เครื่องวัดปริมาณการใช้ไฟฟ้า

Electric meter

นายสันติพงศ์ สันใจ

Santipong Sanjai

นายอภิชาต สุขสวัสดิ์

Aphichat Suksawad

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2561

เครื่องวัดปริมาณการใช้ไฟฟ้า

Electric meter

โดย

นายสันติพงศ์ สันใจ รหัสประจำตัวนักศึกษา 58011291

นายอภิชาต สุขสวัสดิ์ รหัสประจำตัวนักศึกษา 58011406

อาจารย์ที่ปรึกษา

อาจารย์ชินภัทร นันทจิวากรชัย

ปริญญานิพนธ์นี้เป็นสวนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกลาเจาคุณทหารลาดกระบัง

พ.ศ.2561

ปริญญานิพนธ์ปีการศึกษา 2561

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องวัดปริมาณการใช้ไฟฟ้า

Electric meter

ผู้จัดทำ นาย สันติพงศ์ สันใจ รหัสนักศึกษา 58011291

นาย อภิชาต สุขสวัสดิ์ รหัสนักศึกษา 58011406

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

(อาจารย์ชินภัทร นันทจิวากรชัย)

หัวข้อปริญญานิพนธ์ เครื่องวัดปริมาณการใช้ไฟฟ้า
นักศึกษา นาย สันติพงศ์ สันใจ รหัสนักศึกษา 58011291
นาย อภิชาต สุขสวัสดิ์ รหัสนักศึกษา 58011406
ปริญญา วิศวกรรมศาสตร์บัณฑิต
สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์
ปีการศึกษา 2561
อาจารย์ที่ปรึกษา อาจารย์ชินภัทร นันทจิวากรชัย

บทคัดย่อ

โครงงานเรื่องเครื่องวัดปริมาณการใช้ไฟฟ้า เป็นโครงงานประเภทประยุกต์ความรู้ ซึ่งจะนำเสนอแอพพลิเคชันที่สามารถเช็คกระแสและแรงดันการใช้งานของเครื่องใช้ไฟฟ้าภายในบ้านโดยมีการแสดงผลทั้งเป็นค่าและกราฟ มีการจัดทำขึ้นเพื่ออำนวยความสะดวกในการเช็คไฟหากมีอุปกรณ์ชิ้นใดเกิดการมีปัญหาขึ้น ซึ่งเป็นการส่งข้อมูลจากตัววัดมาที่แอพพลิเคชันโดยการส่งผ่านสัญญาณ Wifi ซึ่งเป็นสิ่งที่ทุกครัวเรือนมีเป็นพื้นฐาน

Thesis Title	Electric meter
Student	Santipong Sanjai 58011291
	Apichat Suksawad 58011406
Degree	Bachelor of Engineering
Program	Electronics Engineering
Year	2561
Thesis Advisor	Chinnapat Nantajiwarakornchai

Abstract

Project on electricity consumption measurement is applied of knowledge project. Bring applications that can check the current and voltage of the electrical appliances in the home with both values and graphs. Made to facilitate the check of electricity, if there is any device that has a problem, which is an application that sends data from the module to the application by passing the Wifi what every household has as a basis.

กิตติกรรมประกาศ

โครงงาน เครื่องวัดปริมาณการใช้ไฟฟ้า เสร็จสมบูรณ์ได้เนื่องจากได้รับการสนับสนุนจากอาจารย์ที่ปรึกษา อ.ชินภัทร นันทจิวากรชัย ที่ให้ความรู้ คำแนะนำ ให้การสนับสนุนด้านอุปกรณ์ และสถานที่ในการทำงาน รวมถึงการดูแลอย่างดีในด้านต่างๆ นอกจากนี้ยังได้รับการสนับสนุนจากอาจารย์ทุกท่านในภาควิชาอิเล็กทรอนิกส์ รวมไปถึงตึกภาควิชาอิเล็กทรอนิกส์ ที่เป็นสถานที่ทำงานให้เสร็จลุล่วงไปได้ด้วยดี ทั้งนี้ขอขอบคุณ NCT 127, WayV, NCT Dream ศิลปินที่เป็นแรงบันดาลใจและกำลังใจทางจิตใจของขาพเจ้ามาโดยตลอด ทั้งยังเป็นแรงผลักดันในการทำโครงงานนี้จนแล้วเสร็จ จึงใคร่ขอขอบพระคุณผู้มีอุปการคุณทุกท่านมา ณ ที่นี้

สันติพงศ์ สันใจ

อภิชาต สุขสวัสดิ์

ผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในปัจจุบัน IOT มาเป็นส่วนหนึ่งของการใช้ชีวิตของมนุษย์เราเกือบจะทั้งหมดเพื่อเป็นการอำนวยความสะดวก ซึ่งหากเราสามารถสั่งการเปิด-ปิดสวิตช์ไฟผ่านแอปพลิเคชันได้และวัดการใช้ไฟผ่านทางแอปพลิเคชันได้ก็จะเป็นการอำนวยความสะดวกยิ่งขึ้น จึงเป็นที่มาของโครงงานนี้ ซึ่งก็คือการที่เราควบคุมอุปกรณ์หรือดูปริมาณการใช้งานอุปกรณ์ไฟฟ้าได้ โดยที่ผู้ใช้สามารถเช็คข้อมูลปริมาณการใช้งานผ่านแอปพลิเคชันและดูข้อมูลย้อนหลังได้อีกด้วย

1.2 ความมุ่งหมายและจุดประสงค์ของการศึกษา

ในการทำโครงงานครั้งนี้ จัดทำขึ้นเพื่อ

1.2.1 เป็นการฝึกการปฏิบัติงานเพื่อเป็นพื้นฐานในการปฏิบัติงาน รวมถึงฝึกการทำงานร่วมกับผู้อื่น

1.2.2 บูรณาการความรู้ที่ได้จากการเรียนในภาคทฤษฎีนำมาใช้ในการปฏิบัติงานจริง

1.2.3 ประยุกต์ใช้งานเพื่อปฏิบัติจริงของการส่งสัญญาณผ่านระบบไร้สายและ IOT

1.2.4 ศึกษาการวัดกระแสและแรงดันของสิ่งของที่มีการใช้ไฟในบ้านโดยที่สามารถตรวจสอบการทำงานของเครื่องใช้ไฟฟ้าผ่านแอปพลิเคชัน

1.3 สมมุติฐานการศึกษา

เมื่อได้รับ INPUT ผ่านปุ่มทางแอพพลิเคชั่นก็สามารถควบคุมสวิตช์และก็สามารถกดสวิตช์ได้เช่นเดียวกัน ทั้งยังบอกปริมาณการใช้ไฟฟ้าผ่านแอพพลิเคชันได้

1.4 ขอบเขตการศึกษา

การควบคุมสวิตช์ผ่านแอพพลิเคชันและสามารถกดสวิตช์ด้วยตัวเองได้

การวัดการใช้ไฟฟ้าผ่านตัวบอร์ดไมโครคอนโทรเลอร์โดยมีการบอกค่าเป็นตัวเลขและกราฟ

บทที่ 2

หลักการและทฤษฎี

2.1 ไฟฟ้าในบ้าน

ไฟฟ้าที่ถูกปล่อยออกมาจากปลั๊กไฟ จากเต้ารับภายในบ้านของเรานั้น เป็นไฟฟ้ากระแสสลับที่มีแรงดัน 220V ความถี่ 50Hz กระแสไฟฟ้ารวมที่เราสามารถใช้งานได้จะขึ้นอยู่กับหม้อแปลงไฟฟ้าที่การไฟฟ้าติดตั้งให้ และตรวจดูเลขทุกเดือน (บางบ้านอาจจะเป็นระบบ IoT ที่ไม่มาตรวจที่บ้านแล้ว)

ไฟฟ้าตามบ้านนั้น เป็นระบบไฟฟ้าแบบ 1 เฟส มี 2 สาย ดังนี้

- สายเคอร์เรนท์ (Current line : L) - เป็นสายที่มีกระแสไฟฟ้าไหลอยู่ หากใช้ไขควงวัดไฟทดสอบวัดที่สายเส้นนี้ หลอดไฟที่ไขควงจะติดสว่างขึ้นมา

- สายนิวทรัล (Neutral line : N) - เป็นสายที่ไม่มีกระแสไฟฟ้าไหลอยู่ แต่จำเป็นต้องมีเพื่อให้กระแสไฟฟ้าไหลได้ครบวงจร หากนำไขควงวัดไฟทดสอบสายเส้นนี้ หลอดไฟที่ไขควงจะไม่ติดสว่างขึ้นมา

สำหรับปลั๊กไฟที่มี 3 ขานั้น จะประกอบไปด้วยขา L N แต่จะมีสายดิน (Ground) เพิ่มขึ้นมา เพื่อใช้ป้องกันเหตุการณ์บางอย่างที่ทำให้กิดไฟรั่ว แล้วอาจช็อตผู้ใช้งานได้ การมีสายดินจะช่วยให้ไฟฟ้าที่รั่วเหล่านั้นไหลลงดินแล้วไม่ทำให้ช็อตตัวผู้ใช้ได้

ตัวแปรที่เกี่ยวข้องกับอัตราการใช้ไฟฟ้า

ตัวแปรด้านไฟฟ้าจะมีด้วยกัน 3 ตัว คือ แรงดันไฟฟ้า กระแสไฟฟ้า และความต้านทานทางไฟฟ้า ด้วยตัวแปร 3 ตัวนี้ สามารถแตกแยกตัวแปรที่ใช้ในการวัดอัตราการใช้ไฟฟ้าได้อีก 2 ตัว คือ กำลังไฟฟ้า วัตต์ชั่วโมง

- แรงดันไฟฟ้า (E) มีหน่วยเป็นโวล์ (V)
- กระแสไฟฟ้า (I) มีหน่วยเป็นแอมป์ (A)
- ความต้านทานทางไฟฟ้า (R หรือ อิมพีแดนซ์) มีหน่วยเป็น โอห์ม
- กำลังไฟฟ้า (P) มีหน่วยเป็นวัตต์ (W)
- วัตต์ชั่วโมง (Wh) มีหน่วยเป็นวัตต์ (W)

ค่าที่ต้องวัด แล้วทราบจริง ๆ จะมีเพียง 2 ค่า คือแรงดันไฟฟ้า (E) และกระแสไฟฟ้า (I) ส่วนค่าอื่น ๆ จะสามารถหาได้โดยใช้สูตร

- ค่าความต้านทานทางไฟฟ้า (R หรือ อิมพีแดนซ์) สามารถหาได้จากสูตร R = E / I แต่โดยทั่วไปจะไม่หาค่านี้อยู่แล้ว และไม่ถูกนำมาใช้แสดงเกี่ยวกับการใช้พลังงานไฟฟ้า

- กำลังไฟฟ้า (P) สามารถหาได้จากสูตร P = E * I มักใช้ระบุการใช้พลังงานไฟฟ้า ณ ขณะนั้น

- วัตต์ชั่วโมง (Wh) สามารถหาได้จาก P * T เมื่อ T คือเวลาในหน่วยวินาที แต่ในความเป็นจริงแล้วค่า P มักจะแปรผันไปตามค่า E และ I ซึ่งแต่ละช่วงเวลาค่า E และ I สามารถเปลี่ยนแปลงได้ตลอดเวลา ดังนั้นวิธีการหาวัตต์ชั่วโมง จึงใช้การจับเวลาให้ครบ 1 วินาที นำค่า P มาบวกเข้าไปเรื่อย ๆ ก็จะได้ค่าวัตต์ชั่วโมงออกมา

ในการคำนวณค่าไฟฟ้า เรามักจะใช้หน่วย ยูนิต เป็นหน่วยใช้ในการคำนวณ โดย 1 ยูนิต = 1kWh หรือ 1 ยูนิต = 1000Wh ดังนั้นหากต้องการหาค่ายูนิต จะหาได้จาก วัตต์ชั่วโมง / 1000 สมมุตวัตต์ชั่วโมงได้ 750Wh จะได้ 750 / 1000 = 0.75 ยูนิต

จะเห็นว่าค่าที่เราวัดออกมาทั้งหมด สุดท้ายแล้วจะได้ค่าออกมาเป็นยูนิตที่นำไปคำนวณเป็นค่าไฟฟ้าได้ ซึ่งจะทำให้สามารถทราบได้ว่าเครื่องใช้ไฟฟ้านั้น ๆ ใช้จำนวนเงินได้เท่าไรเมื่อเปิดใช้งานเป็นเวลาหนึ่ง

2.2 การคำนวนค่าไฟฟ้า

การคำนวนค่าไฟฟ้าจะแบ่งตามประเภทของไฟฟ้าที่ใช้ ซึ่งในบทความนี้จะสอนวิธีการคำนวณค่าไฟฟ้าประเภทที่ 1 บ้านอยู่อาศัยอัตราปกติ

การคำนวณค่าไฟฟ้าจะแบ่งออกเป็น 3 ช่วง ดังนี้

- ค่าพื้นฐานไฟฟ้า จะคิดแบบขั้นบันได เช่น 15 ยูนิตแรก จะคิดราคาหนึ่ง 10 ยูนิตถัดไป คิดอีกราคาหนึ่ง เป็นต้น มักจะมีการอัพเดทค่าไฟฟ้าส่วนนี้ทุก ๆ 3 ปี

- ค่าไฟฟ้าผันแปร (Ft) จะเปลี่ยนไปตามราคาพลังงานต้น (ถ่านหิน , น้ำมัน) นำยูนิตมาคิดตรง ๆ และอัพเดททุก ๆ 4 เดือน

- ภาษีมูลค่าเพิ่ม (VAT) 7% ของทั้ง 2 ส่วนด้านบนรวมกัน

ดังนั้น ค่าไฟฟ้าที่ต้องชำระจึงหาได้จาก ค่าพื้นฐานไฟฟ้า + ค่าไฟฟ้าผันแปร + ภาษีมูลค่าเพิ่ม นั่นเอง แต่กรณีใช้ไฟฟ้าไม่เกิน 50 ยูนิตต่อเดือน จะไม่ต้องชำระค่าไฟฟ้าในเดือนนั้น

ค่าไฟฟ้าประเภทที่ 1 บ้านอยู่อาศัยอัตราปกติ จะแบ่งประเภทของผู้ใช้ได้อีก 2 แบบ ดังนี้

ใช้ไฟฟ้าไม่เกิน 150 หน่วย (ยูนิต) ต่อเดือน

คุณสมบัติของผู้ใช้ที่เข้าประเภทนี้ คือ

• มีการใช้ไฟฟ้าต่อเดือนไม่เกิน 150 หน่วยต่อเดือน และไม่เคยใช้เกิน 150 หน่วยต่อเดือนต่อเนื่องเกิน 3 เดือน (เกิน 150 หน่วยต่อเนื่องครบ 3 เดือน จะเข้าประเภทเกิน 150 หน่วย)

• ติดตั้งหม้อแปลงไม่เกิน 5A 1 เฟส

อัตราค่าพื้นฐานไฟฟ้า (ปี พ.ศ.2560) มีดังนี้

- ยูนิตที่ 1 - 15 คิดยูนิตละ 2.3488 บาท (สูงสุด 35.23 บาท)
- ยูนิตที่ 16 - 25 (10 ยูนิตถัดมา) คิดยูนิตละ 2.9882 บาท (สูงสุด 29.88 บาท)
- ยูนิคที่ 26 - 35 (10 ยูนิตถัดมา) คิดยูนิตละ 3.2405 บาท (สูงสุด 32.41 บาท)
- ยูนิคที่ 36 - 100 (65 ยูนิตถัดมา) คิดยูนิตละ 3.6237 บาท (สูงสุด 235.54 บาท)
- ยูนิคที่ 101 - 150 (50 ยูนิตถัดมา) คิดยูนิตละ 3.7171 บาท (สูงสุด 185.86 บาท)
- ยูนิคที่ 151 - 400 (250 ยูนิตถัดมา) คิดยูนิตละ 4.2218 บาท (สูงสุด 1,055.45 บาท)
- ยูนิคที่ 401 เป็นต้นไป คิดยูนิตละ 4.4217 บาท
- ค่าบริการ 8.19 บาท

สมมติ ใช้ไฟฟ้า 105 ยูนิต การคำนวนให้นำค่าสูงสุดของแต่ละขั้นมาบวกกันเรื่อย ๆ กรณีเหลือเป็นเศษค่อยนำจำนวนยูนิตที่เหลือมาคูณตัวเลขแต่ละยูนิตในขั้นนั้น ๆ

- ยูนิตที่ 1 - 15 คิด 35.23 บาท
- ยูนิตที่ 16 - 25 คิด 29.88 บาท
- ยูนิคที่ 26 - 35 คิด 32.41 บาท
- ยูนิคที่ 36 - 100 คิด 235.54 บาท
- ยูนิคที่ 101 - 105 คิด 5 * 3.7171 = 18.5855 บาท ปัดเศษเหลือ 2 หลัก ได้ 18.59 บาท
- ค่าบริการ 8.19 บาท

รวมอัตราค่าไฟฟ้าฐาน คือ 35.23 + 29.88 + 32.41 + 235.54 + 18.59 + 8.19 = 359.84 บาท

ใช้ไฟฟ้าเกิน 150 หน่วย (ยูนิต) ต่อเดือน

คุณสมบัติของผู้ใช้ที่เข้าประเภทนี้ คือ

- มีการใช้ไฟฟ้าต่อเดือนเกิน 150 หน่วยต่อเนื่องครบ 3 เดือน
- ติดตั้งหม้อแปลงเกิน 5A 1 เฟส

อัตราค่าพื้นฐานไฟฟ้า (ปี พ.ศ.2560) มีดังนี้

- ยูนิตที่ 1 - 150 คิดยูนิตละ 3.2484 บาท (สูงสุด 487.26 บาท)
- ยูนิตที่ 151 - 400 (250 ยูนิตถัดมา) คิดยูนิตละ 4.2218 บาท (สูงสุด 1,055.45 บาท)
- ยูนิคที่ 401 เป็นต้นไป คิดยูนิตละ 4.4217 บาท
- ค่าบริการ 38.22 บาท

สมมุต ใช้ไฟฟ้า 200 ยูนิต สามารถคำนวณอัตราค่าไฟฟ้าฐานได้ดังนี้

- ยูนิตที่ 1 - 150 คิด 487.26 บาท
- ยูนิตที่ 151 - 200 คิด 50 * 4.2218 = 211.09 บาท
- ค่าบริการ 38.22 บาท

รวมอัตราค่าไฟฟ้าฐาน คือ 487.26 + 211.09 + 38.22 = 736.57 บาท

ค่าไฟฟ้าผันแปร (Ft)

ค่า Ft จะเปลี่ยนทุก ๆ 3 เดือน และเป็นค่าบวก หรือค่าลบก็ได้ เช่น ช่วงเดือนกันยายน ถึง ธันวาคม ของปี พ.ศ.2560 ค่า Ft อยู่ที่ -15.90 บาท

สมมุต (1) ใช้ไฟฟ้า 105 ยูนิต ค่า Ft หาได้จาก 105 * (-15.90 / 100) = -16.695 บาท ปัดเศษ ได้ -16.7 บาท

สมมุต (2) ใช้ไฟฟ้า 200 ยูนิต ค่า Ft หาได้จาก 200 * (-15.90 / 100) = -31.8 บาท

ภาษีมูลค่าเพิ่ม (VAT) 7%

คิดจากผลรวมของอัตราค่าไฟฟ้าฐาน และ ค่าไฟฟ้าผันแปร (Ft) นำมาคูณด้วย 7 / 100 หรือคูณด้วย 0.07

สมมุต (1) ค่าผลรวมของงอัตราค่าไฟฟ้าฐาน และ ค่าไฟฟ้าผันแปร (Ft) คือ 359.84 + (-16.7) = 343.14 บาท นำมาคิด VAT ได้ 343.14 * 0.07 = 24.0198 บาท ปัดเศษ ได้ 24.02 บาท

สมมุต (2) ค่าผลรวมของงอัตราค่าไฟฟ้าฐาน และ ค่าไฟฟ้าผันแปร (Ft) คือ 736.57 + (-31.8) = 704.77 บาท นำมาคิด VAT ได้ 704.77 * 0.07 = 49.3339 บาท ปัดเศษ ได้ 49.33 บาท

รวมค่าไฟฟ้าที่ต้องชำระ

นำค่าทั้ง 3 รวมกัน คือ ค่าพื้นฐานไฟฟ้า + ค่าไฟฟ้าผันแปร + ภาษีมูลค่าเพิ่ม

สมมุต (1) ค่าพื้นฐานไฟฟ้า + ค่าไฟฟ้าผันแปร + ภาษีมูลค่าเพิ่ม คือ 359.84 + (-16.7) + 24.02 รวมต้องชำระ 367.16 บาท

สมมุต (2) ค่าพื้นฐานไฟฟ้า + ค่าไฟฟ้าผันแปร + ภาษีมูลค่าเพิ่ม คือ 736.57 + (-31.8) + 49.33 รวมต้องชำระ 754.1 บาท

2.3 PZEM-004T

PZEM-004T คือโมดูลวัดไฟ AC 220v โมดูลเดียวสามารถวัดได้ทั้ง แรงดัน (V) ตั้งแต่ 80-260VAC และ กระแส (A) โดยรองรับกระแสสูงสุด 0-100A โดยวัดกระแสผ่าน Coil ไม่สัมผัสกับสายไฟโดยตรง Output ออกผ่านทาง Serial TX RX สามารถต่อกับ Arduino, ESP32, Raspberry Pi, หรือ ESP8266 NodeMCU นิยมทำเป็นโปรเจค Meter IoT แสดงผลการใช้กระแสแบบ Online (ESP8266 ไม่ควรต่อกับ TX RX โดยตรงควรมี Voltage divider หรือ logic converter ก่อน)

รูปที่ 2.1 PZEM-004T

2.3.1 ฟังก์ชัน

2.3.1.1 ฟังก์ชันการวัดพารามิเตอร์ไฟฟ้า (voltage, current, active power, energy).

2.3.1.2 ฟังก์ชันแจ้งเตือนเสมือน Overload (เมื่อมีการแจ้งเตือนเกินกำลังไฟ จะมีไฟและเสียงดังขึ้น)

2.3.1.3 ฟังก์ชันการตั้งค่าการเตือนภัย

2.3.1.4 ฟังก์ชันรีเซ็ตค่า

2.3.1.5 เก็บข้อมูลหากมีการปิดการทำงาน (จัดเก็บพลังงานสะสมไว้ก่อนปิดเครื่อง)

2.3.1.6 ฟังก์ชันการสื่อสารแบบอนุกรม (กับอินเตอร์เฟซอนุกรม TTL ตัวเอง, สามารถสื่อสารกับความหลากหลายของ terminal ผ่านบอร์ดพิน, อ่านและตั้งค่าพารามิเตอร์)

2.3.2 การแสดงผล

2.3.2.1 Power: ทดสอบช่วง : 0 ~ 22kW

ช่วง 0 ~ 10kW, แสดงผลเป็น 0.000 ~ 9.999

ช่วง 10 ~ 22kW, แสดงผลเป็น 10.00 ~ 22.00

2.3.2.2 Energy: ทดสอบช่วง : 0 ~ 9999kWh

ช่วง 0 ~ 10kWh, แสดงผลเป็น 0.000 ~ 9.999

ช่วง 10 ~ 100kWh, แสดงผลเป็น 10.00 ~ 99.99

ช่วง 100 ~ 1000kWh, แสดงผลเป็น 100.0 ~ 999.9

1000 ~ 9999kWh and above, แสดงผลเป็น 1000 ~ 9999.

2.3.2.3 Voltage: ทดสอบช่วง : 80 ~ 260VAC, แสดงผลเป็น 110.0 ~ 220.0

2.3.2.4 Current: ทดสอบช่วง : 0 ~ 100A, แสดงผลเป็น 00.00 ~ 99.99

2.3.3 ปุ่มกดบนบอร์ด

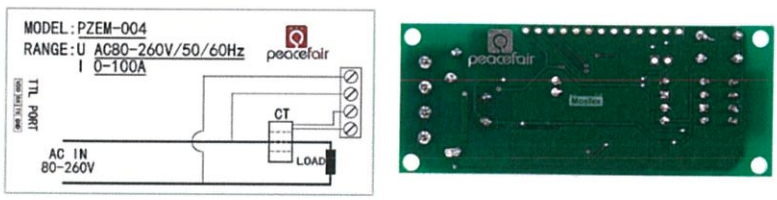
มีปุ่มบนแผงควบคุมสามารถใช้เพื่อรีเซ็ตพลังงาน

วิธีการรีเซ็ตพลังงาน : กดปุ่มค้างไว้เป็นเวลา 5 วินาทีจนกระทั่งหน้าต่างดิจิตอลบนจอแสดงผล

กะพริบจากนั้นปล่อย กดปุ่มอีกครั้งจากนั้นข้อมูลพลังงานจะถูกล้างและออก

หมายเหตุ : หากกดค้างไว้นานเกิน 5 วินาทีจนกระทั่งไม่มีการกะพริบหมายความว่าออกจากสถานะการรีเซ็ต

2.3.4 การเดินสายของอุปกรณ์



รูปที่ 2.2 การเดินสายอุปกรณ์

2.3.4 การแสดงผล

2.3.4.1 Voltage Display

วัดและแสดงแรงดันไฟฟ้าของกระแสไฟฟ้า

2.3.4.2 Current display

วัดและแสดงกระแสไฟฟ้าที่โหลด (กระแสไฟฟ้า)

หมายเหตุ : ค่าทดสอบกระแสเริ่มที่ 10mA แต่โมดูลนี้เป็นของอุปกรณ์ทดสอบกำลังสูงถ้าคุณต้องการความถูกต้องของการทดสอบระดับ mA ไม่แนะนำโมดูลนี้

2.3.4.3 Energy display

วัดและแสดงการใช้พลังงานสะสม

หมายเหตุ : ที่หน่วยวัดต่ำสุดของการวัดพลังงานคือ 0.001kWh ซึ่งหมายความว่าจะเริ่มสะสมจาก 1Wh, ซึ่งเป็นความความละเอียดค่อนข้างสูงสำหรับการทดสอบกำลังโหลดต่ำ (ใน 100W)

2.3.4.4 Power display

วัดและแสดงกำลังโหลดปัจจุบัน

หมายเหตุ : ค่าทดสอบกำลังคือ เริ่มที่ 0.001kW ซึ่งหมายความว่าจะทำที่ 1W แต่โมดูลนี้เป็นอุปกรณ์ทดสอบที่ high power หากต้องการใช้ในช่วงต่ำกว่า 1W ก็จะไม่แนะนำให้ใช้โมดูลนี้

2.3.5 Serial communication

โมดูลนี้มีอินเทอร์เฟซการสื่อสารข้อมู TTL สามารถอ่านและตั้งค่าพารามิเตอร์ที่เกี่ยวข้องผ่านทางพอร์ตได้ แต่ถ้าคุณต้องการสื่อสารกับอุปกรณ์ที่มี USB หรือ RS232 (เช่นคอมพิวเตอร์) จะต้องเชื่อมต่อ ขาบนบอร์ด TTL ที่ต่างกัน (หากต้องการใช้ USB ต้องต่อ TTL เข้ากับ USB pin board; หาต้องการใช้ RS232 ต้องต่อ TTL เข้ากับบอร์ด RS232) ประเภทการต่อที่เฉพาะเจาะจงจะแสดงที่ตารางด้านล่าง

ตารางที่ 2.1 communication protocols of this module

NO.	Function	Head	Data 1 – 5	Sum
1	Voltage	B0	C0 A8 01 01 00 (Computer sends a request to read the voltage value)	1A
		A0	00 E6 02 00 00 (Meter reply the voltage value is 230.2V)	88
2	Current	B1	C0 A8 01 01 00 (Computer sends a request to read the current value)	1B
		A1	00 11 20 00 00 (Meter reply the current value is 17.32A)	D2
3	Active Power	B2	C0 A8 01 01 00 (Computer sends a request to read the active power value)	1C
		A2	08 98 00 00 00 (Meter reply the active power value is 2200w)	42
4	Read energy	B3	C0 A8 01 01 00 (Computer sends a request to read the energy value)	1D
		A3	01 86 9f 00 00 (Meter reply the energy value is 99999wh)	C9
5	Set the module address	B4	C0 A8 01 01 00 (Computer sends a request to set the address, the address is 192.168.1.1)	1E
		A4	00 00 00 00 00 (Meter reply the address was successfully set)	A4
6	Set the power alarm threshold	B5	C0 A8 01 01 14 (computer sends a request to set a power alarm threshold)	33
		A5	00 00 00 00 00 (Meter reply the power alarm threshold was successfully set)	A5

ตัวอย่างของโปรโตคอลการสื่อสาร :

ตั้งค่า the communication address: 192.168.1.1

Send command: B4 C0 A8 01 01 00 1E

Reply data: A4 00 00 00 00 00 A4

หมายเหตุ : ตัวอย่างข้างต้นแสดงให้เห็นว่าการตั้งค่าแอดเดรสการสื่อสารเป็น 192.168.1.1 (ผู้ใช้สามารถกำหนดที่อยู่ของตัวเองตามความต้องการ) การส่งคำสั่งและการตอบกลับข้อมูล ข้อมูลจะแสดงเป็นเลขฐานสิบหก ไบต์สุดท้ายของการส่งและข้อมูลการตอบกลับคือ 1E และ A4 เป็นจำนวนเงินสะสม เมื่อส่งคำสั่ง: B4 + C0 + A8 + 01 + 01 + 00 = 21E (ใช้เลขฐานสิบหก) ข้อมูลสะสมรวมเท่ากับ 21E ใช้เวลาสองไบต์ล่าสุด 1E เพื่อใช้ข้อมูลสะสมในการส่งคำสั่ง ข้อมูลในการตอบ: A4 + 00 + 00 + 00 + 00 + 00 = A4 (ใช้เลขฐานสิบหก) ข้อมูลสะสมรวมคือ A4 ซึ่งเป็นข้อมูลรวมสะสมในคำตอบสุดท้าย

Set the power alarm threshold:20 KW

Send command: B5 C0 A8 01 01 14 33

Reply data: A5 00 00 00 00 00 A5

หมายเหตุ : 14 ในคำสั่งการส่งเป็นค่าปลุก (14 คือการแทนข้อมูลเลขฐานสิบหกซึ่งแปลงเป็นทศนิยมคือ 20) สิ่งที่ควรทราบคือค่าปลุกของโมดูลนี้อ้างอิงหน่วย KW ซึ่งหมายความว่าค่าแจ้งเตือนต่ำสุดคือ 1KW ค่าสูงสุดคือ 22KW

Read the current voltage

Send command: B0 C0 A8 01 01 00 1A

Reply data: A0 00 E6 02 00 00 88

หมายเหตุ : ข้อมูลแรงดันไฟฟ้าที่ได้คือ D1D2D3 = 00 E6 02 ซึ่ง 00 E6 แสดงจำนวนเต็มของแรงดันไฟฟ้า และ 02 แสดงทศนิยมของแรงดันทศนิยมทศนิยมเป็นตัวเลขหนึ่งตัว, แปลงเป็นเลขทศนิยมเป็นเลขฐานสิบหกแปลง 2 เป็นเลขทศนิยมเป็น 2 ดังนั้นค่าแรงดันไฟฟ้าในปัจจุบันคือ 230.2V

Read the current current

Send command: B1 C0 A8 01 01 00 1B

Reply data: A1 00 11 20 00 00 D2

หมายเหตุ : อ่านค่ากระแสเป็น D2D3 = 11 20 ซึ่ง11 แสดงจำนวนเต็มของบิตปัจจุบัน 20 หมายถึงทศนิยมของปัจจุบันทศนิยมในปัจจุบันเป็นตัวเลขสองหลัก 11 แปลงเป็นทศนิยมคือ 17 และ แปลง 20 เป็นทศนิยมคือ 32 ดังนั้นค่าปัจจุบันคือ 17.32 A.

Read the current power

Send command: B2 C0 A8 01 01 00 1C

Reply data: A2 08 98 00 00 00 42

หมายเหตุ : ค่ากำลังไฟฟ้าคือ D1D2 = 08 98 ซึ่งแปลงเป็น 08 98 ถึงทศนิยมคือ 2200 ดังนั้นแรงดันไฟฟ้าในปัจจุบันค่าคือ 2200 วัตต์

Read the energy

Send command: B3 C0 A8 01 01 00 1D

Reply data: A3 01 86 9F 00 00 C9

หมายเหตุ : ค่าพลังงานคือ D1D2D3 = 01 86 9F ซึ่งแปลง 01 86 9F เป็นทศนิยมคือ 99999 ดังนั้นพลังงานสะสมคือ 99999Wh

Illustration of the communication

1. เชื่อมต่อลวดแข็งตามแผนผังสายไฟในรูปที่ 1 และ ตารางที่ 1

2. หลังจากเชื่อมต่อสายไฟแล้วกรุณาเลือกพอร์ตสื่อสารซึ่งเป็นซอฟต์แวร์บนคอมพิวเตอร์ชุดนี้

support communication port: COM2\COM3\COM4,

ข้อควรระวังในการใช้งาน

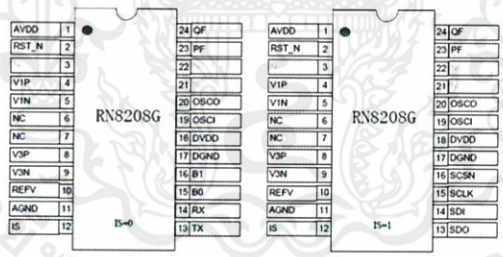
1. โมดูลนี้เหมาะสำหรับใช้ภายในอาคารโปรดอย่าใช้กลางแจ้ง

2. โหลดที่ใช้ไม่ควรเกินกำลังที่กำหนดไว้

Specification parameters

1. Working voltage: 80 ~ 260VAC
2. Test voltage: 80 ~ 260VAC
3. Rated power: 100A/22000W
4. Operating frequency: 45-65Hz
5. Measurement accuracy: 1.0 grade

รูปที่ 2.3 ลายละเอียดบนบอร์ด PZEM-004T



รูปที่ 2.4 ขาของชิป RN8208G

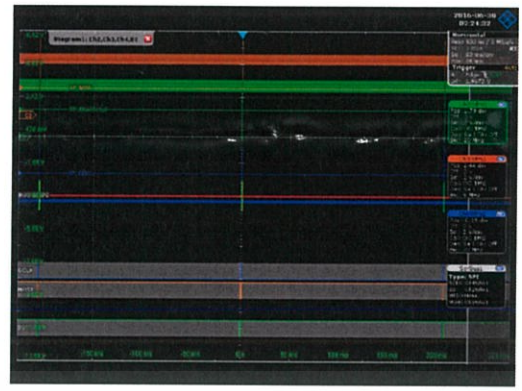
First some scope pictures. of the needed signals :

R9 right side = (DATA FROM ENERGY CHIP)

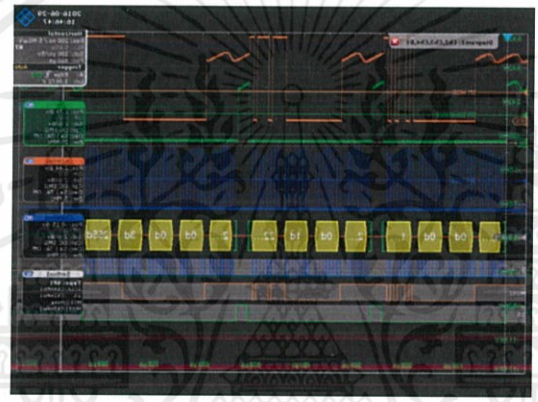
U2 pin 12 (CLOCK TO ENERGY CHIP)

U2 pin 11 (WORD LATCH TO ENERGU CHIP)

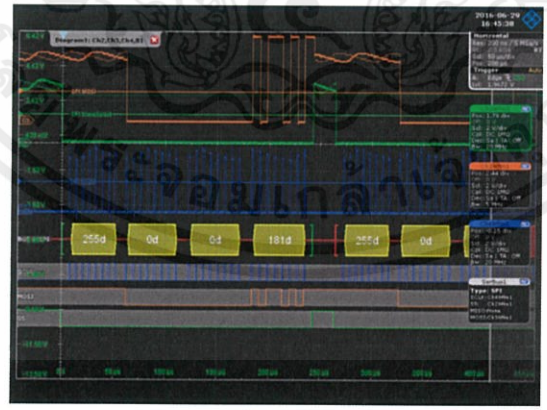
U2 7 (LATCH to DISPLAY CHIP, USED TO SYNC DATA BYTES INTO CORRECT VARIABLES)



รูปที่ 2.25 เวลาในการจัดการข้อมูลและส่งใหม่

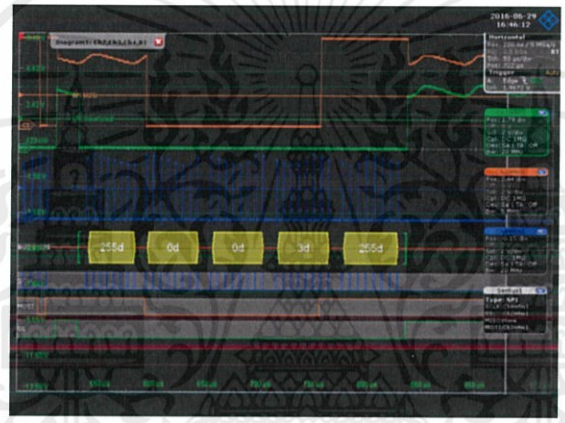


รูปที่ 2.6 The ดีเลย์ระหว่างแต่ละ 8 clocks, ถอยรหัส 8 bits .

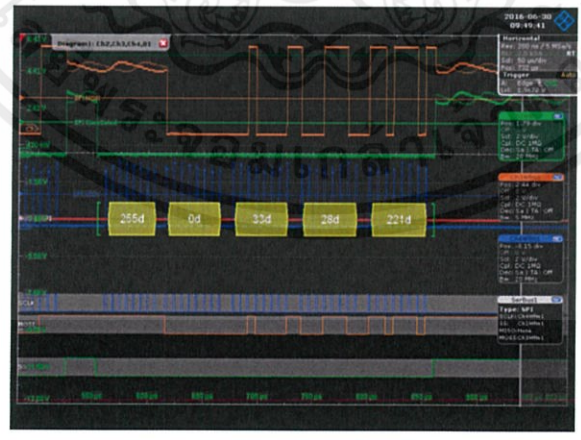


รูปที่ 2.7 VOLTAGE หมายเหตุ : ไบต์คำสั่งอ่าน 255 เนื่องจากข้อมูลจาก RN8208G ลอยตัวในขณะที่ได้รับคำสั่งมิเตอร์อ่าน 0V ในกรณีนี้ เป็นความละเอียด mV เราเพียงแค่เลือกที่จะไม่ใช้ไบต์นั้น

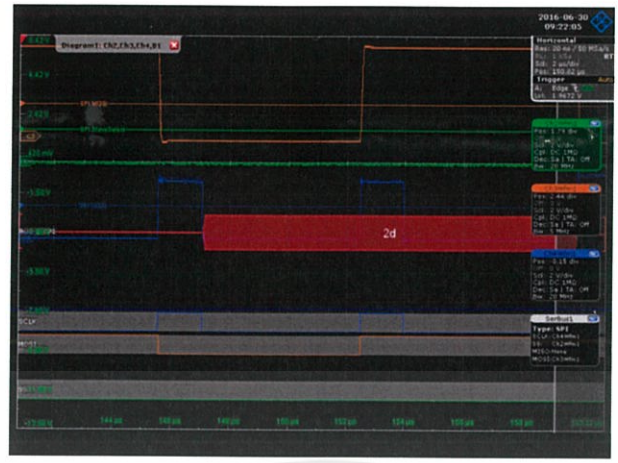
รูปที่ 2.28 CURRENT, ระบบเดียวกัน , เลือกที่จะไม่ใช้ mA



รูปที่ 2.9 WATTS เลือกใช้เฉพาะ 3 ไบต์แรกเท่านั้น



รูปที่ 2.10 WATTS มิเตอร์วัดพลังงานเต็ม 50.0 วัตต์และอ่านค่าในจอ LCD.



รูปที่ 2.11 clock edge vs data tim

2.4 ESP32

ESP32 เป็นชื่อของไอซีไมโครคอนโทรลเลอร์ที่รองรับการเชื่อมต่อ WiFi และ Bluetooth 4.2 BLE ในตัว ผลิตโดยบริษัท Espressif จากประเทศจีน โดยราคา ณ ที่เขียนบทความอยู่นี้ มีราคาไม่เกิน 500 บาท (บอร์ดพัฒนาสำเร็จรูป) โดยตัวไอซี ESP32 มีสเปคโดยละเอียด ดังนี้

1. ซีพียูใช้สถาปัตยกรรม Tensilica LX6 แบบ 2 แกนสมอง สัญญาณนาฬิกา 240MHz
2. มีแรมในตัว 512KB
3. รองรับการเชื่อมต่อรอมภายนอกสูงสุด 16MB
4. มาพร้อมกับ WiFi มาตรฐาน 802.11 b/g/n รองรับการใช้งานทั้งในโหมด Station softAP และ Wi-Fi direct
5. มีบลูทูธในตัว รองรับการใช้งานในโหมด 2.0 และโหมด 4.0 BLE
6. ใช้แรงดันไฟฟ้าในการทำงาน 2.6V ถึง 3V
7. ทำงานได้ที่อุณหภูมิ -40°C ถึง 125°C

นอกจากนี้ ESP32 ยังมีเซ็นเซอร์ต่าง ๆ มาในตัวด้วย ดังนี้

1. วงจรกรองสัญญาณรบกวนในวงจรขยายสัญญาณ
2. เซ็นเซอร์แม่เหล็ก
3. เซ็นเซอร์สัมผัส (Capacitive touch) รองรับ 10 ช่อง
4. รองรับการเชื่อมต่อคลิสตอล 32.768kHz สำหรับใช้กับส่วนวงจรนับเวลาโดยเฉพาะ

ขาใช้งานต่าง ๆ ของ ESP32 รองรับการเชื่อมต่อบัสต่าง ๆ ดังนี้

1. มี GPIO จำนวน 32 ช่อง
2. รองรับ UART จำนวน 3 ช่อง
3. รองรับ SPI จำนวน 3 ช่อง
4. รองรับ I^2C จำนวน 2 ช่อง
5. รองรับ ADC จำนวน 12 ช่อง
6. รองรับ DAC จำนวน 2 ช่อง
7. รองรับ I^2S จำนวน 2 ช่อง
8. รองรับ PWM / Timer ทุกช่อง
9. รองรับการเชื่อมต่อกับ SD-Card

นอกจากนี้ ESP32 ยังรองรับฟังก์ชั่นเกี่ยวกับความปลอดภัยต่าง ๆ ดังนี้

1. รองรับการเข้ารหัส WiFi แบบ WEP และ WPA/WPA2 PSK/Enterprise
2. มีวงจรเข้ารหัส AES / SHA2 / Elliptical Curve Cryptography / RSA-4096 ในตัว

ในด้านประสิทธิ์ภาพการใช้งาน ตัว ESP32 สามารถทำงานได้ดี โดย

1. รับ – ส่ง ข้อมูลได้ความเร็วสูงสุดที่ 150Mbps เมื่อเชื่อมต่อแบบ 11n HT40 ได้ความเร็วสูงสุด 72Mbps เมื่อเชื่อมต่อแบบ 11n HT20 ได้ความเร็วสูงสุดที่ 54Mbps เมื่อเชื่อมต่อแบบ 11g และได้ความเร็วสูงสุดที่ 11Mbps เมื่อเชื่อมต่อแบบ 11b
2. เมื่อใช้การเชื่อมต่อผ่านโปรโตคอล UDP จะสามารถรับ – ส่งข้อมูลได้ที่ความเร็ว 135Mbps
3. ในโหมด Sleep ใช้กระแสไฟฟ้าเพียง 2.5uA
4. จะเห็นได้ว่า ในราคาไม่ถึง 500 บาท (บอร์ดพัฒนาสำเร็จรูป) และโมดูลเปล่าราคาไม่ถึง 400 บาท สามารถให้ประสิทธิ์ภาพได้เกินราคา ด้วยเหตุนี้ ESP32 จึงเหมาะสำหรับนำมาใช้งานมาก ด้วยเหตุผลทางด้านราคา และประสิทธิ์ภาพที่ได้

ฟีเจอร์หลัก

1. 240 MHz dual core Tensilica LX6 microcontroller with 600 DMIPS

2. Integrated 520 KB SRAM

3. Integrated 802.11 b/g/n HT40 Wi-Fi transceiver, baseband, stack and LwIP

4. Integrated dual mode Bluetooth (classic and BLE)

5. 16 MB flash, memory-mapped to the CPU code space

6. 2.3V to 3.6V operating voltage

7. -40°C to +125°C operating temperature

8. On-board PCB antenna / IPEX connector for external antenna

1. มาตรฐานรองรับด้านความปลอดภัย

1. WEP, WPA/WPA2 PSK/Enterprise

2. Hardware-accelerated encryption: AES/SHA2/Elliptical

3. Curve Cryptography/RSA-4096

2. ความสามารถ

1. Max data rate of 150 Mbps@11n HT40, 72 Mbps@11n HT20, 54

2. Maximum transmit power of 19.5 dBm@11b, 16.5 dBm@11g, 15.5 dBm@11n

3. Minimum receiver sensitivity of -97 dBm

4. 5 uA power consumption in Deep-sleep

2.4.1 ESP32 การสื่อสารผ่านโปรโตคอล TCP/IP

โปรโตคอล TCP เป็นหนึ่งในพื้นฐานของโปรโตคอลบนโลกอินเตอร์เน็ต ในโลกของการเชื่อมต่อเครือข่ายแบบอินเตอร์เน็ตจะมีโปรโตคอลที่เป็นพื้นฐานอยู่ 2 ตัว คือ TCP และ UDP

โปรโตคอล UDP

โปรโตคอล UDP เป็นอีกโปรโตคอลหนึ่งบนโลกอินเตอร์เน็ต มีข้อดีคือสามารถรับ – ส่งข้อมูลได้ความเร็วสูงกว่าโปรโตคอล TCP แต่ข้อเสียคือไม่มีการตรวจสอบผลการส่งข้อมูล ทำให้ข้อมูลที่ส่งมีโอกาสส่งไปไม่ถึงผู้รับ โปรโตคอล UDP มักใช้กับข้อมูลที่ต้องการความรวดเร็วและสามารถผิดพลาดได้บ้าง เช่น การรับ-ส่งข้อมูลวีดีโอสตรีมมิ่ง การรับ-ส่งข้อมูลภายในเกมออนไลน์ โปรโตคอลประยุกต์ที่นำไปใช้งาน เช่น DNS SNMP

โปรโตคอล TCP

โปรโตคอล TCP เป็นโปรโตคอลที่ใช้รับ-ส่งข้อมูลที่นิยมใช้งานมากที่สุดในโลกอินเตอร์เน็ต เนื่องจากการรับ – ส่งข้อมูลบนโปรโตคอล TCP จะมีการตรวจสอบผลการรับ – ส่งข้อมูลทุกครั้ง ข้อมูลที่ได้จึงมีความถูกต้องตามลำดับการส่งข้อมูล ทำให้ถูกใช้เป็นพื้นฐานของโปรโตคอลประยุกต์ต่าง ๆ เช่น HTTP FTP SSH IMAP

ในปัจจุบันความเร็วของอินเตอร์เน็ตได้พัฒนาไปมาก ทำให้โปรโตคอล TCP ถูกนำมาใช้งานในด้านสตรีมมิ่งมากขึ้น ส่งผลให้การใช้งานด้านสตรีมมิ่งมีการใช้งานทั้ง TCP และ UDP ในปริมาณที่ใกล้เคียงกัน

การใช้งานโปรโตคอล TCP จะแบ่งออกเป็น 2 ฝั่ง คือ

1. TCP Server คือฝั่งที่ทำหน้าที่เปิดพอร์ตและรอการเชื่อมต่อเข้ามาเพื่อสื่อสารด้วย การใช้ ESP32 ทำเป็น TCP Server จะสะดวกต่อการรับ-ส่งข้อมูลกับอุปกรณ์อื่น ๆ มากที่สุด
2. TCP Client คือฝั่งที่ทำหน้าที่เชื่อมต่อไปยัง Server เพื่อเริ่มการสื่อสาร มักจะหมายถึงอุปกรณ์ที่สามารถเคลื่อนที่ได้ หรือเลือกสื่อสารได้ เช่น คอมพิวเตอร์ สมาร์ทโฟน

ในบทนี้จะเป็นการเรียนรู้ฟังก์ชั่นและทดลองใช้งานการสื่อสารบนโปรโตคอลพื้นฐานที่สุดอย่าง TCP เพื่อปูพื้นฐานไปสู่การใช้โปรโตคอลประยุกต์อื่น ๆ ต่อไป

ฟังก์ชั่นที่เกี่ยวข้องกับการใช้งานโปรโตคอล TCP

ฟังก์ชั่นจะแบ่งเป็นทางด้านสร้าง TCP Server และใช้ TCP Client โดยฟังก์ชั่นเริ่มต้นและฟังก์ชั่นยกเลิกการเชื่อมต่อจะต่างกัน แต่จะมีฟังก์ชั่นที่ใช้รับ-ส่งข้อมูลที่เหมือนกัน

2.4.2 ฟังก์ชั่นใช้สร้าง TCP Server

การสร้าง TCP Server จะต้องเรียกใช้ไลบารี่ WiFi.h ทุกครั้ง และการใช้งานจะต้องสร้างออปเจคของคลาส WiFiServer ขึ้นมานอกฟังก์ชั่นย่อย มีรูปแบบดังนี้

```
WiFiServer::WiFiServer(int port=80, int max_clients=4);
```

มีรายละเอียดของพารามิเตอร์ดังนี้

1. (int) port – พอร์ตที่ต้องการสร้าง TCP Server โดยกำหนดได้ในช่วง 1 – 65535 มีค่าเริ่มต้นเป็น 80 ซึ่งเป็นพอร์ตของโปรโตคอลประยุกต์ HTTP
2. (int) max_clients – จำนวน Client ที่เข้ามาเชื่อมต่อได้พร้อมกัน มีค่าเริ่มต้นเป็น 4

เริ่มต้นสร้าง TCP Server – จะใช้ฟังก์ชั่นย่อย .begin() มีรูปแบบการใช้งานดังนี้

```
void WiFiServer::begin();
```

ไม่มีค่าพารามิเตอร์ และไม่มีการตอบกลับ ฟังก์ชั่นย่อย .begin() จะต้องนำไปไว้ในฟังก์ชั่น setup ทุกครั้ง เพื่อให้เริ่มสร้าง TCP Server ขึ้นมาใช้งาน

ตรวจสอบการเชื่อมต่อเข้ามา – หลังจากใช้ฟังก์ชั่นย่อย .begin() ไปแล้ว จะต้องเริ่มตรวจสอบว่ามี Client เข้ามาเชื่อมต่อหรือไม่ ใช้ฟังก์ชั่นย่อย .available() ซึ่งมีรูปแบบการใช้งานดังนี้

```
WiFiClient WiFiServer::available();
```

ไม่มีค่าพารามิเตอร์ และตอบข้อมูลกลับมาในรูปของคลาส WiFiClient

ฟังก์ชั่นใช้งาน TCP Client

การใช้งาน TCP Client จะต้องเรียกใช้บารี่ WiFi.h ก่อนทุกครั้ง และฟังก์ชั่นย่อยบางส่วนที่ใช้ใน TCP Client จะถูกนำไปใช้งานใน TCP Server ด้วย การจะใช้งาน TCP Client ได้ จะต้องสร้างออปเจคของคลาส WiFiClient ขึ้นมาก่อนใช้งาน โดยสามารถประกาศสร้างไว้นอกฟังก์ชั่นย่อย หรือในฟังก์ชั่นย่อยได้แบบเดียวกับการประกาศตัวแปร ซึ่งการสร้างออปเจคของคลาส WiFiClient จะมีรูปแบบดังนี้

```
WiFiClient::WiFiClient()
```

ไม่มีค่าพารามิเตอร์

เชื่อมต่อกับ Server – ใช้ฟังก์ชั่นย่อย .connect() มีรูปแบบการใช้งานดังนี้

int WiFiClient::connect(IPAddress ip, int port);

หรือ

int WiFiClient::connect(const char *host, int port);

มีรายละเอียดของพารามิเตอร์ดังนี้

- (IPAddress) ip – หมายเลข IP ของ Server ที่ต้องการเชื่อมต่อ
- (const char *) host – ชื่อโดเมน หรือ hostname ที่ต้องการเชื่อมต่อ
- (int) port – พอร์ตที่ต้องการเชื่อมต่อ

และมีค่าที่ตอบกลับเป็นชนิด int หากมีค่ามากกว่า 0 จะหมายถึงเชื่อมต่อสำเร็จ

การตรวจสอบการเชื่อมต่อกับ Server – หลังจากเชื่อมต่อสำเร็จแล้ว ขั้นตอนต่อไปคือการสื่อสาร ในระหว่างการสื่อสารจะต้องมีการตรวจสอบตลอดเวลาว่ายังเชื่อมต่อกับ Server อยู่หรือไม่ เพื่อให้โปรแกรมที่ทำไว้ทำงานไม่ผิดพลาด ซึ่งจะใช้ฟังก์ชั่นย่อย .connected() ในการตรวจสอบ มีรูปแบบการใช้งานดังนี้

int WiFiClient::connected();

ไม่มีค่าพารามิเตอร์ และมีการตอบกลับเป็นข้อมูลชนิด int หากมีค่ามากกว่า 0 จะหมายถึงเชื่อมต่ออยู่

การรับ – ส่งข้อมูล – ฟังก์ชั่นที่รับ – ส่งข้อมูลจะรองรับฟังก์ชั่นเดียวกับการใช้งาน UART ซึ่งรองรับฟังก์ชั่นย่อย .available().readStringUntil() .readString() .read() .write() .print() .println() และฟังก์ชั่นอื่น ๆ อีกมากมาย

ล้างข้อมูลใน Buffer – เมื่อต้องการยกเลิกการเชื่อมต่อ จำเป็นที่จะต้องล้างข้อมูลที่จะยังไม่ได้อ่านออกมาเพื่อคืนพื้นที่ให้กับแรม ซึ่งใช้งานฟังก์ชั่นย่อย .flush(); ในการล้างค่า มีรูปแบบการใช้งานดังนี้

void WiFiClient::flush();

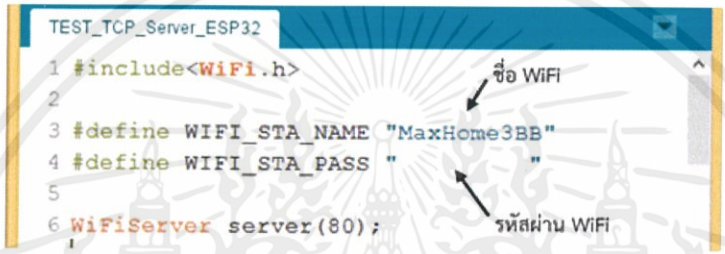
ไม่มีค่าพารามิเตอร์ และไม่มีค่าที่ตอบกลับ ยกเลิกการเชื่อมต่อกับ Server – ใช้ฟังก์ชั่นย่อย .stop() ในการยกเลิก มีรูปแบบการใช้งานดังนี้

void WiFiClient::stop();

ไม่มีค่าพารามิเตอร์ และไม่มีค่าที่ตอบกลับ

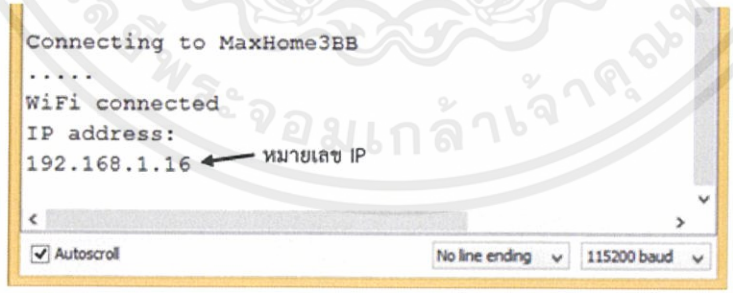
2.4.3 ทดลองสร้าง TCP Server

สามารถเข้าไปคัดลอกโปรแกรมตัวอย่างได้ที่ https://goo.gl/nRaJLS จากนั้นแก้ไข WIFI_STA_NAME และ WIFI_STA_PASS ให้ถูกต้อง



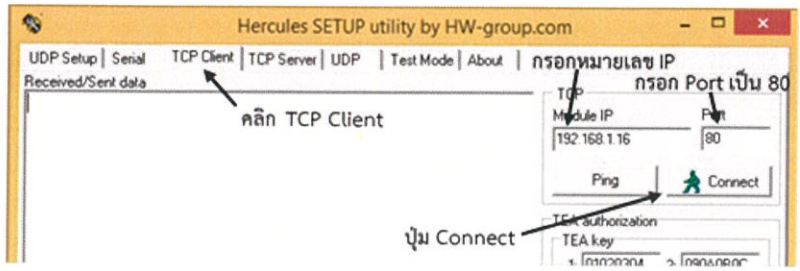
รูปที่ 2.12 ทดลองสร้าง TCP Server 1

เปิด Serial Monitor ขึ้นมาก่อน ปรับไปที่ 115200 band จากนั้นอัพโหลดโค้ดโปรแกรมลงบอร์ด NodeMCU-32S ได้เลย แล้วรอจนกว่า Serial Monitor แสดงหมายเลข IP ขึ้นมา



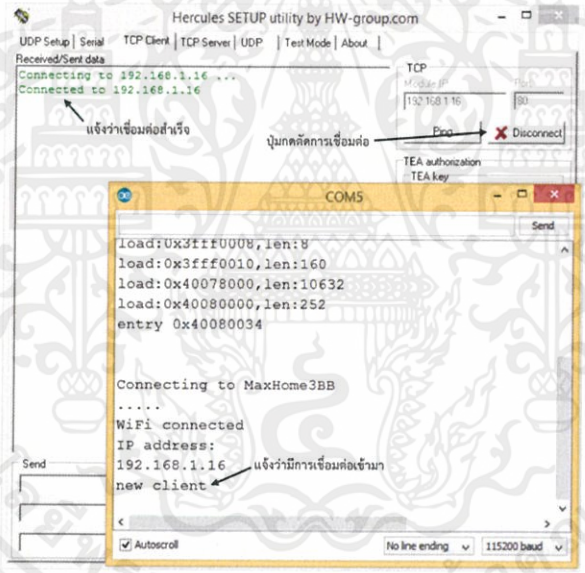
รูปที่ 2.13 ทดลองสร้าง TCP Server 2

ใช้คอมพิวเตอร์ที่อยู่วงแลนเดียวกับ ESP32 หรือเชื่อมต่อ WiFi ตัวเดียวกับ ESP32 เปิดโปรแกรม Hercules ขึ้นมา (ดาวน์โหลด ได้ที่ https://goo.gl/Y5p8ad) เมื่อเปิดขึ้นมาแล้วให้ไปที่ TCP Client แล้วกรอกช่อง Module IP เป็นหมายเลข IP ใน Serial Monitor จากนั้นกรอก Port เป็น 80 แล้วกด Connect



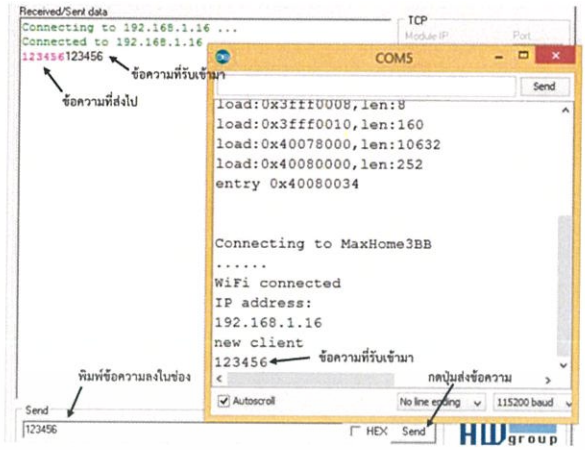
รูปที่ 2.14 ทดลองสร้าง TCP Server 3

เมื่อเชื่อมต่อสำเร็จ ที่โปรแกรม Hercules จะแสดงคำว่า Connected และในหน้าต่าง Serial Monitor จะแสดงคำว่า new client



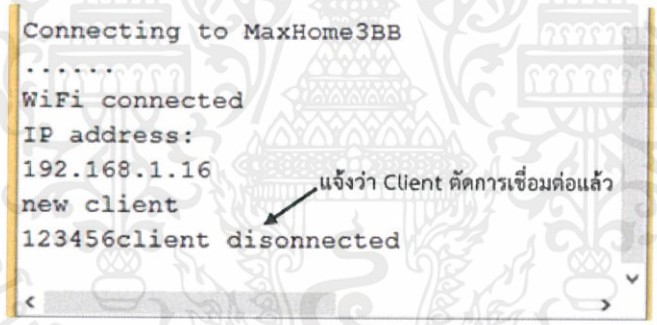
รูปที่ 2.15 ทดลองสร้าง TCP Server 4

ที่โปรแกรม Hercules ในส่วน Send ด้านล่าง เลือกกรอกข้อความลงในช่องและกด Send จะพบว่าข้อความแสดงที่ Serial Monitor และในโปรแกรม Hercules โดยตัวอักษรสีดำหมายถึงตัวอักษรที่ส่งข้อมูลไป และตัวอักษรสีชมพู คือตัวอักษรที่ส่งข้อมูลกลับมา



รูปที่ 2.16 ทดลองสร้าง TCP Server 5

เมื่อกดปุ่ม Disconnect ที่หน้าต่าง Serial Monitor จะแจ้งว่า Client ตัดการเชื่อมต่อแล้ว

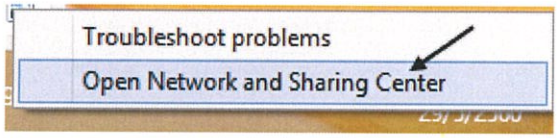


รูปที่ 2.17 ทดลองสร้าง TCP Server 6

2.4.4 ทดลองใช้ TCP Client

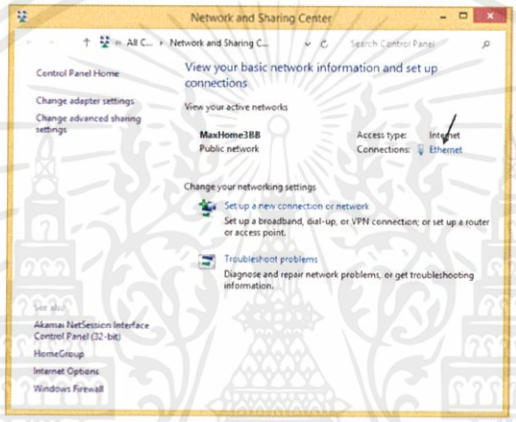
การทดลองใช้ TCP Client จะเป็นการปูพื้นฐานไปสู่การใช้งานโปรโตคอลประยุกต์อื่น ๆ ซึ่งล้วนต้องมีการเชื่อมต่อผ่าน TCP Client ก่อน แล้วจึงเริ่มมีการรับ – ส่งข้อมูลตามข้อกำหนดของแต่ละโปรโตคอล การทดลองนี้จะใช้เครื่องคอมพิวเตอร์ที่อยู่วงแลนเดียวกับ ESP32 เป็นเซิร์ฟเวอร์เพื่อให้ ESP32 ทดสอบเข้ามาเชื่อมต่อเพื่อสื่อสาร

ก่อนอื่นจะต้องมีการตรวจสอบหมายเลข IP ในวงแลนของเครื่องที่จะทำเป็นเซิร์ฟเวอร์เสียก่อน โดยไปที่กลุ่มไอค่อนเมนูด้านล่างขวา คลิกขวาที่ไอค่อน  หรือไอค่อน  แล้วเลือก Open Network and ....



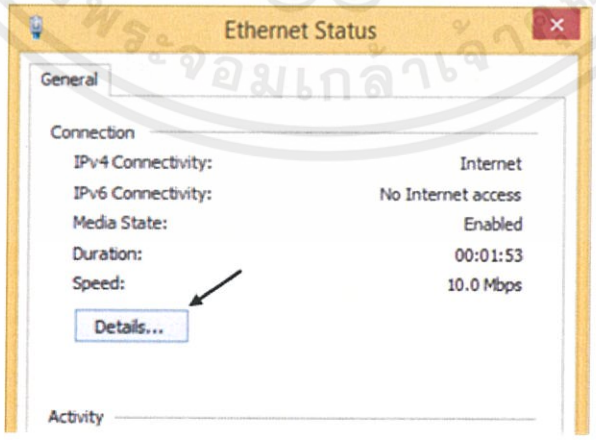
รูปที่ 2.18 ทดลองใช้ TCP Server 1

จากนั้นตรง Connections คลิกที่คำว่า Ethernet หรือ Wi-Fi



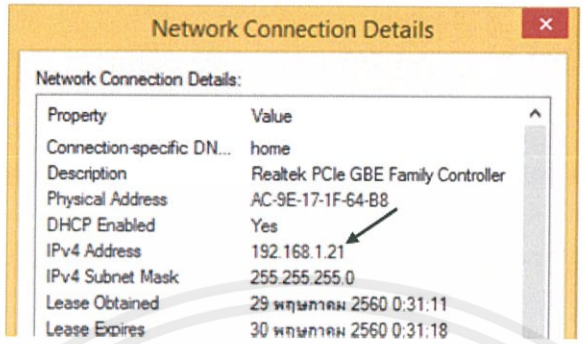
รูปที่ 2.19 ทดลองใช้ TCP Server 2

จากนั้นกดไปที่ปุ่ม Details...



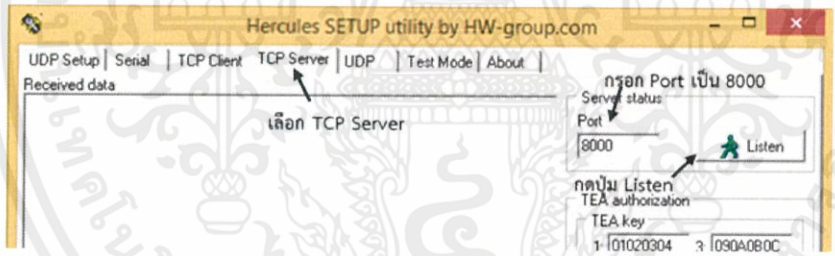
รูปที่ 2.20 ทดลองใช้ TCP Server 3

แล้วเก็บตรงส่วน IPv4 Address ไว้ก่อน เพื่อใช้นำไปแก้ไขในโค้ดโปรแกรมต่อไป



รูปที่ 2.221 ทดลองใช้ TCP Server 4

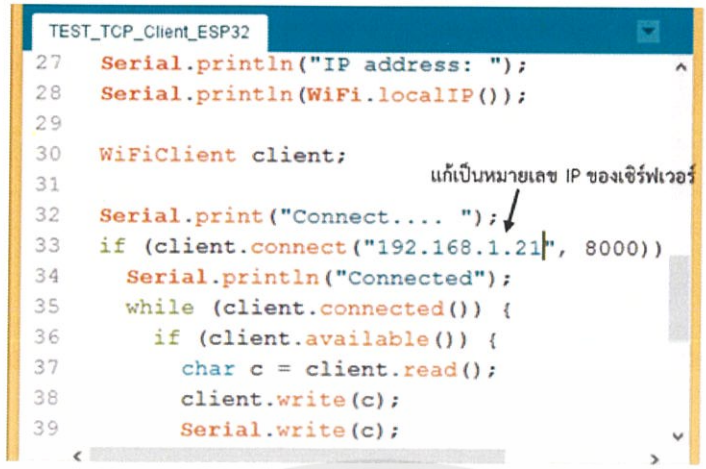
เปิดโปรแกรม Hercules ขึ้นมา จากนั้นกดไปที่ TCP Server ที่ช่อง Port กรอกเป็น 8000 แล้วกดปุ่ม Listen



รูปที่ 2.22 ทดลองใช้ TCP Server 5

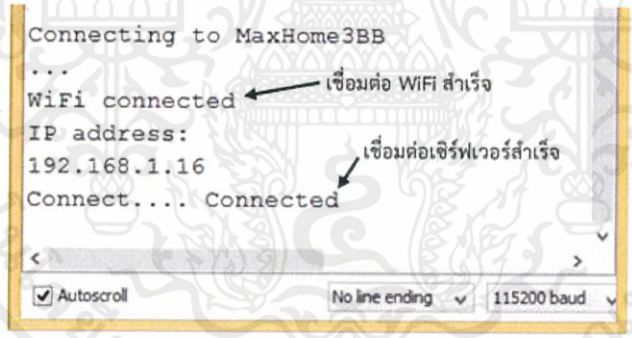
เพียงเท่านี้คอมพิวเตอร์ที่ได้ใช้ทดลองก็ได้ตั้ง TCP Server ขึ้นมาแล้วที่พอร์ต 8000 รอเพียง ESP32 เข้ามาเชื่อมต่อเท่านั้น

จากนั้นไปคัดลอกโค้ดโปรแกรมทดลองได้ที่ https://goo.gl/l8v4Ra นำมาแก้ไขตรงส่วนของฟังก์ชั่นย่อย .connect() โดยแก้พารามิเตอร์ host เป็นหมายเลข IP ที่ได้จากขั้นตอนที่แล้ว แก้ตรงส่วน WIFI_STA_NAME และ WIFI_STA_PASS เป็น WiFi ที่อยู่ในวงแลนวงเดียวกับเครื่องคอมพิวเตอร์ที่ทำเป็นเซิร์ฟเวอร์ แล้วอัพโหลดโค้ดโปรแกรมลงบอร์ด NodeMCU-32S ได้เลย



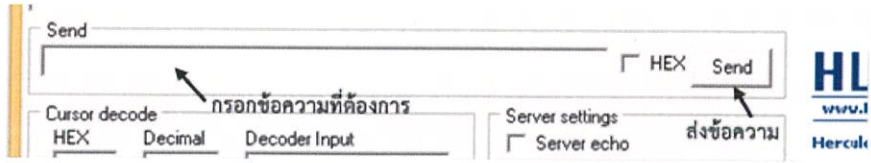
รูปที่ 2.23 ทดลองใช้ TCP Server 6

เมื่ออัพโหลดโปรแกรมเสร็จแล้ว ให้เปิด Serial Monitor ขึ้นมา ปรับเป็น 115200 band แล้วรอจนกว่าจะเชื่อมต่อ WiFi เสร็จ จนเชื่อมต่อกับเซิร์ฟเวอร์ที่สร้างไว้สำเร็จ



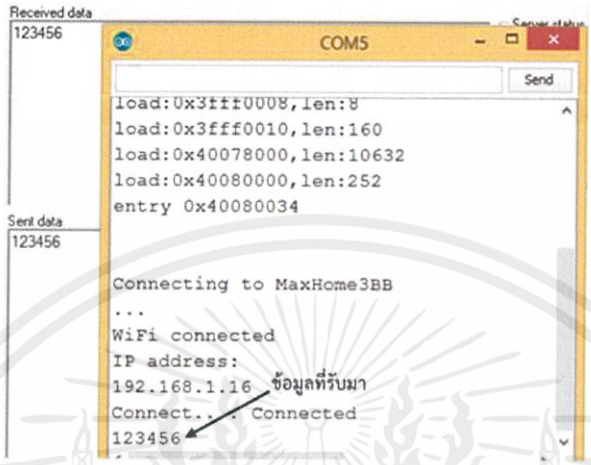
รูปที่ 2.24 ทดลองใช้ TCP Server 7

มาที่โปรแกรม Hercules สามารถทดลองส่งข้อความ โดยพิมพ์ในช่อง Send แล้วกดปุ่ม Send ได้เลย



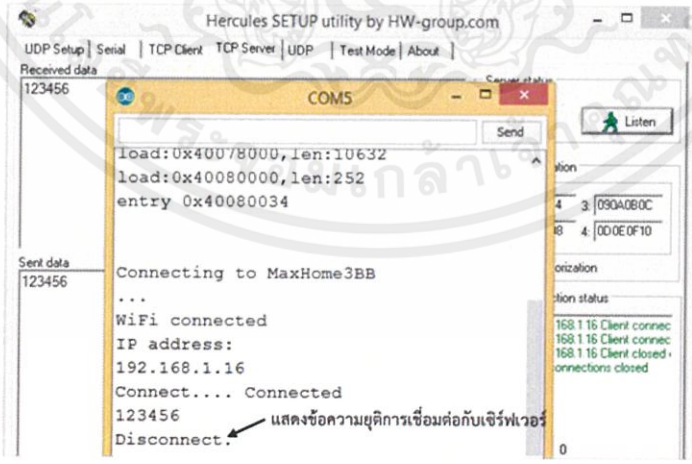
รูปที่ 2.25 ทดลองใช้ TCP Server 8

ที่ช่อง Sent data จะแสดงข้อมูลที่ส่งออกไป และช่อง Received data จะแสดงข้อมูลที่รับเข้ามา และที่ Serial Monitor จะแสดงข้อมูลที่ได้รับมา



รูปที่ 2.26 ทดลองใช้ TCP Server 9

เมื่อกดที่ปุ่ม Close จะเป็นการยุติการสร้าง TCP Server ที่หน้าต่าง Serial Monitor จะแสดงข้อความว่า Disconnect. เมื่อ ESP32 ยุติการเชื่อมต่อกับ Server

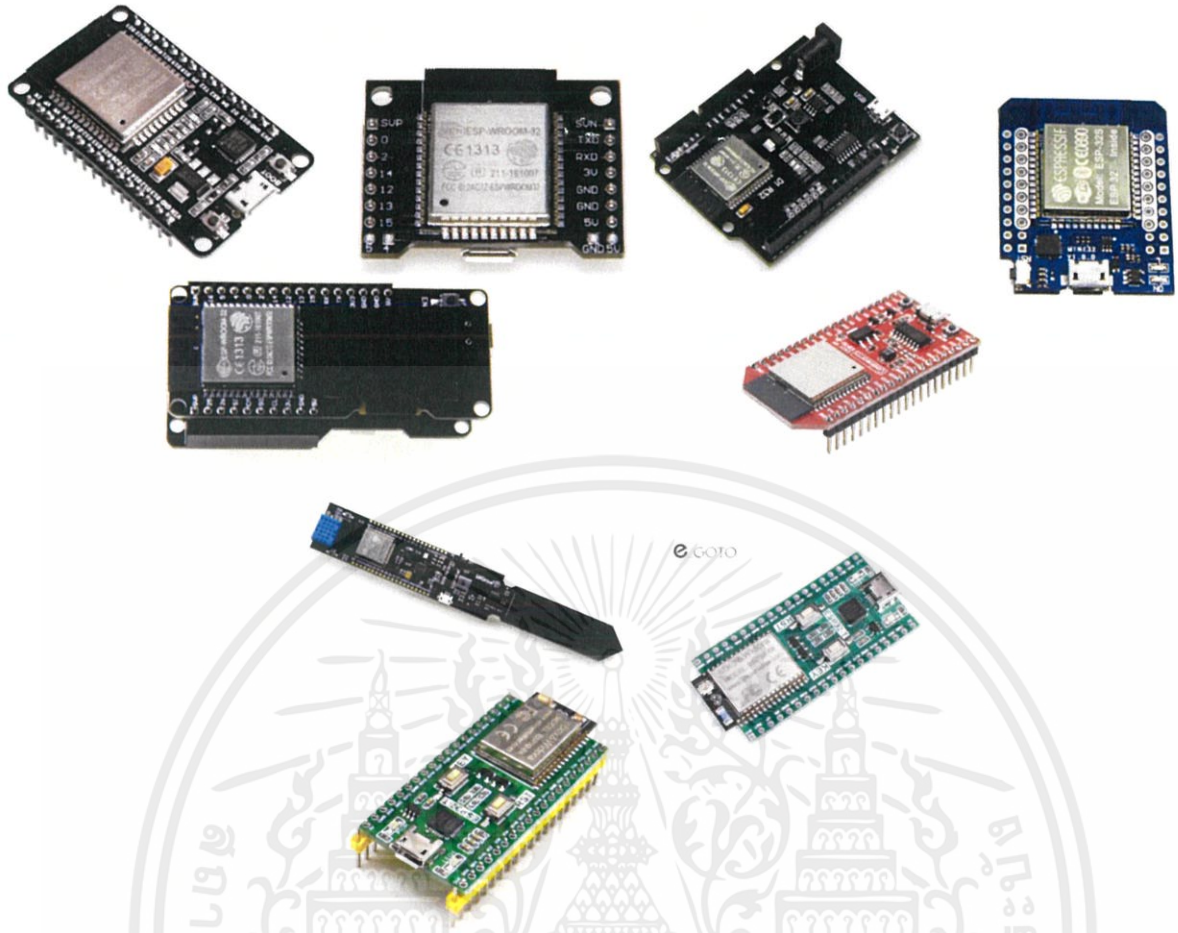


รูปที่ 2.27 ทดลองใช้ TCP Server 10

ซอฟต์แวร์ที่ใช้ในการพัฒนา

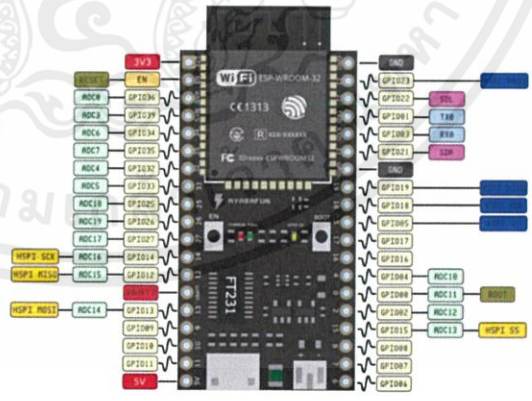
รูปที่ 2.28 ซอฟต์แวร์ที่ใช้ในการพัฒนา

- MicroPython-ESP32 รองรับการใช้งานพื้นฐานภาษา Python ส่วนใหญ่ได้ รองรับการจัดการ WiFi การเชื่อมต่อต่างๆ ของ ESP32
- Espruino on ESP32 ใช้ภาษา JavaScript ในการสั่งงานและรองรับการเขียนโปรแกรมแบบ Text
- ภาษาบล็อก (Block) และ LuaNode ก็รองรับคำสั่งที่ใช้บน Lua จริง ๆ แทบทุกคำสั่ง และรองรับการควบคุม WiFi ได้เต็มรูปแบบ



รูปที่ 2.29 บอร์ด ESP32 ทั่วไป

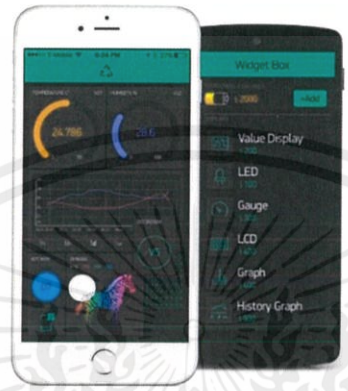
รูปที่ 2.30 WiFi LoRa 32



รูปที่ 2.31 ตำแหน่งขาของ ESP32

2.5 Blynk App

การสร้างต้นแบบโปรเจ็คในสมาร์ทโฟน ที่แบบลากวางปุ่ม ลากวางแถบเลื่อน ลากวางกราฟและวิดเจ็ต ในไม่กี่นาที เครื่องมือเหล่านี้สามารถควบคุมบอร์ด Arduino และรับข้อมูลจากแอพได้



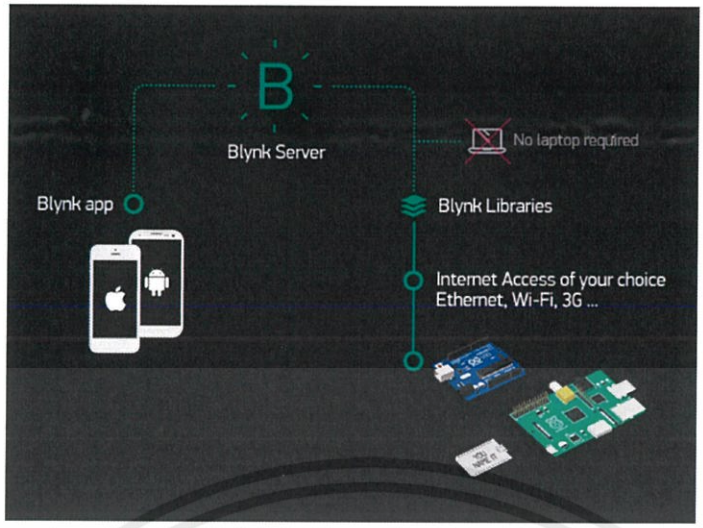
รูปที่ 2.32 ตัวอย่างแอพบน IOS และ Andriod

อุปกรณ์ที่ใช้งานร่วมกับ Blynk

Blynk ไม่ใช่แอปที่ทำงานเฉพาะกับอุปกรณ์ที่เฉพาะเจาะจง แต่ได้รับการออกแบบมาเพื่อสนับสนุนบอร์ดที่ทั่วไปมักใช้อยู่แล้ว เช่น Arduino NodeMCU ESP8266 และทำงานได้กับ iOS และ Android และอื่น ๆ อีกมามาย กดที่นี่ เพื่อดูบอร์ดที่รอบรับ

การทำงานของBlynk

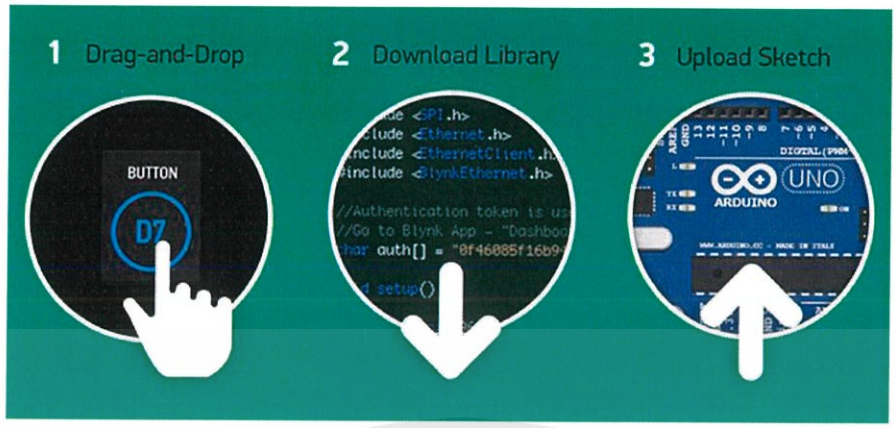
Blynk ทำงานผ่านทางอินเทอร์เน็ต ดังนั้นข้อกำหนดเพียงอย่างเดียวคืออุปกรณ์สามารถเชื่อมต่อกับอินเทอร์เน็ตได้ ไม่ว่าเราจะเลือกการเชื่อมต่อแบบใดก็ตาม Ethernet, Wi-Fi หรืออาจจะเป็น ESP8266 ไลบรารี่ของ Blynk และโค๊ดตัวอย่างจะช่วยให้ออนไลน์ โดยเชื่อมต่อกับ Blynk Server และจับคู่กับสมาร์ทโฟน



รูปที่ 2.33 สถาปัตยกรรมของ Blynk

ไม่ใช่เรื่องง่ายที่จะเอา Arduino เชื่อมต่อออกจากเครือข่ายภายในบ้าน ดังนั้นจึงได้สร้างเซิร์ฟเวอร์ Blynk ขึ้นมา โดยจะจัดการกับการรับรองความถูกต้องและการสื่อสารทั้งหมดและยังเก็บข้อมูลบนบอร์ดในขณะที่สมาร์ทโฟนออฟไลน์ เซิร์ฟเวอร์ Blynk ทำงานบน Java และเป็นโอเพนซอร์ส สามารถเรียกใช้งานได้ภายในเครื่องหากต้องการจริง ๆ การรับส่งข้อความระหว่างแอพพลิเคชั่นเคลื่อนที่ Blynk Server และ Arduino ใช้โปรโตคอลไบนารีที่ง่ายและมีขนาดเล็กและรวดเร็วผ่านซ็อกเก็ต TCP/IP

รูปที่ 2.34 ตัวอย่าง code ของ Blynk เมื่อเราอัพโหลด Arduino จะเชื่อมต่อกับเซิร์ฟเวอร์และจับคู่อุปกรณ์ของเราโดยอัตโนมัติ



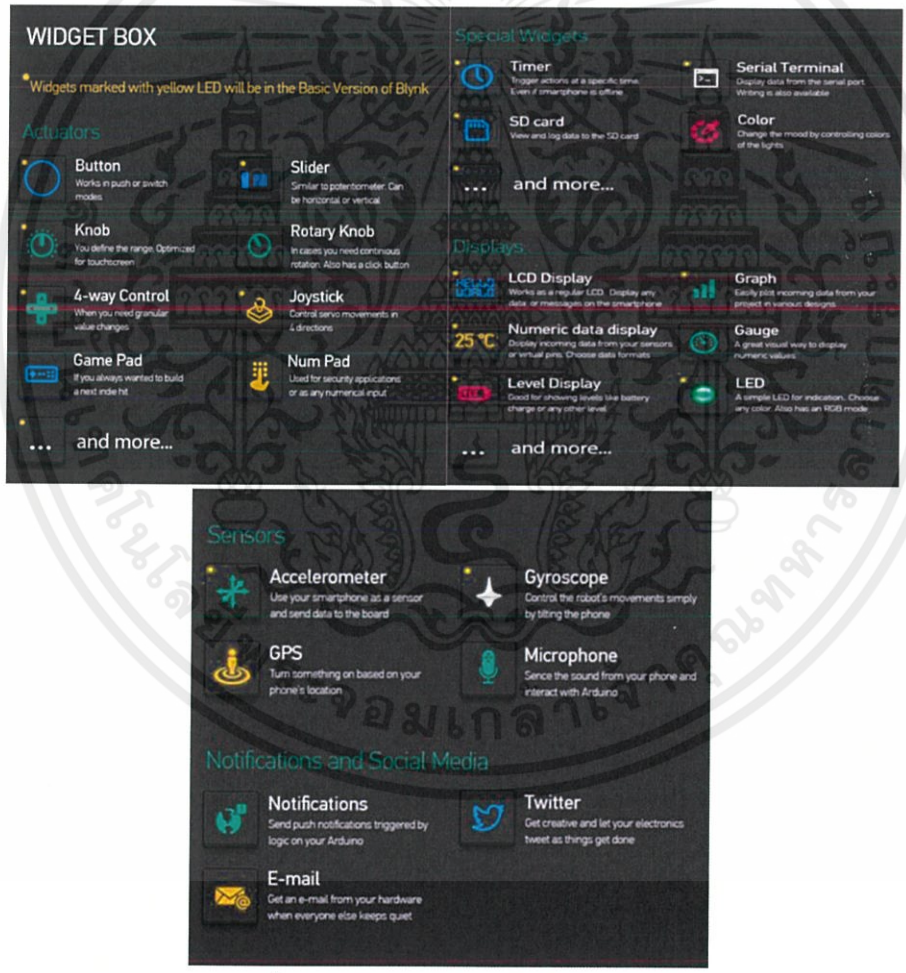
รูปที่ 2.35 ขั้นตอนการใช้งาน

2.5.1 ตัวอย่างการใช้งานของ Blynk

รูปที่ 2.36 มีอะไรเหลือในตู้เย็นบ้าง

รูปที่ 2.37 Phone Drone

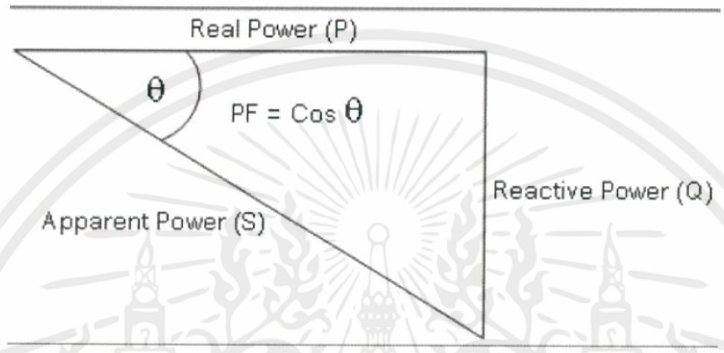
รูปที่ 2.38 การสื่อสารข้อมูลกับพืชสามารถสื่อสารข้อมูลกับพืช โรงเพาะชำ แปลงเกษตร เพื่อดูอุณหภูมิ ความชื้น รดน้ำ และดูและพวกมัน



รูปที่ 2.39 วิตเจ็ตต่างๆบนแอพพลิเคชัน

2.6 Power Factor

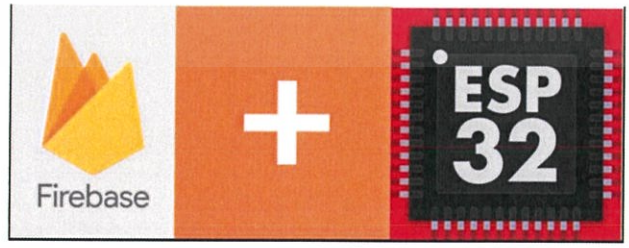
Power Factor หรือ ค่าตัวประกอบกำลัง คือตัวเลขที่บอกถึงกำลังงานไฟฟ้าที่เกิดการทำงานขึ้นจริง P สามารถใช้ประโยชน์ได้จริงๆเทียบกับขนาดของกำลังงานทั้งหมดที่มีอยู่ในระบบไฟฟ้านั้นS ซึ่งกำลังงานส่วนเกินที่ไม่สามารถนำมาใช้งานได้จริง ก็คือ Q สามารถอธิบายเป็นรูปสามเหลี่ยมพลังงานได้ดังนี้



รูปที่ 2.40 รูปสามเหลี่ยมพลังงาน

กำลังงานไฟฟ้าที่ใช้งานได้จริง (Real Power) ใช้สัญลักษณ์ “ P ” มีหน่วยเป็น วัตต์ (Watt : W) กำลังงานที่ปรากฏ (Apparent Power) ใช้สัญลักษณ์ “ S ” มีหน่วยเป็น วีเอ หรือโวลท์-แอมป์ (VA) กำลังงานสูญเสีย หรือส่วนเกิน ที่ไม่สามารถนำไปใช้ประโยชน์ได้ (Reactive Power) ใช้สัญลักษณ์ “ Q ” มีหน่วยเป็นวาร์(VAR)

2.7 Firebase



รูปที่ 2.41 การเชื่อมต่อ ข้อมูล ระหว่าง ESP32 กับ FIREBASE

Firebase มีบริการหลักเป็น Realtime Database เกิดขึ้นด้วยแนวคิดที่คนทำแอพพลิเคชั่นไม่จำเป็นต้องตั้งเซิฟเวอร์เอง และไม่ต้องต้องเขียนโปรแกรมหลังบ้านซ้ำ ๆ แบบเดิม ซึ่งหากคนที่ทำเว็บไซต์ ทำแอพพลิเคชั่น จะทราบดีว่างาน 1 โปรเจค จะต้องมีฐานข้อมูล และจะต้องมีการเก็บตารางของผู้ใช้งาน ระบบ Log ต่าง ๆ มีการติดต่อกับผู้ใช้ ซึ่งเป็นงานที่มีการทำซ้ำ ๆ ตลอดมา ดังนั้น Firebase จึงมาช่วยแก้ปัญหาตรงนี้ได้ ทำให้ไม่ต้องมีการจัดการฐานข้อมูลเอง ไม่ต้องเขียนโปรแกรมหลังบ้านเอง (ด้วยภาษา PHP Python และอื่นๆ) ตัว Firebase ทำไว้ให้หมดแล้ว ในงานด้านแอพพลิเคชั่น ตัว Firebase ถือเป็นบริการฐานข้อมูลออนไลน์ตัวหนึ่ง ซึ่งแอพพลิเคชั่นส่วนใหญ่ต้องใช้งานฐานข้อมูลตรงส่วนนี้ แต่หากมองในมุมของ IoT ตัว Firebase ถือว่าเป็นตัวกลางการเชื่อมต่อทุกอุปกรณ์เข้าด้วยกันได้ โดยมีจุดเด่นคือ เรียลไทม์ และสามารถบันทึกข้อมูลไว้ได้ ในด้านของ API ตัว Firebase ไม่ได้อิงการใช้งานไปกับภาษาใดภาษาหนึ่ง กรณีที่ภาษาใด ๆ ไม่มีไลบารี่ให้ใช้งาน สามารถใช้ REST API (โปรโตคอล HTTP, HTTPs) ในการร้องขอข้อมูล (GET) หรือส่งข้อมูล (PUT) เข้าไปได้เลย

Firebase คือฐานข้อมูลประเภท NoSQL

ฐานข้อมูล MySQL MSSQL และฐานข้อมูลชนิด RDBMS ต่าง ๆ จะมีลักษณะเป็นตารางข้อมูล มีคอลั่ม มีการกำหนดชนิดของข้อมูลไว้อย่างชัดเจน และใช้ภาษา SQL ในการติดต่อเพื่อขอใช้ข้อมูล (SELECT) เพิ่มข้อมูล (INSERT) และลบข้อมูล (DELETE) สามารถกรองเอาเฉพาะข้อมูลที่ต้องการได้ด้วยการใช้ WHERE และบางครั้งมีปัญหาเรื่องช่องโหว่ (SQL Injection ถือเป็นวิธีพื้นฐานที่นิยมใช้และได้ผลมากที่สุดในขณะนี้)

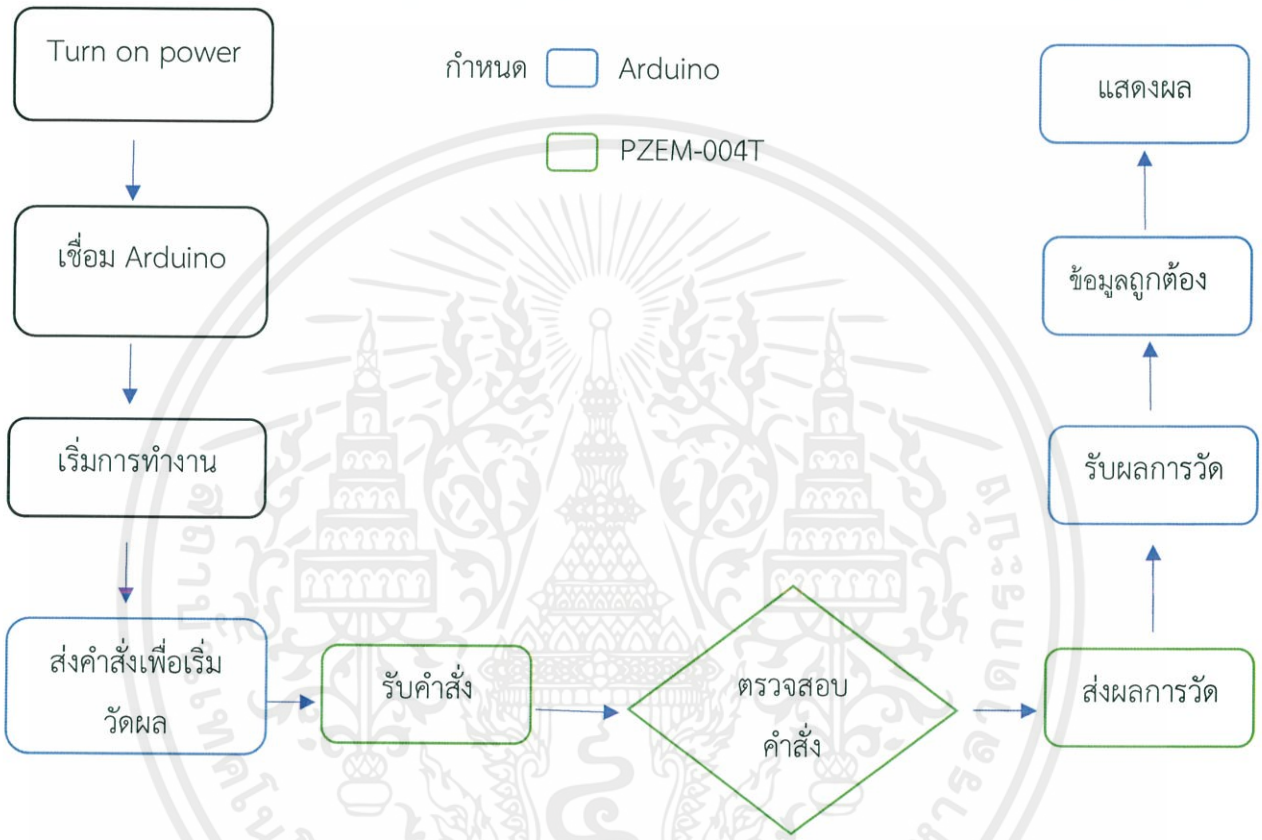
ฐานข้อมูลชนิด NoSQL จะไม่ใช้ภาษา SQL ในการจัดการข้อมูล และออกแบบให้มีความยืดหยุ่นและเน้นความเร็วในการใช้งานมากที่สุด ฐานข้อมูล NoSQL ที่นิยมใช้งานในปัจจุบันคือ MongoDB ซึ่งมีการเก็บข้อมูลเป็นชนิด JSON (เจสัน) มีตารางเหมือนเดิม แต่ไม่มีคอลั่มข้อมูลที่ตายตัว ใน 1 แถว สามารถเก็บข้อมูลได้ทั้งข้อความ (String) ตัวเลข (Number) และอื่น ๆ รวมไปถึงอาเรย์ และออปเจ็ค

Firebase มีการทำงานคล้าย ๆ กับ MongoDB คือมีฐานข้อมูล แต่ไม่มีตาราง มีการเก็บข้อมูลในรูป JSON สามารถเพิ่มข้อมูลไปในออปเจ็คใด ๆ ก็ได้ แต่เก็บเป็นอาร์เรย์ไม่ได้ ถ้าต้องการเพิ่มข้อมูลแบบอาร์เรย์จะต้องใช้การ PUT ข้อมูลเข้าไปต่อท้ายเรื่อย ๆ ซึ่งจะมี Key ที่ Firebase สร้างให้เป็นตัวอ้างอิง

บทที่ 3

การออกแบบ

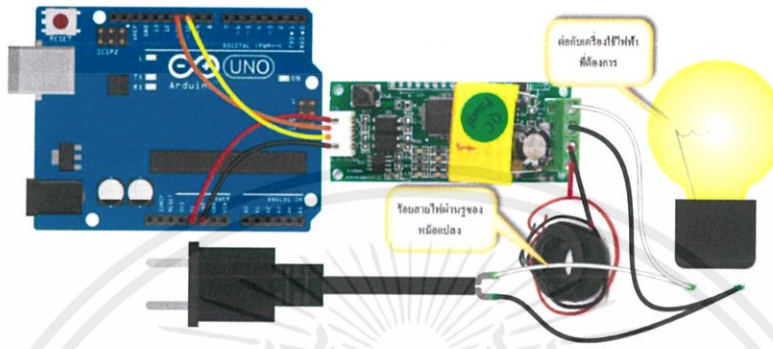
3.1 ขั้นตอนการดำเนินงาน



รูปที่ 3.1 ขั้นตอนการทำงานของระบบวัดพลังงานไฟฟ้าและแสดงผล

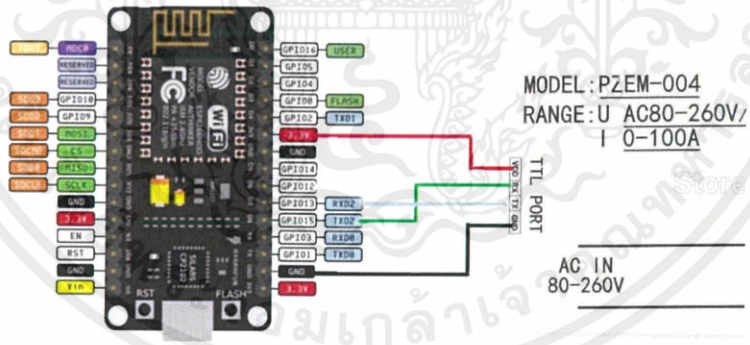
3.2 ออกแบบการทดลอง

3.2.1 ทดลอง PZEM ร่วมกับ Arduino และ หลอดไฟ พร้อมเช็คผลผ่าน Serial Monitor บน Arduino IDE



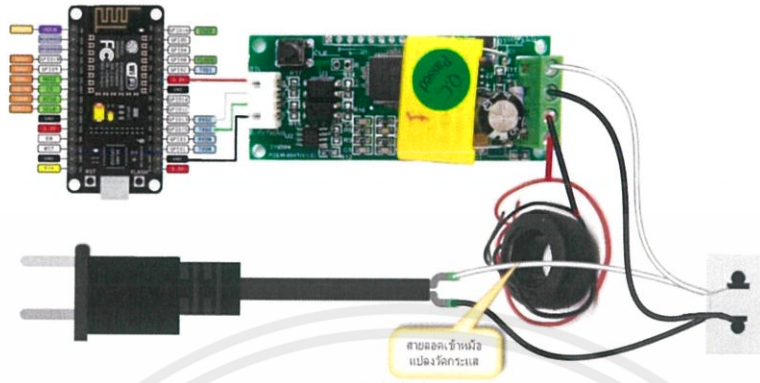
รูปที่ 3.2 บล็อกไดอะแกรมของการต่อวงจรกับArduino

3.2.2 ทดลอง PZEM ร่วมกับ ESP32 และ หลอดไฟ พร้อมเช็คผลผ่าน Serial Monitor บน Arduino IDE และ แอพพลิเคชัน Blynk



รูปที่ 3.3 บล็อกไดอะแกรมของการต่อวงจรกับESP32

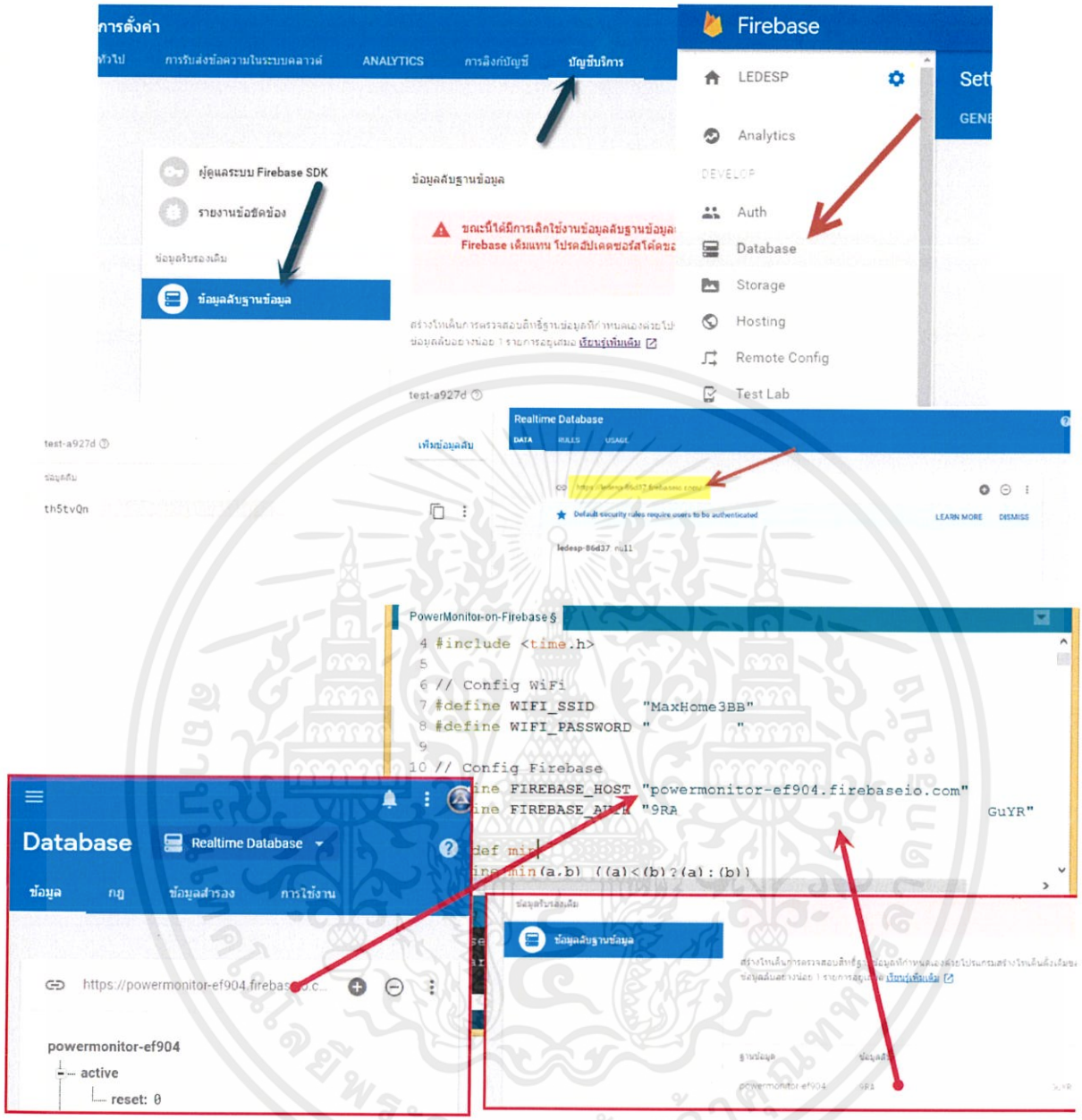
3.2.3 ทดลอง PZEM ร่วมกับ ESP32 และ เต้ารับโดยเปลี่ยนโหลดเป็นอย่างอื่น พร้อมเช็คผลผ่าน Serial Monitor บน Arduino IDE และ แอพพลิเคชัน Blynk



รูปที่ 3.4 บล็อกไดอะแกรมของการต่อวงจรร่วมกับ ESP32 และ เต้ารับ

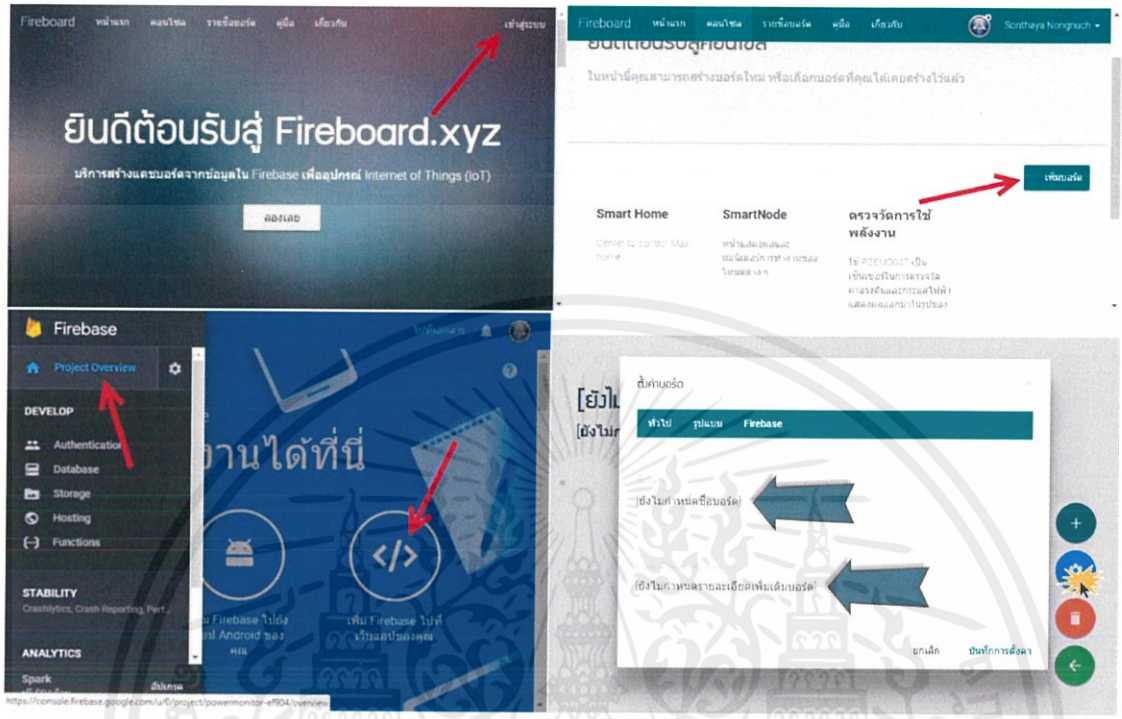
3.2.4 ใช้งาน Firebase ระบบฐานข้อมูลเรียลไทม์จาก Google

รูปที่ 3.5 การสร้างฐานข้อมูลบน Firebase

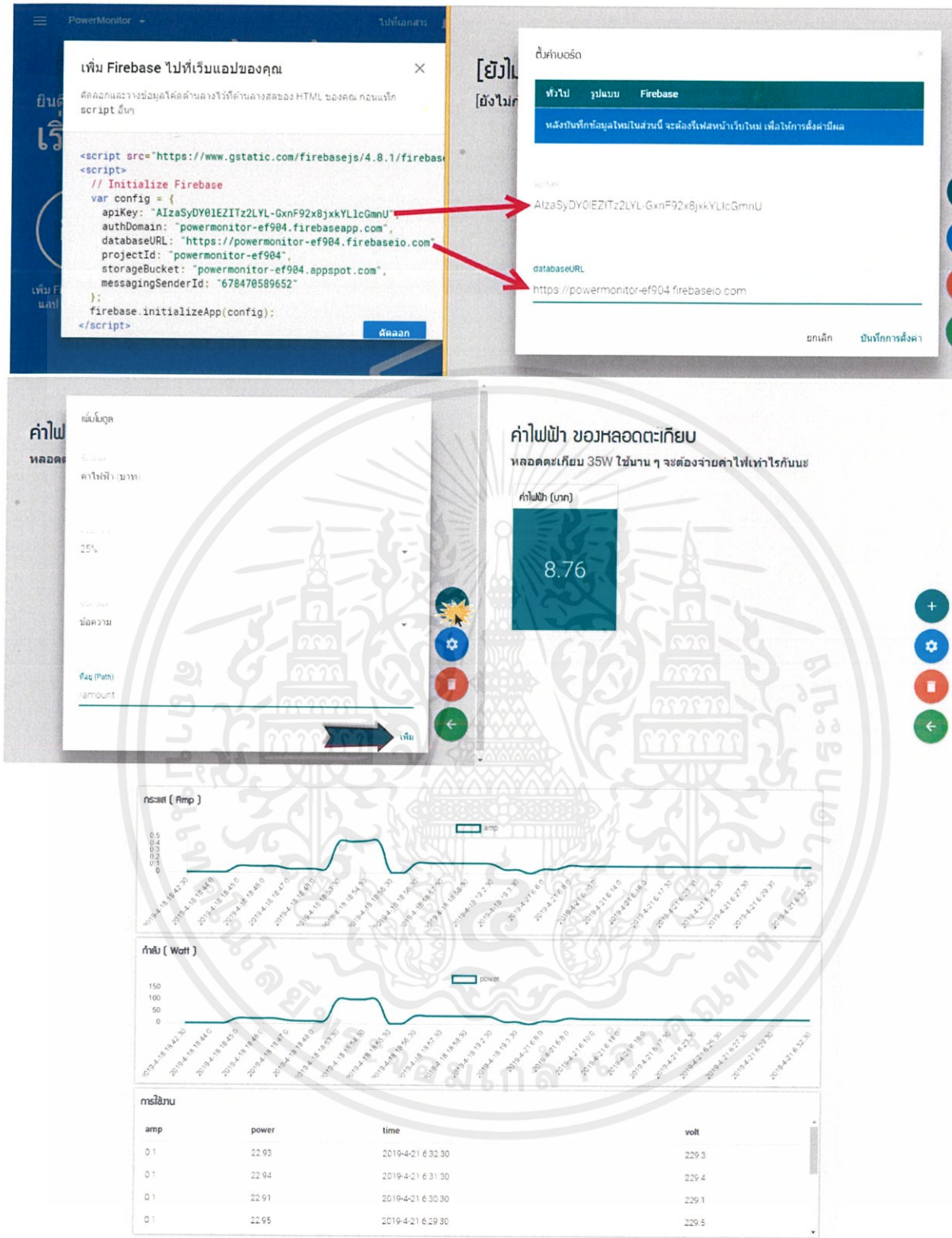


รูปที่ 3.6 ขั้นตอนการนำข้อมูลจากโปรแกรมไปขึ้นฐานข้อมูล Firebase

3.2.5 นำข้อมูลจาก ESP ขึ้นแสดงผลผ่านเว็บไซต์



รูปที่ 3.7 ขั้นตอนการนำข้อมูลจากเว็บ Firebase มาแสดงผล



รูปที่ 3.8 ขั้นตอนการนำข้อมูลจากเว็บ Firebase มาแสดงผล (2)

บทที่ 4

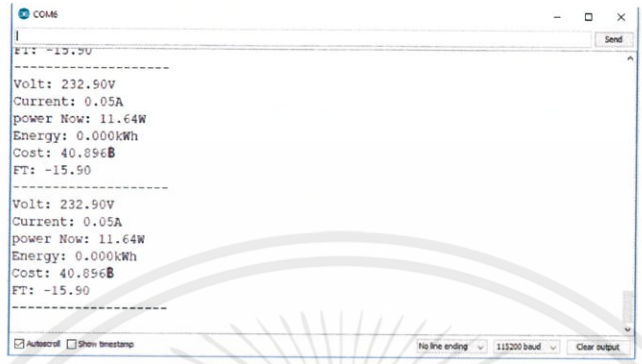
ผลการทดลอง

4.1 ทดลอง PZEM ร่วมกับ Arduino และ หลอดไฟ เช็คผลผ่าน Serial Monitor Arduino IDE

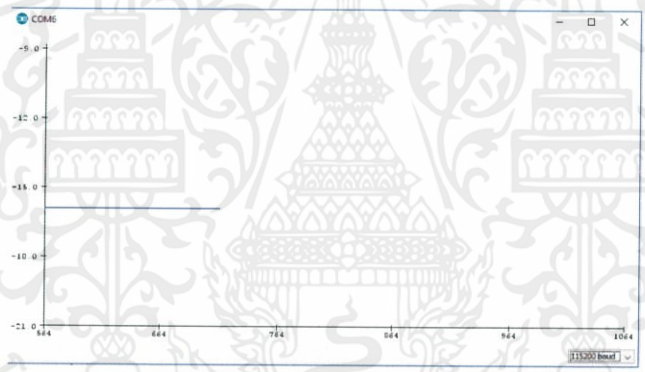
รูปที่ 4.1 การแสดงผลผ่าน Serial moniter 1

รูปที่ 4.2 การแสดงผลค่า FT กับ ปริมาณการใช้ไฟ ในรูปแบบกราฟ 1

4.2 ทดลอง PZEM ร่วมกับ ESP32 และ หลอดไฟ พร้อมเช็คผลผ่าน Serial Monitor บน Arduino IDE และ แอพพลิเคชัน Blynk



รูปที่ 4.3 การแสดงผลผ่าน Serial moniter 2

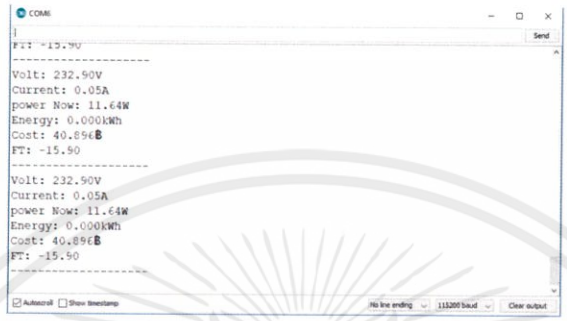


รูปที่ 4.4 การแสดงผลค่า FT กับ ปริมาณการใช้ไฟ ในรูปแบบกราฟ 2

รูปที่ 4.5 การแสดงผลต่างๆบนแอพพลิเคชัน Blynk

4.3 ทดลอง PZEM ร่วมกับ ESP32 และ เต้ารับโดยเปลี่ยนโหลดเป็นอย่างอื่น พร้อมเช็คผลผ่าน Serial Monitor บน Arduino IDE และ แอพพลิเคชัน Blynk

4.3.1 หลอดไฟ

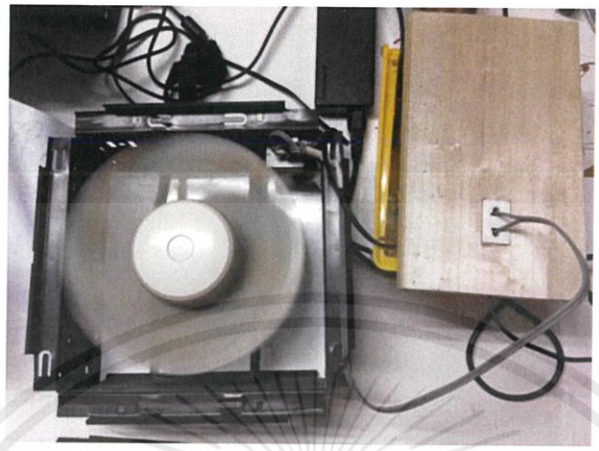


รูปที่ 4.6 การแสดงผลผ่าน Serial moniter 3

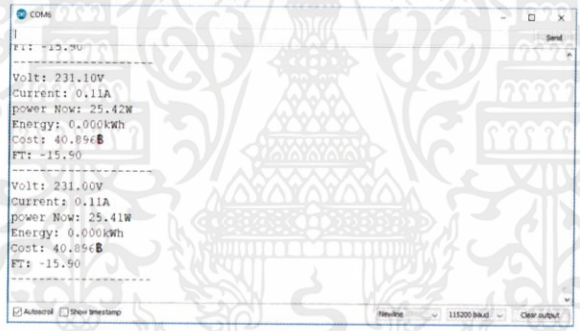
รูปที่ 4.7 การแสดงผลค่า FT กับ ปริมาณการใช้ไฟ ในรูปแบบกราฟ 3

รูปที่ 4.8 การแสดงผลต่างๆบนแอพพลิเคชัน Blynk 2

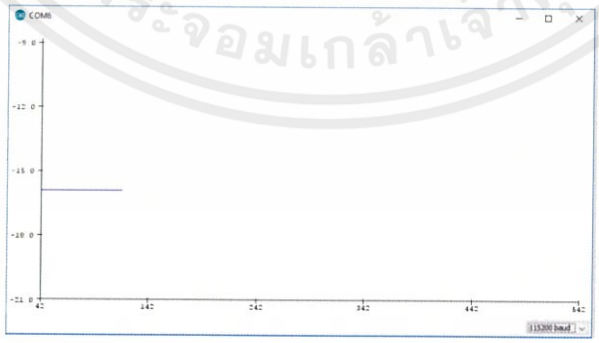
4.3.2 พัดลม



รูปที่ 4.9 โหลดพัดลม



รูปที่ 4.10 การแสดงผลผ่าน Serial moniter 4

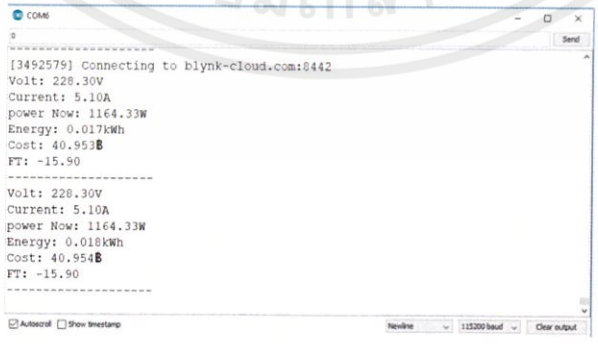


รูปที่ 4.11 การแสดงผลค่า FT กับ ปริมาณการใช้ไฟ ในรูปแบบกราฟ 4

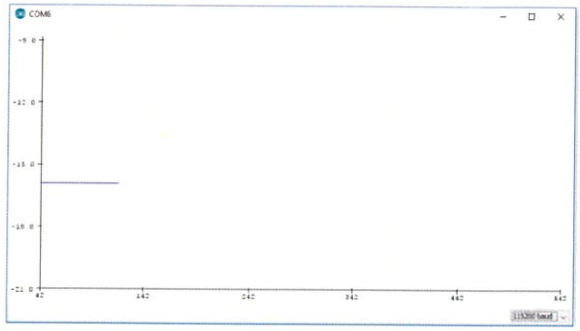
รูปที่ 4.12 การแสดงผลต่างๆบนแอพพลิเคชัน Blynk 3

4.3.3 เตารีด

รูปที่ 4.13 โหลดเตารีด



รูปที่ 4.14 การแสดงผลผ่าน Serial moniter 5

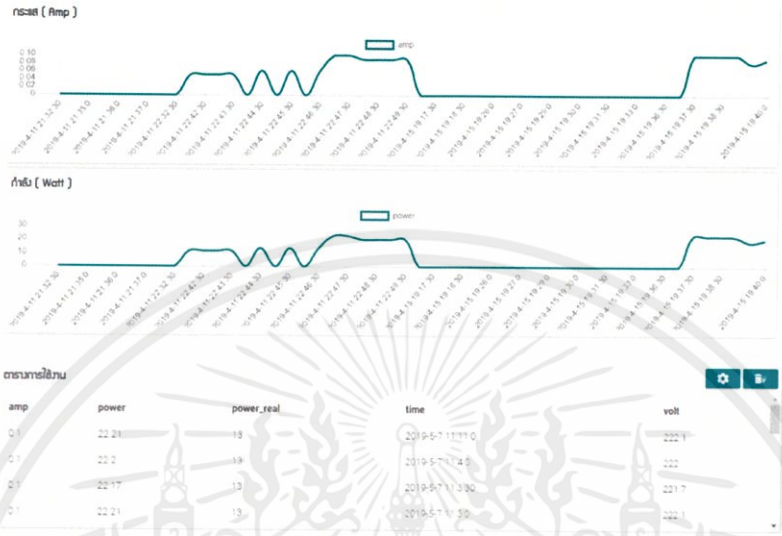


รูปที่ 4.15 การแสดงผลค่า FT กับ ปริมาณการใช้ไฟ ในรูปแบบกราฟ 5

รูปที่ 4.16 การแสดงผลต่างๆบนแอพพลิเคชัน Blynk4

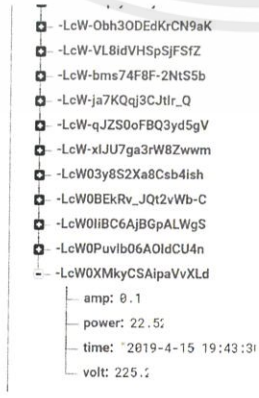
4.4 ทดลอง PZEM ร่วมกับ ESP32 และ เต้ารับโดยเปลี่ยนโหลดเป็นอย่างอื่น พร้อมเช็คผลผ่าน Fireborad

4.4.1 แท็บเลต (Ipad)



รูปที่ 4.17 การแสดงผลผ่าน Fireborad

รูปที่ 4.18 แสดงการเก็บไฟล์ข้อมูลบน Firebase



รูปที่ 4.19 แสดงข้อมูลไฟล์ที่ถูกบันทึกค่า ตาม เวลาที่กำหนดบน Firebase

4.4.2 โทรศัพท์มือถือ (Iphone)

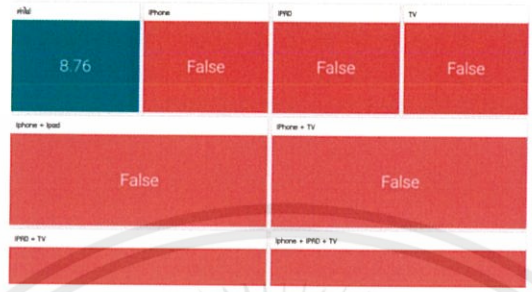
รูปที่ 4.20 การแสดงผลผ่าน Fireborad 2

รูปที่ 4.21 แสดงการเก็บไฟล์ข้อมูลบน Firebase 2

รูปที่ 4.22 แสดงข้อมูลไฟล์ที่ถูกบันทึกค่า ตาม เวลาที่กำหนดบน Firebase 2

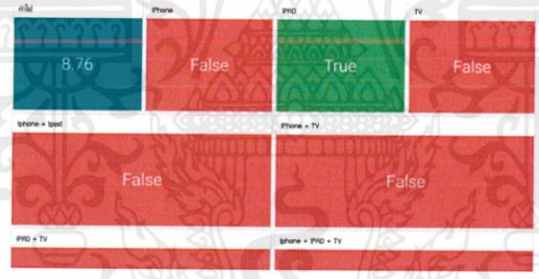
4.5 การเช็คสถานะ ON OFF ของโหลดเมื่อโหลดคือ I pad และ I phone

4.5.1 เมื่อไม่มีโหลด



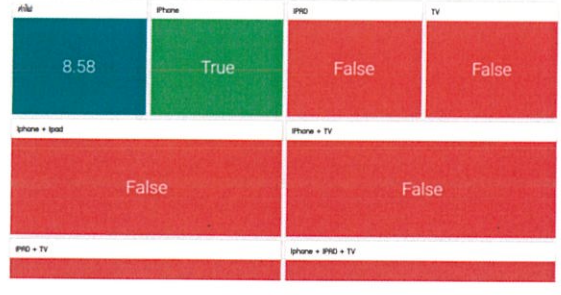
รูปที่ 4.23 สถานะเมื่อไม่มีโหลด

4.5.2 Ipad ชนิดเดียว



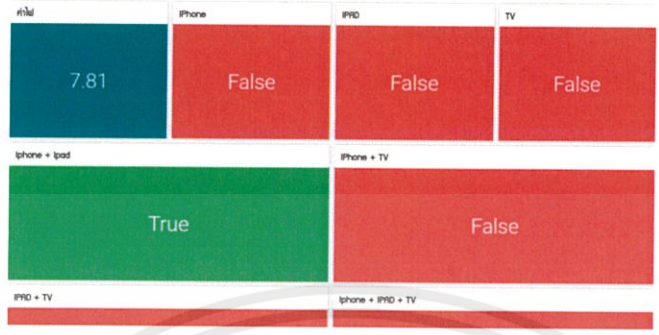
รูปที่ 4.24 สถานะเมื่อมีโหลดชนิดเดียวคือ ipad

4.5.3 Iphone ชนิดเดียว



รูปที่ 4.25 สถานะเมื่อมีโหลดชนิดเดียวคือ iphone

4.5.4 Ipad และ Iphone



รูปที่ 4.26 สถานะเมื่อมีโหลดสองชนิด คือ ipad และ iphone

4.5.5 การทดลองการวัดผลของหลอดไฟ

คุณสมบัติที่ทางบริษัทกำหนด

- Power maximum output 12 W
- Voltage maximum Input 240 V

รูปที่ 4.27 หลอดไฟที่ใช้ในการทดลอง

ตารางที่ 4.2 การรับส่งข้อมูลระหว่าง Arduino และ PZEM-004T วัดการใช้ไฟฟ้าของหลอดไฟ

เวลา(วินาที)	Volt	Current	Energy	Power
1	228.20	0.06	0	13.69
2	228.20	0.06	0	13.69
3	228.20	0.06	0	13.69
4	228.20	0.06	0	13.69

จากการทดลองพบว่าค่า Power ที่วัดได้ ห่างจากคุณสมบัติที่ให้มาคลาดเคลื่อน 8.3%

4.5.6 การทดลองการวัดผลของหลอดไฟ

ตารางที่ 4.3 การรับส่งข้อมูลระหว่าง Esp32 และ PZEM-004T วัดการใช้ไฟฟ้าของหลอดไฟ

เวลา(วินาที)	volt	Current	Energy	Power
1	232.90	0.05	0	11.64
2	232.90	0.05	0	11.64
3	232.90	0.05	0	11.64
4	232.90	0.05	0	11.64

จากการทดลองพบว่าค่า Power ที่วัดได้ ห่างจากคุณสมบัติที่ให้มาคลาดเคลื่อน 3%

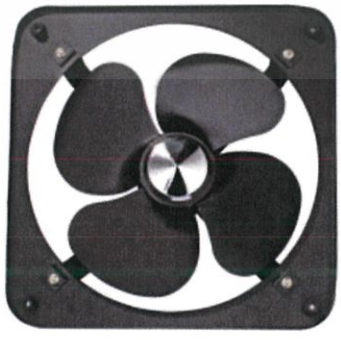
4.5.7 การทดลองการวัดผลของหลอดไฟ

ตารางที่ 4.4 การรับส่งข้อมูลระหว่าง Esp32 และ PZEM-004T วัดการใช้ไฟฟ้าของหลอดไฟ โดย ทำการต่อเต้ารับ ลงในชุดอุปกรณ์

เวลา(วินาที)	volt	Current	Energy	Power
1	232.90	0.05	0	11.64
2	232.90	0.05	0	11.64
3	232.90	0.05	0	11.64
4	232.90	0.05	0	11.64

จากการทดลองพบว่าค่า Power ที่วัดได้ ห่างจากคุณสมบัติที่ให้มาคลาดเคลื่อน 3%

4.5.8 การทดลองการวัดผลของพัดลม



คุณสมบัติที่ทางบริษัทกำหนด

- Power maximum output 28 W
- Voltage maximum Input 240 V

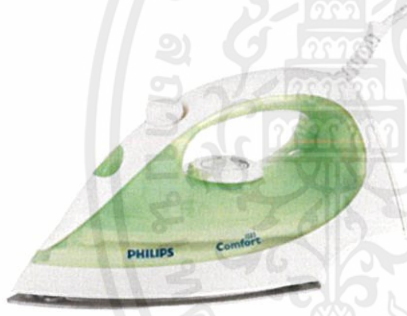
รูปที่ 4.28 พัดลมรุ่น FAD20-4A ที่ใช้ในการทดลอง

ตารางที่ 4.5 การรับส่งข้อมูลระหว่าง Esp32 และ PZEM-004T วัดการใช้ไฟฟ้าของพัดลม โดย ทำการต่อเต้ารับ ลงในชุดอุปกรณ์

เวลา(วินาที)	volt	Current	Energy	Power
1	229.5	5.15	0	25.42
2	229.5	5.15	0	25.41
3	229.5	5.15	0	25.42
4	229.5	5.15	0	25.41

จากการทดลองพบว่าค่า Power ที่วัดได้ ห่างจากคุณสมบัติที่ให้มาคลาดเคลื่อน 9.3%

4.5.9 การทดลองการวัดผลของอุปกรณ์เตารีด



คุณสมบัติที่ทางบริษัทกำหนด

- Power maximum output 1200 W
- Current maximum output 5.45s A
- Voltage maximum Input 240 V

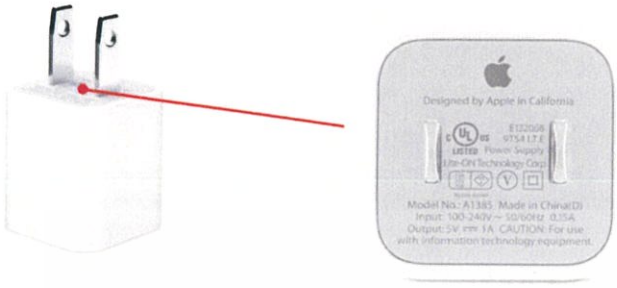
รูปที่ 4.29 เตารีดรุ่น GC1010/01 ที่ใช้ในการทดลอง

ตารางที่ 4.6 การรับส่งข้อมูลระหว่าง Esp32 และ PZEM-004T วัดการใช้ไฟฟ้าของเตารีด โดย ทำการต่อเต้ารับ ลงในชุดอุปกรณ์

เวลา(วินาที)	volt	Current	Energy	Power
1	228.30	5.05	18	1164.33
2	228.30	5.05	18	1164.33
3	228.30	5.05	18	1164.33
4	228.30	5.05	18	1164.33

จากการทดลองพบว่าค่า Power ที่วัดได้ ห่างจากคุณสมบัติที่ให้มาคลาดเคลื่อน 3%

4.5.10 การทดลองการวัดผลของอุปกรณ์ชาร์จโทรศัพท์



คุณสมบัติที่ทางบริษัทกำหนด

- Power maximum output 5 W
- Current maximum output 1 A
- Voltage maximum Input 240 V

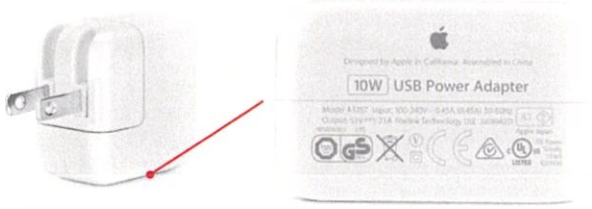
รูปที่ 4.30 อุปกรณ์ชาร์จโทรศัพท์

ตารางที่ 4.7 การแสดงผลข้อมูลผ่าน Fire base วัดการใช้ไฟฟ้าของ โทรศัพท์มือถือ (Iphone)

เวลา	volt	Current	Power	Real Power
19:47:30	225.60	0.05	11.28	6
19:48:00	225.10	0.05	11.25	6
19:48:30	225.20	0.05	11.26	6
19:49:00	225.20	0.05	11.26	6

จากการทดลองพบว่าค่า Power ที่วัดได้ ห่างจากคุณสมบัติที่ให้มาคลาดเคลื่อน 20%

4.5.11 การทดลองการวัดผลของอุปกรณ์ชาร์จแท็บเลต



คุณสมบัติที่ทางบริษัทกำหนด

- Power maximum output 12 W
- Current maximum output 0.54 A
- Voltage maximum Input 240 V

รูปที่ 4.31 อุปกรณ์ชาร์จแท็บเลต

ตารางที่ 4.8 การแสดงผลข้อมูลผ่าน Fire base วัดการใช้ไฟฟ้าของ แท็บเลต (Ipad)

เวลา	volt	Current	Power	Real Power
19:40:00	225.60	0.09	20.304	13
19:40:30	225.40	0.08	18.032	13
19:41:00	225.50	0.10	22.550	13
19:41:30	225.40	0.09	20.286	13

จากการทดลองพบว่าค่า Power ที่วัดได้ ห่างจากคุณสมบัติที่ให้มาคลาดเคลื่อน 8.3%

บทที่ 5

สรุปผลการทดลอง

สรุปผลการทดลอง

จากการทำในโครงงานเรื่อง เครื่องวัดปริมาณการใช้ไฟฟ้า แบ่งออกเป็น 5 ส่วนใหญ่ๆคือ

5.1 ทดลอง PZEM ร่วมกับ Arduino และ หลอดไฟ พร้อมเช็คผลผ่าน Serial Monitor บน Arduino IDE

5.1 ทดลอง PZEM ร่วมกับ ESP32 และ หลอดไฟ พร้อมเช็คผลผ่าน Serial Monitor บน Arduino IDE และ แอพพลิเคชัน Blynk

5.1 ทดลอง PZEM ร่วมกับ ESP32 และ เต้ารับโดยเปลี่ยนโหลดเป็นอย่างอื่น พร้อมเช็คผลผ่าน Serial Monitor บน Arduino IDE และ แอพพลิเคชัน Blynk

5.1 ทดลอง PZEM ร่วมกับ ESP32 และ เต้ารับโดยเปลี่ยนโหลดเป็นอย่างอื่น พร้อมเช็คผลผ่าน Fireborad

5.1 การเช็คสถานะ ON OFF ของโหลดเมื่อโหลดคือ I pad และ I phone

จากการทดลองได้ทำการแสดงผลผ่าน Serial Monitor และ Serial Plotter ซึ่งในส่วนของ Serial Monitor จะเป็นการแสดงค่าของ Voltage, Current, Power, Energy, ค่าไฟ และในส่วนของ Serial Plotter จะแสดงผลเพียงค่า FT กับปริมาณการใช้ไฟ พบว่า การใช้งาน อุปกรณ์ โดย มีโหลด ทุก ๆ 1วินาทีใช้มีการใช้อย่างคงที่ และ การใช้ esp32 กับ pzem พบว่า ความคาดเคลื่อนมากกว่าการใช้ arduino uno กับ pzem ไม่เกิน 1-2 เปอร์เซ็นต์ และพบว่า การทดลองที่ 3.3 หากปรับปริมาณความร้อนของเตารีดปริมาณการใช้ไฟฟ้าจะเปลี่ยนไปซึ่งจะเห็นได้จากกราฟของแอพพลิเคชัน Blynk และจากการทดลองที่ 4 การแสดงข้อมูลผ่าน Fire base และ นำค่าแสดง กราฟ บน Fire board ซึ่งพบว่า ปริมาณการใช้ไฟของ ipad และ iphone จะคงที่และมีการเหวี่ยงของค่า Power บ้างบางครั้ง และเมื่อ ทั้งสองอุปกรณ์ มีแบตเตอรี่ ที่สถานะของทั้งสองแจ้งว่าครบ 100 เปอเซ็นต์ จะพบว่า Power ที่วัดได้ และ ส่งผ่านขึ้นทั้ง Fire base และ Fire borad จะมีค่า เท่ากับ 0 watt โดย ผลการทดลองที่ 5 ผลเป็นการทดลอง แสดง สถานะ ซึ่งจะแสดงตามค่า Power ที่วัดได้และ แยกการใช้งาน ของ อุปกรณ์ บางชนิดอย่างคราวๆ ซึ่งในการทดลองผู้ทำการทดลองได้ ใช้ ipad และ iphone ในการทดลองเก็บค่า สถิติ อย่างที่เห็นได้จากผลการทดลองที่ 5 โดยแสดงผลผ่าน สถานะ ON OFF บน Fire borad โดยสรุปภาพรวมของการทำงานทั้งหมด คือสามารถ บอกปริมาณการใช้ไฟฟ้า ของโหลดที่นำมาใช้มาโดยบอกค่า Voltage Current Power Energy Cost บน Application Blynk ซึ่งสามารถดูสถานะแบบ Realtime และสรุปค่าใช้จ่าย เมื่อ เลิกใช้งานและ สามารถ เช็คปริมาณการใช้ย้อนหลังได้ผ่าน Firebase ซึ่งจะ Update และแสดงค่าต่างๆ ทั้ง Cost Energy Power Current และ Voltage ซึ่งสามารถดูว่าใช้ไปเท่าไร เวลา

ไหน และ ดูองค์รวม การใช้งาน ผ่านกราฟ ทาง Fireboard พร้อมทั้งเช็คสถานะ การ ON OFF ของโหลด I Phone และ I Pad โดยทั้งนี้การเช็คสถานะ ON OFF ของโหลดทั้งสอง จะมี Delay time ประมาน 30 วินาที เนื่องจาก การทำงาน Update ข้อมูลจะถูกส่ง ทุกๆ30วินาที

จากการทดลองในการทดลอง ในการทดลองเครื่องใช้ไฟฟ้า (1) เตารีด GC1010/01 จากการทดลอง พบว่า วัดค่า Power(Real power) ที่ได้จากการที่ Pzem เฉลี่ยได้ 1160 W ซึ่งเมื่อเทียบกับคุณสมบัติของตัว เตารีด GC1010/01 ที่ทางบริษัทให้มาคือ 1200 W ซึ่งคิดเป็นค่า Error เป็น 3% (2) หัวชาร์จโทรศัพท์ Adapter Iphone จากการทดลองพบว่า วัดค่า Power(Real power) โดยเฉลี่ยได้ 6 W ซึ่งเมื่อเทียบกับคุณสมบัติของตัว Adapter Iphone ที่ทางบริษัทให้มา คือ 5 W ซึ่งคิดเป็นค่า Error เป็น 20% (3) หัวชาร์จโทรศัพท์ Adapter Ipad จากการทดลองพบว่า วัดค่า Power(Real power) โดยเฉลี่ยได้ 10 W ซึ่งเมื่อเทียบกับคุณสมบัติของตัว Adapter Ipad ที่ทางบริษัทให้มา คือ 13 W ซึ่งคิดเป็นค่า Error เป็น 8.3%

วิจารณ์ผลการทดลอง

เนื่องจากอุปกรณ์การวัดปริมาณการใช้ไฟฟ้า มีประสิทธิภาพ ไม่สูงมากนักทำให้การวัดปริมาณ บางครั้งเกิดความคาดเคลื่อนและส่งผลการคำนวณการใช้ไฟฟ้า และ การใช้ Firebase ฐานรับข้อมูล ทั้งนี้อยู่ บนบริการการใช้งานฟรี ทำให้เสถียรภาพการส่งข้อมูล เกิดความผิดพลาดบางเวลา Server มีการปิดปรับปรุง ทำให้เกิดความผิดพลาดไม่สามารถจัดส่งข้อมูลขึ้นบน Firebase ได้เพื่อการใช้งานที่สะดวกสะบายจะต้องเสีย บริการรายเดือนหรือรายปีซึ่งมีค่าใช้จ่ายสูง อีกทั้งการเช็คสถานะ ON OFF ไม่สามารถบอกได้ทันทีเนื่องจาก ข้อมูลที่จัดส่งจะ สามารถเช็ค ได้ทุกๆ 30 วินาทีเท่านั่น

เอกสารอ้างอิง

[1] GreatScott!. *DIY WiFi RGB LED Lamp || ESP8266 & Blynk.* [ออนไลน์]. 2017.

ชื่อเว็บไซต์: https://www.youtube.com/watch?v=DkJ1f5UIuak&ab_channel=GreatScott%21

จาก Youtube [11 พฤศจิกายน, 2561]

[2] PDAControl. *Meter PZEM-004 + ESP8266.*

ชื่อเว็บไซต์: http://pdacontrolen.com/meter-pzem-004-esp8266-platform-iot-blynk-app/

จาก pdacontrolen [11 พฤศจิกายน, 2561]

[3] *nemoman. คำนวณค่าไฟฟ้า.*

ชื่อเว็บไซต์: http://nemoman.blogspot.com/2018/01/pzem004t.html

จาก nemoman [11 พฤศจิกายน, 2561]

[4] 김동일. *전압전류 센서, PZEM-004T, 아두이노, Esp8266, D1.*

ชื่อเว็บไซต์: https://www.youtube.com/watch?v=978tKI_3U6U&ab_channel=김동일

จาก Youtube [11 พฤศจิกายน, 2561]

[5] กฟน.. *การคิดค่าไฟฟ้าประเภทต่างๆ.*

ชื่อเว็บไซต์ : https://www.mea.or.th/aboutelectric/116/280/form/11

จาก การไฟฟ้านครหลวง [11 พฤศจิกายน, 2561]

ภาคผนวก

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <IOXhop_FirebaseESP32.h>
#include <BlynkSimpleEsp32.h>
#include <PZEM004T.h>
#include <time.h>
#include <HardwareSerial.h>
#define WIFI_SSID     "TEm"
#define WIFI_PASSWORD "Ehaaaaa1234"
#define BLYNK_PRINT Serial
#define FIREBASE_HOST "https://peee-6c175.firebaseio.com/"
#define FIREBASE_AUTH "c1vh1ckLSORZaVIbRiOuVnlwm0Yd1jltDsxlQ5A"
#ifndef min
#define min(a,b) ((a)<(b)?(a):(b))
#endif
float FT = -15.9;
float calBill(float Unit, float ft, bool over_150_Unit_per_month = false) ;
float i_phone=0;
float i_pad=0;
float i_pad_phone=0;
int timezone = 7;
```

```
#define FIX_FT -15.90
char ntp_server1[20] = "ntp.ku.ac.th";
char ntp_server2[20] = "fw.eng.ku.ac.th";
char ntp_server3[20] = "time.uni.net.th";
char auth[] = "d8ad1c57169641a0bcf670a721de5ae0";
int dst = 0;
unsigned long lastUpdateEnergy = 0, lastUpdateFirebase = 0;
float Volt, Amp, Power, Bill, Energy, Energy2,Power_real,PF;
HardwareSerial hwserial(2);
PZEM004T pzem(&hwserial);  // (RX,TX) connect to TX,RX of PZEM
IPAddress ip(192, 168, 1, 1);

void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..");
  }
  Serial.println();
```

```
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  while (!pzem.setAddress(ip)) {
    Serial.println("Connecting to PZEM...");
    delay(500);
  }
  Blynk.begin(auth, WIFI_SSID, WIFI_PASSWORD);
  configTime(timezone * 3600, dst, ntp_server1, ntp_server2, ntp_server3);
  Serial.println("Waiting for time");
  while (!time(nullptr)) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.println("Now: " + getTime());
  Bill = calBill(Energy, FIX_FT, true);
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}
```

```
void loop() {
  Blynk.run();
  Monitor();
   if(Power>=0 && Power<=14){
       i_phone = Power;
       i_pad_phone = 0;
       i_pad = 0;
   } else if(Power>=14 && Power<=25){
       i_pad = Power;
       i_pad_phone = 0;
       i_phone = 0;
   } else if(Power>=25 && Power<=50){
       i_pad_phone = Power;
       i_pad = 0;
       i_phone = 0;
   }
 if ((millis() - lastUpdateEnergy) >= 1000) {
   lastUpdateEnergy = millis();
   SetBlynk();
   unsigned long time;
   unsigned long startTime = millis();
```

```
    Volt = pzem.voltage(ip);
    Volt = Volt > 0 ? Volt : 0;
    Amp = pzem.current(ip);
    Amp = Amp > 0 ? Amp : 0;
    Power_real = pzem.power(ip);
    Power_real = Power_real > 0 ? Power_real : 0;
    Power = Volt * Amp;
    PF = Power_real / Power;
    Energy += Power;
    if (Power >= 1) {
    time += 1;
    Energy2 += (Power * time) / (1000 * 3600);
  }
    Bill = calBill(Energy2, FIX_FT, false);
  time_t now = time(nullptr);
  struct tm* nowTime = localtime(&now);
  if ((nowTime->tm_sec % 30) == 0) {
    lastUpdateFirebase = millis();
    StaticJsonBuffer<200> jsonBuffer;
    JsonObject& root = jsonBuffer.createObject();
    root["volt"] = Volt;
```

```
    root["amp"] = Amp;
    root["time"] = getTime();
    root["power"] = Power;
    delay(2000);
    String name = Firebase.push("/logPower", root);
    if (Firebase.failed()) {
      Serial.print("pushing failed:");
      Serial.println(Firebase.error());
      delay(2000);
      Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..");
  }
      return;
    }
    Serial.print("pushed: /logPower/");
    Serial.println(name);
```

```
    Firebase.setFloat("/iphone", i_phone);
    if (Firebase.failed()) {
     Serial.print("pushing failed:");
     Serial.println(Firebase.error());
     delay(2000);
     Serial.begin(115200);
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("connecting");
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.println("Connecting to WiFi..");
}
    return;
  }
  Firebase.setFloat("/ipad", i_pad);
   if (Firebase.failed()) {
    Serial.print("pushing failed:");
    Serial.println(Firebase.error());
    delay(2000);
    Serial.begin(115200);
```

```
 WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
 Serial.print("connecting");
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.println("Connecting to WiFi..");}
     return;
   }
   Firebase.setFloat("/ipad_and_iphone", i_pad_phone);
    if (Firebase.failed()) {
     Serial.print("pushing failed:");
     Serial.println(Firebase.error());
     delay(2000);
     Serial.begin(115200);
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("connecting");
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.println("Connecting to WiFi..");
}
    return;
  }
```

```
    Firebase.setFloat("/amount", calBill(Energy / 1000, FIX_FT, false));
    if (Firebase.failed()) {
      Serial.print("pushing failed:");
      Serial.println(Firebase.error());
      delay(2000);
      Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi..."); }
      return ;}
}
     if(Power>=65.00)
       printf("TV and ipad and iphone\n");
     else if(Power>=60.00)
       printf("tV and ipad\n");
     else if(Power>=50.00)
       printf("TV and iphone\n");
     else if(Power>=40.00)
       printf("TV\n");
```

```
      else if(Power>=25.00)
        printf("ipad and iphone\n");
      else if(Power>=14.00)
        printf("ipad\n");
      else if(Power>=10.00)
        printf("iphone\n");
      else
        printf("no\n");
    delay(0); // Disable WDT
} void Monitor() {
  Serial.print("Volt: "); Serial.print(Volt); Serial.print("V\n");
  Serial.print("Current: "); Serial.print(Amp); Serial.print("A\n");
  Serial.print("power Now: "); Serial.print(Power); Serial.print("W\n");
  Serial.print("power real: "); Serial.print(Power_real); Serial.print("W\n");
  Serial.print("PF: "); Serial.print(PF); Serial.print("\n");
  Serial.print("Energy: "); Serial.print(Energy2, 3); Serial.print("kWh\n");
  Serial.print("Cost: "); Serial.print(Bill, 3); Serial.print("฿\n");
  Serial.print("FT: "); Serial.println(FIX_FT);
  Serial.println("--------------------");
}
void SetBlynk() {
```

```
  Blynk.virtualWrite(V0, Volt);
  Blynk.virtualWrite(V1, Amp);
  Blynk.virtualWrite(V2, Power);
  Blynk.virtualWrite(V4, Energy2);
  Blynk.virtualWrite(V5, Bill);
  Blynk.virtualWrite(V6, FIX_FT);
}
BLYNK_WRITE(V7) {
  FT = param[0].asFloat();
}
BLYNK_WRITE(V8) {
  Power = 0.0;
  Energy = 0.0;
  Bill = 0.0; }
String getTime() {
  time_t now = time(nullptr);
  struct tm* newtime = localtime(&now);
  String tmpNow = "";
  tmpNow += String(newtime->tm_year + 1900);
  tmpNow += "-";
  tmpNow += String(newtime->tm_mon + 1);
```

```
  tmpNow += "-";
  tmpNow += String(newtime->tm_mday);
  tmpNow += " ";
  tmpNow += String(newtime->tm_hour);
  tmpNow += ":";
  tmpNow += String(newtime->tm_min);
  tmpNow += ":";
  tmpNow += String(newtime->tm_sec);
  return tmpNow;
}
float calBill(float Unit, float ft, 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 >= 6) 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 / 100);
  total += total * 7 / 100;
  return total;
}
```