct.py
```

5. ผลลัพธ์จะแสดงบน Command Line

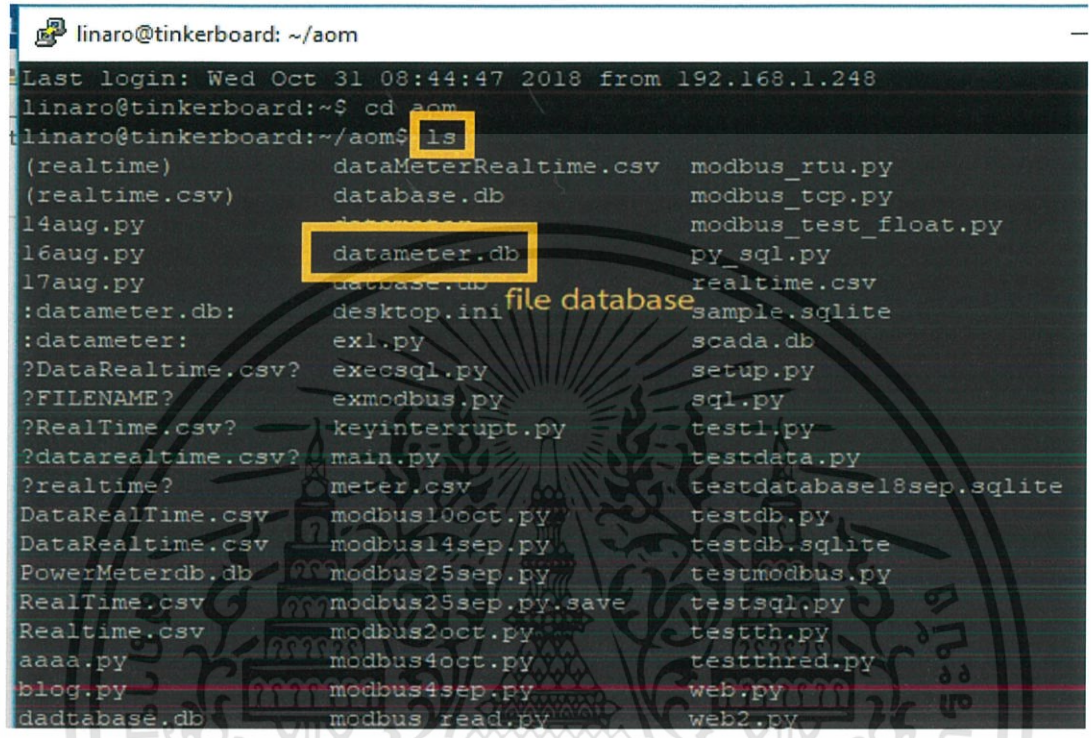



**ภาพที่ 3.9** ผลลัพธ์แต่ละพารามิเตอร์บน Command Line

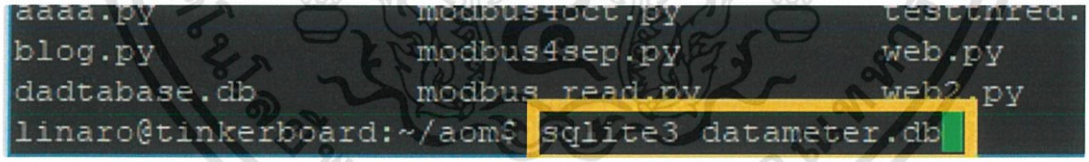
3.5.2 Run Database Function

เปิด Command Line ขึ้นมาอีก Tab เพื่อที่จะใช้ Run Database Function ซึ่งมีขั้นตอนดังนี้

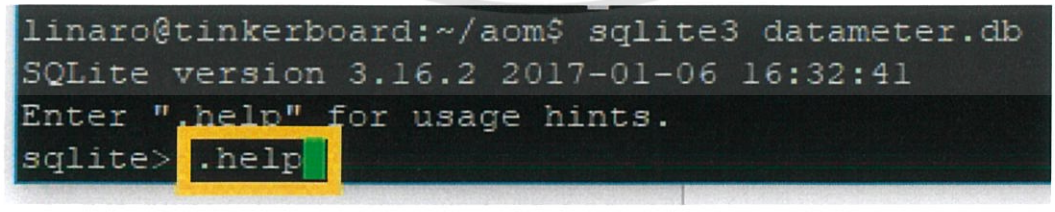
1. พิมพ์คำสั่ง ls เพื่อเปิด List File งานทั้งหมด




2. พิมพ์คำสั่ง sqlite3 datameter.db (ชื่อไฟล์ฐานข้อมูล)




3. พิมพ์คำสั่ง .help



4. พิมพ์คำสั่ง .table เพื่อเรียกตารางที่มีอยู่ใน datameter.db ซึ่งในที่นี่จะแสดงขึ้นมา 2 ตาราง คือ meter และmeterhistory

```
.width NUM1 NUM2 ...   Set column widths for "column" mod
                         Negative values right-justify
sqlite> .table
meter          meterhistory
sqlite>
```

5. พิมพ์คำสั่ง select * from meter; (meter คือชื่อตาราง) เพื่อแสดงค่าที่บันทึกลงฐานข้อมูลบน Command Line

```
meter          meterhistory
sqlite> select * from meter;
```

6. ผลลัพธ์จะแสดงผ่าน Command Line ดังตัวอย่างภาพต่อไปนี้

```
2018-11-20 10:48:17|229.04|231.05|232.61|0.0|0.48|0.0|0.0|81.01|0.0|0.0|81.07|0.0|0.0|0.0|0.0|0.0|
1.0|0.0|81.01|81.01|0.0|-0.97|50.01|7.05|0.11|0.28|1.65|7.41|398.46|401.54|399.8|2.75|3.26|2.94
```

**ภาพที่ 3.10** ตัวอย่างผลลัพธ์ที่บันทึกในฐานข้อมูลแสดงผ่าน Command Line

7. สามารถดูผลลัพธ์ผ่านโปรแกรม SQLite Studio ได้ดังตัวอย่างภาพต่อไปนี้

| date time | V1 | V2 | V3 | Current1 | Current2 | Current3 |
|---|---|---|---|---|---|---|
| 2018-11-19 16:05:40 | 229.04 | 231.21 | 233.6 | 0 | 0.3 | 0 |

| Power1 | Power2 | Power3 | Volt Amps1 | Volt Amps2 | Volt Amps3 |
|---|---|---|---|---|---|
| 0 | 40.3 | 0 | 0 | 43.81 | 0 |

| Volt Amps React1 | Volt Amps React2 | Volt Amps React3 |
|---|---|---|
| 0 | 0 | 0 |

| Power Factor1 | Power Factor2 | Power Factor3 | Tol power | Tol VoltAmp | Tol VAr |
|---|---|---|---|---|---|
| 0 | 0.92 | 0 | 40.3 | 40.3 | 0 |

| Tol power factor | Frequency | Import Wh | Export Wh | Import VArh | Export VArh |
|---|---|---|---|---|---|
| -0.78 | 49.97 | 7.31 | 0.11 | 0.3 | 1.67 |

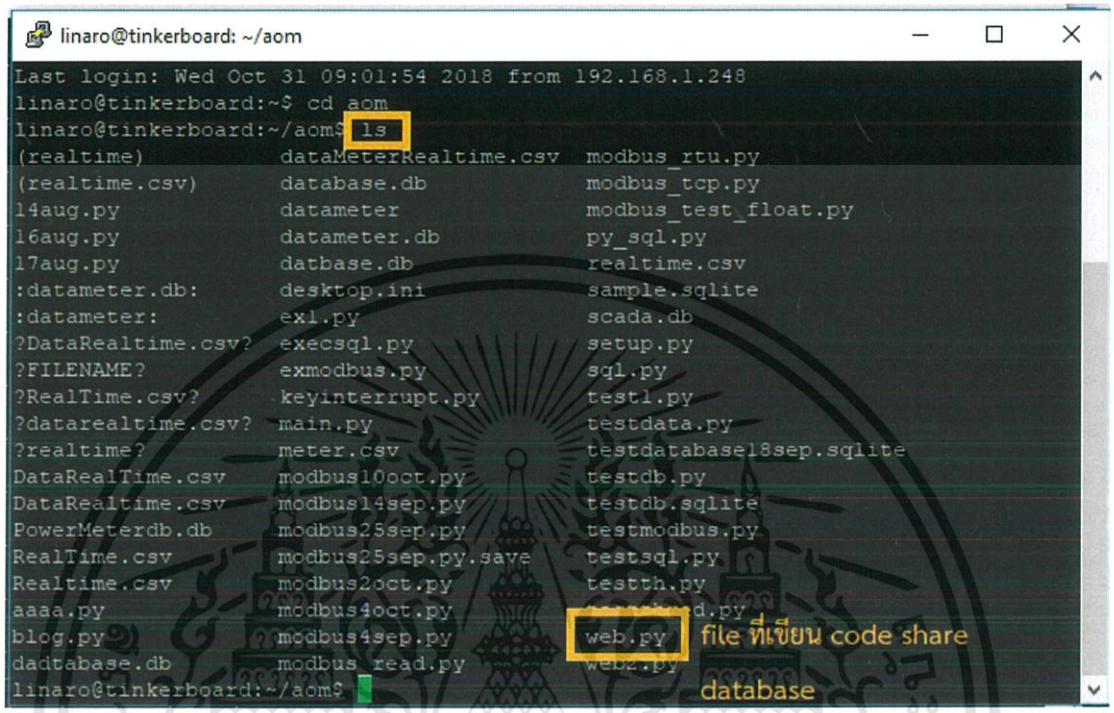
| VAh last reset | L1 to L2 | L2 to L3 | L3 to L1 | P1 LN THD | P2 LN THD | P3 LN THD |
|---|---|---|---|---|---|---|
| 7.68 | 398.59 | 402.53 | 400.66 | 2.5 | 3.4 | 2.78 |

**ภาพที่ 3.11** ตัวอย่างค่าพารามิเตอร์ที่บันทึกในฐานข้อมูลแสดงผ่านโปรแกรม SQLite Studio

3.5.3 Run Web Service Function

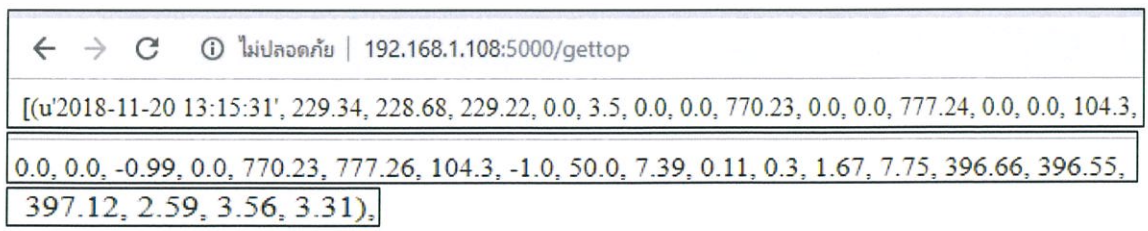
เปิด Command Line ขึ้นมาอีก Tab เพื่อ Run Web Service Function มีขั้นตอนดังนี้

1. พิมพ์คำสั่ง ls เพื่อเปิด List File งาน

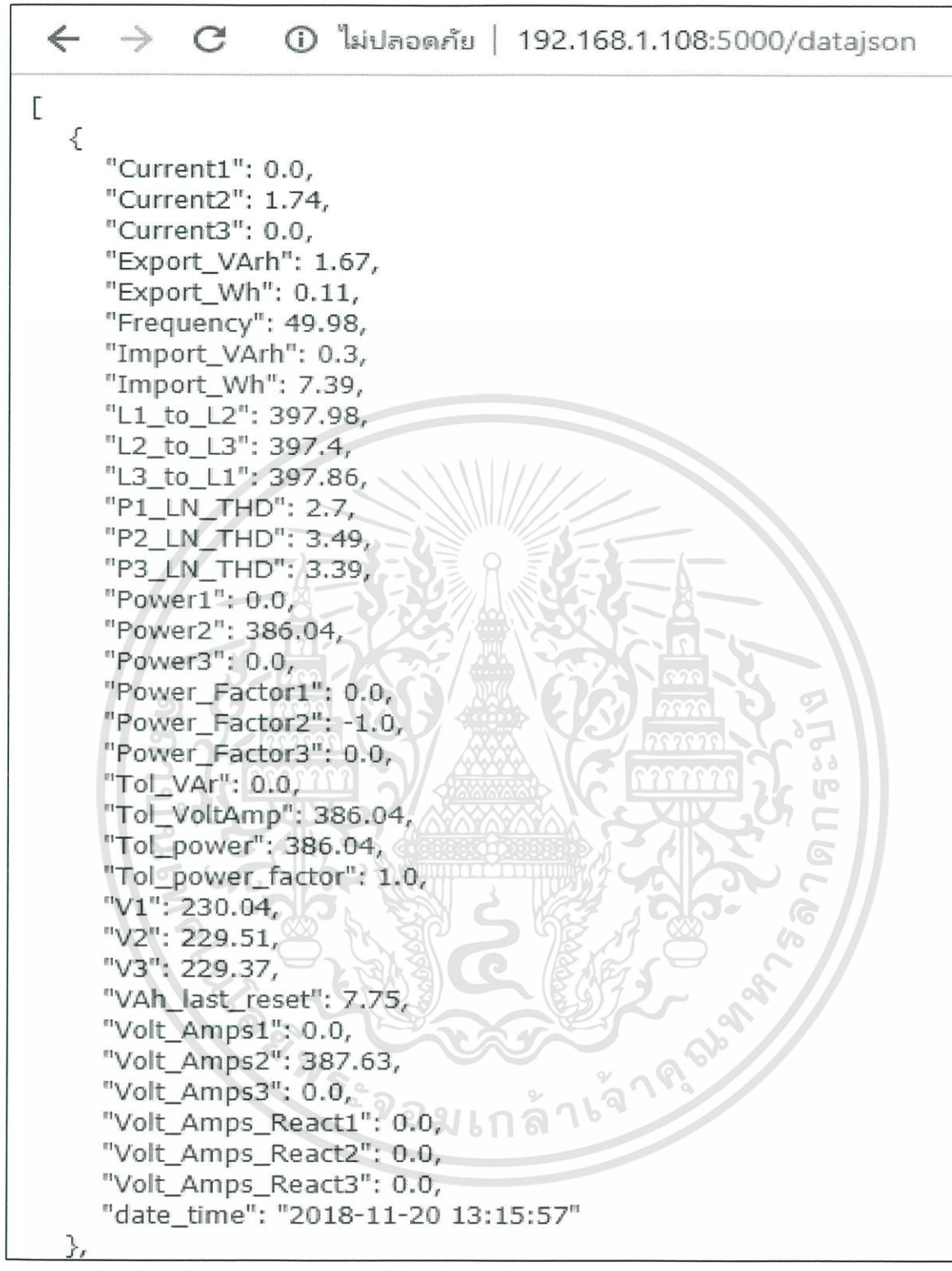



2. พิมพ์คำสั่ง sudo python web.py (คือชื่อไฟล์ที่เขียนโค้ด) เพื่อ Run Web Service

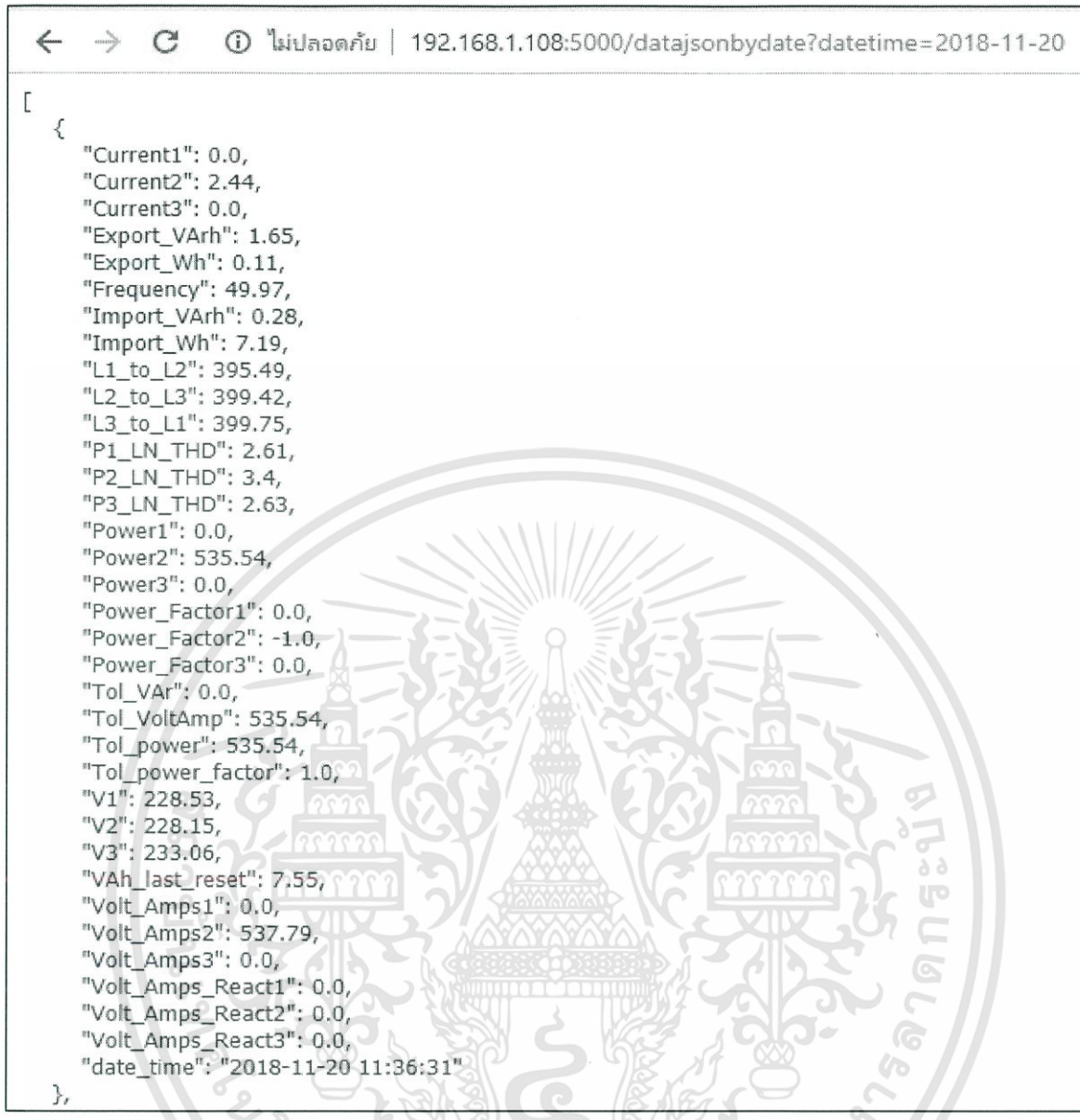

3. เปิด Web Browser แล้วใส่ IP Address ของบอร์ด ที่ช่อง Url เป็น 192.168.1.108:5000/gettop เพื่อแสดงค่าบนเว็บเซอร์วิส สำหรับซอฟต์แวร์ LABView หรือใช้ IP Address 192.168.1.108:5000/datajson และ 192.168.1.108:5000/datajsonbydate สำหรับหน้าเว็บแบบเรียลไทม์และแบบย้อนหลังได้ตามลำดับ




**ภาพที่ 3.12** ตัวอย่างผลลัพธ์บนเว็บเซอร์วิสที่ใช้กับซอฟต์แวร์ LabVIEW




**ภาพที่ 3.13** ตัวอย่างผลลัพธ์บนเว็บเซอร์วิส ที่ใช้กับเว็บแบบเรียลไทม์




ภาพที่ 3.14 ผลลัพธ์บนเว็บเซอร์วิสที่ใช้กับเว็บแบบย้อนหลัง

# บทที่ 4
# ผลการทดลองเครื่องบันทึกข้อมูลสำหรับการวัดพลังงานระยะไกล

## 4.1 กล่าวนำ

ผลการดำเนินงานโครงงานเครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกล ในส่วนของการเขียนโปรแกรมที่ใช้ภาษา Python สามารถแบ่งได้เป็น 3 ส่วนดังหัวข้อต่อไปนี้

4.2 ทดสอบคำสั่งการอ่านค่าพารามิเตอร์จากเพาเวอร์มิเตอร์

4.3 ทดสอบการบันทึกค่าพารามิเตอร์ลงฐานข้อมูล

4.4 ทดสอบการจัดเก็บข้อมูลสำหรับการร้องขอข้อมูลเพื่อการแสดงผลด้วยโปรแกรมซอฟต์แวร์ LabVIEW และหน้าเว็บ มีผลการทดลองดังต่อไปนี้

## 4.2 การทดสอบคำสั่งการอ่านค่าพารามิเตอร์จากเพาเวอร์มิเตอร์

สำหรับส่วนนี้เป็นส่วนแรกของการเขียนโปรแกรมเพื่ออ่านค่าพารามิเตอร์ ของโปรโตคอลมอดบัสจากเพาเวอร์มิเตอร์ โดยจะมีการแปลงจาก Hex to Floating เพื่อให้คนทั่วไปสามารถอ่านค่าได้

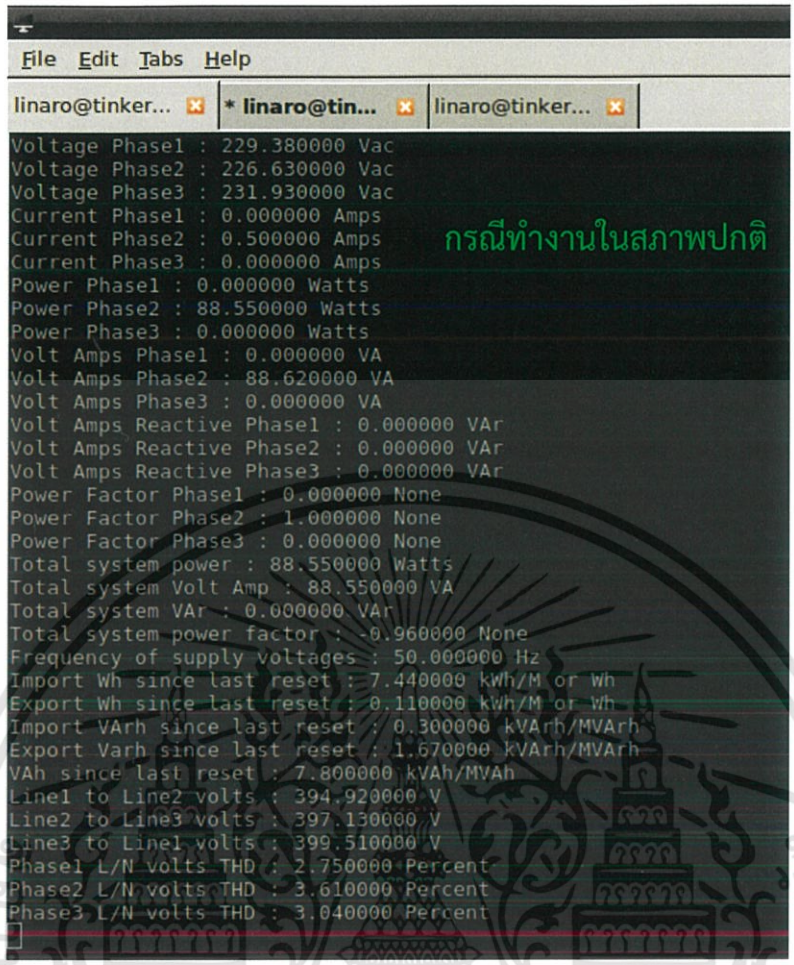
4.2.1 วิธีการทดลอง

ในส่วนแรกนี้มีจุดมุ่งหมายเพื่ออ่านค่าพารามิเตอร์ของโปรโตคอลมอดบัสได้โดยแปลงค่าจาก Hex to Floating

ขั้นแรกต้องโหลด Library ที่ใช้ในการเขียนโปรแกรมอ่านค่าโปรโตคอลมอดบัสและต้องตั้งค่า Stop bit, Parity และ Buadrate ในเพาเวอร์มิเตอร์ให้ตรงกับที่เขียนโปรแกรมด้วย และส่วนต่อไปเป็นการเขียนโปรแกรมในการอ่านค่าพารามิเตอร์ ซึ่งในส่วนนี้จะมีการเขียนโปรแกรมให้แปลงค่าจาก Hex to Floating เพื่อให้คนทั่วไปสามารถอ่านค่าแล้วเข้าใจได้ ซึ่งจะสั่งงานโปรแกรมผ่าน Command Line โดยใช้คำสั่ง sudo python modbusread.py

4.2.2 ผลการทดลอง

จากการดำเนินการข้างต้นสามารถอ่านค่าพารามิเตอร์จากมอดบัสโปรโตคอลได้ โดยค่าที่อ่านได้จะเป็น Floating แต่ถ้าเกิดการ Loss Connection จะมีการแจ้งเตือนโดยมีข้อความว่า “Unexpected error” ด้วย ดังรูปต่อไปนี้




**ภาพที่ 4.1** ค่าแต่ละพารามิเตอร์ที่อ่านได้จากเพาเวอร์มิเตอร์ในกรณีทำงานปกติ


**ภาพที่ 4.2** ค่าแต่ละพารามิเตอร์ที่อ่านได้จากเพาเวอร์มิเตอร์ในกรณี Loss Connection

### 4.3 การทดสอบการบันทึกค่าพารามิเตอร์ลงฐานข้อมูล

เป็นส่วนที่สองซึ่งจะมีการบันทึกค่าพารามิเตอร์ลงฐานข้อมูล โดยจะมีการเขียนโปรแกรมให้ connect กับโปรแกรมฐานข้อมูล ในที่นี้จะเลือกใช้โปรแกรม SQLite Studio ในการบันทึกค่า ซึ่งค่าทั้งหมดจะบันทึกลง Micro SD Card

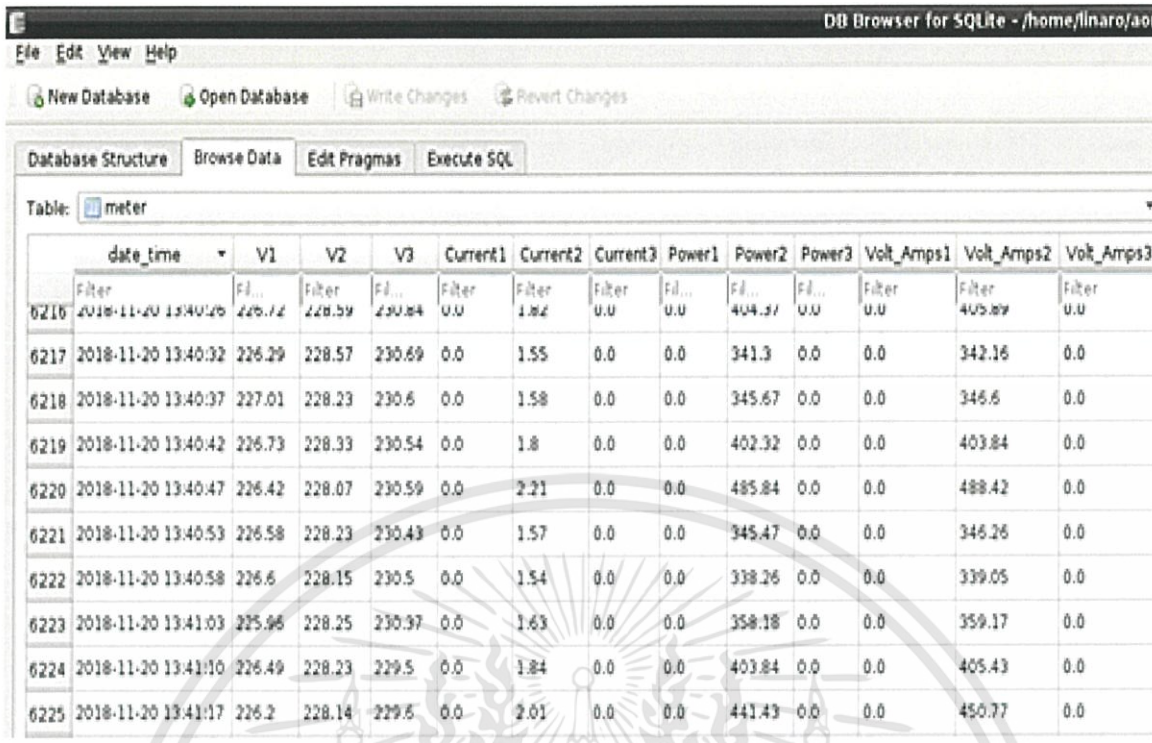
4.3.1 วิธีการทดลอง

ในส่วนนี้จุดประสงค์จะเป็นการเก็บบันทึกค่าพารามิเตอร์ที่แปลงค่ามาเป็น Floating ในขั้นตอนที่แล้วโดยอ่านได้จากเพาเวอร์มิเตอร์ บันทึกลงโปรแกรม SQLite Studio และสามารถ Query ค่าต่าง ๆ เพื่อสามารถใช้ Report ผลลัพธ์ที่จะนำไปวิเคราะห์ต่อไปได้

โดยสร้างตารางเก็บค่าพารามิเตอร์ ทั้งหมด 34 ค่า และอีก 1 ค่า จะเป็น Date Time ในโปรแกรม SQLite Studio แล้วในส่วนของโปรแกรม Visual Studio Code จะต้องโหลด library ที่ใช้ connect กับ database แล้วเขียนโปรแกรมภาษาPython ให้เก็บบันทึกค่าลง Database โดยจะเก็บค่าทุก ๆ 5 วินาที และสามารถเก็บข้อมูลได้นานสุด 1 ปี ซึ่งจะสั่ง run program ผ่านcommand line โดยใช้คำสั่งตามลำดับต่อไปนี้ 1) sqlite3 datameter.db 2) .help 3) .table 4) select * from meter; หรือสามารถเปิดโปรแกรม SQLite Studio เพื่อสั่งการทำงานของการบันทึกค่าได้เลยเช่นกัน

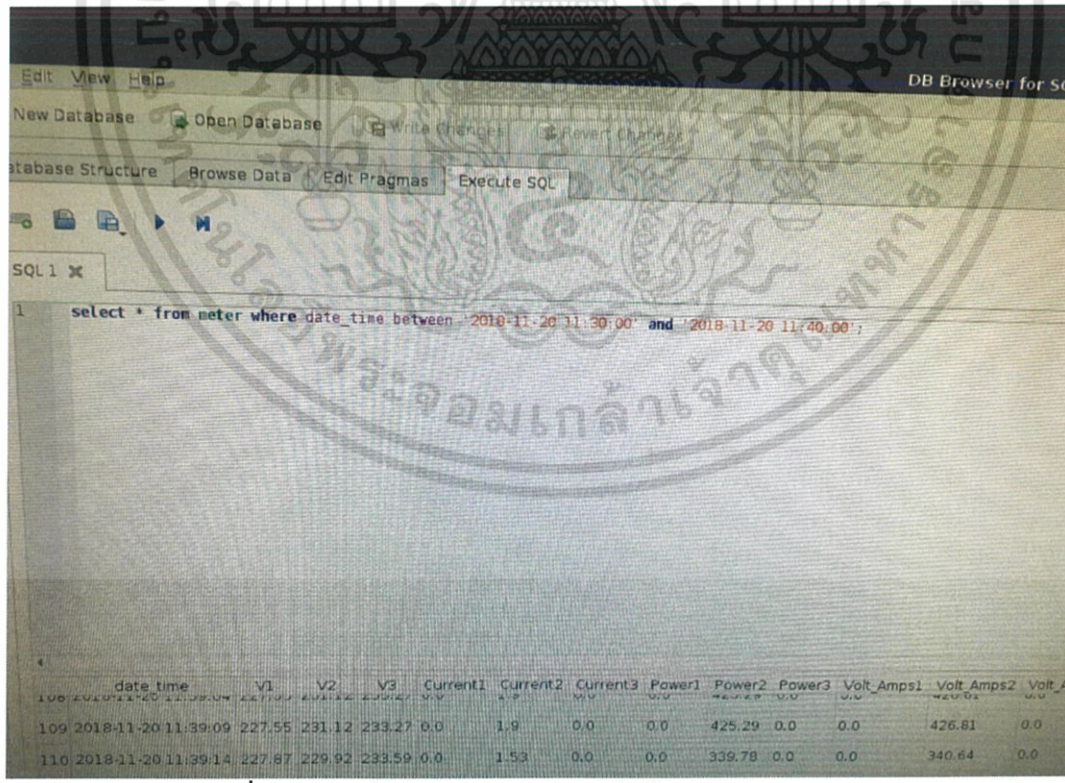
4.3.2 ผลการทดลอง

จากการดำเนินงานในขั้นตอนนี้สามารถบันทึกค่า Parameter และ Date Time ได้ โดยจะบันทึกค่าทุก ๆ 5 วินาที และสามารถ Query ผ่าน Command Line หรือ ผ่านโปรแกรม SQLite Studio ได้เช่นกัน ถ้าเกิดการ Loss Connection จากขั้นตอนการ Read Modbus ในตอนแรก ในส่วนของการบันทึกค่าพารามิเตอร์ลงฐานข้อมูลก็จะไม่ทำงานไปด้วยเช่นกัน ผลลัพธ์แสดงดังภาพต่อไปนี้




| | date_time | V1 | V2 | V3 | Current1 | Current2 | Current3 | Power1 | Power2 | Power3 | Volt_Amps1 | Volt_Amps2 | Volt_Amps3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6216 | 2018-11-20 13:40:26 | 226.72 | 228.59 | 230.84 | 0.0 | 1.82 | 0.0 | 0.0 | 404.37 | 0.0 | 0.0 | 405.89 | 0.0 |
| 6217 | 2018-11-20 13:40:32 | 226.29 | 228.57 | 230.69 | 0.0 | 1.55 | 0.0 | 0.0 | 341.3 | 0.0 | 0.0 | 342.16 | 0.0 |
| 6218 | 2018-11-20 13:40:37 | 227.01 | 228.23 | 230.6 | 0.0 | 1.58 | 0.0 | 0.0 | 345.67 | 0.0 | 0.0 | 346.6 | 0.0 |
| 6219 | 2018-11-20 13:40:42 | 226.73 | 228.33 | 230.54 | 0.0 | 1.8 | 0.0 | 0.0 | 402.32 | 0.0 | 0.0 | 403.84 | 0.0 |
| 6220 | 2018-11-20 13:40:47 | 226.42 | 228.07 | 230.59 | 0.0 | 2.21 | 0.0 | 0.0 | 485.84 | 0.0 | 0.0 | 488.42 | 0.0 |
| 6221 | 2018-11-20 13:40:53 | 226.58 | 228.23 | 230.43 | 0.0 | 1.57 | 0.0 | 0.0 | 345.47 | 0.0 | 0.0 | 346.26 | 0.0 |
| 6222 | 2018-11-20 13:40:58 | 226.6 | 228.15 | 230.5 | 0.0 | 1.54 | 0.0 | 0.0 | 338.26 | 0.0 | 0.0 | 339.05 | 0.0 |
| 6223 | 2018-11-20 13:41:03 | 225.96 | 228.25 | 230.37 | 0.0 | 1.63 | 0.0 | 0.0 | 358.18 | 0.0 | 0.0 | 359.17 | 0.0 |
| 6224 | 2018-11-20 13:41:10 | 226.49 | 228.23 | 229.5 | 0.0 | 1.84 | 0.0 | 0.0 | 403.84 | 0.0 | 0.0 | 405.43 | 0.0 |
| 6225 | 2018-11-20 13:41:17 | 226.2 | 228.14 | 229.6 | 0.0 | 2.01 | 0.0 | 0.0 | 441.43 | 0.0 | 0.0 | 450.77 | 0.0 |

**ภาพที่ 4.3** ตัวอย่างการบันทึกค่าพารามิเตอร์ลงโปรแกรม SQLite Studio




| | date_time | V1 | V2 | V3 | Current1 | Current2 | Current3 | Power1 | Power2 | Power3 | Volt_Amps1 | Volt_Amps2 | Volt_Amps3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 109 | 2018-11-20 11:39:09 | 227.55 | 231.12 | 233.27 | 0.0 | 1.9 | 0.0 | 0.0 | 425.29 | 0.0 | 0.0 | 426.81 | 0.0 |
| 110 | 2018-11-20 11:39:14 | 227.87 | 229.92 | 233.59 | 0.0 | 1.53 | 0.0 | 0.0 | 339.78 | 0.0 | 0.0 | 340.64 | 0.0 |

**ภาพที่ 4.4** ตัวอย่างแสดงการ Query ผ่านโปรแกรม SQLite Studio

```
sqlites select V1 from meter;
220.32
sqlites select V1 from meter;
223.52
```

**ภาพที่ 4.5** การ Query ผ่าน Command Line

## 4.4 การทดสอบการจัดเก็บข้อมูลสำหรับการร้องขอเพื่อการแสดงผลด้วย LabVIEW และหน้าเว็บ

ส่วนที่สามนี้เป็นการแชร์ค่าพารามิเตอร์บนเว็บเซอร์วิสเพื่อให้ซอฟต์แวร์ LabVIEW และเว็บดึงค่าไปแสดงได้บนหน้าจอ

4.4.1 วิธีการทดลอง

ในส่วนนี้มีจุดประสงค์ในการแชร์ค่า Parameter บนเว็บเซอร์วิสโดยการแชร์ข้อมูลจะแบ่งเป็น2 แบบ คือ 1) การส่งผ่านข้อมูลโดยใช้ภาษา Python สำหรับการร้องขอข้อมูลไปใช้ในโปรแกรมซอฟต์แวร์ LabVIEW 2) การส่งผ่านข้อมูลโดยใช้ภาษา JSON สำหรับการร้องขอข้อมูลไปใช้กับการแสดงค่าบนหน้าเว็บ ซึ่งทั้งสองแบบสามารถเชื่อมต่อได้ผ่าน Wi-Fi เท่านั้น

ในการแชร์ข้อมูลจะใช้ Flask ( Flask คือ Web Framework ที่เขียนขึ้นมาสำหรับ Python เพื่อใช้ร่วมกัน Webserver เช่น Apache และได้รับการยอมรับจาก Community We Pages ชั้นนำเช่น Pinterest, LinkedIn เป็นต้น โดย Flask ถูกเรียกว่า Micro Framework เพราะว่า มันไม่ต้องการเครื่องมือ หรือ Library อะไรมาก อีกทั้ง ไม่จำเป็นต้องมีฐานข้อมูล ด้วย แต่อย่างไรก็ตาม Flask ก็ยังรองรับการเพิ่ม extensions พิเศษได้ ถ้ามันรองรับ Flask) โดยจะต้องโหลด Library ที่จำเป็นในการใช้งาน เพื่อจะแชร์ฐานข้อมูลบนเว็บเซอร์วิสซึ่งในส่วนของการเขียนโปรแกรมจะแบ่งเป็น 2 ส่วน คือ

1. การแชร์โดยใช้ภาษา Python สำหรับการดึงค่าไปใช้ในโปรแกรมซอฟต์แวร์ LabVIEW ดังภาพต่อไปนี้

```python
@app.route('/gettop')
def gettop():
    conn = sqlite3.connect('datameter.db')
    data = getdata_base_top(conn)
    return data
```

**ภาพที่ 4.6** โค้ดแสดงผลลัพธ์แบบเรียลไทม์บนซอฟต์แวร์ LabVIEW

2. การส่งผ่านข้อมูลโดยใช้ภาษา JSON สำหรับการดึงค่าไปใช้กับการแสดงค่าบนหน้าเว็บแบบ real time ซึ่งในส่วนนี้สามารถแสดงในส่วนของ history ได้ด้วย ดังรูปต่อไปนี้

```python
@app.route("/datajson")
def getdatatop():
    # conn = sqlite3.connect('datameter.db')
    data = dataJson()
    resp = Response(str(data))
    resp.headers['Access-Control-Allow-Origin'] = '*'
    resp.headers['Content-Type'] = 'application/json'
    return resp
```

**ภาพที่ 4.7** โค้ดแสดงผลลัพธ์แบบเรียลไทม์บนหน้าเว็บ

```python
@app.route("/datajsonbydate")
def getdatabydate():
    # conn = sqlite3.connect('datameter.db')
    datetime = request.args.get('datetime')
    data = dataJsonByDate(datetime)
    resp = Response(str(data))
    resp.headers['Access-Control-Allow-Origin'] = '*'
    resp.headers['Content-Type'] = 'application/json'
    return resp
```

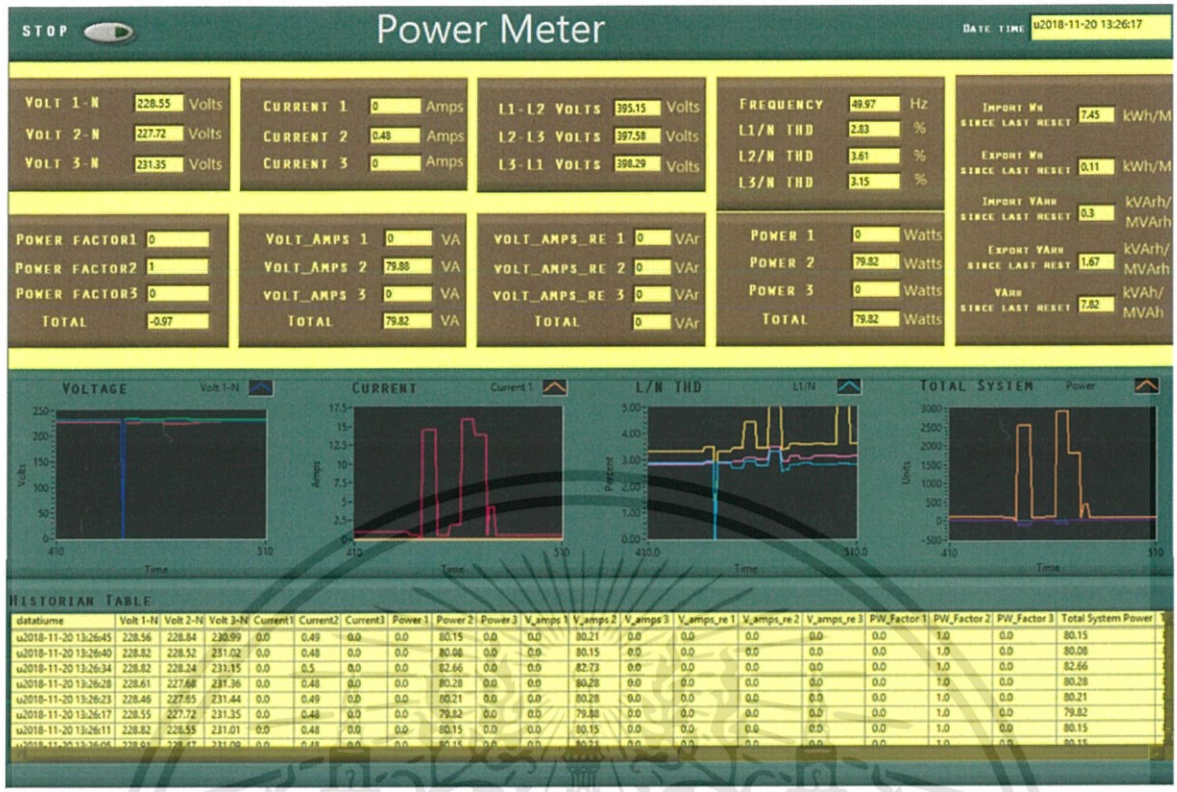
**ภาพที่ 4.8** โค้ดแสดงผลลัพธ์แบบย้อนหลังบนหน้าเว็บ

3. ในการสั่งงานโปรแกรมจะสั่งผ่าน command line โดยใช้คำสั่ง sudo python webjson.py

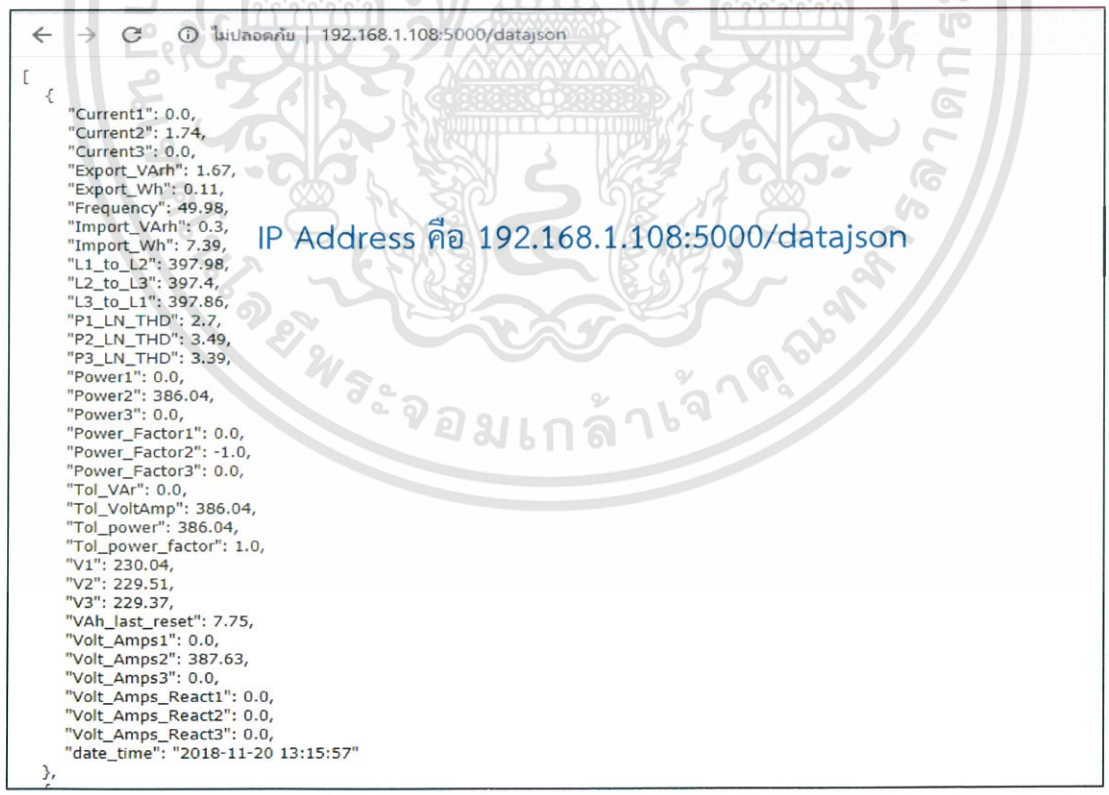
4.4.2 ผลการทดลอง

จากการดำเนินงานสารถแชร์ข้อมูลบนเว็บเซอร์วิสได้ทั้งในส่วนของ Python ซึ่งซอฟต์แวร์ LabVIEW สามารถดึงข้อมูลไปใช้ได้ในแบบเรียลไทม์ และในส่วนของ JSON ซึ่งเว็บสามารถดึงข้อมูลไปแสดงได้ทั้งแบบเรียลไทม์ และแบบย้อนหลังแสดงดังภาพต่อไปนี้

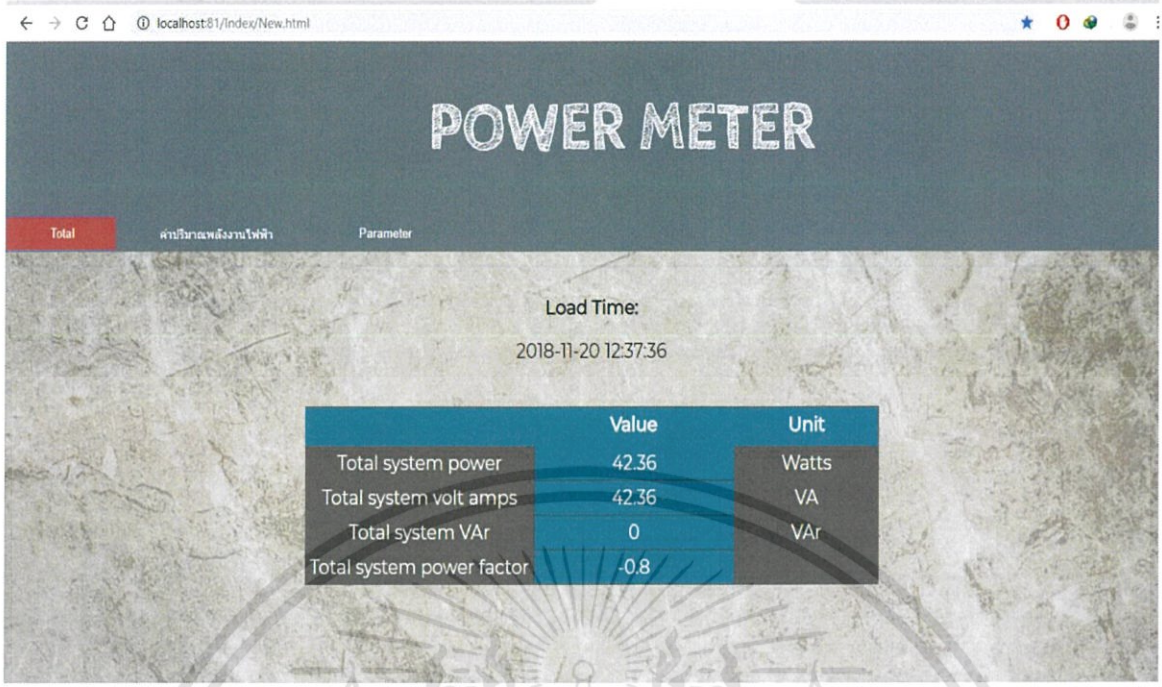

**ภาพที่ 4.9** ผลลัพธ์บนเว็บเซอร์วิสสำหรับนำไปใช้กับซอฟต์แวร์ LabVIEW

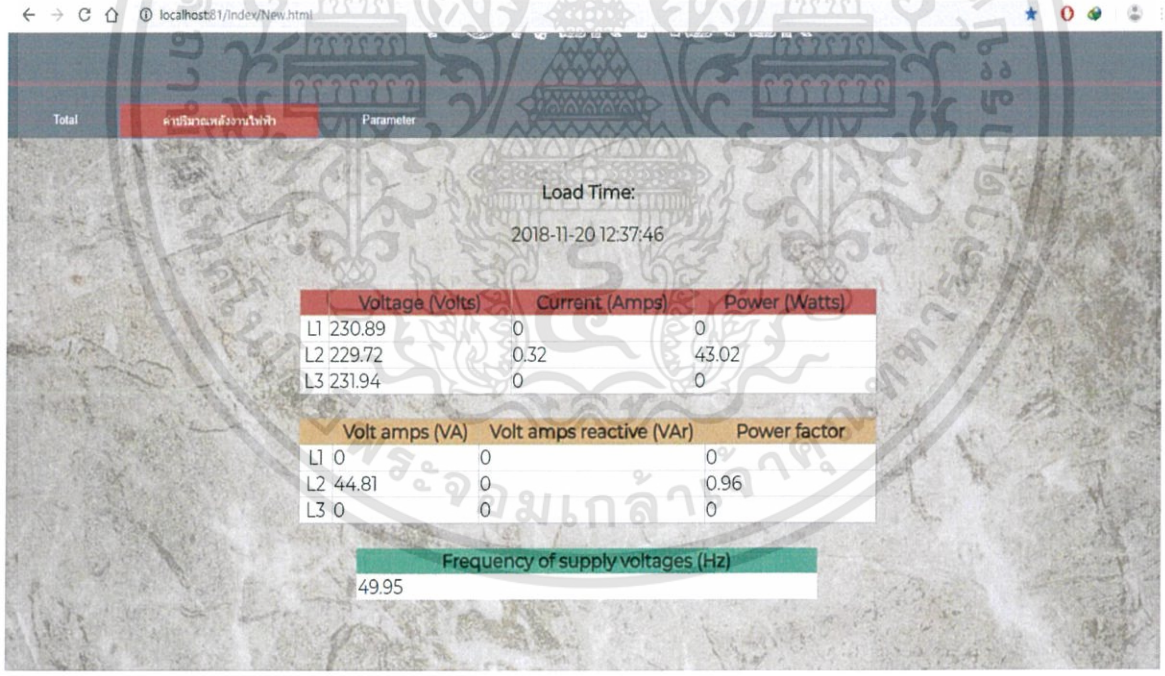


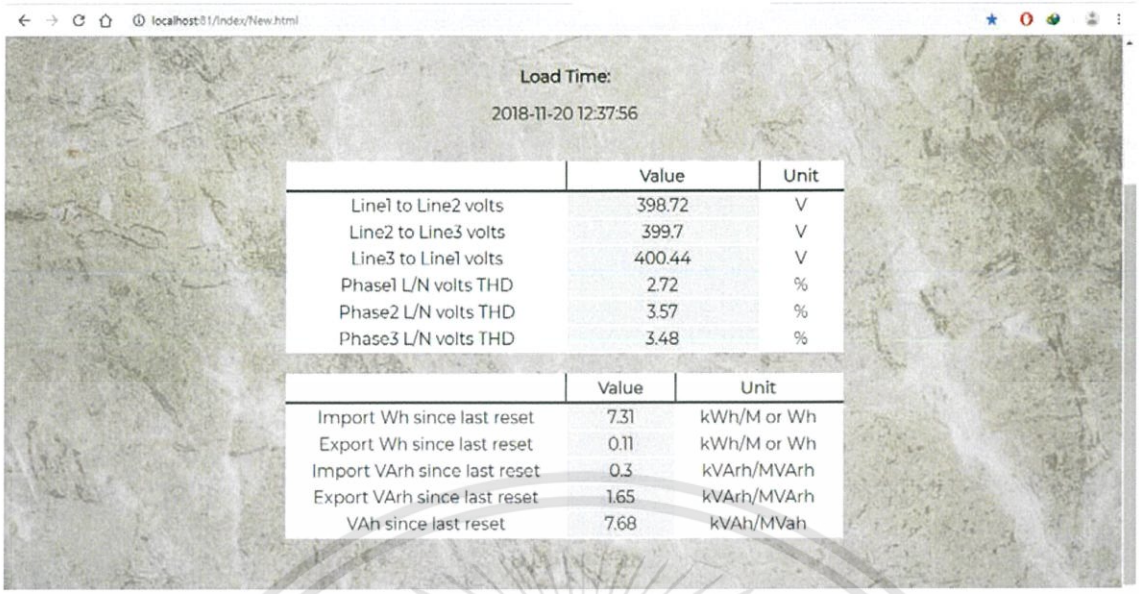

**ภาพที่ 4.10** ตัวอย่างผลลัพธ์ในโปรแกรม LabVIEW



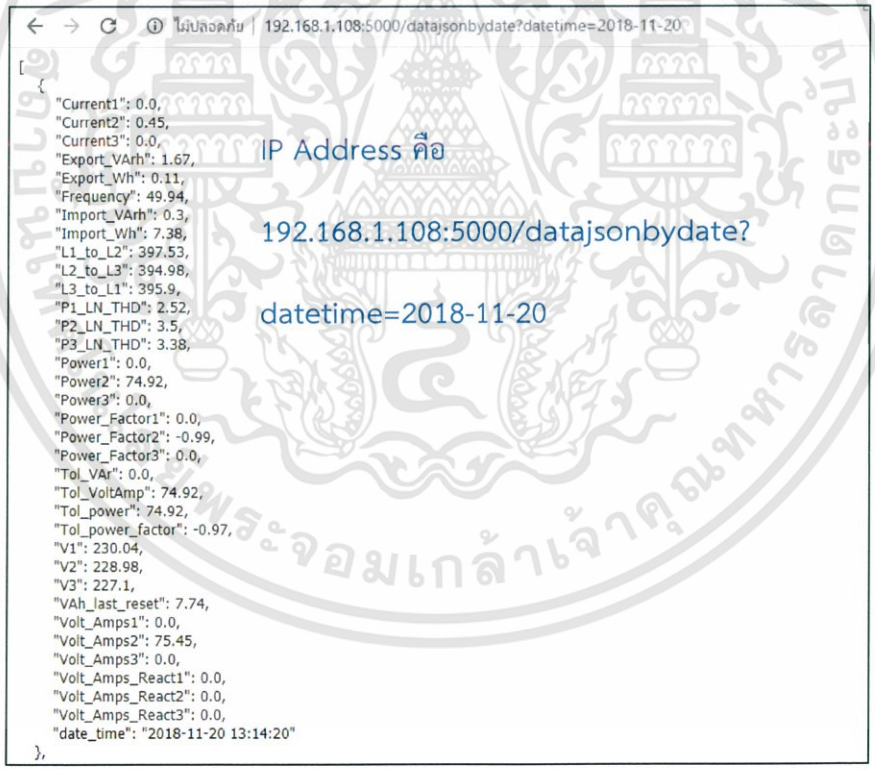

**ภาพที่ 4.11** ผลลัพธ์บนเว็บเซอร์วิสสำหรับนำไปใช้กับเว็บแบบเรียลไทม์



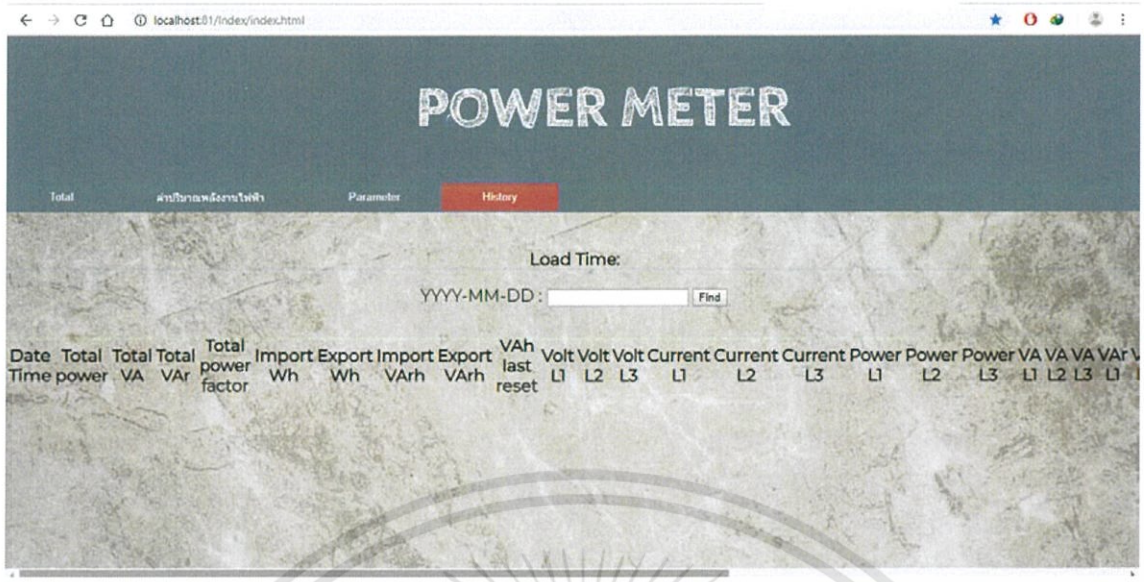



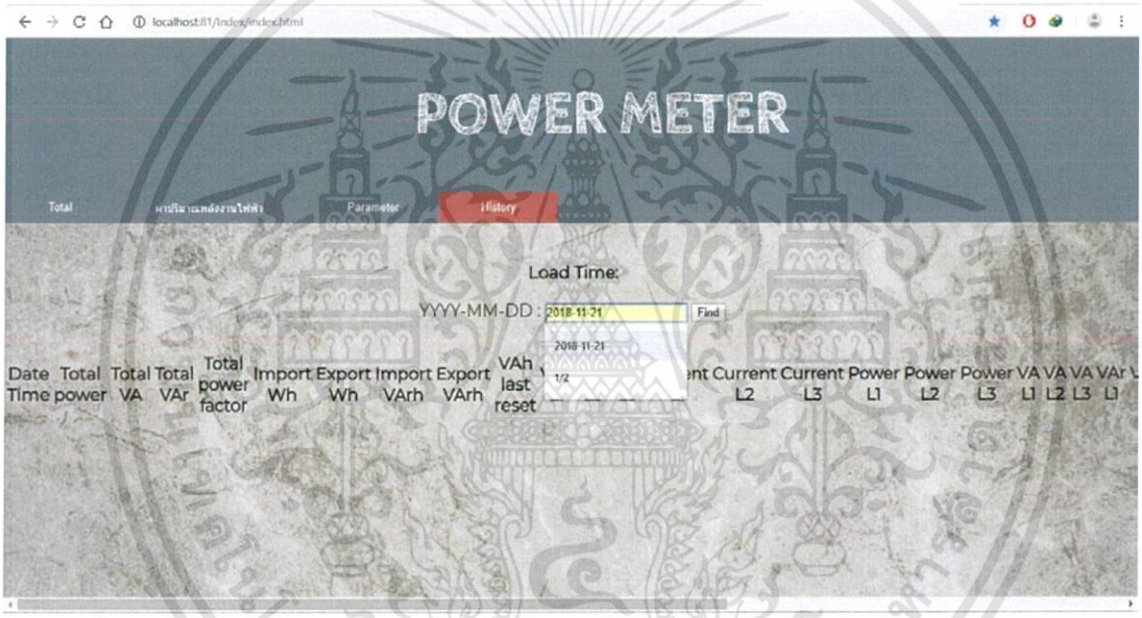


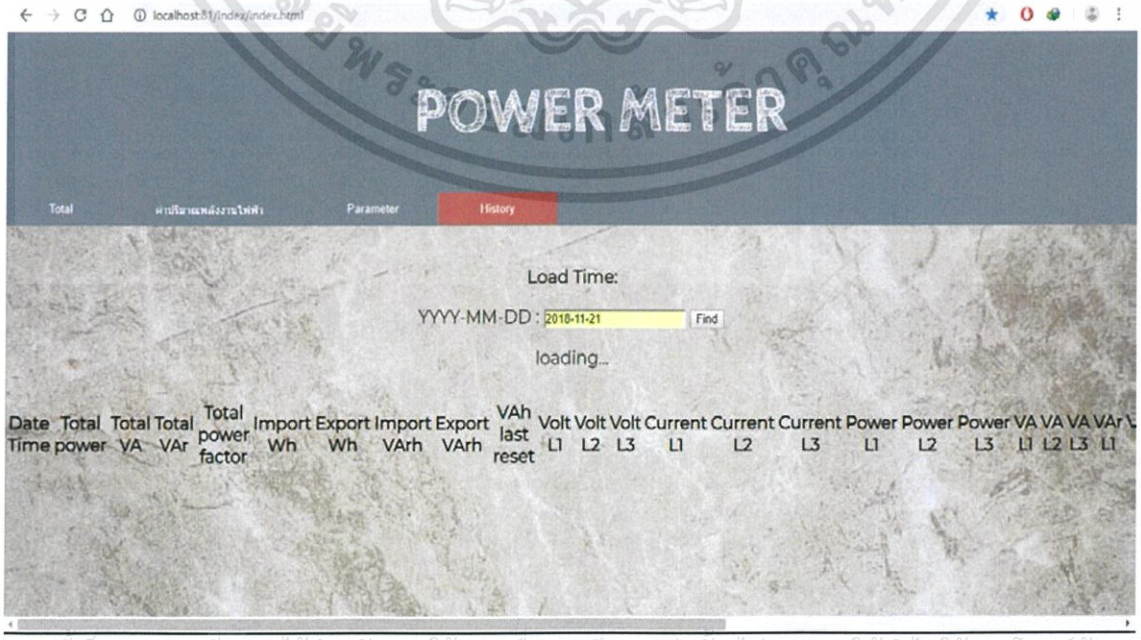

**ภาพที่ 4.12** ตัวอย่างผลลัพธ์บนหน้าเว็บแบบเรียลไทม์

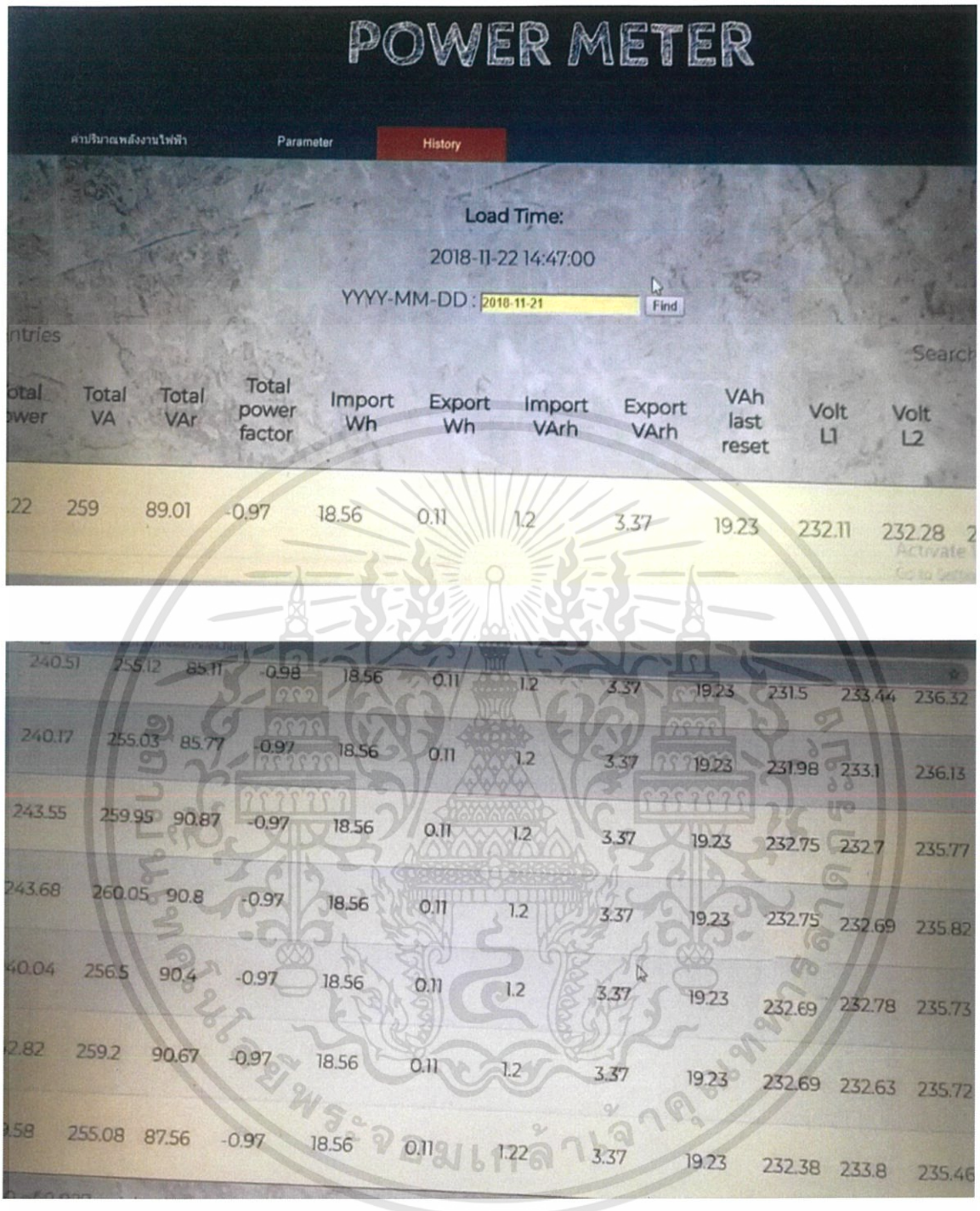



**ภาพที่ 4.13** ผลลัพธ์บนเว็บเซอร์วิสสำหรับนำไปใช้กับเว็บแบบย้อนหลัง






**ภาพที่ 4.14** ตัวอย่างผลลัพธ์บนหน้าเว็บแบบย้อนหลัง

# บทที่ 5
# บทสรุปและข้อเสนอแนะ

## 5.1 บทสรุป

โครงงานเครื่องบันทึกข้อมูลบนพื้นฐานโปรโตคอลมอดบัสโดยใช้บอร์ด ASUS Tinker สำหรับการวัดพลังงานระยะไกลนี้เป็นการทำร่วมกับบริษัท สเกด้า ออโตเมชั่น จำกัด ซึ่งการทำโครงงานเรื่องนี้ต้องใช้ความรู้หลายด้านได้แก่ การเขียนโปรแกรมเพื่ออ่านโปรโตคอลมอดบัส-อาร์ทียู การเขียนโปรแกรมเพื่อบันทึกค่าพารามิเตอร์ลงฐานข้อมูล และการเขียนโปรแกรมเพื่อแชร์ข้อมูลให้ในส่วนของซอฟต์แวร์ LabVIEW และเว็บนำข้อมูลไปแสดงได้อีกด้วย จากการดำเนินงานสรุปได้ว่าโครงงานเรื่องนี้สามารถอ่านค่าพารามิเตอร์จากโปรโตคอลมอดบัส-อาร์ทียูได้ โดยค่าที่อ่านได้จะเป็น floating แต่ถ้าเกิดการ Loss Connection จะมีการแจ้งเตือนด้วยเช่นกัน หลังจากนั้นจะนำค่าที่อ่านได้บันทึกลงฐานข้อมูลแต่ถ้าเกิดการ Loss Connect จะไม่สามารถบันทึกลงฐานข้อมูลได้ และส่วนที่แชร์ข้อมูลก็สามารถแสดงค่าบนซอฟต์แวร์ LabVIEW และหน้าเว็บได้ แต่ต้องเชื่อมต่อผ่านสัญญาณ Wi-Fi เท่านั้นโดยใช้ IP Address เข้าถึงสู่อินเตอร์เน็ตหรือที่เรียกว่าเทคโนโลยี IIoT (Industrial Internet of Thing)

## 5.2 ปัญหาและอุปสรรค

1. เพาเวอร์มิเตอร์รุ่น EASTRON SMART X96-3 ไม่มี Datasheet จึงต้องใช้ Datasheet ของรุ่นอื่นที่ใกล้เคียงกันมาอ้างอิงในการทำงาน

2. ความรู้และความเข้าใจในการเขียนโปรแกรมภาษา Python, JSON และการบันทึกค่าลงฐานข้อมูล

3. อุปกรณ์ Hardware ในการเชื่อมต่อระบบ

4. ระบบปฏิบัติการ Operating System ที่ใช้ในการเชื่อมต่ออุปกรณ์ต่าง ๆ ของคอมพิวเตอร์ให้สามารถทำงานร่วมกับโปรแกรมประยุกต์ที่ใช้ในการทำงานมีปัญหาในการรองรับแค่บางโปรแกรม

## 5.3 ข้อเสนอแนะ

1. ในส่วนของการเขียนโปรแกรมอาจมีการปรับปรุงให้มีการแจ้งเตือนสถานะของการทำงานโดยใช้ LED ต่อเข้ากับบอร์ด ASUS Tinker

2. ปรับปรุงในส่วน Hardware โดยทำเป็น Plant

# เอกสารอ้างอิง


[1] ***Tinker board.*** [ออนไลน์]. แหล่งที่มา *https://www.asus.com/th/Motherboards/Tinker-Board/* (6 สิงหาคม 2561)

[2] ***Python Tutorial.*** [ออนไลน์]. แหล่งที่มา *https://www.w3schools.com/python/* (6 สิงหาคม 2561)

[3] สุพจน์ สง่ากอง และ ปิยะ นากสงค์. **การเขียนโปรแกรมภาษา Python.** พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : โปรดีไซน์ จำกัด,2561.

[4] ผศ.สุชาติ คุ้มมะณี. **การเขียนโปรแกรมกับมัลติเธรดและระบบเครือข่าย. [ฉบับอิเล็กทรอนิกส์].** Programming expert with Python เชี่ยวชาญการเขียนโปรแกรมด้วยไพธอน, หน้า 343.

[5] ***ปัญญาประดิษฐ์เริ่มควบรวมกับ IoT และ IIoT.*** [ออนไลน์]. แหล่งที่มา *https://www.theeleader.com/iot/ปัญญาประดิษฐ์-iot-iiot-content02/* (31 สิงหาคม 2561)

[6] ***CT วัดกระแส (Measuring Current Transformer).*** [ออนไลน์]. แหล่งที่มา *https://www.pmk.co.th/shop/ct* (7 กันยายน 2561)

[7] ***RS485.*** [ออนไลน์]. แหล่งที่มา *https://www.omi.co.th/th/article/rs485* (8 กันยายน 2561)

[8] ***How To Download & Install SQLite.*** [ออนไลน์]. แหล่งที่มา *http://www.sqlitetutorial.net/download-install-sqlite/* (11 กันยายน 2561)

[9] ***SQLite – Python.*** [ออนไลน์]. แหล่งที่มา *https://www.tutorialspoint.com/sqlite/sqlite_python.htm* ( 12 กันยายน 2561)

[10] ***Date and Time Functions.*** [ออนไลน์]. แหล่งที่มา *https://www.sqlite.org/lang_datefunc.html* ( 14 กันยายน 2561)

[11] ***Python multithreading.*** [ออนไลน์]. แหล่งที่มา *http://www.w3big.com/th/python/python-multithreading.html* ( 10 ตุลาคม 2561)

[12] ***MODBUS.*** [ออนไลน์]. แหล่งที่มา *https://www.omi.co.th/th/article/modbus* ( 18 ตุลาคม 2561)

# เอกสารอ้างอิง (ต่อ)


[13] ***วิธีติดตั้ง Module เสริมของ Python ด้วย pip.*** [ออนไลน์]. แหล่งที่มา *http://www.mindphp.com/forums/viewtopic.php?f=144&t=38007* ( 19 ตุลาคม 2561)

[14] ***Python ๑๐๑.*** [ออนไลน์]. แหล่งที่มา *https://www.cp.eng.chula.ac.th/books/wp-content/uploads/sites/5/2018/08/python101_workbook_v1.0.2.pdf* ( 20 ตุลาคม 2561)

[15] ***Web service คืออะไร.*** [ออนไลน์]. แหล่งที่มา *https://saixiii.com/what-is-webservice* ( 24 ตุลาคม 2561)

[16] ***Flask.*** [ออนไลน์]. แหล่งที่มา *http://flask.pocoo.org/* ( 25 ตุลาคม 2561)

[17] ***Flask-JSON.*** [ออนไลน์]. แหล่งที่มา *https://flask-json.readthedocs.io/en/latest/* ( 8 พฤศจิกายน 2561)

[18] ***THD คืออะไร คำนวณอย่างไงนะ?.*** [ออนไลน์]. แหล่งที่มา *http://thaieeandestuff.blogspot.com/2016/02/thd.html* ( 14 พฤศจิกายน 2561)

[19] ***Wi-Fi คืออะไร มีประโยชน์อย่างไร WiFi มีมาตรฐานอะไรบ้าง.*** [ออนไลน์]. แหล่งที่มา *https://www.xn--12cg1cxchd0a2gzc1c5d5a.net/wifi/* ( 15 พฤศจิกายน 2561)