รายงานสหกิจศึกษาฉบับสมบูรณ์

มาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบ
Energy meter monitoring over NB-IoT systems

นายสรรเสริญ เกิดแก่น

ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

รายงานสหกิจศึกษาฉบับสมบูรณ์

มาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบ

Energy meter monitoring over NB-IoT systems

นายสรรเสริญ เกิดแก่น

ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

ชื่อโครงงานสหกิจศึกษา มาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบ
ชื่อ-สกุล นักศึกษา นายสรรเสริญ เกิดแก่น
คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมโทรคมนาคม
ชื่อ-สกุล อาจารย์นิเทศ ดร. สถาพร พรหมวงศ์
ชื่อ-สกุล ผู้นิเทศงาน นายธีรยสถ์ อุดมมณีธนกิจ และ นายราวิน ประยูรหงษ์
สถานประกอบการ บริษัท ทรู คอร์ปอเรชั่น จำกัด (มหาชน)

บทคัดย่อ

บริษัท ทรู คอร์ปอเรชั่น จำกัด (มหาชน) เป็นผู้ให้บริการเครือข่ายโทรศัพท์เคลื่อนที่ในประเทศไทย ซึ่งผู้จัดทำได้เข้าไปทำโครงงานสหกิจศึกษาร่วมกับหน่วยงานภายในที่มีหน้าที่ดูแลเกี่ยวกับการเพิ่มประสิทธิภาพการทำงานของระบบเครือข่ายของพื้นที่ให้บริการทั่วประเทศ หนึ่งในปัญหาที่หน่วยงานพบเจอก็คือการมีการรายงานปริมาณการใช้ไฟฟ้าที่ผิดปกติจากการไฟฟ้าฯ ทำให้บริษัทฯ ต้องเสียค่าใช้จ่ายเกินความจำเป็น รวมถึงยังต้องส่งวิศวกรหรือช่างเทคนิคออกไปตรวจสอบยังไซต์งานที่อยู่ห่างไกลออกไปอีกด้วย

ระบบมาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบที่ได้พัฒนาขึ้นมานั้น จะช่วยในการมอนิเตอร์ปริมาณการใช้งานไฟฟ้า และค่าพารามิเตอร์ทางไฟฟ้าอื่น ๆ โดยไม่จำเป็นต้องส่งวิศวกรหรือช่างเทคนิคออกไปตรวจสอบ เป็นผลให้หน่วยงานสามารถลดค่าใช้จ่ายที่ไม่จำเป็นลงได้อย่างมาก

คำสำคัญ : มาตรวัดพลังงานไฟฟ้า, ไอโอทีแบนด์แคบ, การมอนิเตอร์, การแจ้งเตือน

Cooperative Title: Energy meter monitoring over NB-IoT systems
Student intern name: Mr. Sansoen Koedkaen
Faculty: Engineering **Department:** Telecommunication Engineering
Advisor name: Dr. Sathaporn Promwong
Mentor name: Mr. Teerayot Udommaneetanakit and Mr. Rawin Prayoonhong
Company: True Corporation Public Company Limited

ABSTRACT

True Corporation Public Co., Ltd. is a mobile network service provider in Thailand. The organizer does the cooperative project with the department who responsible for optimized the efficiency of regional network service. One of the problems that our department encountered was the abnormal reported of the electricity usage from electricity authority. Which cause of company unnecessary expense because we need to send engineers or technicians to examine the remote cell sites.

Energy meter monitoring over NB-IoT systems has been developed is helpful with electricity usage and electrical parameters monitoring without sending the staff to examine the remote cell sites. Which effective of significantly unnecessary expense reduction.

Keywords: Energy meter, NB-IoT, Monitoring, Alarm

กิตติกรรมประกาศ

โครงงาน “มาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบ” ต้องขอขอบคุณเป็นอย่างสูงที่สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง คณะวิศกรรมศาสตร์ที่ได้กรุณาจัดโครงการสหกิจศึกษา โครงงานนี้ไม่อาจสำเร็จลุล่วงไปได้หากขาดการสนับสนุนจากหลาย ๆ ฝ่าย ดังนี้

อาจารย์ ดร. สถาพร พรหมวงศ์ ซึ่งเป็นอาจารย์นิเทศผู้ที่ได้ให้คำปรึกษา และชี้แนะแนวทางในการทำงาน คอยสนับสนุนช่วยเหลือเมื่อมีปัญหาตลอดการทำโครงการนี้คุณธีรยสถ์ อุดมมณีธนกิจ Assistant Director คุณราวิน ประยูรหงษ์ Engineering Specialist ที่คอยให้คำปรึกษา ช่วยแนะนำแนวทางในการแก้ปัญหา รวมทั้งคอยสนับบสนุนอุปกรณ์ที่ใช้ในการดำเนินงาน สถานที่ ตลอดระยะเวลาในการดำเนินงาน

ขอขอบคุณ พี่ ๆ และเพื่อน ๆ ภายในบริษัท ทรู คอร์ปอเรชั่น จำกัด (มหาชน) ทุกคนที่คอยให้ความช่วยเหลือ ให้คำปรึกษาในเรื่องต่าง ๆ และให้ความรู้เพิ่มเติมที่เป็นประโยชน์ต่อผู้จัดทำ

สุดท้ายนี้ขอขอบพระคุณ บิดา มารดา ที่คอยส่งเสริมและสนับสนุนทางด้านปัจจัยในการทำโครงงาน และคอยให้กำลังใจในการทำโครงการสหกิจศึกษานี้

นายสรรเสริญ เกิดแก่น
ผู้จัดทำ

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

ในปัจจุบันเทคโนโลยีการสื่อสารแบบไร้สายได้เข้ามาทำให้การติดต่อสื่อสารนั้นง่ายดายและสะดวกสบายมากขึ้น ทั้งการสื่อสารระหว่างบุคคลกับบุคคล บุคคลกับอุปกรณ์ หรืออุปกรณ์กับอุปกรณ์ เทคโนโลยีการสื่อสารแบบไร้สายนั้นมีมากมายหลายมาตรฐาน แต่ละมาตรฐานก็จะมีข้อดีและข้อเสียแตกต่างกันไป หนึ่งในมาตรฐานเทคโนโลยีการสื่อสารแบบไร้สายที่น่าสนใจก็คือ ระบบการสื่อสารผ่านระบบเครือข่ายไอโอทีแบบแบนด์แคบ (NB-IoT) เนื่องจากเป็นเทคโนโลยีที่นำมาใช้ในการรับส่งข้อมูลที่มีขนาดเล็ก เหมาะกับการนำมาประยุกต์ใช้เข้ากับการรับส่งข้อมูลที่มีอุปกรณ์จำนวนมาก เนื่องจากสิ้นเปลืองแบนด์วิธของเครือข่ายน้อย ทำให้สามารถรองรับอุปกรณ์ได้หลายตัว

โครงงานนี้มีวัตถุประสงค์เพื่อสร้างระบบสังเกตการณ์ปริมาณการใช้งานไฟฟ้าและค่าพารามิเตอร์ทางไฟฟ้าของไซต์งานต่าง ๆ ที่อยู่กระจายทั่วประเทศ อันเนื่องมาจากปัญหาการมีการเรียกเก็บค่าไฟฟ้าที่สูงผิดปกติจากการไฟฟ้าฯ ส่งผลให้สิ้นเปลืองงบประมาณเกินความจำเป็น และการที่มีการใช้งานไฟฟ้าเกินพิกัดของเครื่องสำรองไฟฟ้า จนทำให้เกิดเหตุการณ์ระบบล่มอันเนื่องมาจากเครื่องสำรองไฟฟ้าไม่สามารถจ่ายไฟฟ้าให้ได้ตามที่อุปกรณ์ในไซต์งานต้องการ

ซึ่งในปัจจุบันทางหน่วยงานยังคงต้องแก้ไขปัญหาด้วยการให้วิศวกรออกเดินทางไปตรวจสอบยังไซต์งาน ทำให้มีงบประมาณในการเดินทางเพิ่มมากขึ้น ระบบมาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบที่ได้สร้างขึ้นนั้นจะมาทดแทนการส่งวิศวกรออกไปสำรวจ โดยระบบจะทำให้สามารถตรวจสอบปริมาณการใช้งานไฟฟ้าและค่าพารามิเตอร์ทางไฟฟ้าต่าง ๆ ได้แบบเรียลไทม์และแบบย้อนหลังผ่านระบบการแสดงผลแบบ Dashboard ของ Cloud Server ที่ได้จัดทำขึ้น และนอกจากนี้ยังสามารถสร้างรายงานปริมาณการใช้งานไฟฟ้าย้อนหลัง เพื่อนำไปเปรียบเทียบรายงานการใช้ไฟฟ้าของการไฟฟ้าฯ ได้อีกด้วย

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อนำ NB-IoT ซึ่งเป็นเทคโนโลยีใหม่ มาประยุกต์ใช้ในการรายงานผลมาตรวัดพลังงานไฟฟ้า เพื่อลดค่าใช้จ่ายในการเดินทางไปตรวจสอบมาตรวัดเนื่องจากมีความผิดปกติของค่าไฟฟ้า

2) เพื่อสังเกตการณ์และแจ้งเตือนได้ทันทีเมื่อมีค่าไฟฟ้าสูงผิดปกติ

3) เพื่อเก็บสถิติการใช้งานพลังงานไฟฟ้าสำหรับการวางแผนการจัดการพลังงานไฟฟ้าให้เป็นไปอย่างมีประสิทธิภาพสูงสุด

4) เพื่อจัดทำมาตรวัดพลังงานไฟฟ้าต้นแบบสำหรับการพัฒนาให้รองรับในระบบไฟฟ้ากำลังที่มีขนาดใหญ่ขึ้น (ระบบ 3 เฟส) สำหรับการนำไปติดตั้งในไซต์งานขนาดใหญ่ (Large Node)

1.3 ขอบเขตของโครงงาน

1) สร้างมาตรวัดพลังงานไฟฟ้าที่สามารถนำประยุกต์เข้ากับระบบ NB-IoT ของบริษัทได้

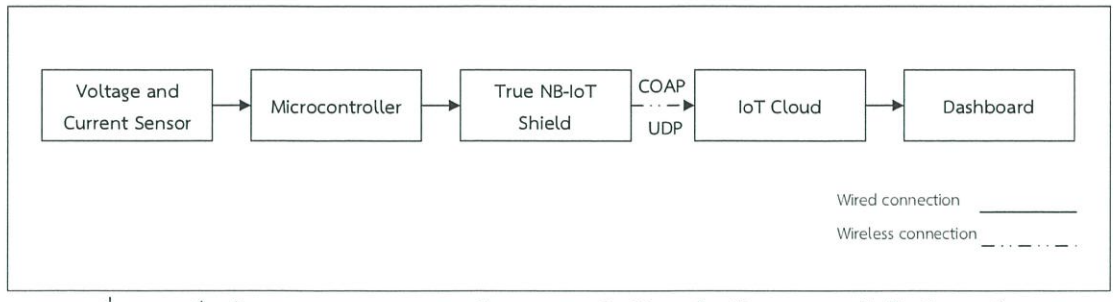
2) สามารถเรียกดูปริมาณการใช้พลังงานไฟฟ้าและค่าพารามิเตอร์ทางไฟฟ้าผ่าน Dashboard ของระบบ Cloud ได้ทุกที่ ทุกเวลา

3) สามารถนำระบบควบคุมไปติดตั้งกับเซ็นเซอร์วัดพลังงานไฟฟ้าที่มีกำลังสูงขึ้นในอนาคตได้

1.4 วิธีการดำเนินงานโครงงาน

ตารางที่ 1.1 แผนการดำเนินงาน

ลำดับ	รายละเอียด	ภาคการศึกษาที่ 1 ปีการศึกษา 2561			
		เดือนที่ 1	เดือนที่ 2	เดือนที่ 3	เดือนที่ 4
1	ศึกษาการทำงานของมาตรวัดพลังงานไฟฟ้า	←	→		
2	ศึกษาการคิดอัตราค่าไฟฟ้าของการไฟฟ้าส่วนภูมิภาคและการไฟฟ้านครหลวง	←	→		
3	ศึกษาการทำงานของระบบ NB-IoT		←	→	
4	ทำการทดลองเชื่อมต่อ NB-IoT Shield กับ Microcontroller เพื่อทำการส่งข้อมูลไปยัง Cloud Server		←	→	
5	จัดทำมาตรวัดพลังงานไฟฟ้าและเขียนโปรแกรมสำหรับการควบคุมมาตรวัดพลังงานไฟฟ้า เพื่อส่งข้อมูลไปยัง Cloud Server			←	→
7	ทดสอบกการทำงานของมาตรวัดพลังงานไฟฟ้าและจัดทำ Dashboard สำหรับการแสดงผล			←	→



รูปที่ 1.1 บล็อกไดอะแกรมของระบบสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบ

1.5 ผลที่คาดว่าจะได้รับ

1) สามารถสร้างระบบสังเกตการณ์มาตรวัดพลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบเพื่อนำไปใช้ในการตรวจสอบปริมาณการใช้งานไฟฟ้าและค่าพารามิเตอร์ทางไฟฟ้า ของไซต์งาน หรือตามบ้านเรือนได้

2) ได้รับความรู้ในการเชิงปฏิบัติของระบบไอโอทีแบนด์แคบ ซึ่งเป็นความรู้ที่ได้ศึกษามาจากการเรียนในภาคทฤษฎีในปีการศึกษาก่อนหน้า

3) ได้ฝึกฝนทักษะความสามารถในการเขียนโปรแกรม และการจัดทำฮาร์ดแวร์ทางไฟฟ้ากำลัง

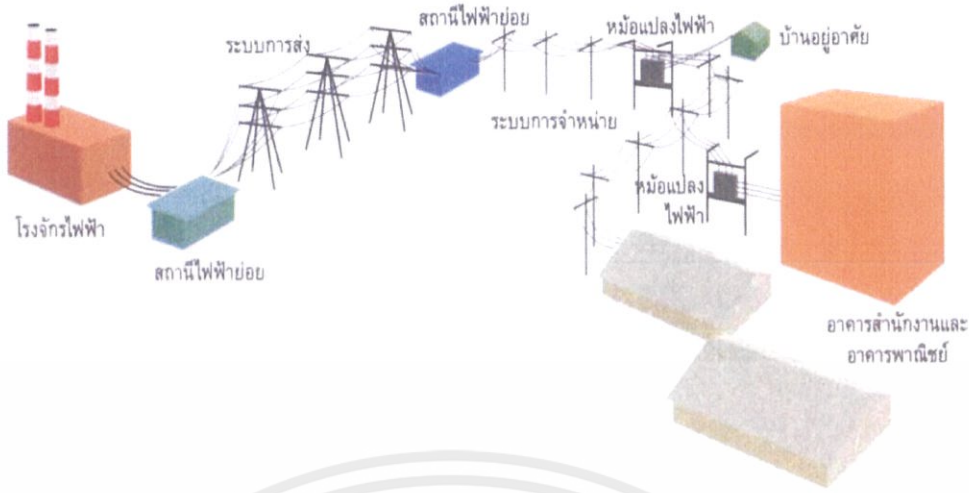
บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 ระบบไฟฟ้ากำลัง

พลังงานไฟฟ้าเป็นพลังงานที่ส่งผ่านตัวนำและหม้อแปลงไฟฟ้าไปยังผู้ใช้ปลายทาง โดยการไฟฟ้าผู้ให้ บริการเองก็มีการจัดระบบของการส่งจ่ายไฟฟ้าไปยังอาคารธุรกิจ โรงงานหรือบ้านพักอาศัย โดยส่งจ่ายแรงดัน ในเขตเมืองและย่านชุมชนเป็นหลายช่วงแรงดัน โดยหม้อแปลงไฟฟ้าจำหน่ายที่แปลงแรงดันสูงให้เป็นแรงดัน ต่ำนั้นจะถูกติดตั้งในบริเวณใกล้จุดที่มีความต้องการใช้ไฟฟ้าให้มากที่สุด เพื่อลดการสูญเสียในสายและลดปัญหาแรงดันตกในสายไฟ ปัญหาการสูญเสียในสายไฟและในหม้อแปลงอาจมีคนจำนวนมากมองข้าม แต่ท่าน ทราบหรือไม่ว่าในปี พ.ศ. 2547 มีรายงานว่าการสูญเสียในสายส่งไฟฟ้าของระบบส่งจ่ายไฟฟ้ารวมทั้งประเทศ สูงเกือบ 10,000 ล้านหน่วยทีเดียว เมื่อพิจารณาถึงระบบไฟฟ้าในอาคารขนาดใหญ่หรือในโรงงานอุตสาหกรรม แล้ว มีการประมาณการไว้ว่ามีการสูญเสียพลังงานไฟฟ้าในหม้อแปลงประมาณ 2-3 % และมีการสูญเสีย พลังงานไฟฟ้าในสายไฟฟ้าไม่เกิน 1 % โดยการสูญเสียในระบบไฟฟ้านี้อยู่ในวิสัยที่จะบริหารจัดการให้น้อยลง ได้ เมื่อเทียบกับกรณีที่ไม่ได้ให้ความสนใจเลย นอกจากนี้หากไม่มีการบริหารจัดการระบบไฟฟ้าของโรงงานหรืออาคารให้อยู่ในเกณฑ์ที่ควรจะเป็น นอกจากจะมีการสูญเสียในระบบมากแล้ว ยังอาจถูกการไฟฟ้าเรียกเก็บ ค่าตัวประกอบกำลังไฟฟ้า เป็นค่าใช้จ่ายเพิ่มเติมคล้ายค่าปรับในเอกสารค่าไฟฟ้าอีกด้วยก็ได้ และที่เรียกว่าเป็น ค่าปรับ เพราะค่าใช้จ่ายส่วนนี้สามารถหลีกเลี่ยงได้ด้วยการบริหารระบบไฟฟ้านั่นเอง

โดยปกติแล้ว สายส่งไฟฟ้าแรงสูงจะมีการส่งจ่ายแรงดันอยู่หลายค่า เพื่อส่งจ่ายและจำหน่ายพลังงานไฟฟ้าอย่างเหมาะสม โดยปกติแล้วผู้ใช้ไฟสามารถเลือกขอใช้แรงดันไฟฟ้าได้ตามความต้องการ หากไม่ขัดกับ ระเบียบปฏิบัติของทางการไฟฟ้า และมีแนวสายไฟฟ้าพร้อมให้บริการ โดยปกติแล้วอาคารขนาดใหญ่หรือโรงงานมักจะซื้อไฟฟ้าแรงสูงจากการไฟฟ้า หากเป็นพื้นที่ให้บริการของการไฟฟ้านครหลวง (กฟน.) ก็จะมี แรงดัน 12-24 kV แต่ถ้าเป็นพื้นที่ให้บริการของการไฟฟ้าส่วนภูมิภาค (กฟภ.) ก็จะมีแรงดัน 22-33 kV (ค่าไฟฟ้า เก็บในอัตราเดียวกัน) ในกรณีที่เป็นผู้ใช้ไฟฟ้าขนาดใหญ่มากนั้นมักจะเลือกซื้อไฟจากการไฟฟ้าด้วยแรงดัน 69 kV หรือ 115 kV เพราะแรงดันที่สูงขึ้นนั้น จะมีค่าไฟฟ้าที่ถูกกว่า แต่การลงทุนต้านระบบไฟฟ้าของผู้ใช้ก็จะสูงขึ้นด้วย



รูปที่ 2.1 รูปแบบระบบส่งจ่ายและจำหน่ายไฟฟ้า [1]

จากรูปที่ 2.1 แสดงรูปแบบระบบส่งจ่ายและจำหน่ายไฟฟ้า จะสังเกตได้ว่ากรณีบ้านอยู่อาศัย ซึ่ง เป็นผู้ใช้ไฟฟ้ารายย่อยมีจำนวนมากและซื้อไฟฟ้าแรงต่ำจากการไฟฟ้า การไฟฟ้าจะลงทุนเรื่อง ระบบไฟฟ้าทั้งหมด ตั้งแต่หม้อแปลงจำหน่าย เสาไฟ สายไฟไปจนถึงเครื่องวัดหน่วยหน้าบ้านผู้ซื้อไฟ หากมีปัญหาหาที่หม้อ แปลง เสาไฟฟ้า สายไฟขาด การไฟฟ้าจะเป็นผู้บำรุงรักษาแก้ไขปัญหาเอง ทั้งหมด เพราะเป็นสมบัติของการ ไฟฟ้า ขณะที่กรณีของอาคารขนาดใหญ่หรือโรงงานอุตสาหกรรม นั้น การไฟฟ้าจะเดินไฟฟ้าแรงสูงมาถึงสถานที่ ๆ ใช้ไฟ ณ ตำแหน่งที่ทำการติดตั้งเครื่องวัด โดย หลักการแล้วค่าใช้จ่ายตั้งแต่หลังเครื่องวัดไปนั้น ผู้ใช้ไฟ จะต้องลงทุนเองทั้งหมด นับตั้งแต่หม้อแปลง เสาไฟ สายไฟไปจนถึงระบบป้องกัน เพราะเป็นสมบัติของผู้ใช้ไฟ ในกรณีนี้หากเกิดเหตุหม้อแปลง ระเบิด หรือรถชนเสาไฟฟ้าภายในโรงงานแล้ว การไฟฟ้าจะไม่เข้ามาแก้ไข ให้เพราะไม่ได้อยู่ใน ขอบข่ายความรับผิดชอบของทางการไฟฟ้า จากกรณีผู้ซื้อไฟฟ้าทั้ง 2 กรณีที่กล่าวถึงนี้ เมื่อ เปรียบเทียบกันแล้ว อาจตั้งข้อสังเกตได้ดังนี้

กรณีบ้านอยู่อาศัยที่ซื้อไฟฟ้าแรงต่ำ ค่าไฟฟ้าน่าจะแพงกว่าซื้อไฟฟ้าแรงสูง เพราะการไฟฟ้า ต้องลงทุน ด้านระบบมากกว่ากรณีที่ซื้อไฟฟ้าแรงสูง

- กรณีบ้านอยู่อาศัยที่ซื้อไฟฟ้าแรงต่ำ การสูญเสียในหม้อแปลงและสายส่ง ภาระตกอยู่กับการ ไฟฟ้า จึง เป็นอีกเหตุผลหนึ่งที่น่าจะทำให้ค่าไฟฟ้าแพงกว่าซื้อไฟฟ้าแรงสูง
- กรณีบ้านอยู่อาศัยที่ซื้อไฟฟ้าแรงต่ำ ค่าใช้จ่ายในการบำรุงรักษาและบริหารระบบ ภาระตก อยู่กับการ ไฟฟ้า จึงเป็นอีกเหตุผลหนึ่งที่น่าจะทำให้ค่าไฟฟ้าแพงกว่าซื้อไฟฟ้าแรงสูง
- ข้อสังเกตข้างต้นสามารถสะท้อนให้เห็นได้จากอัตราค่าไฟฟ้า

จากข้อสังเกตข้างบนนี้ จึงชี้ให้เห็นได้ว่าผู้ซื้อไฟฟ้าแรงสูง จะต้องลงทุนด้านระบบไฟฟ้า มากกว่า แต่ก็มีค่าไฟฟ้าที่ถูกกว่า อย่างไรก็ดี จะเห็นได้ว่า การสูญเสียในหม้อแปลงและสายส่ง ตลอดจนค่าใช้จ่ายในการ บำรุงรักษาและบริหารระบบเป็นภาระที่ตกอยู่กับผู้ซื้อไฟฟ้าแรงสูงด้วย ดังนั้น การบริหารจัดการการใช้ไฟฟ้า ในโรงงานหรืออาคารจึงเป็นสิ่งที่หลีกเลี่ยงไม่ได้ และ

ไม่ควรมองข้าม นอกจากนี้ สิ่งที่ผู้ซื้อไฟฟ้าแรงสูงควร ทราบอีกประเด็นหนึ่งก็คือ โครงสร้างค่าไฟฟ้าที่มีข้อปลีกย่อยมากกว่ากรณีบ้านอยู่อาศัย โดยมีการจำแนกประเภทของผู้ใช้ไฟฟ้าเป็น 7 ประเภท คือ

- ประเภทที่ 1 บ้านอยู่อาศัย
- ประเภทที่ 2 กิจการขนาดเล็ก
- ประเภทที่ 3 กิจการขนาดกลาง
- ประเภทที่ 4 กิจการขนาดใหญ่
- ประเภทที่ 5 กิจการเฉพาะอย่าง
- ประเภทที่ 6 ส่วนราชการและองค์กรที่ไม่แสวงหากำไร
- ประเภทที่ 7 สูบน้ำเพื่อการเกษตร

โดยรายละเอียดอัตราค่าไฟฟ้าของการไฟฟ้าส่วนภูมิภาคดูได้จากหัวข้อที่**ภาคผนวก ก**

แต่ละประเภทมีรายละเอียดของโครงสร้างค่าไฟฟ้าและเงื่อนไขที่แตกต่างกันไป กรณีของบ้านอยู่อาศัยนั้น โครงสร้างค่าไฟฟ้าเป็นแบบอัตราก้าวหน้า (ยิ่งใช้ไฟมาก ค่าไฟต่อหน่วยยิ่งแพงขึ้น) แต่การพิจารณาจะดูจาก หน่วยไฟฟ้าที่ใช้เท่านั้น ซึ่งต่างกับผู้ซื้อไฟฟ้าแรงสูงที่ลูจากข้อมูลหลายตัว ดังที่แสดงในตารางที่ 2.1 โดยแสดงให้เห็นถึงโครงสร้างค่าไฟฟ้าของผู้ใช้ประเภทที่ 3 - 5 ที่เกี่ยวข้องกับอาคารและโรงงานอุตสาหกรรม ทั้งนี้จะ เห็นได้ว่าประเภทผู้ใช้ไฟฟ้ามีหลายประเภท มีการเรียกเก็บค่าใช้จ่ายหลายส่วน และยังมีเกณฑ์ค่าไฟฟ้าขั้นต่ำอีกด้วย

ตารางที่ 2.1 องค์ประกอบของโครงสร้างค่าไฟฟ้าของผู้ใช้ประเภทที่ 3 – 5 [1]

ประเภทผู้ใช้ไฟฟ้า	ค่าพลังงาน (ค่า kWh)	ค่า Demand	ค่า PF	ค่าบริการ	มีเกณฑ์ค่าไฟขั้นต่ำ
ประเภทที่ 3 กิจการขนาดกลาง					
3.1 อัตราปกติ					
3.2 อัตราตามช่วงเวลาของการใช้	✓	✓	✓		✓
(TOU Tariff)	✓	✓	✓	✓	✓
ประเภทที่ 4 กิจการขนาดใหญ่					
4.1 อัตราตามช่วงเวลาของวัน (TOD					
Tariff)	✓	✓	✓		✓
4.2 อัตราตามช่วงเวลาของการใช้					
(TOU Tariff)	✓	✓	✓	✓	✓
ประเภทที่ 5 กิจการเฉพาะอย่าง					
5.1 อัตรปกติ					
5.2 อัตราตามช่วงเวลาของการใช้	✓	✓	✓		✓
(TOU Tariff)	✓	✓	✓	✓	✓

จากตารางที่ 2.1 หากพิจารณาในด้านของการอนุรักษ์พลังงานและการบริหารต้นทุน อาจตั้งข้อสังเกตได้ เป็นประเด็นต่างได้ดังข้างล่างนี้ และอาจนำเสนอองค์ประกอบของโครงสร้างค่าไฟฟ้าได้เป็นแผนภาพดังรูปที่ 2.2

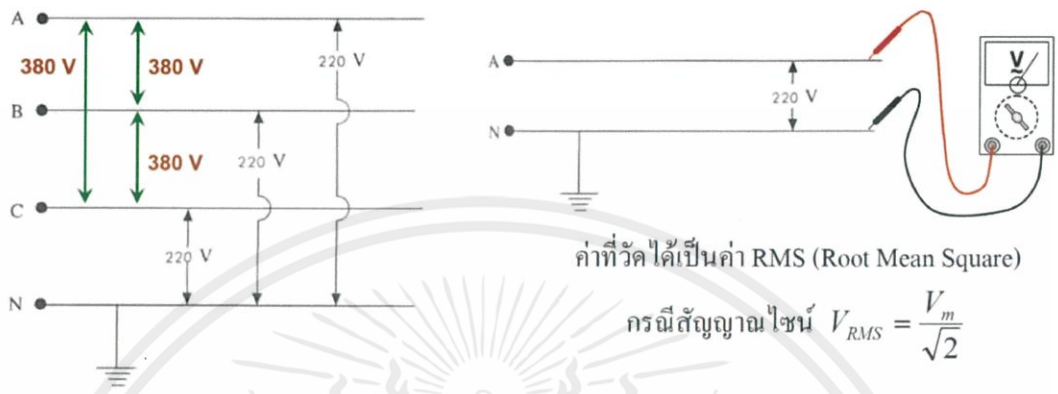
- ถ้ามีการอนุรักษ์พลังงาน หน่วยใช้ไฟ (kWh) ก็จะลดลง ค่าใช้จ่ายก็จะลดลง
- ถ้ามีการควบคุมพฤติกรรมการใช้ไฟฟ้าให้สม่ำเสมอได้ ค่า Demand ก็นำจะลดลง ค่าใช้จ่ายก็ อาจจะลดลง
- ถ้ามีการบริหารระบบไฟฟ้าได้ดี ค่าตัวประกอบกำลังไฟฟ้า (PF) มีค่าสูงกว่า 0.85 ตลอดเวลาก็ไม่ ควรจะต้องจ่ายค่าตัวประกอบกำลังไฟฟ้า (เป็นค่าใช้จ่ายที่หลีกเลี่ยงได้)
- ถ้ามีการอนุรักษ์พลังงานแล้วเป็นผลให้หน่วยใช้ไฟ (kWh) ลดลงแล้วค่า Ft ที่จ่ายก็จะลดลง รวมถึง ภาษีมูลค่าเพิ่ม (VAT) ก็จะลดลงด้วย

รูปที่ 2.2 โครงสร้างค่าไฟฟ้าโดยรวม [1]

2.1.1 แรงดันไฟฟ้ากระแสสลับ

ระบบไฟฟ้าในบ้าน อาคารและโรงงาน เป็นระบบไฟฟ้ากระแสสลับ ที่มีสัญญาณตามทฤษฎีเป็นรูปคลื่นไซน์ มีความถี่ 50 Hz โดยทั่วไปแล้วมีแรงดันระหว่างสายด้านแรงต่ำ 380 V แรงดันเฟส 220 V ดังรูปที่ 2.3 ซึ่งค่าแรงดันที่กล่าวถึงนี้ ไม่ใช่ค่ายอดคลื่น (Peak) แต่เป็นค่า RMS (Root Mean

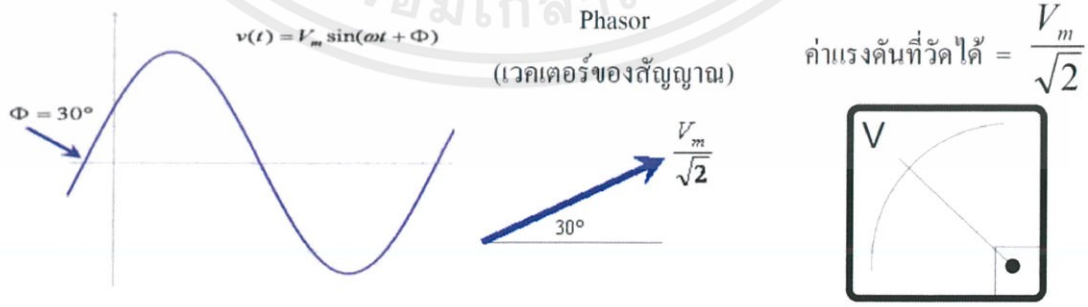
Square) เฉพาะกรณีของสัญญาณรูปคลื่นไซน์ $V_{RMS} = \frac{V_m}{\sqrt{2}}$ ถ้าสัญญาณเพี้ยนไปจากรูปคลื่นไซน์ ค่าที่วัดได้อาจมีความผิดพลาดทั้งนี้ขึ้นอยู่กับความสามารถของเครื่องวัด



รูปที่ 2.3 ลักษณะการวัดแรงดันระหว่างสาย แรงดันเฟส และค่า RMS [1]

2.1.2 การนำข้อมูลแรงดันไฟฟ้ากระแสสลับมาใช้งาน

เนื่องจากแรงดันไฟฟ้ากระแสสลับเป็นสัญญาณรูปคลื่นไซน์ ที่มีทั้งขนาดและเฟสของสัญญาณ ซึ่งการคำนวณมีความยุ่งยาก ในทางทฤษฎีจึงเขียนแทนด้วยเฟเซอร์ (Phasor) ซึ่งเป็นเวคเตอร์ชนิดหนึ่ง โดยลดรูปลงเหลือเพียงขนาดและมุม โดยมีความยาวของเวคเตอร์เท่ากับค่า RMS และมีมุมเท่ากับมุมเฟสของสัญญาณ (วัดตามทิศทวนเข็มนาฬิกา เป็นมุมบวก) ดังรูปที่ 2.4 ซึ่งปกติแล้ว ในการใช้ไฟฟ้าโดยทั่วไป การทราบแต่ขนาด (ค่า RMS) ของสัญญาณแรงดันก็เพียงพอที่จะปฏิบัติงานได้ ทำให้การวัดโดยทั่วไปละในเรื่องของมุมไป แต่เมื่อต้องทำการคำนวณหรือวิเคราะห์อย่างถูกต้อง ต้องให้วิศวกรหรือช่างเทคนิคที่มีความรู้เป็นผู้วิเคราะห์คำนวณ

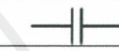


รูปที่ 2.4 ตัวอย่างสัญญาณแรงดันไฟฟ้า ค่าทางคณิตศาสตร์ และค่าที่วัดได้ [1]

2.1.3 โหลดในวงจรไฟฟ้ากระแสสลับ

วงจรไฟฟ้ากระแสสลับ จะมีความซับซ้อนกว่าวงจรไฟฟ้ากระแสตรงมาก แต่อาจกล่าวโดยสรุปได้ว่าในวงจรไฟฟ้ากระแสสลับจำแนกโหลดพื้นฐานได้เป็น 3 กลุ่ม ดังตารางที่ 2.2 ซึ่งในสภาพการใช้งานจริง อาจมีคุณลักษณะเป็นการผสมกันของโหลดมากกว่า 1 กลุ่ม การคำนวณจึง เป็นการคำนวณเลขจำนวนเชิงซ้อนแบบเวคเตอร์ ซึ่งในที่นี้ขอนำเสนอเพื่อชี้ให้เห็นถึงคุณสมบัติเด่น ๆ ที่แตกต่างกัน เพื่อจะใช้ประกอบการอธิบายต่อไป

ตารางที่ 2.2 ลักษณะโหลดพื้นฐานในวงจรไฟฟ้ากระแสสลับ [1]

	R ตัวต้านทาน (Resistor)	L ตัวเหนี่ยวนำ (Inductor)	C ตัวเก็บประจุ (Capacitor)
1. สัญลักษณ์			
2. ความต้านทานในวงจร หน่วยเป็น Ω	R	$X_L = \omega L = 2\pi fL$	$X_c = \frac{1}{\omega C} = \frac{1}{2\pi fC}$
3. ลักษณะทางคณิตศาสตร์	จำนวนจริง	จำนวนจินตภาพ +j	จำนวนจินตภาพ -j
4. วิธีการคำนวณ	แบบเวคเตอร์	แบบเวคเตอร์	แบบเวคเตอร์
5. ผลทางสัญญาณไฟฟ้า	I มีเฟสตรงกับ V	I มีเฟสตามหลัง V 90° หรือ I Lag V 90°	I มีเฟสนำหน้า V 90° หรือ I Lead V 90°
6. ชนิดกำลังไฟฟ้าที่ใช้	วัตต์ (W) เป็นจำนวนจริง	วาร์ (Var) เป็นจำนวนจินตภาพ +j	วาร์ (Var) เป็นจำนวนจินตภาพ -j
7. ลักษณะการใช้กำลังไฟฟ้า	ใช้แล้วหมดไป	สะสมแล้วจ่ายคืน	สะสมแล้วจ่ายคืน
8. ค่าตัวประกอบกำลังไฟฟ้า	1	0	0
9. ค่าที่วัดได้จาก Wattmeter	$= I^2R$	0 W	0 W

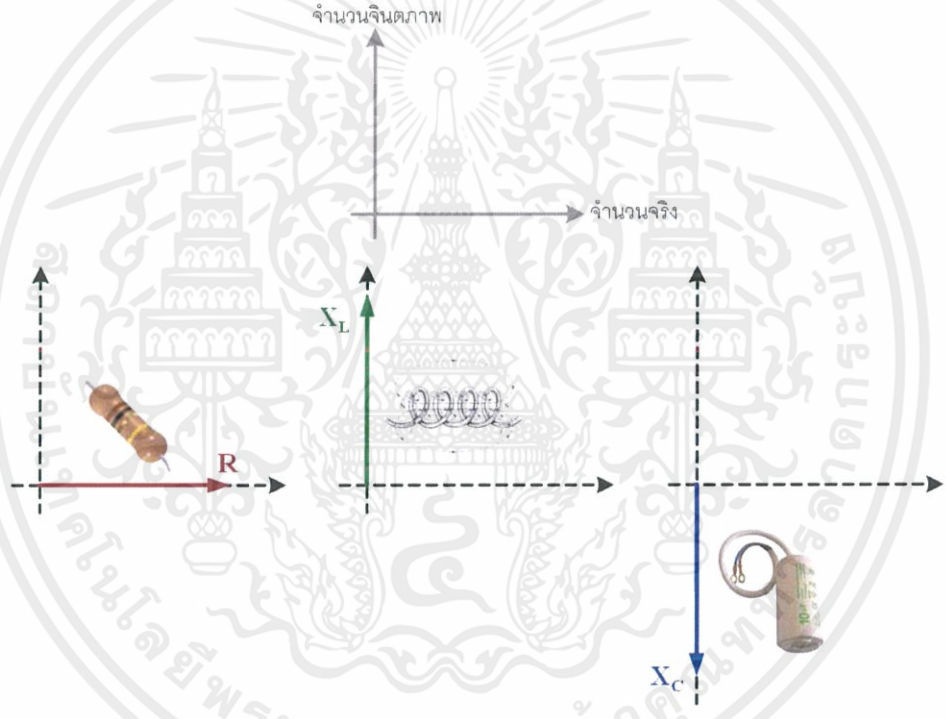
ข้อสังเกตที่พึงทราบ

- L และ C ถูกนำมาใช้ในรูปของค่า X_L และ X_C มีหน่วยเป็น Ω เช่นเดียวกับ R แต่เป็นจำนวนจินตภาพที่สามารถหักล้างกันเองได้ เพราะ X_L และ X_C มีทิศทางตรงข้ามกัน (ทิศทาง $+j$ กับ $-j$) ดังรูปที่ 2.5 ดังนั้น หากต้องการหักล้างคุณสมบัติของ X_L ก็อาจนำ X_C ที่เหมาะสมมาต่อในวงจรได้
- เพราะ X_L ทำให้สัญญาณกระแสตามหลังแรงดัน (I Lag V) 90° และ X_C ทำให้สัญญาณกระแสนำหน้าแรงดัน (I Lead V) 90° ดังนั้น ถ้าในวงจรมีคุณลักษณะเป็นการผสมกันของโหลดมากกว่า 1 กลุ่ม เช่น $R + jX_L$ หรือ $R - jX_C$ ก็จะทำให้ความต้านทาน

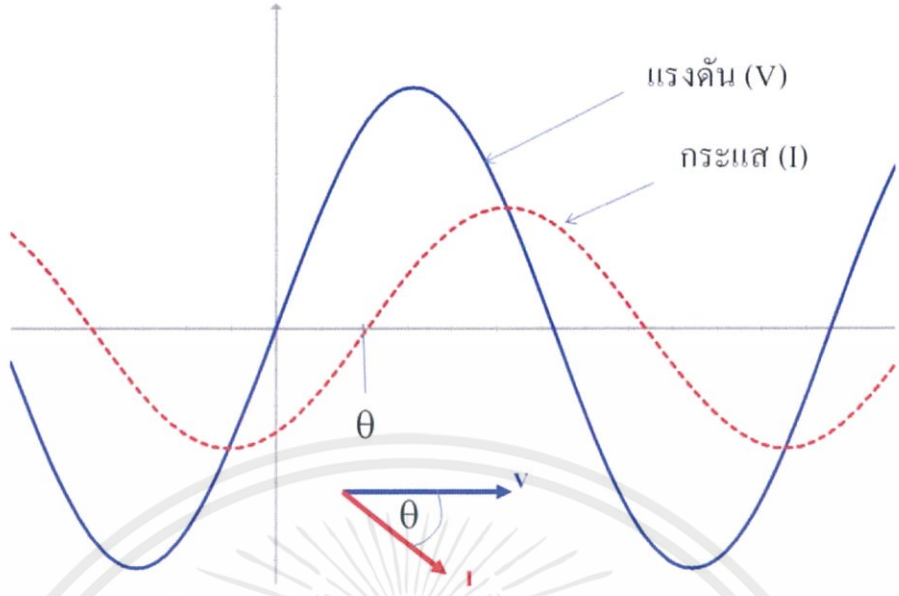
$Z = R + jX_L$ มีมุมอยู่ระหว่าง 0° ถึง + 90 ° ทำให้ I Lag V มีมุมอยู่ระหว่าง 0° ถึง - 90 °

$Z = R - jX_C$ มีมุมอยู่ระหว่าง 0° ถึง - 90 ° ทำให้ I Lead V มีมุมอยู่ระหว่าง 0° ถึง + 90 °

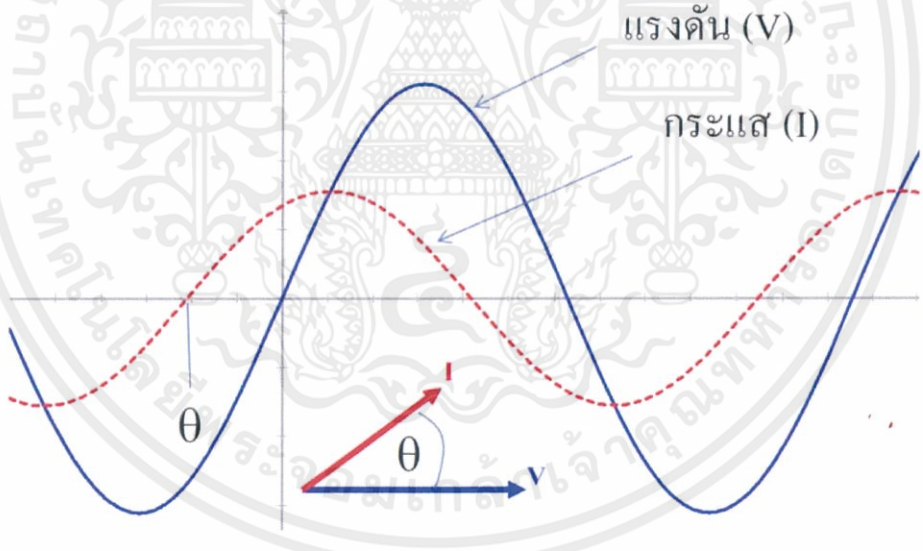
กล่าวคือ โหลด $R + jX_L$ ทำให้ I Lag V ดังรูปที่ 2.6 และโหลด $R - jX_C$ ทำให้ I Lead V ดังรูปที่ 2.7 ส่วนโหลด R นั้น I และ V มีมุมตรงกัน โดยเขียนเป็นภาพเชิงสรุปได้ดังรูปที่ 2.8



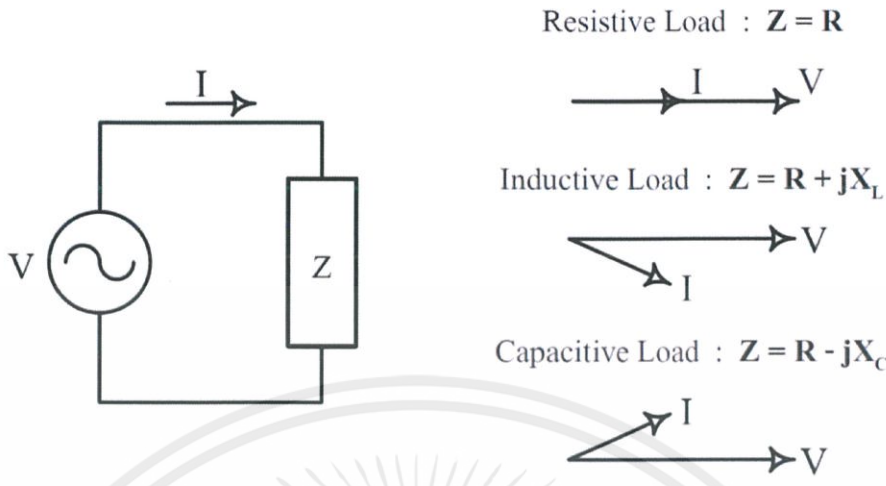
รูปที่ 2.5 ลักษณะทางเวคเตอร์ของ R X_L และ X_C [1]



รูปที่ 2.6 ลักษณะสัญญาณกระแสตามหลังแรงดัน I Lag V [1]

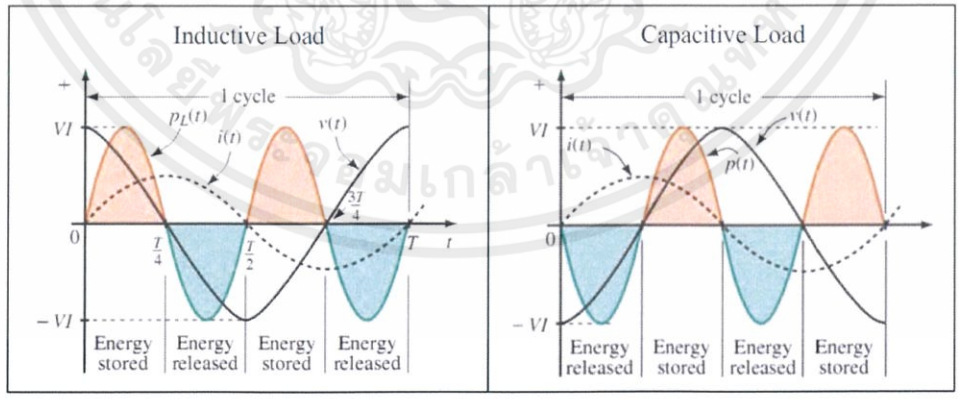


รูปที่ 2.7 ลักษณะสัญญาณกระแสนำหน้าแรงดัน I Lead V [1]



รูปที่ 2.8 ลักษณะเวคเตอร์ของ I และ V ในกรณีของโหลดชนิดต่าง ๆ [1]

- L และ C เป็นอุปกรณ์ที่ใช้กำลังไฟฟ้า 0 W (กำลังไฟฟ้าเฉลี่ย = 0 W) แต่ใช้กำลังไฟฟ้าในหน่วย VAR ซึ่งเป็นจำนวนจินตภาพ โดย VAR ของ L มีทิศทางตรงข้ามกับ VAR ของ C (สามารถหักล้างกันได้) ทั้งนี้ ทั้ง L และ C มีลักษณะการใช้กำลังไฟฟ้าคือ สะสมแล้วคืนกำลังไฟฟ้าออกมา (จึงใช้กำลังไฟฟ้า 0 W) ดังรูปที่ 2.9 ซึ่งจะสังเกตได้ว่าจังหวะของการสะสมและคืนพลังงานของโหลดทั้ง 2 ชนิดนั้นไม่ตรงกัน และสอดคล้องกับทางคณิตศาสตร์ที่ว่า VAR ของ L มีทิศทางตรงข้ามกับ VAR ของ C

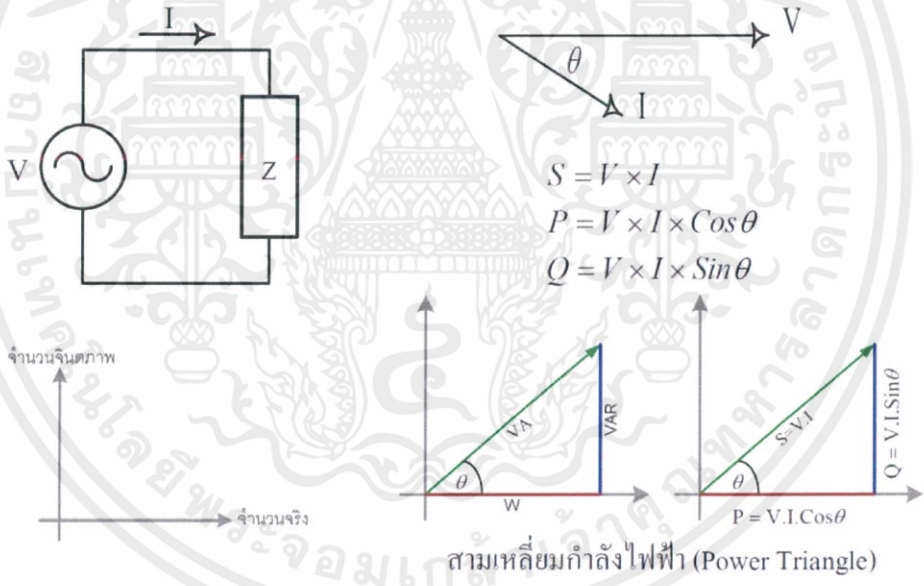


รูปที่ 2.9 ลักษณะการสะสมแล้วคืนกำลังไฟฟ้าของ L และ C [1]

2.1.4 กำลังไฟฟ้าในวงจรกระแสสลับ

วงจรไฟฟ้ากระแสสลับมีกำลังไฟฟ้า 3 ชนิด (3 หน่วย) ปกติจะคำนวณเฉพาะขนาด มีรายละเอียดดังนี้

- Average Power (P) หน่วยเป็น W คำนวณได้จาก $P = VI\cos\theta$ บ้างก็เรียกว่า Real Power
- Reactive Power (Q) หน่วยเป็น VAR คำนวณได้จาก $Q = VI\sin\theta$
- Apparent Power (S) หน่วยเป็น VA คำนวณได้จาก $S = VI = \sqrt{P^2 + Q^2}$ ดังรูปที่ 2.10
- θ คือมุมระหว่าง V กับ I ไม่เจาะจงว่า Lead หรือ Lag เป็นมุมเดียวกับ θ ในสามเหลี่ยมกำลังไฟฟ้า
- ปกติจะคำนวณเฉพาะขนาด เพราะ W และ VAR มีทิศทางที่แน่นอนบนแกนจำนวนจริงและจำนวนจินตภาพ เพียงรู้ว่าเป็น VAR ของ L (Lag) หรือเป็น VAR ของ C (Lead) ก็สามารถคำนวณได้



รูปที่ 2.10 สมการคำนวณกำลังไฟฟ้าและสามเหลี่ยมกำลังไฟฟ้า [1]

- เรียกค่า $\cos\theta$ ว่าค่าตัวประกอบกำลังไฟฟ้า (Power Factor) หรือ PF มีค่าระหว่าง 0 ถึง 1 และ $PF = \frac{S}{P}$ ด้วย
- กำลังไฟฟ้าในหน่วย W คือกำลังไฟฟ้าที่ทำให้เกิดงาน สามารถเอามาใช้ประโยชน์ได้ เป็นเลขจำนวนจริง ต่างกับกำลังไฟฟ้าในหน่วย VAR ที่เป็นจำนวนจินตภาพ (เป็นเลขจำนวนที่สมมุติขึ้น) เอามาใช้ทำงานไม่ได้

- จากรูปที่ 3.10 จะเห็นว่า ถ้า VAR มาก VA ก็จะมาก ถ้า VAR น้อย VA ก็จะน้อย เพราะ VA เป็นด้านปิดมุมฉาก โดยที่ VAR มากหรือน้อย กำลังไฟฟ้าในหน่วย W ก็ยังคงเท่าเดิม (ระบบสามารถทำงานได้เท่าเดิม)
- VAR กับ W ตั้งฉากกัน การมากขึ้นหรือน้อยลงของ VAR จึงไม่ส่งผลต่อ W แต่มีผลต่อขนาดของ VA
- ถ้าระบบใช้กำลังไฟฟ้าในหน่วย VAR มาก VA ก็จะมาก หมายความว่า ถ้า VAR มาก หรือพูดอีกนัยหนึ่งตัวประกอบกำลังไฟฟ้ามีค่าต่ำ กำลังไฟฟ้าในหน่วย VA จะมากซึ่งถ้า VA มาก ก็หมายความว่ากระแสในระบบมาก ($S = VI$ และถือว่า V คงที่) เป็นผลให้มีการสูญเสียในรูปของ I^2R ในหม้อแปลงและสายไฟมาก
- หม้อแปลงมีพิกัดกำลังเป็น VA กรณีที่ตัวประกอบกำลังไฟฟ้ามีค่าต่ำ อาจมองอีกมุมหนึ่งได้ว่า ระบบใช้กำลังไฟฟ้าเป็น VA มาก แต่ได้กำลังไฟฟ้าเป็น W น้อย หากหม้อแปลงจ่ายไฟเต็มพิกัด ก็จ่ายโหลดได้น้อยกว่ากรณีที่ VA เท่ากัน แต่ตัวประกอบกำลังไฟฟ้ามีค่าสูง (มีค่าเข้าใกล้ 1)

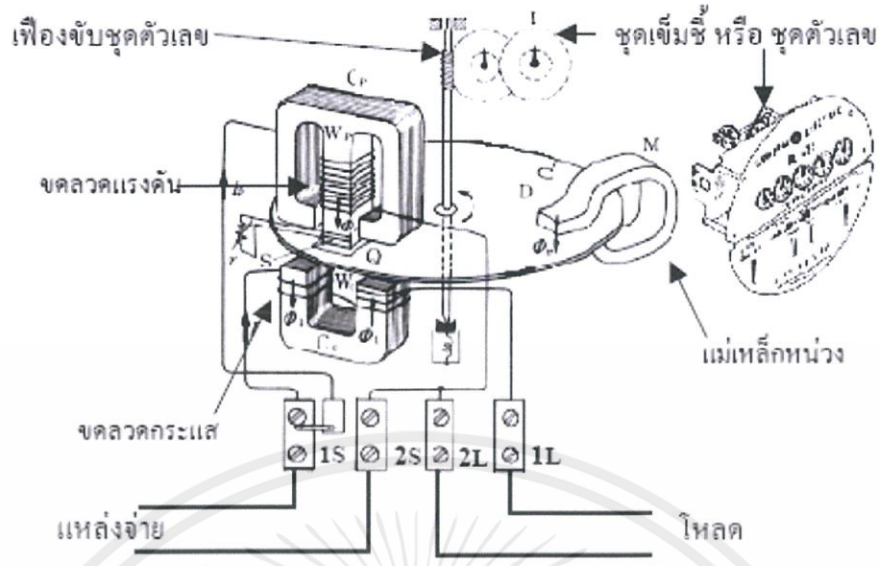
2.2 มาตรวัดพลังงานไฟฟ้า

วัตต์ฮาวร์มิเตอร์ส่วนใหญ่เป็นเครื่องวัดที่ทำงานด้วยการเหนี่ยวนำไฟฟ้าถูกสร้างขึ้นมาเพื่อวัดปริมาณกำลังไฟฟ้ากระแสสลับทั้งในบ้านเรือนและในโรงงานอุตสาหกรรม โดยมีหน่วยวัดพลังงานไฟฟ้าเป็นกิโลวัตต์ชั่วโมง (Kilowatt-hour) โดยวัตต์ฮาวร์มิเตอร์ หรือ กิโลวัตต์ฮาวร์มิเตอร์ในปัจจุบันที่นิยมใช้งานทั่ว ๆ ไปในปัจจุบันนี้ จะมี 2 ประเภทหลัก ๆ คือ

2.2.1 วัตต์ฮาวร์มิเตอร์ หรือ กิโลวัตต์ฮาวร์มิเตอร์แบบจานหมุน

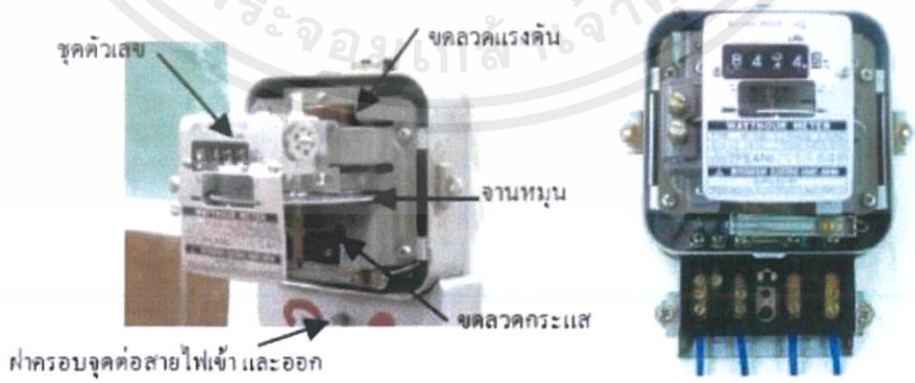
วัตต์ฮาวร์มิเตอร์ 1 เฟส (single phase watt-hour meter) แบบจานหมุนมีหลักการทำงานเหมือนกับวัตต์มิเตอร์ชนิดที่ทำงานด้วยการเหนี่ยวนำไฟฟ้า และมีส่วนประกอบที่เหมือนกันคือ ขดลวดกระแสไฟฟ้า (Current coil) และขดลวดแรงดันไฟฟ้า (Potential coil) ส่วนที่แตกต่างกันก็คือในวัตต์มิเตอร์จะแสดงค่าด้วยการบ่ายเบนของเข็มชี้ ซึ่งใช้ชี้ค่าบนสเกลส่วนวัตต์ฮาวร์มิเตอร์จะแสดงค่าโดยใช้แม่เหล็กเหนี่ยวนำให้เกิดกระแสไหลวนทำให้จานหมุนและใช้ชุดเฟืองไปขับชุดตัวเลขหรือชุดเข็มชี้ให้แสดงค่าออกมาบนหน้าปัทม์

1) โครงสร้าง ดังรูปประกอบด้วยขดลวดกระแสต่ออนุกรมกับโหลด และขดลวดแรงดันต่อขนานกับโหลด ขดลวดทั้งสองชุดจะพันอยู่บนแกนเหล็กที่ออกแบบโดยเฉพาะและมีจานอะลูมิเนียมบาง ๆ ยึดติดกับแกนหมุน วางอยู่ในช่องว่างระหว่างขดลวดทั้งสอง



รูปที่ 2.11 โครงสร้างของวัตต์ฮาวร์มิเตอร์ 1 เฟส แบบเหนี่ยวนำไฟฟ้า [1]

2) หลักการทำงานขดลวดกระแสและขดลวดแรงดันทำหน้าที่สร้างสนามแม่เหล็กส่งผ่านไปยังจานอะลูมิเนียมที่วางอยู่ระหว่างขดลวดทั้งสอง ทำให้เกิดแรงดันไฟฟ้าเหนี่ยวนำและมีกระแสไหลวน (Eddy current) เกิดขึ้นในจานอะลูมิเนียม แรงต้านระหว่างกระแสไหลวนและสนามแม่เหล็กของขดลวดแรงดันจะทำให้เกิดแรงผลักขึ้น จานอะลูมิเนียมจึงหมุนไปได้ ที่แกนของจานอะลูมิเนียมจะมีเฟืองติดอยู่ เฟืองนี้จะไปขับชุดตัวเลขที่หน้าปัทม์ของเครื่องวัด แรงผลักที่เกิดขึ้นจะเป็นสัดส่วนระหว่างความเข้มของสนามแม่เหล็กของขดลวดแรงดันและกระแสไหลวนในจานอะลูมิเนียม และขึ้นอยู่กับจำนวนรอบของขดลวดด้วย ส่วนจำนวนรอบการหมุนของจานอะลูมิเนียมขึ้นอยู่กับการใช้พลังงานไฟฟ้าของโหลด สำหรับรูป เป็นโครงสร้างของกิโลวัตต์อาวร์มิเตอร์ 1 เฟสของการไฟฟ้าส่วนภูมิภาค

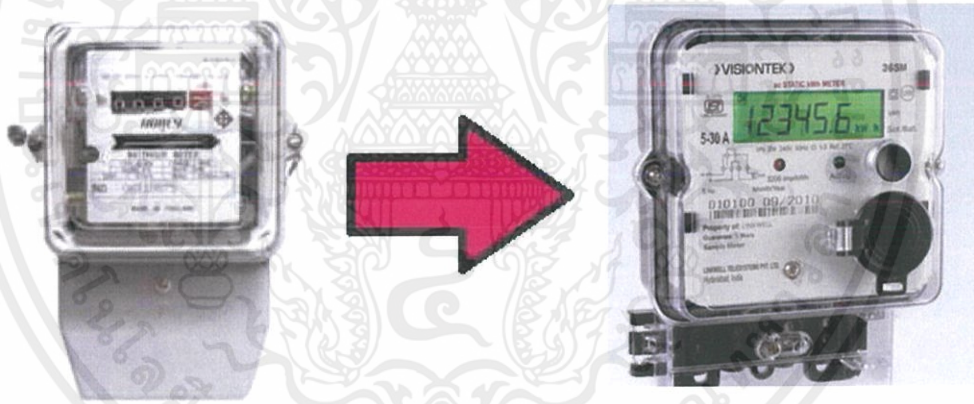


รูปที่ 2.12 กิโลวัตต์อาวร์มิเตอร์ของการไฟฟ้าส่วนภูมิภาค [1]

3) การนำไปใช้งาน การต่อวัตต์ฮาวร์มิเตอร์หรือกิโลวัตต์ฮาวร์มิเตอร์เพื่อใช้วัดปริมาณพลังงานไฟฟ้า ดังรูปโดยด้านที่ต่อกับแหล่งจ่ายจะมีตัวเลขกำกับไว้ คือ 1S และ 2S ส่วนด้านที่ต่อไปยังโหลดจะมีตัวเลขกำกับไว้คือ 1L และ 2L ตัวอักษร S ย่อมาจากคำว่า “Supply” หมายถึงด้านที่จ่ายไฟเข้า ส่วนอักษร L ย่อมาจากคำว่า “Load” หมายถึงด้านที่ต่อกับโหลดไฟฟ้า ส่วนตัวเลข 1 หมายถึงต่อกับสายไฟ (Line) และเลข 2 หมายถึง สายนิวทรอล (Neutral)

2.2.2 วัตต์ฮาวร์มิเตอร์ หรือ กิโลวัตต์ฮาวร์มิเตอร์แบบดิจิทัล

ในปัจจุบันวัตต์ฮาวร์มิเตอร์ หรือกิโลวัตต์ฮาวร์มิเตอร์ ได้ถูกใช้อย่างแพร่หลายในกลุ่มลูกค้าทางอุตสหกรรม ซึ่งมีปริมาณการใช้พลังงานไฟฟ้าที่สูง (แบบ 3 เฟส) โดยการทำงานของวัตติฮาวร์มิเตอร์แบบดิจิทัลแบบอตุสาหกรรม จะใช้การเชื่อมต่ออุปกรณ์วัดกระแสและรูปแบบสัญญาณไฟฟ้าแบบดิจิทัลที่ทำงานโดยใช้หลักการเหนี่ยวนำสนามแม่เหล็กไฟฟ้า เพื่อเก็บค่าการวัดทางไฟฟ้าแบบดิจิทัล แล้วนำมาคำนวณหาค่าพารามิเตอร์ทางไฟฟ้าต่าง ๆ จาก Microcontroller ที่มีประสิทธิภาพสูงด้วยกระบวนการประมวลผลสัญญาณดิจิทัล (DSP) ยกตัวอย่างเช่น ปริมาณการบริโภคพลังงาน ความต่างศักย์ กระแส ความถี่ เป็นต้น



รูปที่ 2.13 วัตต์ฮาวร์มิเตอร์แบบจานหมุน (ซ้าย) และวัตต์ฮาวร์มิเตอร์แบบดิจิทัล (ขวา) [2]

สำหรับการใช้งานในภาคครัวเรือน (แบบ 1 เฟส) ด้วยข้อจำกัดทางต้นทุนและความต้องการจำนวนมาก วัตต์ฮาวร์มิเตอร์แบบดิจิทัลที่ใช้นั้นไม่จำเป็นต้องมีฟังก์ชันที่มากเท่าแบบที่ใช้ในทางอตุสาหกรรม จึงใช้หลักการทำงานโดยการใช้เซ็นเซอร์วัดกระแสและความต่างศักย์แบบต่อตรงเข้ากับสายส่งกำลังไฟฟ้าที่จ่ายเข้ากับบ้านเรือน เพื่อใช้วัดปริมาณการบริโภคพลังงานไฟฟ้าสำหรับการคิดค่าบริการของการไฟฟ้าฯ นอกจากนี้แล้ว วัตต์ฮาวร์มิเตอร์แบบดิจิทัลยังสามารถนำไปพัฒนาต่อในด้านนวัตกรรม Internet of Things (IoT) ได้อีกด้วย

โดยวัตต์ฮาวร์มิเตอร์หรือกิโลวัตต์ฮาวร์มิเตอร์แบบดิจิทัลนั้นสามารถแบ่งประเภทหลัก ๆ ได้ 3 ประเภท คือ

1) วัตต์ฮาวร์มิเตอร์ดิจิทัล

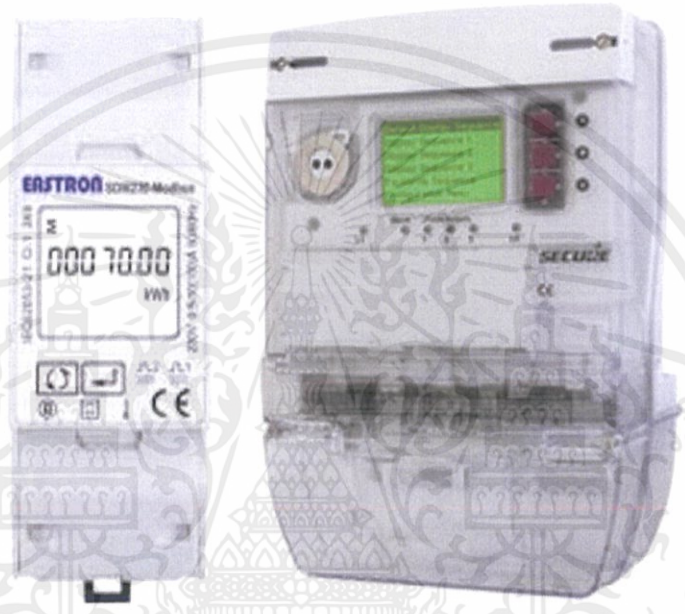
โดยที่วัตต์ฮาวร์มิเตอร์แบบนี้นั้น จะมีฟังก์ชันสำหรับการใช้งานคล้าย ๆ กับวัตต์ฮาวร์มิเตอร์แบบจานหมุน แต่จะมีการนำเซ็นเซอร์วัดความต่างศักย์ไฟฟ้าและเซ็นเซอร์วัดกระแสไฟฟ้า เข้ามาทำงานแทนการหมุนของจานหมุนจากสนามแม่เหล็ก และจะใช้การบันทึกปริมาณการใช้งานพลังงานไฟฟ้าลงหน่วยความจำภายใน แทนการใช้ระบบเฟืองทดรอบในการหมุนตัวเลขปริมาณการใช้งาน ซึ่งวัตต์ฮาวร์มิเตอร์แบบดิจิทัล จะมีความแม่นยำสูงกว่าวัตต์ฮาวร์มิเตอร์แบบจานหมุน เนื่องจากวัตต์ฮาวร์มิเตอร์แบบจานหมุนนั้น จะมีความความไวต่อการรบกวนจากสภาพแวดล้อมได้มากกว่า เช่น สนามแม่เหล็กไฟฟ้าจากภายนอก ความเอียงในแนวดิ่งของการติดตั้ง เป็นต้น ซึ่งสิ่งเหล่านี้จะไม่เป็นการรบกวนต่อวัตต์ฮาวร์มิเตอร์แบบดิจิทัล

รูปที่ 2.14 วัตต์ฮาวร์มิเตอร์แบบดิจิทัล

(ที่มา : http://www.visiontek.co.in/energy-meters/36sm.html)

2) วัตต์ฮาวร์มิเตอร์ดิจิทัลแบบมีช่องทางการสื่อสารภายนอก

วัตต์ฮาวร์มิเตอร์ดิจิตัลชนิดนี้ จะมีความสามารถที่เพิ่มขึ้นมาจากวัตต์ฮาวร์ดิจิทัลจากในรูปที่ 2.14 คือวัตต์ฮาวร์มิเตอร์ชนิดนี้ จะมีช่องทางการเชื่อมต่อสำหรับรับส่งข้อมูลกับอุปกรณ์ภายนอกได้ เช่น ผ่านทางสายสัญญาณ ผ่านทาง Wireless ผ่านทาง Bluetooth เป็นต้น ซึ่งข้อดีของวัตต์ฮาวร์มิเตอร์ชนิดนี้ จะสามารถส่งข้อมูลไปยังอุปกรณ์อื่น ๆ เพื่อนำข้อมูลไปใช้ในการประมวลผลการปริมาณการใช้งานไฟฟ้า สำหรับแจ้งค่าใช้บริการ หรือใช้ในการเก็บสถิติการใช้งานได้



รูปที่ 2.15 วัตต์ฮาวร์มิเตอร์ดิจิทัลแบบมีช่องทางการสื่อสารภายนอก

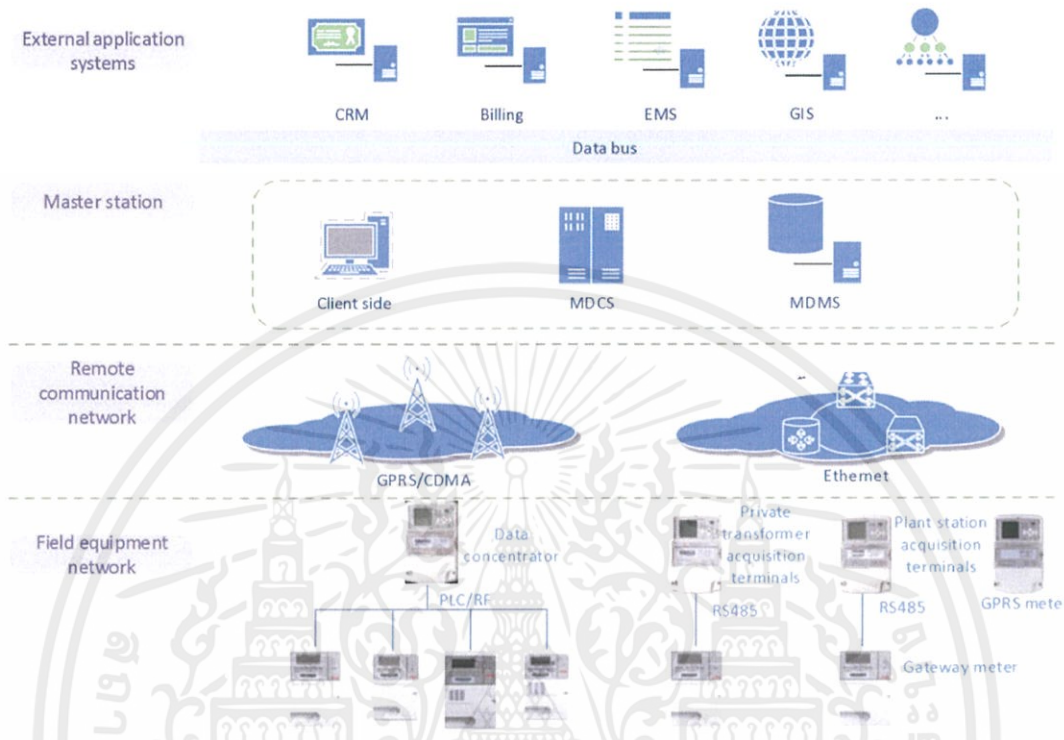
(ที่มา : http://www.eastrongroup.com/productsview/136.html)

3) วัตต์ฮาวร์มิเตอร์ดิจิทัลแบบ AMR และ AMI

วัตต์ฮาวร์มิเตอร์ดิจิทัลชนิดนี้เป็นหนึ่งในอุปกรณ์ที่ใช้ในเทคโลยี Smart Grid ซึ่งเป็นโครงข่ายไฟฟ้าที่ใช้เทคโนโลยีการสื่อสาร เข้ามาบริหารจัดการพลังงานไฟฟ้า เช่น การประมาณการผลิตไฟฟ้าในแต่ละช่วงเวลาให้เพียงพอต่อการใช้งาน การเลือกแหล่งทรัพยากรสำหรับการผลิตไฟฟ้า เป็นต้น โดยระบบ Smart Grid จะสามารถเชื่อมต่อเข้ากับวัตต์ฮาวร์มิเตอร์แบบ AMR (Automatic Meter Reading) และ AMI (Advance Metering Infrastructure) Prepaid Meter ซึ่งมีความสามารถในการวัดปริมาณการใช้งานในแต่ละช่วงเวลา ผ่านตัวกลางต่าง ๆ เช่น สายนำสัญญาณ หรือผ่านระบบโทรศัพท์เคลื่อนที่ GSM ทำให้นำข้อมูลปริมาณการใช้งานของผู้ใช้มาคิดค่าบริการได้โดยไม่ต้องใช้การส่งพนักงานออกไปอ่านค่ามิเตอร์

วัตต์ฮาวร์มิเตอร์ดิจิทัลชนิดนี้ จะทำการสื่อสารกับระบบต้นทางของผู้ให้บริการผ่านตัวกลางต่าง ๆ ไปยัง Gateway ขอระบบ ที่ทำหน้าที่เป็นอุปกรณ์จัดเก็บและประมวลผลข้อมูลผู้ใช้งาน และ

ทำการส่งต่อไปให้ยังระบบ MDMS (Meter Data Management System) ดังรูปที่ 2.16 เพื่อนำข้อมูลไปใช้ประโยชน์ในการบริหารจัดการต่อไป



รูปที่ 2.16 โครงสร้างการสื่อสารของวัตต์ฮาวร์มิเตอร์ดิจิทัลแบบ AMI

(ที่มา : http://www.topsmetering.com/topscomm-ami-solution_n17)

รูปที่ 2.17 วัตต์ฮาวร์มิเตอร์ดิจิทัลแบบ AMI

(ที่มา : http://forthmeter.blogspot.com/2013/07/forth-amr.html)

2.3 IoT (Internet of Things)

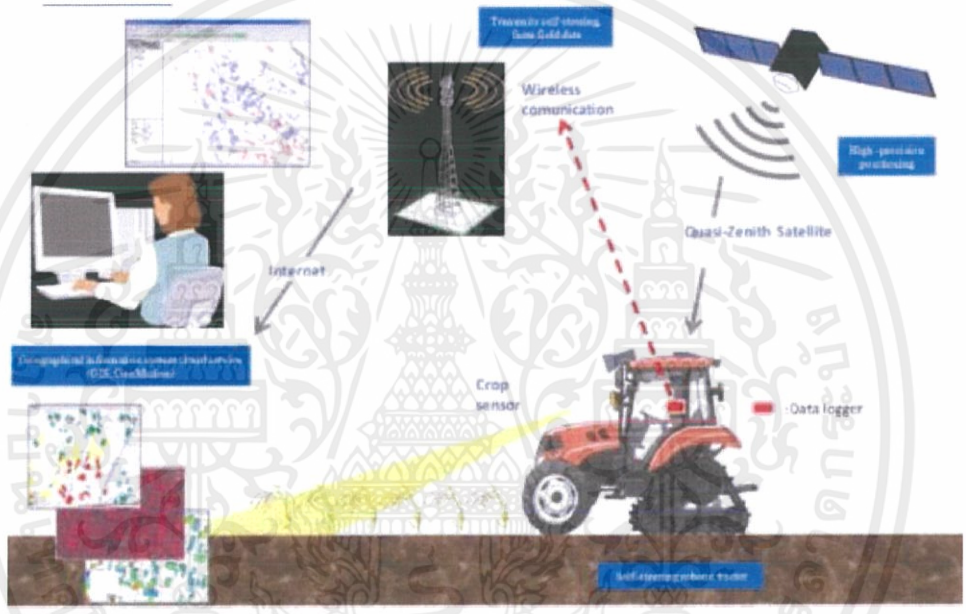
Internet of Things หรือ IoT หมายถึง สิ่งของต่าง ๆ ที่ถูกนำมาเชื่อมต่อเป็นเครือข่าย เพื่อรับส่งข้อมูลสำหรับทำการแลกเปลี่ยน หรือทำการประมวลผล หรือเพื่อควบคุมการทำงานของสิ่งของต่าง ๆ ที่ ถูกเชื่อมต่อไว้ สิ่งของที่ที่ได้กล่าวไว้นั้น หมายถึง “Things” หรือ “Device” อาจจะเป็นเครื่องใช้ในสำนักงาน เครื่องมือเครื่องจักรกลการเกษตร เครื่องจักรในโรงงานอุตสาหกรรม หุ่นยนต์แขนกล อุปกรณ์ภายในอาคารหรือที่พักอาศัย อุปกรณ์เครื่องใช้ในชีวิตประจำวัน เช่น นาฬิกา รถยนต์ ตู้เย็น อุปกรณ์แจ้งเตือนภัย ฯลฯ โดยสิ่งที่กล่าวถึงนั้น จะมีเซ็นเซอร์ซึ่งมีหน้าที่คอยตรวจจับปริมาณทางสภาวะแวดล้อมต่าง ๆ มีระบบเครือข่ายทำการติดต่อสื่อสารส่งข้อมูลกันระหว่างอุปกรณ์ (Machine to machine communication) เสมือนหนึ่งว่าอุปกรณ์มีการติดต่อสื่อสารกันได้เอง โดยอาศัยโปรโตคอล (Protocol) และระบบเครือข่าย เช่น ในระบบ Smart Farm เมื่อระบบอ่านค่าเซ็นเซอร์ตรวจจับความชื้นของดิน เมื่อพบว่าดินมีสภาพที่แห้ง ก็จะส่งสัญญาณให้ระบบรดน้ำทำงาน ฯลฯ ซึ่งได้มีการประเมินว่าอุปกร์ IoT จะเติบโตและมีการใช้งานรวมสูงถึง 20,000 ล้านตัวในปี 2020 โดยมีมูลค่าสูงถึง 3 ล้านล้านเหรียญสหรัฐ

รูปที่ 2.18 การเชื่อมต่อสิ่งของ (Things/Device) เป็นเครือข่าย IoT

(ที่มา : https://semielectronics.com/sensors-lifeblood-internet-things)

เทคโนโลยี IoT ทำให้อุปกรณ์หรือสิ่งของต่าง ๆ สามารถติดต่อสื่อสารกันเองได้ ดังนั้น IoT จึงเข้ามามีบทบาทกับการดำรงชีวิตประจำวันของมนุษย์ในแง่ของความสะดวกสบาย ความรวดเร็ว ความปลอดภัยในหลายแขนง ได้แก่

- ด้านการเกษตร การนำเทคโนโลยีสมัยใหม่เข้ามาผสมผสานเข้ากับงานด้านการเกษตร หรือ Smart Farm ช่วยเพิ่มผลผลิตและคุณภาพการเกษตร เช่น การติดตั้งเซ็นเซอร์ตรวจสภาพอากาศ สภาพดิน แล้วรายงานเข้าสู่ระบบ เพื่อให้ผู้ดูแลหรือระบบบริหารจัดการควบคุมสภาพแวดล้อมให้เหมาะสม เช่น ระบบรดน้ำพืชผลอัตโนมัติ เปิดพัดลมระบายอากาศ จากรูปที่ 2.19 เป็นระบบ Smart Farm โดย Yanmar และ Hitachi ซึ่งระบบดังกล่าวประกอบด้วยรถไถไร้คนขับ (Self-steering Tractor) ทำหน้าที่ไถหรือเตรียมผิวดิน หว่านต้นกล้า และเก็บเกี่ยวโดยอัตโนมัติ โดยระบบจะมีการเชื่อมต่อกับระบบ GPS เพื่อกำหนดตำแหน่งให้รถวิ่งไปตามพื้นที่ส่วนต่าง ๆ และมีการรายงานสถานะมายังห้องควบคุมผ่านการเชื่อมต่อแบบไร้สาย



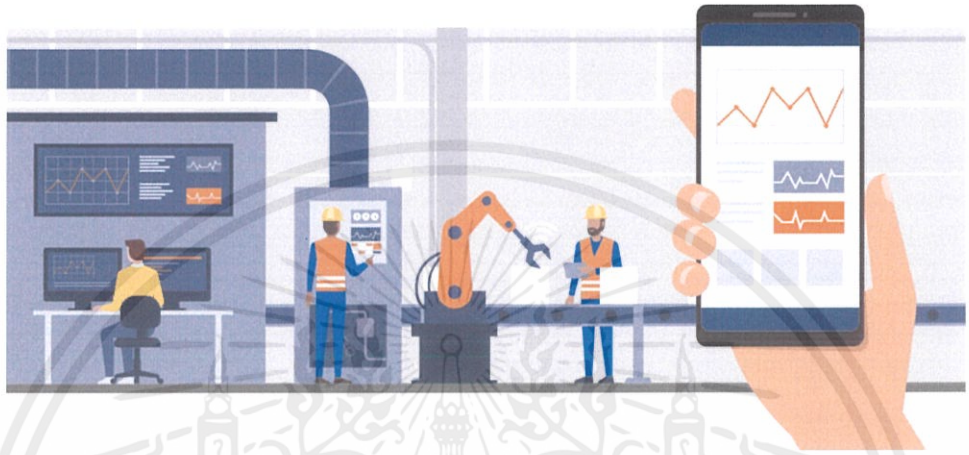
รูปที่ 2.19 ตัวอย่างการใช้งานด้านการเกษตร

(ที่มา www.yanmar.com)

- ด้านสุขภาพและการแพทย์ ช่วยแจ้งเตือนหรือดูแลเกี่ยวกับสุขภาพ เช่น เซ็นเซอร์ตรวจวัดชีพจร หรือค่าอื่น ๆ ที่จำเป็นต่อผู้ที่ต้องการดูแลเป็นพิเศษ
- ด้านการศึกษา การจัดการเรียนการสอนโดยใช้อุปกรณ์ RFID NFC การบันทึกคะแนน ระบบการบันทึกและตรวจสอบการมาเรียนของผู้เรียน
- ด้านการเงิน การใช้เงินสดระบบอิเล็กทรอนิกส์ ลดขั้นตอนหรือลดคนที่จะต้องดำเนินการลดการจัดการระหว่างผู้ซื้อและผู้ขาย
- ด้านการรักษาความปลอดภัย โดยใช้ระบบตรวจจับตำแหน่ง การติดตั้งระบบ GPS (Global Positioning System) ยานหนะแล้วแจ้งเตือนมายังสมาร์ทโฟน การใช้ RFID ID Card หรือ

สายรัดข้อมือเพื่อตรวจสอบผู้ที่เข้ามาในสถานที่ต่าง ๆ เพื่อป้องกันคนแปลกหน้าเข้ามาในสถานที่สำคัญ การใช้เงินสดระบบอิเล็กทรอนิกส์

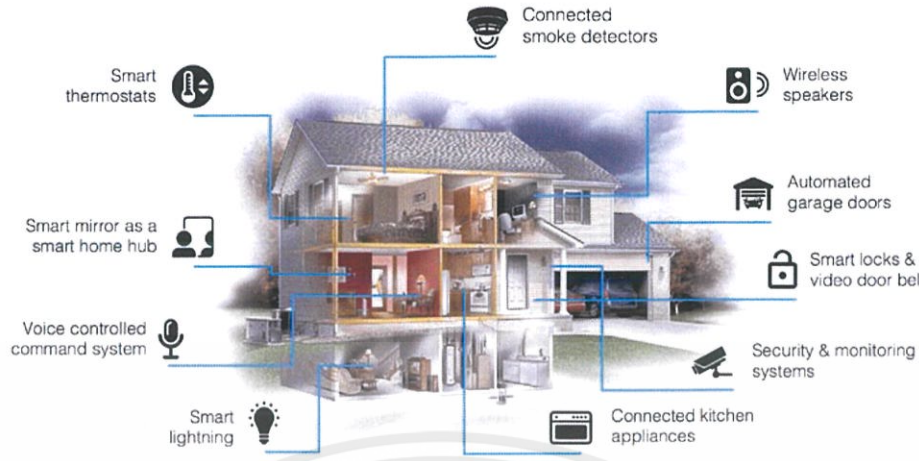
- ด้านอุตสาหกรรมและการผลิต อุปกรณ์อัตโนมัติและเซ็นเซอร์ช่วยบันทึกและวิเคราะห์ข้อมูลเพื่อเพิ่มผลผลิตและลดต้นทุน ดังรูปที่ 2.20



รูปที่ 2.20 ตัวอย่างการใช้งานด้านอุตสาหกรรม

(ที่มา : https://www.alliedtelesis.com/blog/digital-transformation-industry-40-hype-roi)

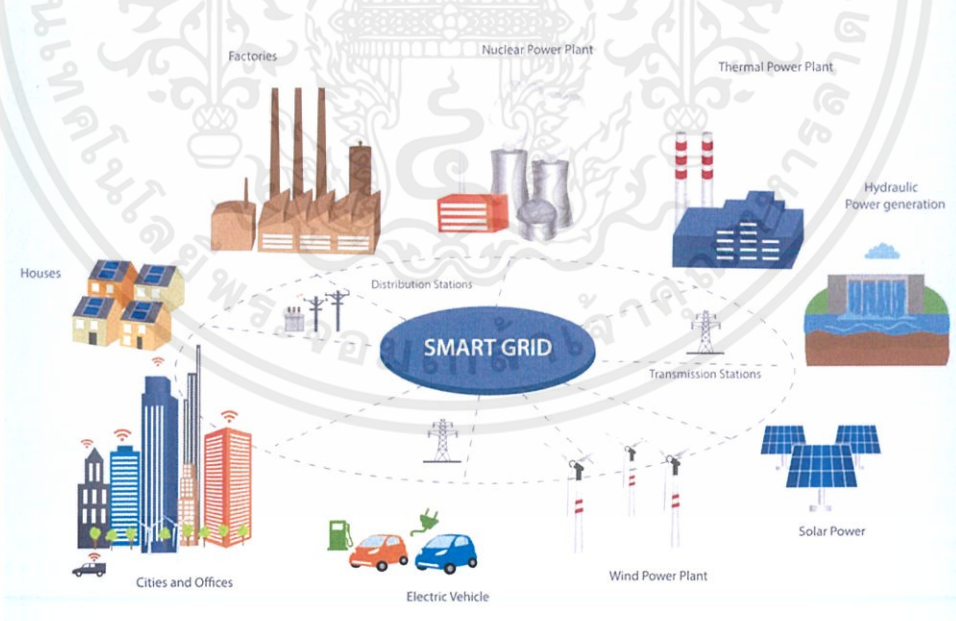
- Smart City นำเทคโนโลยีมาปรับใช้เพื่อทำให้คุณภาพชีวิตของประชากรดีขึ้น เช่น การจัดการพลังงานไฟฟ้า ระบบจัดการน้ำ ระบบจัดการขยะ เป็นต้น
- Smart Home นำเทคโนโลยีมาควบคุมอุปกรณ์ต่าง ๆ ภายในหรือภายนอกบ้าน เพื่อให้เกิดความสะดวกสบายและความปลอดภัย เช่น ระบบปิดเปิดประตูอัตโนมัติ เซ็นเซอร์ตรวจจับการเคลื่อนไหว การเตือนภัยผู้บุกรุก การเปิดปิดอุปกรณ์ไฟฟ้าอัตโนมัติ ระบบหมุนเวียนระบายอากาศอัตโนมัติ ระบบเปิดไฟแสงสว่างอัตโนมัติในยามกลางคืนเมื่อผู้อยู่อาศัยตื่นขึ้นเพื่อเข้าห้องน้ำ ดังรูปที่ 2.21 เป็นต้น



รูปที่ 2.21 ตัวอย่างการใช้งานภายในบ้านที่อยู่อาศัย

(ที่มา : https://www.dirror.com/en/smart-home-infographic)

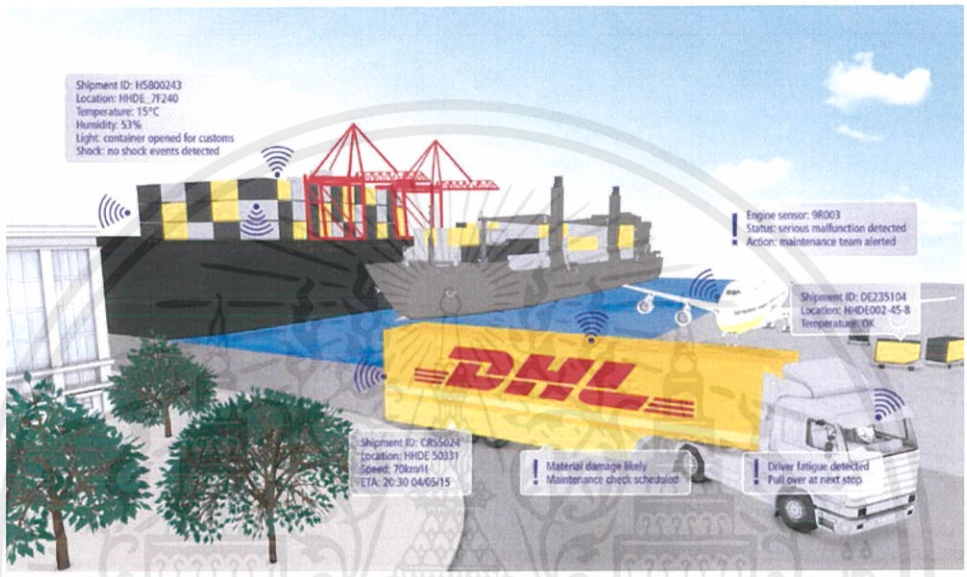
- Smart Grid โครงข่ายไฟฟ้าอัจริยะ เป็นการใช้เทคโนโลยีสารสนเทศและการสื่อสารมาบริหารจัดการควบคุมการผลิต ส่ง และจ่ายพลังงานไฟฟ้า ดังรูปที่ 2.22



รูปที่ 2.22 ตัวอย่างการใช้งานในระบบ Smart Grid ในด้านต่าง ๆ

(ที่มา : https://brandinside.asia/smart-meter-smart-grid-iot-technology)

- ด้าน Logistic เป็นการนำเทคโนโลยีมาช่วยในด้านคมนาคม การขนส่งสินค้า เช่น จากรูปที่ 2.23 เป็นการใช้เทคโนโลยี Iot มาทำหน้าที่รายงานสถานะการขนสิ่งสินค้าของ DHL ซึ่งมีการรายงานสถานะของตู้สินค้าว่าขณะนี้อยู่ที่ไหนแล้ว ในตู้มีอุณหภูมิและความชื้นเป็นเท่าใด ปกติหรือไม่ บางตู้อาจจะติดเซ็นเซอร์เพื่อรายงานหรือเตือนว่ามีการสั่นสะเทือนหรือการกระแทกที่จะทำให้สินค้าเสียหายหรือไม่



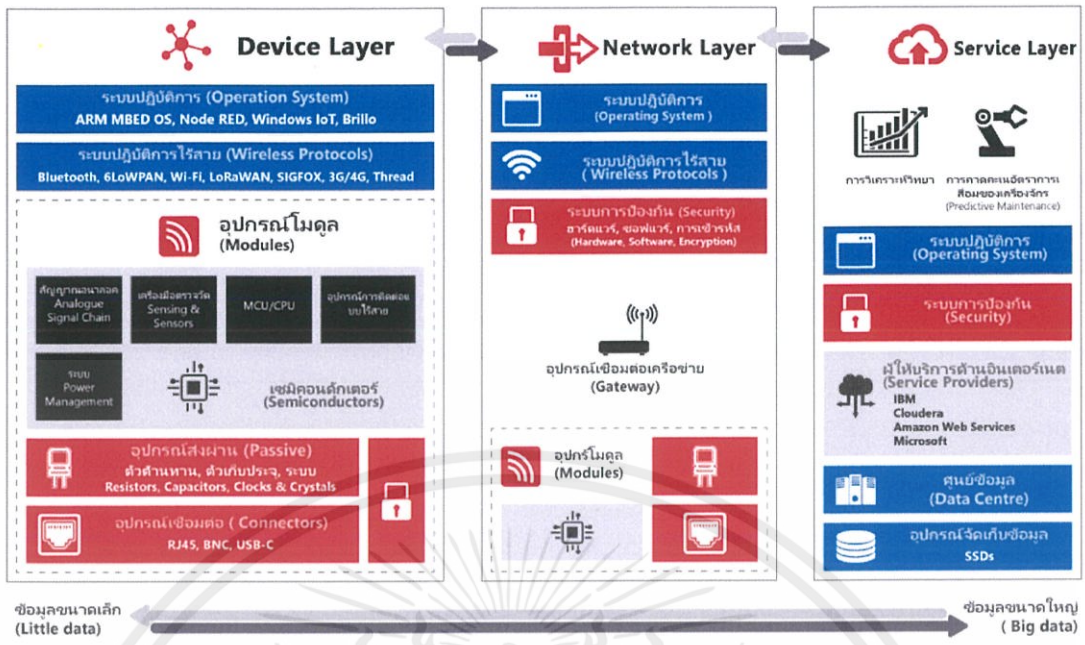
รูปที่ 2.23 ตัวอย่างการใช้งานด้านการขนส่ง

(ที่มา : https://delivering-tomorrow.com)

สถาปัตยกรรม IoT

สถาปัตยกรรม IoT พื้นฐานเบื้องต้น สามารถแบ่งตามองค์ประกอบการทำงานของอุปกรณ์ต่าง ๆ เป็น Layer ได้ดังนี้

- Device Layer เป็นชั้นของอุปกรณ์ต่าง ๆ (Physical Things/Device) เช่น บอร์ดหรือฮาร์ดแวร์ IoT เซ็นเซอร์ และตัวกระทำการหรือตัวทำงาน (Actuators) เช่น มอเตอร์เซอร์โว ตัวปิดเปิดกลอน รีเลย์ หลอดไฟ กระดิ่ง เตาไฟฟ้า อุปกรณ์เปล่งเสียง ฯลฯ
- Network Layer เป็นชั้นของการเชื่อมต่อผ่านระบบเครือข่าย โดยส่วนใหญ่ IoT มักใช้งานระบบเครือข่ายแม่ข่ายที่อยู่ในชั้น Service Layer ได้แก่ ข้อมูลที่อ่านจากเซ็นเซอร์ ข้อมูลที่ผ่านการประมวลผลแล้วและส่งไปควบคุมอุปกรณ์
- Service Layer เป็นชั้นของเครื่องให้บริการ คล้าย ๆ กับเป็นเครื่องแม่ข่าย ที่เชื่อมต่อสื่อสารกับ Device/Things ต่าง ๆ เข้าด้วยกัน โดยในชั้นนี้อาจมี Application สำหรับการจัดการระบบรวมด้วยก็ได้ เช่น หน้าควบคุมหรือแสดงผลข้อมูล (Control Panel/Dashboard) ฯลฯ



รูปที่ 2.24 สถาปัตยกรรม IoT แบบลำดับตาม Layer [5]

ตัวอย่างเช่น ในโรงงานแห่งหนึ่ง ประกอบด้วย IoT ระบบเซ็นเซอร์และรบบระบายความร้อนที่ทำงานโดยการปั้มน้ำออกจากแหล่งเก็บน้ำที่อยู่ห่างไกลส่งมาหล่อเย็นเครื่องจักร เมื่ออุณหภูมิสูงผิดปกติ ระบบระบายความร้อนก็จะปั้มน้ำส่งมาระบายความร้อนเครื่องจักร ลักษณะการทำงานคือ เซ็นเซอร์วัดอุณหภูมิเครื่องจักร (จัดว่าอยู่ใน Device Layer) ทำการวัดค่าอุณหภูมิของเครื่องจักรขณะนั้น แล้วส่งข้อมูลผ่านชั้นเครือข่าย (Network Layer) ไปยังเครื่องให้บริการอยู่ใน Service Layer จากข้อมูลดังกล่าวก็จะถูกส่งไปประมวลผล หรือส่งต่อไปยังปั้มเพื่อทำการปั้มสูบน้ำส่งมาระบายความร้อนเครื่องจักร ฯลฯ

โดยรวมแล้ว สถาปัตยกรรม IoT ถ้าเปรียบเทียบแล้ว ก็อาจคล้ายกับสถาปัตยกรรมไคลเอ็นต์/เซิร์ฟเวอร์ของระบบคอมพิวเตอร์ โดย Device/Things ในชั้น Device Layer ก็คือเครื่องไคลเอ็นต์ ส่วนเครื่องให้บริการในชั้น Service Layer ก็จะทำหน้าที่คล้ายกับเครื่องแม่ข่ายเซิร์ฟเวอร์นั่นเอง

2.4 NB-IoT (Narrow Band Internet of Things)

ในยุคแรกของ IoT การเชื่อมต่อเครือข่ายแบบไร้สายจะอาศัยระบบเครือข่ายไร้สายระยะใกล้ ได้แก่ Bluetooth Wi-Fi และระบบเครือข่ายระยะไกลด้วยโมดูลเชื่อมต่อกับสัญญาณโทรศัพท์เคลื่อนที่ (GSM/GPRS) แต่ปัจจุบันนี้ การเชื่อมต่อเครือข่ายของ IoT มีตัวเลือกมากขึ้น โดยในไทยมีเทคโนโลยี LoRa และ NB-IoT ที่สามารถเชื่อมต่อระยะไกลได้ถึง 5 ถึง 10 กิโลเมตร และใช้พลังงานต่ำสำหรับอุปกรณ์ IoT ซึ่งจะแตกต่างจากการใช้โมดูลโทรศัพท์เคลื่อนที่ ที่ค่อนข้างใช้พลังงานเยอะและใช้แบนด์วิธที่กว้างกว่า โดยทั่วไปแล้ว อุปกรณ์ IoT ไม่ได้ต้องการการใช้อัตราการรับส่งข้อมูลที่มาก เพราะการนำมาประยุกต์ใช้งานของอุปกรณ์ IoT ส่วนใหญ่แล้วจะนำมาใช้เพียงการรับส่งค่า

ของเซ็นเซอร์ต่าง ๆ หรือส่งคำสั่งในการสั่งงานอุปกรณ์ IoT ดังนั้นจึงไม่จำเป็นต้องใช้โมดูลโทรศัพท์เคลื่อนที่ ที่มีอัตราการรับส่งข้อมูลสูง เนื่องจากทำให้เปลืองทรัพยากรโดยไม่จำเป็น NB-IoT จึงเป็นทางเลือกในการประยุกต์การใช้งานที่แท้จริงของ IoT

NB-IoT (Narrow Band Internet of Things) เป็นมาตรฐานที่จัดอยู่ในกลุ่ม Low Power Wide Area Neetwork (LPWAN) ถูกพัฒนาโดยกลุ่มมาตรฐานโทรคมนาคม (3GPP) ดังตารางที่ 2.3 ซึ่งมีจุดประสงค์เพื่อใช้สำหรับการเชื่อมต่ออุปกรณ์ IoT ต่าง ๆ เข้าสู่อินเตอร์เน็ตโดยใช้เครือข่ายโทรศัพท์มือถือ จุดมุ่งหมายของ NB-IoT คือการที่อุปกรณ์ประเภทส่งข้อมูลอัตราความเร็วต่ำ (Low Data Rate) และใช้เครือข่ายเดียวกับเครือข่ายเซลลูลาร์มือถือ ถ้ามองง่าย ๆ อุปกรณ์ IoT ที่เชื่อมต่อกับ NB-IoT ก็เทียบได้กับโทรศัพท์มือถือเล็ก ๆ เครื่องหนึ่งที่รับส่งสัญญาณด้วยความเร็วที่ต่ำลงนั่นเอง โดยใช้เสาหรือสถานีฐานตัวเดียวกับโทรศัพท์มือถือ

ตารางที่ 2.3 มาตรฐาน 3GPP Narrowband Cullular Standards

	LTE Cat 1	LTE Cat 0	Lte Cat M1 (eMTC)	LTE Cat NB1 (NB-IoT)	EC-GSM-IoT
3GPP Release	Release 8	Release 12	Release 13	Release 13	Release 13
Downlink Peak Rate	10 Mbit/s	1 Mbit/s	1 Mbit/s	250 kbit/s	474 kbit/s (EDGE)
Uplink Peak Rate	5 Mbit/s	1 Mbit/s	1 Mbit/s	250 kbit/s (multi-tone) 20 kbit/s (single-tone)	474 kbit/s (EDGE) 2 Mbit/s (EGPRS2B)
Latency	50 – 100 ms	not deployed	10 – 15 ms	1.6 – 10 s	700 ms – 2 s
No. of Antennas	2	1	1	1	1 - 2
Duplex Mode	Full Duplex	Full/Half Duplex	Full/Half Duplex	Half Deplex	Half Duplex
Device Receive Bandwidth	1.4–20 MHz	1.4–20 MHz	1.4 MHz	180 kHz	200 kHz
Recevier Chains	2 (MIMO)	1 (SISO)	1 (SISO)	1 (SISO)	1 - 2
Device Transmit Power	23 dBm	23 dBm	20/23 dBm	20/23 dBm	23/33 dBm

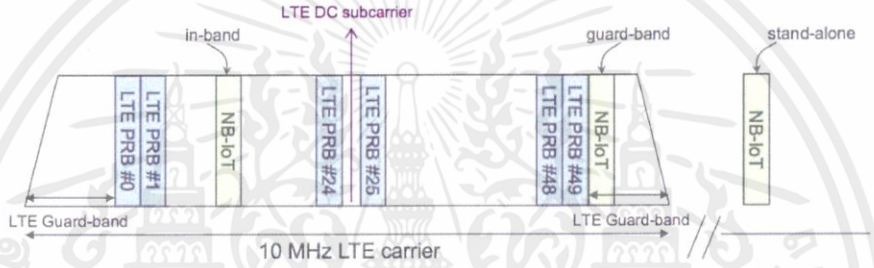
(ที่มา : https://en.wikipedia.org/wiki/Narrowband_IoT)

จะเห็นได้ว่า NB-IoT นั้นเป็นมาตรฐานย่อยของมาตรฐาน LTE (Long Term Evolution) ที่ออกโดย 3GPP โดยการนำ LTE นั้นมาจำกัดแบนด์ โดยที่หนึ่งย่านความถี่ของ NB-IoT จะมีแบนด์วิธอยู่ที่ 200 kHz ในสัญญาณขา Downlink นั้นจะมีการมอดูเลตแบบ OFDM (Orthogonal frequency-division multiplexing) และสัญญาณในขา Uplink นั้นจะมีการมอดูเลตแบบ SC-FDMA โดยมาตรฐาน NB-IoT จะมีโหมดการทำงานอยู่ 3 โหมด ดังรูปที่ 2.25 ได้แก่

- **Stand alone** เป็นการแยกคลื่นความถี่ออกมาให้บริการ NB-IoT โดยตรงโดยไม่ใช้สัญญาณในช่วงที่ให้บริการ LTE อยู่มาใช้

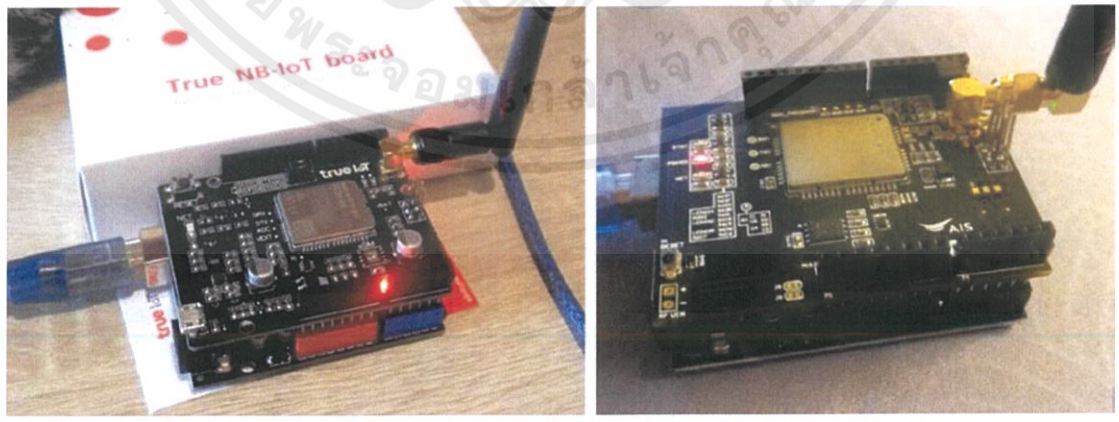
- **Guard band** เป็นการนำช่องสัญญาณของระบบ LTE มาให้บริการ NB-IoT โดยจะนำช่องสัญญาณในช่วงที่เป็น Guard band ของระบบ LTE มาให้บริการเป็นช่องสัญญาณ NB-IoT
- **In-band** เป็นการนำช่องสัญญาณ LTE ที่มีการแบ่งออกเป็น Sub-carrier ย่อย ๆ มาให้บริการช่องสัญญาณ NB-IoT

ซึ่งการเลือกใช้โหมดการให้บริการนั้นจะขึ้นอยู่กับอุปกรณ์ของสถานีฐาน ว่ารองรับมาตรฐาน NB-IoT ในโหมดการทำงานใด แต่สำหรับในประเทศไทยแล้ว ผู้ให้บริการเครือข่ายโทรศัพท์มือถือที่มีการให้บริการ NB-IoT นั้นมีเพียง 2 เจ้า ได้แก่ TRUE และ AIS โดยทั้งสองเจ้านั้นได้ให้บริการ NB-IoT ผ่านคลื่นความถี่ 900 MHz ในโหมด Guard band ทั้งคู่



รูปที่ 2.25 โหมดการทำงานของมาตรฐาน LTE Cat NB1 (NB-IoT) [6]

โดยในปัจจุบันได้มีผู้ให้บริการที่มีการขายอุปกรณ์ NB-IoT และให้บริการเครือข่าย NB-IoT ในประเทศไทยอยู่ 2 ราย ได้แก่ TRUE และ AIS ซึ่งได้วางจำหน่าย NB-IoT Shield ที่สามารถนำไปใช้งานได้กับ Microcontroller รุ่น Arduino UNO R3 ดังรูปที่ 2.26



รูปที่ 2.26 NB-IoT Shield สำหรับ Arduino TRUE (ซ้าย) AIS (ขวา)

(ที่มา : https://playelek.com)

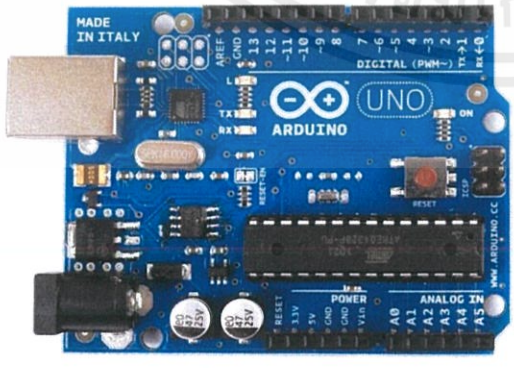
2.5 Arduino Microcontroller

Arduino เป็น Platform ต้นแบบด้านอิเล็กทรอนิกส์แบบ Open source ซึ่งใช้ Hardware และ Software ที่ยืดหยุ่นและใช้งานง่าย มีไว้สำหรับนักเรียนนักศึกษา นักออกแบบ ผู้ที่สนใจเป็นงานอดิเรกและทุกคนที่สนใจในการสร้างอุปกรณ์ที่ตอบสนองต่อการสั่งงาน หรือทำงานด้วยตนเองได้ เดิมที Arduino ก่อตั้งมาด้วยผู้ร่วมก่อตั้งทั้งหมด 5 คน คือ Massimo Banzi, David Cuartielles,David Mellis, Tom Igoe, และ Gianluca Martino โดยเริ่มโครงการมาตั้งแต่ช่วงปี ค.ศ. 2005 ความหมายของคำว่า Arduino แปลว่า เพื่อนแท้ (Strong friend หรือ Brave friend) ในภาษาอิตาลี โดยผู้ก่อตั้งมีความตั้งใจให้ราคาของอุปกรณ์นั้นถูกเมื่อเทียบกับ Microcontroller ตระกูลอื่น ๆ เพื่อให้ทุกคนสามารถเข้าถึงได้โดยง่าย Platform Arduino ได้ออกแบบมาเพื่อให้ใช้งานง่าย ผู้ใช้งานไม่จำ เป็นต้องมีความรู้เกี่ยวกับโครงสร้างสถาปัตยกรรมภายในซีพียูโดยรู้เพียงว่าบอร์ด Arduino ที่เลือกมาใช้งานนั้นมีขาที่ใช้งานอะไรบ้างมีคุณสมบัติต่าง ๆ อะไรบ้างก็สามารถใช้งานได้ ด้วยประสบการณ์และจำนวนการใช้งานของผู้ใช้นั้นมีจำนวนมาก Arduino จึงถูกใช้งานด้านต่าง ๆ มากมาย เนื่องจากการเขียนโปรแกรมสำหรับควบคุมการทำงานของ Arduino มีความง่ายและยืดหยุ่นสามารถใช้งานในระดับสูงได้อีกด้วย เครื่องมือที่ใช้สาหรับเขียนโค้ดควบคุมมีเวอร์ชั่นที่สามารถรันได้ในทุกระบบปฏิบัติการไม่ว่าจะเป็น Macintosh Windows หรือแม้กระทั่ง Linux ก็ตาม ทำให้ได้รับความนิยมเป็นอย่างสูง Platform Arduino ประกอบไปด้วย

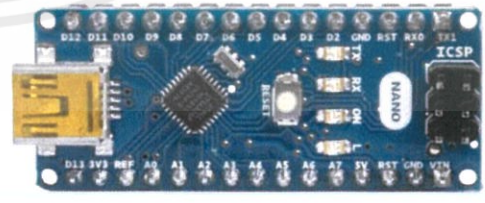
1. Hardware
2. Software

2.5.1 Hardware

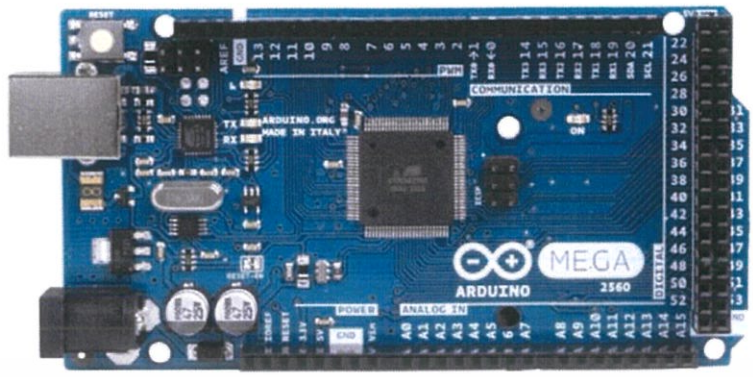
Arduino เป็นแผ่นวงจรอิเล็กทรอนิกส์ขนาดเล็กที่มี Microcontroller เป็นชิ้นส่วนหลักประกอบร่วมกับอุปกรณ์อิเล็กทรอนิกส์อื่น ๆ เพื่อให้ง่ายต่อการใช้งานหรือที่เรียกรวมกันว่า “บอร์ด Arduino” โดยบอร์ด Arduino ก็มีหลายรุ่นให้เลือกใช้ตามความเหมาะสมของงาน โดยในแต่ละรุ่นอาจมีความแตกต่างกันในเรื่องของขนาดของบอร์ด หรือสเปค เช่น จำนวนของขารับส่งสัญญาณ แรงดันไฟที่ใช้ ประสิทธิภาพของ MCU เป็นต้น ซึ่งมีตัวอย่างบางรุ่นดังรูปที่ 2.27



(ก) บอร์ด Arduino UNO R3



(ข) บอร์ด Arduino Nano



(ค) บอร์ด Arduino Mega 2560

รูปที่ 2.27 ตัวอย่างบอร์ด Arduino รุ่นต่าง ๆ [7]

Arduino ถูกใช้ประโยชน์ในลักษณะเดียวกับ MCU คือ ใช้ติดต่อสื่อสารและควบคุมอุปกรณ์ไฟฟ้าอื่นๆ ด้วยการเขียนโปรแกรมให้กับ MCU เพื่อควบคุมการรับส่งสัญญาณทางไฟฟ้าตามเงื่อนไขต่าง ๆ ซึ่งแต่ระรุ่นจะมีการใช้ MCU ในรุ่นต่าง ๆ กันไปทำให้ความสามารถของแต่ละรุ่นมีความแตกต่างกัน โดยมีรายละเอียดดังตารางที่ 2.4

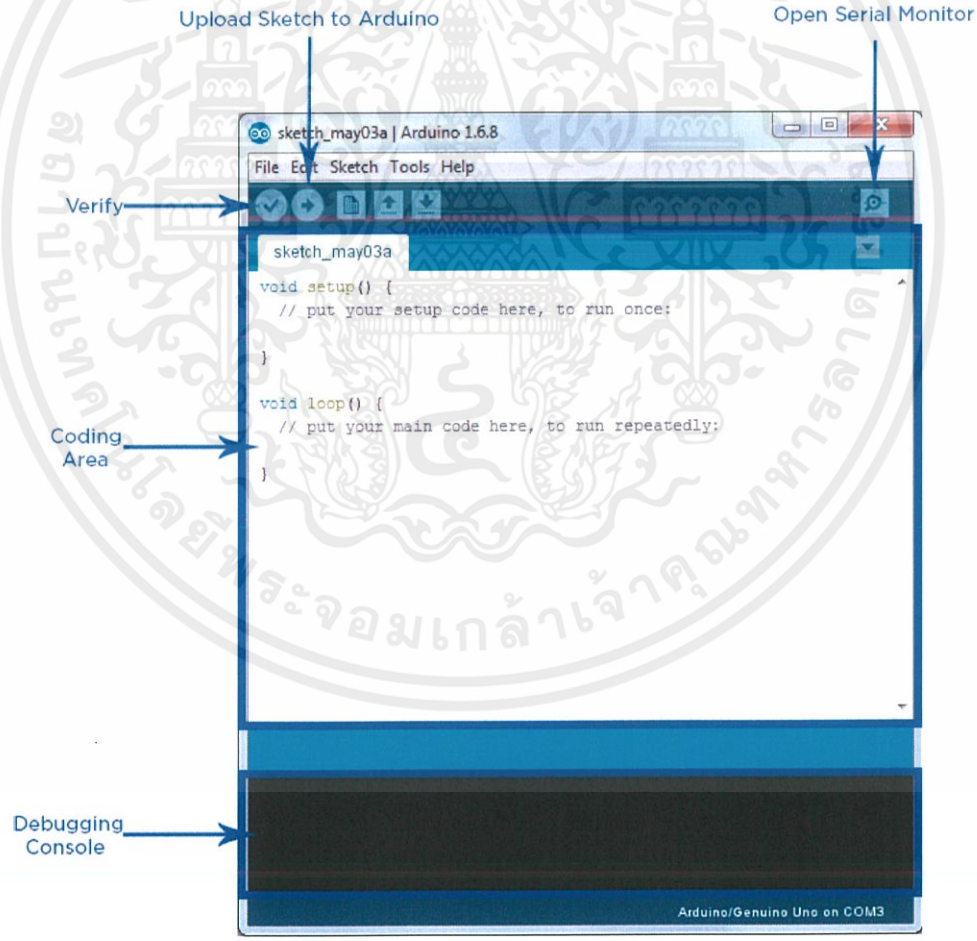
ตารางที่ 2.4 ตารางเปรียบเทียบคุณสมบัติของบอร์ด Arduino [7]

Name	Processor	Operating Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
101	Intel® Curie	3.3 V / 7-12V	32MHz	6 / 0	14 / 4	-	24	196	Regular	-
Gemma	ATtiny85	3.3 V / 4-16 V	8 MHz	1 / 0	3 / 2	0.5	0.5	8	Micro	0
LilyPad	ATmega168V	2.7-5.5 V	8MHz	6 / 0	14 / 6	0.512	1	16	-	-
	ATmega328P	2.7-5.5 V								
LilyPad SimpleSnap	ATmega328P	2.7-5.5 V	8 MHz	4 / 0	9 / 4	1	2	32	-	-
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16 / 0	54 / 15	4	8	256	Regular	4
Micro	ATmega32U4	5 V / 7-12 V	16 MHz	12 / 0	20 / 7	1	2.5	32	Micro	1
Pro	ATmega168	3.3 V / 3.35-12 V	8 MHz	6 / 0	14 / 6	0.512	1	16	-	1
	ATmega328P	5 V / 5-12 V	16 MHz			1	2	32		
Pro Mini	ATmega328P	3.3 V / 3.35-12 V	8 MHz	6 / 0	14 / 6	1	2	32	-	1
		5 V / 5-12 V	16 MHz							
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6 / 0	14 / 6	1	2	32	Regular	1
Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6 / 1	14 / 10	-	32	256	2 Micro	2
Due	ATSAM3X8E	3.3 V / 7-12 V	84 MHz	12 / 2	54 / 12	-	96	512	2 Micro	4
Esplora	ATmega32U4	5 V / 7-12 V	16 MHz	-	-	1	2.5	32	Micro	-
Ethernet	ATmega328P	5 V / 7-12 V	16 MHz	6 / 0	14 / 4	1	2	32	Regular	-
Leonardo	ATmega32U4	5 V / 7-12 V	16 MHz	12 / 0	20 / 7	1	2.5	32	Micro	1
Mega ADK	ATmega2560	5 V / 7-12 V	16 MHz	16 / 0	54 / 15	4	8	256	Regular	4
Mini	ATmega328P	5 V / 7-9 V	16 MHz	8 / 0	14 / 6	1	2	32	-	-
Nano	ATmega168	5 V / 7-9 V	16 MHz	8 / 0	14 / 6	0.512	1	16	Mini	1
	ATmega328P					1	2	32		

2.5.2 Software

การเขียนโปรแกรมควบคุมการทำงาน Arduino Microcontroller มักจะเขียนด้วยภาษาซี ซึ่งเป็นภาษาโปรแกรมที่มีความยืดหยุ่นมากกว่าและสามารถพัฒนางานได้เร็วกว่าการเขียนโปรแกรมด้วยภาษา Assembly การเขียนโปรแกรมเพื่อควบคุมการทำงานของ Arduino ยึดหลักวิธีการเขียนตามไวยากรณ์ภาษา C ดังนั้นเมื่อสามารถเขียนโปรแกรมควบคุมการทำงาน Microcontroller Arduino นี้ได้ก็สามารถนำความรู้ไปเขียนโปรแกรมภาษาซี Microcontroller อื่น ๆ ได้

Arduino ถูกใช้ประโยชน์ในลักษณะเดียวกับ MCU คือ ใช้ติดต่อสื่อสารและควบคุมอุปกรณ์ไฟฟ้าอื่นๆ ด้วยการเขียนโปรแกรมให้กับ MCU เพื่อควบคุมการรับส่งสัญญาณทางไฟฟ้าตามเงื่อนไขต่าง ๆ โดยจะใช้ Software Arduino IDE ดังรูปที่ 2.28 ในการเขียนโปรแกรมคำสั่งการทำงานของบอร์ด Arduino ซึ่ง Arduino IDE เป็น IDE (Integrated Development Environment) ที่ได้รับความนิยมอย่างแพร่หลาย อันเนื่องมาจากสามารถเพิ่ม Library ต่าง ๆ ที่มีคนอื่นพัฒนาไว้ก่อนแล้วมาพัฒนาต่อ ทำให้เกิดสังคมแห่งการแบ่งปันทรัพยากรทาง Software ที่มีขนาดใหญ่มาก



รูปที่ 2.28 ส่วนประกอบของ Software Arduino IDE

(ที่มา : https://core-electronics.com.au/tutorials/arduino-ide-tutorial.html)

ตัวอย่างการประยุกต์ใช้ Arduino ในชีวิตประจำวัน เช่น ระบบเปิด/ปิดไฟในบ้านอัตโนมัติ ระบบรดน้ำต้นไม้อัตโนมัติ ระบบเปิด/ปิดประตูอัตโนมัติ ระบบเครื่องซักผ้าหยอดเหรียญ หรือ ใช้ควบคุมความเร็วและทิศทางการหมุนของคุมมอเตอร์ เป็นต้น

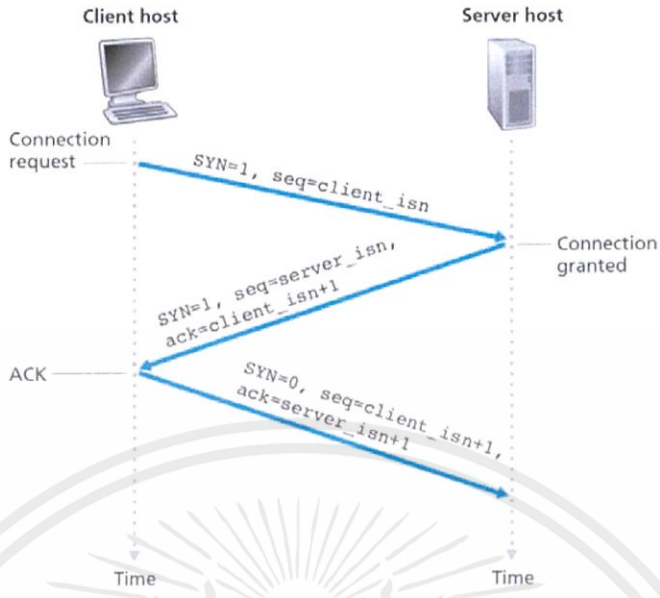
รูปที่ 2.29 ตัวอย่างการใช้บอร์ด Arduino ในการสร้างระบบรดน้ำต้นไม้อัตโนมัติ

(ที่มา : http://wiznetmuseum.com/portfolio-items/plant-watering-with-arduino)

2.6 TCP (Transmission Control Protocol)

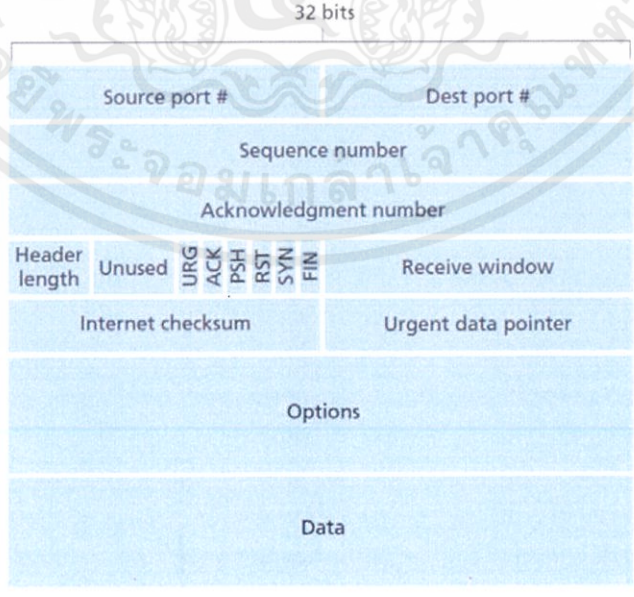
ในปัจจุบันนี้ การรับส่งข้อมูลที่มีพื้นฐานบนความน่าเชื่อถือนั้นหลีกเลี่ยงไม่พ้นที่จะใช้โปรโตคอล TCP ซึ่งเป็น Internet Transport-layer ใน OSI Model และ TCP/IP Model ซึ่งมีการเชื่อมต่อที่มีเส้นทางที่แน่นอน น่าเชื่อถือ

TCP ถูกกล่าวว่าเป็นการเชื่อมต่อแบบ Connection-oriented เพราะว่าก่อนที่แอพลิเคชันที่ใช้งานจะทำการส่งข้อมูลได้นั้น ต้องมีขั้นตอนในการเริ่มต้นการทำงานทั้งหมด 2 ขั้นตอน คือ ขั้นแรกจะมีการทำ “Three-way Handshake” ดังรูปที่ 2.30 ซึ่งเป็นการที่ผู้ส่งและผู้รับทำการส่งส่วนข้อของมูลหรือชุดข้อมูลที่ทั้งสองได้ทำการตกลงกันไว้แล้วและทราบรูปแบบของข้อมูลทั้งผู้ส่งและผู้รับเพื่อเป็นการยืนยันว่าสามารถรับส่งข้อมูลได้ จากนั้นขั้นตอนถัดมาถึงจะเริ่มทำการส่งข้อมูลจริง ๆ ในเส้นทางเดิมที่ได้ทำการจัดตั้งไว้แล้วตามลำดับของข้อมูล และเมื่อทำการรับส่งข้อมูลเสร็จแล้ว ก็จะมีการส่ง Acknowledge เพื่อยืนยันว่าข้อมูลที่ส่งไปแล้วนั้น ได้ส่งถึงผู้รับแล้ว



รูปที่ 2.30 การทำ Three-way handshake ของ TCP [8]

การเชื่อมต่อแบบ TCP ไม่ใช่การเชื่อมต่อแบบ end-to-end TDM (Time Division Multiplex) หรือ FDM (Frequency Division Multiplex) ที่เป็นรูปแบบวงจรใน Switched network รวมถึงไม่ใช่ Virtual circuit ด้วย เนื่องจาก TCP ทำงานอยู่บนระดับระบบเท่านั้น ไม่ได้เป็นส่วนหนึ่งของอุปกรณ์เครือข่าย โดยที่ TCP จะทำการแยกข้อมูลออกเป็นส่วน ๆ เรียกว่า Segment ก่อนที่จะทำการส่งออกไปพร้อมกับ TCP Header ซึ่งมีโครงสร้างดังรูปที่ 2.31

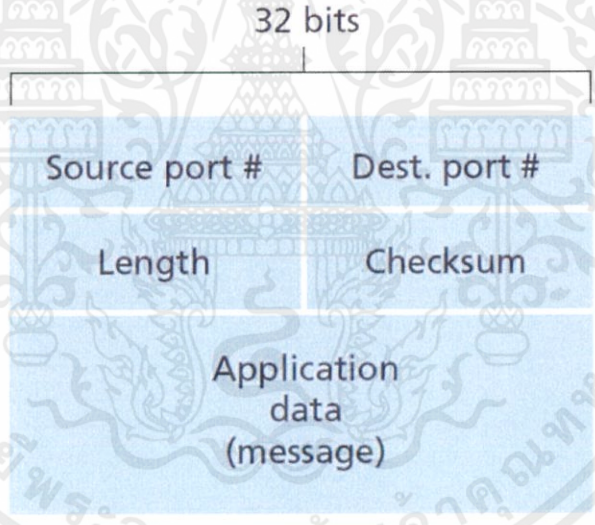


รูปที่ 2.31 โครงสร้างของ TCP Segment [8]

เนื่องจากการรับส่งข้อมูลในรูปแบบ TCP นั้นมีความต้องการความน่าเชื่อถือในการรับส่งข้อมูล จึงต้องมีขั้นตอนในการสร้างเส้นทางที่ตายตัว และมีการยืนยันการได้รับข้อมูลที่ผู้ส่งได้ทำการส่งไปในทุกขั้นตอน จึงทำให้อาจจะมีการเกิด Buffer ในการส่งข้อมูล ส่งผลให้ข้อมูลที่ส่งไปไม่ต่อเนื่อง จึงเหมาะกับการนำมาประยุกต์ใช้ในการรับส่งข้อมูลที่ต้องการความถูกต้อง เช่น การประยุกต์ใช้เข้ากับโปรโตคอล HTTP ของ WWW เป็นต้น

2.7 UDP (User Datagram Protocol)

UDP เป็นรูปแบบการรับส่งข้อมูลที่มีขนาดเล็กน้ำหนักเบา โดยจะมีการแบ่งข้อมูลออกเป็นส่วนเล็ก ๆ เรียกว่า Datagram ซึ่งมีโครงสร้างดังรูปที่ 2.32 UDP เป็น Internet Transport-layer ใน OSI Model และ TCP/IP Model มีการเชื่อมต่อที่ไม่แน่นอน และไม่น่าเชื่อถือ เนื่องจากข้อมูลที่ถูกส่งโดย UDP จะไม่มีการยืนยันว่าข้อมูลที่ถูกส่งนั้นได้ไปถึงปลายทางเรียบร้อยแล้ว ซึ่งหากต้องการให้สามารถยืนยันได้ก็ต้องกำหนดให้ผู้รับส่งข้อความตอบกลับมาว่าได้รับข้อมูลเรียบร้อยแล้ว



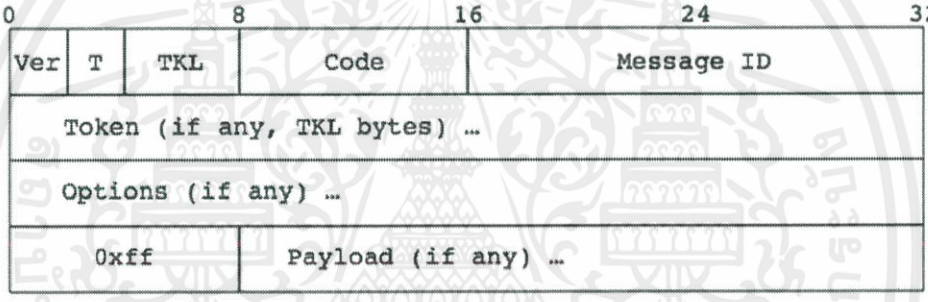
รูปที่ 2.32 โครงสร้างของ UDP Datagram [8]

UDP มีจุดเด่นที่ความเร็ว ขนาดเล็ก และไม่มีการทำงานเกี่ยวการส่งข้อมูลซ้ำหรือคำนวณอัตราการส่งข้อมูล ซึ่งจะเหมาะกับการส่งข้อมูลแบบ Realtime (เรียลไทม์) เช่น ระบบ Voice over IP ระบบ Video Streaming เป็นต้น ซึ่งข้อมูลที่สูญหายบางส่วนหรือข้อมูลที่เกิด Delay (ดิเลย์) จะถูกละความสนใจไปมันจะส่งข้อมูลได้เร็วกว่า แบบ TCP และจะไม่มีการสร้างการเชื่อมต่อที่ตายตัวเกิดขึ้น ทำให้ข้อมูลที่วิ่งในเครือข่ายมีน้อยลงด้วยเป็นการสื่อสารแบบ Connection-less คือข้อมูลจะถูกแบ่งเป็นชิ้น ๆ ตามที่อยู่ปลายทาง แล้วผ่านตัวกลางไปยังปลายทาง อาจจะใช้เส้นทางคนละเส้นทางกันก็ได้ ไม่เหมือนกับ TCP ที่ต้องส่งไปเส้นทางเดิมที่จัดตั้งไว้ตามลำดับของข้อมูล รวมทั้ง

ข้อมูลแต่ละชิ้นอาจจะถึงก่อนหลังแตกต่างกันไปได้ด้วย ทำให้การเริ่มต้นส่งทำได้รวดเร็ว ไม่ต้องเสียเวลาสร้างการเชื่อมต่อ

2.8 CoAP (Constrained Application Protocol)

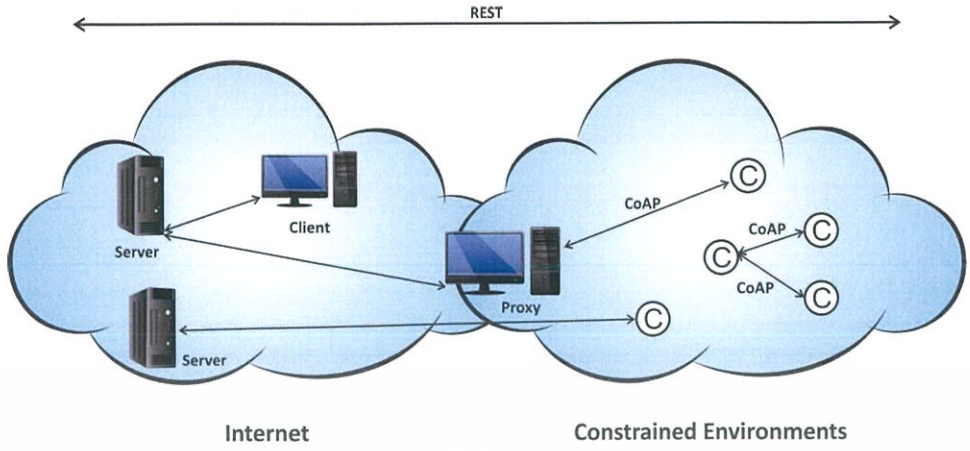
CoAP เป็นมาตรฐานที่ถูกพัฒนาขึ้นมาใหม่โดย IETF (Internet Engineering Task Force) ในปี 2014 โดยถูกออกแบบให้คล้ายกับโปรโตคอล HTTP ซึ่งเป็น Document transfer protocol แต่มีโครงสร้างดังรูปที่ 2.33 ซึ่งขนาดเล็กกว่ามาก (มี Header แบบคงที่ขนาด 4 byte) เพราะตัดส่วนที่ไม่จำเป็นทิ้งและทำงานบน UDP ซึ่งเป็น protocol ที่ไม่มีการสร้างการเชื่อมต่อระหว่างอุปกรณ์ต้นทางและปลายทาง จึงส่งข้อมูลได้เร็วมากแต่ไม่การรับรองว่าข้อมูลจะถูกส่งไปยังปลายทางอย่างแน่นอนและถูกต้องตามลำดับ การส่งซ้ำและเรียงลำดับข้อมูลต้องไปทำบนระดับชั้น Application เท่านั้น



รูปที่ 2.33 โครงสร้างข้อความของโปรโตคอล CoAP

(ที่มา : https://www.slideshare.net/basuke/coap-talkmitiot)

CoAP เป็นสถาปัตยกรรมแบบ Client/Server โดย Client จะทำการร้องขอทรัพยากรไปที่ Server โดยตรง จากนั้น Server จะทำการตอบกลับคำร้องพร้อมกับ Option 'Content-Type' เพื่อว่าบอก Client ว่ากำลังจะได้รับข้อมูลในรูปแบบไหนกลับไป (เช่น JSON, XML, CBOR เป็นต้น) โดย Client สามารถใช้คำสั่ง GET PUT POST และ DELETE เพื่อจัดการทรัพยากรบน Server ด้วย URL และ Query string คล้ายกับ REST API ที่ใช้การใน HTTP ซึ่ง CoAP ถูกนำมาใช้งานอย่างแพร่หลายสำหรับการพัฒนาระบบ IoT เนื่องจากรองรับการใช้งานอุปกรณ์ที่มีจำนวนมาก มีที่ตั้งอยู่ในพื้นที่ต่าง ๆ ที่ยากต่อสัญญาณเครือข่ายจะเข้าถึง อีกทั้งข้อจำกัดด้านพลังงานที่ออกแบบไว้ให้อุปกรณ์เหล่านั้นใช้แบตเตอรี่ได้นานเป็นแรมปี ซึ่งการใช้ Protocol HTTP over TCP/IP แบบเดิมนั้นไม่มีประสิทธิภาพพอ



รูปที่ 2.34 สถาปัตยกรรมการรับส่งข้อมูลของโปรโตคอล CoAP [9]

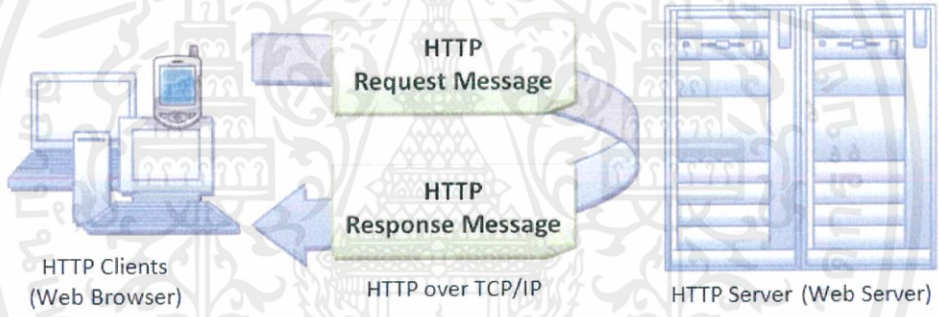
ในการสภาปัตยกรรมการรับส่งข้อมูลของ CoAP นั้นจะมีการแลกเปลี่ยนทรัพยากรกันโดยตรงอุปกรณ์ Client Sensor Node ซึ่งสามารถทำหน้าที่เป็นทั้ง Server และ Client ในเวลาเดียวกัน โดยอาจจะให้ Client Sensor Node ตัวนี้ทำหน้าที่เป็น Gateway คอยจัดการข้อมูล ก่อนที่จะส่งข้อมูลไปยัง Server ก็ได้ ในมุมของนักพัฒนาแล้ว CoAP มีความคล้ายคลึงกับ HTTP มาก ซึ่งทำให้การดึงข้อมูลจากเซ็นเซอร์ไม่ต่างจากการดึงข้อมูลผ่าน Web API ของ HTTP ซึ่งอาจจะเปรียบ CoAP ได้ว่าเป็น REST API สำหรับ MCU อีกทั้งยังเป็นโปรโตคอลที่มีความปลอดภัย เพราะมีการเข้ารหัสแบบ DTLS (เทียบเท่ากับ 3072-bit RSA key) ซึ่งสามารถทำงานบนอุปกรณ์ขนาดเล็กได้

2.9 HTTP (Hypertext Transfer Protocol)

คือโปรโตคอลที่เป็นรากฐานของเวิลด์ไวด์เว็บ (World Wide Web : WWW) และสามารถนำมาประยุกต์ใช้ได้กับ Application จำพวก Cliend/Server ใด ๆ ที่มีลักษณะของความเป็น Hypertext ชื่อของโปรโตคอลนี้อาจทำให้เกิดความเข้าใจผิดว่า HTTP เป็นโปรโตคอลที่ใช้ในการโอนย้าย Hypertext แต่ความจริงแล้วมันเป็นโปรโตคอลสำหรับส่งผ่านข้อมูล ซึ่งมีความสามารถเพียงพอที่จะก่อให้เกิดลักษณะของ Hypertext ข้อมูลที่ถูกเคลื่อนย้ายผ่านโปรโตคอลนี้อาจเป็นเพียงข้อความธรรมดา Hypertext ภาพ เสียง หรือข้อมูลอื่นใดที่ช่วยให้ผู้ใช้เข้าถึงอินเทอร์เน็ตได้ ในการใช้งานเครือข่ายคอมพิวเตอร์ทั่วไปหรือในเครือข่ายอินเตอร์เน็ตก็ตามจะมีการส่งผ่านข้อมูลไป มาระหว่างเครื่องคอมพิวเตอร์หรือข้ามเครือข่ายออกไป ระบบคอมพิวเตอร์ที่เชื่อมต่ออยู่ในแต่ละเครือข่ายอาจจะใช้ฮาร์ดแวร์และซอฟต์แวร์ที่เหมือนกันหรือแตกต่างกันได้ ดังนั้นการที่จะทำให้สามารถส่งผ่านข้อมูลถึงกันและตีความได้อย่างได้อย่างถูกต้องจะต้องมีการกำหนดข้อตกลงในการสื่อสารกันเสียก่อน เรียกว่าจะต้องกำหนดระเบียบวิธีในการติดต่อกันให้ตรงกัน เปรียบเหมือนกับการสื่อสารกันของมนุษย์เรา ถ้าเราต้องการจะติดต่อกับผู้คนต่างเชื้อชาติต่างภาษากันให้เข้าใจกันได้ถูกต้องตรงกันก็จะต้องตกลงกำหนดกันเสียก่อนว่า จะติดต่อกันอย่างไร ด้วยภาษาใดที่จะเข้าใจกันได้ เช่น ปัจจุบันมีการใช้ภาษาอังกฤษเป็นภาษากลางในการติดต่อกันมาก ทำให้เราพูดได้ว่า ภาษาอังกฤษเปรียบเสมือนเป็นภาษามาตรฐานในการสื่อสารของมนุษย์ได้ ถ้าพูดในแง่การสื่อสารข้อมูล เราก็พูดได้

ว่า ภาษาอังกฤษเป็นโปโตคอลในการสื่อสารของมนุษย์ที่มีการใช้งานอย่างแพร่หลาย เช่นเดียวกันกับโปรโตคอล TCP/IP เป็นโปรโตคอลหลักที่ใช้ในการสื่อสารข้อมูลในเครือข่ายอินเตอร์เน็ต

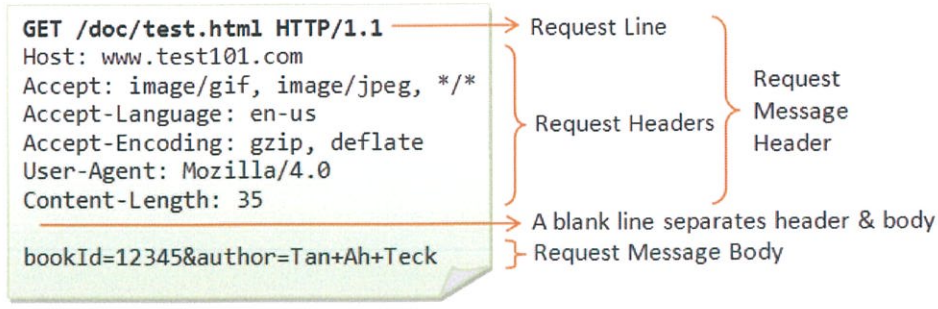
HTTP เป็นโปรโตคอลแบบ Client/Server ในลักษณะ transaction-oriented คือมีการติดต่อระหว่างโปรแกรม 2 โปรแกรม ซึ่งโดยทั่วไปได้แก่ Web browsers และ Webserver เพื่อให้มีความน่าเชื่อถือ HTTP จึงใช้ประโยชน์จากโปรโตคอล TCP แต่ถึงกระนั้น HTTP ก็เป็นโปรโตคอลที่ "ปราศจากสถานะ" กล่าวคือ การติดต่อในแต่ละครั้งเป็นอิสระต่อกัน โดยการเชื่อมต่อระหว่างClient และ Server จะถูกสร้างขึ้นมาใหม่สำหรับการติดต่อในแต่ละครั้ง และถูกตัดขาดจากกันทันทีที่การติดต่อเสร็จสิ้นสมบูรณ์ (HTTP1.0) ถึงแม้ว่าข้อกำหนดของ HTTP จะไม่ได้ระบุความสัมพันธ์ในแบบหนึ่งต่อหนึ่งระหว่างการติดต่อและช่วงเวลาของการเชื่อมต่อเช่นนี้ไว้ก็ตามที คุณสมบัติ "ปราศจากสถานะ" ดังกล่าวของโปรโตคอล HTTP นี้เหมาะสมต่อการนำมาประยุกต์ใช้เป็นอย่างยิ่ง การใช้งานเว็บบราวเซอร์นั้นโดยปกติเกี่ยวข้องกับการรับเอากลุ่มของเว็บเพจและเอกสารเข้ามา ซึ่งการดำเนินการตรงนี้เกิดขึ้นรวดเร็วมาก โดยเว็บเพจและเอกสารเหล่านี้อาจมาจากเซิร์ฟเวอร์ที่แตกต่างกันไป



รูปที่ 2.35 การเชื่อมต่อแบบ Client/Server ใน HTTP [11]

2.9.1 HTTP Request

การ Request เป็นการที่ Client ร้องขอข้อมูลจาก Server ซึ่งเราจะเรียกข้อมูลนั้นว่า HTTP Request message โดย HTTP Request message ดังกล่าวจะประกอบไปด้วย 3 ส่วนหลัก ๆ คือ Request Line Header Line และ Message body ซึ่งมีโครงสร้างดังรูปที่ 2.36



รูปที่ 2.36 รูปแบบของ HTTP Request Message [11]

โดยที่ **Request Line** จะประกอบไปด้วย

1. Request method
    - **GET** : ร้องขอเอกสาร (entity) ซึ่งถูกระบุไว้ใน Request-URI เป็น Method ที่ใช้ร้องขอโดยทั่วไป และใช้ในการ Submit Form ในรูป Query Parameter
    - **POST** : ใช้เพื่อร้องขอเอกสารจาก server ที่ถูกระบุไว้ใน Request-URI แต่ข้อมูลการร้องขอใส่ไว้ในส่วน Message Body
    - **HEAD** : คล้ายกับ GET ยกเว้นว่า server ไม่คืน Message body
    - **PUT** : ร้องขอให้เก็บ enclosed entity ไว้ในทรัพยากรที่ถูกระบุใน Request-URI
    - **DELETE** : ร้องขอให้ลบทรัพยากรที่ถูกระบุใน Request-URI
    - **TRACE** : ใช้เพื่อขอให้ทำ remote, application-layer loop-back
    - **CONNECT** : ใช้กับ proxy ซึ่งสามารถเปลี่ยนเป็น tunnel ได้อย่าง dynamic (เช่น SSL tunneling)
    - **OPTIONS** : คืนค่าของ HTTP methods ที่ server สนับสนุน ใช้สำหรับการตรวจสอบ functionality ของ web server
2. URL (Uniform Resource Locator)
3. HTTP Version ซึ่งจะมี 2 เวอร์ชั่นคือ 1.0 และเวอร์ชั่นปัจจุบัน 1.1

ส่วน **Request Headers Line** ใช้เมื่อต้องการใส่รายละเอียดในการ Request เพิ่มเติม จะเป็นส่วนที่ไม่จำเป็นต้องมีก็ได้ มีตัวอย่างดังตารางที่ 2.5 และส่วนประกอบสุดท้ายคือ **Request Message Body** จะใช้สำหรับส่งข้อมูลให้เครื่องเซิร์ฟเวอร์ ซึ่งจะมากับ POST Method

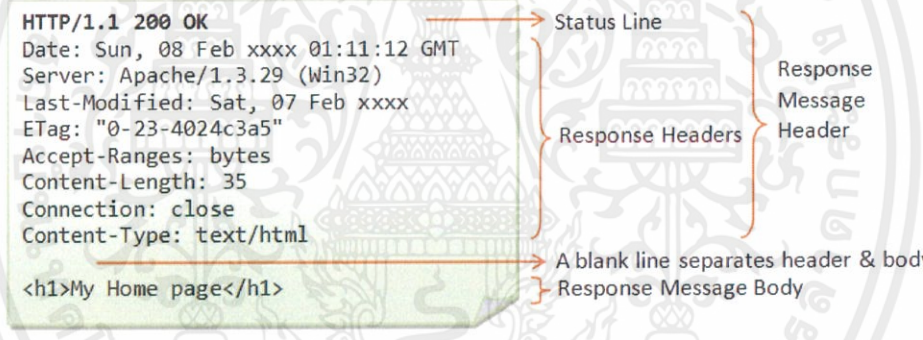
ตารางที่ 2.5 ตัวอย่าง Http Request Headers Field [11]

ชื่อ Header	คำอธิบาย	ตัวอย่าง
Content-Type	ประเภทของเนื้อหา	Content-Type: image/jpeg
Content-Length	ความยาวของเนื้อหา	Content-Length: 221

ชื่อ Header	คำอธิบาย	ตัวอย่าง
Connection	การเชื่อมต่อ	Connection: keep-alive
Cookie	การให้ cookie	Cookie: PREF=ID
Accept-Charset	Character sets ที่ยอมรับ	Accept-Charset: utf-8
Accept-Encoding	List ของ encodings ที่ยอมรับ	Accept-Encoding: gzip, deflate
Accept-Language	List of ของภาษามนุษย์สำหรับ response	Accept-Language: th

2.9.2 HTTP Response

การ Response เป็นการที่ Server ตอบรับ Request จาก Client ซึ่งเราจะเรียกข้อความที่ตอบรับนั้นว่า HTTP Response message โดย HTTP Response message ดังกล่าวจะประกอบไปด้วย 3 ส่วนหลัก ๆ คือ Status Line Response Headers และ Message body ซึ่งมีโครงสร้างดังรูปที่ 2.37



รูปที่ 2.37 รูปแบบของ HTTP Request Message [11]

โดยที่ **Status Line** จะประกอบไปด้วย

1. HTTP Version ซึ่งจะมี 2 เวอร์ชั่นคือ 1.0 และเวอร์ชั่นปัจจุบัน 1.1
2. Status Code เป็นตัวเลข 3 หลัก ที่บ่งบอกสถานะต่าง ๆ ตัวอย่างเช่น
    - **200 OK** : การร้องขอสำเร็จ วัตถุที่ร้องอยู่ใน Message
    - **301 Move Permanently** : วัตถุที่ร้องขอถูกย้ายไปยังที่ตั้งใหม่ถาวร โดยที่ตั้งใหม่อยู่ที่ Header Line (Location:)
    - **403 Forbidden** : Server ได้รับการร้องขอ แต่ปฏิเสธการให้ใช้งาน
    - **404 Not Found** : Server ไม่พบวัตถุที่ร้องขอ
    - **500 Internal Server Error** : Server ประมวลผลแล้วเกิดปัญหาภายใน ไม่สามารถดำเนินการตามการร้องขอได้

3. คำอธิบาย Status Code

ส่วน **Response Headers** คือ รายละเอียดเพิ่มเติมที่ Server ส่งกลับมาแสดงตัวอย่างดังตารางที่ 2.6 และส่วนประกอบสุดท้าย **Response Message Body** คือ ข้อมูลที่ได้จากการร้องขอ หรือเรียกว่าเนื้อหาก็ได้ ซึ่งอาจจะเป็น ไฟล์ html ไฟล์ css ไฟล์ javascript ไฟล์ json หรือไฟล์รูปภาพหรือไฟล์อื่น ๆ ก็ได้

ตารางที่ 2.6 ตัวอย่าง Http Response Headers Field [11]

ชื่อ Header	คำอธิบาย	ตัวอย่าง
Content-Type	ประเภทของเนื้อหา	Content-Type: image/jpeg
Content-Length	ความยาวของเนื้อหา	Content-Length: 221
Connection	การเชื่อมต่อ	Connection: keep-alive
Set-Cookie	การให้ cookie กับ Client	Set-Cookie: PREF=ID
Location	ที่ตั้งสำหรับ redirection หรือ resource ใหม่	http://www.google.co.th/?gws_rd=cr&ei=mrXpU4mIJMff8AWp7ID4BQ
Refresh	ใช้ในการ redirection หรือ resource ที่สร้างใหม่ ภายหลังเวลาที่กำหนด	Refresh: 5; url=http://www.xyz.com/:
Transfer-Encoding	Encoding ที่ใช้ในการส่ง เช่น chunked, compress, deflate, gzip	Transfer-Encoding: chunked (ในกรณียังไม่รู้ขนาดข้อมูลแน่นอน)

2.10 ภาษาซีสำหรับ Microcontroller Arduino

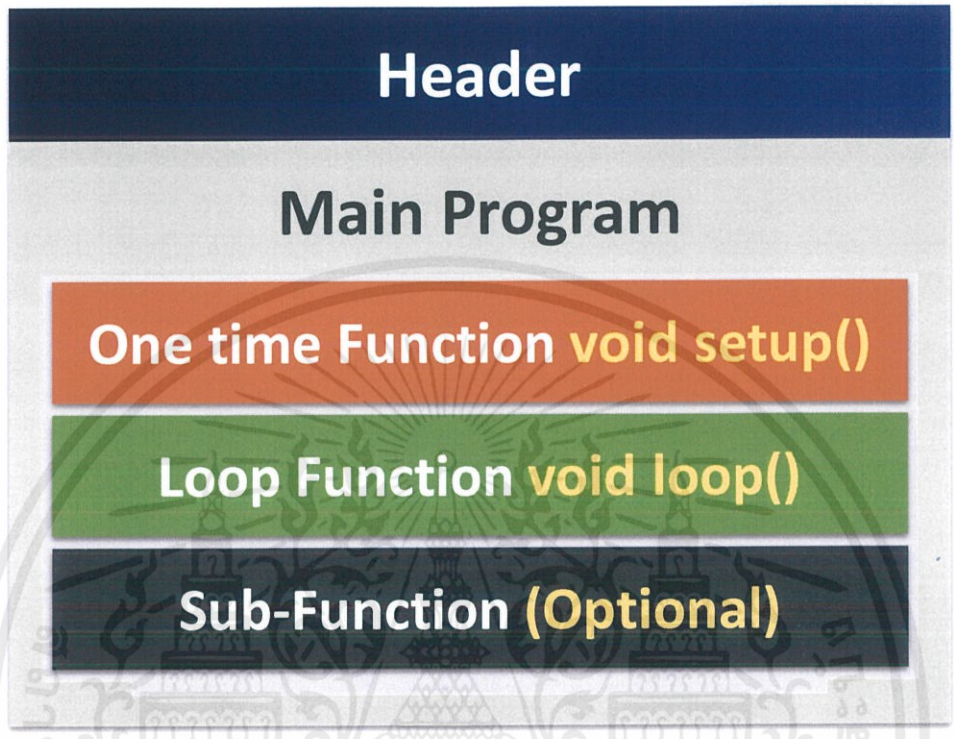
Microcontroller ไม่ว่าจะเป็นตระกูลใดก็ตามจะทำงานได้ก็ต่อเมื่อมีชุดคำสั่งที่สั่งให้ทำงานตามที่ต้องการที่เรียกว่าโปรแกรม โดยคำสั่งหรือโปรแกรมที่ Microcontroller เข้าใจและสามารถทำงานได้อยู่ในรูปของรหัสลอจิก 0 และ 1 หากนำลอจิกมาจับกลุ่มก็เป็นเลขฐาน 16 ที่เรียกว่าภาษาเครื่องซึ่งภาษาเครื่องเป็นภาษาที่มนุษย์ไม่สามารถเข้าใจได้เนื่องจากเป็นเลขฐาน 16 ทั้งหมด ดังนั้นในการเขียนโปรแกรมจึงจำเป็นต้องใช้ภาษาที่มนุษย์สามารถเข้าใจได้ โดยภาษาที่มนุษย์เข้าใจได้และใกล้เคียงกับภาษาเครื่องมากที่สุดคือภาษา Assembly แต่เนื่องจากการพัฒนางานโดยใช้ภาษา Assembly เป็นไปได้ยากและซับซ้อน เพื่อให้ง่ายและรวดเร็วต่อการพัฒนาโปรแกรมใช้งาน Microcontroller ภาษาที่เหมาะสมคือภาษาซี

โครงสร้างภาษาซีสำหรับ Arduino ถูกจัดใหม่ให้ง่ายต่อผู้ใช้งานเบื้องต้นซึ่งผู้ออกแบบได้จัดวางให้ผู้ใช้งานได้ใช้งานง่ายซึ่งโครงสร้างหลัก ๆ จะมีเพียง 3 ส่วนเท่านั้นคือ

1. **setup** เป็นส่วนที่เก็บฟังก์ชั่นที่ทา งานครั้งเดียว
2. **loop** เป็นส่วนที่เก็บฟังก์ชั่นที่เมื่อทา งานครบแล้วจะวนกลับมาทา ซ้า ใหม่ตั้งแต่ต้นแต่ถ้าต้องการเขียนโปรแกรมขั้นสูงสามารถเขียนในส่วนหัวโปรแกรมและส่วนของฟังก์ชั่นรองที่เขียนขึ้นใช้งานเองเพื่อให้ใช้งานสะดวกมากยิ่งขึ้นได้เช่นเดียวกับภาษาซีมาตรฐาน

3. function เป็นส่วนที่เก็บฟังก์ชั่นเพิ่มเติมที่สามารถนำมาเรียกใช้เพิ่มเติมได้ในโปรแกรมในส่วนอื่น โดนไม่ต้องประกาศ Header เพิ่มเติม

ซึ่งสามารถแสดงรูปแบบโครงสร้างได้ดังรูปที่ 2.38

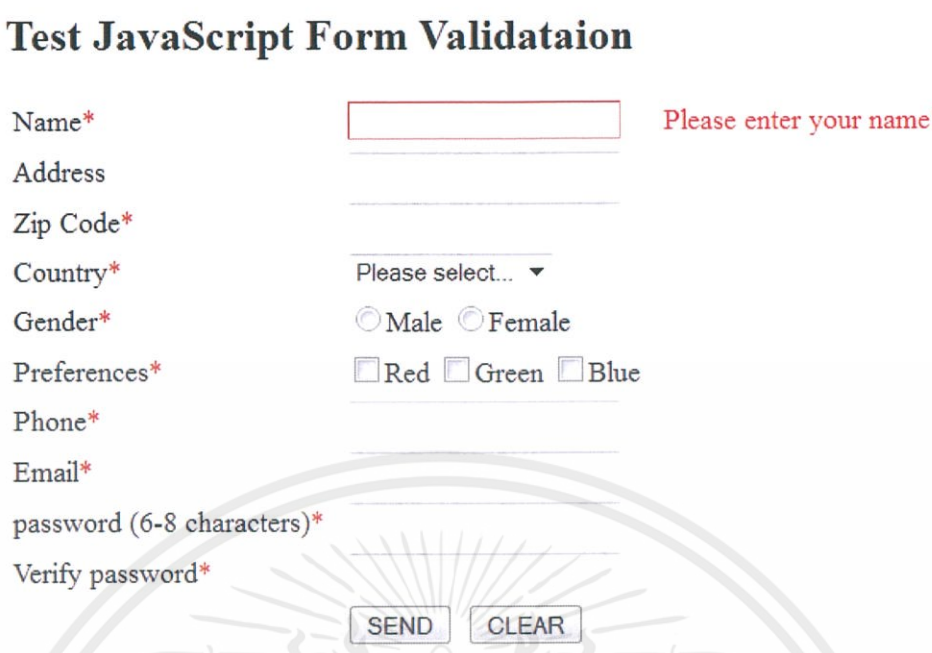


รูปที่ 2.38 โครงสร้างภาษาซีสำหรับ Arduino

2.11 Javascript

Javascript คือ ภาษาคอมพิวเตอร์ สำหรับการเขียนโปรแกรมบนระบบอินเทอร์เน็ต ที่เกิดมาเพื่อเป็นส่วนเสริมสำคัญของภาษา HTML (Hypertext Markup Language) เปรียบเทียบอย่างง่าย เมื่อ HTML คือตัวโครงสร้างของเว็บไซต์ CSS (Cascading Style Sheets) ช่วยตกแต่งโครงสร้างให้สวยงาม เจ้า Javascript ก็จะเสริมให้เว็บไซต์สามารถโต้ตอบกับผู้ใช้งานได้มากขึ้น ไม่ได้เป็นเพียงเว็บไซต์ที่มีโครงสร้างธรรมดา ๆ เท่านั้น

Javascript มีการนำไปใช้กันหลากหลาย โดยที่พบเห็นบ่อย ๆ จะเป็นการนำไปประยุกต์ใช้กับการทำแบบฟอร์มในการกรอกข้อมูลต่าง ๆ ดังตัวอย่างในรูปที่ 2.39 Javascript สามารถช่วยตรวจสอบข้อมูลที่ผู้ใช้งานกรอกในเบื้องต้น แล้วแจ้งเตือนทันที อาทิ มีการลืมกรอกข้อมูลในบางช่อง มีการกรอกข้อมูลผิดรูปแบบ หรือช่วย HTML ในการซ่อนเนื้อหา และให้แสดงขึ้นมาในเวลาที่ต้องการ และความสามารถอื่น ๆ อีกหลากหลายสุดแล้วแต่นักพัฒนาจะรังสรรค์เลือกใช้เพื่อให้เว็บไซต์ของตนมีความโดดเด่น เนื่องจากเขียนง่าย และรองรับการใช้งานแทบทุก Web browsers ในปัจจุบัน ทำให้ Javascript เป็นภาษาแห่งอนาคต นิยมใช้ และน่าเรียนรู้ไว้เป็นที่สุด



รูปที่ 2.39 ตัวอย่างการนำ Javascript ไปใช้ในการ Validate Form

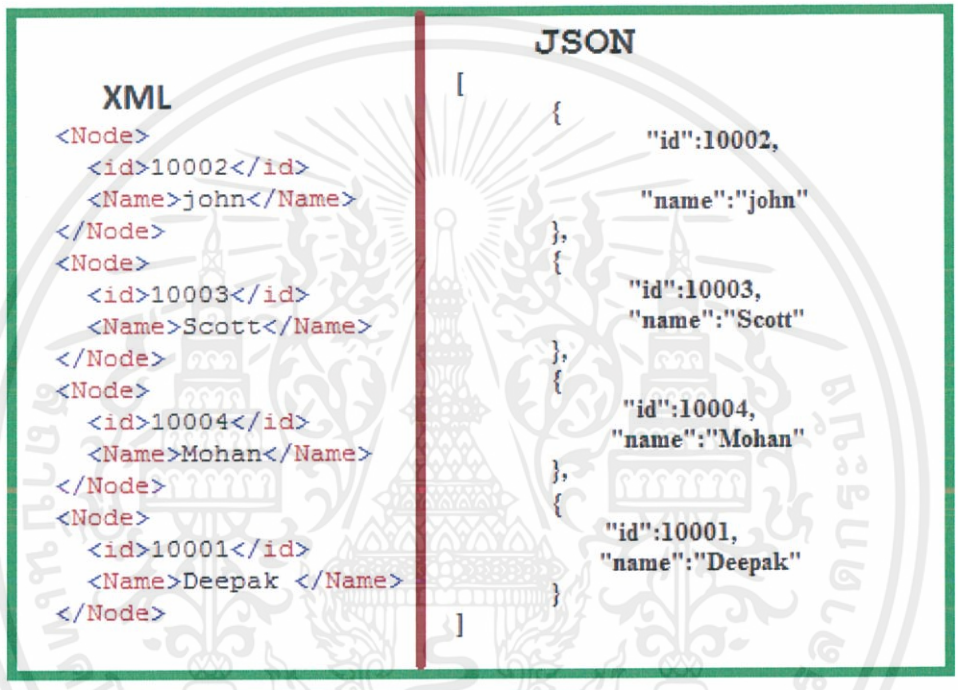
(ที่มา : http://www.ntu.edu.sg)

นอกจาก Javascript จะมีความสามารถในการตอบโต้กับผู้ใช้แล้ว ยังมีความสามารถอื่น ๆ ดังนี้

- ช่วยลดภาระการทำงานทางฝั่งผู้ให้บริการหรือ Server เนื่องจาก Web browsers ของทางฝั่งผู้ใช้บริการหรือ Client จะสามารถประมวลผลได้เอง
- มีกลไกในการตรวจสอบ การเปรียบเทียบ การตัดสินใจ การประมวลผล และสามารถสร้างฟังก์ชันได้เอง
- สามารถใช้งานร่วมกับเทคโนโลยีอื่น ๆ ได้ เช่น ActiveX CGL Java และ Plug-In โดยไม่ขึ้นกับ Platform ใด ๆ
- สามารถเปลี่ยนรูปแบบ Webpage ของเอกสาร HTML หรือ XHTML จาก Static มาเป็นแบบ Dynamic ที่สามารถโต้ตอบกับ Client ได้
- ใช้งานง่าย เพราะมีลักษระเป็น Interpreter แบบ Text File ฝังอยู่ในเอกสาร HTML หรือ XHTML จึงสามารถทำงานบน Web Browsers ได้ทันที โดยไม่ต้องคอมไพล์ (Compile) โดยเขียนโปรแกรมเหมือนกับภาษา Java
- ใช้รูปแบบคำสั่งเหมือนกับภาษา Java เช่น คำสั่งเพื่อดำเนินการทางคณิตศาสตร์ ตรรกศาสตร์ สตริง รวมทั้ง คำสั่งควบคุมลำดับการดำเนินงาน ได้แก่ if while และ for เป็นต้น
- เรียนรู้ง่าย เหมาะสำหรับนำไปใช้พัฒนาโปรแกรมบนระบบอินเตอร์เน็ต

2.12 JSON (JavaScript Object Notation)

JSON เป็นรูปแบบการแลกเปลี่ยนรับส่งข้อมูลที่ได้รับความนิยมสูง เนื่องจากมีขนาดเล็กและกระชับกว่า XML ทำให้เปลือง Bandwidth ในการรับส่งข้อมูลน้อยกว่า ดังนั้นจึงมีการนำมาประยุกต์ใช้งานกันอย่างแพร่หลายในการใช้เป็นรูปแบบการส่งข้อมูลในระบบ IoT ซึ่งเป็นระบบที่ต้องการให้มี Bandwidth ในการรับส่งข้อมูลให้น้อยที่สุด เพราะจะเป็นการประหยัดพลังงาน และช่วยทำให้สามารถรองรับอุปกรณ์ได้จำนวนมาก ๆ ในหนึ่งเซลล์ไซต์ นอกจากนี้ JSON ยังใช้งานง่าย ผู้ใช้งานที่เริ่มต้นใหม่สามารถทำความเข้าใจได้อย่างรวดเร็ว

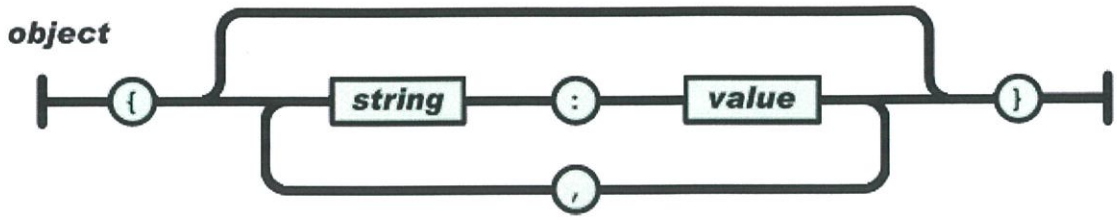


รูปที่ 2.40 การเปรียบเทียบข้อมูลชุดเดียวกันที่เก็บในรูปแบบ XML และ JSON

(ที่มา : http://zoro.creostories.co/what-is-json-format)

2.12.1 JSON Object

JSON Object มีลักษณะคล้ายกับข้อมูล 1 Reccord สามารถมี Field เดียวหรือหลาย Field ก็ได้ ตัวอย่าง JSON Object แบบ 1 Field เดียว ซึ่งมีโครงสร้างของ Object ดังรูปที่ 2.41



รูปที่ 2.41 โครงสร้างของ JSON Object

(ที่มา : https://www.json.org)

จากรูปที่ 2.41 เมื่อนำมาเขียนตัวอย่างข้อมูลในรูป JSON Format แล้วจะได้ว่า

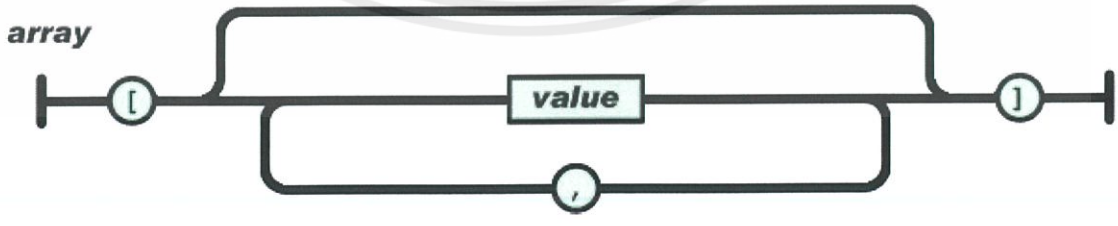
```
{
        “Temperature” : 27.5
}
```

หรือในกรณีที่ข้อมูลมีหลาย Field สามารถเขียนได้ว่า

```
{
        “Temperature” : 27.5,
        “Humidity” : 67
}
```

2.12.1 JSON Array

Jason Array จะมีข้อมูลหลาย Record ตัวอย่างโครงสร้างแบบนี้ คล้ายกับ Array 2 มิติ ซึ่งตัวอย่างในรูปที่ 2.42 เป็นการจัดเก็บข้อมูลในรูปแบบ JSON Array แบบ 3 Record โดย JSON Array จะมีโครงสร้างดังรูปที่ 2.42



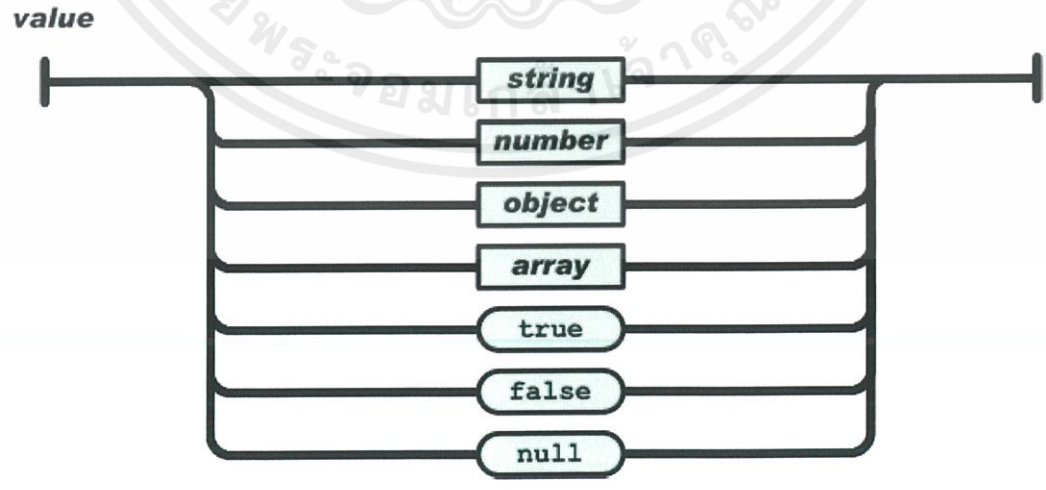
รูปที่ 2.42 โครงสร้างของ JSON Array

(ที่มา : https://www.json.org)

จากรูปที่ 2.42 เมื่อนำมาเขียนตัวอย่างข้อมูลในรูป JSON Format แล้วจะได้ว่า

```
{
    {
            “Site_ID” : 1,
            “Temperature” : 25,
             “Humidity” : 65
    },
    {
            “Site_ID” : 2,
            “Temperature” : 27.5,
             “Humidity” : 67
    },
    {
            “Site_ID” : 3,
            “Temperature” : 23.2,
             “Humidity” : 43
    }
}
```

จากรูปแบบที่ได้ยกตัวอย่างมานั้น เป็นเพียงส่วนหนึ่งของการจัดรูปแบบข้อมูลของ JSON Data Format ซึ่งนอกจาก Objects และ Array แล้ว ยังมีโครงสร้างอื่น ๆ อีกดังรูปที่ 2.43 โดยผู้ใช้งานสามารถเลือกรูปแบบการจัดเก็บข้อมูลที่เหมาะสม มาประยุกต์ใช้กับการทำงานได้



รูปที่ 2.43 โครงสร้างประเภทต่าง ๆ ของ JSON Data Format

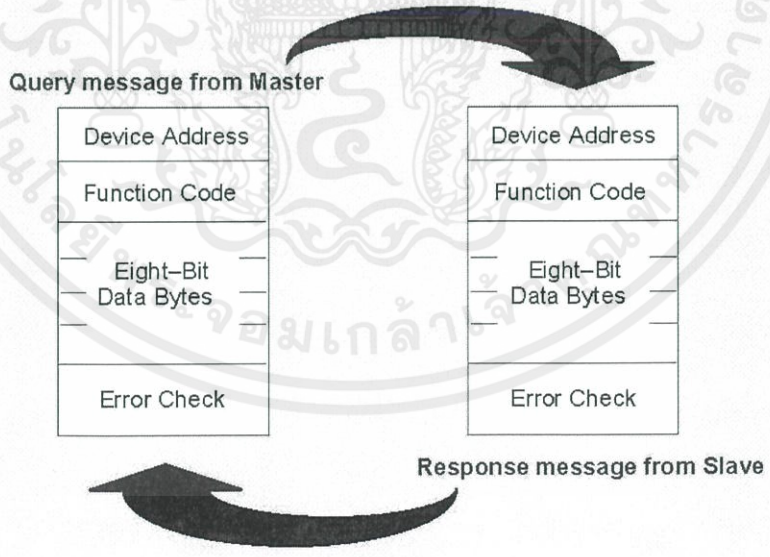
(ที่มา : https://www.json.org)

2.13 โปรโตคอล Modbus

โปรโตคอล MODBUS เป็นโปรโตคอลเพื่อสื่อสารข้อมูล Input/Output และ Register ภายใน PLC ซึ่งถูกคิดค้นโดย Modicon (ปัจจุบันคือบริษัท Schneider Electric) โปรโตคอล MODBUS ได้เป็นที่ยอมรับกันอย่างกว้างขวางในการติดต่อสื่อสารที่เป็นแบบ Network Protocol อันเนื่องมาจาก MODBUS เป็นระบบเปิด ไม่มีค่าใช้จ่าย เชื่อมต่อและพัฒนาง่าย พร้อมทั้งยังสามารถนำโปรโตรคอลนี้ไปใช้งานในอุปกรณ์อื่นๆ เช่น Digital Power Meter, RTU (Remote Terminal Unit), Remote I/O, PLC เป็นต้น นอกจากนี้ MODBUS ยังสามารถรองรับและใช้งานร่วมกับ Application จำพวก SCADA และ HMI Software ได้อีกด้วย

โปรโตคอล MODBUS เป็นการสื่อสารข้อมูลในลักษณะ Master/Slave ดังรูปที่ 2.44 ซึ่งเป็นการสื่อสารจากอุปกรณ์แม่ (Master) เครื่องเดียว ส่วนใหญ่มักเป็นซอฟต์แวร์คอมพิวเตอร์หรืออุปกรณ์แสดงผล HMI ไปยังอุปกรณ์ลูก (Slave) ได้หลาย ๆ เครื่อง โดยสามารถกำหนดหมายเลขอุปกรณ์ได้สูงสุด 255 เครื่อง โดยมีลักษณะการส่งข้อมูล 2 แบบ คือ ข้อมูลแบบแอสกี (ASCII) และข้อมูลแบบเลขฐานสอง (Binary) ในโปรโตคอล MODBUS ที่สื่อสารข้อมูลแบบ ASCII จะเรียก MODBUS ASCII และโปรโตคอล MODBUS ที่สื่อสารข้อมูลแบบเลขฐานสอง จะเรียก MODBUS RTU ทำให้มีความแตกต่างในการกำหนดค่าพอร์ตสื่อสาร

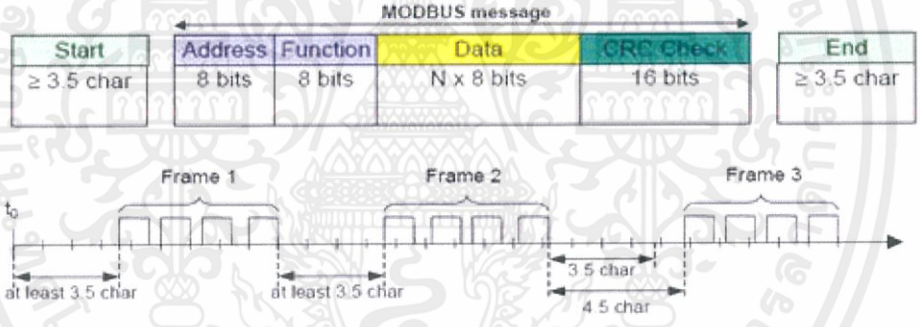
การรับส่งข้อมูลด้วยโปรโตคอล MODBUS สามารถเลือกได้ 2 โหมด คือ โหมด ASCII และโหมด RTU ซึ่งทั้ง 2 โหมดนี้มีความแตกต่างกันที่การกำหนดรูปแบบของชุดข้อมูลภายในเฟรม จะเลือกโหมดใดก็ได้แต่มีเงื่อนไขว่า อุปกรณ์ทุกตัวที่ต่อร่วมกันอยู่ในบัสหรือเครือข่ายเดียวกัน จะต้องตั้งให้เลือกใช้โหมดเดียวกันทั้งหมด



รูปที่ 2.44 การติดต่อสื่อสารของ MODBUS แบบ Master/Slave [13]

2.13.1 MODBUS RTU

เฟรมข้อมูลในโหมด RTU 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 หากไม่ตรงกันแสดงว่าเกิดความผิดพลาดในการรับส่งข้อมูลในโหมด RTU 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 นั้น สามารถเลือกเป็นแบบคู่ (Even Parity) หรือคี่ (Odd Parity) ก็ได้ และหากต้องการออกแบบให้สอดคล้องกับอุปกรณ์ที่มีใช้กันทั่วไปมากที่สุด ควรเลือกแบบคู่โดยที่สามารถปรับเปลี่ยนเป็นแบบคี่หรือไม่มีการตรวจสอบ Parity (No Parity) ได้ด้วย



รูปที่ 2.45 ลักษณะเฟรมข้อมูลของ MODBUS RTU [13]

With Parity Checking

Start	1	2	3	4	5	6	7	8	Par	Stop

Without Parity Checking

Start	1	2	3	4	5	6	7	8	Stop	Stop

รูปที่ 2.46 ลักษณะข้อมูลแต่ละไบต์ของ MODBUS RTU [13]

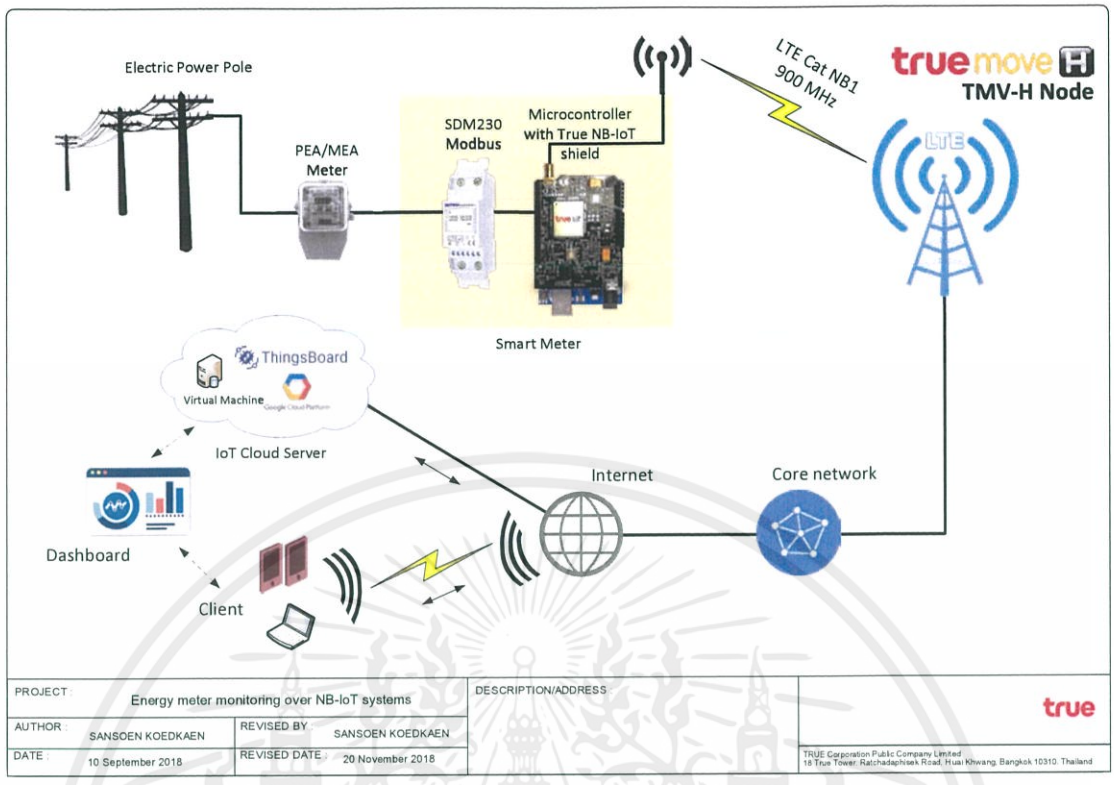
บทที่ 3
การออกแบบและการจัดทำโครงงาน

ในขั้นตอนการดำเนินงานจัดทำโครงงานนี้ จะแบ่งออกเป็น 3 ส่วนหลัก ๆ โดยในขั้นแรกผู้จัดทำจะทำการออกแบบการสร้างระบบมาตรวัดสังเกตการณ์พลังงานไฟฟ้าทั้งในด้านซอฟแวร์และฮาร์ดแวร์รวมถึงการติดตั้งระบบ IoT Cloud Server ส่วนที่สองก็จะเป็นการนำอุปกรณ์ต่าง ๆ ที่ได้ทำการออกแบบไว้ มาติดตั้งเข้าด้วยกัน และส่วนสุดท้าย จะเป็นการนำอุปกรณ์ที่ได้ทำติดตั้งเข้าด้วยกันเรียบร้อยแล้วมาทำการทดสอบการทำงานและเก็บผลการทดสอบเพื่อนำมาวิเคราะห์ค่าความคลาดเคลื่อนว่าสามารถยอมรับได้หรือไม่

3.1 การออกแบบระบบโดยรวม

โครงงานมาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบ เป็นการสร้างระบบสังเกตการณ์การใช้พลังงานไฟฟ้าของไซต์งาน โดยการนำมาตรวัดพลังงานไฟฟ้ายี่ห้อ Eastron รุ่น SDM230 มาประยุกต์ใช้เข้ากับ Microcontroller โดยตัวมาตรวัดจะสื่อสารกับ Microcontroller ด้วยโพรโตคอล RS485 Modbus ซึ่งเป็นโพรโตคอลที่ใช้ในการสื่อสารทางสายแบบอนุกรม เพื่อส่งพารามิเตอร์ที่วัดได้ให้กับ Microcontroller ในการประมวลผลต่อไป

เมื่อ Microcontroller ได้รับข้อมูลเข้ามาแล้ว ก็จะทำการประมวลผลข้อมูลแล้วส่งต่อให้ NB-IoT Shield เพื่อทำการส่งไปยัง IoT Cloud ผ่านทางโพรโตคอล CoAP หรือ UDP ผ่านทางเครือข่ายโทรศัพท์มือถือที่ถูกจัดไว้สำหรับอุปกรณ์ NB-IoT แล้วนำข้อมูลที่ได้ไปเก็บไว้บนฐานข้อมูลของ IoT Cloud Server จากนั้นผู้ใช้สามารถเรียกดูข้อมูลการใช้งานพลังงานไฟฟ้าผ่านทาง Dashboard ด้วยการเชื่อมต่อ Internet และเรียกใช้งานจาก Web browser หรือ Application ของ IoT Cloud ที่มีการจัดเตรียมไว้

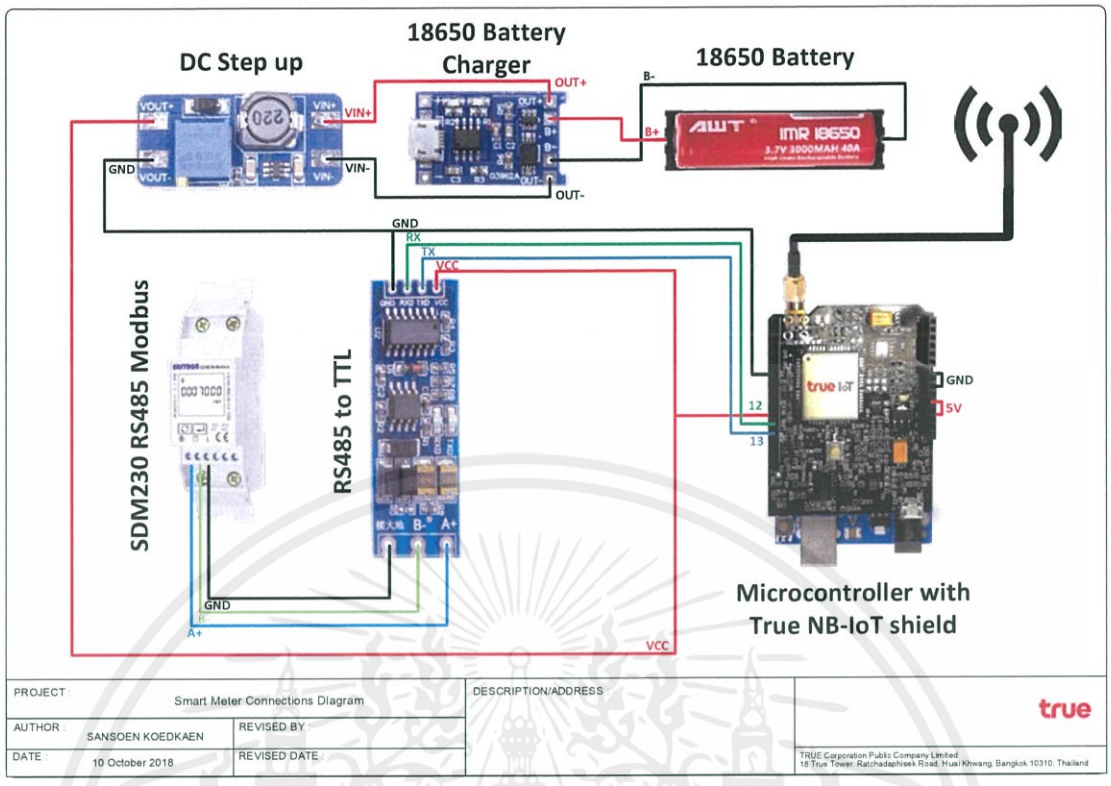


รูปที่ 3.1 ภาพรวมการทำงานของระบบสังเกตการณ์พลังงานไฟฟ้า

3.2 การออกแบบระบบมาตรวัดพลังงานไฟฟ้า

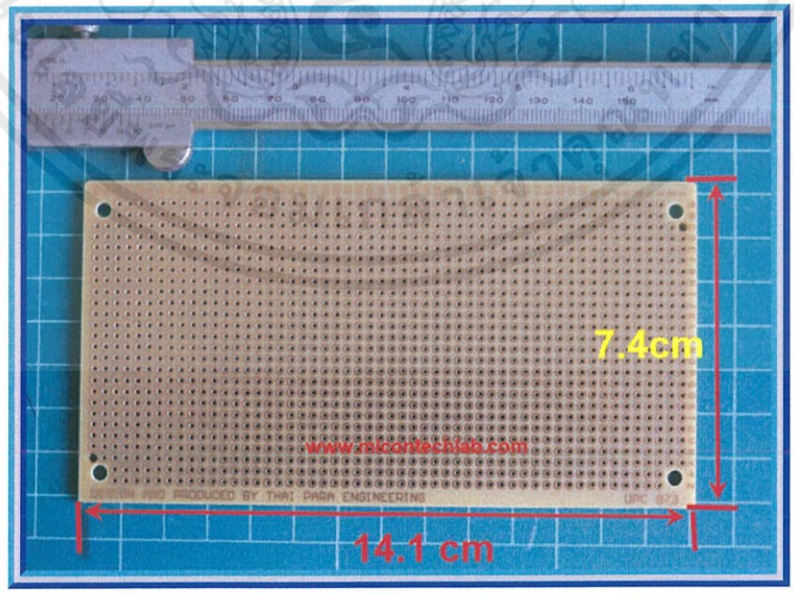
ในส่วนของมาตรวัดพลังงานไฟฟ้า (ส่วนของ Smart Meter ในรูปที่ 3.1) จะเป็นส่วนของระบบที่ทำหน้าที่เป็นอุปกรณ์วัดค่าพารามิเตอร์ทางไฟฟ้าต่าง ๆ และทำการส่งข้อมูลขึ้นไปยัง Cloud Server โดยอุปกรณ์ที่จะนำมาเชื่อมต่อกันนั้นได้แก่

- Microcontroller Arduino UNO R3
- True NB-IoT Shield
- Eastron SDM230 Modbus
- RS485 to TTL Module
- 18650 Battery with holder
- 18650 Battery Charger
- DC Step up



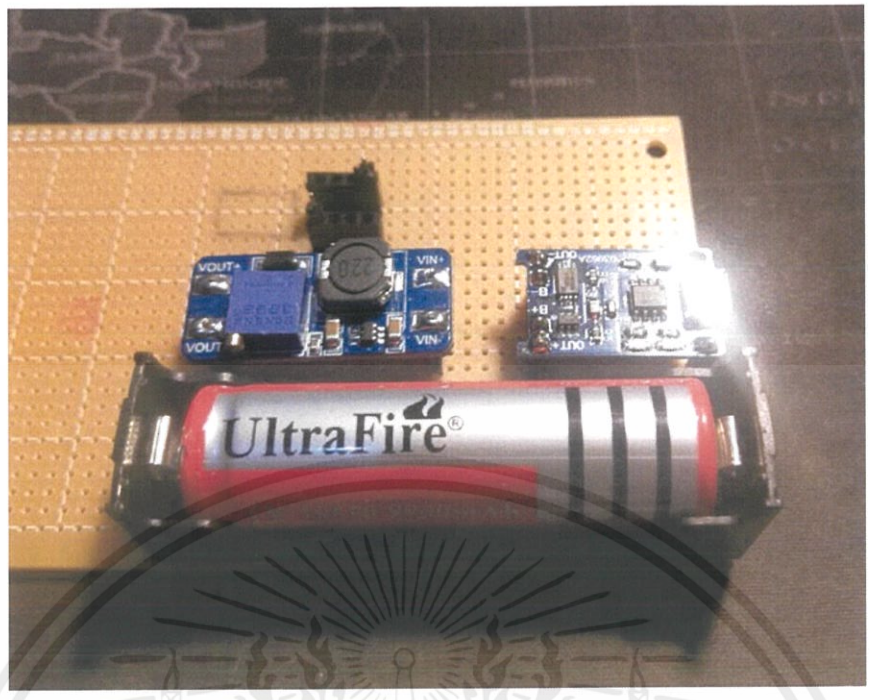
รูปที่ 3.2 การเชื่อมต่ออุปกรณ์ต่าง ๆ ของมาตรวัดสังเกตการณ์พลังงานไฟฟ้า

โดยการนำอุปกรณ์ต่าง ๆได้แก่ 18650 Battery holder, 18650 Battery charger, DC Step up และ RS485 to TTL Module มาทำการบัดกรีลงบนแผ่นวงจรเอนกประสงค์ดังรูปที่ 3.3 และ 3.4



รูปที่ 3.3 แผ่นวงจรเอนกประสงค์

(ที่มา : http://www.micontechlab.com/product/1238)



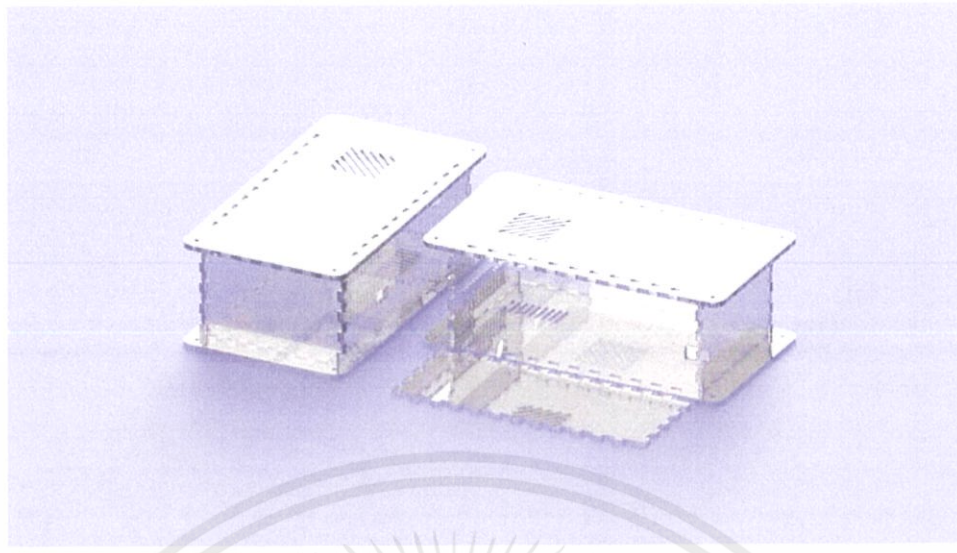
รูปที่ 3.4 การบัดกรีอุปกณ์ต่าง ๆ ลงบนแผ่นวงจรเอนกประสงค์

จากรูปที่ 3.2 โมดูล RS485 to TTL จะมีหน้าที่แปลงรูปแบบสัญญาณ RS485 Modbus ให้เป็นสัญญาณ UART TTL เพื่อทำการเชื่อมต่อเข้ากับช่องทางการสื่อสารแบบอนุกรมของ Microcontroller โดยในที่นี้จะทำการใช้ Software Serial มาเพิ่มช่องทางการสื่อสารอนุกรมใน Pin 12 และ 13 เนื่องจาก Arduino UNO R3 นั้นมีช่องทางการสื่อสารอนุกรมเพียงช่องทางเดียว

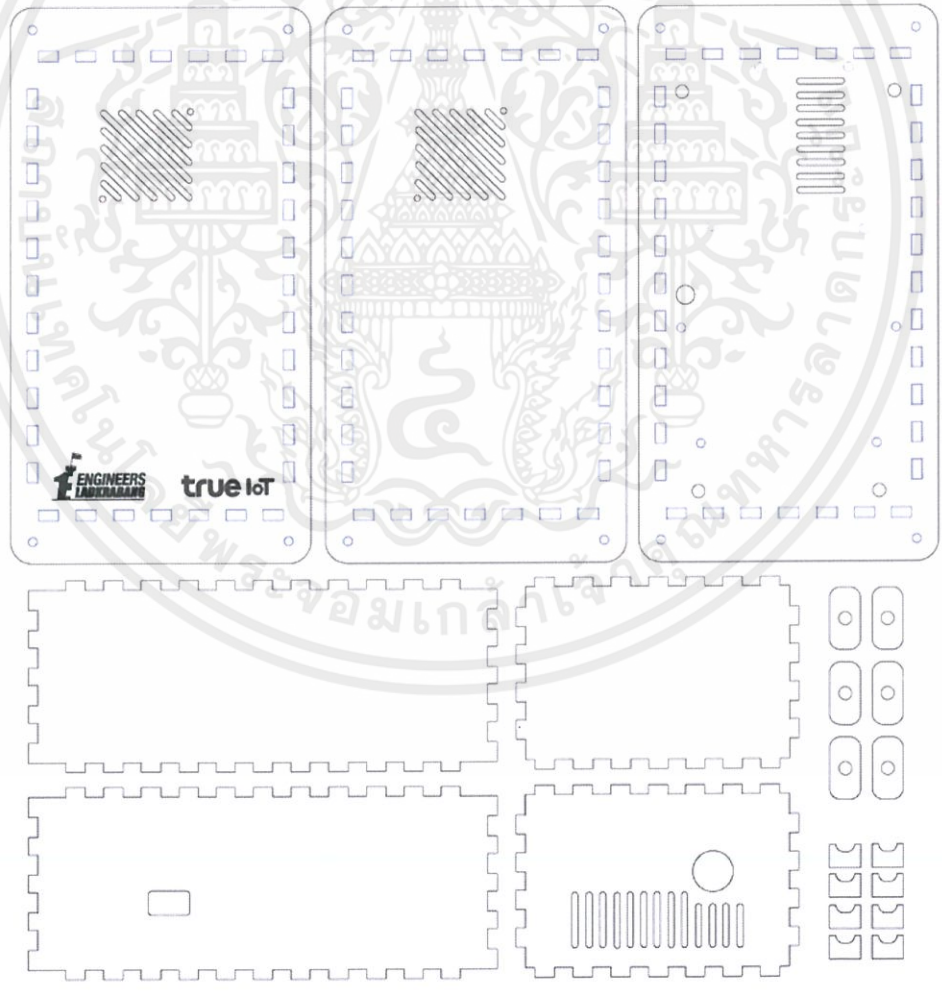
เนื่องจากผู้จัดทำมีความต้องการให้ระบบสามารถทำงานได้ตลอดเวลา จึงได้ออกแบบให้ระบบแหล่งจ่ายไฟเป็นแบตเตอรีชนิด Li-ion 18650 เพื่อป้องกันการเกิดปัญหากระแสไฟฟ้าดับ โดยจะมีโมดูล 18650 Battery Charger คอยชาร์จไฟให้ในขณะที่ระบบไฟฟ้าทำงานปกติ จากนั้นจึงใช้ DC Step up มาทำการ Boost ความต่างศักย์ที่แบตเตอรีจ่ายมาให้เป็น 5V เพื่อนำไปจ่ายเลี้ยงให้กับอุปกรณ์ต่างๆ ได้แก่ RS485 to TTL Module และ Microcontroller

3.3 การออกแบบกล่องบรรจุและการติดตั้ง

ผู้จัดทำได้ออกแบบกล่องบรรจุโดยเลือกใช้วัสดุเป็นแผ่นอะคริลิคใส มาตัดด้วยเครื่องตัดแบบเลเซอร์ โดยมีการออกแบบดังรูปที่ 3.5 และ 3.6



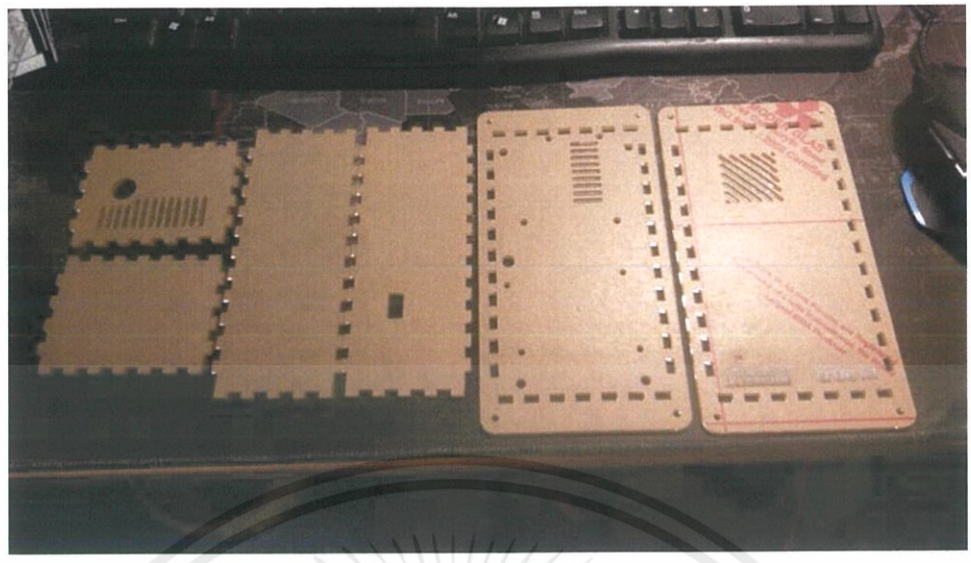
รูปที่ 3.5 กล่องบรรจุที่ได้ทำการออกแบบ



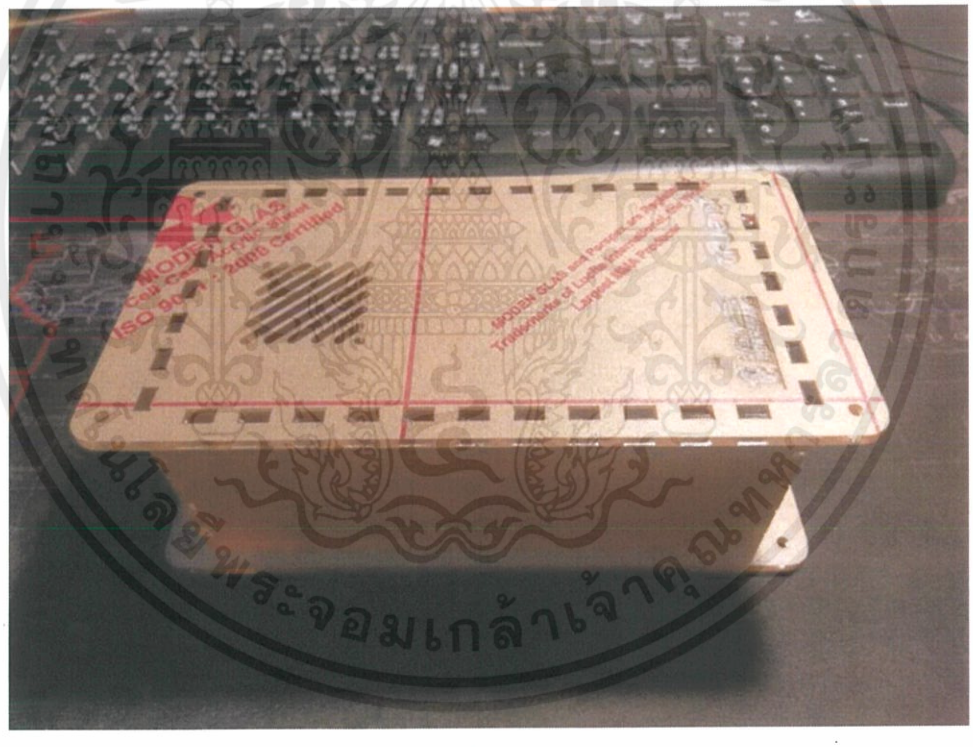
รูปที่ 3.6 ไฟล์สำหรับการตัดแผ่นอะคริลิคด้วยเครื่องตัดเลเซอร์

หลังจากออกแบบเรียบแล้วจึงใช้เครื่องตัดเลเซอร์ตัดแผ่นอะคริลิคตามที่ออกแบบไว้ดังรูปที่ 3.7 เมื่อทำการตัดแผ่นอะคลิริคครบเรียบร้อยแล้วดังรูปที่ 3.8 จึงนำมาประกอบกอบเข้ากันเป็นกล่องบรรจุภัณฑ์และใช้สกรูและเสารองแผ่นวงจรดังรูปที่ 3.9

รูปที่ 3.7 การตัดแผ่นอะคริลิคด้วยเครื่องตัดเลเซอร์

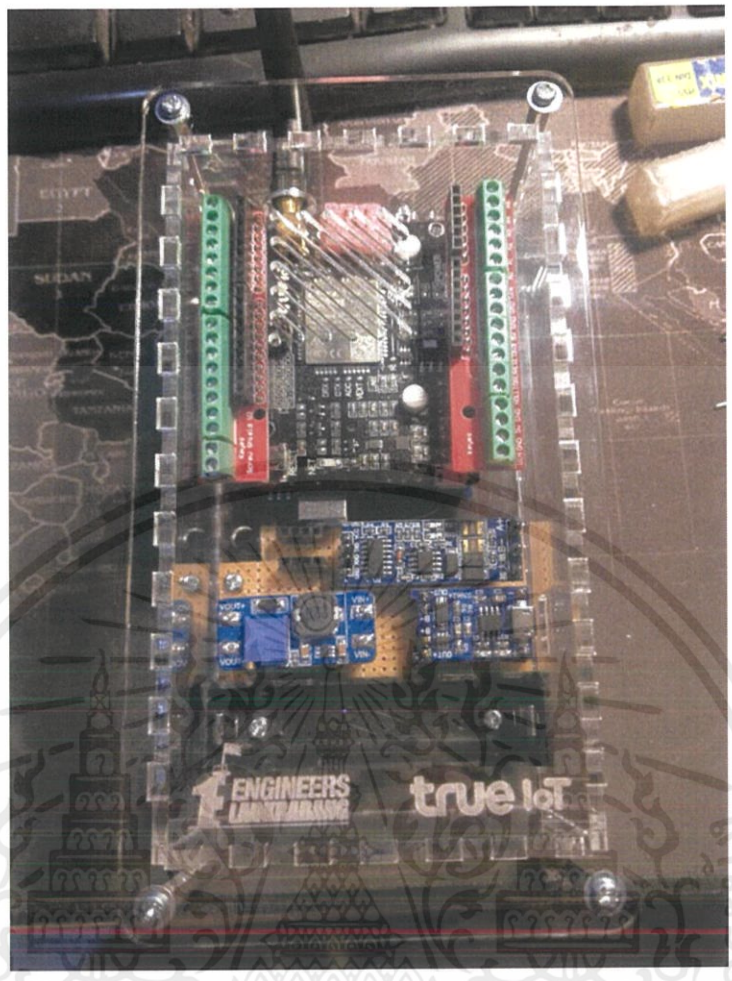


รูปที่ 3.8 แผ่นอะคริลิคที่ตัดครบเรียบร้อยแล้ว



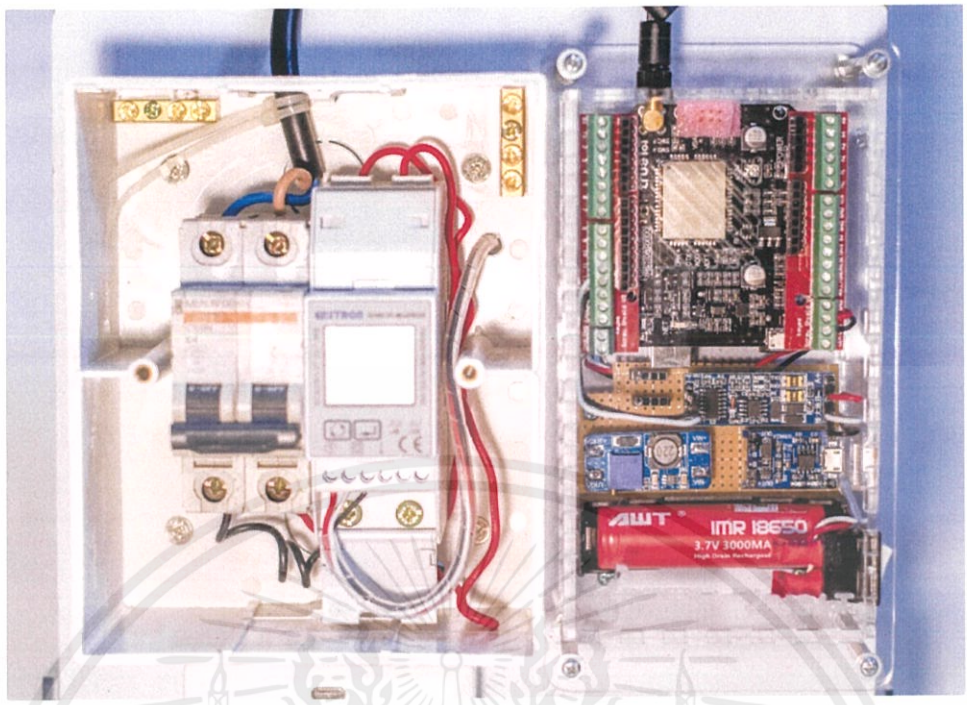
รูปที่ 3.9 แผ่นอะคริลิคที่ถูกประกอบเป็นกล่องบรรจุ

เมื่อทำการนำอุปกรณ์มาตรวัดพลังงานไฟฟ้ามาบัดกรีลงบนแผงวงจรเอนกประสงค์ดังรูปที่ 3.4 และทำการสร้างกล่องบรรจุดังรูปที่ 3.9 เรียบร้อยแล้ว จึงนำอุปกรณ์มาตรวัดพลังงานไฟฟ้าและ Microcontroller มาติดตั้งลงในกล่องบรรจุภัณฑ์ดังรูปที่ 3.10

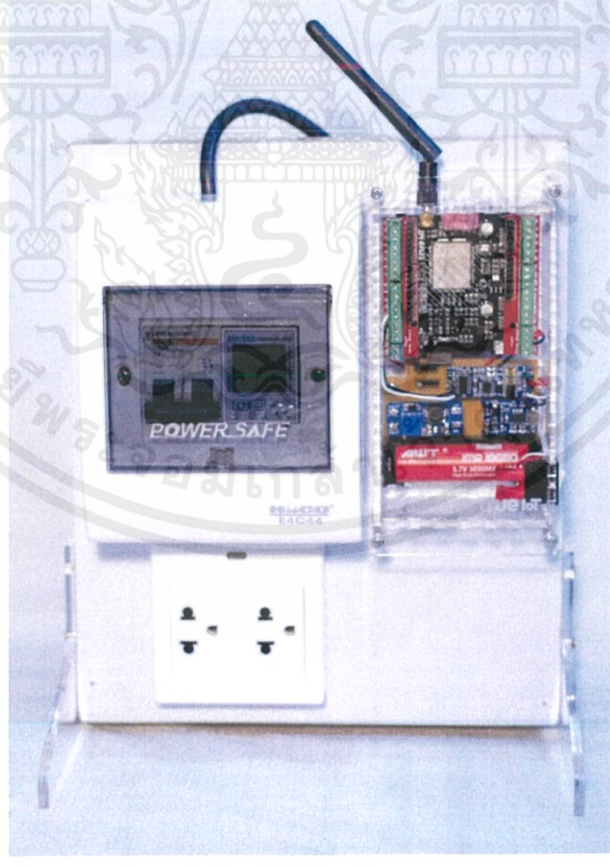


รูปที่ 3.10 การนำอุปกรณ์มาตรวัดพลังงานไฟฟ้าและ Microcontroller ติดตั้งลงกล่องบรรจุ

เมื่อทำการประกอบอุปกรณ์มาตรวัดพลังงานไฟฟ้าเรียบร้อยแล้ว จึงนำมาเชื่อมต่อเข้ากับมาตรวัดพลังงานไฟฟ้า Eastron SDM230 บนแผงวงจรสาธิต โดยทำการเดินสายเชื่อมต่อสำหรับการสื่อสารแบบอนุกรม RS485 และยึดติดอุปกรณ์มาตรวัดพลังงานไฟฟ้าให้แน่นหนา ดังรูปที่ 3.11 และ 3.12



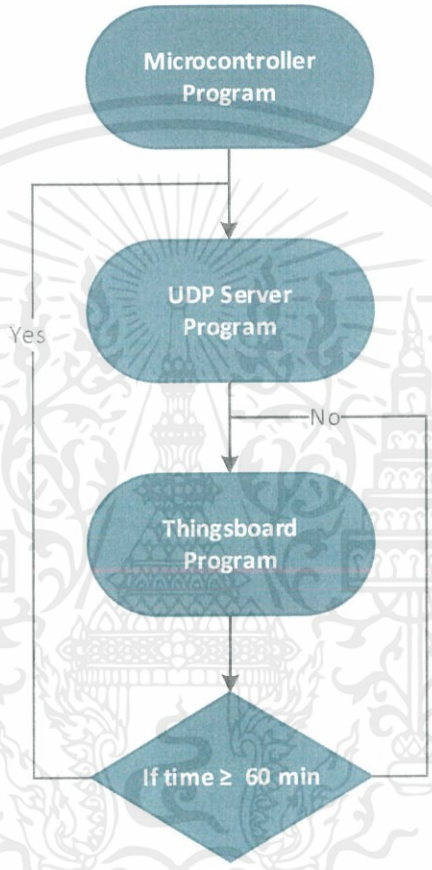
รูปที่ 3.11 การติดตั้งอุปกรณ์มาตรวัดพลังงานไฟฟ้า



รูปที่ 3.12 การติดตั้งมาตรวัดพลังงานไฟฟ้าเข้ากับแผงวงจรสาธิต

3.4 การออกแบบโปรแกรมของระบบ

โครงสร้างโปรแกรมของระบบมาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบนั้นจะถูกแบ่งออกเป็น 3 ส่วนหลัก ๆ เนื่องจากระบบทั้งหมดนั้นทำงานร่วมกันระหว่าง Microcontroller, IoT Cloud Server และบริการจากผู้ให้บริการภายนอก โดยแสดงได้ดังแผนผังการทำงานของระบบในรูปที่ 3.1



รูปที่ 3.13 แผนผังการทำงานร่วมกันของโปรแกรมในส่วนต่าง ๆ

จากรูปที่ 3.13 จะเป็นการแสดงรูปแบบการทำงานร่วมกันของโปรแกรมต่าง ๆ ที่ทำหน้าที่ในระบบมาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบ ซึ่งในส่วนของ Mricrocontroller Program จะเป็นโปรแกรมการทำงานที่อยู่บน Microcontroller มีหน้าที่คอยอ่านค่าพารามิเตอร์ทางไฟฟ้าจากมาตรวัดพลังงานไฟฟ้า Eastron SDM230 และคอยส่งข้อมูลขึ้น IoT Cloud Server ผ่าน NB-IoT Shield

ถัดมาในส่วน UDP Server Program จะเป็นโปรแกรมส่วนที่คอยอัพเดทค่า Ft และเดือนในปัจจุบันให้กับ Microcontroller เพื่อให้สามารถคำนวณค่าไฟฟ้าได้ถูกต้องแม่นยำมากขึ้น ซึ่งโปรแกรมในส่วนนี้จะทำงานอยู่บน IoT Cloud Server และสุดท้ายในส่วนของ Thingsboard Program จะเป็น IoT Cloud Platform ที่คอยจัดการข้อมูลที่ได้รับจาก Microcontroller เช่น การ

จัดเก็บข้อมูลลงฐานข้อมูล การแสดงผลบน Dashboard และการส่งข้อมูลไปให้กับระบบให้บริการภายนอก เป็นต้น

3.4.1 ส่วนของ Microcontroller Program

โปรแกรมในส่วนนี้จะมีหน้าที่หลักในการติดต่อกับมาตรวัดพลังงานไฟฟ้า Eastron SDM230 ผ่านทาง MODBUS RTU Protocol ซึ่งทำการเชื่อมต่อเข้ากับ RS485 to TTL Module และ NB-IoT Shield ผ่าน AltSoftSerial ดังรูปที่ 3.2 โดยมี Source Code ดังนี้

```
#include "True_NB_bc95.h"
#include <Wire.h>
#include <SoftwareSerial.h>
#include <SDM.h>
#include <AltSoftSerial.h>
#include <EEPROM.h>

AltSoftSerial Serial2;
SoftwareSerial swSerSDM(12, 13);
SDM sdm(swSerSDM, 4800, NOT_A_PIN);

//Variables
int voltage;
int freq;
int power;
String current;
String unit;
String pf;
String bill;
float unit_chk;
bool over_150_Unit_per_month;
long req_time;
String msg_buf;
float ft;
int cur_month;
int last_month;
float last_unit;

True_NB_bc95 modem;
String udpRemoteIP = "[redacted]";
int udpRemotePort = 5683;
String monthly_port = "32231";
char iotToken[] = "[redacted]\0";

//Define json format
char jsonData1[] =
"{\"voltage\":%d,\"freq\":%d,\"power\":%d,\"unit\":%s}\0";
char jsonData2[] =
"{\"current\":%s,\"bill\":%s,\"pf\":%s,\"m%d\":%s}\0";
char buff1[48];
char buff2[64];
int jsonData_len1 = 0;
int jsonData_len2 = 0;
```

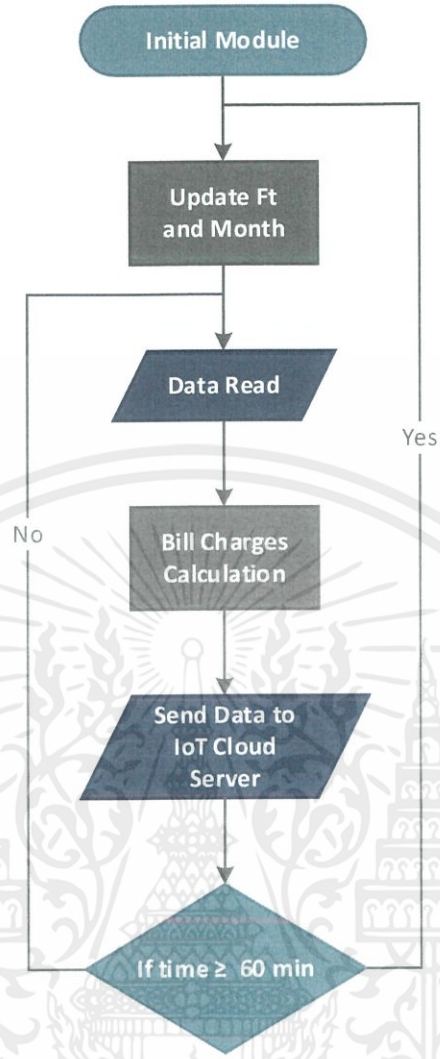
```
CoapPacketTrueIoT coap;
char sock[] = "0\0";
void setup() {
  int eeAddress = 0;
  Serial2.begin(9600);
  sdm.begin();
//initialize SDM communication
  modem.init(Serial2);
  modem.initModem();
  //Serial.println( "IMEI = " + modem.getIMEI() );
  //Serial.println( "IMSI = " + modem.getIMSI() );
  while (!modem.register_network());
  delay(1000);
  //Serial.println( "Device IP = " + modem.check_ipaddr() );
  modem.create_UDP_socket( 4700, sock);
  monthly_req();
  EEPROM.get(eeAddress, last_unit);
  delay(50);
  eeAddress = eeAddress + sizeof(float);
  EEPROM.get(eeAddress, last_month);
}

void loop() {
  delay(30000);
  if(millis() - req_time >= 60*60000){
    bc95_restart();
    monthly_req();
    while(ft == 0 || cur_month == 0){
      bc95_restart();
      monthly_req();
    }
    monthly_proc();
    req_time = millis();
  }
  //Read Sensor values
  voltage = sdm.readVal(SDM230_VOLTAGE);
  delay(100);
  current = sdm.readVal(SDM230_CURRENT);
  delay(100);
  freq = sdm.readVal(SDM230_FREQUENCY);
  delay(100);
  power = sdm.readVal(SDM230_POWER);
  delay(100);
  unit_chk = sdm.readVal(SDM230_TOTAL_ACTIVE_ENERGY) -
last_unit;
  delay(100);
  pf = sdm.readVal(SDM230_POWER_FACTOR);
  delay(100);
  unit = sdm.readVal(SDM230_TOTAL_ACTIVE_ENERGY) - last_unit;
  over_150_Unit_per_month = (unit_chk >= 150) ? true : false;
  bill = calBill(unit_chk, ft, over_150_Unit_per_month);
  tb_send();
}
```

```
void tb_send(){
  sprintf(buff1, jsonData1, (int)voltage, (int)freq,
(int)power, unit.c_str()  );
  jsonData_len1 = strlen(buff1);
  sprintf(buff2, jsonData2, current.c_str(), bill.c_str(),
pf.c_str(), cur_month, unit.c_str()  );
  jsonData_len2 = strlen(buff2);
  //Serial.println(buff2);
  modem.postRequest( iotToken, buff1, &coap );
  modem.sendUDPPacket2( sock, udpRemoteIP, udpRemotePort,
&coap, jsonData_len1);
  delay(250);
  modem.postRequest( iotToken, buff2, &coap );
  modem.sendUDPPacket2( sock, udpRemoteIP, udpRemotePort,
&coap, jsonData_len2);
}

float calBill(float Unit, float ft_rate, bool
over_150_Unit_per_month) {
  float Service = over_150_Unit_per_month ? 38.22 : 8.19;
  float total = 0;
  if (!over_150_Unit_per_month) {
    float Rate15 = 2.3488;
    float Rate25 = 2.9882;
    float Rate35 = 3.2405;
    float Rate100 = 3.6237;
    float Rate150 = 3.7171;
    float Rate400 = 4.2218;
    float RateMore400 = 4.4217;

    if (Unit >= 0) total += min(Unit, 15) * Rate15;
    if (Unit >= 16) total += min(Unit - 15, 10) * Rate25;
    if (Unit >= 26) total += min(Unit - 25, 10) * Rate35;
    if (Unit >= 36) total += min(Unit - 35, 65) * Rate100;
    if (Unit >= 101) total += min(Unit - 100, 50) * Rate150;
    if (Unit >= 151) total += min(Unit - 150, 250) * Rate400;
    if (Unit >= 401) total += (Unit - 400) * RateMore400;
  } else {
    float Rate150 = 3.2484;
    float Rate400 = 4.2218;
    float RateMore400 = 4.4217;

    total += min(Unit, 150) * Rate150;
    if (Unit >= 151) total += min(Unit - 150, 150) * Rate400;
    if (Unit >= 401) total += (Unit - 400) * RateMore400;
  }

  total += Service;
  total += Unit * ft_rate;
  total += total * 7 / 100;

  return total;
}

void monthly_req(){
```

```
  send_udp_request("month");
  delay(5000);
  send_udp_request("ft");
  //Serial.println(cur_month);
}

void send_udp_request(String req_cmd)
{
  if(req_cmd == "month"){
    modem.sendUDPstr(udpRemoteIP, monthly_port, "month");
    delay(5000);
    msg_buf = check_udp_incoming();
    cur_month = string2int(msg_buf);
    //Serial.println("Request sent : month");
    ////Serial.println(ft, 4);
  } else if(req_cmd == "ft"){
    modem.sendUDPstr(udpRemoteIP, monthly_port, "ft");
    delay(5000);
    msg_buf = check_udp_incoming();
    ft = string2float(msg_buf);
    //Serial.println("Request sent : ft");
    ////Serial.println(cur_month);
  }
}

String check_udp_incoming()
{
  String tmp_buf;
  String data_in;
  String str_incom = "";
  int indx = 0;
  int indx1 = 0;

  if (Serial2.available())
  {
    tmp_buf = Serial2.readString();
    ////Serial.println("incomeing" + tmp_buf);
    if (tmp_buf.indexOf("+NSONMI") > 0)
    {
      Serial2.println("AT+NSORF=0,12"); // send read data
from modem
      delay(500);
      if (Serial2.available())
      {
        ////Serial.println("incomeing2");
        data_in = Serial2.readString();
        ////Serial.println(data_in);
        indx = data_in.indexOf("3A"); // find ':'
        indx1 = data_in.indexOf("3B"); // find ';'
        str_incom = data_in.substring(indx + 2, indx1);
        str_incom = modem.hex2string(str_incom);
        //str_lampstts = data_in;
        ////Serial.println("data:" + str_lampstts);
      }
    }
```

```
  }
  return str_incom;
}

float string2float(String ft_str)
{
  float ft_data;
  ft_data = strtol(ft_str.c_str(), 0, 10)/10000.00000;
  return ft_data;
}

int string2int(String int_str)
{
  int int_data;
  int_data = strtol(int_str.c_str(), 0, 10);
  return int_data;
}

void bc95_restart(){
  modem.init(Serial2);
  modem.initModem();
  while (!modem.register_network());
  delay(1000);
  modem.create_UDP_socket( 4700, sock);
}

void monthly_proc()
{
  int eeAddress = 0;
  if(cur_month - last_month == 1 || cur_month - last_month ==
-11 ){
    last_unit = sdm.readVal(SDM230_TOTAL_ACTIVE_ENERGY);
    EEPROM.put(eeAddress, last_unit);
    delay(50);
    eeAddress = eeAddress + sizeof(float);
    EEPROM.put(eeAddress, cur_month);
    last_month = cur_month;
  }
}
```

โดยโปรแกรมในส่วนนี้สามารถนำมาเขียนเป็นแผนภาพแสดงการทำงานได้ดังรูปที่ 3.14



รูปที่ 3.14 การทำงานในส่วนของ Microcontroller Program

จากรูปที่ 3.14 จะเห็นได้ว่าโปรแกรมในส่วนของ Microcontroller Program จะแบ่งการทำงานออกเป็น 6 ส่วน ได้แก่

1. **Initial Module** โปรแกรมในส่วนนี้ จะมีหน้าที่ทำการเริ่มการทำงานของ Module ที่เชื่อมต่ออยู่กับ Microcontroller ได้แก่ RS485 to TTL Module, Eastron SDM230 ที่ทำหน้าที่ติดต่อสื่อสารเพื่อรับค่าพารามิเตอร์ทางไฟฟ้าที่วัดได้ และ True NB-IoT Shield เพื่อทำการเชื่อมต่อไปยังสถานีฐานสำหรับการส่งข้อมูลขึ้น IoT Cloud Server
2. **Update Ft and Month** โปรแกรมในส่วนนี้ จะมีหน้าที่ในการส่งคำสั่งร้องขอขอมูลค่า Ft และลำดับเดือนในปัจจุบันจาก IoT Cloud Server ผ่านการส่ง UDP Datagram โดยเมื่อส่งคำสั่งร้องขอเรียบร้อยแล้ว IoT Cloud Server จะตอบกลับค่า Ft และลำดับเดือนกลับมาเพื่อใช้สำหรับการคำนวณค่าไฟฟ้า และการรายงานผลเชิงสถิติ
3. **Data Read** โปรแกรมในส่วนนี้ จะมีหน้าที่ในการส่งคำสั่งร้องขอข้อมูลในรูปแบบ MODBUS Protocol ไปยังมาตรวัดพลังงานไฟฟ้า Eastron SDM230 ผ่านทาง RS485 to TTL

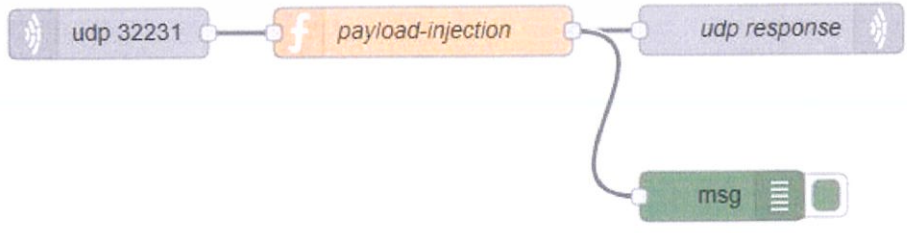
Module ซึ่งจะได้ข้อมูลพารามิเตอร์ทางไฟฟ้ากลับมา เพื่อนำไปใช้งานในการคำนวณค่าไฟฟ้าและการรายงานผล

4. **Bill Charges Calculation** โปรแกรมในส่วนนี้ จะมีหน้าที่ในการคำนวณค่าไฟฟ้าในรอบบิลเดือนปัจจุบัน โดยใช้ปริมาณการใช้ไฟฟ้าในหน่วย kWh ที่ได้จากการทำงานของโปรแกรมในส่วนที่ 3 และค่า Ft จากการทำงานของโปรแกรมในส่วนที่ 2 มาประกอบในการคำนวณค่าใช้ไฟฟ้า
5. **Send Data to IoT Cloud Server** โปรแกรมในส่วนนี้ จะมีหน้าที่ส่งข้อมูลพารามิเตอร์ที่วัดได้ในปัจจุบันจากโปรแกรมในส่วนที่ 3 และค่าไฟฟ้าที่คำนวณได้จากโปรแกรมในส่วนที่ 4 ไปยัง IoT Cloud Server เพื่อทำการจัดเก็บข้อมูลลงฐานข้อมูลและนำข้อมูลไปใช้ต่อกับผู้ให้บริการภายนอก
6. **If time ≥ 60 min** โปรแกรมในส่วนนี้จะมีหน้าที่ตรวจสอบว่า Microcontroller ทำงานไปแล้วนานเท่าไร หากมีการทำงานมากกว่า 60 นาที จะมีการส่งขั้นตอนการทำงานไปยังโปรแกรมในส่วนที่ 2 เพื่อทำการอัพเดทค่า Ft และลำดับเดือนในปัจจุบัน หากต่ำกว่า 60 นาที จะส่งไปยังขั้นตอนการทำงานของโปรแกรมในส่วนที่ 3

จะเห็นได้ว่าการทำงานของโปรแกรมในส่วนของ Microcontroller Program จะมีการทำงานเป็นแบบวนซ้ำ (Loop) โดยเมื่อ Microcontroller มีการทำงานครบ 60 นาทีก็จะมีการทำการอัพเดทค่า Ft และลำดับเดือนโดยอัตโนมัติ

3.4.2 ส่วนของ UDP Server Program

โปรแกรมในส่วนนี้จะมีหน้าที่หลักในการอัพเดทค่า Ft และลำดับเดือนในปัจจุบันที่ถูกต้องให้กับ Microcontroller เพื่อให้การคำนวณค่าไฟฟ้าและการบันทึกข้อมูลพารามิเตอร์ทางไฟฟ้าเป็นไปอย่างถูกต้อง โดยทางผู้จัดทำจะนำ Node-Red เข้ามาประยุกต์ใช้กับการเขียนโปรแกรมโดยที่ Node-Red นั้นจะเป็นเครื่องมืออำนวยความสะดวกในการเขียนโปรแกรมภาษา JavaScript ในรูปแบบ Flow-based Programming โดยมีการเขียน Flow การทำงานดังนี้



รูปที่ 3.15 Flow การทำงานของโปรแกรมในส่วนของ UDP Server Program

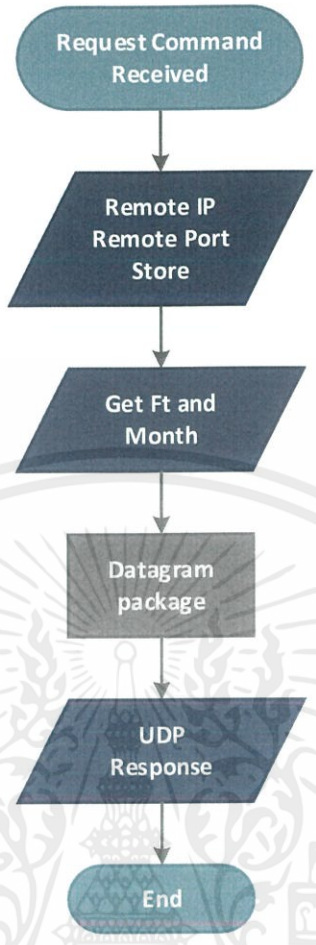
จากรูปที่ 3.15 จะเห็นได้ว่าโปรแกรมในส่วนของ Microcontroller Program จะแบ่งการทำงานออกเป็น 3 ส่วน ได้แก่

1. **udp 32231** โหนดนี้จะเป็นส่วนของ UDP Server ที่คอยรอรับคำสั่งร้องขอข้อมูล Ft และลำดับเดือนในปัจจุบันจาก Microcontroller ที่จะส่งเข้ามาที่พอร์ท 32231 และคอยทำการบันทึก IP และ Port ของ Microcontroller ที่ส่งคำสั่งร้องขอข้อมูลเข้ามาไว้เพื่อใช้ในการตอบกลับ
2. **payload-injection** โหนดนี้จะเป็นส่วนที่คอยบรรจุข้อมูลค่า Ft และลำดับเดือนลงใน Datagram เพื่อเตรียมการส่งกลับไปยัง Microcontroller โดยมี Source Code ของโหนดดังนี้

```
var ft = -1590;
var month = new Date().getMonth() + 1;
if(msg.payload == "month"){
    msg.payload = ":"+month+";";
} else if(msg.payload == "ft"){
    msg.payload = ":"+ft+";";
}
return msg;
```

3. **udp response** โหนดนี้จะทำหน้าที่ส่งข้อมูลที่ได้จากโหนดในข้อ 2 กลับไปยัง Microcontroller ที่ได้ทำการร้องขอข้อมูลเข้ามาตาม IP และ Port ที่ได้บันทึกไว้จากโหนดในข้อที่ 1

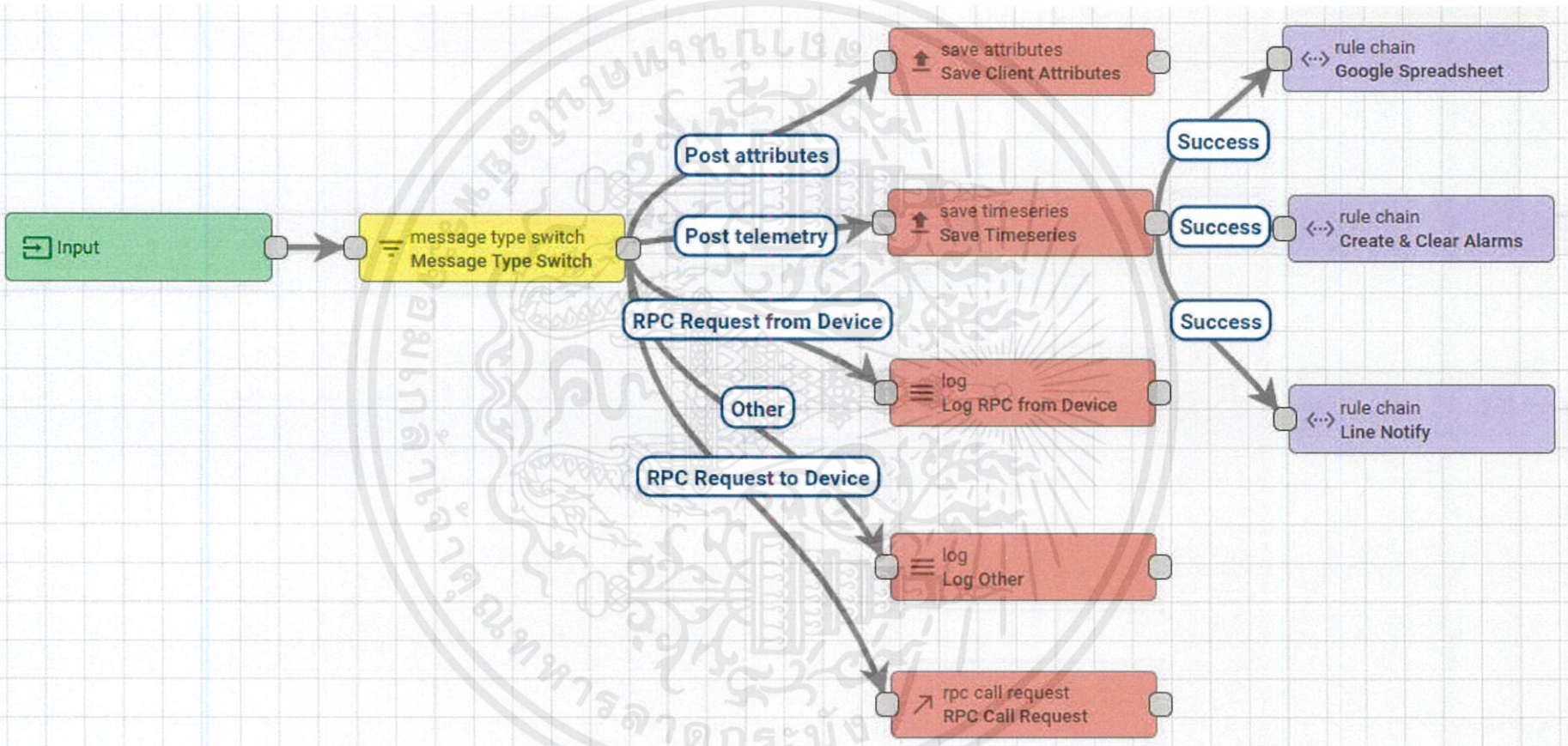
โดยโปรแกรมในส่วนของ UDP Server Program ทั้งหมดนั้นสามารถนำมาเขียนเป็นแผนภาพแสดงการทำงานได้ดังนี้



รูปที่ 3.16 การทำงานในส่วนของ UDP Server Program

3.4.3 ส่วนของ Thingsboard Program

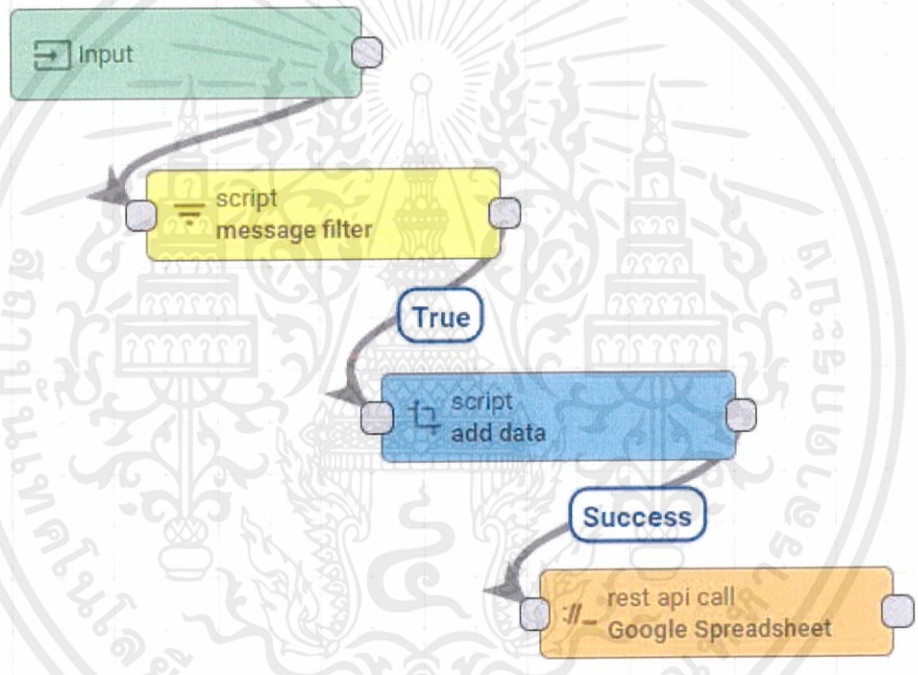
โปรแกรมส่วนนี้จะเป็นส่วนของ Thingsboard ซึ่งเป็น IoT Cloud Platform ที่ทำงานอยู่บน IoT Cloud Server โดย Thingsboard นั้นจะเป็น IoT Cloud Platform แบบ Open-source สามารถนำมาปรับแต่งฟังก์ชันการแสดงผล และฟังก์ชันการประมวลผลข้อมูลได้โดยใช้ Rule Chain ในการกำหนดฟังก์ชันการทำงาน ซึ่งเป็นการเขียนโปรแกรมภาษา JavaScript ในรูปแบบ Flow-based Programming โดยมี Root Rule Chain ที่เป็น Flow หลักดังนี้



รูปที่ 3.17 Flow การทำงาน Root Rule Chain ของ Thingsboard

จากรูปที่ 3.17 จะเห็นได้ว่าเมื่อมีข้อมูลที่ถูกส่งมาจาก Microcontroller มายัง Port 5683 ของ IoT Cloud Server แล้ว ข้อมูลจะถูกนำไปแยกประเภทของข้อมูลผ่าน Flow Message Type Switch โดยข้อมูลที่ได้รับจาก Microcontroller นั้นจะเป็นประเภท Post telemetry ซึ่งข้อมูลดังกล่าวจะถูกส่งต่อไปยัง Flow ฟังก์ชันเสริมของระบบ ประกอบด้วย

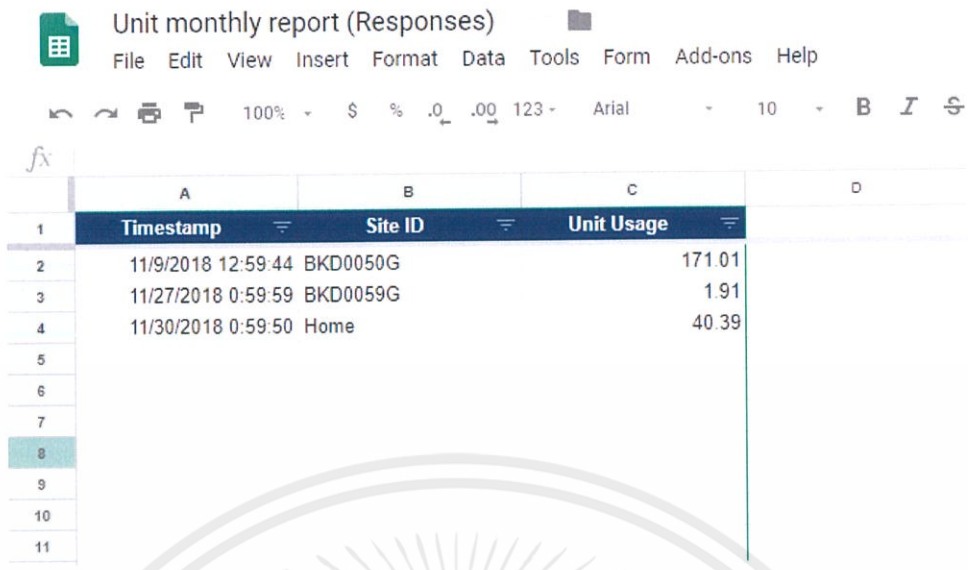
1. **Google Spreadsheet** โดย Flow การทำงานนี้จะเป็นการส่งข้อมูล Site ID และปริมาณการใช้งานพลังงานไฟฟ้าในหน่วย kWh ไปยัง Google Spreadsheet เพื่อทำรายงานปริมาณการใช้งานพลังงานไฟฟ้าของแต่ละไซต์งานในหนึ่งเดือน โดย Flow จะทำการส่งข้อมูลโดยใช้โปรโตคอล HTTP ด้วยวิธี GET Method ซึ่งมีการเขียนโปรแกรมของ Flow ดังรูปที่ 3.18



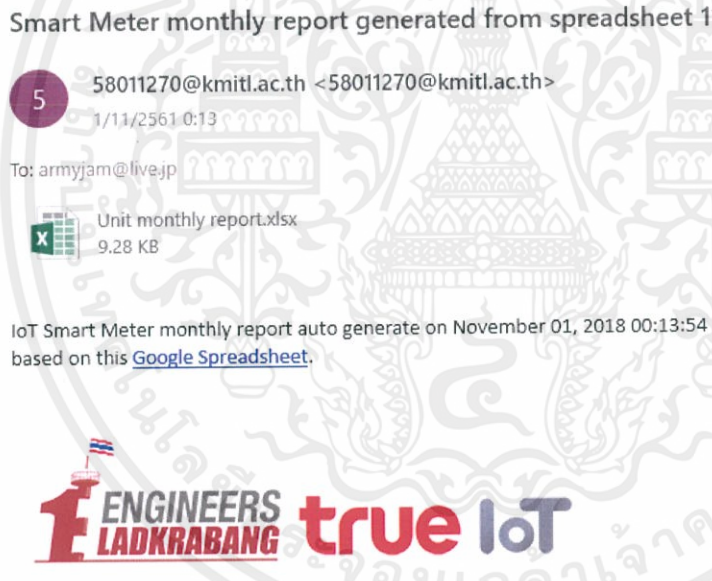
รูปที่ 3.18 Flow การทำงาน Google Spreadsheet ของ Thingsboard

โดยเมื่อข้อมูลถูกส่งไปยัง Google Spreadsheet เรียบร้อยแล้ว จะถูกจัดเก็บในรูปแบบเอกสารไฟล์ Excel ดังรูปที่ 3.19 และผู้จัดทำได้ทำการเขียน Macro ให้ระบบส่งรายงานปริมาณการใช้งานพลังงานไฟฟ้าไปยังอีเมล์ของผู้ดูแลระบบในทุก ๆ ต้นเดือน เพื่อให้สามารถนำรายงานไปเปรียบเทียบกับรายงานจากทางการไฟฟ้าฯ ได้ดังรูปที่ 3.20

รายงานปริมาณการใช้พลังงานไฟฟ้าในรูปแบบไฟล์ Excel บน Google Spreadsheet

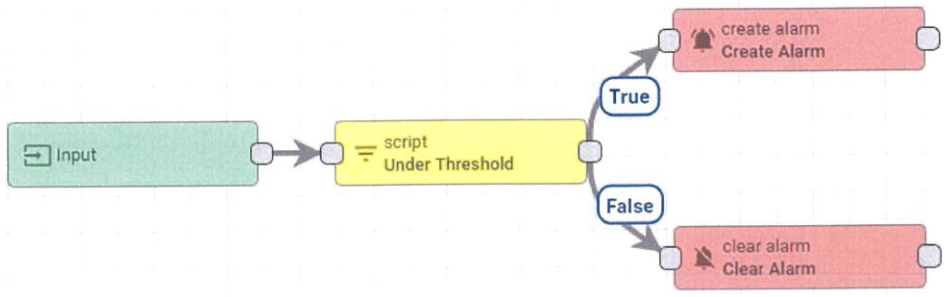


รูปที่ 3.19 รายงานปริมาณการใช้พลังงานไฟฟ้าในรูปแบบไฟล์ Excel บน Google Spreadsheet



รูปที่ 3.20 การส่งอีเมล์รายงานในรูปแบบไฟล์ Excel ทุก ๆ ต้นเดือน

2. **Create & Clear Alarms** โดย Flow การทำงานนี้จะเป็นการสร้าง Alarm ในกรณีที่ไซต์งานนั้น ๆ มีการใช้งานปริมาณเกิน Threshold ที่กำหนดไว้ โดยจะมีการแจ้งเตือนไปยัง Dashboard ของระบบ มีการเขียนโปรแกรมของ Flow ดังรูปที่ 3.21

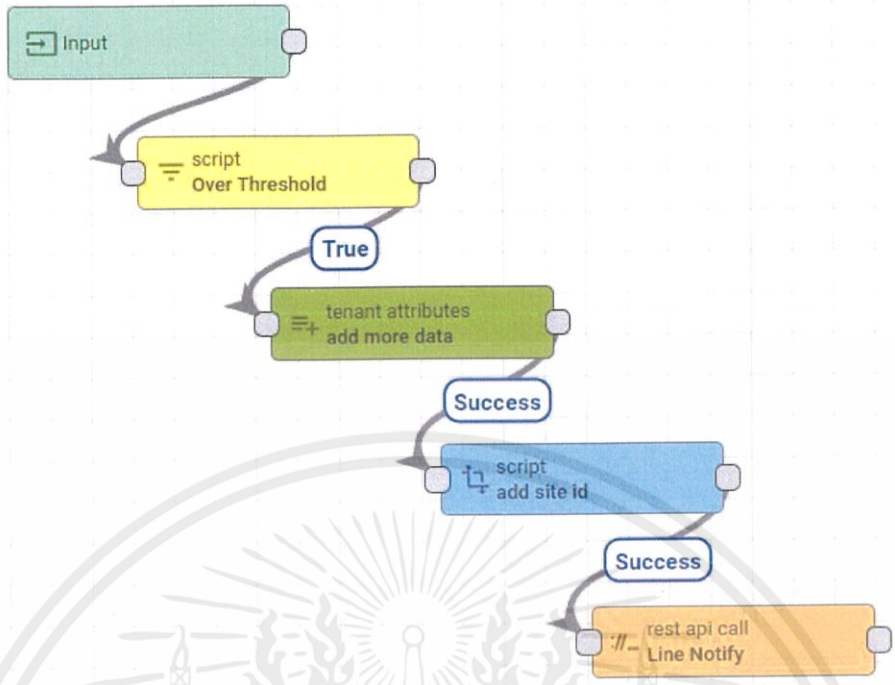


รูปที่ 3.21 Flow การทำงาน Create & Clear Alarms ของ Thingsboard

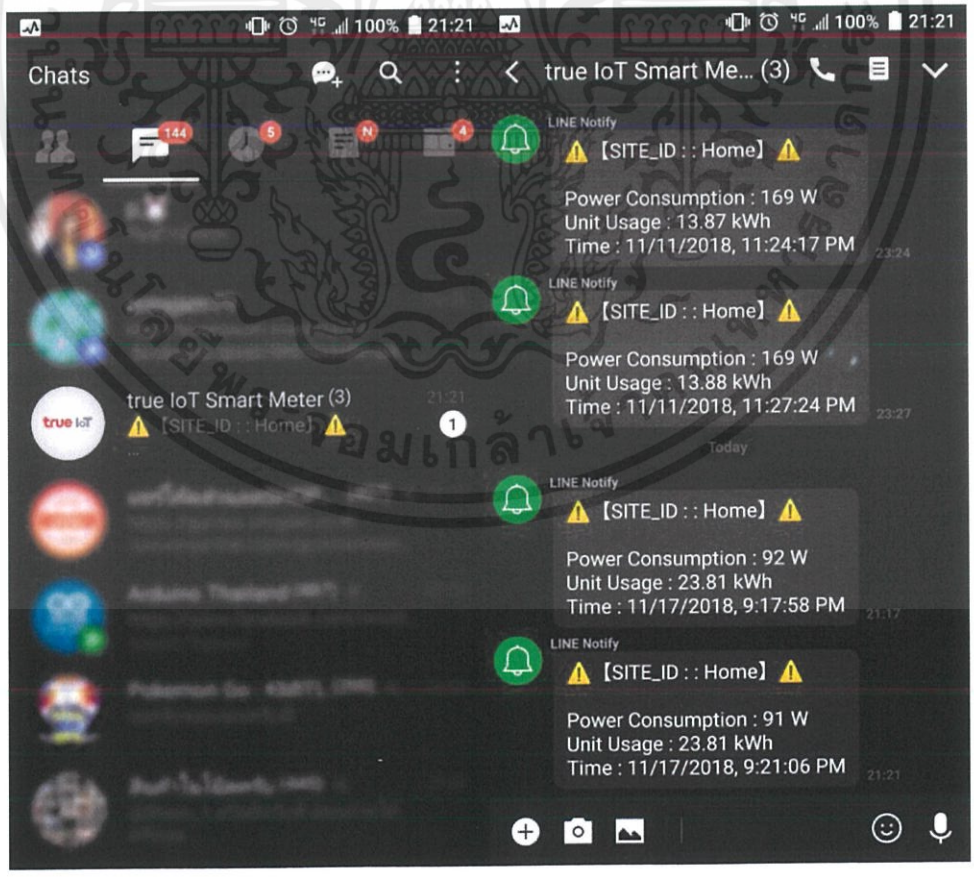
3. **Line Notify** โดย Flow นี้จะทำงานเมื่อมีปริมาณการใช้พลังงานไฟฟ้าเกิน Threshold ที่กำหนดไว้ โดยจะทำการส่งข้อมูล Site ID ปริมาณการใช้งานพลังงานไฟฟ้าในหน่วย kWh และค่าไฟฟ้าจากการประมาณการไปยัง Service Line Notify เพื่อแจ้งเตือนปริมาณการใช้งานพลังงานไฟฟ้าของแต่ละไซต์งาน โดย Flow จะทำการส่งข้อมูลโดยใช้โปรโตคอล HTTP ด้วยวิธี POST Method โดยมีการตั้งค่า HTTP Request Field ดังตารางที่ 3.1 ซึ่งมีการเขียนโปรแกรมของ Flow ดังรูปที่ 3.22

ตารางที่ 3.1 HTTP Post Request Field ของระบบ Line Notify

Headers	
Content-type	application/x-www-form-urlencoded
Authorization	Bearer [redacted]
Payloads (JSON)	
Message	`msg.payload.siteid + '] ⚠ \n\nPower Consumption : '+msg.payload.power + ' W' + '\nUnit Usage : ' + msg.payload.unit + ' kWh' + '\nTime : ' + datetime`



รูปที่ 3.22 Flow การทำงาน Create & Clear Alarms ของ Thingsboard



รูปที่ 3.23 การแจ้งเตือนปริมาณการใช้งานไฟฟ้าผ่านแอพลิเคชั่น Line

3.5 การออกแบบ Cloud Server

ในส่วนของ Cloud Server นั้นผู้จัดทำได้ใช้บริการ Virtual Machine ของ Google Cloud เนื่องจากสามารถติดตั้งใช้งานได้ง่าย และมีการให้เครดิตสำหรับใช้งานฟรีในกรณีศึกษา โดยได้มีรายละเอียดเชิงเทคนิคของเครื่อง Server ตารางที่ 3.2

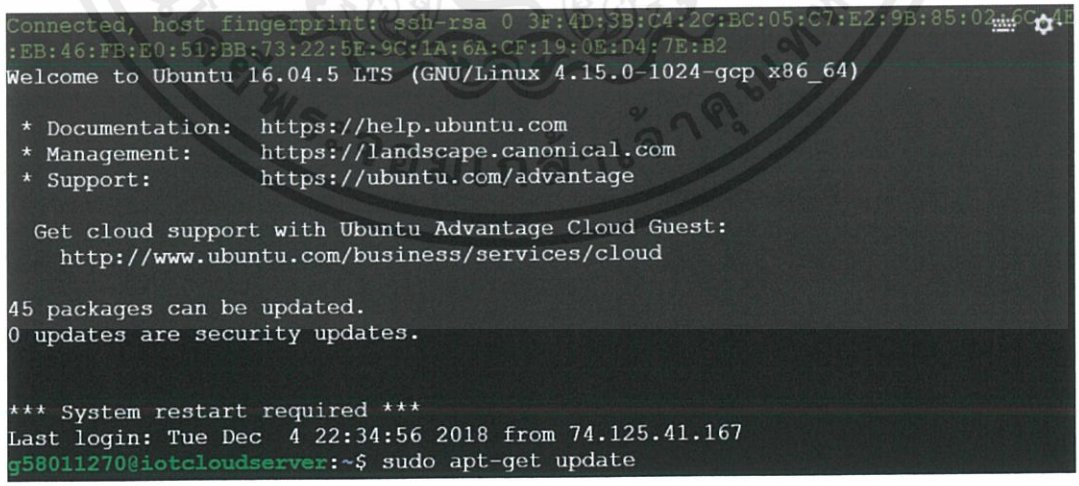
ตารางที่ 3.2 รายละเอียดเชิงเทคนิคของ Cloud Server

Operating System	Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-1024-gcp x86_64)
CPU	1vCPU (Intel Ivy Bridge)
Memory	3 GB
Storage	HDD 20 GB
Zone	asia-east1-b
Host	blackarmy.tech
Network	Google cloud asia-east1 default networks

โดยที่ Cloud Server ที่ได้สร้างขึ้นมานั้น จะมีการติดตั้ง Service ที่ไว้คอยให้บริการทั้งหมด 2 ตัว ได้แก่ Node-Red และ Thingsboard

3.5.1 การติดตั้ง Node-Red

Node-Red มีหน้าที่เป็น Flow-based Programming สำหรับภาษา JavaScript ซึ่ง Node-red จะคอยทำงานร่วมกับ NodeJS โดยมีขั้นตอนการติดตั้ง Node-Red บนระบบปฏิบัติการ Ubuntu 16.04.5 โดยใช้ Command Terminal ของระบบปฏิบัติการดังรูปที่ 3.24



รูปที่ 3.24 การใช้งาน Command Terminal ของ IoT Cloud Server

1. อัพเดทแพ็กเกจของระบบปฏิบัติการให้เป็นเวอร์ชั่นล่าสุด

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

2. ติดตั้ง NodeJS

```
$ sudo apt-get install nodejs-legacy
```

3. ติดตั้ง NPM Package Manager

```
$ sudo apt-get install npm
```

4. ติดตั้ง Node-Red

```
$ sudo npm install -g --unsafe-perm node-red node-red-admin
```

5. เพิ่มข้ออนุญาต Firewall ของ Port 1880 สำหรับ Node-Red

```
$ sudo ufw allow 1880
```

6. ตั้งค่า Node-Red ให้เป็น system service

```
$ sudo vi /etc/systemd/system/node-red.service
```

จากนั้นแก้ไขไฟล์ node-red.service ให้เป็นดังนี้

```
[Unit]
Description=Node-RED
After=network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/node-red-pi --max-old-space-size=128
-v
Restart=on-failure
KillSignal=SIGINT

WorkingDirectory=/home/adrian
User=adrian

[Install]
WantedBy=multi-user.target
```

7. ทำการรัน Node-Red Service

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable node-red.service
$ sudo systemctl start node-red.service
$ sudo systemctl status node-red.service
```

3.5.2 การติดตั้ง Thingsboard

Thingsboard มีหน้าที่คอยรับข้อมูลที่ถูกส่งมาจาก Microcontroller ผ่านทางโปรโตคอล CoAP บน Port 5683 ของ Cloud Server โดยจะมีการทำงานร่วมกับ Java Service โดยมีขั้นตอนการติดตั้ง Thingsboard บนระบบปฏิบัติการ Ubuntu 16.04.5 โดยใช้ Command Terminal ของระบบปฏิบัติการดังนี้

1. อัพเดทแพ็กเกจของระบบปฏิบัติการให้เป็นเวอร์ชั่นล่าสุด

```
$ sudo apt-get update
```

2. ติดตั้ง Java Runtime Environment (JRE)

```
$ sudo apt-get install default-jre
```

3. ติดตั้ง Java Development Kit (JDK)

```
$ sudo apt-get install default-jdk
```

4. ดาวน์โหลด Oracle JDK

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
```

5. ติดตั้ง Oracle JDK 8

```
$ sudo apt-get install oracle-java8-installer
```

6. ดาวน์โหลด Thingsboard Installation Package

```
$ wget
https://github.com/thingsboard/thingsboard/releases/download/
v2.2/thingsboard-2.2.deb
```

7. ติดตั้ง Thingsboard

```
$ sudo dpkg -i thingsboard-2.2.deb
```

8. ทำการรัน Thingsboard Service

```
$ sudo service thingsboard start
```

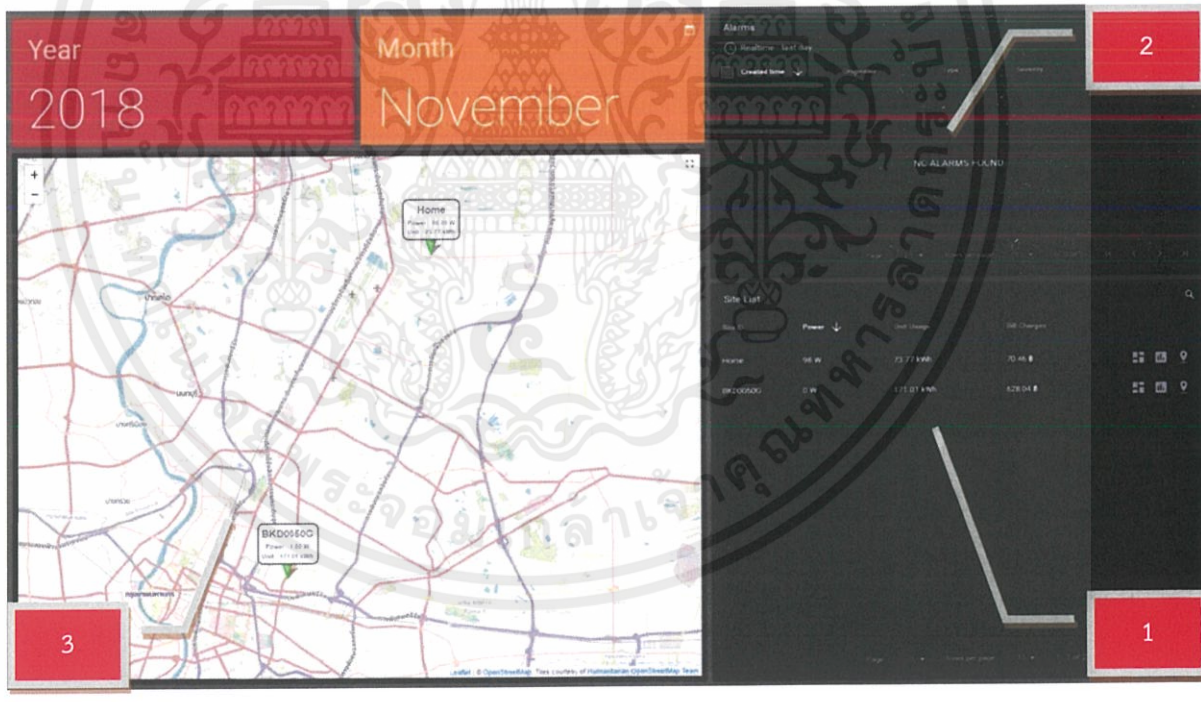
3.6 การออกแบบ Dashboard

หลังจากผู้จัดทำได้นำ IoT Cloud Platform ของ Thingsboard มาติดตั้งเข้ากับ Virtual Machine ของทาง Google cloud ซึ่ง Thingsboard นั้นเป็น IoT Cloud Platform ที่เป็น Open-source software จึงไม่มีค่าใช้จ่ายในการนำมาติดตั้ง และยังสามารถปรับแต่ง Widget และ Rule-chain ต่าง ๆได้อย่างอิสระ โดย Thingsboard นั้นจะใช้ภาษา JavaScript เป็นภาษาหลักในการปรับแต่งค่าต่าง ๆ และใช้การรับส่งข้อมูลที่อยู่ในรูปแบบ JSON data format

โดยผู้จัดทำได้มีการออกแบบหน้าต่าง Dashboard เป็น 4 ส่วนหลักๆ ได้แก่

1. Main state Dashboard
2. Real-time Dashboard state
3. Device details state
4. Yearly unit usage state

ซึ่งมีรูปแบบและการแสดงผลข้อมูลดังนี้

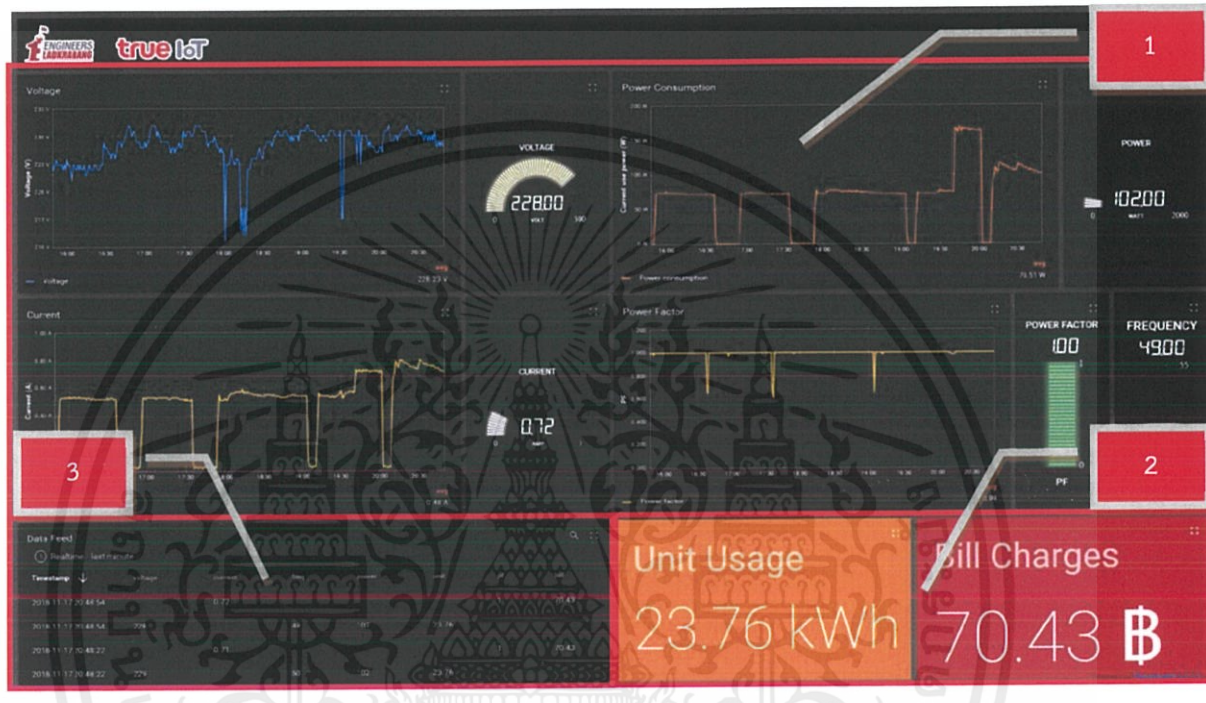


รูปที่ 3.25 หน้าต่าง Main state dashboard

จากรูปที่ 3.25 จะมีการแสดงผลข้อมูลเป็น 3 ส่วนหลักๆ ได้แก่ส่วนที่ 1 คือตารางแสดงรายการไซต์งานของที่ได้นำมาตรวัดพลังงานไฟฟ้าไปติดไว้ ซึ่งจะมีข้อมูล Site ID กำลังไฟฟที่กำลังใช้ ปริมาณการใช้ไฟฟ้าในเดือนนั้นๆ และการประมาณการค่าไฟฟ้า

ส่วนที่ 2 เป็นการแสดง Site Alarm ที่เคยเกิดขึ้นในช่วงเวลาที่ผ่านมา ซึ่งหน้าต่างนี้ สามารถกำหนดเงื่อนไขในการแจ้งเตือนได้ เช่น การแจ้งเตือนเมื่อมีการใช้กำลังไฟฟ้าเกินที่กำหนดไว้ เป็นต้น

ส่วนที่ 3 เป็นการแสดงที่อยู่ของไซต์งานแต่ละไซต์ ซึ่งมีการระบุตำแหน่งด้วยพิกัด Latitude และ Longitude และสีของ Pin ในแต่ละไซต์งานนั้นเป็น Indicator ที่บ่งบอกระดับการใช้กำลังไฟฟ้าตามเงื่อนไขที่กำหนดไว้ เช่น หากใช้งานไฟฟ้ามากกว่า 1,000 วัตต์ Pin จะเปลี่ยนเป็นสีแดง เป็นต้น



รูปที่ 3.26 หน้าต่าง Real-time dashboard state

จากรูปที่ 3.26 เป็นการแสดงผลข้อมูลพารามิเตอร์ทางไฟฟ้าของแต่ละไซต์งานแบบเรียลไทม์ โดยไซต์งานจะถูกเลือกจากหน้า Main state dashboard ซึ่งหน้าต่างนี้มีจะมีการแสดงผลอยู่ทั้งหมด 3 ส่วนหลักๆ ได้แก่ ส่วนที่ 1 เป็นการแสดงค่าพารามิเตอร์ทางไฟฟ้า ประกอบไปด้วย ค่าความต่างศักย์ ค่ากำลังไฟฟ้าที่ใช้อยู่ ปริมาณกระแสไฟฟ้า ตัวประกอบกำลัง และความถี่

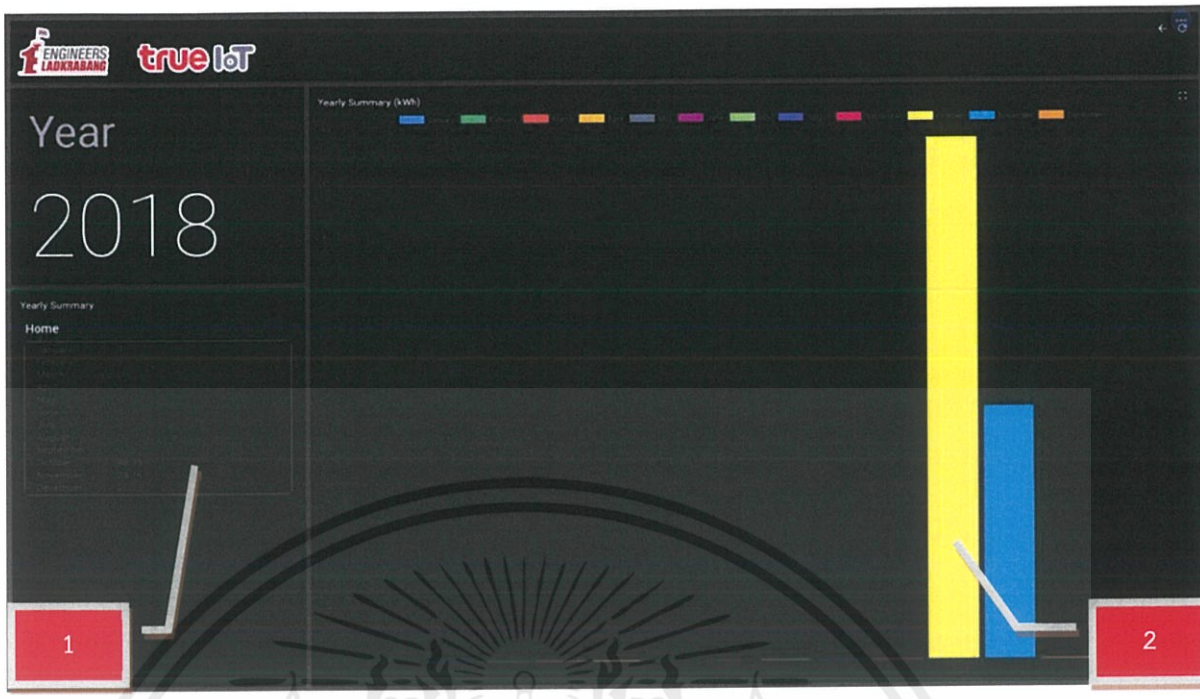
ส่วนที่ 2 เป็นการแสดงปริมาณการใช้งานไฟฟ้าในหน่วย kWh ในรอบเดือนปัจจุบัน และค่าไฟฟ้าจากการประมาณการจากปริมาณการใช้งาน

ส่วนที่ 3 เป็นการแสดงชุดข้อมูลล่าสุดที่ถูกส่งมาจากมาตรวัดพลังงานไฟฟ้า ซึ่งประกอบไปด้วยค่าความต่างศักย์ ประมาณกระแสไฟฟ้า ความถี่ ค่ากำลังไฟฟ้าที่ใช้อยู่ ตัวประกอบกำลัง และค่าไฟฟ้าจากการประมาณการ

รูปที่ 3.27 หน้าต่าง Device details state

จากรูปที่ 3.27 จะเป็นหน้าที่แสดงข้อมูลรายละเอียดเกี่ยวกับไซต์งานซึ่งจะถูกเลือกจากการกดที่ Pin ของไซต์งานจากหน้า Main state dashboard โดยจะมีการแสดงข้อมูลอยู่ 2 ส่วนหลักๆ ได้แก่ ส่วนที่ 1 จะเป็นการแสดงผลที่อยู่ของไซต์งานบนแผนที่ และจะมีการแสดงข้อมูลปริมาณกำลังไฟฟ้าที่ใช้งานในปัจจุบัน กับปริมาณการใช้งานไฟฟ้าในหน่วย kWh ในรอบเดือนปัจจุบัน

ส่วนที่ 2 เป็นการแสดงรายละเอียดข้อมูลของไซต์งาน ประกอบไปด้วย กำลังไฟฟ้าที่ใช้งานในปัจจุบัน ปริมาณการใช้งานไฟฟ้าในหน่วย kWh ในรอบเดือนปัจจุบัน ค่าไฟฟ้าที่ได้จากการประมาณการ ที่อยู่ของไซต์งาน และตำแหน่งในรูปแบบ Latitude Longitude



รูปที่ 3.28 หน้าต่าง Yearly unit usage summary

จากรูปที่ 3.28 จะเป็นการแสดงผลผลสรุปปริมาณการใช้งานพลังงานไฟฟ้าในแต่ละเดือนของปีปัจจุบัน ซึ่งไซต์งานที่นำมาแสดงผลจะถูกเลือกจากหน้า Main state dashboard โดยจะมีการแสดงผลข้อมูลที่ถูกแบ่งเป็นสองส่วนหลักๆ ได้แก่ ส่วนที่ 1 การแสดงผลปริมาณการใช้งานพลังงานไฟฟ้าในหน่วย kWh ในรูปแบบตาราง

ส่วนที่ 2 เป็นการแสดงผลปริมาณการใช้งานพลังงานไฟฟ้าในหน่วย kWh ในหน่วย kWh ที่เป็นรูปแบบกราฟแท่ง

3.7 การจับเก็บผลการทดลอง

3.7.1 การทดลองการทำงานของ NB-IoT Shield

ในการทดลองนี้จะเป็นการทดลองเขียนคำสั่งเพื่อให้ Microcontroller สั่งงาน NB-IoT Shield ผ่านการสื่อสารแบบอนุกรมให้เริ่มต้นการทำงาน และทำการเชื่อมต่อเข้ากับเครือข่าย NB-IoT ของผู้ให้บริการ โดยมีการเขียนโปรแกรมคำสั่งดังนี้

```
#include "True_NB_bc95.h" // NB-IoT Shield Library
#include <AltSoftSerial.h> // Software Serial Library

AltSoftSerial Serial2;
True_NB_bc95 modem;
CoapPacketTrueIoT coap;

void setup() {
```

```
  Serial.begin(9600); // Start Serial Communication (RX0,
TX1)
  Serial2.begin(9600); // Start Serial Communication (RX8,
TX9)
  delay(3000);
  Serial.println("Starting...");
  modem.init(Serial2); // Send initial command to NB-IoT
Shield by Software Serial
  modem.initModem(); // Shield Initial command
  Serial.println( "IMEI = " + modem.getIMEI() ); // Print NB-
IoT Shield IMEI no.
  Serial.println( "IMSI = " + modem.getIMSI() ); // Print NB-
IoT Shield IMSI no.
  while (!modem.register_network()); // Connect to NB-IoT
Shield network
  delay(1000);
  Serial.println( modem.check_ipaddr() ); // Print Private IP
address
}

void loop() {
  delay(1000);
}
```

โดยที่โปรแกรมนี้จะเป็นการเรียกใช้ Library True_NB_bc95 เพื่อนำเข้าชุดคำสั่งสำหรับการสั่งงาน NB-IoT Shield โดยจะการทำงานหลักๆ 2 ส่วน นั่นคือ modem.initModem เพื่อเริ่มการทำงานของ NB-IoT Shield และ modem.register_network เพื่อเชื่อมต่อ NB-IoT Shield เข้ากับเครือข่าย NB-IoT ของผู้ให้บริการ

3.7.2 การทดลองส่งข้อมูลไปยัง IoT-Cloud Server

ในการทดลองนี้จะเป็นการทดสอบส่งข้อมูลจาก NB-IoT Shield ไปยัง Cloud Server ผ่านโพรโตคอล CoAP โดยได้มีการจำลองชุดข้อมูลที่อยู่ในรูปแบบ JSON Data Format ขึ้นมาสำหรับการทดลองส่งข้อมูล ซึ่งได้มีการเขียนโปรแกรมสำหรับการส่งข้อมูลดังนี้

```
#include "True_NB_bc95.h" // NB-IoT Shield Library
#include <AltSoftSerial.h> // Software Serial Library

AltSoftSerial Serial2;
True_NB_bc95 modem;
CoapPacketTrueIoT coap;

String udpRemoteIP = "23.101.19.163"; // Server IP address
int udpRemotePort = 5683; // Server CoAP service port

char iotToken[] = "ngSQG83rFeo9nVsvj0Bx\0"; // access token
char jsonData[] = "{\"temperature\":%d, \"humidity\":%d }\0";
// json Data format

char buff[48];
```

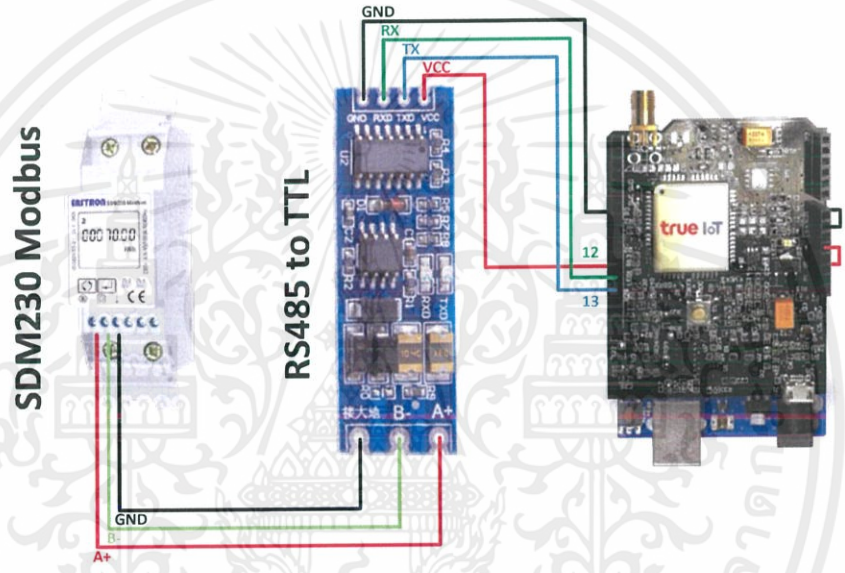
```
int jsonData_len = 0;
long start = 0;
char sock[] = "0\0";

CoapPacketTrueIoT coap;

void setup() {
  Serial.begin(9600); // Start Serial Communication (RX0,
TX1)
  Serial2.begin(9600); // Start Serial Communication (RX8,
TX9)
  delay(3000);
  Serial.println("Starting...");
  modem.init(Serial2); // Send initial command to NB-IoT
Shield by Software Serial
  modem.initModem(); // Shield Initial command
  Serial.println( "IMEI = " + modem.getIMEI() ); // Print NB-
IoT Shield IMEI no.
  Serial.println( "IMSI = " + modem.getIMSI() ); // Print NB-
IoT Shield IMSI no.
  while (!modem.register_network()); // Connect to NB-IoT
Shield network
  delay(1000);
  Serial.println( modem.check_ipaddr() ); // Print Private IP
address
  modem.create_UDP_socket( 4700, sock);
}


void loop() {
  delay(3000);
  if (millis() - start > 9000) { // Repeat Every 9 sec.
    start = millis(); // store count time
    sprintf(buff, jsonData, (int)random(10,20),
(int)random(50,60) ); // Parse random value to data packet
    jsonData_len = strlen(buff); // Calculate the length of
data
    Serial.println(buff); // Display the packet
    modem.postRequest( iotToken, buff, &coap ); //
PostRequest
    modem.sendUDPPacket2( sock, udpRemoteIP, udpRemotePort,
&coap, jsonData_len); // Send Datagram
  }
}
```

การทำงานของโปรแกรมนี้ในส่วนของฟังก์ชัน void setup() นั้นจะเป็นการเริ่มต้นการทำงานของ NB-IoT Shield เหมือนกับในโปรแกรมในข้อ 3.7.1 ซึ่งการทำงานหลักของโปรแกรมนี้จะอยู่ในฟังก์ชัน void loop() ซึ่งจะเป็นการสุ่มชุดข้อมูลที่อุณหภูมิและความชื้นสัมพัทธ์ ที่อยู่ในรูปแบบของ JSON format ขึ้นมา และทำการส่งไปยัง Cloud server ด้วยคำสั่ง modem.sendUDPPacket2

3.7.3 การทดลองเชื่อมต่อ Microcontroller กับมาตรวัดพลังงานไฟฟ้า

ในการทดลองนี้จะเป็นการนำมาตรวัดพลังงานไฟฟ้า Eastron SDM230 Modbus มาเชื่อมต่อเข้ากับ Microcontroller เพื่อทำการอ่านค่าความต่างศักย์(V) กระแส(A) ความถี่(Hz) กำลังงาน(W) ปริมาณการใช้งาน(kWh) และตัวประกอบกำลัง

เนื่องจากมาตรวัดพลังงานไฟฟ้า Eastron SDM230 Modbus นั้นรองรับการสื่อสารด้วยโพรโตคอล RS485 Modbus ซึ่งเป็นโพรโตคอลที่สื่อสารแบบอนุกรมผ่านสายนำสัญญาณแบบอนุกรม จึงต้องมีการเชื่อมต่อกับโมดูลแปลงรูปแบบสัญญาณจาก RS485 ให้เป็น UART TTL เพื่อเชื่อมต่อกับ Microcontroller Arduino UNO R3



รูปที่ 3.29 การเชื่อมต่อมาตรวัดพลังงานไฟฟ้า Eastron SDM230 เข้ากับ Microcontroller

จากรูปที่ 3.29 โมดูล RS485 to TTL จะมีหน้าที่แปลงรูปแบบสัญญาณ RS485 Modbus ให้เป็นสัญญาณ UART TTL เพื่อทำการเชื่อมต่อเข้ากับช่องทางการสื่อสารแบบอนุกรมของ Microcontroller โดยในที่นี้จะทำการใช้ Software Serial มาเพิ่มช่องทางการสื่อสารอนุกรมใน Pin 12 และ 13 เนื่องจาก Arduino UNO R3 นั้นมีช่องทางการสื่อสารอนุกรมเพียงช่องทางเดียว จากนั้นจึงทำการเขียนโปรแกรมเพื่อทำการอ่านค่าความต่างศักย์(V) กระแส(A) ความถี่(Hz) กำลังงาน(W) ปริมาณการใช้งาน(kWh) และตัวประกอบกำลัง ซึ่งสามารถเขียนโปรแกรมควบคุมการทำงานได้ดังนี้

```
#include <SoftwareSerial.h>                //import SoftwareSerial
library
#include <SDM.h>                           //import SDM library
SoftwareSerial swSerSDM(12,13);            //config SoftwareSerial
(rx->pin12 / tx->pin13)
SDM sdm(swSerSDM, 9600, NOT_A_PIN);        //config SDM

void setup() {
```

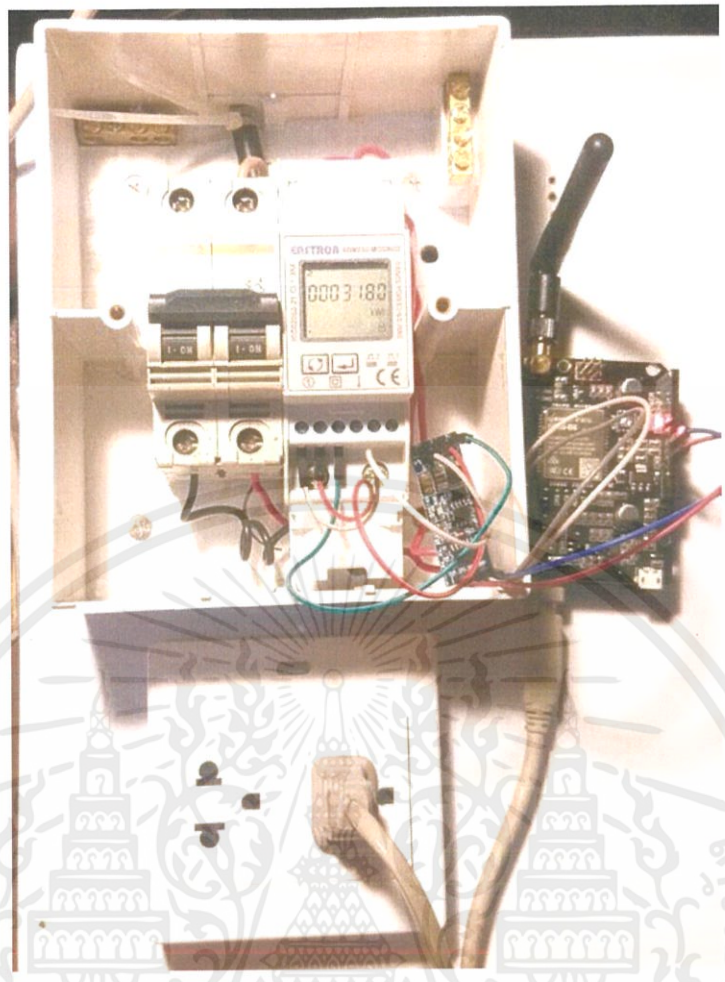
```
  Serial.begin(115200);                    //initialize serial
  sdm.begin();                             //initialize SDM
communication
}
void loop() {
  Serial.print("Voltage:    ");
  Serial.print(sdm.readVal(SDM230_VOLTAGE), 2);  //display
voltage
  Serial.println("V");
  delay(50);
  Serial.print("Current:    ");
  Serial.print(sdm.readVal(SDM230_CURRENT), 2);   //display
current
  Serial.println("A");
  delay(50);
  Serial.print("Frequency: ");
  Serial.print(sdm.readVal(SDM230_FREQUENCY), 2);   //display
frequency
  Serial.println("Hz");
  delay(50);
  Serial.print("Power:      ");
  Serial.print(sdm.readVal(SDM230_POWER), 2);   //display
power
  Serial.println("W");
  delay(50);
  Serial.print("Unit Usage:      ");
  Serial.print(sdm.readVal(SDM230_TOTAL_ACTIVE_ENERGY),
2);   //display power
  Serial.println("kWh");
  delay(50);
  Serial.print("Power Factor:      ");
  Serial.print(sdm.readVal(SDM230_POWER_FACTOR),
2);   //display power
  delay(1000);    //wait a while before next loop
}
```

โดยโปรแกรมนี้จะมีการทำงานหลักอยู่ในฟังก์ชัน void loop() ซึ่งจะมีคำสั่งที่ใช้สำหรับอ่านค่าความต่างศักย์(V) กระแส(A) ความถี่(Hz) กำลังงาน(W) ปริมาณการใช้งาน(kWh) และตัวประกอบกำลัง โดยจะใช้คำสั่ง sdm.readVal(Parameter) โดยค่า Parameter เป็นค่าหน่วยที่ต้องการอ่าน ซึ่งแสดงไว้ดังตารางที่ 3.2

ตารางที่ 3.3 ค่า Parameter ของคำสั่ง sdm.readVal

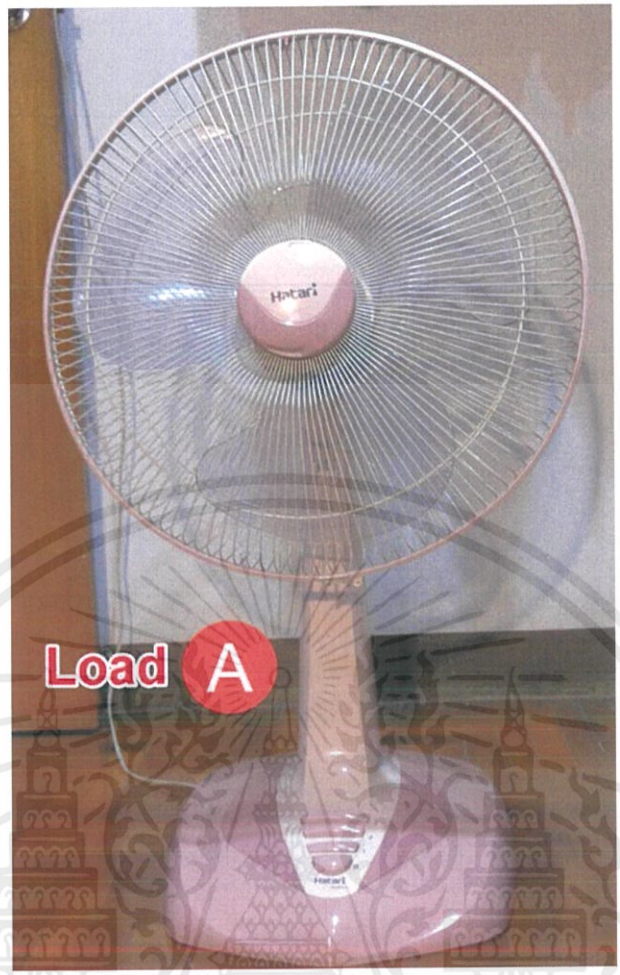
Parameter	ปริมาณ	หน่วย
SDM230_VOLTAGE	Voltage	V
SDM230_CURRENT	Current	A
SDM230_POWER	Power	W
SDM230_ACTIVE_APPARENT_POWER	Active Power	VA
SDM230_REACTIVE_APPARENT_POWER	Reactive Power	VAR
SDM230_POWER_FACTOR	Power Factor	-
SDM230_PHASE_ANGLE	Phase Angle	°
SDM230_FREQUENCY	Frequency	Hz
SDM230_IMPORT_ACTIVE_ENERGY	Import Active Energy	Wh
SDM230_EXPORT_ACTIVE_ENERGY	Export Active Energy	Wh
SDM230_IMPORT_REACTIVE_ENERGY	Import Reactive Energy	VARh
SDM230_EXPORT_REACTIVE_ENERGY	Export Reactive Energy	VARh

Parameter	ปริมาณ	หน่วย
SDM230_TOTAL_SYSTEM_POWER_DEMAND	Total Power Demand	W
SDM230_MAXIMUM_SYSTEM_POWER_DEMAND	Maximum Power Demand	W
SDM230_CURRENT_POSITIVE_POWER_DEMAND	Current Positive Power Demand	W
SDM230_MAXIMUM_POSITIVE_POWER_DEMAND	Maximum Positive Power Demand	W
SDM230_CURRENT_REVERSE_POWER_DEMAND	Current Reverse Power Demand	W
SDM230_MAXIMUM_REVERSE_POWER_DEMAND	Maximum Reverse Power Demand	W
SDM230_CURRENT_DEMAND	Current Demand	A
SDM230_MAXIMUM_CURRENT_DEMAND	Maximum Current Demand	A
SDM230_TOTAL_ACTIVE_ENERGY	Total Active Energy	kWh
SDM230_TOTAL_REACTIVE_ENERGY	Total Reactive Energy	kVARh
SDM230_CURRENT_RESETTABLE_TOTAL_ACTIVE_ENERGY	Current Resettable Total Active Energy	Wh
SDM230_CURRENT_RESETTABLE_TOTAL_REACTIVE_ENERGY	Current Resettable Total Reactive Energy	VARh



รูปที่ 3.30 การเชื่อมต่ออุปกรณ์ขณะทำการทดสอบ

จากรูปที่ 3.30 นั้นเป็นการเชื่อมต่อมาตรวัดพลังงานไฟฟ้าเข้ากับโมดูล RS485 to TTL และ Microcontroller ดังการออกแบบในรูปที่ 3.29 โดยในการทดลองนี้จะนำ **Load A** ซึ่งเป็นพัดลมตั้งโต๊ะมาร่วมใช้ในการทดสอบ ดังรูปที่ 3.31



รูปที่ 3.31 โหลดทดสอบ Load A

3.7.4 การทดลองส่งพารามิเตอร์ทางไฟฟ้าที่วัดได้ขึ้น Cloud Server

ในการทดลองนี้ ผู้จัดทำได้ทำการนำข้อมูลทางไฟฟ้าที่สามารถวัดได้ ประกอบไปด้วยค่าความต่างศักย์(V) กระแส(A) ความถี่(Hz) กำลังงาน(W) ปริมาณการใช้งาน(kWh) และตัวประกอบกำลัง มาเตรียมการส่งไปยัง Cloud Server เพื่อจัดเก็บข้อมูลและนำไปแสดงผลในขั้นตอนถัดไป โดยในขั้นตอนนี้จะเป็นการเขียนโปรแกรมการส่งข้อมูลไปยัง Cloud Server ซึ่งจะทำการจัดข้อมูลให้อยู่ในรูปแบบ JSON Format และทำการส่งผ่านโพรโตคอล CoAP ไปยัง Cloud Server และใช้โปรแกรม WireShark เพื่อตรวจสอบความถูกต้องของข้อมูลที่ทำการส่ง โดยมีการเขียนโปรแกรมดังนี้

```
#include "True_NB_bc95.h"
#include <Wire.h>
#include <SoftwareSerial.h>                    //import
SoftwareSerial library
#include <SDM.h>
#include <AltSoftSerial.h>

AltSoftSerial Serial2;
SoftwareSerial swSerSDM(12, 13);               //config
SoftwareSerial
```

```
SDM sdm(swSerSDM, 4800, NOT_A_PIN);        //config SDM buad
rate

//Variables
int voltage;
int freq;
int power;
String current;
String unit;
String pf;
String bill;
long start = 0;
char sock[] = "0\0";
True_NB_bc95 modem;
String udpRemoteIP = "23.101.19.163";           //Server IP
int udpRemotePort = 5683;                        //Server Port
char iotToken[] = " ngSQG83rFeo9nVsvj0Bx\0";     //Access
token
//Define json format
//Define json format for int
char jsonData1[] = "{\"voltage\":%d, \"freq\":%d,
\"power\":%d}\0";
//Define json format for float
char jsonData2[] = "{\"current\":%s, \"unit\":%s,
\"pf\":%s}\0";
char buff1[48];
char buff2[48];
int jsonData_len1 = 0;
int jsonData_len2 = 0;

CoapPacketTrueIoT coap;

void setup() {                                  //initialize SDM230
and NB-IoT Shield
  Serial.begin(9600);
  Serial2.begin(9600);
  sdm.begin();

  Serial.println("Starting to connect NB-IoT Network...");
  modem.init(Serial2);
  modem.initModem();
  Serial.println( "IMEI = " + modem.getIMEI() );
  Serial.println( "IMSI = " + modem.getIMSI() );
  while (!modem.register_network());
  delay(1000);
  Serial.println( "Device IP = " + modem.check_ipaddr() );
  modem.create_UDP_socket( 4700, sock);
}

void loop() {
  delay(5000);
  //Read Sensor values
  voltage = sdm.readVal(SDM230_VOLTAGE);
  delay(100);
  current = sdm.readVal(SDM230_CURRENT);
```

```
  delay(100);
  freq = sdm.readVal(SDM230_FREQUENCY);
  delay(100);
  power = sdm.readVal(SDM230_POWER);
  delay(100);
  unit = sdm.readVal(SDM230_TOTAL_ACTIVE_ENERGY);
  delay(100);
  pf = sdm.readVal(SDM230_POWER_FACTOR);
  delay(100);

  tb_send(); // Send CoAP Packet
}

void tb_send(){                                    //CoAP Packet Send
Function
  if (millis() - start > 5000) {
    start = millis();
    sprintf(buff1, jsonData1, (int)voltage, (int)freq,
(int)power  );
    jsonData_len1 = strlen(buff1);
    Serial.println(buff1);
    sprintf(buff2, jsonData2, current.c_str(), unit.c_str(),
pf.c_str()  );
    jsonData_len2 = strlen(buff2);
    Serial.println(buff2);
    modem.postRequest( iotToken, buff1, &coap );
    modem.sendUDPPacket2( sock, udpRemoteIP, udpRemotePort,
&coap, jsonData_len1);
    delay(250);
    modem.postRequest( iotToken, buff2, &coap );
    modem.sendUDPPacket2( sock, udpRemoteIP, udpRemotePort,
&coap, jsonData_len2);
  }
}
```

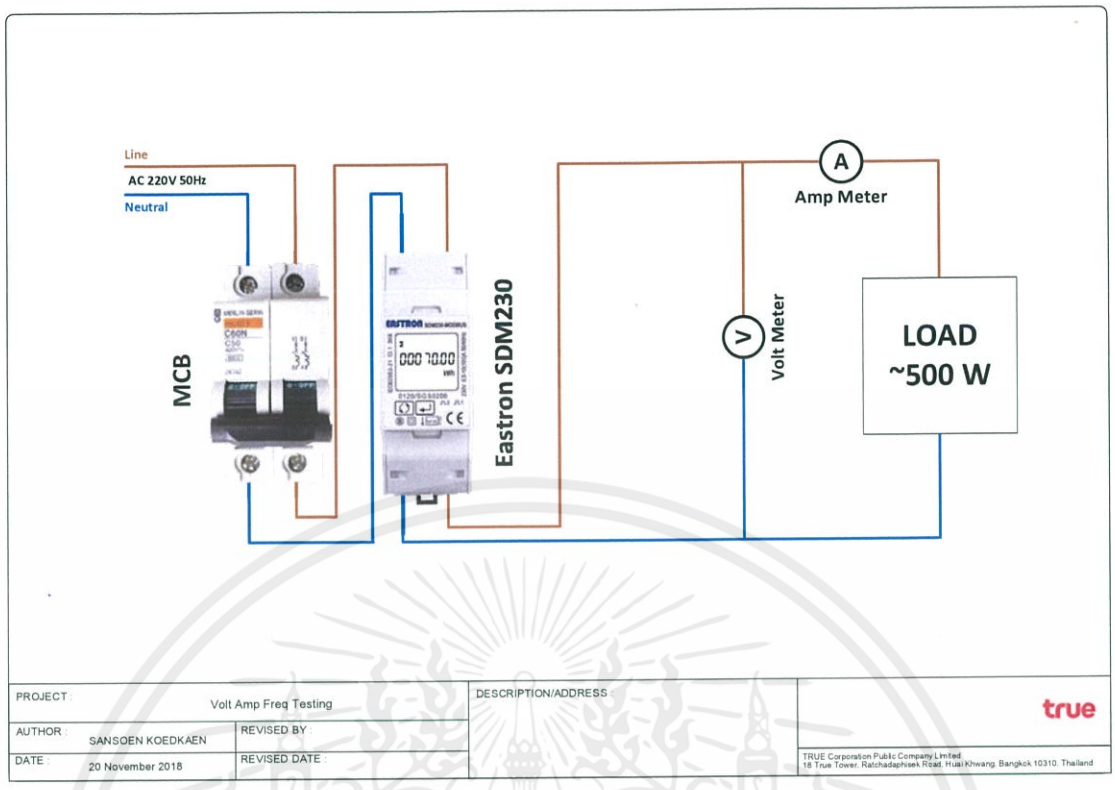
โดยโปรแกรมนี้จะมีส่วนการทำงานหลักของโปรแกรมอยู่ในฟังก์ชัน void loop() และ tb_send() โดยที่ในฟังก์ชัน vode loop() นั้น เป็นการส่งคำสั่งอ่านค่าการวัดทางไฟฟ้าไปยังมาตรวัดพลังงานไฟฟ้า Eastron SDM320 Modbus เพื่อทำการอ่านค่าความต่างศักย์ (V), กระแส (A), ความถี่ (Hz), กำลังงาน (W), ปริมาณการใช้งาน (kWh), และตัวประกอบกำลัง แล้วนำมาจัดเก็บลงตัวแปร voltage, current, freq, power, unit และ pf ตามลำดับ จากนั้นจึงเรียกใช้ฟังก์ชัน tb_send() เพื่อทำการส่งค่าในตัวแปรไปยัง Cloud Server ที่ IP address 23.101.19.163 port 5683 ด้วยโปรโตคอล CoAP

3.7.5 การทดสอบความคลาดเคลื่อนของมาตรวัดพลังงานไฟฟ้า (V, A, Hz)

ผู้จัดทำได้ทำการทดลองวัดความคลาดเคลื่อนของค่าความต่างศักย์ กระแส และความถี่เทียบกับอุปกรณ์วัดมาตรฐาน โดยใช้เครื่องมือวัดเป็น Multi-function Meter ในการวัดความต่างศักย์ และ Clamp Meter ในการวัดกระแสไฟฟ้า ดังรูปที่ 3.32

รูปที่ 3.32 เครื่องวัดมาตรฐาน Multi-function Meter (ซ้าย) และ Clamp Meter (ขวา)

โดยผู้จัดทำได้เตรียมการการทดสอบ โดยการเชื่อมต่อมาตรวัดพลังงานไฟฟ้าเข้ากับเครื่องมือวัดมาตรฐานโดยใช้ Multi-function Meter เป็นโวลต์มิเตอร์ และ Clamp Meter เป็นแอมป์มิเตอร์ ดังแผงภาพในรูปที่ 3.33



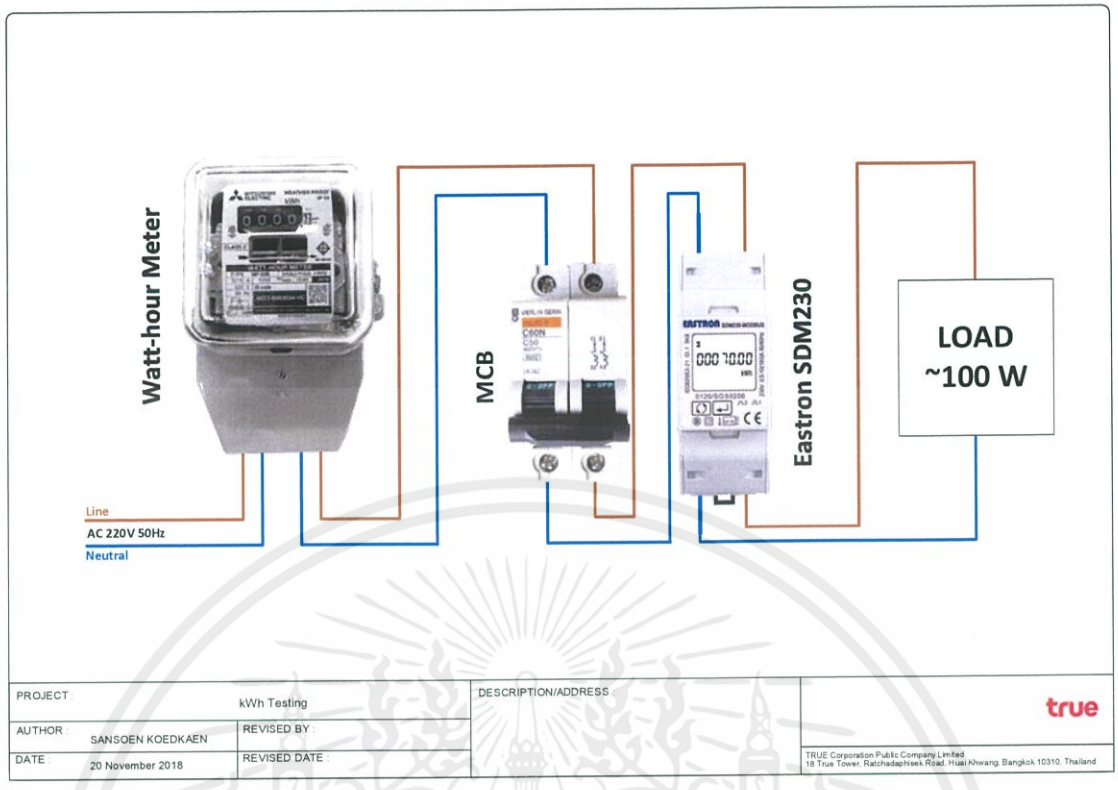
รูปที่ 3.33 การเชื่อมต่ออุปกรณ์เพื่อวัดความคลาดเคลื่อน

เมื่อทำการเชื่อมมาตรวัดพลังงานไฟฟ้าเข้ากับเครื่องมือวัดมาตรฐานเรียบร้อยแล้ว จึงเริ่มทำการวัดค่าโดยการนำโหลดที่ใช้กำลังไฟฟ้าประมาณ 500 วัตต์ (หม้อหุงข้าว) มาเสียบเข้ากับเต้ารับ และทำการเปิดการใช้งานโหลดเป็นเวลาประมาณ 20 นาที โดยขณะนั้นก็ได้ทำการบันทึกค่าความต่างศักย์ กระแส และความถี่ทุกๆ 30 วินาที ดังรูปที่ 3.34

รูปที่ 3.34 ภาพขณะทำการทดสอบมาตรวัดพลังงานไฟฟ้า

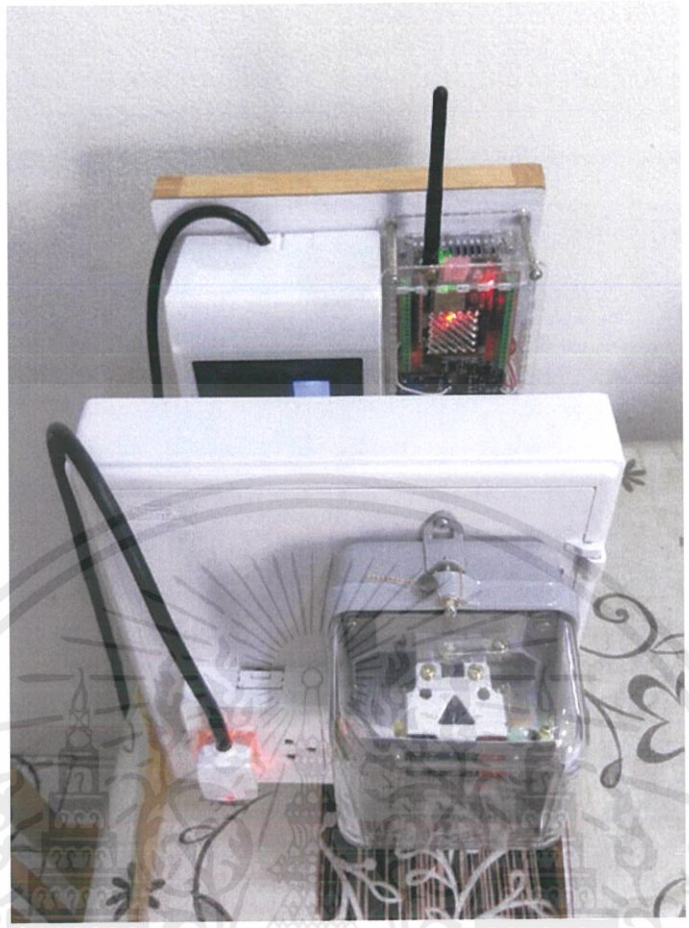
3.7.6 การทดสอบความคลาดเคลื่อนของมาตรวัดพลังงานไฟฟ้า (kWh)

ในการทดสอบความคลาดเคลื่อนครั้งนี้ เป็นการทดสอบความคลาดเคลื่อนปริมาณการใช้งานไฟฟ้าในหน่วย kWh ซึ่งผู้จัดทำได้นำมาตรวัดพลังงานไฟฟ้าแบบทั่วไปที่ใช้ตามบ้านเรือนมาทำการวัดเปรียบเทียบกับมาตรวัดพลังงานไฟฟ้าที่ได้สร้างขึ้น โดยมีการเชื่อมต่อดังแผนภาพที่ 3.35

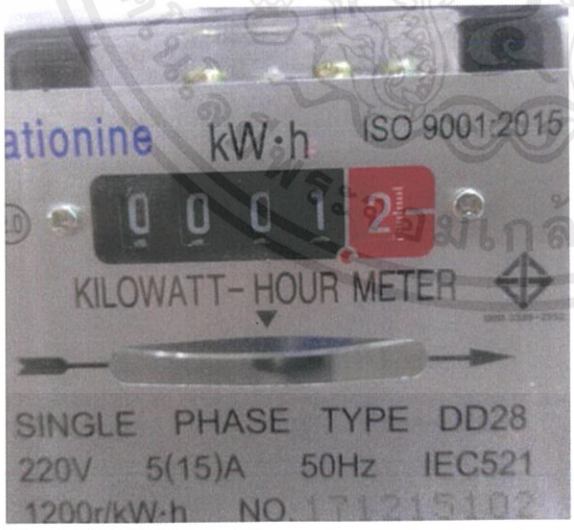


รูปที่ 3.35 การเชื่อมต่ออุปกรณ์เพื่อวัดความคลาดเคลื่อน

โดยผู้จัดทำได้ทำการทดสอบโดยการต่อโหลดทดสอบ (ตู้เย็น Inverter) เข้ากับมาตรวัดพลังงานไฟฟ้าที่ได้จัดทำขึ้น ซึ่งในการทดสอบครั้งนี้จะแตกต่างจากการทดสอบในข้อที่ 3.7.5 เนื่องจากเป็นการทดสอบที่ใช้เวลานานกว่า และมีการบันทึกค่าในช่วงเวลาที่ไม่คงที่ เพื่อตรวจสอบความคลาดเคลื่อนของปริมาณการใช้งานไฟฟ้า โดยใช้เวลาทำการทดสอบทั้งหมดประมาณ 15 วัน โดยการอ่านค่าการใช้งานไฟฟ้าจากตัว Watt-hour meter และมาตรวัดพลังงานไฟฟ้า Eastron SDM230 และทำการบันทึกค่าที่อ่านได้จากเครื่องมือทั้งสอง ดังรูปที่ 3.36 3.37 และ 3.38



รูปที่ 3.36 การเชื่อมต่อระหว่าง Watt-hour meter (หน้า) กับ มาตรวัดพลังงานไฟฟ้า Eastron SDM230 (หลัง)



รูปที่ 3.37 ค่า kWh ของ Watt-hour meter เมื่อเริ่มทำการทดสอบ

รูปที่ 3.38 ค่า kWh ของ Eastron SDM230 เมื่อเริ่มทำการทดสอบ

3.7.7 การทดสอบความถูกต้องของฟังก์ชันประมาณการค่าไฟฟ้า

ผู้จัดทำได้ทำการเขียนฟังก์ชันการทำงานของ Microcontroller สำหรับการประมาณการค่าไฟฟ้าจากปริมาณการใช้ไฟฟ้า โดยอ้างอิงจากอัตราค่าไฟฟ้าประเภทบ้านอยู่อาศัยทั่วไป อัตราปกติ แบบก้าวหน้า จากประกาศของการไฟฟ้านครหลวง ฉบับ เดือนพฤศจิกายน 2558 โดยมีอัตราการคิดค่าไฟฟ้าดังตารางที่ 3.4 และ 3.5

ตารางที่ 3.4 อัตราค่าไฟฟ้าประเภทบ้านอยู่อาศัยอัตราปกติ (ใช้ไม่เกิน 150 หน่วย/เดือน) [14]

ลำดับหน่วย (kWh)	อัตราค่าไฟ (บาท/หน่วย)	ค่าบริการ (บาท/เดือน)
15 หน่วยแรก (หน่วยที่ 1-15)	2.3488	8.19
10 หน่วยต่อไป (หน่วยที่ 16-5)	2.9882	
10 หน่วยต่อไป (หน่วยที่ 26-35)	3.2405	
65 หน่วยต่อไป (หน่วยที่ 36-100)	3.6237	
50 หน่วยต่อไป (หน่วยที่ 101-150)	3.7171	
250 หน่วยต่อไป (หน่วยที่ 151-400)	4.2218	
เกินกว่า 400 หน่วย (หน่วยที่ 401 เป็นต้นไป)	4.4217	

ตารางที่ 3.5 อัตราค่าไฟฟ้าประเภทบ้านอยู่อาศัยอัตราปกติ (ใช้เกิน 150 หน่วย/เดือน) [14]

ลำดับหน่วย (kWh)	อัตราค่าไฟ (บาท/หน่วย)	ค่าบริการ (บาท/เดือน)
150 หน่วยแรก (หน่วยที่ 1-150)	3.2484	38.22
250 หน่วยต่อไป (หน่วยที่ 151-400)	4.2218	
เกินกว่า 400 หน่วย (หน่วยที่ 401 เป็นต้นไป)	4.4217	

จากนั้นจึงทำการเขียนโปรแกรมในส่วนของฟังก์ชันการประมาณการค่าไฟโดยใช้เงื่อนไขตามตารางที่ 3.4 และ 3.5 ได้ดังนี้

```
float calBill(float Unit, float ft_rate, bool
over_150_Unit_per_month) {
  float Service = over_150_Unit_per_month ? 38.22 : 8.19;
  float total = 0;
  if (!over_150_Unit_per_month) {
    float Rate15 = 2.3488;
    float Rate25 = 2.9882;
    float Rate35 = 3.2405;
    float Rate100 = 3.6237;
    float Rate150 = 3.7171;
```

```
    float Rate400 = 4.2218;
    float RateMore400 = 4.4217;

    if (Unit >= 0) total += min(Unit, 15) * Rate15;
    if (Unit >= 16) total += min(Unit - 15, 10) * Rate25;
    if (Unit >= 26) total += min(Unit - 25, 10) * Rate35;
    if (Unit >= 36) total += min(Unit - 35, 65) * Rate100;
    if (Unit >= 101) total += min(Unit - 100, 50) * Rate150;
    if (Unit >= 151) total += min(Unit - 150, 250) * Rate400;
    if (Unit >= 401) total += (Unit - 400) * RateMore400;
  } else {
    float Rate150 = 3.2484;
    float Rate400 = 4.2218;
    float RateMore400 = 4.4217;

    total += min(Unit, 150) * Rate150;
    if (Unit >= 151) total += min(Unit - 150, 150) * Rate400;
    if (Unit >= 401) total += (Unit - 400) * RateMore400;
  }

  total += Service;
  total += Unit * ft_rate;
  total += total * 7 / 100;

  return total;
}
```

จากโปรแกรมในด้านบนจะมีการรับค่าตัวแปร Unit คือปริมาณการใช้งานไฟฟ้าในหน่วย kWh ตัวแปร ft_rate คือค่า Ft หรือค่า Float time และตัวแปร over_150_Unit_per_month คือค่า flag สำหรับการเลือกอัตรการใช้งานว่าจะเป็นการคำนวณแบบเกิน 150 หน่วยต่อเดือนหรือไม่

โดยในส่วนแรกของฟังก์ชัน จะมีการกำหนดค่าบริการรายเดือนจากตัวแปร over_150_Unit_per_month ซึ่งถูกส่งเข้ามาในฟังก์ชัน

จากนั้นฟังก์ชันจะทำงานโดยการกำหนดเงื่อนไขแบ่งการกำหนดอัตราการคิดค่าไฟฟ้าเป็น 2 เงื่อนไข โดยเงื่อนไขแรก จะเป็นในกรณีที่ปริมาณการใช้งานไฟฟ้ามีมากกว่า 150 หน่วยต่อเดือน และเงื่อนไขที่สอง เป็นกรณีที่ปริมาณการใช้งานไฟฟ้า 150 หน่วยต่อเดือน

จากนั้นปริมาณการใช้งานไฟฟ้าจะถูกแบ่งเป็นช่วง ตามอัตราค่าไฟฟ้าแต่ละเงื่อไข และถูกคูณด้วยจำนวนหน่วยในขั้นๆ และถูกรวมกันเป็นค่าไฟฟ้าดิบ

เมื่อได้ค่าไฟฟ้าดิบแล้ว ก็จะถูกเก็บไว้ในตัวแปร Service ซึ่งจะถูกนำมาบวกเพิ่มกับค่าบริการรายเดือน ค่า Ft และภาษีมูลค่าเพิ่ม 7% จึงได้เป็นค่าไฟฟ้าสุทธิที่ประมาณการได้จากปริมาณการใช้ไฟฟ้า

จากนั้นจึงทำการอัพโหลดโปรแกรมลง Microcontroller โดยผู้จัดทำจะทำการทดสอบโดยการกำหนดปริมาณการใช้งานเป็น 102.32 หน่วย และ 173.23 หน่วยตามลำดับ แล้วจึงนำค่าไฟฟ้าที่

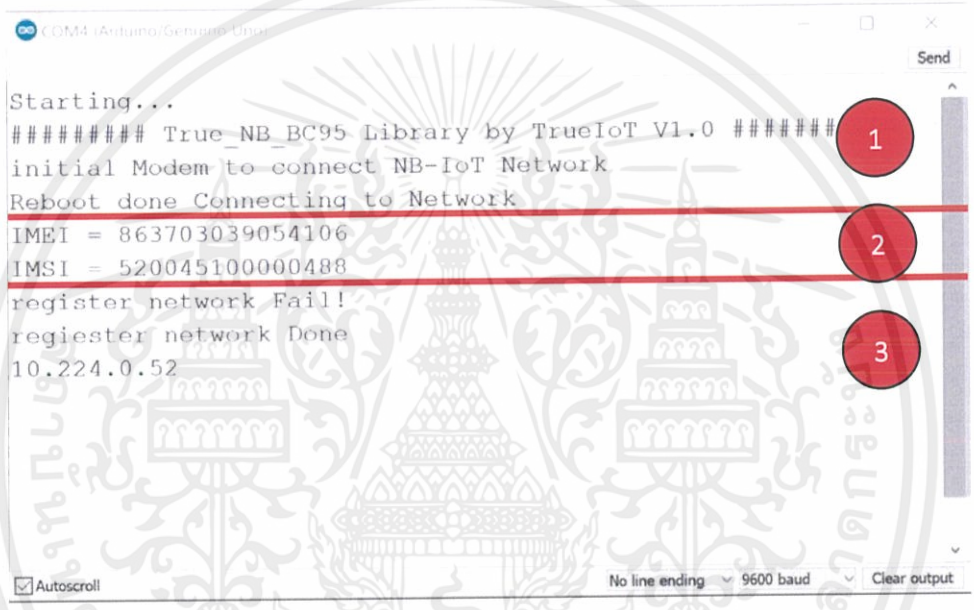
ประมาณการได้ไปเทียบกับเว็บไซต์คำนวณค่าไฟฟ้าของการไฟฟ้านครหลวง โดยทำการบันทึกผลจาก Serial Monitor ของโปรแกรม Arduino IDE

บทที่ 4

ผลการทดลอง

4.1 ผลการทดลองการทำงานของ NB-IoT Shield

ในการทดลองนี้จะเป็นการทดลองเขียนคำสั่งเพื่อให้ Microcontroller สั่งงาน NB-IoT Shield ผ่านการสื่อสารแบบอนุกรมให้เริ่มต้นการทำงาน และทำการเชื่อมต่อเข้ากับเครือข่าย NB-IoT ของผู้ให้บริการ

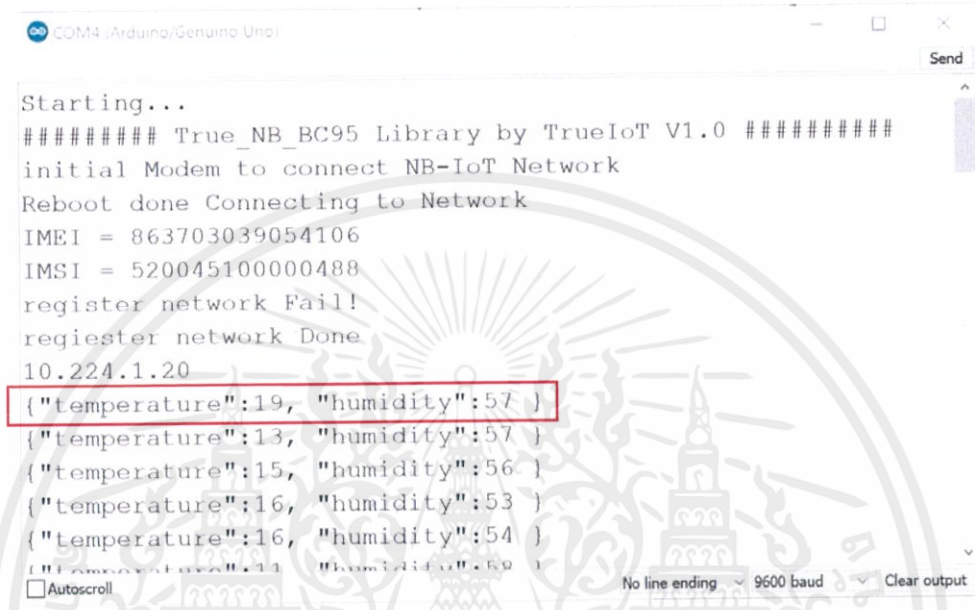


รูปที่ 4.1 ผลการทำงานของโปรแกรมในข้อ 3.7.1 จาก Serial Monitor

จากรูปที่ 4.1 เป็นผลลัพธ์การทำงานของโปรแกรมในข้อ 3.7.1 โดยในส่วนที่ 1 เป็นการเริ่มต้นการทำงานของ NB-IoT Shield ส่วนที่ 2 เป็นการแสดงผลหมายเลข IMEI และ IMSI ของ NB-IoT Shield และส่วนที่ 3 เป็นการเชื่อมต่อเข้ากับเครือข่าย NB-IoT ของผู้ให้บริการ ซึ่งจะพบว่าอุปกรณ์ NB-IoT ที่ใช้ทำการทดสอบนั้นมี Private IP เป็น 10.224.0.52

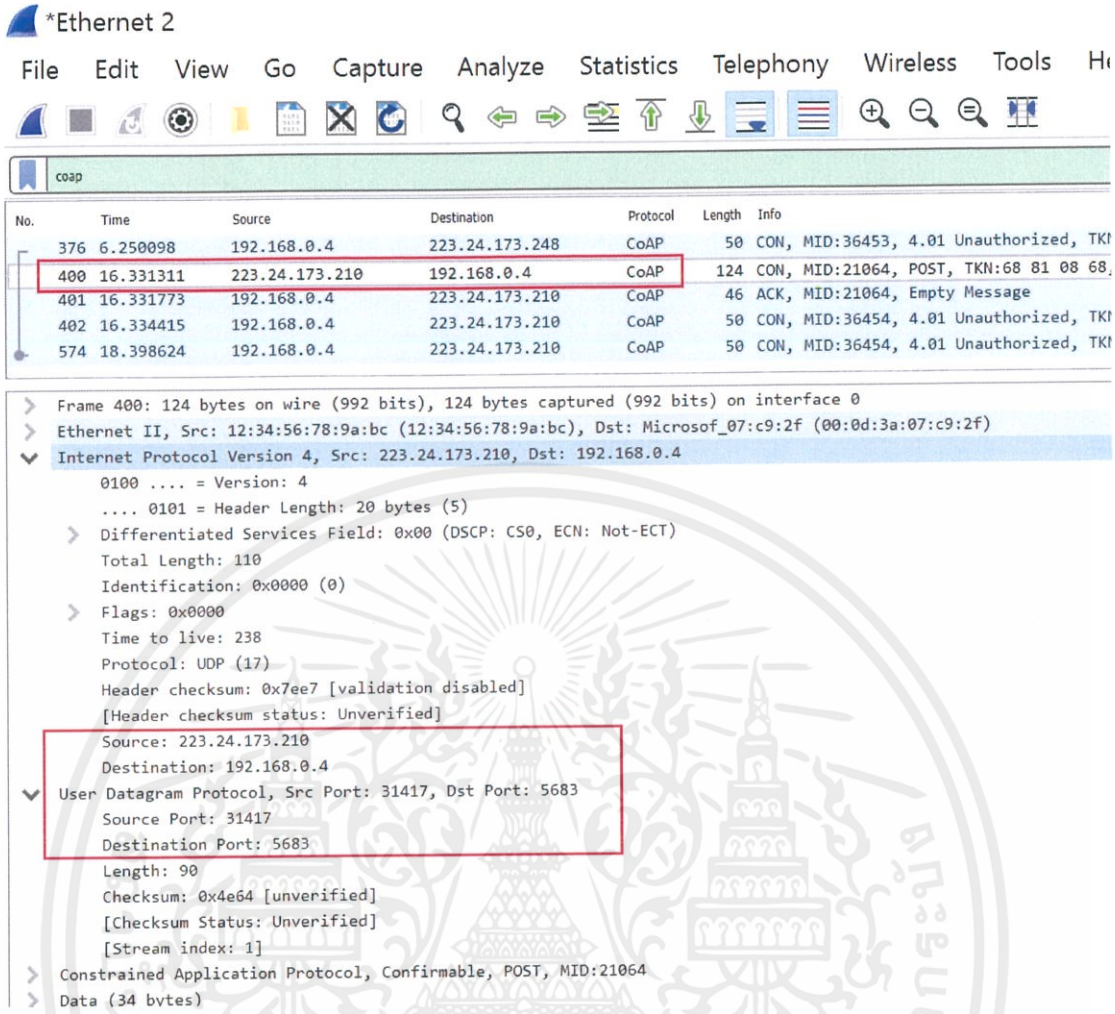
4.2 ผลการทดลองส่งข้อมูลไปยัง IoT-Cloud Server

ในการทดลองนี้จะเป็นการทดสอบส่งข้อมูลจาก NB-IoT Shield ไปยัง Cloud Server ผ่านโพรโตคอล CoAP โดยได้มีการจำลองชุดข้อมูลที่อยู่ในรูปแบบ JSON Data Format ขึ้นมาสำหรับการทดลองส่งข้อมูล



รูปที่ 4.2 ผลการทำงานของโปรแกรมในข้อ 3.7.2 จาก Serial Monitor

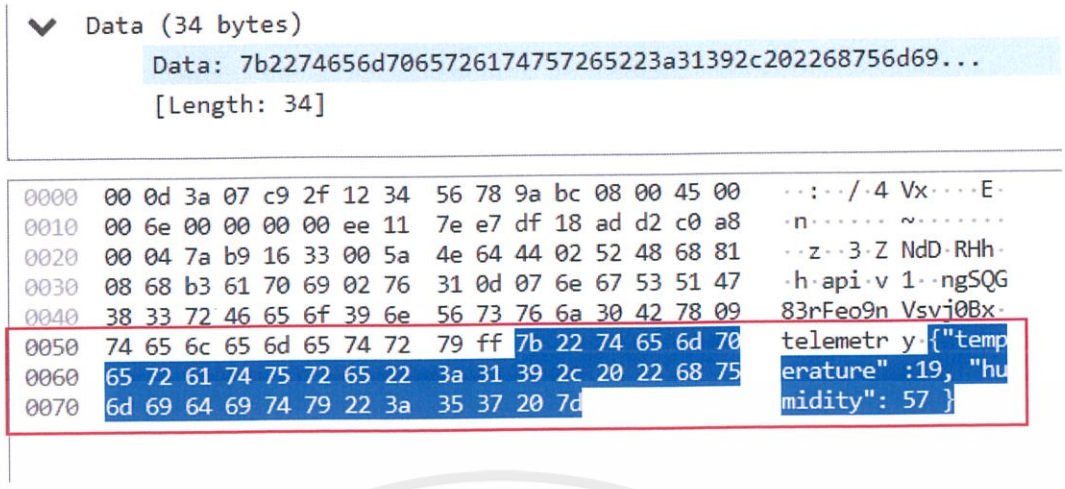
จากรูปที่ 4.2 จะเป็นการทำงานของโปรแกรมในข้อ 3.7.2 ซึ่งจะเป็นการสุ่มข้อมูลอุณหภูมิและความชื้นสัมพัทธ์แล้วทำการแปลงข้อมูลให้อยู่ในรูปแบบ JSON Format จากนั้นจึงทำการส่งข้อมูลไปยัง Cloud Server ที่ IP address 23.101.19.163 โดยใช้ Port 5683 ซึ่งเป็น Port ของ CoAP Service โดยข้อมูลที่ทำการส่งไปยัง Cloud Server คือ {"temperature":19, "humidity":57 } หลังจากนั้นจึงใช้โปรแกรม WireShark เพื่อตรวจจับ Datagram ที่ Cloud Server สามารถรับได้



รูปที่ 4.3 การทดสอบตรวจจับ Datagram บน Cloud Server ด้วยโปรแกรม WireShark

จากรูปที่ 4.3 จะเห็นได้ว่ามี Datagram ถูกส่งเข้ามายัง Cloud Server ดังแพ็คเกจลำดับที่ 400 ซึ่งมีต้นทางมาจาก NB-IoT Shield ที่มี Public IP address 223.24.173.210 ผ่าน port 31417 และมีปลายทางที่ Cloud Server ที่มี Internal IP address 192.168.0.4 ผ่าน port 5683 ซึ่งเป็นการส่งผ่านโพรโตคอล CoAP

จากนั้นจึงทำการ Decode ข้อมูลที่ได้รับมาด้วยโปรแกรม WireShark เพื่อทำการตรวจสอบความถูกต้องข้อมูลที่ได้ทำการส่ง

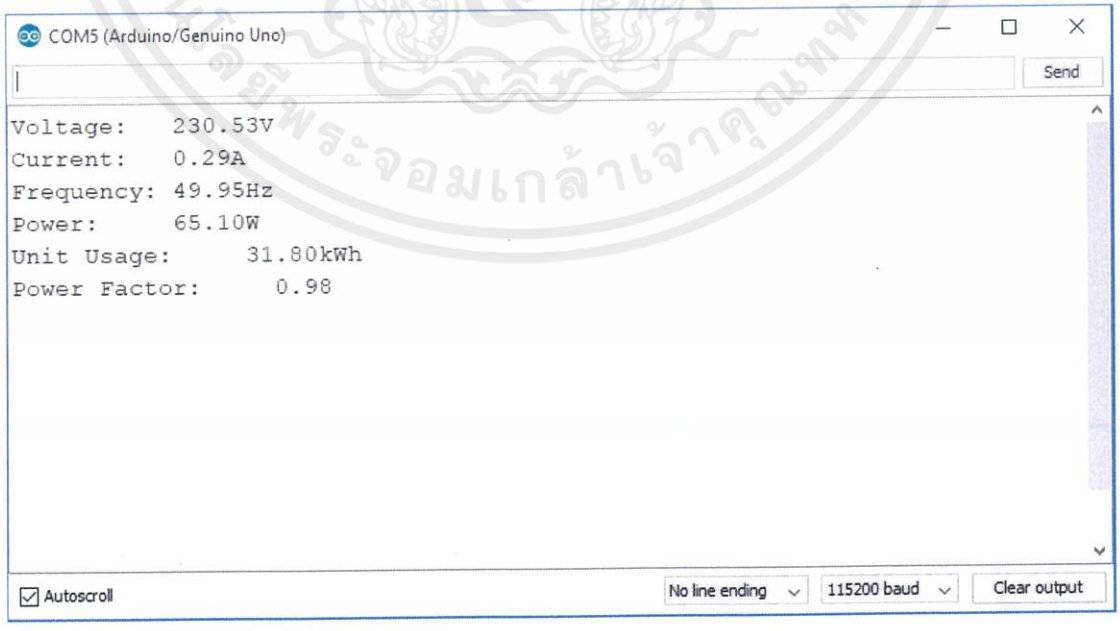


รูปที่ 4.4 ข้อมูลที่รับได้หลังจากผ่านการ Decode ด้วยโปรแกรม WireShark

จากรูปที่ 4.4 จะเห็นได้ว่า ข้อมูลที่ได้รับหลังจากผ่านการ Decode เรียบร้อยแล้วคือ {"temperature":19, "humidity":57 } ซึ่งตรงกับผลที่ได้จาก Serial Monitor ของฝั่งส่ง (NB-IoT Shield) จากรูปที่ 4.2

4.3 ผลการทดลองเชื่อมต่อ Microcontroller กับมาตรวัดพลังงานไฟฟ้า

ในการทดลองนี้จะเป็นการนำมาตรวัดพลังงานไฟฟ้า Eastron SDM230 Modbus มาเชื่อมต่อเข้ากับ Microcontroller เพื่อทำการอ่านค่าความต่างศักย์(V) กระแส(A) ความถี่(Hz) กำลังงาน(W) ปริมาณการใช้งาน(kWh) และตัวประกอบกำลัง

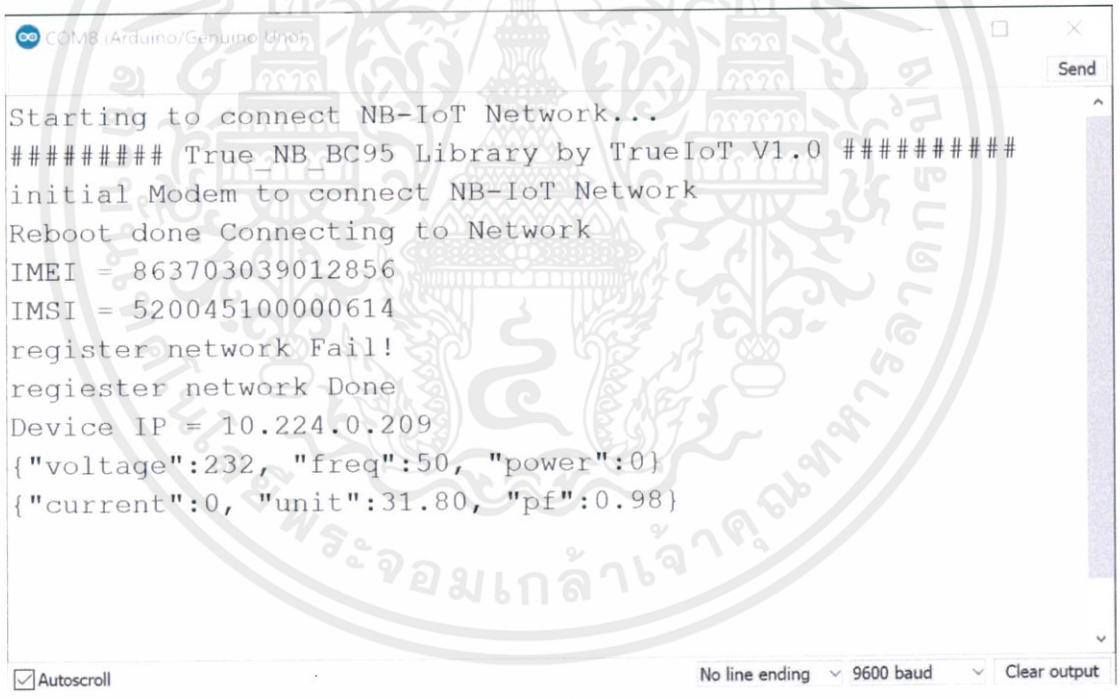


รูปที่ 4.5 ผลการทำงานของโปรแกรมในข้อ 3.7.3 จาก Serial Monitor

จากรูปที่ 4.5 เป็นผลการทำงานของโปรแกรมในข้อ 3.7.3 ซึ่งเป็นการดึงค่าความต่างศักย์ (V) กระแส (A) ความถี่ (Hz) กำลังงาน (W) ปริมาณการใช้งาน (kWh) และตัวประกอบกำลัง ที่มาตรวัดพลังงานไฟฟ้า Eastron SDM 230 Modbus วัดได้ออกมาแสดงผล โดยเมื่อทำการนำค่าที่อ่านได้จาก Serial Monitor นั้นมาเปรียบเทียบกับค่าที่อ่านได้จากจอแสดงผลของมาตรวัด พบว่ามีความถูกต้องทั้งหมด

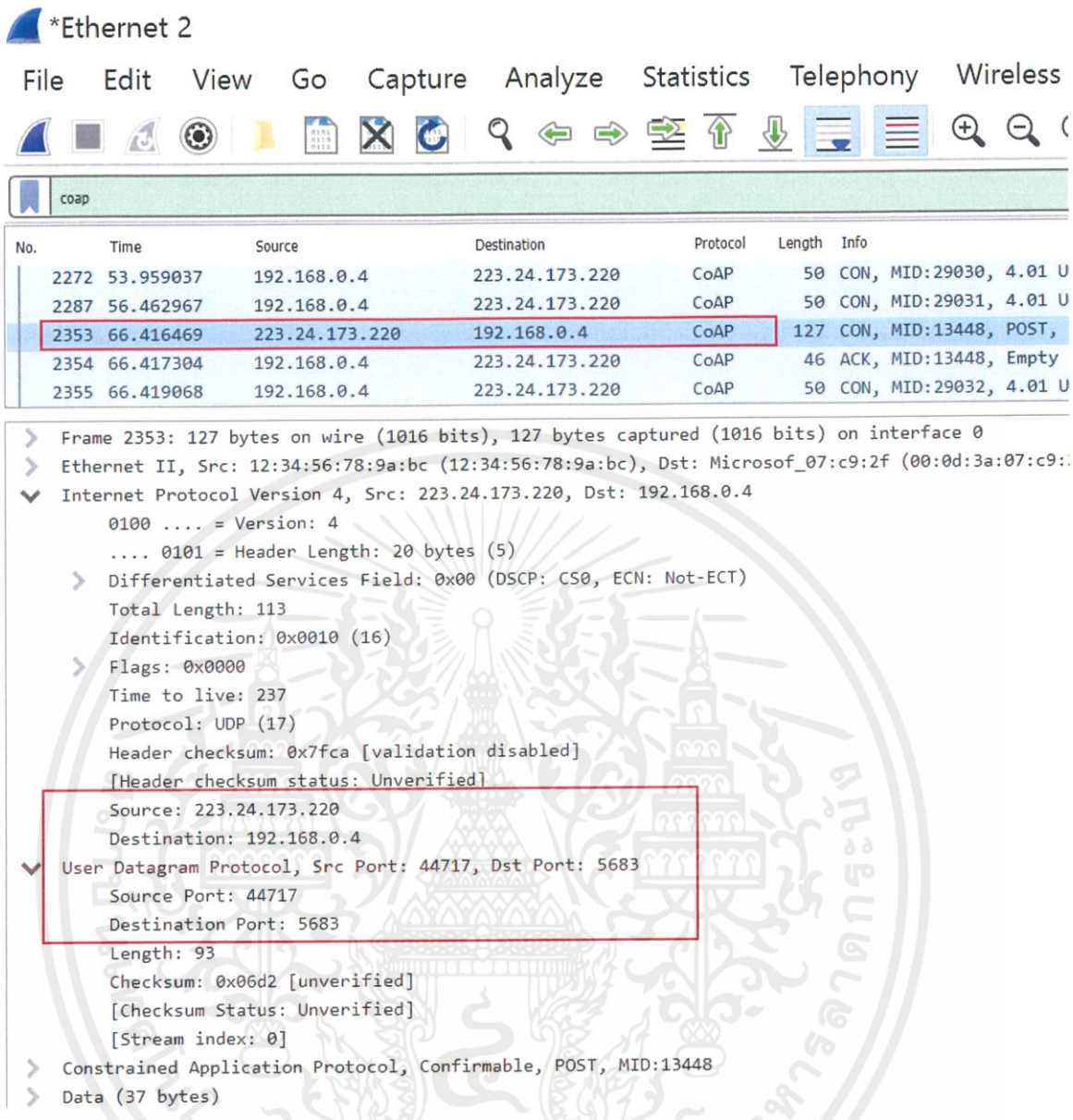
4.4 ผลการทดลองส่งพารามิเตอร์ทางไฟฟ้าที่วัดได้ขึ้น Cloud Server

ในการทดลองนี้ ผู้จัดทำได้ทำการนำข้อมูลทางไฟฟ้าที่สามารถวัดได้ ประกอบไปด้วยค่าความต่างศักย์(V) กระแส(A) ความถี่(Hz) กำลังงาน(W) ปริมาณการใช้งาน(kWh) และตัวประกอบกำลัง มาเตรียมการส่งไปยัง Cloud Server เพื่อจัดเก็บข้อมูลและนำไปแสดงผลในขั้นตอนถัดไป โดยในขั้นตอนนี้จะเป็นการเขียนโปรแกรมการส่งข้อมูลไปยัง Cloud Server ซึ่งจะทำการจัดข้อมูลให้อยู่ในรูปแบบ JSON Format และทำการส่งผ่านโพรโตคอล CoAP ไปยัง Cloud Server และใช้โปรแกรม WireShark เพื่อตรวจสอบความถูกต้องของข้อมูลที่ทำการส่ง

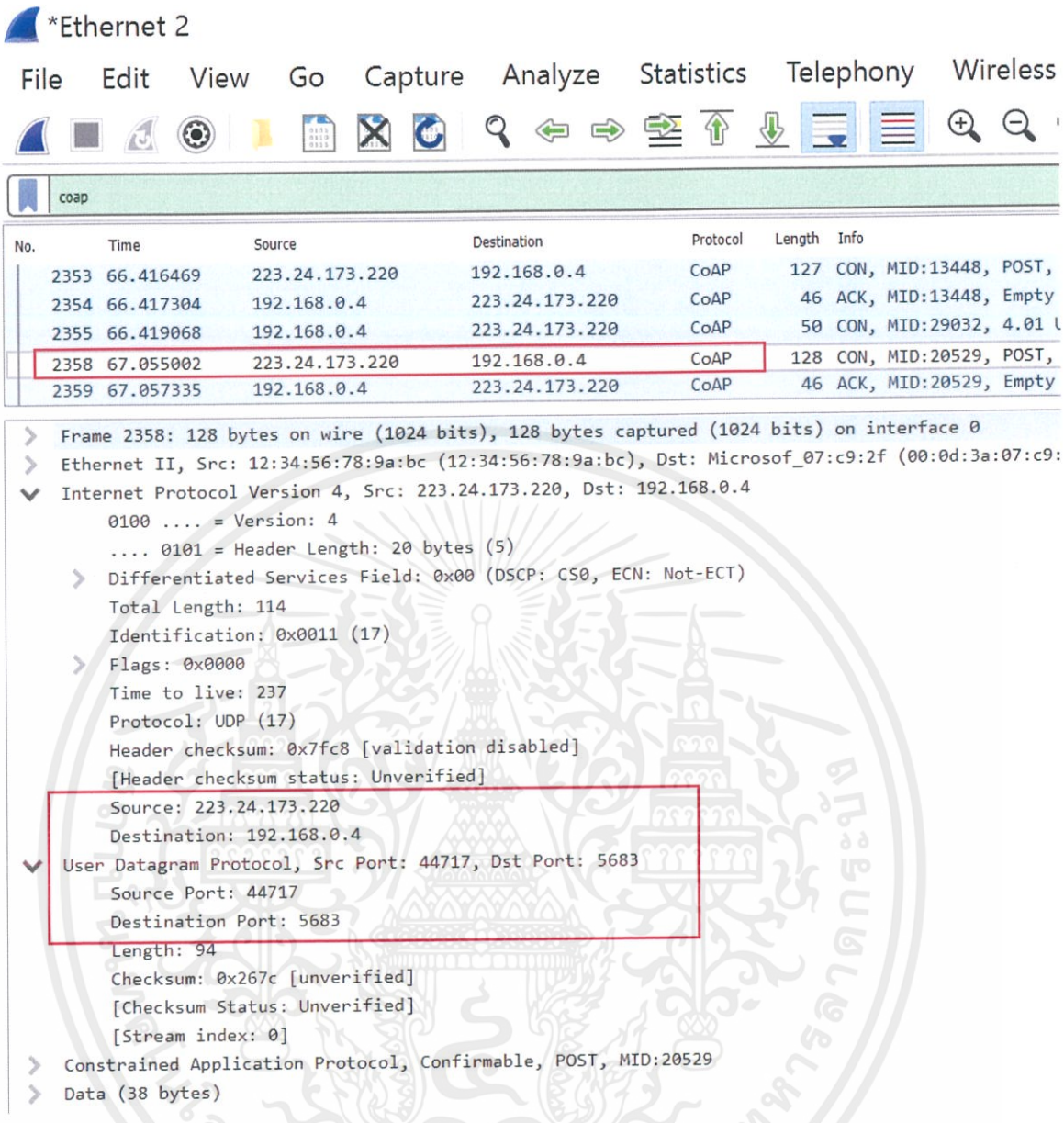


รูปที่ 4.6 ผลการทำงานของโปรแกรมในข้อ 3.7.4 จาก Serial Monitor

จากรูปที่ 4.6 จะเห็นได้ว่าข้อมูลที่ทำการจัดให้อยู่ในรูปแบบ JSON Format นั้นแบ่งเป็น 2 ชุดข้อมูล ได้แก่ {“voltage”:232, “ferq”:50, “power”:0} และ {“current”:0, “unit”:31.8, “pf”:0.98} จากนั้นจึงใช้โปรแกรม WireShark ทำการตรวจจับ Datagram ที่รับได้ในฝั่ง Cloud Server



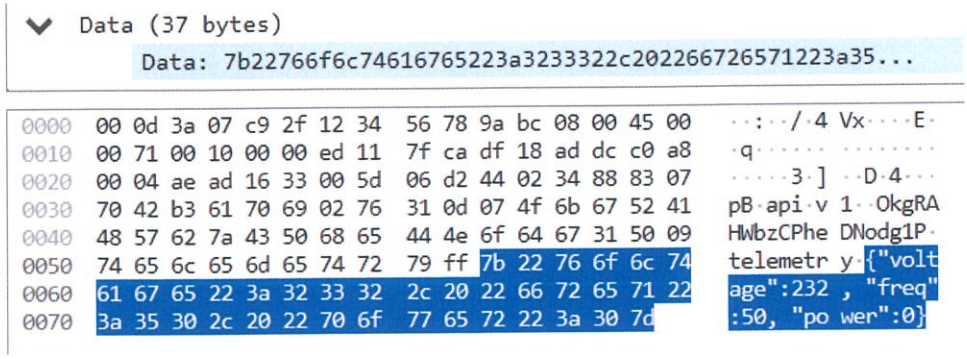
รูปที่ 4.7 การทดสอบตรวจจับ Datagram ชุดที่ 1 บน Cloud Server ด้วยโปรแกรม WireShark



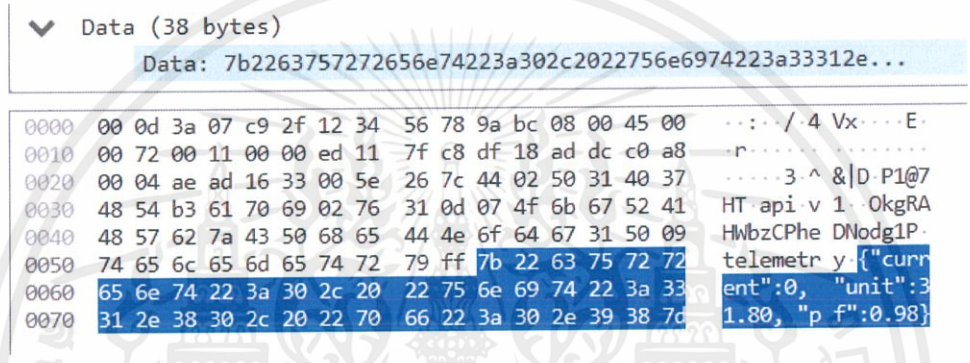
รูปที่ 4.8 การทดสอบตรวจจับ Datagram ชุดที่ 2 บน Cloud Server ด้วยโปรแกรม WireShark

จากรูปที่ 4.7 และ 4.8 จะเห็นได้ว่ามี Datagram ถูกส่งเข้ามายัง Cloud Server ดังแพ็คเกจลำดับที่ 2353 และ 2358 ซึ่งมีต้นทางมาจาก NB-IoT Shield ที่มี Public IP address 223.24.173.220 ผ่าน port 44717 และมีปลายทางที่ Cloud Server ที่มี Internal IP address 192.168.0.4 ผ่าน port 5683 ซึ่งเป็นการส่งผ่านโพรโตคอล CoAP

จากนั้นจึงทำการ Decode ข้อมูลที่ได้รับมาด้วยโปรแกรม WireShark เพื่อทำการตรวจสอบความถูกต้องข้อมูลที่ได้ทำการส่ง



รูปที่ 4.9 ข้อมูลชุดที่ 1 ที่รับได้หลังจากผ่านการ Decode ด้วยโปรแกรม WireShark



รูปที่ 4.10 ข้อมูลชุดที่ 2 ที่รับได้หลังจากผ่านการ Decode ด้วยโปรแกรม WireShark

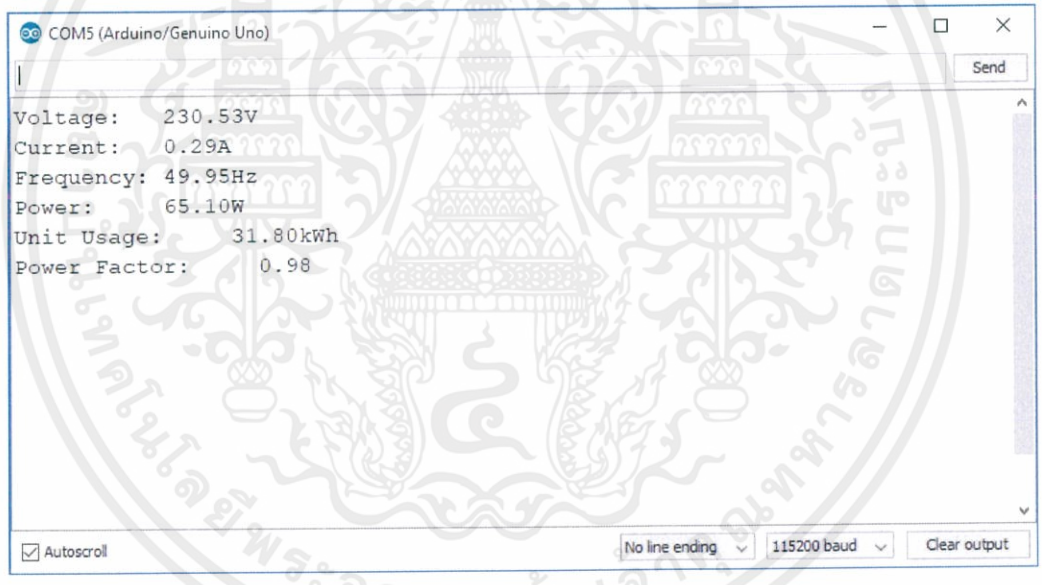
จากรูปที่ 4.9 และ 4.10 จะเห็นได้ว่า ข้อมูลที่ได้รับหลังจากผ่านการ Decode เรียบร้อยแล้ว คือ {“voltage”:232, “ferq”:50, “power”:0} และ {“current”:0, “unit”:31.8, “pf”:0.98} ซึ่งตรงกับผลที่ได้จาก Serial Monitor ของฝั่งส่ง (NB-IoT Shield) จากรูปที่ 4.6

4.5 ผลการทดสอบความคลาดเคลื่อนของมาตรวัดพลังงานไฟฟ้า (V, A, Hz)

ผู้จัดทำได้ทำการทดลองวัดความคลาดเคลื่อนของค่าความต่างศักย์ กระแส และความถี่เทียบกับอุปกรณ์วัดมาตรฐาน โดยใช้เครื่องมือวัดเป็น Multi-function Meter ในการวัดความต่างศักย์ และ Clamp Meter ในการวัดกระแสไฟฟ้า

โดยผู้จัดทำได้เตรียมการการทดสอบ โดยการเชื่อมต่อมาตรวัดพลังงานไฟฟ้าเข้ากับเครื่องมือวัดมาตรฐานโดยใช้ Multi-function Meter เป็นโวลต์มิเตอร์ และ Clamp Meter เป็นแอมป์มิเตอร์

เมื่อทำการเชื่อมมาตรวัดพลังงานไฟฟ้าเข้ากับเครื่องมือวัดมาตรฐานเรียบร้อยแล้ว จึงเริ่มทำการวัดค่าโดยการนำโหลดที่ใช้กำลังไฟฟ้าประมาณ 500 วัตต์ (หม้อหุงข้าว) มาเสียบเข้ากับเต้ารับ และทำการเปิดการใช้งานโหลดเป็นเวลาประมาณ 20 นาที โดยขณะนั้นก็ได้ทำการบันทึกค่าความต่างศักย์ กระแส และความถี่ทุก ๆ 30 วินาที โดยขณะที่ทำการทดสอบ มาตรวัดพลังงานที่ได้จัดทำขึ้นก็จะมีการบันทึกค่าและส่งค่าออกมายังคอมพิวเตอร์ และแสดงผลผ่าน Serial Monitor บน Arduino IDE ดังรูปที่ 4.11



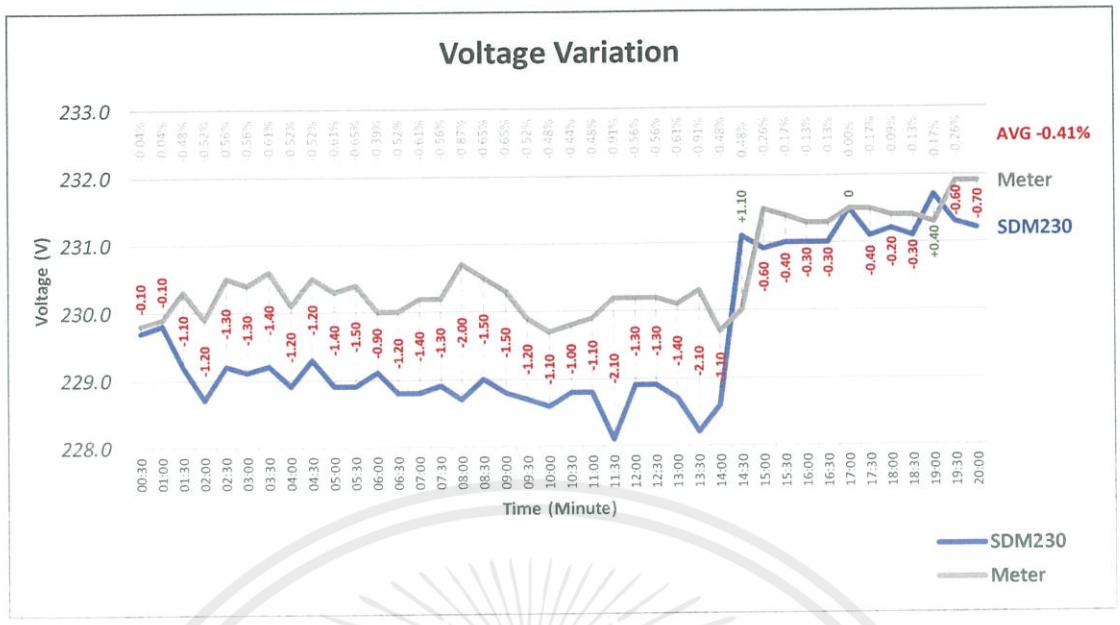
รูปที่ 4.11 ผลลัพธ์ที่ได้ขณะทำการทดสอบมาตรวัดพลังงานไฟฟ้า

เมื่อสิ้นสุดการทดสอบ จึงนำค่าความต่างศักย์ กระแส และความถี่ ที่บันทึกได้จากการอ่านเครื่องมือวัดมาตรฐาน และที่ได้จากมาตรวัดพลังงานไฟฟ้าที่ได้ทำการสร้างขึ้น มาบันทึกลงในตารางแสดงดังตารางที่ 4.1

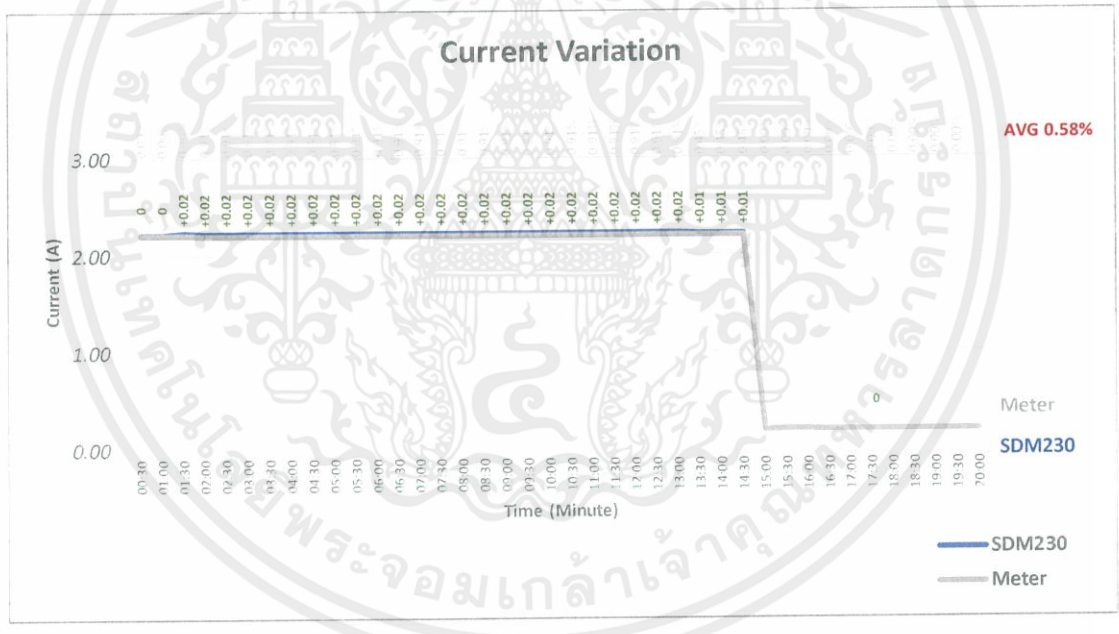
ตารางที่ 4.1 ค่าความต่างศักย์ กระแส และความถี่ ที่จดบันทึกไว้ขณะทำการทดสอบ

Sampling (30 s. Interval)	Voltage		Error (%)	Current		Error (%)	Frequency		Error (%)
	Meter	SDM230		Meter	SDM230		Meter	SDM230	
1	229.8	229.7	-0.04	2.21	2.21	0.00	49.93	49.95	0.04
2	229.9	229.8	-0.04	2.21	2.21	0.00	49.95	49.94	-0.02
3	230.3	229.2	-0.48	2.21	2.23	0.90	49.94	49.95	0.02
4	229.9	228.7	-0.52	2.20	2.22	0.91	49.94	50.00	0.12
5	230.5	229.2	-0.56	2.20	2.22	0.91	49.99	50.00	0.02
6	230.4	229.1	-0.56	2.20	2.22	0.91	49.94	49.95	0.02
7	230.6	229.2	-0.61	2.20	2.22	0.91	49.92	49.90	-0.04
8	230.1	228.9	-0.52	2.20	2.22	0.91	49.92	50.00	0.16
9	230.5	229.3	-0.52	2.20	2.22	0.91	49.96	50.00	0.08
10	230.3	228.9	-0.61	2.20	2.22	0.91	49.94	49.95	0.02
11	230.4	228.9	-0.65	2.20	2.22	0.91	49.92	50.00	0.16
12	230.0	229.1	-0.39	2.20	2.22	0.91	49.96	50.05	0.18
13	230.0	228.8	-0.52	2.20	2.22	0.91	50.00	50.05	0.10
14	230.2	228.8	-0.61	2.20	2.22	0.91	50.01	50.10	0.18
15	230.2	228.9	-0.56	2.20	2.22	0.91	50.03	50.00	-0.06
16	230.7	228.7	-0.87	2.20	2.22	0.91	49.98	50.05	0.14
17	230.5	229.0	-0.65	2.20	2.22	0.91	49.99	50.05	0.12
18	230.3	228.8	-0.65	2.20	2.22	0.91	50.03	49.95	-0.16
19	229.9	228.7	-0.52	2.20	2.22	0.91	49.96	49.95	-0.02
20	229.7	228.6	-0.48	2.20	2.22	0.91	49.92	49.95	0.06
21	229.8	228.8	-0.44	2.20	2.22	0.91	49.94	49.95	0.02
22	229.9	228.8	-0.48	2.20	2.22	0.91	49.93	49.95	0.04
23	230.2	228.1	-0.91	2.20	2.22	0.91	49.91	49.95	0.08
24	230.2	228.9	-0.56	2.20	2.22	0.91	49.95	50.00	0.10
25	230.2	228.9	-0.56	2.20	2.22	0.91	50.00	50.00	0.00
26	230.1	228.7	-0.61	2.20	2.22	0.91	49.98	50.00	0.04
27	230.3	228.2	-0.91	2.20	2.21	0.45	49.93	49.95	0.04
28	229.7	228.6	-0.48	2.20	2.21	0.45	49.93	49.95	0.04
29	230.0	231.1	0.48	2.20	2.21	0.45	49.93	49.95	0.04
30	231.5	230.9	-0.26	0.19	0.19	0.00	49.98	50.00	0.04
31	231.4	231.0	-0.17	0.20	0.20	0.00	49.98	49.95	-0.06
32	231.3	231.0	-0.13	0.20	0.20	0.00	49.92	49.95	0.06
33	231.3	231.0	-0.13	0.19	0.19	0.00	49.93	49.95	0.04
34	231.5	231.5	0.00	0.19	0.19	0.00	49.95	50.00	0.10
35	231.5	231.1	-0.17	0.19	0.19	0.00	49.94	49.95	0.02
36	231.4	231.2	-0.09	0.19	0.19	0.00	49.94	50.00	0.12
37	231.4	231.1	-0.13	0.19	0.19	0.00	49.95	49.95	0.00
38	231.3	231.7	0.17	0.19	0.19	0.00	49.96	49.95	-0.02
39	231.9	231.3	-0.26	0.19	0.19	0.00	49.95	50.00	0.10
40	231.9	231.2	-0.30	0.19	0.19	0.00	50.03	50.05	0.04

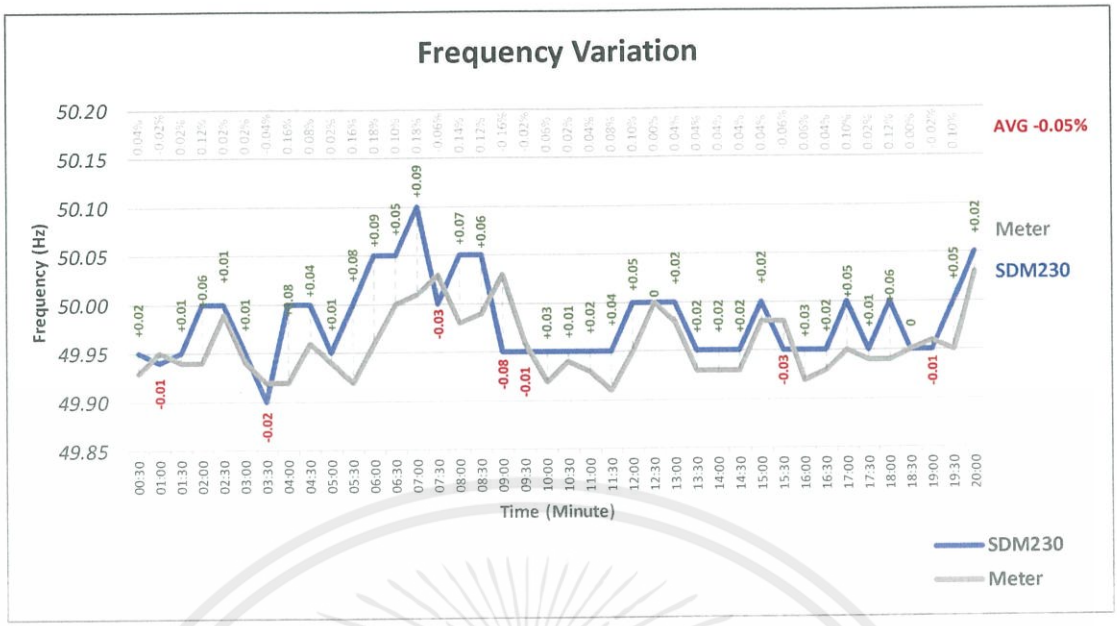
จากค่าที่บันทึกได้ในตารางที่ 4.1 สามารถนำมาพล็อทกราฟเพื่อแสดงความคลาดเคลื่อนได้ดังรูปที่ 4.12 4.13 และ 4.14



รูปที่ 4.12 กราฟแสดงความคลาดเคลื่อนของความต่างศักย์



รูปที่ 4.13 กราฟแสดงความคลาดเคลื่อนของกระแส



รูปที่ 4.14 กราฟแสดงความคลาดเคลื่อนของความถี่

จากรูปที่ 4.12 เป็นการแสดงผลค่าความต่างศักย์ที่วัดได้ตลอดการทดสอบ พบว่าค่าเฉลี่ยของความผิดพลาด เมื่อเทียบระหว่างเครื่องมือวัดมาตรฐาน กับมาตรวัดพลังงานไฟฟ้าที่ได้สร้างขึ้น มีค่าเฉลี่ยอยู่ที่ -0.41%

จากรูปที่ 4.13 เป็นการแสดงผลค่ากระแสที่วัดได้ตลอดการทดสอบ พบว่าค่าเฉลี่ยของความผิดพลาด เมื่อเทียบระหว่างเครื่องมือวัดมาตรฐาน กับมาตรวัดพลังงานไฟฟ้าที่ได้สร้างขึ้น มีค่าเฉลี่ยอยู่ที่ 0.58%

จากรูปที่ 4.14 เป็นการแสดงผลค่าความถี่ที่วัดได้ตลอดการทดสอบ พบว่าค่าเฉลี่ยของความผิดพลาด เมื่อเทียบระหว่างเครื่องมือวัดมาตรฐาน กับมาตรวัดพลังงานไฟฟ้าที่ได้สร้างขึ้น มีค่าเฉลี่ยอยู่ที่ 0.05%

จากผลการทดสอบค่าความต่างศักย์ กระแส และความถี่ พบว่า ค่าความผิดพลาดที่เกิดขึ้นเป็นที่ยอมรับได้ของหน่วยงานของข้าพเจ้า และนอกจากนี้ ค่าความผิดพลาดที่เกิดขึ้น อาจจะเกิดขึ้นจากการเตรียมการทดสอบ เช่น การวางตำแหน่งของ Clamp Meter หรือ ความสกปรกของข้อต่อของเครื่องมือวัดมาตรฐาน เป็นต้น

4.6 ผลการทดสอบความคลาดเคลื่อนของมาตรวัดพลังงานไฟฟ้า (kWh)

ในการทดสอบความคลาดเคลื่อนครั้งนี้ เป็นการทดสอบความคลาดเคลื่อนปริมาณการใช้งานไฟฟ้าในหน่วย kWh ซึ่งผู้จัดทำได้นำมาตรวัดพลังงานไฟฟ้าแบบทั่วไปที่ใช้ตามบ้านเรือนมาทำการวัดเปรียบเทียบกับมาตรวัดพลังงานไฟฟ้าที่ได้สร้างขึ้น

โดยผู้จัดทำได้ทำการทดสอบโดยการต่อโหลดทดสอบ (ตู้เย็น Inverter) เข้ากับมาตรวัดพลังงานไฟฟ้าที่ได้จัดทำขึ้น ซึ่งในการทดสอบครั้งนี้จะแตกต่างจากการทดสอบในข้อที่ 4.5 เนื่องจากเป็นการทดสอบที่ใช้เวลานานกว่า และมีการบันทึกค่าในช่วงเวลาที่ไม่คงที่ เพื่อตรวจสอบความคลาดเคลื่อนของปริมาณการใช้งานไฟฟ้า โดยใช้เวลาทำการทดสอบทั้งหมดประมาณ 15 วัน โดยการอ่านค่าการใช้งานไฟฟ้าจากตัว Watt-hour meter และมาตรวัดพลังงานไฟฟ้า Eastron SDM230 และทำการบันทึกค่าที่อ่านได้จากเครื่องมือทั้งสอง

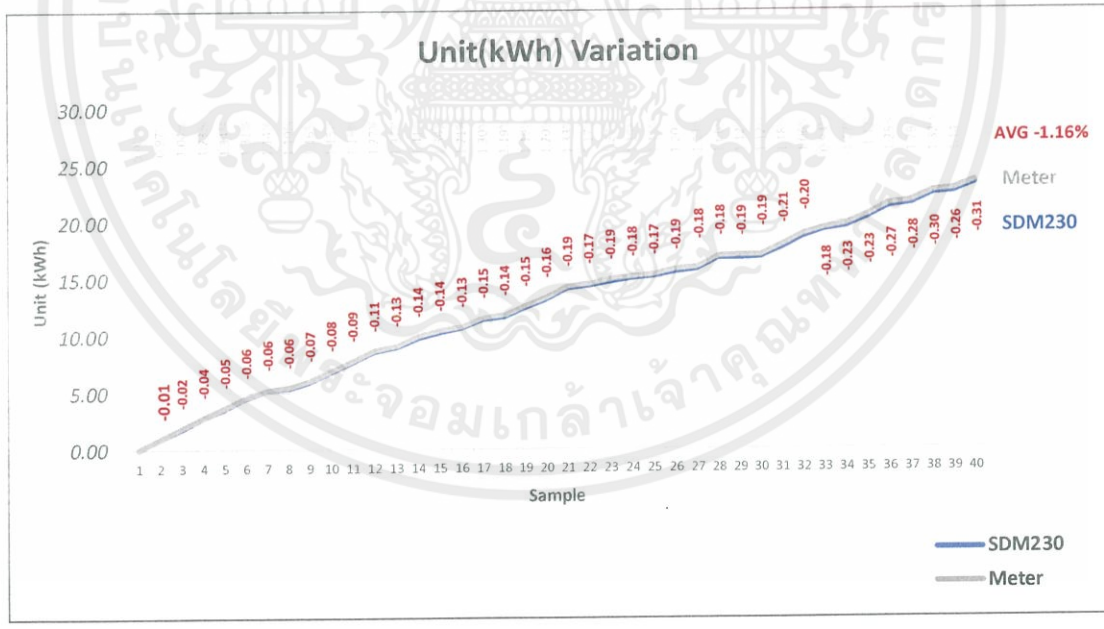
เมื่อสิ้นสุดการทดสอบ จึงนำปริมาณการใช้งานไฟฟ้า ที่บันทึกได้จากการอ่านเครื่องมือวัดมาตรฐาน (Watt-hour Meter) และที่ได้จากมาตรวัดพลังงานไฟฟ้าที่ได้ทำการสร้างขึ้น มาบันทึกลงในตาราง แสดงดังตารางที่ 4.2

ตารางที่ 4.2 ปริมาณการใช้งานไฟฟ้า ที่จดบันทึกไว้ขณะทำการทดสอบ

Sampling	Unit Usage (kWh)		
	Watt-hour Meter	Eastron SDM230	Error (%)
1	0.00	0.00	0%
2	1.03	1.02	-0.97%
3	1.92	1.90	-1.04%
4	2.84	2.80	-1.23%
5	3.72	3.67	-1.34%
6	4.57	4.51	-1.31%
7	5.27	5.21	-1.14%
8	5.46	5.40	-1.10%
9	6.05	5.98	-1.16%
10	6.96	6.88	-1.15%
11	7.80	7.71	-1.15%
12	8.68	8.57	-1.27%
13	9.11	8.98	-1.37%
14	9.93	9.79	-1.41%
15	10.46	10.32	-1.34%
16	10.78	10.65	-1.21%
17	11.56	11.41	-1.30%
18	11.78	11.64	-1.19%
19	12.57	12.42	-1.19%
20	13.36	13.20	-1.20%
21	14.32	14.13	-1.33%
22	14.54	14.37	-1.17%
23	14.92	14.73	-1.27%
24	15.17	14.99	-1.19%

Sampling	Unit Usage (kWh)		
	Watt-hour Meter	Eastron SDM230	Error (%)
25	15.34	15.17	-1.11%
26	15.78	15.59	-1.20%
27	15.98	15.80	-1.13%
28	16.95	16.77	-1.06%
29	16.98	16.79	-1.12%
30	17.02	16.83	-1.12%
31	17.86	17.65	-1.18%
32	18.84	18.64	-1.06%
33	19.45	19.27	-0.93%
34	19.74	19.51	-1.17%
35	20.54	20.31	-1.12%
36	21.52	21.25	-1.25%
37	21.78	21.50	-1.29%
38	22.65	22.35	-1.32%
39	22.81	22.55	-1.14%
40	23.54	23.23	-1.32%

จากค่าที่บันทึกได้ในตารางที่ 4.2 สามารถนำมาพล็อทกราฟเพื่อแสดงความคลาดเคลื่อนได้ดังรูปที่ 4.15



รูปที่ 4.15 กราฟแสดงความคลาดเคลื่อนของปริมาณการใช้งานไฟฟ้า

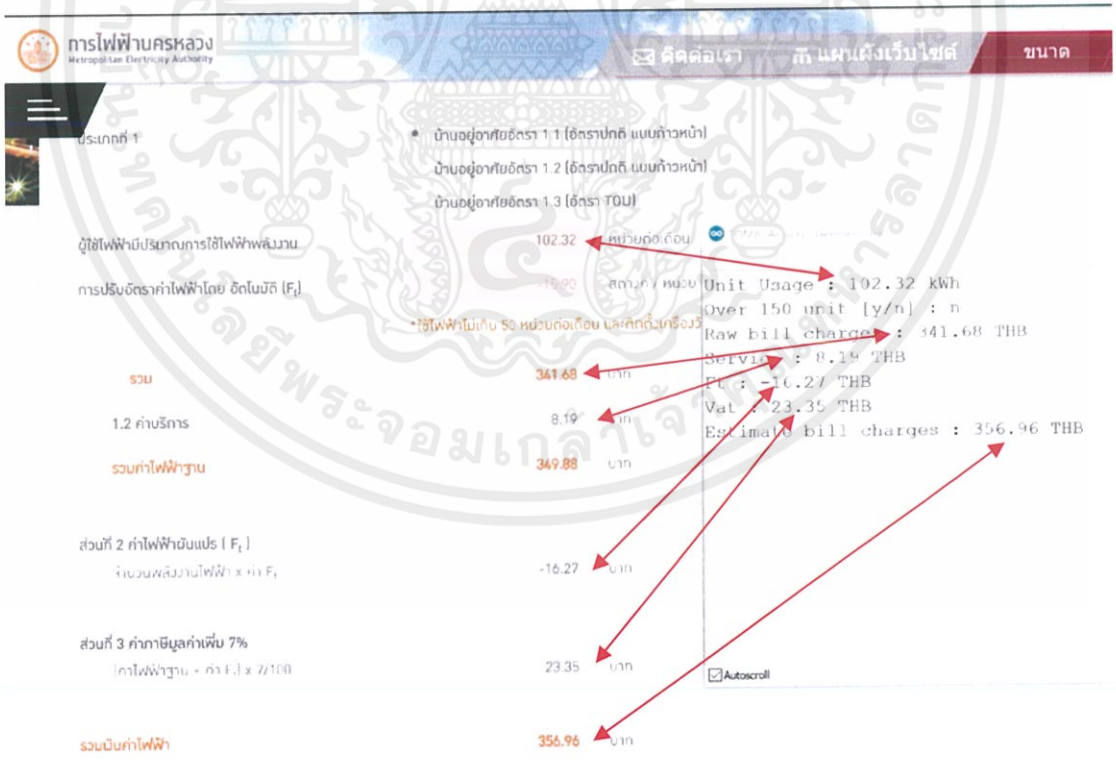
จากรูปที่ 4.15 เป็นการแสดงผลค่าความถี่ที่วัดได้ตลอดการทดสอบ พบว่าค่าเฉลี่ยของความผิดพลาด เมื่อเทียบระหว่างเครื่องมือวัดมาตรฐาน กับมาตรวัดพลังงานไฟฟ้าที่ได้สร้างขึ้น มีค่าเฉลี่ยอยู่ที่ -1.16%

จากผลการทดสอบปริมาณการใช้งานไฟฟ้า พบว่า ค่าความผิดพลาดที่เกิดขึ้น เป็นที่ยอมรับได้ของหน่วยงานของข้าพเจ้า และนอกจากนี้ ค่าความผิดพลาดที่เกิดขึ้น อาจจะเกิดขึ้นจากการเตรียมการทดสอบ เช่น การบริโภคพลังงานของเครื่องมือวัดทั้งสอง เป็นต้น

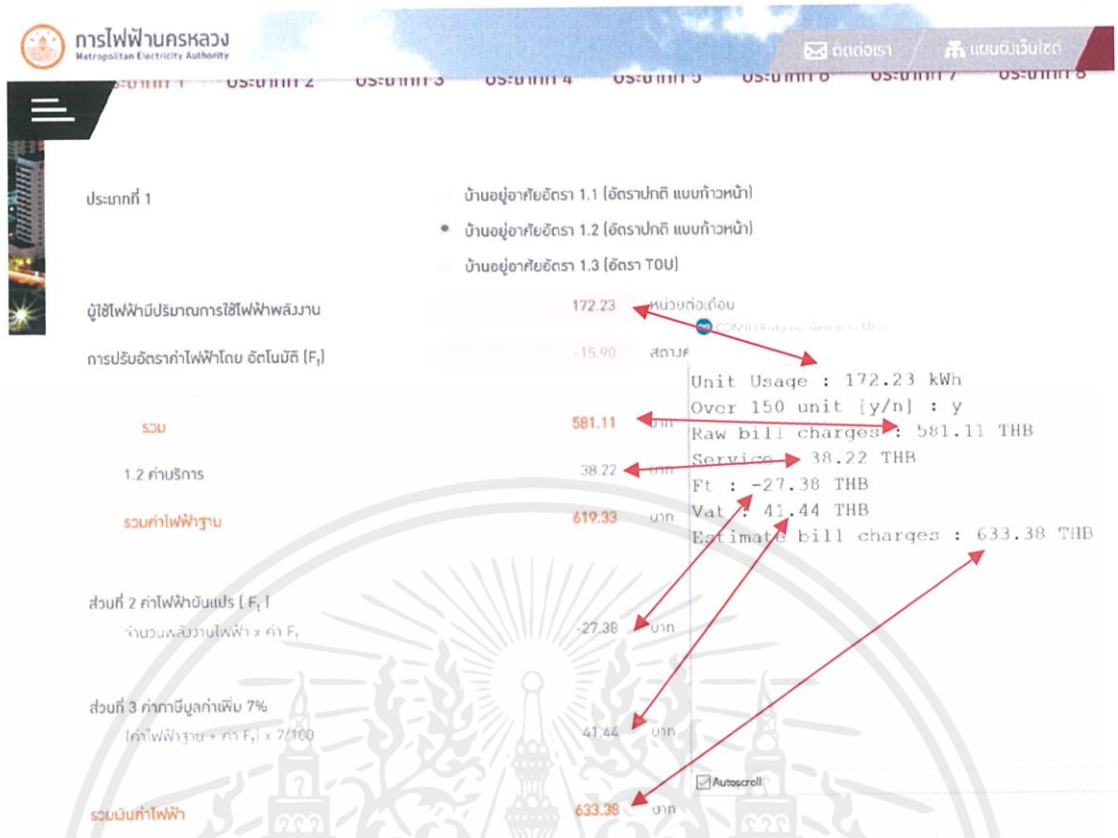
4.7 ผลการทดสอบความถูกต้องของฟังก์ชันประมาณการค่าไฟฟ้า

ผู้จัดทำได้ทำการทดสอบฟังก์ชันการทำงานของ Microcontroller สำหรับการประมาณการค่าไฟฟ้าจากปริมาณการใช้ไฟฟ้า โดยอ้างอิงจากอัตราค่าไฟฟ้าประเภทบ้านอยู่อาศัยทั่วไป อัตราปกติ แบบก้าวหน้า จากประกาศของการไฟฟ้านครหลวง ฉบับ เดือนพฤศจิกายน 2558

โดยผู้จัดทำจะทำการทดสอบโดยการกำหนดปริมาณการใช้งานเป็น 102.32 หน่วย และ 173.23 หน่วยตามลำดับ แล้วจึงนำค่าไฟฟ้าที่ประมาณการได้ไปเทียบกับเว็บไซต์คำนวณค่าไฟฟ้าของการไฟฟ้านครหลวง โดยทำการบันทึกผลจาก Serial Monitor ของโปรแกรม Arduino IDE ดังรูปที่ 4.16 และ 4.17



รูปที่ 4.16 การเปรียบเทียบค่าไฟฟ้าที่ประมาณการได้โดยมีประมาณการใช้งาน 102.32 kWh



รูปที่ 4.17 การเปรียบเทียบค่าไฟฟ้าที่ประมาณการได้โดยมีประมาณการใช้งาน 172.23 kWh

จากรูปที่ 4.16 และ 4.17 จะแสดงผลเปรียบเทียบได้ดังตารางที่ 4.3 ซึ่งจะเห็นได้ว่า ค่าไฟฟ้าที่ได้ประมาณการจากฟังก์ชันคำนวณค่าไฟฟ้า นั้นมีความถูกต้องตามระบบคำนวณค่าไฟฟ้าของการไฟฟ้านครหลวง ซึ่งอ้างอิงจากอัตราค่าไฟฟ้าประเภทบ้านอยู่อาศัยทั่วไป อัตราปกติแบบก้าวหน้า จากประกาศของการไฟฟ้านครหลวง ฉบับ เดือนพฤศจิกายน 2558

ตารางที่ 4.3 ผลการเปรียบเทียบการคำนวณค่าไฟฟ้า

ปริมาณการใช้ (kWh)	102.32		172.23	
จำนวนเงิน (บาท)	จาก กฟน.	คำนวณ	จาก กฟน.	คำนวณ
ค่าไฟฟ้าดิบ	341.68	341.68	581.11	581.11
ค่าบริการ	8.19	8.19	38.22	38.22
ค่าไฟฟ้าผันแปร (Ft)	-16.27	-16.27	-27.38	-27.38
ภาษีมูลค่าเพิ่ม	23.35	23.35	41.44	41.44
รวมค่าไฟฟ้า	356.96	356.96	633.38	633.38

บทที่ 5

สรุปผลและข้อเสนอแนะ

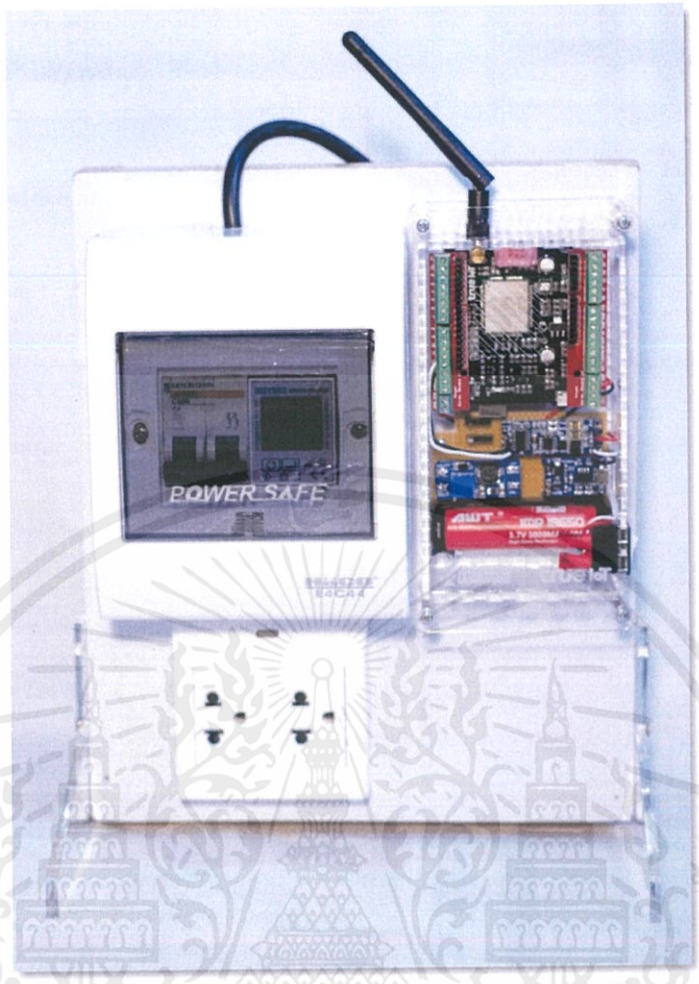
5.1 สรุปผลการดำเนินงาน

จากปัญหาที่หน่วยงานได้ประสบพบเจอในตอนแรกนั้น ทำให้องค์กรต้องเสียค่าใช้จ่ายในการส่งวิศวกรและช่างเทคนิคออกไปยังสถานีฐานที่ห่างไกลออกไป เพื่อตรวจสอบปริมาณการใช้งานไฟฟ้าและค่าพารามิเตอร์ทางไฟฟ้าอื่น ๆ ซึ่งปัญหาเหล่านี้ได้รับการแก้ไขด้วยการนำมาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบไปติดตั้ง ทำให้สามารถตรวจสอบปริมาณการใช้ไฟฟ้าและค่าพารามิเตอร์ทางไฟฟ้าต่าง ๆ ได้ทันที ทุกที่ ทุกเวลา ผ่านทางระบบ Dashboard โดยไม่ต้องส่งวิศวกรและช่างเทคนิคออกไปตรวจสอบ นอกจากนี้ เพื่อป้องกันความเสียหายของอุปกรณ์ในสถานีฐานและความปลอดภัยของพนักงานที่เข้าไปปฏิบัติหน้าที่ ระบบยังสามารถแจ้งเตือนความผิดปกติในระบบไฟฟ้าของไซต์งานได้หลากหลายรูปแบบและทันท่วงทีก่อนจะเกิดความเสียหาย

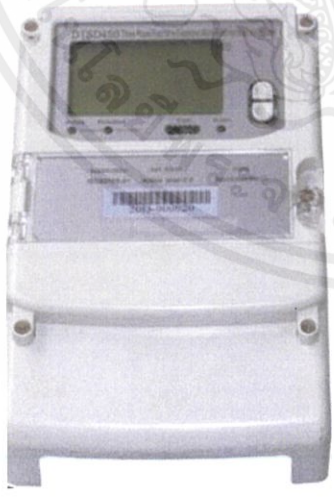
โดยสามารถนำระบบที่ได้สร้างขึ้นมาเปรียบเทียบกับผลิตภัณฑ์อื่น ๆ ในท้องตลาดที่มีวัตถุประสงค์ในการใช้งาน และมีการใช้เทคโนโลยีที่ใกล้เคียงกันได้ดังตารางที่ 5.1

ตารางที่ 5.1 ระบบที่สร้างขึ้นเทียบกับอุปกรณ์อื่นในท้องตลาด

	SDM230	Model A	Model B
เทคโนโลยี	NB-IoT	GPRS	NB-IoT
มาตรฐาน	IEC 62053-21	IEC62053-21	IEC 62053-23
ประเภทการใช้งาน	ภายใน	ภายใน	ภายใน
ความแม่นยำ	+-5%	+-5%	+-2%
ฟังก์ชันการวัด	Volt,Amp,Wh, kVAR,PF,Freq, kVA,Etc.	Wh only	Wh/Date Hour
การแจ้งเตือนผ่าน Line / E-mail	Yes	No	No
ระบบ Cloud Server และ Dashboard	Yes&can Customize	No	No
การระบุตำแหน่ง	Yes(in pipeline)	No	No
ราคา	83 USD	220 USD	n/a
การจำหน่ายอย่างเป็นทางการ	In soon	Yes	No



รูปที่ 5.1 ระบบมาตรวัดสังเกตการณ์พลังงานไฟฟ้าที่ได้สร้างขึ้น (SDM230)



รูปที่ 5.2 ระบบมาตรวัดสังเกตการณ์พลังงานไฟฟ้า (Model A)

รูปที่ 5.3 ระบบมาตรวัดสังเกตการณ์พลังงานไฟฟ้า (Model B)

5.2 ประโยชน์ของโครงงาน

1) ช่วยลดค่าใช้จ่ายในการเดินทางไปตรวจสอบสถานีฐาน
2) ช่วยให้สามารถสังเกตการณ์ความผิดปกติของระบบไฟฟ้าของสถานีฐานได้
3) ช่วยป้องกันความเสียหายของอุปกรณ์สถานีฐานที่เกิดขึ้นจากความผิดปกติของระบบไฟฟ้า
4) ช่วยลดค่าใช้จ่ายในการจัดซื้อระบบมาตรวัดสังเกตการณ์พลังงานไฟฟ้า

5.3 ปัญหาและอุปสรรค

การทำโครงงานในครั้งนี้ มีอุปสรรคในการทำงานค่อนข้างหลายด้าน ทั้งในเรื่องของเวลาในการทำโครงงาน เนื่องจากมีเวลาในการทำงานค่อนข้างจำกัด และปัญหาทางด้านอุปกรณ์ NB-IoT Shield ที่เป็นผลิตภัณฑ์ตัวใหม่ของบริษัทฯ ทำให้มีทรัพยากรและบทความข้อมูลเกี่ยวกับผลิตภัณฑ์ที่มารองรับค่อนข้างน้อย ส่งผลให้ต้องใช้เวลาศึกษาอุปกรณ์ค่อนข้างนาน ทั้งในด้านฮาร์ดแวร์ และซอฟแวร์ นอกจากนี้แล้วยังต้องมีการเรียนรู้ภาษาโปรแกรมอื่น ๆ ที่ต้องประยุกต์ใช้เพิ่มมากขึ้นอีกด้วย

ด้วยข้อจำกัดทางเวลา ทำให้ระบบที่สร้างขึ้นนั้นยังมีข้อจำกัดบางอย่างที่ยังทำการศึกษาไม่เสร็จสิ้น เช่น การลดขนาดวงจรรวม การเพิ่มประสิทธิภาพในการบริโภคพลังงานของระบบ เป็นต้น

5.4 แนวทางในการพัฒนาต่อ

ในการนำระบบมาตรวัดสังเกตการณ์พลังงานไฟฟ้าบนระบบไอโอทีแบนด์แคบที่ได้สร้างขึ้นไปใช้งานจริง ๆ ในวงกว้างนั้น ระบบควรจะได้รับการพัฒนาในด้านของวงจรของระบบ โดยการลดและตัดชิ้นส่วนอุปกรณ์ที่ไม่จำเป็นต่อการใช้งานออก และทำการรวมวงจรทั้งหมดเข้าไว้ด้วยกันเป็นวงจรเดียวกัน เพื่อให้สามารถติดตั้งและผลิตในจำนวนมาก ๆ ได้ง่าย ลดต้นทุน รวมถึงเพิ่มประสิทธิภาพด้านการบริโภคพลังงานของระบบอีกด้วย

บรรณานุกรม

[1] ดร. ธิรินทร์ ณ ถลาง. *คู่มือผู้รับผิดชอบด้านพลังงาน(อาคาร) พ.ศ. 2553.* กรมพัฒนาพลังงานทดแทนและอนุรักษ์พลังงาน กระทวงพลังงาน, 2553

[2] Khanaresa Chonsawat. *การแบ่งประเภทของ Energy Meter.* factomart. แหล่งข้อมูล : https://www.factomart.com/th/factomartblog/type-of-energy-meter/topic256, ทำการค้นคว้าเมื่อ 12 กันยายน 2561

[3] พิชิต จินตโกศลวิทย์. *สมาร์ทกริด กริดไฟฟ้าอัจฉริยะ (ตอนที่ 7).* thailandindustry. แหล่งข้อมูล : http://www.thailandindustry.com/onlinemag/view2.php?id=60, ทำการค้นคว้าเมื่อ 15 กันยายน 2561

[4] ดร.กอบเกียรติ สระอุบล. *พัฒนา IoT บนแพลตฟอร์ม Arduino และ Rasberry Pi.* ด่านสุธาการพิมพ์ จำกัด, 2561

[5] *อินเทอร์เน็ตในทุกสิ่ง (Internet of Things).* thailandindustry. แหล่งข้อมูล : http://www.thailandindustry.com/onlinemag/view2.php?id=60,ทำการค้นคว้าเมื่อ 27 กันยายน 2561

[6] Y.-P. Eric Wang. *A Primer on 3GPP Narrowband Internet of Things (NB-IoT).* Ericsson Research, 2017

[7] ประภาส สุวรรณเพชร. *เรียนรู้และลองเล่น Arduino เบื้องต้น.* แผนกวิชาช่างอิเล็กทรอนิกส์ วิทยาลัยเทคนิคชัยภูมิ, 2560

[8] James F. Kurose. *Computer Networking 6th Edition.* PEARSON, 2013

[9] *ทำความรู้จักกับ MQTT และ CoAP โปรโทคอลสำหรับรับส่งข้อมูลบนเครือข่าย IoT.* adslthailand. แหล่งข้อมูล : http://www.adslthailand.com/post/mqtt-coap-comparison-iot-protocol, ทำการค้นคว้าเมื่อ 14 ตุลาคม 2561

[10] *Hypertext Transfer Protocol (HTTP).* มหาวิทยาลัยสงขลานครินทร์. แหล่งข้อมูล : http://staff.cs.psu.ac.th/noi/cs344-481/group11_Http/HTTP.htm, ทำการค้นคว้าเมื่อ 14 ตุลาคม 2561

[11] Zako. *ทำความรู้จักกับ HTTP.* ZakoSchool. แหล่งข้อมูล : https://zakoschool.herokuapp.com/บทความที่เกี่ยวข้อง/0, ทำการค้นคว้าเมื่อ 15 ตุลาคม 2561

[12] Wutti Tarn. *เริ่มเรียน Javascript (JS).* Medium.com. แหล่งข้อมูล : https://medium.com/@wuttitarn/เริ่มเรียน-javascript-js-cf8fb165545e, ทำการค้นคว้าเมื่อ 17 ตุลาคม 2561

บรรณานุกรม (ต่อ)

[13] PONGSAKRIVERPLUS. *PLC Protocol: การสื่อสารแบบ Modbus Protocol.* RIVERPLUS. แหล่งข้อมูล : https://riverplusblog.com/2011/08/18/plc-protocol-การสื่อสารแบบ-modbus-protocol/, ทำการค้นคว้าเมื่อ 14 ตุลาคม 2561

[14] Webmaster. *PLC Protocol: อัตราค่าไฟฟ้าฉบับ พ.ย. 2558.* กฟน., 2558

ประวัติผู้จัดทำ

ชื่อ–สกุล สรรเสริญ เกิดแก่น

วัน เดือน ปีเกิด 27 กรกฎาคม 2539

ที่อยู่ปัจจุบัน 3/49 หมู่ 18 ตำบลคูคต อำเภอลำลูกกา

จังหวัดปทุมธานี 12130

ประวัติการศึกษา

พ.ศ. 2557 มัธยมศึกษาตอนปลาย

โรงเรียนนวมินทราชินูทิศ สวนกุหลาบวิทยาลัย ปทุมธานี

พ.ศ. 2561 วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง