รายงานโครงงานระบบสมองกลฝังตัว

เรื่อง โปรแกรมวัดกระแสไฟฟ้าแสดงผลผ่านเว็บไซต์

จัดทำโดย

1.นายจิรายุส ผัดผล รหัสประจำตัว 65030034 สาขาวิชา ครุศาสตร์วิศวกรรม

2.นายธีภพ ศรีเพชร รหัสประจำตัว 65030109 สาขาวิชา ครุศาสตร์วิศวกรรม

อาจารย์ที่ปรึกษา อ.ปิยะ จิตธรรมมาภิรมย์

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรมและเทคโนโลยี

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2567

คำนำ

ในปัจจุบัน การตรวจวัดกระแสไฟฟ้าตามจุดต่างๆ มักจะใช้มิลติมิเตอร์แบบดั้งเดิม ทั้งแบบอนาล็อกและดิจิทัล ซึ่งยังมีข้อจำกัดในเรื่องของการอ่านค่าและความเสี่ยงในการตรวจวัด โครงงานนี้จึงได้พัฒนาโปรแกรมวัดกระแสไฟฟ้าแสดงผลผ่านเว็บไซต์ เพื่อช่วยให้การวัดกระแสไฟฟ้าเป็นไปอย่างสะดวกและแม่นยำมากยิ่งขึ้น พร้อมทั้งสามารถแสดงข้อมูลแบบเรียลไทม์ผ่านหน้าเว็บไซต์ได้

สารบัญ

สารบัญรูปภาพ

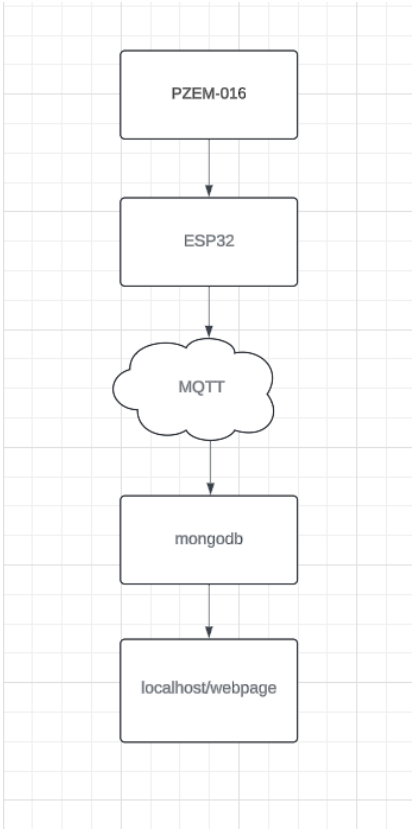
หน้า

บทที่ 1

ความเป็นมาความสำคัญ

ปัญหาที่ผู้จัดทำโครงงานได้ศึกษาพบว่าการวัดกระแสไฟฟ้าในระบบหรืออุปกรณ์ต่าง ๆ จำเป็นต้องมีการเช็คค่าเป็นระยะ ซึ่งบางครั้งทำให้เกิดความยุ่งยากและอันตราย นอกจากนี้ยังขาดระบบที่สามารถตรวจวัดและแสดงผลค่าได้แบบเรียลไทม์บนเว็บไซต์ ผู้จัดทำสนใจพัฒนาโปรแกรมที่สามารถวัดค่ากระแสไฟฟ้าและส่งข้อมูลไปยังเว็บไซต์แบบเรียลไทม์ผ่าน MQTT ซึ่งจะช่วยให้สามารถส่งและรับข้อมูลอย่างมีประสิทธิภาพ ผู้ใช้สามารถเข้าถึงข้อมูลได้จากทุกที่ผ่านเว็บเบราว์เซอร์

แผนผังการทำงาน/หลักการทำงาน/ผังงานของโปรแกรม



รูปที่ 1 แผนการทำงานของโปรแกรมวัดกระแสไฟฟ้าแสดงผลผ่านเว็บไซต์

วัตถุประสงค์

1. เพื่อพัฒนาระบบวัดกระแสไฟฟ้าที่สามารถส่งข้อมูลได้แบบเรียลไทม์
2. เพื่อออกแบบโปรแกรมที่สามารถแสดงข้อมูลกระแสไฟฟ้าผ่านหน้าเว็บไซต์
3. เพื่อศึกษาการใช้ MQTT ในการส่งข้อมูล

ขีดความสามารถของโครงงาน

1. สามารถวัดกระแสไฟฟ้าได้ในจุดที่ที่ต้องการตรวจสอบ
2. ส่งข้อมูลกระแสไฟฟ้าแบบเรียลไทม์ผ่าน MQTT
3. แสดงผลข้อมูลกระแสไฟฟ้าผ่านหน้าเว็บไซต์

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้โปรแกรมวัดกระแสไฟฟ้าที่สามารถส่งข้อมูลได้แบบเรียลไทม์
2. ได้โครงสร้างของโปรแกรมการวัดกระแสไฟฟ้าโดยใช้ MQTT
3. ได้ความรู้เกี่ยวกับการใช้งาน MQTT

บทที่ 2

เทคโนโลยีที่ใช้

1.เซ็นเซอร์ PZEM-016

1.1เซ็นเซอร์ PZEM-016 คือ

เซ็นเซอร์ PZEM-016 เป็นอุปกรณ์ที่ใช้สำหรับวัดพารามิเตอร์ทางไฟฟ้าต่าง ๆ เช่น แรงดันไฟฟ้า (Voltage), กระแสไฟฟ้า (Current), กำลังไฟฟ้า (Power), พลังงานไฟฟ้าสะสม (Energy) และ ปัจจัยกำลังไฟฟ้า (Power Factor) ในระบบไฟฟ้ากระแสสลับ (AC) โดยเฉพาะสำหรับไฟฟ้า 100V ถึง 260V AC ที่มีกระแสสูงสุดถึง 100A ผ่านการใช้เซ็นเซอร์ CT (Current Transformer) ซึ่งเซ็นเซอร์ PZEM-016 เหมาะสำหรับการใช้งานในระบบตรวจสอบการใช้พลังงานไฟฟ้าในบ้าน, โรงงาน หรืออุปกรณ์ต่าง ๆ ที่ต้องการการตรวจวัดกระแสและแรงดันอย่างแม่นยำ และสามารถนำข้อมูลที่วัดได้มาใช้ในการควบคุมหรือปรับแต่งการใช้พลังงานไฟฟ้าอย่างมีประสิทธิภาพ

1.2การทำงานของเซ็นเซอร์ PZEM-016

เซ็นเซอร์ PZEM-016 ใช้ในการวัดค่าทางไฟฟ้าหลัก ๆ เช่น กระแสไฟฟ้า แรงดันไฟฟ้า กำลังไฟฟ้า พลังงานสะสม และปัจจัยกำลังไฟฟ้าในระบบไฟฟ้ากระแสสลับ (AC) โดยใช้ Current Transformer (CT) สำหรับวัดกระแส และเชื่อมต่อกับสายไฟเพื่อวัดแรงดันโดยตรง จากนั้นข้อมูลที่วัดได้จะถูกส่งผ่านโปรโตคอล RS485 ไปยังไมโครคอนโทรลเลอร์หรือคอมพิวเตอร์PZEM-016 เหมาะสำหรับระบบที่ต้องการติดตามและควบคุมการใช้พลังงานแบบเรียลไทม์ ข้อมูลจากเซ็นเซอร์สามารถแสดงผลบนเว็บไซต์ผ่าน REST API เพื่อให้ผู้ใช้ตรวจสอบค่าต่าง ๆ จากระยะไกลได้อย่างสะดวก ซึ่งช่วยเพิ่มประสิทธิภาพในการตรวจสอบและจัดการพลังงานไฟฟ้าในระบบ

รูปที่ 2 เซ็นเซอร์ PZEM-016

2. ESP-WROOM-32 Microcontroller

ESP32 เป็นไมโครคอนโทรลเลอร์ที่รวมทั้ง Wi-Fi และ Bluetooth เข้าด้วยกันในตัวเดียว มีการออกแบบมาเพื่อใช้ในการพัฒนาอุปกรณ์ Internet of Things (IoT) ซึ่งช่วยให้การเชื่อมต่ออินเทอร์เน็ตและการสื่อสารแบบไร้สาย ESP32 เป็นที่นิยมใช้ในโครงการที่ต้องการเชื่อมต่อกับเครือข่าย Wi-Fi หรือ Bluetooth ในการรับส่งข้อมูล

รูปที่ 3 ESP-WROOM-32 Microcontroller

3.MQTT

3.1 MQTT คือ

MQTT (Message Queuing Telemetry Transport) เป็นโปรโตคอลการสื่อสารแบบเบา (lightweight messaging protocol) ที่ออกแบบมาเพื่อให้การส่งข้อมูลระหว่างอุปกรณ์ IoT (Internet of Things) มีความรวดเร็วและมีประสิทธิภาพ โดยเฉพาะในสภาพแวดล้อมที่มีการเชื่อมต่อที่ไม่เสถียรหรือมีแบนด์วิธต่ำ

3.2 ลักษณะการทำงานของ MQTT

การทำงานของ MQTT ถูกออกแบบมาให้มีความยืดหยุ่นและมีประสิทธิภาพในการสื่อสารระหว่างอุปกรณ์ โดยสามารถส่งข้อมูลแบบเรียลไทม์ในสภาพแวดล้อมที่มีการเชื่อมต่อที่ไม่เสถียร โดยมี broker ทำหน้าที่เป็นตัวกลางในการจัดการการรับส่งข้อมูลระหว่างผู้เผยแพร่และผู้สมัคร

3.3 ส่วนประกอบหลักของ MQTT

1. Publisher: ผู้เผยแพร่ (Publisher) เป็นอุปกรณ์หรือแอปพลิเคชันที่ส่งข้อมูลหรือข้อความไปยัง broker โดย publisher จะไม่รู้จักผู้สมัคร (subscriber) ที่รับข้อมูล แต่ส่งข้อมูลไปยัง broker ที่จะทำหน้าที่จัดการการส่งต่อไปยังผู้สมัครที่สนใจ
2. Broker: เป็นเซิร์ฟเวอร์กลางที่ทำหน้าที่รับข้อความจากผู้เผยแพร่ และกระจายข้อความเหล่านั้นไปยังผู้สมัครที่สนใจ โดย broker จะเก็บข้อมูลที่เกี่ยวข้องกับการสมัคร (subscription) ของผู้สมัครแต่ละคน เพื่อให้สามารถส่งข้อความได้ตามที่ผู้สมัครต้องการ

3. Subscriber: ผู้สมัคร (Subscriber) เป็นอุปกรณ์หรือแอปพลิเคชันที่สมัครรับข้อมูลจาก broker โดยสามารถเลือกหัวข้อ (topic) ที่ต้องการรับข้อมูลได้ เมื่อมีข้อมูลใหม่ที่เกี่ยวข้องกับหัวข้อที่สมัครไว้ Broker จะส่งข้อความไปยังผู้สมัครนั้นทันที
4. Topic: เป็นหัวข้อที่ใช้ในการจัดระเบียบการส่งข้อความ โดยข้อความจากผู้เผยแพร่จะถูกส่งไปยังหัวข้อที่กำหนด และผู้สมัครสามารถสมัครรับข้อมูลจากหัวข้อที่ต้องการได้ การใช้ topic ช่วยให้การสื่อสารระหว่างผู้เผยแพร่และผู้สมัครเป็นไปอย่างมีประสิทธิภาพ
5. Quality of Service (QoS): MQTT มีระดับการให้บริการ (Quality of Service) ที่ช่วยกำหนดความเชื่อถือได้ในการส่งข้อมูล

   - QoS 0: ส่งข้อมูลแบบไม่รับประกัน (At most once) ข้อมูลอาจไม่ถูกส่งถึงผู้สมัคร

   - QoS 1: ส่งข้อมูลอย่างน้อยหนึ่งครั้ง (At least once) ข้อมูลจะถูกส่งถึงผู้สมัครแต่สามารถส่งซ้ำได้

   - QoS 2: ส่งข้อมูลเฉพาะหนึ่งครั้ง (Exactly once) ข้อมูลจะถูกส่งถึงผู้สมัครโดยไม่มีการซ้ำ
6. Persistent Session: MQTT รองรับการจัดการเซสชันแบบถาวร (persistent session) ซึ่งช่วยให้ผู้สมัครสามารถกลับมาออนไลน์และรับข้อความที่ถูกส่งในระหว่างที่ออฟไลน์ได้ โดย broker จะเก็บข้อความที่ถูกส่งระหว่างที่ผู้สมัครออฟไลน์ไว้เพื่อส่งต่อเมื่อผู้สมัครกลับมาออนไลน์
7. Retained Messages: เมื่อผู้เผยแพร่ส่งข้อความที่ถูกกำหนดให้เป็น "retained message" Broker จะเก็บข้อความนั้นไว้ และเมื่อมีผู้สมัครเข้ามาที่หัวข้อนั้นครั้งแรก Broker จะส่งข้อความที่ถูกเก็บไว้ให้กับผู้สมัครทันที

8. Last Will and Testament (LWT): MQTT รองรับฟีเจอร์ LWT ซึ่งอนุญาตให้ผู้เผยแพร่กำหนดข้อความสุดท้ายที่จะถูกส่งไปยังหัวข้อเฉพาะ เมื่อผู้เผยแพร่เกิดการตัดการเชื่อมต่อโดยไม่ได้ตั้งใจ ฟีเจอร์นี้ช่วยให้ผู้สมัครสามารถรู้ได้ว่าผู้เผยแพร่ได้ตัดการเชื่อมต่อ

3.4 การประยุกต์ใช้ MQTT ในโครงงาน

ในโครงงาน MQTT จะทำหน้าที่เป็นช่องทางการสื่อสารระหว่างเซ็นเซอร์ที่วัดกระแสไฟฟ้าและระบบที่แสดงผลข้อมูลบนหน้าเว็บ ข้อมูลกระแสไฟฟ้าที่วัดได้จากเซ็นเซอร์จะถูกส่งไปยังเว็บเซิร์ฟเวอร์ผ่าน MQTT จากนั้นผู้ใช้งานสามารถดูข้อมูลได้แบบเรียลไทม์ผ่านหน้าเว็บ

4.Node.js/Express.js

4.1 Node.js/Express.js คืออะไร

Node.js เป็นแพลตฟอร์มที่ใช้ในการพัฒนาแอปพลิเคชันฝั่งเซิร์ฟเวอร์ด้วยภาษา JavaScript ที่สามารถประมวลผลงานแบบ asynchronous และจัดการการทำงานแบบ non-blocking I/O ได้อย่างมีประสิทธิภาพ มันทำงานบน single-threaded แต่สามารถจัดการกับงานหลายๆ งานพร้อมกันผ่าน event-driven architecture Node.js จึงเหมาะสำหรับการพัฒนาเว็บแอปพลิเคชันที่ต้องการความเร็วและประสิทธิภาพสูง เช่น การประมวลผล real-time, การทำ API, หรือการเชื่อมต่อฐานข้อมูล

Express.js เป็น framework ที่สร้างขึ้นบน Node.js เพื่อช่วยให้การพัฒนาเว็บแอปพลิเคชันเป็นเรื่องง่ายและเป็นระบบมากขึ้น Express.js ช่วยจัดการเส้นทาง (routing) การรับ-ส่งข้อมูล และ middleware ต่างๆ ที่ใช้ในแอปพลิเคชัน

4.2 คุณสมบัติของ Node.js/Express.js

1. Node.js
   - Non-blocking I/O: รองรับการทำงานหลายงานพร้อมกันโดยไม่ต้องรอการทำงานแต่ละคำสั่งเสร็จสมบูรณ์ก่อน
   - Event-driven architecture: ช่วยให้จัดการงานหลายอย่างได้พร้อมกันโดยใช้ทรัพยากรน้อย
   - Cross-platform: สามารถทำงานได้ทั้งบน Windows, macOS, และ Linux
2. Express.js
   - Routing: จัดการเส้นทาง (URL) ได้อย่างมีประสิทธิภาพ ช่วยให้สามารถกำหนดการทำงานสำหรับแต่ละเส้นทางของเว็บแอปพลิเคชันได้สะดวก
   - Middleware: ช่วยจัดการกับการประมวลผล request/response ก่อนที่จะส่งถึงฟังก์ชันหลัก ทำให้เพิ่มความยืดหยุ่นและการควบคุมในแอปพลิเคชัน
   - RESTful API: Express.js เป็นที่นิยมในการพัฒนา RESTful API เนื่องจากมีโครงสร้างที่เรียบง่ายและประสิทธิภาพสูง

4.3การใช้งาน Node.js/Express.js

Node.js มักใช้สำหรับการพัฒนาแอปพลิเคชันเว็บแบบ real-time หรือที่ต้องการประมวลผลจำนวนมาก เช่น แอปพลิเคชันการแชท, การแจ้งเตือน, หรือระบบที่ต้องการการทำงานแบบ real-time ในขณะที่ Express.js เป็น framework ที่ช่วยให้การสร้างเส้นทางการรับส่งข้อมูลบนเว็บแอปพลิเคชัน

4.4 การประยุกต์ Node.js/Express.js ใช้ในโครงงาน

ในโครงงาน "โปรแกรมวัดกระแสไฟฟ้าแสดงผลผ่านเว็บไซต์" Node.js จะทำหน้าที่เป็นพื้นฐานในการสร้างเว็บเซิร์ฟเวอร์สำหรับรับส่งข้อมูลกระแสไฟฟ้าจากเซ็นเซอร์ โดย Express.js จะช่วยจัดการเส้นทาง MQTT และประมวลผลข้อมูลที่ได้รับจากเซ็นเซอร์ และแสดงผลข้อมูลเหล่านั้นบนหน้าเว็บ

5.HTML/CSS

5.1 HTML/CSS คืออะไร

HTML เป็นภาษาหลักในการสร้างโครงสร้างของเว็บเพจ โดยทำหน้าที่จัดการการแสดงเนื้อหาต่างๆ เช่น ข้อความ, รูปภาพ, ลิงก์, ตาราง และส่วนประกอบอื่นๆ ของหน้าเว็บ HTML ใช้ "แท็ก" เพื่อระบุว่าแต่ละส่วนของเนื้อหาควรแสดงอย่างไร

CSS เป็นภาษาที่ใช้ในการจัดรูปแบบและตกแต่งเว็บเพจที่สร้างจาก HTML ทำให้เว็บเพจดูสวยงามและมีสไตล์ โดย CSS ใช้ในการควบคุมการแสดงผล เช่น สีตัวอักษร, ขนาดของฟอนต์, การจัดวางรูปแบบ, การเว้นระยะห่าง, การจัดแนว และอื่นๆ โดยไม่ต้องแก้ไข HTML โดยตรง

5.2 การประยุกต์ HTML/CSS ใช้ในโครงงาน

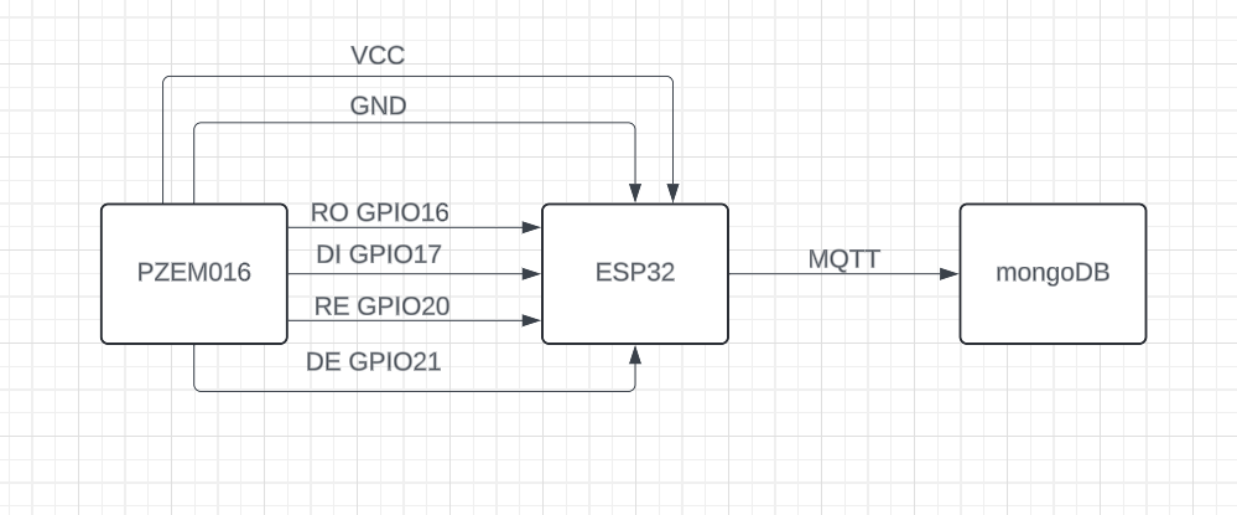
ในโครงงาน HTML จะใช้ในการสร้างโครงสร้างพื้นฐานของหน้าเว็บที่แสดงผลข้อมูลกระแสไฟฟ้า ส่วน CSS จะช่วยปรับแต่งการแสดงผลของข้อมูลเหล่านั้นให้ดูน่าสนใจและเป็นระเบียบมากขึ้น เช่น การจัดข้อมูลกระแสไฟฟ้า

บทที่ 3

วงจร/โปรแกรม

บทนำ

1. วงจร



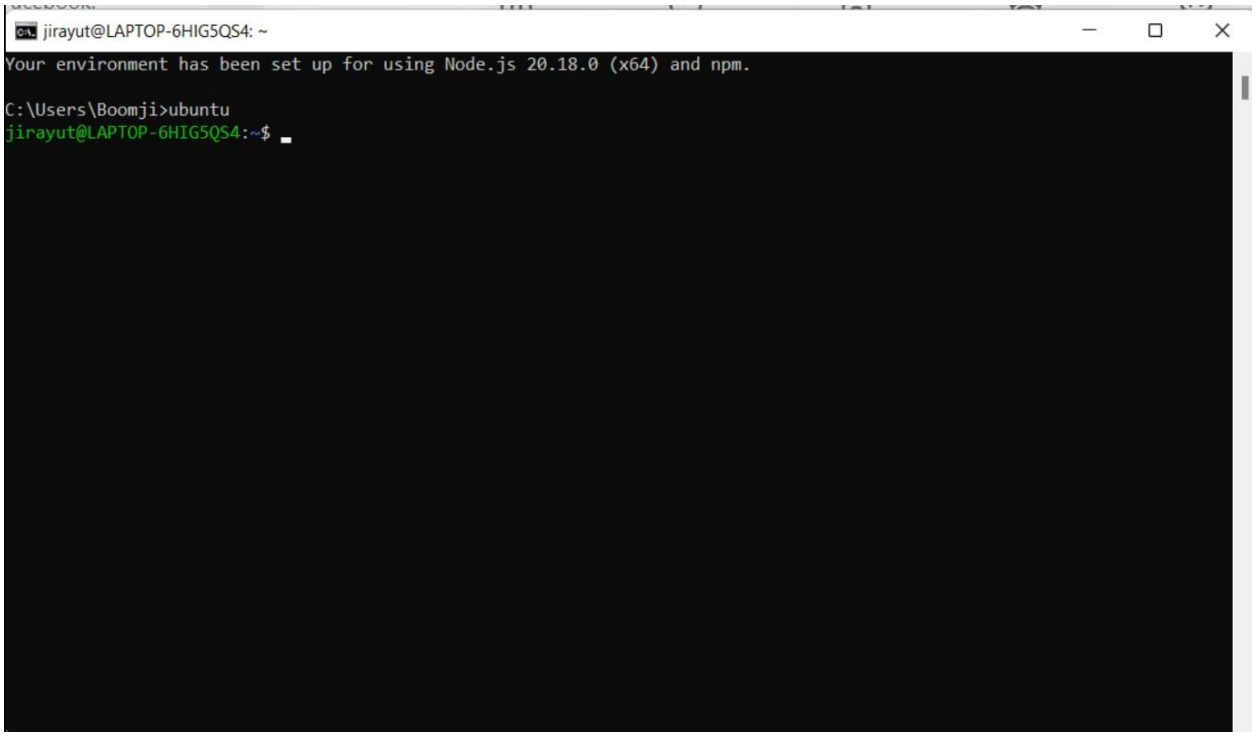
รูปที่ 4 แผนภาพการเชื่อมต่อวงจรของโครงงานโปรแกรมวัดอุณหภูมิโดยใช้HTTP

จากรูปที่ 4 เป็นรูปของวงจรในโครงงานโปรแกรมวัดกระแสไฟฟ้าโดยใช้ MQTT โดยจะมีอุปกรณ์ในวงจรได้แก่ เซ็นเซอร์ PZEM016 และบอร์ด esp32 เชื่อมต่อเข้าด้วยกัน จากเซ็นเซอร์ PZEM016 ต่อเข้ากับบอร์ด esp32 ที่ขา GPIO16,17,20,21 เพื่อส่งข้อมูลดิบ Power, Voltage และ Current ไปแปลงข้อมูลที่บอร์ด esp32

2.โปรแกรม

ในโครงงานโปรแกรมวัดกระแสไฟฟ้าโดยใช้ MQTT จะมีการใช้โปรแกรมดังนี้

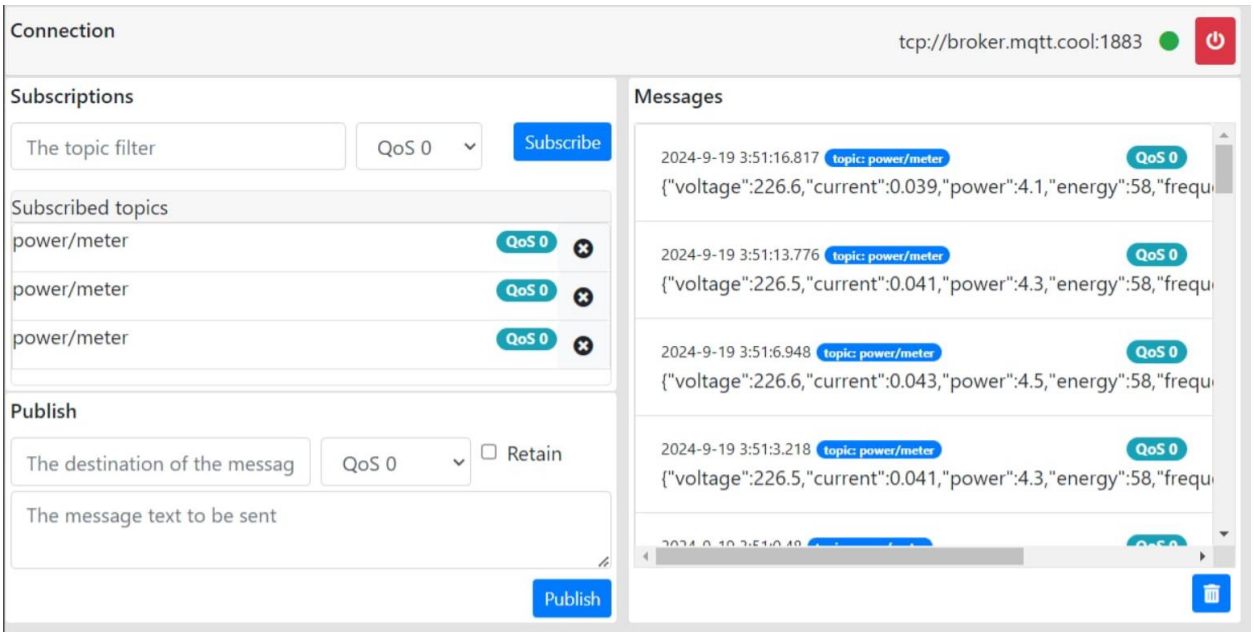
2.1 command prompt



รูปที่ 5 command prompt

จากรูปที่ 5 เป็นโปรแกรม command prompt บนระบบปฏิบัติการ Window โดยจะมีการลงส่วนเสริม Ubuntu เพื่อใช้งาน Docker ในการสร้าง container ได้แก่ nginx , express และ mongoDB โดยการเขียนโค้ดต่างๆเข้าไป

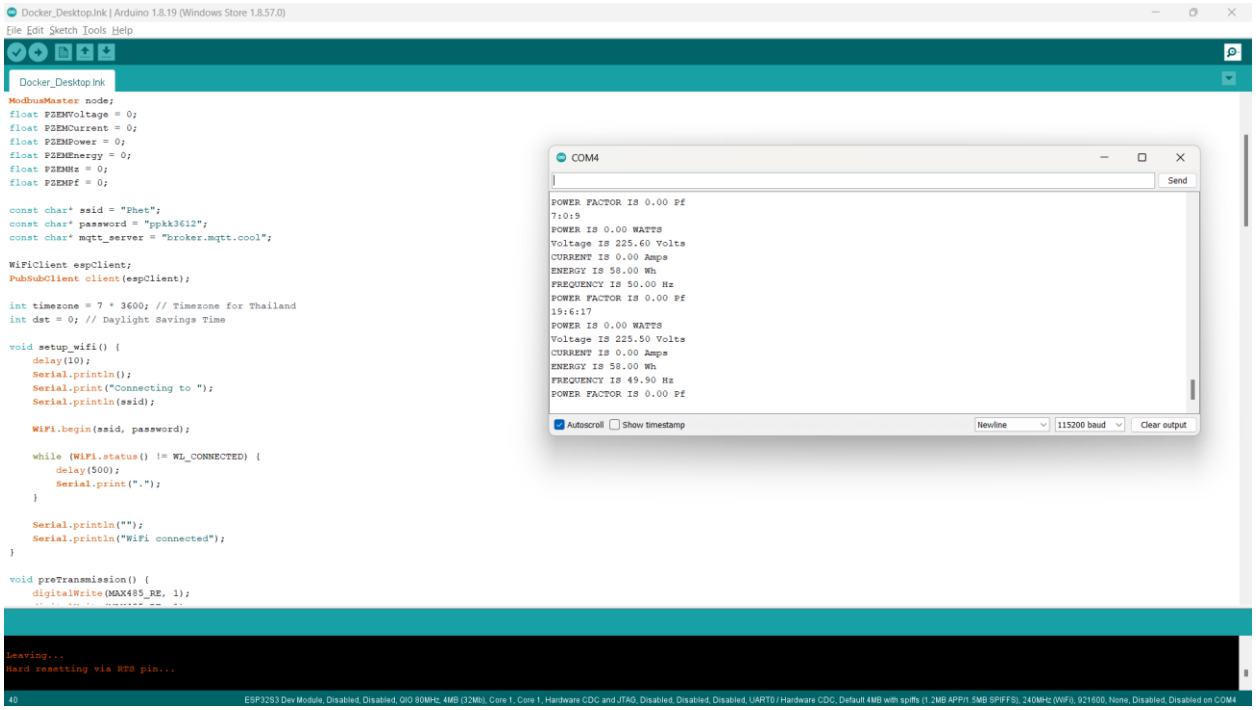
2.2 MQTT Client



รูปที่ 6 MQTT Client

จากรูปที่ 6 เป็นรูปแสดงหน้าเว็บสำหรับจัดการ MQTT โดยผู้ใช้ได้สมัครรับหัวข้อ power/meter หลายครั้ง ข้อความที่ได้รับประกอบด้วยข้อมูลเกี่ยวกับแรงดันไฟฟ้า (voltage), กระแส (current), กำลังไฟฟ้า (power), การใช้พลังงาน (energy) และค่าความถี่ (frequency) ของระบบไฟฟ้า

2.3 Arduino IDE



รูปที่ 7 โปรแกรมArduino IDE

จากรูปที่ 7 เป็นโปรแกรม Arduino IDE ที่รันคำสั่งลงบอร์ด esp32 ในการับค่าเซ็นเซอร์ PZEM016 และบอร์ด esp32 เชื่อมต่อเข้าด้วยกัน จากเซ็นเซอร์ PZEM016 ต่อเข้ากับบอร์ด esp32 ที่ขา GPIO16,17,20,21 เพื่อแปลงข้อมูลดิบ Power, Voltage และ Current ไปแปลงข้อมูลเพื่อให้แสดงค่า Power, Voltage และ Current ที่ถูกต้อง พร้อมส่งขึ้นใช้งาน MQTT

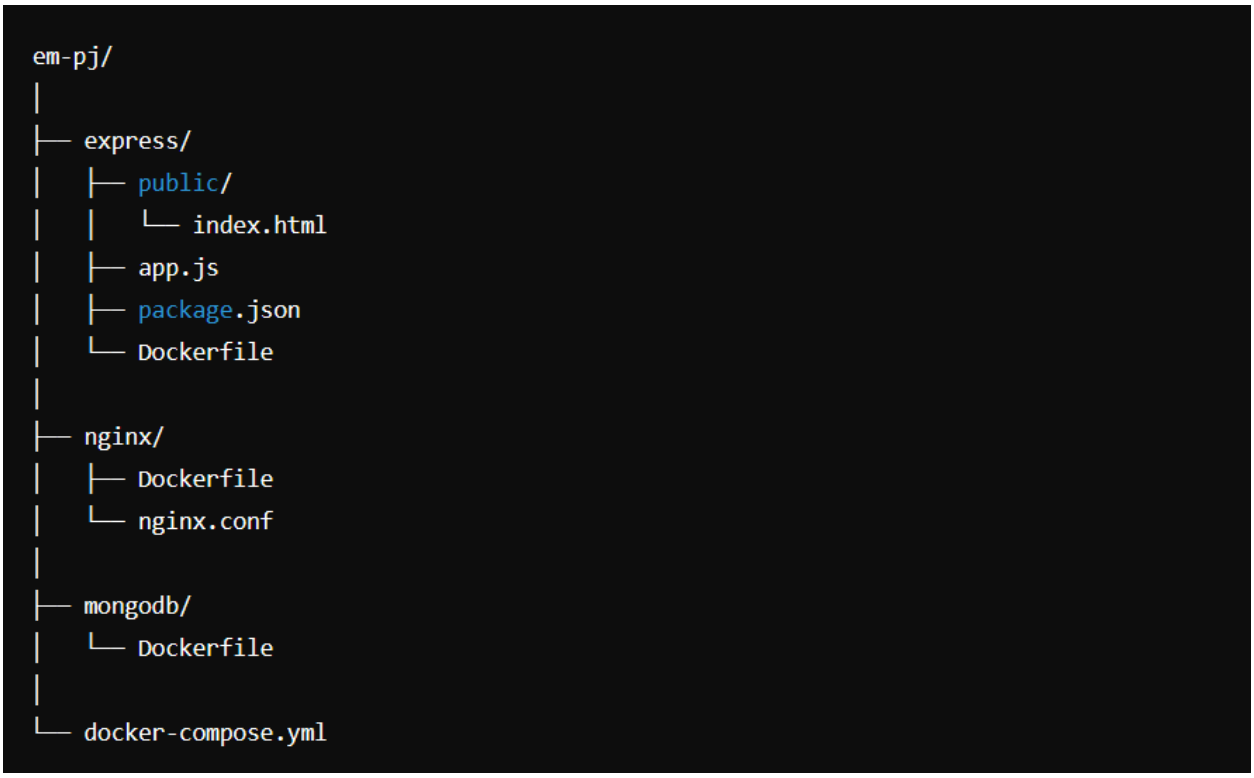
บทที่ 4

การใช้งาน

บทนำ

ในโครงงานโครงงานโปรแกรมวัดกระแสไฟฟ้าโดยใช้ MQTT จะสามารถใช้งานได้ต้องประกอบไปด้วย 3 ส่วนได้แก่

1.ส่วนของการทำงานบน Command Prompt ที่ใช้ Ubuntu



รูปที่ 8 แผนผังไฟล์ต่างๆ

จากรูปที่ 8 จะมีการสร้างโฟลเดอร์ในการทำงานขึ้นมาชื่อ em-pj ภายในนั้นจะมี 3 โฟลเดอร์และอีก 1 ไฟล์

1.1 โฟลเดอร์ express

ในโฟลเดอร์ express จะมีโฟลเดอร์publicและไฟล์app.js , Dockerfile และ package.json ดังนี้

1.1.1 โฟลเดอร์ public

ในโฟลเดอร์นี้จะมีไฟล์ index.html ที่คอยใช้ในการตกแต่งหน้าเว็บเช่นแสดงค่า แสดงค่า Power, Voltage , Current เวลาและวันที่

1.1.2 ไฟล์ app.js

เป็นไฟล์ที่ใช้กำหนดการบันทึกค่าข้อมูลลง database ลง mongoDB โดยจะใช้การกำหนด mongodb://mongodb:27017/sensorDB และค่า api key

1.1.3 ไฟล์ package.json

เป็นไฟล์ที่กำหนดส่วนเสริมที่ใช้ในโครงงานโครงงานโปรแกรมวัดกระแสไฟฟ้าโดยใช้ MQTT เช่น cors express และ mongoose ในการติดตั้งต้องใช้คำสั่ง

npm install cors express mongoose แล้วใช้คำสั่ง npm init -y

1.1.4 ไฟล์ Dockerfile

เป็นไฟล์ที่ใช้กำหนดการทำงานของ Docker

1.2 โฟลเดอร์ nginx

จะมีไฟล์ Dockerfile และ nginx.conf

1.2.1 ไฟล์ Dockerfile

เป็นไฟล์ที่ใช้กำหนดการทำงานของ Docker

1.2.2 ไฟล์ nginx.conf

เป็นไฟล์ที่กำหนดค่าของ nginx เป็นเว็บเซิร์ฟเวอร์ที่ระบุเส้นทางของ port ต่างๆ

1.3 โฟลเดอร์ mongodb

ในนี้จะมีไฟล์ Dockerfile เป็นไฟล์ที่ใช้กำหนดการทำงานของDocker

2. ส่วนของการทำงานบน Arduino IDE

บน Arduino IDE ต้องเขียนโค้ดการทำงานร่วมระหว่างเซ็นเซอร์กับบอร์ดesp32ให้สามารถรับค่าของเซ็นเซอร์และการเชื่อมต่อกับMQTT

```
const char* ssid = "Phet";
const char* password = "ppkk3612";
const char* mqtt_server = "broker.mqtt.cool";

WiFiClient espClient;
PubSubClient client(espClient);

int timezone = 7 * 3600; // Timezone for Thailand
int dst = 0; // Daylight Savings Time

void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
}
```

รูปที่ 9 ส่วนการเชื่อมต่ออินเทอร์เน็ต

จากรูปที่ 9 เป็นส่วนของการกำหนด ssid , password ของบอร์ดesp32 เพื่อเชื่อมต่ออินเทอร์เน็ตในบอร์ด esp32

```
const char* ssid = "Phet";
const char* password = "ppkk3612";
const char* mqtt_server = "broker.mqtt.cool";

WiFiClient espClient;
PubSubClient client(espClient);

int timezone = 7 * 3600; // Timezone for Thailand
int dst = 0; // Daylight Savings Time

void setup_wifi() {
    delay(10);
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
}
```

รูปที่ 10 ส่วนของการกำหนด MQTT server

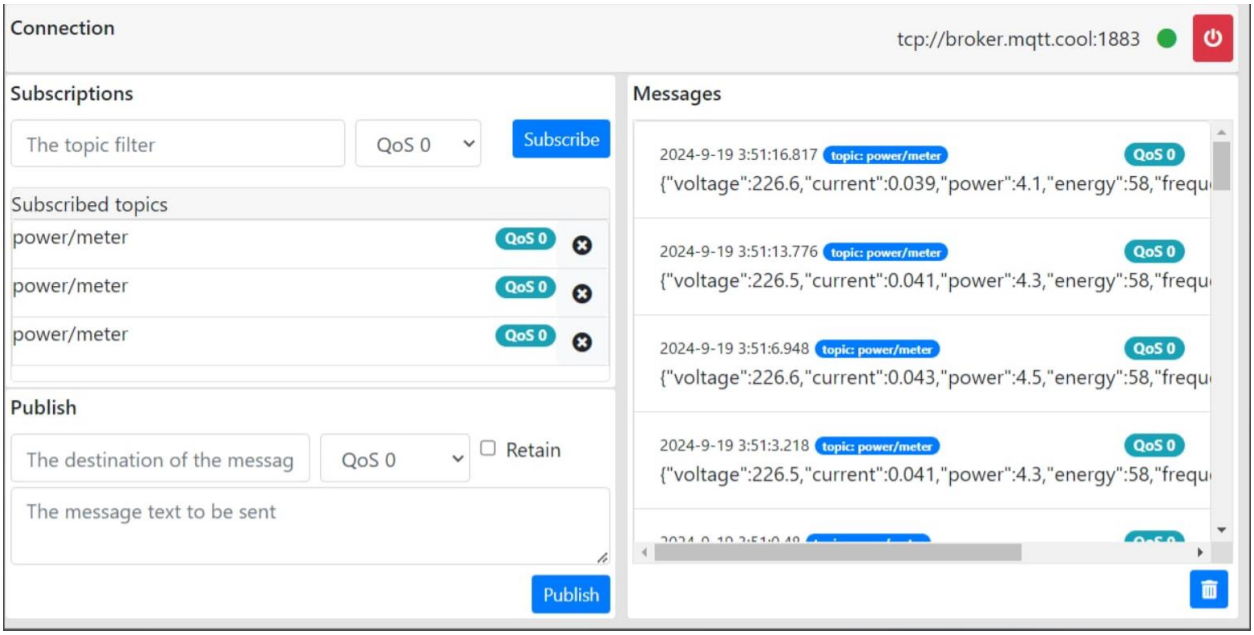
จากรูปที่ 10 เป็นส่วนการทำงานที่กำหนด MQTT server เพื่อเชื่อมต่อไปยัง MongoDB เพื่อบันทึกข้อมูลลงฐานข้อมูล

3. ส่วนของการทำงานบน MQTT

MQTT ทำงานผ่านกระบวนการหลัก 3 ส่วน

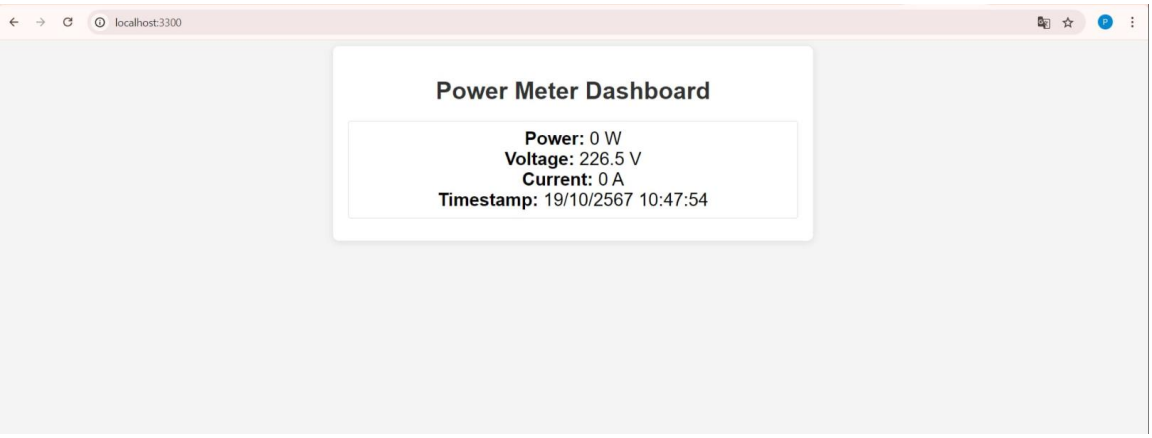
1. Broker: ตัวกลางในการรับส่งข้อมูลระหว่างผู้ส่ง (publisher) และผู้รับ (subscriber) ผ่านหัวข้อ (topics) โดยผู้ใช้ต้องเชื่อมต่อกับ broker ก่อน.
2. Publisher: อุปกรณ์หรือโปรแกรมที่ส่งข้อมูลไปยังหัวข้อที่กำหนด ผู้ส่งข้อมูลไม่ต้องรู้ว่าใครรับ เพียงแค่ส่งไปที่หัวข้อเท่านั้น
3. Subscriber: อุปกรณ์หรือโปรแกรมที่สมัครรับข้อมูลจากหัวข้อที่สนใจ เมื่อมีข้อความใหม่ถูกส่งมาที่หัวข้อนั้น ผู้รับก็จะได้รับข้อมูลอัตโนมัติ

QoS (Quality of Service) คือตัวกำหนดระดับความน่าเชื่อถือในการส่งข้อมูล โดยมี 3 ระดับ (0, 1, 2) ขึ้นอยู่กับความสำคัญของการส่งข้อมูลว่าจะส่งมากน้อยแค่ไหน



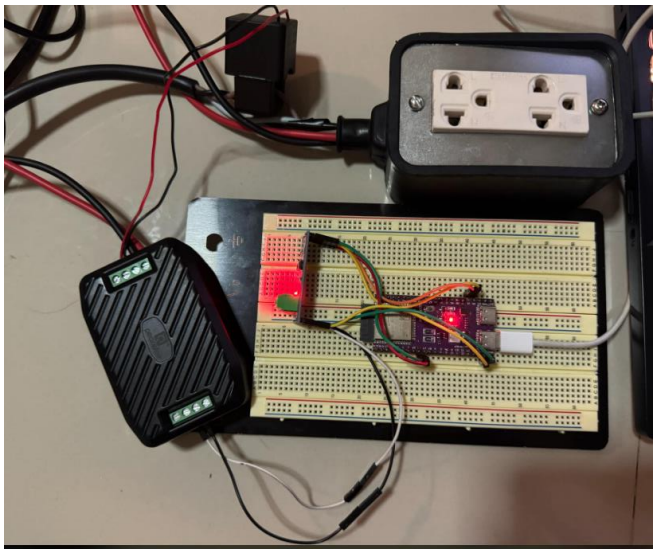
รูปที่ 11 docker container บน Docker Desktop

จากรูปที่ 11 เป็นผลลัพธ์จาการกรเชื่อมต่อกับ MQTT โดยได้สมัครรับหัวข้อ power/meter เพื่อข้อมูลเกี่ยวกับแรงดันไฟฟ้า (voltage), กระแส (current), กำลังไฟฟ้า (power), การใช้พลังงาน (energy) และค่าความถี่ (frequency) ที่อ่านจากเซ็นเซอร์



รูปที่ 12 เว็บแอปพลิเคชั่นที่localhost:3300

จากรูปที่ 12 เป็นการเข้าเบราว์เซอร์แล้วค้นหา localhost:3300 เพื่อเข้าถึงหน้าเว็บแอปพลิเคชั่นที่แสดงข้อมูลต่างๆ



รูปที่ 13 ชิ้นงาน

จากรูปที่ 13 เป็นส่วนของชิ้นงานที่ประกอบไปด้วย esp32 เซ็นเซอร์ PZEM061 และปลั๊กไฟที่พร้อมใช้งานแล้ว

บทที่ 5

ข้อดี/ข้อเสีย

จากการทำโครงงานโปรแกรมวัดกระแสไฟฟ้าโดยใช้MQTT ประสบผลสำเร็จจะได้หน้าเว็บแอปพลิเคชั่นและไฟล์การทำงานต่าง โดยผลลัพธ์จะมีดังนี้

ข้อดี

1. การตรวจสอบกระไฟฟ้าทำได้แบบเรียลไทม์
2. ข้อมูลแสดงผลเป็นดิจิทัล
3. ตรวจสอบการใช้พลังงานของเครื่องใช้ไฟฟ้า

ข้อเสีย

1. ต้องพึ่งพาการเชื่อมต่ออินเทอร์เน็ต
2. ต้องมีการตั้งค่าเซิร์ฟเวอร์และระบบเครือข่ายให้ถูกต้อง

อ้างอิง

Dekker, E. PZEM-016 Energy Monitor with Arduino. EvertDekker.com. สืบค้นจาก https://evertdekker.com/wp/?p=1307

Solarduino. PZEM-014/016 AC Energy Meter with Arduino. สืบค้นจาก https://solarduino.com/pzem-014-or-016-ac-energy-meter-with-arduino/

Dekker, E. Pzem016Test.ino - GitHub. GitHub. สืบค้นจาก https://github.com/EvertDekker/Pzem016Test/blob/master/Pzem016Test.ino

Yongyoot. PZEM-016 Arduino & NodeMCU. สืบค้นจาก https://yongyoot--eee01-blogspot-com.translate.goog/2020/06/pzem-016-arduino-nodemcu.html

Solar4Living. PZEM Arduino Modbus. สืบค้นจาก http://solar4living.com/pzem-arduino-modbus.htm

Arduino Forum. RS485 DC Communication Module & Arduino MEGA. สืบค้นจาก https://forum.arduino.cc/t/communication-between-rs485-dc-communication-module-and-arduino-mega/1092104

Pkarun. Blynk PZEM-016 with NodeMCU - GitHub. GitHub. สืบค้นจาก https://github.com/pkarun/Blynk-PZEM-016-with-NodeMCU