การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 48
The 48th Electrical Engineering Conference (EECON-48)
วันที่ 19-21 พฤศจิกายน 2568 ณ โรงแรมฟูราม่า จังหวัดเชียงใหม่

ระบบแสดงผลการใช้พลังงานไฟฟ้าสองแหล่งโดยใช้ Node-Red และ ThingsBoard

Dual-Source Energy Monitoring System via Node-Red and ThingsBoard

อำนวย ทีจันทึก[1*] ประพล จารตะคุ[1] และ ณรงค์ พิมพ์ปรุ [1]

[1]ศูนย์เครื่องมือวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยเทคโนโลยีสุรนารี amnuay@sut.ac.th* prapol@sut.ac.th narong_pim@sut.ac.th

บทคัดย่อ

บทความนี้นำเสนอการวิเคราะห์และแสดงผลการใช้พลังงานไฟฟ้าของสหกรณ์ออมทรัพย์ มหาวิทยาลัยเทคโนโลยีสุรนารี โดยใช้พลังงานจากสองแหล่ง ได้แก่ ระบบเซลล์แสงอาทิตย์ขนาด 10 กิโลวัตต์แบบ on-grid และไฟฟ้าจากการไฟฟ้าส่วนภูมิภาค ข้อมูลจากเครื่องวัดพลังงานไฟฟ้าถูกประมวลผลด้วย Node-RED Platform และจัดเก็บในรูปแบบอนุกรมเวลาด้วย ThingsBoard Platform ผลการดำเนินงานพบว่า Node-RED สามารถแปลงข้อมูลจากเครื่องวัดที่ใช้ Modbus Array เป็น IEEE-754 Floating Point ได้อย่างถูกต้อง พร้อมทั้งคำนวณการใช้พลังงานรายวัน ส่วน ThingsBoard สามารถนำเสนอข้อมูลการใช้ไฟฟ้าแบบเวลาจริง และสืบค้นย้อนหลังได้ทั้งรายช่วงเวลา รายวัน และรายเดือน ซึ่งช่วยเพิ่มประสิทธิภาพในการติดตามและวิเคราะห์รูปแบบการใช้พลังงานไฟฟ้า

คำสำคัญ: การใช้พลังงานไฟฟ้า, Node-RED, ThingsBoard, Modbus, ระบบเซลล์แสงอาทิตย์, การติดตามแบบเวลาจริง

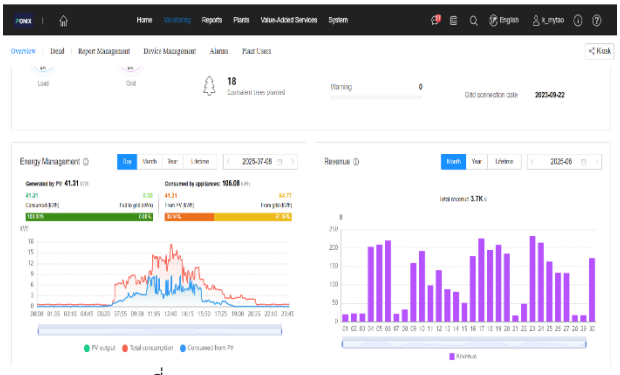
Abstract

This paper presents an analysis and visualization of electricity consumption at the Savings Cooperative of Suranaree University of Technology. The system utilizes two energy sources: a 10-kilowatt on-grid solar photovoltaic system and the Provincial Electricity Authority grid. Energy data collected from both sources were processed using the Node-RED platform and stored in a time-series database on the ThingsBoard platform. The results show that Node-RED can accurately convert Modbus Array data from the energy meters into IEEE-754 32-bit floating point values, such as voltage, current, active power, frequency, power factor, and active energy (kWh), and compute daily energy consumption. In addition, ThingsBoard provides real-time monitoring and supports historical queries by time intervals, daily, and monthly, thereby enhancing the efficiency of electricity usage monitoring and analysis.

Keywords: Electricity consumption, Node-RED, ThingsBoard, Modbus, Solar photovoltaic system, Real-time monitoring

1. บทนำ

ปัจจุบันการใช้พลังงานทดแทน โดยเฉพาะพลังงานแสงอาทิตย์ มีแนวโน้มเพิ่มขึ้นอย่างต่อเนื่อง อย่างไรก็ตาม เพื่อให้การบริหารจัดการพลังงานเป็นไปอย่างมีประสิทธิภาพ จำเป็นต้องมีระบบตรวจสอบและวิเคราะห์ข้อมูลการผลิตและการใช้พลังงานไฟฟ้าแบบเรียลไทม์ โดยทั่วไปในปัจจุบันอุปกรณ์ Inverter ที่ใช้ในระบบโซล่าเซลล์ก็จะมี Cloud Platform ไว้บริการลูกค้าอยู่แล้ว ในการใช้งานจะต้องมี User และ Password เพื่อเข้าไปตรวจสอบข้อมูลบน Cloud Platform [1]



รูปที่ 1 Dashboard Fusion Solar ของ HUAWEI

สหกรณ์ออมทรัพย์มหาวิทยาลัยเทคโนโลยีสุรนารี ได้ติดตั้งระบบโซล่าเซลล์ขนาดกำลังผลิต 10 กิโลวัตต์แบบ on-grid มีการใช้งาน Cloud Platform ของผลิตภัณฑ์ดังกล่าวเช่นกัน แต่เนื่องด้วยสหกรณ์ฯ มีสมาชิกจำนวนไม่น้อยกว่า 3 พันคน และสมาชิกสหกรณ์ฯ ต้องการรับทราบข้อมูลการใช้พลังงานไฟฟ้าแบบสาธารณะ ซึ่งข้อมูลที่ได้จากระบบ Cloud Platform ของผลิตภัณฑ์ดังกล่าวมีรายละเอียดพอสมควรดังแสดงในรูปที่ 1 แต่สมาชิกต้องการทราบข้อมูล แบบสรุป ดังนี้ 1) ใช้ไฟฟ้าในหนึ่งวันเท่าไร 2) โซล่าเซลล์ผลิตไฟฟ้าได้เท่าไร 3) ต้องจ่ายค่าไฟฟ้าเท่าไร

ดังนั้นบทความนี้จึงนำเสนอวิธีการใช้ Node-RED Platform แปลงข้อมูลที่ได้จากเครื่องวัดพลังงานไฟฟ้าแบบ Modbus Array เป็น IEEE-754 Floating Point พร้อมทั้งประมวลผลข้อมูลการใช้ไฟฟ้าในหนึ่งวัน และใช้ ThingsBoard Platform แสดงข้อมูลการใช้พลังงานไฟฟ้าใน Dashboard แบบสรุป พร้อมทั้งสร้าง Link แบบสาธารณะ สำหรับดูข้อมูลบน Web Site

2. ทฤษฎีที่เกี่ยวข้อง

2.1 Node-RED

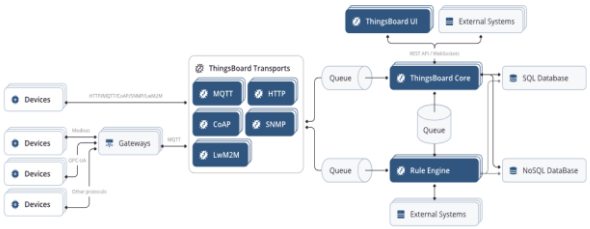
Node-RED คือเครื่องมือสำหรับการเขียนโปรแกรมแบบ visual programming ที่ใช้สำหรับเชื่อมต่ออุปกรณ์, API, และบริการต่าง ๆ เข้าด้วยกัน โดยใช้แนวคิดของ "โหนด" (Node) ที่ลากและวาง (drag-and-drop) บนหน้าจอกราฟิก เพื่อสร้าง flow หรือกระบวนการทำงานอัตโนมัติ

จุดเด่นของ Node-RED ใช้งานง่ายไม่จำเป็นต้องมีความรู้การเขียนโค้ดเชิงลึก เพียงแค่ลากโหนดต่าง ๆ มาต่อกัน นิยมใช้ในงาน Internet of Things (IoT) สำหรับเชื่อมต่อเซ็นเซอร์, อุปกรณ์สมาร์ทโฮม, หรือระบบคลาวด์ มีคลังโหนด (node library) ให้เลือกใช้จำนวนมาก และสามารถเขียนโหนดเองได้ด้วยภาษา JavaScript ซึ่ง Node-RED สร้างขึ้นบนแพลตฟอร์ม Node.js จึงรันได้บนอุปกรณ์ขนาดเล็กอย่าง Raspberry Pi ไปจนถึงเซิร์ฟเวอร์ขนาดใหญ่ [2]

ตัวอย่างการใช้งาน เช่น อ่านข้อมูลจากเซ็นเซอร์แล้วส่งข้อมูลไปเก็บในฐานข้อมูล รับข้อมูลจาก MQTT แล้วแจ้งเตือนผ่าน LINE หรืออีเมล สร้าง dashboard สำหรับแสดงผลข้อมูลแบบเรียลไทม์

2.2 ThingsBoard Platform

ThingsBoard เป็นแพลตฟอร์มโอเพ่นซอร์สสำหรับงานด้าน IoT ที่ออกแบบพร้อมใช้งานได้ทันที เพราะมีฐานข้อมูลในตัวเอง, สามารถวิเคราะห์ข้อมูลที่เข้ามา, ประมวลผลข้อมูลเพื่อแจ้งเตือน, ควบคุมอุปกรณ์ปลายทาง, สามารถสร้าง Dashboard แบบ dynamic และ responsive ได้ จากรูปที่ 2 โครงสร้างการทำงาน ThingsBoard Platform สามารถรองรับการสื่อสารจากอุปกรณ์ IoT ได้หลายโปรโตคอล คือ HTTP, MQTT และ CoAP [3]



รูปที่ 2 โครงสร้างการทำงาน ThingsBoard Platform

2.3 Modbus RTU protocol

Modbus RTU protocol เป็นโปรโตคอลที่ถูกพัฒนาเพื่อใช้กับงานอุตสาหกรรม สามารถนำมาใช้งานโดยไม่มีค่าใช้จ่าย การสื่อสารเป็นแบบ Master และ Slave โดย อุปกรณ์ Master จะทำหน้าที่สอบถามหรือสั่งเขียนข้อมูลบนอุปกรณ์ Slave ส่วนอุปกรณ์ Slave จะคอยรับคำสั่งแล้วตอบข้อมูลหรือเขียนข้อมูลตามที่อุปกรณ์ Master ร้องขอ โดยมีเฟรมการสื่อสาร Modbus RTU protocol 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 รายละเอียดตามรูปที่ 3 เมื่ออุปกรณ์ Master ต้องการอ่านค่าจากจากอุปกรณ์ Slave จะต้องส่ง Request packet ออกไป และรอ Response packet กลับมา ดังแสดงในรูปที่ 4 [3]

Start	หมายเลข	รหัสคำสั่ง	ชุดข้อมูล	CRC	Stop (Start)
≥3.5 Character time	1 ไบต์	1 ไบต์	N x 1 ไบต์	2 ไบต์	≥3.5 Character time

รูปที่ 3 ลักษณะเฟรมข้อมูลของ Modbus RTU

Slave address: 100 = 64 (hex) Start register 10 = 000A (hex)

Request packet:

Slave	Function	Start register		# of registers (3)		CRC checksum	
64	03	00	0A	00	03	2C	3C

Response packet:

Slave	Function	Byte count	Register 1		Register 2		Register 3		CRC checksum	
64	03	06	2E	CE	2E	E8	2F	13	0D	58

รูปที่ 4 เฟรมข้อมูล Request packet และ Response packet

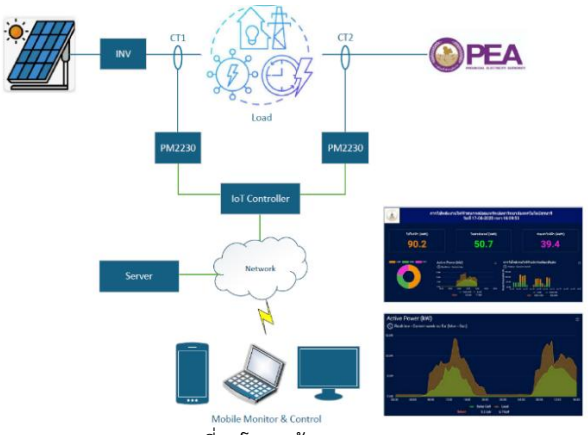
2.4 IEEE-754 Floating Point

IEEE-754 Floating Point คือมาตรฐานสากลสำหรับการแทนและคำนวณเลขทศนิยม (floating-point) ในคอมพิวเตอร์และอุปกรณ์อิเล็กทรอนิกส์ มาตรฐานนี้ถูกกำหนดขึ้นโดยสถาบัน IEEE ตั้งแต่ปี 1985 เพื่อให้การคำนวณเลขทศนิยมมีความแม่นยำ เชื่อถือได้ และสามารถแลกเปลี่ยนข้อมูลระหว่างระบบต่าง ๆ ได้อย่างถูกต้อง [4]

โครงสร้างของ IEEE-754 Floating Point ค่าทศนิยมในมาตรฐาน IEEE-754 จะถูกแทนด้วยส่วนประกอบหลัก 3 ส่วน ได้แก่

- Sign (เครื่องหมาย): 1 บิต ใช้บอกว่าเป็นเลขบวก (0) หรือเลขลบ (1)
- Exponent (เลขชี้กำลัง): ใช้แทนเลขชี้กำลังของฐาน 2 โดยมีการเพิ่มค่า bias เพื่อให้แทนเลขชี้กำลังลบได้
- Mantissa หรือ Significand (นัยสำคัญ): แทนค่าตัวเลขหลักที่สำคัญในรูปแบบฐาน 2

3. วิธีดำเนินการ



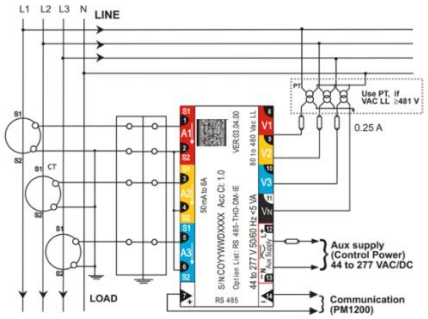
รูปที่ 5 โครงสร้างของระบบ

จากรูปที่ 5 เป็นโครงสร้างของระบบ ประกอบด้วยเครื่องวัดพลังงานไฟฟ้า Schneider รุ่น PM2230 จำนวน 2 ตัว อุปกรณ์ IoT Controller และคอมพิวเตอร์แม่ข่ายที่ติดตั้ง Node-RED และ ThingsBoard

โดยอุปกรณ์ IoT Controller จะทำหน้าที่อ่านค่าจากเครื่องวัดพลังงานไฟฟ้าทั้ง 2 ตัว และส่งข้อมูลแบบ Modbus Array ผ่านเครือข่ายคอมพิวเตอร์ไปที่คอมพิวเตอร์แม่ข่ายเพื่อให้ Node-RED แปลงข้อมูลให้อยู่ในรูปแบบ IEEE-754 Floating Point พร้อมทั้งประมวลผลข้อมูล ต่อจากนั้นก็จะข้อมูลที่ประมวลผลแล้วไปให้ ThingsBoard เพื่อแสดงผลข้อมูลต่อไป

3.1. เครื่องวัดพลังงานไฟฟ้า

เครื่องวัดพลังไฟฟ้าที่ได้ดำเนินการในบทความนี้และ Schneider รุ่น PM2230 โดยมีการต่อใช้งานตามรูปที่ 6

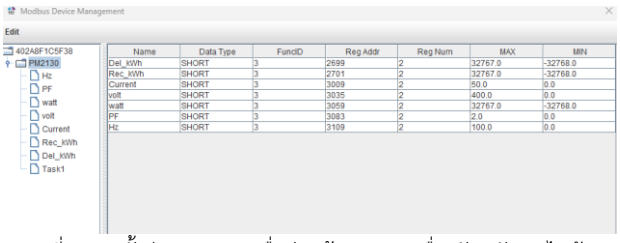


รูปที่ 6 ไดอะแกรมการต่อเครื่องวัดพลังงานไฟฟ้า

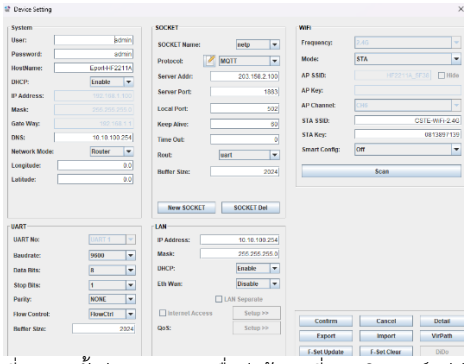
สำหรับค่าที่อ่านค่าได้จากเครื่องวัดพลังงานไฟฟ้าผ่านช่องสื่อสาร RS485 ด้วย Modbus RTU protocol จะประกอบด้วยค่าต่าง ๆ ดังนี้ Voltage แบบ line-to-line และ line-to-neutral, Current, Frequency, Active power (W), Reactive power (VAR) และ Apparent power (VA), Power Factor และ Power Quality [5]

3.2. IoT Controller

ในบทความนี้ใช้อุปกรณ์ ยี่ห้อ hi-flying รุ่น HF2211 สำหรับอ่านข้อมูลภายเครื่องวัดพลังงานไฟฟ้าผ่านช่องสื่อสาร RS485 และส่งข้อมูลผ่านเครือข่ายคอมพิวเตอร์ไปที่คอมพิวเตอร์แม่ข่าย โดยการใช้งานสามารถตั้งค่าเพื่ออ่านข้อมูลจากเครื่องวัดพลังงานไฟฟ้าดังแสดงในรูปที่ 7 และตั้งค่าการส่งข้อมูลไปที่คอมพิวเตอร์แม่ข่ายดังรูปที่ 8 [6]



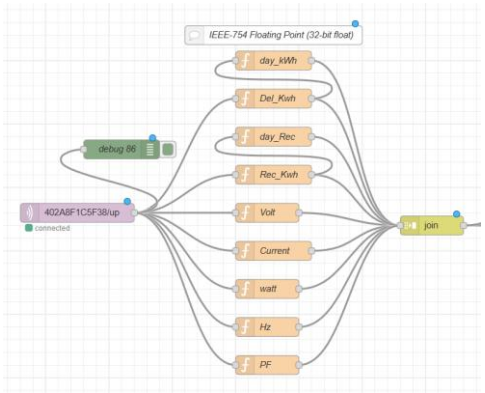
รูปที่ 7 การตั้งค่า HF2211 เพื่ออ่านข้อมูลจากเครื่องวัดพลังงานไฟฟ้า



รูปที่ 8 การตั้งค่า HF2211 เพื่อส่งข้อมูลที่คอมพิวเตอร์แม่ข่าย

3.3. Node-RED ประมวลผลข้อมูล

Node-RED จะรับข้อมูลจาก HF2211แบบ Modbus Array และทำแปลงข้อมูลให้อยู่ในรูปแบบ IEEE-754 Floating Point แสดงในรูปที่ 9 และส่งข้อมูลไปที่ Flow ประมวลผลข้อมูล ในรูปที่ 10



รูปที่ 9 flow แปลงข้อมูลให้อยู่ในรูปแบบ IEEE-754 Floating Point

การแปลงข้อมูลให้อยู่ในรูปแบบ IEEE-754 Floating Point ใน Function node ของ Node-RED สามารถดำเนินการแปลงอาร์เรย์ เป็นเลขทศนิยม (decimal) โดยข้อมูล Modbus Register จะเป็นแบบ 16-bit สองชุด (word) รวมเป็นค่า 32-bit IEEE-754 แบบ floating point หรือ integer

1.ใน Node-RED กรณีต้องการแปลงข้อมูล 32-bit เป็น Signed Integer ผลลัพธ์จะได้เป็นเลขทศนิยมฐานสิบแบบ signed 32-bit สามารถทำได้ดังนี้

```
let arr = msg.payload; // [18015, -25687]
let buf = Buffer.alloc(4);
buf.writeInt16BE(arr[0], 0);  // เขียนค่า 18015 ที่ตำแหน่ง 0
buf.writeInt16BE(arr[1], 2);  // เขียนค่า -25687 ที่ตำแหน่ง 2
msg.payload = buf.readInt32BE(0);
// อ่านเป็น signed 32-bit integer
return msg;
```

2.ใน Node-RED กรณีต้องการแปลงข้อมูล 32-bit เป็น IEEE-754 Floating Point สามารถทำได้ดังนี้

let arr = msg.payload; // [18015, -25687]

let buf = Buffer.alloc(4);

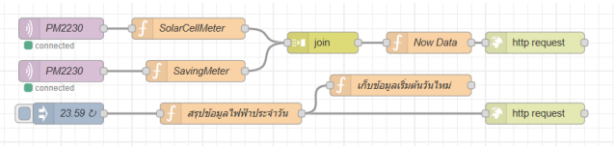
buf.writeInt16BE(arr[0], 0);

buf.writeInt16BE(arr[1], 2);

msg.payload = buf.readFloatBE(0); // อ่านเป็น float 32-bit

return msg;

3. หากต้องการใช้ node สำเร็จรูป ให้ติดตั้ง node-red-contrib-buffer-parser หรือ node-red-contrib-double เพื่อแปลงอาร์เรย์หรือบัฟเฟอร์เป็นเลขทศนิยมได้โดยตรง



รูปที่ 10 Flow ประมวลผลข้อมูล

ในรูปที่ 10 function node *SolarCellMeter* และ *SavingMeter* จะเป็นข้อมูล IEEE-754 Floating Point (32-bit float) และนำข้อมูลมารวมกันที่ node join ต่อจากนั้นส่งข้อมูลไปที่ function node *Now Data* เพื่อประมวลผลข้อมูลและส่งต่อไปแสดงผลที่ ThingsBoard ด้วย node http request

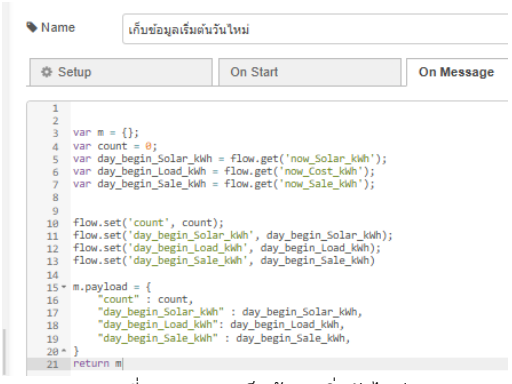
```
var ActivePowerSaving = parseFloat(W1) + parseFloat(W2);
var ActivePowerA = parseFloat(Wa1) + parseFloat(Wa2);
var ActivePowerB = parseFloat(Wb1) + parseFloat(Wb2);
var ActivePowerC = parseFloat(Wc1) + parseFloat(Wc2);

//============================
flow.set('now_Solar_kWh',t.m1Del); //ไฟฟ้าจาก SolarCell
flow.set('now_Cost_kWh', t.m2Del); //ไฟหลังงานไฟฟ้า
flow.set('now_Sale_kWh',t.m2Rec);  //ปล่อยย้อนเข้าระบบ pea
flow.set('now_W_Solar',W1);        //Active Poewr Solar
//============================

m.payload = {
    "count" : count,

    "now_Cost_kWh":parseFloat(t.m2Del).toFixed(2),   //หน่วยใช้ไฟฟ้าสะสม
    "now_Solar_kWh" :parseFloat(t.m1Del).toFixed(2), //หน่วย Solar Cell สะสม
    "now_Sale_kWh": parseFloat(t.m2Rec).toFixed(2),  //หน่วยไฟฟ้าเหลือ

    //===== Saving Acttive Power =====
    "now_Total_Load_kW": parseFloat(ActivePowerSaving).toFixed(3),
    "now_phase_A": parseFloat(ActivePowerA).toFixed(3),
    "now_phase_B": parseFloat(ActivePowerB).toFixed(3),
    "now_phase_C": parseFloat(ActivePowerC).toFixed(3),
    "now_V1" : parseFloat(t.m2V1).toFixed(2),
    "now_V2": parseFloat(t.m2V2).toFixed(2),
    "now_V3": parseFloat(t.m2V3).toFixed(2),

    //=====Solar Cell Active Power ===
    "now_Solar_KW": parseFloat(W1).toFixed(2),
    //===== PEA Active Power =====
    "now_pea_KW": parseFloat(W2).toFixed(2),

}
return m;
```

รูปที่ 11 Script ประมวลผลข้อมูล *Now Data* ก่อนจะส่งต่อไปแสดงผล

ในรูปที่ 10 function node *สรุปข้อมูลไฟฟ้าประจำวัน* และ *เก็บข้อมูลเริ่มต้นวันใหม่* จะใช้ประมวลผลข้อมูลข้อมูลสรุปการใช้พลังงานไฟฟ้าทุกวันก่อนเที่ยงคืนและเก็บข้อมูลไว้ประมวลผลในวันถัดไป


```
var new_day_Solar_kWh = flow.get('day_begin_Solar_kWh'); //m1del new day
var new_day_Load_kWh = flow.get('day_begin_Load_kWh');   //m2del new day
var new_day_Sale_kWh = flow.get('day_begin_Sale_kWh');   //m2rec new day

var Solar_kWh = flow.get('now_Solar_kWh'); //m1 del
var Cost_kWh = flow.get('now_Cost_kWh');   //m2 del
var Sale_kWh = flow.get('now_Sale_kWh');   //m2 rec

//==== call calculator
var total_solar = Solar_kWh - new_day_Solar_kWh;
var total_load = (Cost_kWh - new_day_Load_kWh) + ((Solar_kWh - new_day_Solar_kWh) - (Sale_kWh - new_day_Sale_kwh));
var total_sale = Sale_kWh - new_day_Sale_kwh;
var total_pea = total_load - total_solar;
//if (total_pea < 0) total_pea = 0;

//------ Load = pea + solar ------------
var load_use_pea = Cost_kWh - new_day_Load_kWh;
var load_use_solar = total_solar - total_sale;

var m ={};
m.payload ={
    "day_total_load": total_load,
    "day_total_solar": total_solar,
    "day_total_sale": total_sale,
    "day_total_pea": total_pea,
}
return m
```

รูปที่ 12 Script ประมวลผลข้อมูลประจำวัน




```
var m = {};
var count = 0;
var day_begin_Solar_kWh = flow.get('now_Solar_kWh');
var day_begin_Load_kWh = flow.get('now_Cost_kWh');
var day_begin_Sale_kWh = flow.get('now_Sale_kWh');


flow.set('count', count);
flow.set('day_begin_Solar_kWh', day_begin_Solar_kWh);
flow.set('day_begin_Load_kWh', day_begin_Load_kWh);
flow.set('day_begin_Sale_kWh', day_begin_Sale_kWh)

m.payload = {
    "count" : count,
    "day_begin_Solar_kWh" : day_begin_Solar_kWh,
    "day_begin_Load_kWh": day_begin_Load_kWh,
    "day_begin_Sale_kWh" : day_begin_Sale_kWh,
}
return m
```

รูปที่ 13 Script เก็บข้อมูลเริ่มวันใหม่

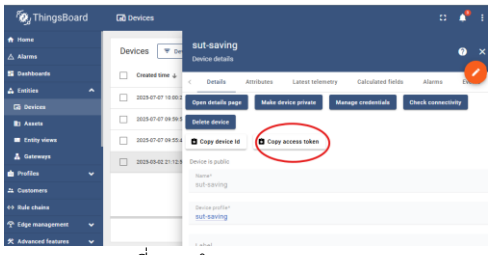
3.4. การใช้ ThingsBoard แสดงข้อมูล

การใช้งาน ThingsBoard แสดงผลข้อมูลต้องดำเนินการดังนี้

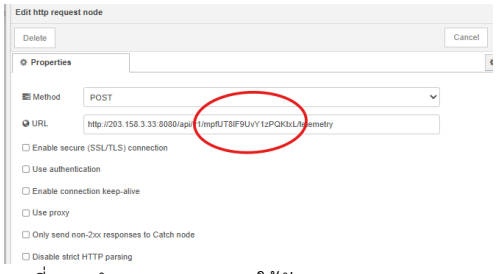
1) เข้าสู่ระบบ ของ ThingsBoard ด้วย Account ที่กำหนด

2) ดำเนินการสร้างอุปกรณ์จาก Menu Device Manager

3) สำเนา Access Token ได้จาก Menu Device detail เพื่อดังแสดงในรูปที่ 14 และนำไปใช้กับ node http request ของ Node-RED ดังรูปที่ 15



รูปที่ 14 สำเนา Access Token



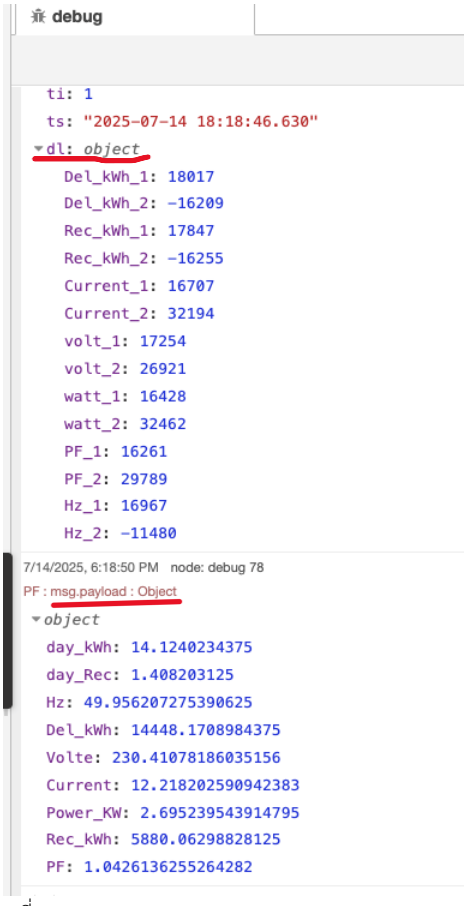
รูปที่ 14 นำ Access Token ใช้กับ node http request

รูปที่ 15 การติดตั้งระบบที่ใช้ในบทความนี้

4. ผลการทดลอง

4.1. ข้อมูลที่ HF2211 ส่งไปที่ Node-RED

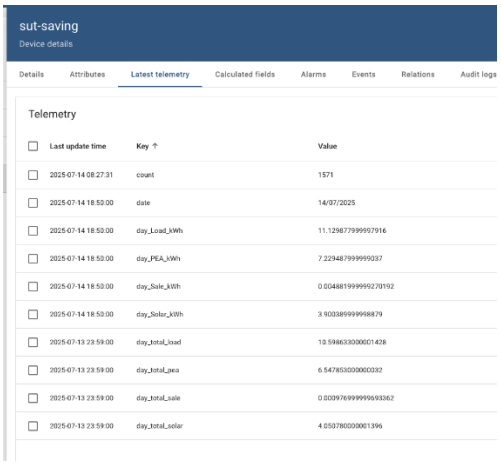
ผลการทดลองแสดงในรูปที่ 16 ค่า dl : object คือ Modbus Array ที่ส่งให้ Function node และ msg.payload คือ IEEE-754 Floating Point คือ Output ที่ออกจาก Function node



รูปที่ 16 Modbus Array และ IEEE-754 Floating Point

4.2. ผลการใช้ ThingsBoard Platform

ข้อมูลที่ ThingsBoard รับจาก Node-RED จะถูกจัดเก็บแบบอนุกรมเวลา (time series database) ตามรูปที่ 17



รูปที่ 17 ข้อมูลที่ ThingsBoard รับจาก Node-RED

รูปที่ 18 รูปแบบการแสดงผลผ่าน Dashboard ของ ThingsBoard

รูปที่ 19 ตำแหน่งติดตั้งเครื่องวัดพลังงานไฟฟ้า

รูปที่ 18 แสดงการใช้ไฟฟ้า 2 แหล่งได้แก่ ไฟฟ้าจากโซล่าเซลล์คือสีเขียว และการไฟฟ้าส่วนภูมิภาคคือสีม่วง โดยที่โหลดการใช้งานจริงคือสีส้ม ในรูปที่ 19 สามารถคำนวณการใช้งานพลังงานไฟฟ้าของอาคาร จากเครื่องวัดทั้งสองตัวโดยเอาค่าหน่วยการใช้ไฟฟ้ามารวมกัน

หน่วยการใช้ไฟฟ้า = เครื่องวัดพลังงานไฟฟ้า1 + เครื่องวัดพลังงานไฟฟ้า2

เครื่องวัดพลังงานไฟฟ้าตัวที่ 1 คือ เครื่องวัดการผลิตพลังงานไฟฟ้าจากโซล่าเซลล์ และเครื่องวัดพลังงานไฟฟ้าตัวที่ 2 คือ เครื่องวัดพลังงานไฟฟ้าจากการไฟฟ้าส่วนภูมิภาค

ใน Dashboard ของ ThingsBoard Platform สามารถสืบค้นข้อมูลย้อนหลัง ตามช่วงเวลา, รายวัน และรายเดือน เนื่องจากมีการเก็บข้อมูลแบบอนุกรมเวลาอยู่แล้ว

5. สรุป

ผลจากการดำเนินงานของบทความนี้ Node-RED Platform สามารถแปลงข้อมูลที่รับจากเครื่องวัดพลังงานไฟฟ้าแบบ Modbus Array เป็น IEEE-754 Floating Point (32-bit float) ได้ถูกต้องตามที่แสดงค่าบนหน้าจอของเครื่องวัดพลังงานไฟฟ้า และสามารถประมวลผลการใช้พลังงานไฟฟ้าแบบรายวันได้ ในส่วน ThingsBoard Platform จะแสดงข้อมูลแบบเวลาจริงได้, สามารถสืบค้นย้อนหลังแบบตามช่วงเวลา, แบบรายวัน และแบบรายเดือนได้

เนื่องจากมีการแสดงผลแบบเวลาจริงและแสดงข้อมูลไฟฟ้าสองแหล่งได้แก่ ระบบเซลล์แสงอาทิตย์และจากการไฟฟ้าส่วนภูมิภาค มีข้อดีคือสามารถควบคุมการใช้พลังไฟฟ้าได้ ตรวจสอบการใช้พลังงานไฟฟ้าได้อย่างถูกต้องแม่นตรง สามารถนำข้อมูลการใช้พลังงานไฟฟ้าไปวิเคราะห์และบริหารจัดการได้อย่างมีประสิทธิภาพ

เอกสารอ้างอิง

[1] “SmartPVMS 25.1.0 FusionSolar SmartPVMS User Manual”,https://support.huawei.com/enterprise/en/doc/EDOC1100462185 (Access 2025/07/01)

[2] “Node-RED User Guide ”https://nodered.org/docs/user-guide/ (Access 2025/07/01)

[3] อำนวย ทีจันทึก ประพลจารตะคุ และนพดลเสียงใหม่, “ ระบบอ่านมิเตอร์ไฟฟ้า 3 เฟส แบบอัตโนมัติและแสดงผลด้วย ThingsBoard”, การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครังที 44 2564, หน้า 696-699

[4] “IEEE Standard for Floating-Point Arithmetic (IEEE 754)” , https://en.wikipedia.org/wiki/IEEE_754 (Access 2025/07/01)

[5] “Register list for Powerlogics PM2000 series”, https://www.se.com/th/en/faqs/FA311706/ (Access 2025/07/01)

[6] “HF2211A User Manual” , http://www.hi-flying.com/wi-fi-iot/hf2211a (Access 2025/07/01)

อำนวย ทีจันทึก การศึกษาปริญญาโทวิศวกรรมโทรคมนาคมจากมหาวิทยาลัยเทคโนโลยีสุรนารี ปัจจุบันเป็นพนักงานองค์กรของรัฐในตำแหน่งวิศวกรสังกัดศูนย์เครื่องมือวิทยาศาสตร์และเทคโนโลยีมหาวิทยาลัยเทคโนโลยีสุรนารี งานวิจัยที่สนใจ ระบบสมองกลฝังตัว

ประพล จาระตะคุ การศึกษาปริญญาโทวิศวกรรมโทรคมนาคมจากมหาวิทยาลัยเทคโนโลยีสุรนารี ปัจจุบันเป็นพนักงานองค์กรของรัฐในตำแหน่งวิศวกรสังกัดศูนย์เครื่องมือวิทยาศาสตร์และเทคโนโลยีมหาวิทยาลัยเทคโนโลยีสุรนารี งานวิจัยที่สนใจการออกแบบวงจรทางด้านความถี่สูง

ณรงค์ พิมพ์ปรุ การศึกษาปริญญาโทครุศาสตรอุตสาหกรรมมหาบัณฑิตสาขาวิศวกรรมไฟฟ้า มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ปัจจุบันเป็นพนักงานองค์กรของรัฐในตำแหน่ง นายช่างไฟฟ้าสังกัดศูนย์เครื่องมือวิทยาศาสตร์และเทคโนโลยีมหาวิทยาลัยเทคโนโลยีสุรนารี งานวิจัยที่สนใจ การตรวจสอบงานเชื่อมโลหะแบบไม่ทำลาย