เทคนิคการบีบอัดข้อมูลสำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคารบนพื้นฐานของมาตรฐาน IEEE1888

นายพงษ์พจน์ ชัยบุญเรือง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2556

A DATA COMPRESSION TECHNIQUE FOR BUILDING ENERGY MANAGEMENT SYSTEM BASED ON IEEE 1888 STANDARD

Mr. Pongpot Chaiboonruang

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2013

หัวข้อวิทยานิพนธ์	เทคนิคการบีบอัดข้อมูลสำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคารบนพื้นฐานของมาตรฐาน IEEE1888
โดย	นายพงษ์พจน์ ชัยบุญเรือง
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร. วันเฉลิม โปรา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร. บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(รองศาสตราจารย์ ดร. เอกชัย ลีลารัศมี)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ผู้ช่วยศาสตราจารย์ ดร. วันเฉลิม โปรา)

.....กรรมการภายนอกมหาวิทยาลัย

(ดร. วิษุวัฒน์ ปลอดประดิษฐ์)

พงษ์พจน์ ชัยบุญเรือง : เทคนิคการบีบอัดข้อมูลสำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคารบนพื้นฐานของมาตรฐาน IEEE1888. (A DATA COMPRESSION TECHNIQUE FOR BUILDING ENERGY MANAGEMENT SYSTEM BASED ON IEEE 1888 STANDARD) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร. วันเฉลิม โปรา, 115 หน้า.

วิทยานิพนธ์นี้นำเสนอเทคนิคการบีบอัดข้อมูลสำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคารบนพื้นฐานของมาตรฐาน IEEE1888 โดยการปรับปรุงระบบจัดการพลังงานไฟฟ้าในมาตรฐาน IEEE1888 เล็กน้อยเพื่อให้ระบบรองรับข้อมูลที่มีรูปแบบการบีบอัด ซึ่งได้ทำการพัฒนา Proxy ที่ใช้กระบวนการบีบอัด Efficient XML Interchange (EXI) ในการบีบอัดข้อมูลในมาตรฐาน IEEE1888 ให้มีขนาดข้อมูลเล็กลง เพื่อให้เหมาะแก่การสื่อสารในช่องทางที่มีการจราจรหนาแน่น หรือช่องทางการสื่อสารที่มีราคาแพง เช่น 3G เป็นต้น โดยเกทเวยที่พัฒนาขึ้นมานี้จะทำการสื่อสารระหว่างมาตรฐาน IEEE1888 กับเครือข่ายเซ็นเซอร์ไร้สายพลังงานต่ำ IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) ที่ถูกออกแบบมาให้สื่อสารกับโปรโตคอล TCP/IP โดยเฉพาะ ภายในเครือข่าย 6LoWPAN นี้ได้ทำการพัฒนาโนด 2 ชนิด คือ โนดวัดสภาพแวดล้อม และโนดวัดพลังงานไฟฟ้า ในโนดวัดสภาวะแวดล้อมนั้นจะมีเซ็นเซอร์ที่วัดอุณหภูมิ ความชื่น แสงสว่าง และการเคลื่อนไหว ที่นำไปติดตั้งภายในอาคารเพื่อดูผลกระทบของค่าที่วัดได้จากเซ็นเซอร์ต่อการใช้พลังงาน ในโนดวัดพลังงานไฟฟ้านั้นจะใช้ไอซีวัดพลังงาน STPM01 ในการวัดพลังงานไฟฟ้า และมี Relay ตัดต่อไฟฟ้า ที่นำไปใช้ในการวัดการใช้พลังงานของเครื่องใช้ไฟฟ้า อีกทั้งยังสามารถควบคุมการทำงาน และแสดงผลข้อมูลได้จากแอพพลิเคชั่นที่พัฒนาขึ้นบนระบบปฏิบัติการ Android ที่มีรูปแบบตามแผนผังของห้องปฏิบัติการวิจัยการออกแบบวงจรฝังตัว และวงจรรวม

ภาควิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2556

5570299721 : MAJOR ELECTRICAL ENGINEERING
KEYWORDS: IEEE1888 / COMPRESSION TECHNIQUE / EXI / BUILDINGS ENERGY MANAGEMENT SYSTEM / BEMS

PONGPOT CHAIBOONRUANG: A DATA COMPRESSION TECHNIQUE FOR BUILDING ENERGY MANAGEMENT SYSTEM BASED ON IEEE 1888 STANDARD. ADVISOR: ASST. PROF. WANCHALERM PORA, Ph.D., 115 pp.

This thesis present a data compression technique for building energy management system based on IEEE1888 standard. The standard is modified such that the system can communicate with compression data format. The Efficient XML Interchange (EXI) compression algorithm is selected to squeeze down the IEEE1888 data. EXI together with XML Schema Definition (XSD) are used to explain IEEE1888 XML data structure and hints so that the name of elements or attribute of the compressed data can be ignored. The ignored information can be reproduced from XSD file when decompressing. The Membrane Service Proxy was employed to develop an IEEE1888 compression proxy, which using EXI compression algorithm. To test this idea, all three IEEE1888 components are developed including 6LoWPAN sensor networks, 6LoWPAN gateway with built-in compression proxy and BEMS application on the Android OS. The proposed system is developed using two types of sensor nodes, e.g. Energy Metering node and Environment node. Energy Metering node is configured to measure real time voltage, current, power, energy, and frequency of an electric appliances. This node also allows remotely connect and disconnect the appliances from their power sources. Environment node is set up to measure humidity, temperature, luminance, and motion detection inside a room. This node is also equipped with a buzzer to warn, when energy usage exceeds a predefined limit. The BEMS application has been developed on Android operating system. The experimental site is located at Embedded System and IC Design Research Laboratory (ESID), Chulalongkorn University.

Department: Electrical Engineering Student's Signature .....
Field of Study: Electrical Engineering Advisor's Signature .....
Academic Year: 2013

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งของผู้ช่วยศาสตราจารย์ ดร.วันเฉลิม โปรา อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งท่านได้ให้คำแนะนำ ข้อคิดเห็นต่างๆ และกำลังใจในการทำวิทยานิพนธ์และงานวิจัยด้วยดีตลอดมา

ขอขอบคุณ อาจารย์ เพื่อนๆ พี่ๆ และน้องๆ ในห้องปฏิบัติการวิจัยการออกแบบและประยุกต์วงจรรวมทุกคนสำหรับความช่วยเหลือ คำแนะนำ และขอบคุณสำหรับมิตรภาพและความรู้สึกดี ๆ ที่มีให้กันมาโดยตลอด

สุดท้ายนี้ ข้าพเจ้าใคร่ขอกราบขอบพระคุณบิดา-มารดา อันเป็นที่เคารพรัก ที่คอยดูแลเอาใจใส่ และให้กำลังใจแก่ข้าพเจ้าด้วยดีเสมอมาจนกระทั่งมีวิทยานิพนธ์เล่มนี้

ได้รับการสนับสนุนทุนการศึกษาจาก ทุนอุดหนุนการศึกษาระดับบัณฑิตศึกษา จุฬาลงกรณ์มหาวิทยาลัย เพื่อเฉลิมฉลองวโรกาสที่พระบาทสมเด็จพระเจ้าอยู่หัวทรงเจริญพระชนมายุครบ 72 พรรษา บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1

บทนำ

1.1. แนวเหตุผลในการทำวิทยานิพนธ์

ปัจจุบันประเทศไทยกำลังประสบปัญหาทางด้านพลังงานไฟฟ้า เนื่องจากการเพิ่มจำนวนของประชากร และการพัฒนาทางด้านเทคโนโลยี ส่งผลให้เกิดปัญหาการขาดแคลนพลังงานไฟฟ้า และการทำลายทรัพยากร ซึ่งการผลิตพลังงานไฟฟ้าส่วนใหญ่ในปัจจุบันจะนำทรัพยากรอาทิเช่น ถ่านหิน หรือ ก๊าซธรรมชาติมาใช้ในการผลิต จึงเป็นเหตุผลให้ทรัพยากรเหล่านี้ลดน้อยลง ดังนั้นในการผลิตพลังงานไฟฟ้าทรัพยากรที่ใช้ผลิตแล้วย่อมหมดไป และ ยังทำให้ประเทศไทยต้องเสียเงินจำนวนมากมายในการนำเข้าพลังงาน

จากเหตุผลข้างต้นระบบจัดการพลังงานไฟฟ้าภายในอาคารจึงเข้ามามีบทบาทในการควบคุมการทำงานของเครื่องใช้ไฟฟ้าต่างๆที่ติดตั้งอยู่ภายในอาคาร เพื่อทำให้เกิดการใช้พลังงานไฟฟ้าอย่างมีประสิทธิภาพสูงสุด โดยจะรวมถึงการงดใช้และการใช้น้อยที่สุดเท่าที่จะไม่ทำให้ประสิทธิภาพการทำกิจกรรมอื่นๆ (Productivity) ในอาคารต้องเสียหายลง หรือก่อให้เกิดผลเสียทางสุขภาพใดๆกับผู้ใช้ในอาคาร [1] ซึ่งผลลัพธ์ของระบบจัดการพลังงานไฟฟ้าภายในอาคารจะทำให้ลดการใช้พลังงานไฟฟ้าลงได้

ระบบจัดการพลังงานไฟฟ้าภายในอาคารทั่วไปนั้นจะมีศูนย์กลางควบคุมที่ทำหน้าที่ในการเก็บข้อมูลต่างๆ อาทิเช่น การเคลื่อนไหว, พลังงานไฟฟ้า, สถานะของระบบปรับอากาศ (Heating Ventilation and Air Conditioning System, HVAC System), สภาพอากาศ, และข้อมูลอื่นๆที่สามารถใช้ในการควบคุมเครื่องใช้ไฟฟ้า ซึ่งระบบจัดการพลังงานไฟฟ้าจะต้องมีราคาในการติดตั้ง และซ่อมบำรุงที่น้อยที่สุด [2]

จะเห็นได้ว่าภายในอาคารมีอุปกรณ์มากมายหลายชนิด และผู้ผลิต การที่จะทำให้อุปกรณ์เหล่านี้สามารถที่จะส่งข้อมูลระหว่างกันได้จะต้องใช้มาตรฐานในการสื่อสาร หรือโพรโทคอล (Protocol) ชนิดเดียวกันในการสื่อสาร ซึ่งในวิทยานิพนธ์นี้จะเลือกใช้มาตรฐาน IEEE1888

มาตรฐาน IEEE 1888 เป็น มาตรฐานสำหรับระบบควบคุมผ่านเครือข่ายชุมชนสีเขียว (Standard for Ubiquitous Green Community Control Network, UGCC) IEEE1888 ได้กำหนดโครงสร้างพื้นฐานการสื่อสารที่มีจุดมุ่งหมายที่จะสร้างเครือข่ายรูปแบบใหม่ของระบบจัดการพลังงาน และการอนุรักษ์พลังงาน สำหรับอาคารขนาดเล็ก และขนาดกลาง โดยมาตรฐาน IEEE1888 ได้กำหนด แพลตฟอร์มการจัดการ, การตรวจสอบ, และการควบคุม โดยมีรูปแบบเป็นโพรโทคอลเปิด

การที่จะทำให้อุปกรณ์ทั้งหลายเหล่านี้สามารถสื่อสารบนมาตรฐาน IEEE1888 ได้นั้น จะต้องมีอุปกรณ์ที่เป็นจุดศูนย์กลางในการรวบรวมข้อมูลซึ่งมีชื่อเรียกว่าเกทเวย์ (Gateway) โดยจะช่วยในการเชื่อมต่อระบบการสื่อสารบนมาตรฐาน IEEE1888 ไปยังรูปแบบอื่นอาธิเช่น RS485, Lonworks, 1-Wire, BACnet, PLC, และ ZigBee เป็นต้น ดังนั้นในระบบจัดการพลังงานไฟฟ้าขนาดใหญ่จะมีเกทเวย์จำนวนมากจึงทำให้การช่องทางสื่อสารในระบบหนาแน่น และในระบบเครือข่ายที่มีค่าใช้จ่ายสูงเช่น 3G (3rd generation mobile telecommunication) จำเป็นต้องทำให้ข้อมูลที่ใช้ในการสื่อสารนั้นมีขนาดเล็กที่สุด เพื่อเป็นการลดค่าใช้จ่าย

งานวิจัยนี้จึงได้นำเสนอการพัฒนาเกทเวย์ที่สามารถทำการบีบอัดข้อมูลในมาตรฐาน IEEE1888 ซึ่งจะช่วยลดขนาดของข้อมูลที่ใช้ในการสื่อสารบนมาตรฐาน IEEE1888 และยังเหมาะสมสำหรับใช้ในช่องทางการสื่อสารที่มีราคาแพง เช่น การสื่อสาร 3G เป็นต้น

1.2. วัตถุประสงค์ของการวิจัย

1. พัฒนาเกทเวย์ (Gateway) ที่สื่อสารระหว่างมาตรฐาน IEEE1888 กับ มาตรฐาน 6LoWPAN และสามารถบีบอัดข้อมูล (Data Compression) ได้
2. ศึกษาเทคนิคที่ใช้ในการบีบอัดข้อมูลชนิดไม่มีสูญเสียที่เหมาะสมแก่การบีบอัดข้อมูลในมาตรฐาน IEEE1888
3. พัฒนาเซ็นเซอร์โหนด (Sensor noads) ในมาตรฐาน 6LoWPAN ที่ทำการวัดการใช้พลังงาน และสภาวะแวดล้อมภายในอาคาร
4. พัฒนาโปรแกรมประยุกต์ (Application) บนระบบปฏิบัติการ แอนดรอยด์ (Android) ที่ใช้ในระบบจัดการพลังงานไฟฟ้าภายในอาคารบนมาตรฐาน IEEE1888 เพื่อที่จะจำลองการทำงานของระบบจัดการพลังงานไฟฟ้า

1.3. ขอบเขตของการวิจัย

1. เขียนแอพพลิเคชั่นในระบบปฏิบัติการแอนดรอย์เพื่อจัดการพลังงานไฟฟ้าภายใน อาคาร ซึ่งมีคุณสมบัติดังต่อไปนี้
   - ใช้มาตรฐาน IEEE1888 ในชั้นโปรแกรมประยุกต์ (Application Layer)
   - ใช้ระบบเครือข่าย TCP/IP ในการสื่อสาร
   - มีส่วนติดต่อกับผู้ใช้งานในการแสดงผล และควบคุมการทำงาน
   - สามารถควบคุมการทำงานของเครื่องใช้ไฟฟ้าได้

2. ใช้ชุดพัฒนา 6LoWPAN รุ่น CC-6LOWPAN-DK-868 ของบริษัท Texas Instruments ในการสร้างเกทเวย์
3. เซ็นเซอร์โหนดสามารถส่งข้อมูลการใช้พลังงานไฟฟ้า และสภาวะแวดล้อม ไปยังเกทเวย์ได้
4. เกทเวย์สามารถเชื่อมต่อการสื่อสารระหว่างมาตรฐาน IEEE1888 กับ 6LoWPAN ได้
5. พัฒนาการบีบอัดข้อมูลในมาตรฐาน IEEE1888
6. ทดสอบการทำงานของแอพพลิเคชั่น และเกทเวย์ ในมาตรฐาน IEEE1888

1.4. วิธีดำเนินการวิจัย

1. ศึกษามาตรฐาน IEEE1888 ดังแสดงในหัวข้อที่ 2.1
2. ศึกษาการเขียนแอพพลิเคชั่นในระบบปฏิบัติการแอนดรอด์ ดังแสดงในหัวข้อที่ 4.5
3. ศึกษามาตรฐาน 6LoWPAN ดังแสดงในหัวข้อที่ 2.2
4. ศึกษาการทำงานของ SOAP (Simple Object Access Protocol) เว็บเซอร์วิส ดังแสดงในหัวข้อที่ 4.3
5. เขียนคลังโปรแกรมบริการในชั้นโปรแกรมประยุกต์ตามมาตรฐาน IEEE1888 ดังแสดงในหัวข้อที่ 4.5.1
6. เขียนแอพพลิเคชั่นในการติดต่อกับผู้ใช้งาน ดังแสดงในหัวข้อที่ 4.5
7. ศึกษาการเขียนโปรแกรมในชุดพัฒนา 6LoWPAN ดังแสดงในหัวข้อที่ 3.4
8. เขียนโปรแกรมเกทเวย์ เชื่อมต่อการสื่อสารระหว่างมาตรฐาน IEEE1888 กับ 6LoWPAN ดังแสดงในหัวข้อที่ 4.3
9. ศึกษาเทคนิคในการบีบอัดข้อมูล ดังแสดงในหัวข้อที่ 5.1
10. เขียนโปรแกรมบีบอัดข้อมูลบนมาตรฐาน IEEE1888 ดังแสดงในหัวข้อที่ 4.2
11. ทดสอบการทำงานของระบบ ดังแสดงในหัวข้อที่ 5.4
12. สรุปผลและเขียนรูปเล่มวิทยานิพนธ์

1.5. ลำดับขั้นตอนในการเสนอผลการวิจัย

วิทยานิพนธ์นี้แบ่งเนื้อหาออกเป็น 6 บท ดังต่อไปนี้ บทที่ 1 เป็นบทนำซึ่งกล่าวถึงแนวเหตุผลในการทำวิทยานิพนธ์ วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย วิธีดำเนินงานวิจัย รวมทั้งประโยชน์ที่คาดว่าจะได้รับ บทที่ 2 จะกล่าวถึง ความรู้พื้นฐานและหลักการที่เกี่ยวข้อง ซึ่งประกอบด้วย มาตรฐาน IEEE1888 ระบบเครือข่าย 6LoWPAN และ กระบวนการบีบอัดข้อมูล EXI บทที่ 3 กล่าวถึง การออกแบบสถาปัตยกรรมระบบเครือข่าย และฮาร์ดแวร์ ที่ทำการออกแบบให้ระบบเครือข่ายแต่ละประเภทมีชนิดของข้อมูลที่ใช้ในการสื่อสารแต่ต่างกันออกไป การออกแบบฮาร์ดแวร์ของ สตอเรจ เกทเวย โหนดวัดสภาวะแวดล้อม โหนดวัดพลังงาน และ แอพพลิเคชั่น บทที่ 4 กล่าวถึงการออกแบบซอฟต์แวร์ระบบ ซึ่งอธิบายถึงการทำงานร่วมกันของซอฟต์แวร์ภายในระบบ และการออกแบบซอฟต์แวร์ในส่วนต่างๆ ที่ประกอบไปด้วยซอฟต์แวร์ Proxy ที่ทำการบีบอัดข้อมูลในมาตรฐาน IEEE1888 โดยใช้กระบวนการบีบอัด EXI ซอฟต์แวร์แปลงรูปแบบข้อมูล IEEE1888 ไปยัง 6LoWPAN ซอฟต์แวร์ 6LoWPAN ในโหนดวัดสภาพแวดล้อม และโหนดวัดพลังงาน รวมถึงแอพพลิเคชั่นจัดการการใช้พลังงานภายในอาคารบนระบบปฏิบัติการ Android บทที่ 5 กล่าวถึง ผลการทดลองบีบอัดข้อมูล และแอพพลิเคชั่น ซึ่งจะได้ทำการเปรียบเทียบผลการบีบอัดข้อมูลในกระบวนการ EXI กับกระบวนการอื่นๆ ผลการบีบอัดข้อมูลที่ได้มาจาก Proxy ผลการทดสอบการใช้งานแอพพลิเคชั่นในการแสดงผลข้อมูล และควบคุมการทำงาน ท้ายที่สุด บทที่ 6 กล่าวถึง สรุปผลการวิจัย ว่าสามารถนำงานวิจัยนี้ไปพัฒนาต่อได้อย่างไรบ้าง มีสิ่งใดที่ทำแล้วไม่ประสบผลสำเร็จบ้าง และข้อเสนอแนะ จากการทำงานวิจัยนี้

บทที่ 2

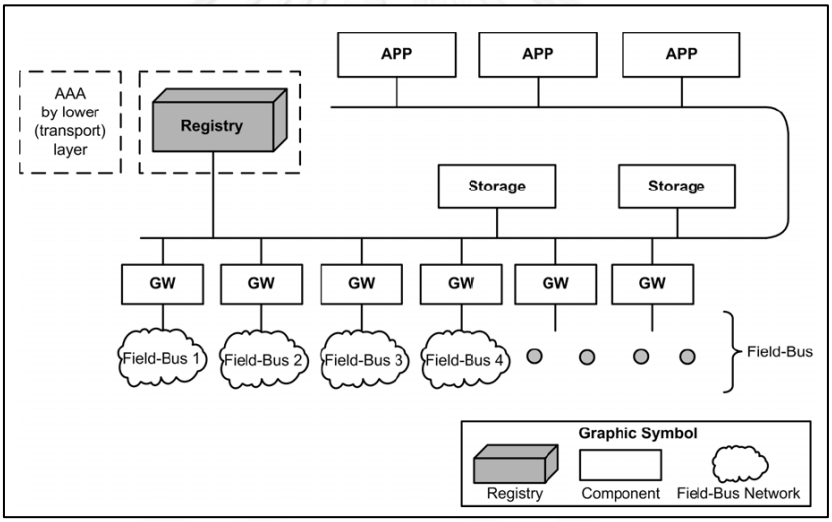
ความรู้พื้นฐาน และหลักการที่เกี่ยวข้อง

ในบทนี้ได้จะกล่าวถึงความรู้พื้นฐานและหลักการที่เกี่ยวข้องในการทำวิทยานิพนธ์ซึ่งประกอบด้วย มาตรฐาน IEEE1888 ระบบเครือข่าย 6LoWPAN และ กระบวนการบีบอัดข้อมูล EXI

2.1. มาตรฐาน IEEE1888

ในมาตรฐาน IEEE1888 ได้กำหนดสถาปัตยกรรมระบบเครือข่าย ลำดับการส่งข้อมูล โพรโทคอลการสื่อสาร และโครงสร้างข้อมูล ดังจะอธิบายดังหัวข้อย่อยต่อไปนี้

2.1.1. สถาปัตยกรรมระบบเครือข่าย



รูปที่ 2-1 สถาปัตยกรรมเครือข่าย

มาตรฐาน IEEE1888 [3] นั้นทำงานในชั้นโปรแกรมประยุกต์ (Application Layer) บนเครือข่ายแบบ TCP/IP โดยมีเป้าหมายหลักเพื่อที่จะสามารถแลกเปลี่ยนข้อมูลผ่านทางระบบเครือข่ายอย่างมีประสิทธิภาพ และอุปกรณ์หลากชนิด หลายยี่ห้อทำงานร่วมกันได้ มีองค์ประกอบหลักคือ เกทเวย์ (GW), สตอเรจ (Storage), และ แอปพลิเคชั่น (APP) ซึ่งมีอินเตอร์เฟซการสื่อสารเดียวกัน ส่วนรีจิสทรี (Registry) นั้นมีอินเตอร์เฟซการสื่อสารที่แตกต่างออกไปโดยจะแยกออกจากกลุ่มดังแสดงในรูปที่ 2-1

1. เกทเวย์ (Gateway, GW) เป็นส่วนประกอบที่มีหน้าที่ในเชื่อมต่ออุปกรณ์ต่างๆ ที่สื่อสารโดยใช้มาตรฐานอื่น เข้ากับระบบเครือข่าย IEEE1888 ยกตัวอย่างเช่น การอ่านค่าทางด้าน

กายภาพจากเซ็นเซอร์ ซึ่งโดยมากใช้พลังงานจากแบตเตอรี่ไม่เหมาะกับการสื่อสารตามมาตรฐาน IEEE1888 โดยตรง หรืออาจจะเป็นอุปกรณ์ที่มีใช้กันอยู่นานแล้วและใช้การสื่อสารแบบอื่น เกทเวย์ยังสามารถแปลงรูปแบบของข้อมูล IEEE1888 ไปยังรูปแบบข้อมูลตามมาตรฐานอื่นได้ด้วย เช่น MODBUS เพื่อเปิด/ปิดสวิทช์ เป็นต้น อนึ่ง อุปกรณ์ เซนเซอร์ หรือสวิทช์แต่ละตัว จะถูกเรียกว่า Point ในระบบ IEEE1888 Point แต่ละตัวจะมี URI ที่แตกต่างกัน ในแต่ละ Point อาจมีข้อมูลหลายประเภท ตัวอย่างเช่น Point ชนิดที่เป็นเตารีด อาจมีข้อมูลสถานะการเปิด/ปิด และข้อมูลอุณหภูมิ เป็นต้น ข้อมูลของ Point เหล่านี้จะมี URI เช่นกัน และถูกเรียกว่า PointSet

2. สตอเรจ (Storage) เป็นส่วนประกอบที่ทำหน้าที่ในการเก็บประวัติข้อมูลต่างๆ ของระบบอย่างมีลำดับ ข้อมูลที่เก็บนั้นจะมีโครงสร้างของข้อมูลตามมาตรฐาน IEEE1888 ข้อมูลที่เก็บอาจได้มาจาก เกทเวย์ หรือ แอพพลิเคชั่น ตัวอย่างเช่น ข้อมูลอุณหภูมิของห้องประชุม, ข้อมูลของการเคลื่อนไหว (motion detection), ข้อมูลการใช้พลังงานของเครื่องปรับอากาศ ข้อมูลการสั่งปิด/เปิด หรือหรี่หลอดไฟฟ้า เป็นต้น

3. แอพพลิเคชั่น (Application, APP) เป็นส่วนประกอบที่ติดต่อกับผู้ใช้ โดยการแสดงผลและรับคำสั่ง อ่านของมูลของเซ็นเซอร์ หรือสั่งการทำงาน โดยมียูสเซอร์อินเตอร์เฟส (User Interface) ที่แสดงสถานการณ์ของระบบซึ่งสามารถรับข้อมูลควบคุมการทำงานได้ เช่น เวลาในการเปิดปิดเครื่องใช้ไฟฟ้า อีกทั้งแอพพลิเคชั่นยังสามารถที่จะทำการวิเคราะห์ข้อมูลอ่านมาจากเซ็นเซอร์ในแบบเวลาจริง (Real time) อีกด้วย

4. รีจิสทรี (Registry) ทำหน้าที่นายทะเบียน เชื่อมการทำงานขององค์ประกอบแต่ละส่วนเข้าด้วยกันอย่างเหมาะสม และอิสระต่อกัน โดยคอยตรวจดูว่ามีอุปกรณ์ชนิดใดอยู่ในระบบบ้าง และให้บริการตอบข้อซักถามแก่ เกทเวย์ สตอเรจ หรือ แอพพลิเคชั่น เมื่อร้องขอรายการอุปกรณ์ตามคุณสมบัติ เช่น ชนิด หรือ ที่ตั้ง ของอุปกรณ์ เป็นต้น

2.1.2. ลำดับในการส่งข้อมูล

รูปที่ 2-2 แสดงลำดับของการทำงานของโพรโทคอลใน คอมโพเน้นท์ และรีจิสทรี โดยลูกศรทึบแสดงถึงการสื่อสารกันระหว่างคอมโพเน้นท์ไปยังคอมโพเน้นท์ ส่วนลูกศรประนั้นแสดงถึงการสื่อสารกันระหว่างคอมโพเน้นท์ไปยังรีจิสทรี

กรณี A: ลงทะเบียนส่วนประกอบต่างๆ (1) ลงทะเบียน Point IDs ทั้งหมดของเกทเวย์ (2) ลงทะเบียน Point IDs ทั้งหมดที่สตอเรจเก็บข้อมูล (3) ลงทะเบียน Point IDs ทั้งหมดที่แอพพลิเคชั่นอ่านข้อมูล และ มีไว้ใช้งาน

กรณี B: ค้นหา URI (Uniform Resource Identifier) ที่ต้องการโดยใช้ โพรโทคอลเรียกถาม (Query Protocol) โดยกำหนด Point

กรณี C: ค้นหา URI ของสตอเรจโดยระบุ Component

กรณี D: เกทเวย์ทำการเขียนข้อมูลที่ได้มาจากเซ็นเซอร์ไปยังสตอเรจ

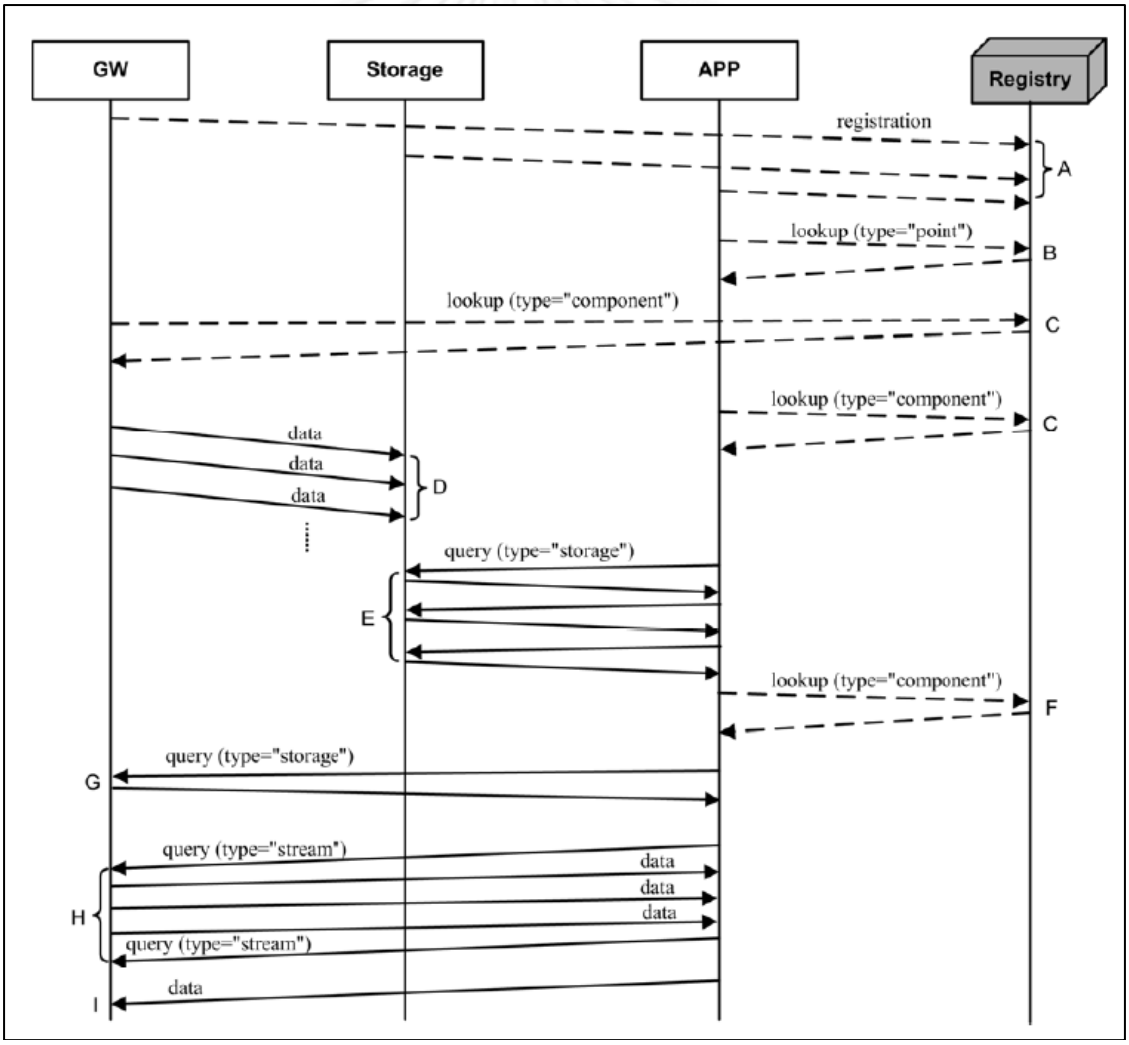
กรณี E: แอพพลิเคชั่นร้องขอข้อมูลจากสตอเรจ ซึ่งถ้าข้อมูลที่รับมีขนาดใหญ่ ข้อมูลนั้นควรที่จะถูกแบ่งเป็นส่วนๆ แล้วค่อยถูกส่งออกไปอย่างเป็นลำดับ

กรณี F: ค้นหา URI ของเกทเวย์โดยระบุ Component

กรณี G: แอพพลิเคชั่นกำหนดเหตุการณ์เกทเวย์

กรณี H: แอพพลิเคชั่นส่งข้อมูลของเหตุการณ์ไปยังเกทเวย์อย่างเป็นระยะ จากนั้นเกทเวย์ตอบรับการอัพเดทมายังแอพพลิเคชั่น

กรณี I: แอพพลิเคชั่นสั่งการทำงานไปยังเกทเวย์



รูปที่ 2-2 ลำดับในการส่งข้อมูลในการสื่อสาร

2.1.3. โพรโทคอลการสื่อสาร

มาตรฐานนี้ได้กำหนดโพรโทคอลในการสื่อสารระหว่างคอมโพเน้นท์กับคอมโพเน้นท์ โดยให้นำ Simple Object Access Protocol [4] มาใช้ในการส่งข้อความระหว่างคอมโพเน้นท์กับคอมโพเน้นท์

2.1.3.1. โพรโทคอลที่ใช้ในการสื่อสารระหว่างคอมโพเน้นท์กับคอมโพเน้นท์

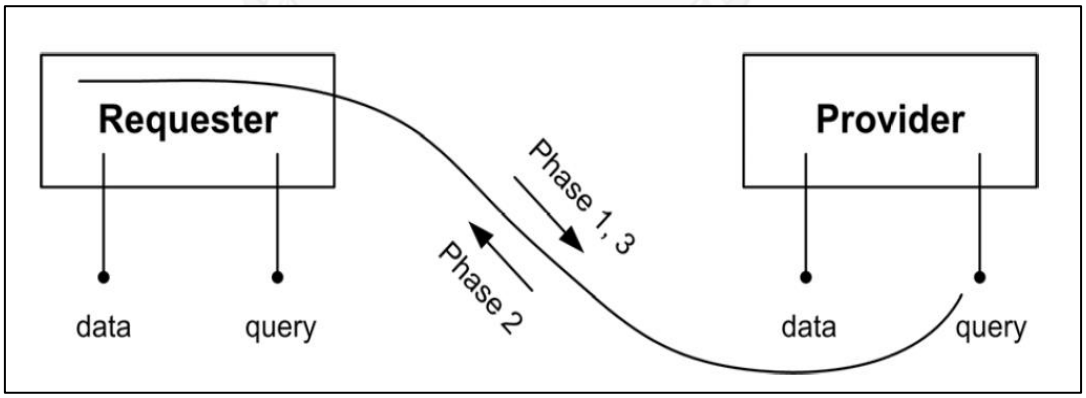
ในส่วนนี้จะได้อธิบายถึงโพรโทคอลย่อยในการสื่อสารระหว่างคอมโพเน้นท์กับคอมโพเน้นท์ซึ่งประกอบไปด้วย 3 โพรโทคอลดังนี้

- *โพรโทคอล FETCH* ใช้สำหรับร้องขอข้อมูลจากคอมโพเน้นท์ปลายทาง
- *โพรโทคอล WRITE* ใช้สำหรับเขียนข้อมูลไปยังคอมโพเน้นท์ปลายทาง
- *โพรโทคอล TRAP* ใช้สำหรับแจ้งให้คอมโพเน้นท์ปลายทางให้บันทึกชนิดเหตุการณ์ที่สนใจต่อเมื่อเหตุการณ์นั้นเกิดขึ้นให้ปลายทางเขียนข้อมูลกลับมายังต้นทางทันทีโดยไม่ต้องร้องขอ

รายละเอียดของโพรโทคอลทั้งสามจะกล่าวในหัวข้อย่อยถัดไป

2.1.3.2. โพรโทคอล FETCH

โพรโทคอล FETCH ใช้สำหรับร้องขอข้อมูลจากคอมโพเน้นท์ปลายทาง ซึ่งในรูปที่ 2-3 แสดงถึงแผนภาพในการรับส่งข้อมูล ประกอบไปด้วย*ผู้ร้องขอ* (Requester) และ*ผู้ให้* (Provider)



รูปที่ 2-3 ขั้นตอนการรับส่งข้อมูลของโพรโทคอล FETCH

Phase1: ผู้ร้องขอเริ่มการร้องขอข้อมูล โดยการส่ง Point และช่วงเวลาของข้อมูลที่ต้องการไปยังผู้ให้ และควรที่จะแจ้งขนาดสูงสุดของที่เก็บข้อมูลชั่วคราว (Buffer) ไปด้วยแต่ถ้าหากไม่มีการแจ้ง ผู้ให้จะใช้ค่ากลางคือ หนึ่งร้อยข้อมูล (100 Value)

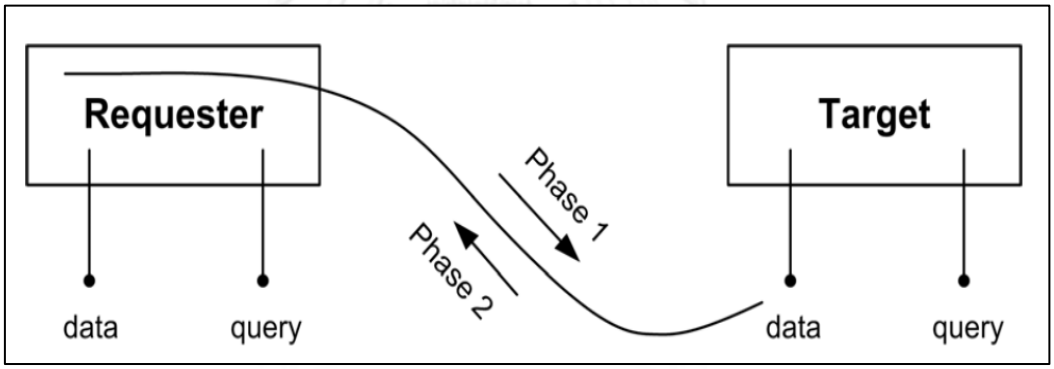
Phase2: ผู้ให้ส่งข้อมูลกลับมายังผู้ร้องขอ ข้อมูลอาจถูกแบ่งออกเป็นชุดย่อยๆ ในกรณีที่ขนาดของข้อมูลที่จะส่งใหญ่เกินกว่าผู้ร้องขอจะรับได้ หรือใหญ่จนทำให้ผู้ให้จะต้องใช้ทรัพยากรในการคำนวณมากเกินไป ในแต่ละชุดข้อมูลย่อย ผู้ให้จะส่ง*ตำแหน่งของข้อมูล* (Cursor) ในชุดถัดไปให้กับผู้ร้องขอด้วย ยกเว้นมีการส่งข้อมูลเพียงชุดเดียว หรือส่งชุดข้อมูลย่อยเป็นชุดสุดท้าย

Phase3: ถ้าหากผู้ร้องขอได้รับตำแหน่งของข้อมูลจาก Phase2 แสดงว่าข้อมูลยังไม่สิ้นสุด ผู้ร้องขอจะต้องร้องขอข้อมูลพร้อมกับส่งตำแหน่งของข้อมูลไปยังผู้ให้อีกครั้ง และกลับไปทำงานใน Phase2

ถ้าหากผู้ร้องขอไม่ได้รับตำแหน่งของข้อมูลจาก Phase2 แสดงว่าข้อมูลที่ได้รับมาเป็นข้อมูลชุดสุดท้าย ดังนั้นจึงสิ้นสุดการทำงานของ FETCH โพรโทคอล

2.1.3.3. **โพรโทคอล WRITE**

โพรโทคอล WRITE ใช้สำหรับโอนถ่ายข้อมูลไปยังคอมโพเน้นท์ปลายทาง รูปที่ 2-4 แสดงถึงแผนภาพในการรับส่งข้อมูล ประกอบไปด้วย*ผู้ร้องขอ* และ*เป้า* (Target)



รูปที่ 2-4 ขั้นตอนการรับส่งข้อมูลของโพรโทคอล WRITE

Phase1: ผู้ร้องขอส่งข้อมูลที่ต้องการจะเขียนไปยังเป้า

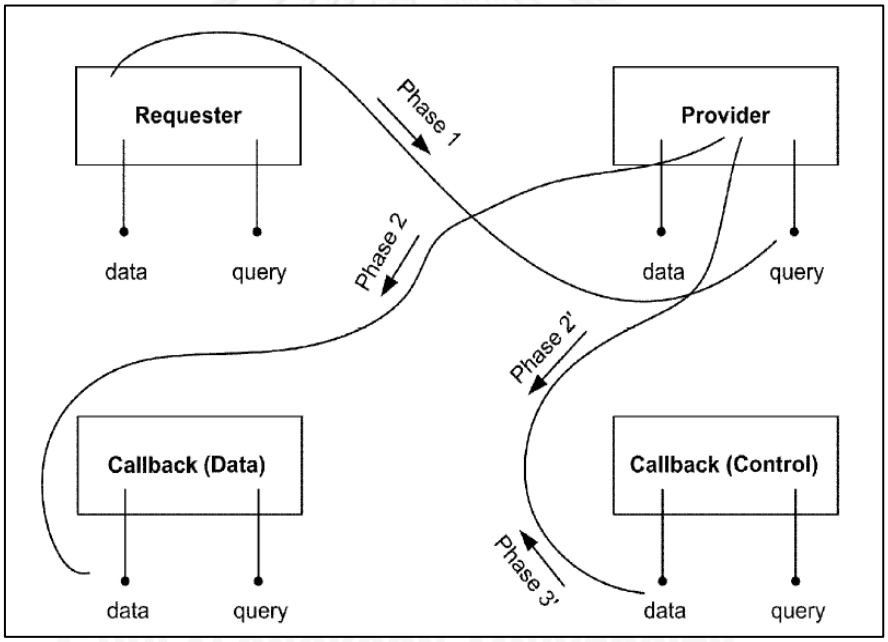
Phase2: เป้าตอบกลับไปยังผู้ร้องขอว่าข้อมูลที่ได้รับมาถูกต้องสมบูรณ์ หรือเกิดข้อผิดพลาดอะไรขึ้น

2.1.3.4. **โพรโทคอล TRAP**

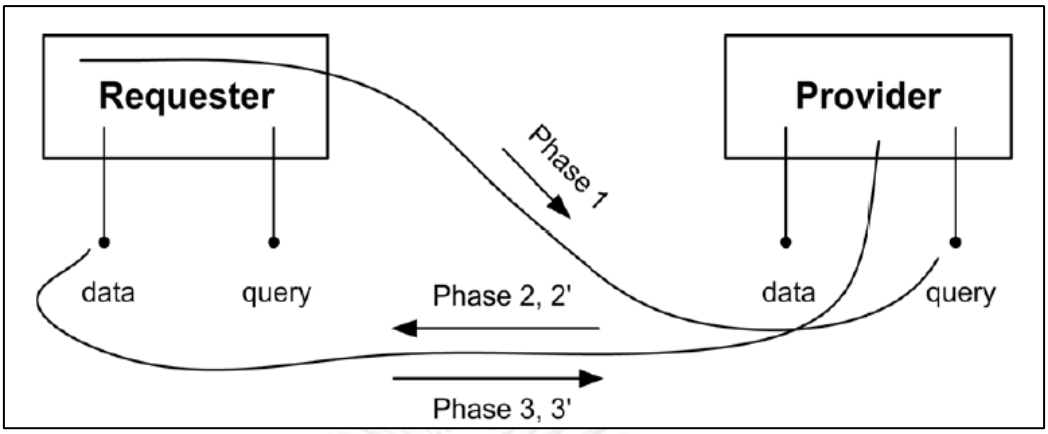
โพรโทคอล TRAP ใช้สำหรับการบันทึกเหตุการณ์ที่ต้องการแจ้งเตือน และโอนถ่ายข้อมูลเมื่อเกิดเหตุการณ์ โดยได้ระบุชื่อของคอมโพเน้นต่างๆดังนี้

- **ผู้ร้องขอ คือ คอมโพเน้นที่ส่งเหตุการณ์ที่ต้องการแจ้งเตือนไปยังผู้ให้**
- **ผู้ให้ คือ คอมโพเน้นที่ส่งข้อมูลเมื่อเกิดเหตุการณ์ที่ตรงกับที่ได้บันทึกไว้**
- **ผู้ได้รับการตอบข้อมูล (Callback (Data)) คือ คอมโพเน้นที่ได้รับข้อมูลจาก ผู้ให้เมื่อเกิดเหตุการณ์**
- **ผู้ได้รับการตอบการควบคุม (Callback (Control)) คือ คอมโพเน้นที่ได้รับสัญญาณควบคุม จากผู้ให้เมื่อเกิดเหตุการณ์**

จากรูปที่ 2-5 แสดงให้เห็นถึงการทำงานร่วมกันของคอมโพเน้นต่างๆ ทั้งหมด แต่ในทางปฏิบัติผู้ร้องขอ ผู้ได้รับการตอบข้อมูล และผู้ได้รับการตอบการควบคุม มักจะเป็นคอมโพเน้นเดียวกันดังแสดงการทำงานในรูปที่ 2-6 ขั้นตอนการรับส่งข้อมูลอธิบายได้ดังนี้



รูปที่ 2-5 ขั้นตอนการรับส่งข้อมูลของโพรโทคอล TRAP (รูปแบบการนำไปใช้งานแบบทั่วไป)



รูปที่ 2-6 ขั้นตอนการรับส่งข้อมูลของโพรโทคอล TRAP (รูปแบบการนำไปใช้งานในทางปฏิบัติ)

Phase1: ผู้ร้องขอส่งคำขอบันทึกเหตุการณ์ที่ต้องการแจ้งเตือนไปยังผู้ให้ ซึ่งจะต้องกำหนดเวลาหมดอายุของการเฝ้าดูเหตุการณ์ในหน่วยวินาทีไว้ด้วย พร้อมกับ URI ของผู้ได้รับการตอบข้อมูล และ URI ของผู้ได้รับการตอบการควบคุม

Phase2: ผู้ให้ตอบกลับไปยังผู้ได้รับการตอบข้อมูล เมื่อเกิดเหตุการณ์ที่ตรงกับที่ได้บันทึกไว้ในเวลาที่กำหนด

Phase3: ผู้ได้รับการตอบข้อมูล ตอบกลับไปยังผู้ให้ ว่าข้อมูลที่ได้รับมาถูกต้อง หรือเกิดข้อผิดพลาดขึ้น

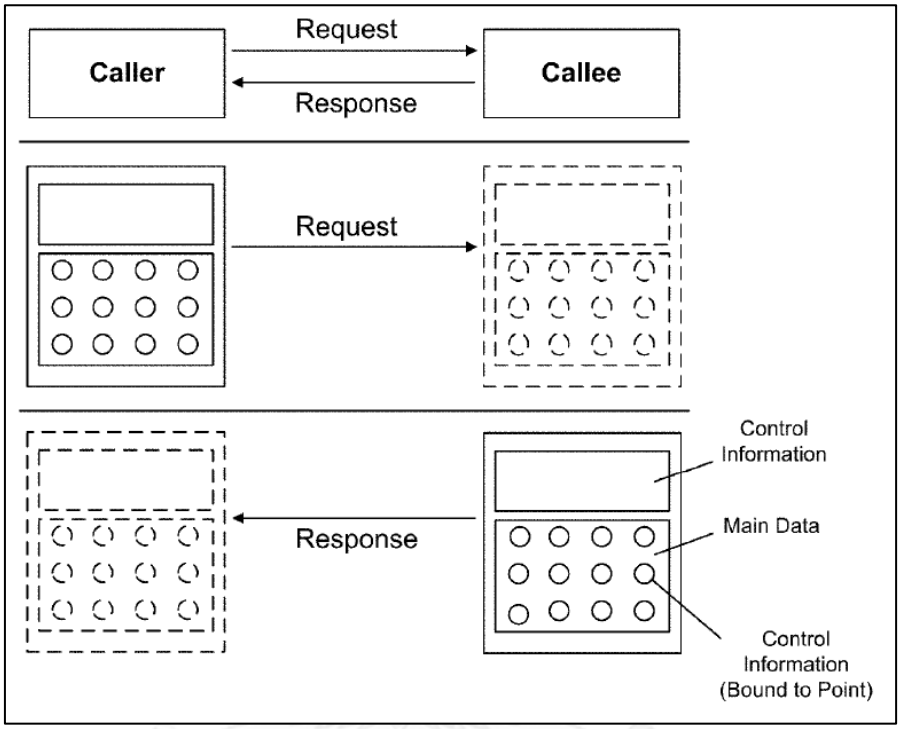
Phase2': ถ้าหากใน Phase3 เกิดความผิดพลาดขึ้นอาจเกิดจากความผิดพลาดในช่องทางการสื่อสารระหว่างผู้ให้กับผู้ได้รับการตอบข้อมูลผู้ให้ จะแจ้งความผิดพลาดไปยังผู้ได้รับการตอบการควบคุม

Phase3': ผู้ได้รับการตอบการควบคุมจะตอบกลับไปยังผู้ให้ทราบว่าว่าข้อมูลที่ได้รับมาใน Phase2' ถูกต้อง หรือเกิดข้อผิดพลาดขึ้น เพื่อยืนยันว่าช่องทางการสื่อสารอื่นใช้งานไม่ได้ด้วยหรือไม่

2.1.4. โครงสร้างของข้อมูลที่ใช้ในการสื่อสาร

การสื่อสารระหว่างคอมโพเน้นท์กับคอมโพเน้นท์นั้นได้นำโพรโทคอล RPC (Remote Procedure Call) มาใช้งาน ดังแสดงในรูปที่ 2-7 ซึ่ง *ผู้เรียก* (Caller) จะส่งข้อมูลที่เรียกว่า Requester-data ไปยัง *ผู้ถูกเรียก* (Callee) และ ผู้ถูกเรียกจะตอบกับโดย Response-data ไปยังผู้เรียก โครงสร้างข้อมูลของ Requester-data และ Response-data มีลักษณะเดียวกัน จึงขอเรียกรวมว่าข้อมูล RPC

ข้อมูล RPC นั้นจะถูกแบ่งออกเป็น 2 ส่วนคือ ส่วนหัว (Header Part) และ ส่วนใจความ (Body Part) โดยส่วนหัวจะประกอบไปด้วยข้อมูลที่ทำการควบคุม ตัวอย่างเช่น ข้อมูลการร้องขอ ข้อมูลแสดงการรับทราบ (Acknowledgement) และข้อมูลแสดงความผิดพลาด เป็นต้น ส่วนใจความจะประกอบไปด้วยออบเจ็ค (Object) ของ Point หรือ PointSet ที่มีค่า URI พร้อมด้วยข้อมูล ตัวอย่างเช่น ข้อมูลที่อ่านมาจากเซ็นเซอร์ เป็นต้น

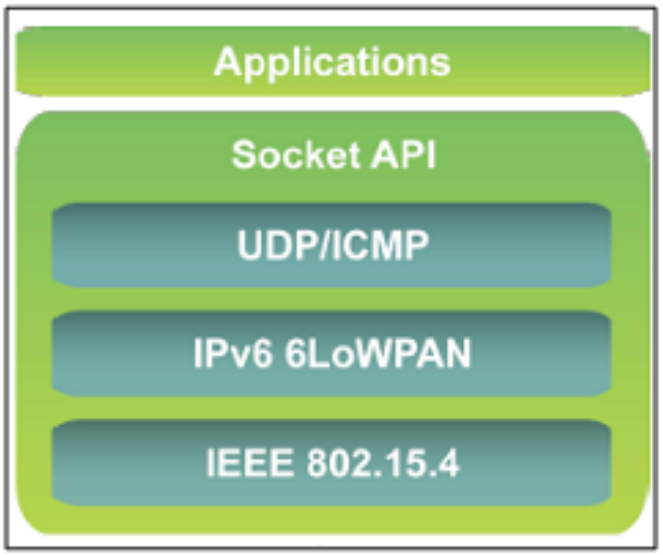


รูปที่ 2-7 โครงสร้างของข้อมูลที่ในในการสื่อสาร

2.2. ระบบเครือข่าย 6LoWPAN

6LoWPAN เป็นชื่อย่อของ IPv6 over Low Power Wireless Personal Area Network ชื่อนี้ได้มาจากกลุ่มผู้ทำงานใน Internet Engineering Task Force (IETF) ได้พัฒนาให้ใช้ IPv6 บน การสื่อสารที่มีชั้น Physical และ Datalink ตามมาตรฐาน IEEE 802.15.4 [5] (ที่กำหนดไว้ใน RFC4919 [6] ด้วย) ซึ่ง IEEE 802.15.4 กำหนดให้การสื่อสารเป็นเครือข่ายส่วนตัว (Personal Area Network) แบบไร้สาย (Wireless) และกินไฟต่ำ (Low Power)

กลุ่ม 6LoWPAN มุ่งเป้าที่จะกำหนดกลไกการบีบอัดในส่วนหัวที่อนุญาตให้ IPv6 ส่งแพ็กเก็ต และรับสัญญาณได้จากพื้นฐานของ IEEE 802.15.4 ได้ทั้งหมด ชั้นโพรโทคอลสำหรับเครือข่าย 6LoWPAN ถูกสร้างขึ้นในแนวทางที่คล้ายกับชั้นโพรโทคอลของอินเทอร์เน็ตทั่วไปมาก ดังแสดงในรูปที่ 2-8



รูปที่ 2-8 ลำดับชั้นการทำงาน 6LoWPAN

2.2.1. ลักษณะเด่นของ 6LoWPAN

- **แพ็กเก็ตขนาดเล็ก ขนาดแพ็กเก็ตที่ใหญ่ที่สุดของชั้น Physical ของ IEEE 802.15.4 คือ 128 bytes ในแต่ละชั้นจะมีข้อมูลที่เป็นส่วนหัวอยู่ กรณีที่แย่ที่สุด IEEE 802.15.4 จะเหลือที่ไว้เพียง 81 Byte สำหรับแพ็กเก็ตของชั้น IP**

- **สนับสนุน Address ทั้งแบบ 16 บิตซึ่งเป็นเอกลักษณ์ของ PAN (Personal Area Network) และแบบ IEEE 64 บิตซึ่งถูกใช้เป็น MAC address**

- **ใช้แบนด์วิดท์ต่ำ อัตราการรับส่งข้อมูลอาจเป็น 250 kbps, 40 kbps หรือ 20 kbps (กำหนดไว้ในชั้น Physical)**

- **มักจะใช้กับอุปกรณ์ที่กินพลังงานต่ำ และ มีราคาถูก ใช้งานในปริมาณมากในคราวเดียว เช่น Network Sensor**

- **ตำแหน่งของอุปกรณ์มักจะไม่ถูกกำหนดไว้ก่อนและนอกจากนี้อุปกรณ์เหล่านี้สามารถเคลื่อนที่ไปยังตำแหน่งใหม่ได้**

- **อุปกรณ์ 6LoWPAN สามารถเข้าโหมด sleep ได้เป็นเวลานานเมื่อต้องการประหยัดพลังงาน สามารถแบ่งเป็น 2 กลุ่ม ตามมาตรฐาน IEEE 802.15.4 คือ**

1. FFD (Full Function Device) เป็นอุปกรณ์ (หรือเรียกว่าโหนดก็ได้) ที่จะตื่นตลอดเวลา มักใช้กับอุปกรณ์ที่มีแหล่งพลังงานจากภายนอก สามารถใช้งานเป็นตัวประสานงาน

(coordinator) ของ PAN หรือเป็นโหนดทั่วไปก็ได้ หากทำงานเป็นตัวประสานงานจะเป็นตัวเชื่อมเครือข่ายติดต่อสื่อสารให้กับโหนดอื่นๆ โดยจะถ่ายทอดข้อความ จากโหนดลูกข่ายต่างๆ ไปยังโหนดแม่ข่ายของมัน

2. RFD (Reduced Function Devices) เป็นโหนดที่ติดต่อสื่อสารเฉพาะกับตัวประสานงานและตัวมันเองไม่สามารถทำงานเป็นตัวประสานงานได้ มักใช้กับอุปกรณ์ที่ใช้แหล่งพลังงานของตัวเอง เช่น แบตเตอรี่ หรือที่มีทรัพยากรต่ำ เช่น microcontroller อย่างง่าย ขนาดหน่วยความจำเล็ก เป็นต้น

2.2.2. เทคโนโลยี IP ในเครือข่ายส่วนตัวพลังงานต่ำ (LoWPAN)

การใช้งานเทคโนโลยี IP นั้นคาดว่าจะให้ผลประโยชน์ดังต่อไปนี้

- **ความแพร่หลายอย่างมากของเครือข่าย IP ทำให้ใช้โครงสร้างพื้นฐานที่มีอยู่อย่างกว้างขวางอยู่แล้ว**
- **เทคโนโลยีพื้นฐานของ IP ที่มีตัวตนแล้วทำให้เป็นที่รู้จักเปิดกว้างและใช้ได้ฟรี พิสูจน์แล้วว่าสามารถใช้งานได้จริง**
- **เครื่องมือสำหรับการวิเคราะห์ การจัดการ ของเครือข่าย IP มีปริมาณมาก**
- **อุปกรณ์พื้นฐานที่รองรับ IP สามารถเชื่อมต่อในระบบเครือข่ายอินเทอร์เน็ตอื่นๆ ได้อย่างง่ายโดยไม่ต้องการหน่วยงานระดับกลาง เหมือนเกตเวย์แปล (translation gateways) หรือพรอกซี่ (proxies) (เช่น GW ของ IEEE1888)**

การใช้งานอินเทอร์เน็ตบน IEEE 802.15.4 มีข้อควรพิจารณาอยู่หลายข้อที่แตกต่างจากผู้ใช้อุปกรณ์อินเทอร์เน็ตบนเทคโนโลยีชนิดอื่นๆ เช่น อีเทอร์เนต (IEEE 802.3) หรือ Wi-Fi (IEEE 802.11) ตัวอย่างเช่น

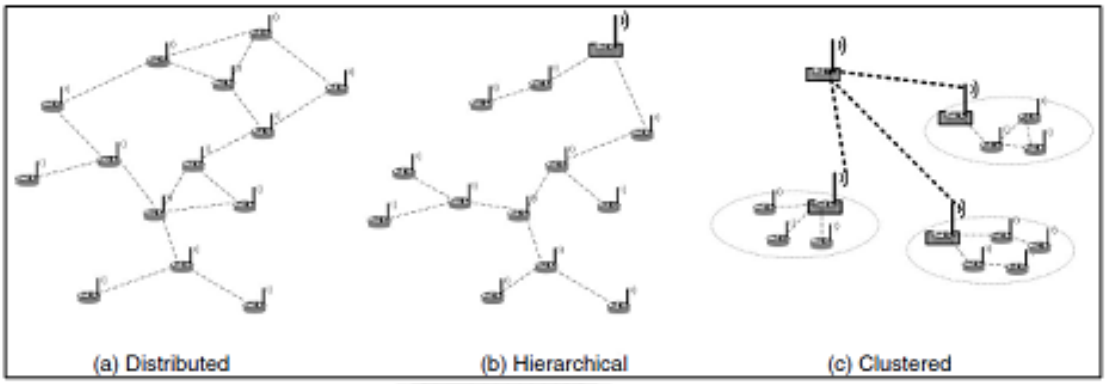
- **อุปกรณ์หลายอย่างใน เครือข่ายส่วนตัวพลังงานต่ำ ทำให้ระบบเครือข่ายต้องกำหนดค่า IP Address ให้อัตโนมัติ ซึ่งเป็นความสามารถหนึ่งของ IPv6**
- **ถ้ามีอุปกรณ์จำนวนมากจำเป็นต้องใช้ IP address จำนวนมากเช่นกัน การใช้ IPv6 ทำให้มี IP address เพิ่มขึ้นมาก เพียงพอต่อจำนวนอุปกรณ์**
- **ขนาดของแพ็กเก็ตที่จำกัด (81 bytes ในกรณีที่แย่ที่สุด) เฮดเดอร์ของ IPv6 และ**
- **เลเยอร์บนจำเป็นต้องมีการบีบอัดไม่ทางใดก็ทางหนึ่ง**

- **ขนาดของเครือข่ายไม่ได้ครอบคลุมพื้นที่มาก**
- **ข้อมูลในชั้น Application มักมีขนาดเล็ก**

ดังนั้น 6LoWPAN จึงกำหนดให้ใช้ IPv6 ในชั้น Protocol แต่ใช้ UDP ในชั้น Network

2.2.3. การพิจารณาเส้นทาง (Routing Considerations)

เครือข่ายไร้สายส่วนตัวพลังงานต่ำ (LoWPAN) สนับสนุนโทโพโลจี ต่างๆ ดังแสดงรูปที่ 2-9 แสดงสถาปัตยกรรมเครือข่ายเซนเซอร์ในโทโพโลจีกระจาย (Distributed), โทโพโลจีลำดับชั้น (Hierachical) และโทโพโลจีกลุ่มย่อย (Clustered)



รูปที่ 2-9 แสดงสถาปัตยกรรมเครือข่ายเซนเซอร์

การพิจารณาใน Routing โพรโทคอล ได้แก่

- **Routing โพรโทคอลต้องกำหนดค่าใช้จ่ายต่ำหรือไม่ ในข้อมูลแพ็กเก็ต**
- **เพื่อให้มีค่าใช้จ่ายต่ำ และพลังงานต่ำ จะต้องใช้การคำนวณ และหน่วยความจำที่น้อยที่สุด ดังนั้นการจัดการกับตาราง routing ขนาดใหญ่จึงเป็นสิ่งที่อันตราย**
- **การจัดการที่เหมาะสมในกรณีของการมีโหนดที่หลับเพื่อประหยัดพลังงาน**

2.2.4. หน้าที่ (Functions) ของ LoWPAN

เป้าหมายหลักของ LoWPAN คือการลดค่าใช้จ่ายของแพ็กเก็ต แบนด์วิดท์ และความต้องการในการประมวลผล

2.2.4.1. การปรับขนาดแพ็กเก็ต

เพื่อประหยัดพลังงาน และลดการชนกันของขแพ็กเก็ต ขนาดของแพ็กเก็ตจึงถูกกำหนดให้เป็น 128 octets ซึ่งมีขนาดต่ำกว่าค่ากลางของ แพ็กเก็ต IPv6 ปกติคือ 1,280 octets อยู่มาก จึงต้องมีการทำ*แฟรกเมนต์เตชั่น* (Fragmentation)

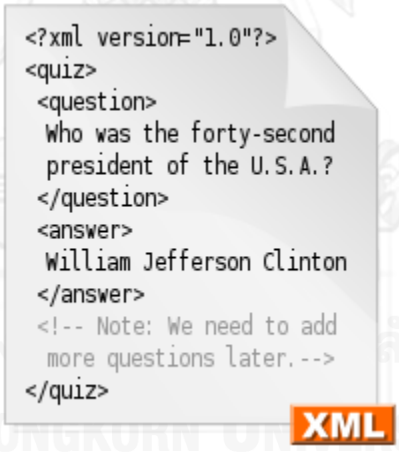
2.2.4.2. การบีบอัดเฮดเดอร์ (Header Compression)

ในชั้น IP อาจมีที่เหลือเพียง 81 octets และส่วนหัวของ IPv6 จะกินที่ 40 octets เหลือ 41 octets ชั้น Network ที่เลือกใช้ โพรโตคอล UDP ซึ่งใช้ส่วนหัว 8 octets และ ทำการแฟรกเมนท์เตชั่นหรือสร้างเลเยอร์ใหม่จะใช้ octets มากขึ้นกว่าเดิม ทำให้เหลือ bytes ไว้น้อยมาก สำหรับใช้ในการบรรจุข้อมูลชั้นประยุกต์ จุดนี้ทำให้เกิดความต้องการในการบีบอัดเฮดเดอร์

2.2.4.3. ความละเอียดของ Address

IPv6 โหนดถูกกำหนด IP address ไว้ 128 บิต 6LoWPAN ถูกระบุวิธีการสำหรับสร้าง IPv6 แบบ Stateless เพื่อลดค่าใช้จ่ายในการตั้งค่าผ่านโฮสท์ (host)

2.3. กระบวนการบีบอัดข้อมูล EXI



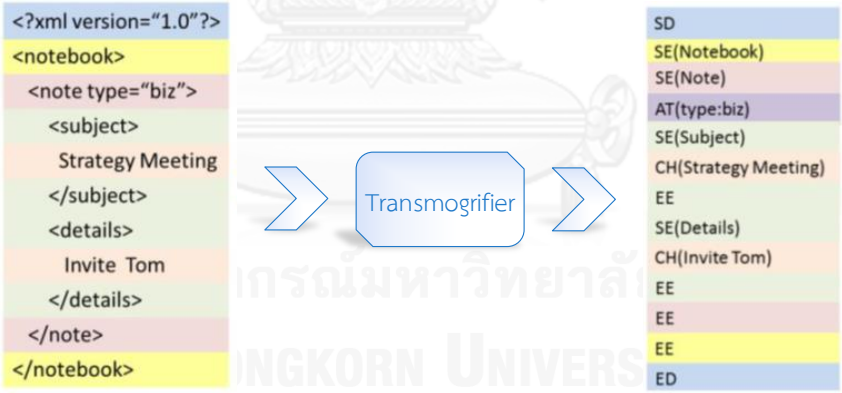
รูปที่ 2-10 ข้อมูลในรูปแบบ XML

XML [7] มีโครงสร้างที่ไม่ซับซ้อนทำให้ง่ายต่อการอ่าน ยืดหยุ่นต่อการใช้งาน และมีการประยุกต์ใช้งานที่หลากหลายดังแสดงในรูปที่ 2-10 ไฟล์ XML นั้นมีขนาดเล็ก สามารถใช้งานแลกเปลี่ยนข้อมูลได้ในหลาย Platform โดยไม่ต้องเปลี่ยนแปลงข้อมูล แต่การที่ทำให้ XML เป็นข้อมูลที่มีรูปแบบลำดับชั้นสวยงาม สะดวกต่อการอ่านของมนุษย์นั้น จะต้องส่งข้อมูลโสหุ้ยที่เรียกว่าแท็ก (tag) ซึ่งเป็นข้อความที่อยู่ระหว่างเครื่องหมายน้อยกว่ากับมากกว่า เช่น <question> เป็นจำนวนมาก

คอมพิวเตอร์ไม่ต้องการชื่อแท็ก ที่ยาวสำหรับระบุลำดับชั้นข้อมูล นอกจากนี้ คอมพิวเตอร์สามารถคาดการณ์ล่วงหน้า และอาจประมวลผลข้อมูลได้โดยปราศจากสิ่งเหล่านี้ได้ ยกตัวอย่างเช่น tag ที่ใช้ปิด <question> ย่อมเป็น tag <\question> เสมอ การบีบอัดโดยการส่งข้อมูลที่สำคัญที่สุด และสิ่งที่มนุษย์สามารถอ่านได้ถูกแทนที่ด้วยการเข้ารหัสที่รัดกุมทำให้ขนาดของข้อมูลลดลงได้มาก อีกทั้งสามารถส่งจากคอมพิวเตอร์เครื่องหนึ่งไปยังอีกเครื่องด้วยความเร็วที่มากกว่า อาจประมวลผลได้รวดเร็วกว่า และหากเลือกกระบวนวิธีที่เหมาะสม ที่คอมพิวเตอร์ปลายทางสามารถเรียกคืนข้อมูลต้นฉบับโดยไม่เกิดความเสียหาย

EXI [8] เป็นมาตรฐานการบีบอัดข้อมูลสำหรับรับส่ง จัดเก็บ หรือแลกเปลี่ยน XML ที่มีประสิทธิภาพ ซึ่ง*องค์กรระหว่างประเทศที่ทำงานด้านการพัฒนาเทคโนโลยีเว็บ* World Wide Web Consortium (W3C) สนับสนุนให้ใช้งานเพื่อการนี้ กระบวนการบีบอัดของ EXI ได้กำหนดกลยุทธ์อย่างรัดกุมมากสำหรับการอธิบายโครงสร้างของข้อมูล XML ไว้สองขั้นตอน ขั้นตอนแรก EXI ได้อธิบายวิธีการที่ใช้สำหรับจัดกลุ่มข้อมูลให้อยู่ในรูปแบบ “channels” ซึ่งจะรวมค่าต่างๆ ให้ได้มากที่สุดเท่าที่จะเป็นไปได้ภายในความยาวข้อมูลที่น้อยที่สุด ในขั้นตอนที่สอง ข้อมูลเหล่านี้จะถูกบีบอัดโดยใช้กระบวนการบีบอัด Deflate [9] เพื่อที่จะทำให้ข้อมูลเหล่านี้เล็กมากที่สุดเท่าที่จะเป็นไปได้ การดำเนินการตามกระบวนการทั้งสองนี้จะได้ข้อมูลที่เล็กกว่ากระบวนการบีบอัดไฟล์ทั่วไป เช่น GZip หรือ กระบวนการอื่นๆ ที่ไม่ได้พิจารณาถึงโครงสร้างของ XML

2.3.1. Default Processing



รูปที่ 2-11 การแปลงข้อมูล XML ให้อยู่ในรูปแบบ EXI

ขั้นตอนแรกของการบีบอัดตามมาตรฐาน EXI เรียกว่า Default Processing ซึ่งจะแปลงข้อมูล XML ในรูปแบบปกติให้อยู่ในรูปแบบ EXI โดยละเลยข้อมูลบางส่วนที่ไม่จำเป็นดังแสดงในรูปที่ 2-11 การเริ่มต้นข้อมูลจะถูกแทนด้วยเครื่องหมาย SD (Start Document) และ การสิ้นสุดข้อมูลจะถูกแทนด้วยเครื่องหมาย ED (End Document) จากนั้นแท็กเริ่มต้นจะถูกแทนด้วยเครื่องหมาย SE (Start Element) ซึ่งถ้าหากใช้แท็กในชื่อนั้นครั้งแรกจะต้องเก็บข้อมูลของชื่อไปด้วย แต่ถ้าหากมีการใช้งานแท็กในชื่อนั้นแล้วให้อ้างไปยังแท็กนั้นที่ถูกใช้งานในครั้งแรก เช่นเดียวกับ

คุณสมบัติจะถูกแทนด้วยเครื่องหมาย AT (Attribute) ข้อมูลที่เป็นตัวอักษรจะถูกแทนด้วยเครื่องหมาย CH (Character) และ แท็กสิ้นสุดข้อมูลจะถูกแทนด้วยเครื่องหมาย EE (End Element) แท็กสิ้นสุดข้อมูลจะไม่มีการเก็บชื่อ เนื่องจากเราสามารถรู้ได้ว่าแท็กสิ้นสุดข้อมูลนี้คู่กับแท็กเริ่มข้อมูลที่ใกล้ที่สุด ความยาวเครื่องหมายที่นำมาแทนนั้นมีขนาดเพียง 2 - 6 บิต ขึ้นอยู่กับความถี่ซึ่งมาตรฐานได้กำหนดไว้ตายตัวแล้ว หลังจากที่ได้ข้อมูลในรูปแบบ EXI ในขั้นตอนที่สองข้อมูลนี้จะถูกบีบอัดด้วยกระบวนการ DEFLATE เพื่อทำให้ข้อมูลมีขนาดเล็กที่สุด

2.3.1.1. **EXI ร่วมกับ XSD**

กระบวนการ EXI ยังรองรับการใช้งานร่วมกับ XSD (XML Schema Definition) ซึ่งแยกเก็บชื่อแท็ก และโครงสร้างข้อมูล XML อีกทั้งยังตระหนักถึงชนิดของข้อมูล ทำให้มีส่วนช่วยในการเพิ่มประสิทธิภาพในการบีบอัดข้อมูลของกระบวนการ EXI โดยลดพื้นที่ในการเก็บชื่อแท็ก และช่วยแปลงข้อมูลบางส่วนให้อยู่ในรูปของไบนารี่ เช่น ข้อมูลที่เป็นชนิด Integer, Float, และ วันเวลา เป็นต้น แทนที่จะเก็บเป็นรหัส ASCII ซึ่งใช้ที่เก็บมากกว่า ก่อนใช้กระบวนการนี้ต้องแน่ใจว่าทั้งผู้ส่งและผู้รับข้อมูลจะมีไฟล์ XSD อยู่แล้ว และใช้ให้ตรงกัน

2.3.1.2. **การลดข้อมูลซ้ำด้วยตารางข้อมูล String**

XML

```
<?xml version="1.0"?>
<favorites>
 <weapon>
  bow
 </weapon>
 <decoration>
  bow
 </decoration>
 <gesture>
  bow
 </gesture>
 <stateroom>
  bow
 </stateroom>
 <leg>
  bow
 </leg>
</favorites>
```

String Table

String ID	String
S1	bow

EXI

```
SD
SE(favorites)
SE(weapon)
CH(bow)
EE
SE(decoration)
S1
EE
SE(gesture)
S1
EE
SE(stateroom)
S1
EE
SE(leg)
S1
EE
EE
ED
```

รูปที่ 2-12 ตารางเก็บข้อมูล String ในการแปลงข้อมูล EXI

บ่อยครั้งที่เนื้อหาของข้อมูลจะปรากฏอยู่หลายที่ในข้อมูล XML ไฟล์หนึ่งๆ สามารถถูกจัดเก็บได้อย่างประหยัด ซึ่ง EXI ใช้งานตาราง String เมื่อ EXI พบกับลักษณะของ String ในระหว่างกระบวนการ Default Processing ข้อมูล String เหล่านี้จะถูกเพิ่มไปยัง Output stream

และจะถูกเพิ่มไปยัง String table อีกด้วย เพื่อในครั้งถัดไปที่ค่า String ปรากฏบนข้อมูล EXI จะทำเครื่องหมายจากจุดที่ต้องการอ้างอิงไปยังค่าที่ถูกเก็บไว้แล้วในตาราง String แทนที่จะจัดเก็บ String นั้นเป็นครั้งที่สอง ดังนั้นเมื่อมีข้อมูลจำนวนมากที่ถูกอ้างอิงภายในข้อมูล XML ก็หมายถึงการประหยัดพื้นที่เพิ่มขึ้น

ข้อมูล XML ดังแสดงในรูปที่ 2-12 จะประกอบไปด้วยรายการของสิ่งที่ได้รับความนิยมรายการแรกเป็นอาวุธที่ได้รับความนิยม คือ bow ซึ่งจะถูกเพิ่มเข้าไปใน Output stream และกลายเป็นข้อมูลรายการแรกในตาราง String จากนั้นเครื่องประดับที่ได้รับความนิยม คือ bow ข้อมูลนี้จะถูกอ้างอิงไปยังข้อมูลในตาราง String ลำดับที่ 1 และเพิ่มการอ้างอิงนี้เข้าไปใน Output stream ดังนั้นทุกๆ Element ที่ใช้คุณลักษณะของ b-o-w สามารถใช้การอ้างอิงแทนการเก็บข้อมูลในตัวแปร String

ในฝั่งที่ทำการถอดรหัสข้อมูลจะทำการสร้างตาราง String ขึ้นมาใหม่อีกครั้งหนึ่งในลำดับเดียวกันกับตัวแปร String ที่ประกฏขึ้น โดย String ที่ปรากฏขึ้นเป็นครั้งแรกจะถูกเพิ่มไปยังตารางซึ่งจะทำให้ข้อมูลต่าง จะถูกเพิ่มเข้าในตาราง String ก่อนที่ถูกอ้างอิงโดยเครื่องหมาย จากขั้นตอนดังกล่าวทำให้สามารถสร้างข้อมูลต้นฉบับได้อย่างสมบูรณ์

ถ้าหากกำลังทำงานกับระบบที่มีค่า bandwidth หรือ memory ที่จำกัด สามารถที่จะจำกัดจำนวนของ string ที่ถูกเพิ่มไปยังตารางได้ (String ถูกแทนที่บน first-in, first-out basis) อีกทั้งยังสามารถที่จะจำกัดความยาวของข้อมูล String ที่อยู่ในตารางได้อีกด้วย เนื่องจากข้อมูล String ที่ยาวกว่านั้นมีความน่าจะเป็นที่จะซ้ำกันค่อนข้างน้อย และไม่มีประโยชน์ต่อการนำกลับมาใช้ใหม่

2.3.2. การจัดเรียงข้อมูล (Data Alignment)

หลังจาก Default Processing จะต้องมีการจัดเรียงข้อมูล ซึ่งรวมถึงกระบวนการบีบอัดด้วย การจัดเรียงข้อมูลเลือกได้ 4 ระดับ ระดับแรกจะได้ *กระแสแบบตรงไบต์* (Byte-aligned Stream), ระดับสองจะได้ *กระแสก่อนบีบอัด* (Precompressed Stream), ระดับสามจะได้ *กระแสบีบอัด* (Compressed Stream), และระดับสูงสุดจะได้ *กระแสอัดบิต* (Bit-packed Stream) ระดับที่สูงขึ้นจะมีประสิทธิภาพในการบีบอัดข้อมูลที่สูงขึ้น ซึ่งส่งผลต่อการบีบอัดข้อมูล XML ให้มีประสิทธิภาพมากยิ่งขึ้น

2.3.2.1. **กระแสตรงไบต์**

กระแสตรงไบต์เป็นรูปแบบการแปลงข้อมูลที่มีผลออกมาแล้วมนุษย์สามารถอ่านได้มากที่สุด การเรียงข้อมูลในรูปแบบนี้มีประโยชน์สำหรับการตรวจสอบการทำงานของ EXI ข้อมูลจะถูกเข้ารหัสโดยยังมีขอบเขตข้อมูลเป็นไบต์ตามปกติ และสามารถที่จะมองเห็นข้อความบางอย่างได้ใน

เนื้อหาได้ คุณลักษณะของสัญลักษณ์จะปรากฏเมื่อได้รับค่าต่างๆ จากตาราง String ซึ่งจะเห็นได้ว่าการใช้ตาราง String นั้นทำให้ EXI ได้ลดขนาดของไฟล์ลงได้เป็นอย่างมาก

Notebook.xml

```
<?xml version="1.0"?>
<notebook>
  <note category="business">
    <subject>
      Strategy Meeting
    </subject>
    <details>
      Invite Tom.
    </details>
  </note>
  <note category="business">
    <subject>
      Strategy Meeting
    </subject>
    <details>
      Invite Joan.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Invite Tom.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Buy steaks.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Invite Joan.
    </details>
  </note>
</notebook>
```

Notebook.exi

```
€      notebook note
       category
Business subject
       Strategy Meeting
     ??details
       Invite Tom.
     ??????????
       Invite Joan.
     ??
personal??
       Saturday BBQ
     ????????????????
       Buy steaks.
     ??????????
```

รูปที่ 2-13 การจัดเรียงข้อมูลแบบกระแสตรงไบต์ (Byte-aligned Streams)

ข้อมูลกระแสตรงไบต์อาจไม่เหมาะกับการส่งไฟล์ เพราะยังไม่ได้รับการบีบอัดให้มีขนาดเล็กที่สุด ซึ่งสามารถสังเกตจากรูปที่ 2-13

Notebook.xml

```
<?xml version="1.0"?>
<notebook>
  <note category="business">
    <subject>
      Strategy Meeting
    </subject>
    <details>
      Invite Tom.
    </details>
  </note>
  <note category="business">
    <subject>
      Strategy Meeting
    </subject>
    <details>
      Invite Joan.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Invite Tom.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Buy steaks.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Invite Joan.
    </details>
  </note>
</notebook>
```

Notebook.exi

```
€???
business??
       Strategy Meeting
     ???
       Invite Tom.
     ??????????
       Invite Joan.
     ????
personal??
       Saturday BBQ
     ??????????????????
       Buy steaks.
     ????????????????
```

รูปที่ 2-14 การจัดเรียงข้อมูลแบบ Byte-aligned Streams ร่วมกับไฟล์โครงสร้าง XSD

ถ้าการบีบอัดข้อมูล EXI ทำงานร่วมกับข้อมูลโครงสร้าง XML หรือ XSD การบีบอัดจะมีประสิทธิภาพมากขึ้น โดย XSDจะเป็นตัวบอกถึงโครงสร้างข้อมูล และ ชนิดของข้อมูลใน XML ส่งผลให้สามารถละเลยชื่อของ Element และ Attribute ลงได้ดังแสดงในรูปที่ 2-14

เมื่อข้อมูล XML ถูกบีบอัดโดย EXI ที่ใช้งานร่วมกับ XSD หากเครื่องปลายทางมีการแคชไฟล์ XSD อยู่ ทำให้มีแต่ข้อมูล EXI ที่ต้องส่งไปเท่านั้น การใช้ XSD เหมาะกับ ข้อมูลที่มีโครงสร้าง XML ซ้ำๆ ไม่กี่รูปแบบ มีเพียงตัวข้อมูลที่เปลี่ยนไปตาม เวลา ชนิดของเซนเซอร์ หรือที่ตั้ง อย่างเช่นการใช้งานของ IEEE 1888

2.3.2.2. **กระแสก่อนบีบอัด**

การจัดเรียงข้อมูลเพื่อให้ได้กระแสก่อนบีบอัดจะเพิ่มประสิทธิภาพการบีบอัดข้อมูล ข้อมูล XML ที่ต้องรักษาลำดับชั้นโครงสร้างเพื่อทำให้ง่ายต่อการเข้าใจของมนุษย์ แต่คอมพิวเตอร์ไม่ได้สนใจมีความเกี่ยวข้องกับความเสมือนจริงของเนื้อหา มันสามารถจัดเก็บข้อมูลใหม่ด้วยวิธีที่มีประสิทธิภาพมากที่สุดสำหรับจัดเก็บ และเปลี่ยนรูปแบบของข้อมูล โดยการนำตัวแปรที่เหมือนกันมาใส่ใน Channel ที่ข้อมูลต่างๆ จะถูกรวมกันอย่างหนาแน่น โดยเฉพาะอย่างยิ่งข้อมูล วันที่, ตัวเลข, และค่าตัวแปรบูลลีน

Notebook.xml

```
<?xml version="1.0"?>
<notebook>
  <note category="business">
    <subject>
      Strategy Meeting
    </subject>
    <details>
      Invite Tom.
    </details>
  </note>
  <note category="business">
    <subject>
      Strategy Meeting
    </subject>
    <details>
      Invite Joan.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Invite Tom.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Buy steaks.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Invite Joan.
    </details>
  </note>
</notebook>
```

Notebook.exi

```
€          notebook note
           category subject
??details??????????????????????????
??????
business?
personal??
   Strategy Meeting
  ?
   Saturday BBQ
  ??
   Invite Tom.

   Invite Joan.
  ??
   Buy steaks.
  ?
```

รูปที่ 2-15 การจัดเรียงข้อมูลแบบกระแสก่อนบีบอัด (Precompressed Stream)

ข้อมูลที่ผ่านการจัดเรียงข้อมูลแบบกระแสก่อนบีบอัดดังแสดงในรูปที่ 2-15 จะเห็นข้อมูลเป็นกลุ่มๆ กลุ่มของข้อมูล element เป็นสิ่งที่ปรากฏขึ้นเป็นสิ่งแรก ถัดมาจะเป็นค่าของ attribute ค่าของวิชา (subject) และค่าของรายละเอียด (details) เมื่อไฟล์ถูกทำการคลายการบีบ

อัดตัวอ่านข้อมูล EXI จะอ่านค่าจากบนลงล่างในแต่ละกลุ่มข้อมูล เพื่อสร้างข้อมูลต้นฉบับขึ้นมาใหม่อีกครั้ง

2.3.2.3. **Compressed Streams**

รูปที่ 2-16 การจัดเรียงข้อมูลแบบ Compressed Streams

เมื่อได้กระแสก่อนบีบอัด ผู้ส่งมีทางเลือกที่จะบีบอัดโดยใช้กรรมวิธี Deflate ข้อมูลที่ผ่านการบีบอัดจะมีลักษณะดังรูปที่ 2-16 เมื่อเปิดในโปรแกรมแก้ไขข้อความ (Editor) จะเห็นว่ารูปแบบ และโครงสร้างข้อมูลที่ปรากฏในข้อมูล XML ถูกละทิ้งไป ทำให้คอมพิวเตอร์สามารถเก็บและแปลงค่าข้อมูล EXI ได้อย่างมีประสิทธิภาพ เป็นผลให้สามารถลดโสหุ้ยของข้อมูลของ XML ได้เกือบทั้งหมด

การใช้ Default Processing ร่วมกับการบีบอัด ทำให้ข้อมูลที่ได้มีขนาดเล็กลงมาก อาจมากกว่า 7 เท่าเมื่อเทียบกับการใช้ GZip ในการบีบอัดข้อมูล XML

การบีบอัดข้อมูลให้มีขนาดเล็กที่สุดนั้นไม่สามารถทำได้ในทุกกรณี เช่นในกรณีที่ผู้รับมี Processing Power ต่ำ แต่ข้อมูลมีปริมาณน้อย แต่กรณีที่ไฟล์ที่จะส่งมีขนาดใหญ่มาก หรือกรณีที่ทำงานกับระบบที่มีพื้นที่ในการจัดเก็บข้อมูล หรือช่องทางการสื่อสารจำกัด จะได้รับประโยชน์อย่างมากจากการใช้การบีบอัดข้อมูลโดยใช้ EXI

2.3.2.4. กระแสอัดบิต (Bit-packed Stream)

Notebook.xml

```
<?xml version="1.0"?>
<notebook>
  <note category="business">
    <subject>
      Strategy Meeting
    </subject>
    <details>
      Invite Tom.
    </details>
  </note>
  <note category="business">
    <subject>
      Strategy Meeting
    </subject>
    <details>
      Invite Joan.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Invite Tom.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Buy steaks.
    </details>
  </note>
  <note category="personal">
    <subject>
      Saturday BBQ
    </subject>
    <details>
      Invite Joan.
    </details>
  </note>
</notebook>
```

Notebook.exi

```
€B[›ÝX›ÛÚä¹½ÑBXØ]YÛÜžB˜ÚÚ[™\Ü
òæêÄÔÊÆéÞ90º2³¼²²º4·3...$!'Ñ...
¥±Ï
    Invite Tom.
  2?P??AD          -ÎÍ.Œ¤     Mì-
ÅÁDBœ\œÛÛ˜[?ÐQ›
£«“#
É^Q??€@B^HÝXZÜË,^€@?€
```

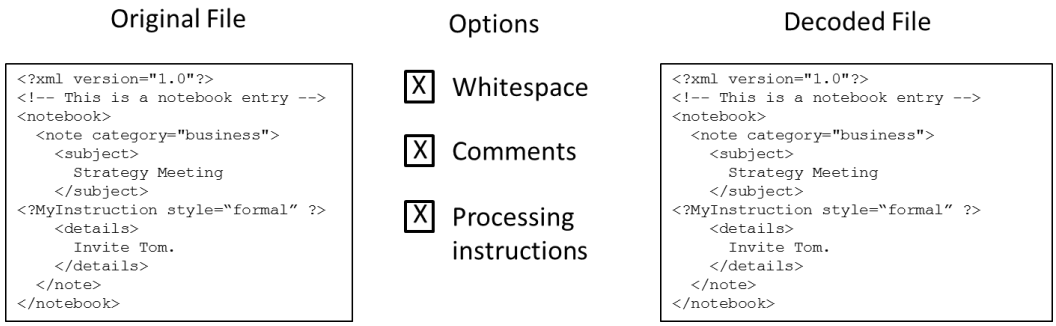
รูปที่ 2-17 การจัดเรียงข้อมูลแบบกระแสอัดบิต (Bit-packed Stream)

ก่อนการบีบอัดในหัวข้อก่อนหน้า ข้อมูลยังเป็นแบบตรงไบต์อยู่ ผู้ส่งอาจเลือกจัดเรียงข้อมูลเสียใหม่โดยการจัดเรียงแบบอัดบิตดังแสดงในรูปที่ 2-17 เป็นค่าเริ่มต้น (default value) ข้อมูลจะถูกรวมกลุ่มไปด้วยกันในกระแสขาออก (Output Stream) โดยไม่ต้องคำนึงถึงขอบเขตข้อมูลในแต่ละไบต์ ในบางครั้งการเรียงข้อมูลแบบนี้จะได้ข้อมูลที่เป็นข้อความที่อ่านได้ ถ้าขอบเขตระหว่างไบต์เมื่ออัดบิตแล้ว บังเอิญตรงกับข้อมูลก่อนการอัดบิต (เช่น Invite Tom) แต่ข้อความส่วนใหญ่ที่ได้นั้นมนุษย์จะไม่สามารถอ่านได้

โดยทั่วไปกระแสที่ผ่านการอัดบิตก่อนการใช้กระบวนการบีบอัด Deflate จะได้รับการบีบอัดที่มากขึ้น แลกกับการประมวลผลที่มากขึ้น

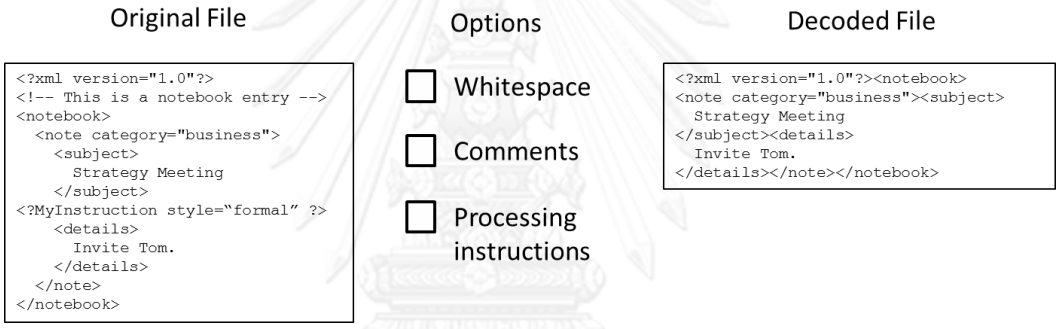
2.3.3. ตัวเลือกในการจัดเก็บข้อมูล

ช่องว่าง ข้อความอธิบายข้อมูล และเนื้อหาที่ไม่สำคัญในข้อมูล XML เป็นสิ่งที่มีประโยชน์สำหรับผู้พัฒนา แต่ในการใช้งานจริงไม่ได้ใช้งานข้อมูลเหล่านี้ ในการบีบอัดข้อมูล ก็ไม่มีเหตุผลที่จะเก็บสงวนข้อมูลตรงส่วนนี้ไว้ สามารถที่จะทิ้งข้อมูลเหล่านี้ได้ ซึ่งจะช่วยลดขนาดข้อมูลได้เป็นอย่างมาก



รูปที่ 2-18 ข้อมูลที่รักษา ช่องว่าง ข้อความอธิบายข้อมูล และ การจัดเรียงข้อมูล

ถ้าหากจำเป็นที่จะต้องรักษาข้อมูลตรงส่วนนี้ไว้ EXI Transmogrifier จะมีตัวเลือกที่ใช้ในการรักษาข้อมูลในส่วนนี้ไว้ได้ดังแสดงในรูปที่ 2-18



รูปที่ 2-19 ข้อมูลที่ละเลย ช่องว่าง ข้อความอธิบายข้อมูล และ การจัดเรียงข้อมูล

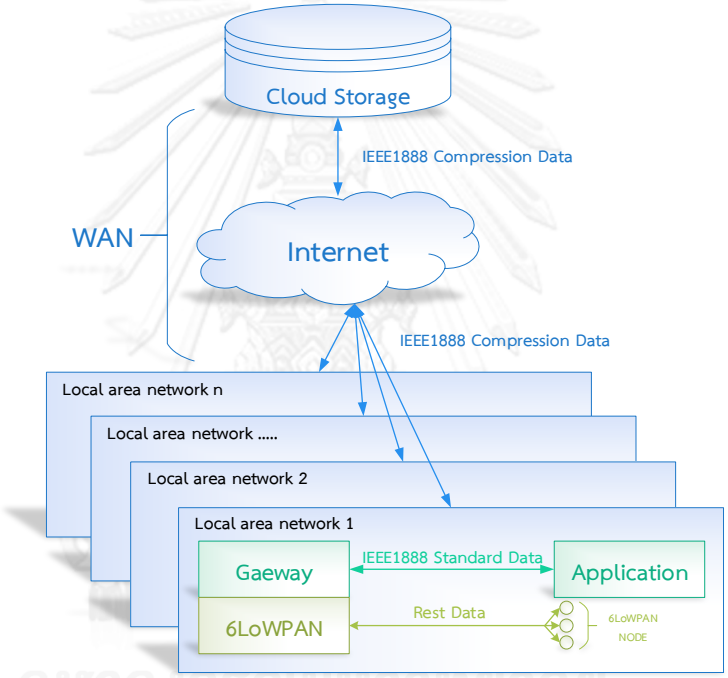
การเก็บข้อมูลเฉพาะส่วนที่ต้องการใช้งานก่อนการบีบอัดข้อมูล จะทำให้สามารถเพิ่มประสิทธิภาพด้านความเร็ว และการลดขนาดของไฟล์ได้ดังแสดงในรูปที่ 2-19

บทที่ 3

การออกแบบสถาปัตยกรรมระบบเครือข่าย และฮาร์ดแวร์

ในบทนี้ได้กล่าวถึงการออกแบบสถาปัตยกรรมระบบเครือข่าย และฮาร์ดแวร์ ที่ทำการออกแบบให้ระบบเครือข่ายแต่ละประเภทมีชนิดของข้อมูลที่ใช้ในการสื่อสารแตกต่างกันออกไป การออกแบบฮาร์ดแวร์ของ สตอเรจ เกทเวย์ โหนดวัดสภาวะแวดล้อม โหนดวัดพลังงาน และแอพพลิเคชั่น

3.1. การออกแบบสถาปัตยกรรมระบบเครือข่าย



รูปที่ 3-1 ระบบเครือข่ายที่ได้ทำการออกแบบ

ระบบเครือข่ายที่นำเสนอในงานวิจัยนี้ถูกออกแบบมาเพื่อให้ระบบจัดการพลังงานภายในอาคารหลายๆ อาคาร แต่ใช้งานสตอเรจในการเก็บข้อมูลร่วมกันดังแสดงในรูปที่ 3-1 ภายในระบบจัดการพลังงานภายในอาคารหนึ่งอาคารจะประกอบไปด้วยคอมโพเน้นท์ 2 ชนิด คือ เกทเวย์ และ โปรแกรมประยุกต์ ที่สามารถทำงานได้โดยไม่ต้องพึ่งพาสตอเรจตลอดเวลา โปรแกรมประยุกต์สามารถอ่านข้อมูลล่าสุด และสั่งการควบคุมการทำงานของ Actuator ไปยังเกทเวย์ได้โดยตรง ส่วนสตอเรจนั้นจะทำการเก็บข้อมูลบันทึกย้อนหลัง หรือประมวลผลข้อมูลเป็นรายเดือน เป็นต้น [10] ดังนั้นเพื่อให้ใช้งานระบบเครือข่ายอย่างมีประสิทธิภาพจึงเลือกใช้งานระบบเครือข่าย และรูปแบบของข้อมูลตามความเหมาะสม 3 ประเภท ดังต่อไปนี้

1. Wide Area Network (WAN) ระบบเครือข่ายประเภทนี้รองรับพื้นที่การสื่อสารกว้างระดับชุมชน ระดับเมือง ระดับประเทศ หรือแม้แต่ครอบคลุมได้ทั่วโลก มีปริมาณข้อมูลหนาแน่นแต่มีความเร็วจำกัด และมีค่าใช้จ่าย เช่นระบบ ADSL ระบบ 3G เป็นต้น ดังนั้นการสื่อสารข้อมูลในระบบเครือข่ายนี้ควรที่จะทำให้ข้อมูลมีขนาดเล็กที่สุด จึงได้ปรับปรุงการใช้งานข้อมูลในมาตรฐาน IEEE1888 โดยเพิ่มส่วนของการบีบอัด เพื่อที่จะทำให้เกิดความรวดเร็วในการสื่อสารข้อมูล อีกทั้งยังสามารถช่วยลดค่าใช้จ่ายในระบบเครือข่ายที่คิดค่าใช้งานตามปริมาณข้อมูล เช่น 3G เป็นต้น

2. Local Area Network (LAN) ระบบเครือข่ายประเภทนี้จะครอบคลุมพื้นที่จำกัด แต่มีความเร็วในการสื่อสารข้อมูลที่รวดเร็ว และมักไม่เสียค่าใช้จ่าย ดังนั้นคอมโพเน้นท์ในระบบเครือข่ายนี้สามารถที่จะใช้งานข้อมูลในมาตรฐาน IEEE1888 ตามปกติในการสื่อสารข้อมูลได้

3. IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) ระบบเครือข่ายนี้จะใช้ในการสื่อสารข้อมูลกับ เซ็นเซอร์ และ Actuator ที่รับค่าของสภาวะแวดล้อมของอาคาร การใช้พลังงาน และ ควบคุมการทำงานของเครื่องใช้ไฟฟ้าต่างๆ ซึ่งระบบเครือข่ายประเภทนี้ควรมีราคาที่ถูก มีการใช้พลังงานไฟฟ้าที่ต่ำ ง่ายต่อการติดตั้ง โดยในมาตรฐานของ 6LoWPAN นั้นความยาวของข้อมูลที่ใช้ในการสื่อสารนั้นค่อนข้างจำกัดเพียง 128 ไบต์ ดังนั้นจึงได้ออกแบบข้อมูลที่มีลักษณะเฉพาะที่มีลักษณะคล้ายคลึงกับมาตรฐาน IEEE1888 อยู่ในรูปแบบ XML มีขนาดเล็ก และง่ายต่อการเขียนโปรแกรม

3.2. การออกแบบฮาร์ดแวร์สตอเรจ

ฮาร์ดแวร์ของสตอเรจนั้นอาจจะใช้งานเครื่องคอมพิวเตอร์ตั้งโต๊ะแบบอุสาหกรรมหากระบบมีขนาดเล็ก (มีการ Write หรือ Fetch ไม่มาก) แต่ถ้าหากระบบขนาดใหญ่จะต้องใช้เครื่องแม่ข่ายที่มีความเร็วการประมวลผล และความจุฮาร์ดดิกส์ (*Hard disk*) ที่ใช้ในการเก็บข้อมูลใหญ่ขึ้นกว่าเดิม โดยปกติแล้วสตอเรจจะต้องมีคุณสมบัติขั้นต่ำดังต่อไปนี้

- **ความเร็วของ CPU อย่างน้อย 1 GHz Pentium 4**
- **ขนาดของแรม *RAM* 512 เมกะไบต์**
- **ฮาร์ดดิกส์ขนาด 10 กิ๊กกะไบต์**
- **สามารถเชื่อมต่ออีเธอร์เน็ต *Ethernet* ที่มีความเร็ว 10/100 Mbps**

ในการวิจัยนี้ได้เลือกใช้ใช้งานเครื่องคอมพิวเตอร์ตั้งโต๊ะสำหรับใช้ในที่ทำงาน เนื่องจากมีราคาถูก และไม่ได้เปิดต่อเนื่องเป็นเวลาแรมเดือน

3.3. การออกแบบฮาร์ดแวร์เกทเวย์

ฮาร์ดแวร์ของเกทเวย์ได้นำเอา BeagleBone Black ดังแสดงในรูปที่ 3-2 ซึ่งเป็นคอมพิวเตอร์สำหรับระบบสมองกลฝังตัวที่มีขนาดเล็กมาใช้ในการสร้างเกทเวยที่ทำการเชื่อมต่อมาตรฐาน IEEE1888 เข้ากับ 6LoWPAN ซึ่งจำเป็นจะต้องมีการแปลงส่วนหัว (Header) ข้อมูลในรูปแบบ 6LoWPAN ให้เป็นส่วนหัวของ IPV6 อีกทั้งต้องแปลงข้อมูลที่ได้รับมาให้อยู่ในรูปแบบ UDP ให้อยู่ในมาตรฐาน IEEE1888 ด้วย ดังนั้นจึงจำเป็นอย่างมากที่จะต้องใช้หน่วยประมวลผลที่ข้อนข้างมีความเร็วสูงอย่าง BeagleBone Black ซึ่งมีคุณสมบัติตามหัวข้อ 3.3.1

3.3.1. คุณสมบัติของ BeagleBone Black [11]

รูปที่ 3-2 บอร์ด BeagleBone Black

3.3.1.1. คุณสมบัติด้านฮาร์ดแวร์

- **โปรเซสเซอร์ AM335x 1GHz ARM® Cortex-A8**
- **แรม DDR3 ขนาด 512 เมกะไบต์**
- **หน่วงความจำ eMMC ชนิดแฟลชบนตัวบอร์ตขนาด 2 กิ๊กกะไบต์**
- **หน่วยเร่งการประมวลทางด้านสามมิติ**
- **หน่วยเร่งการประมวลผลเลขทศนิยม NEON**
- **หน่วยควบคุมชนิดเรียลไทม์ที่สามารถโปรแกรมได้ Programmable Real-Time Unit (PRU) 2 หน่วย**

3.3.1.2. การเชื่อมต่อของ BeagleBone Black

- **USB Client สำหรับจ่ายไฟฟ้าให้บอร์ด และสำหรับสื่อสาร 1 พอร์ต**

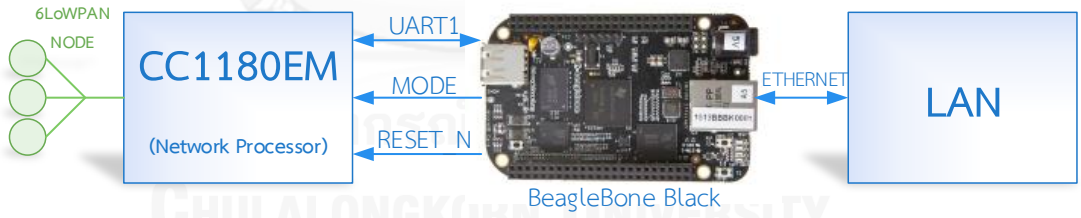
- **USB Host 1 พอร์ต**
- **อีเธอร์เน็ต 1 พอร์ต**
- **High-Definition Multimedia Interface (HDMI) 1 พอร์ต**
- **2 x 23 พินเฮดเดอร์ สำหรับเชื่อมต่อ UART, GPIO และขาอื่นๆ ของโปรเซสเซอร์ 2 ชุด**

3.3.1.3. **ซอฟต์แวร์ที่รองรับ**

- **Ångström Linux**
- **Android**
- **Ubuntu**
- **Cloud9 IDE on Node.js w/ BoneScript library**
- **ฯลฯ**

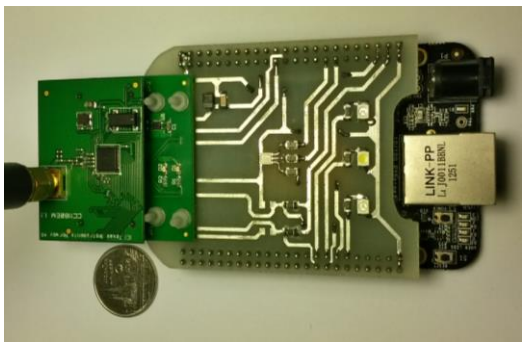
ในโครงการนี้ได้ใช้งานระบบปฏิบัติการ Ubuntu เพราะมีประสิทธิภาพดี และมีความคุ้นเคยกว่าระบบปฏิบัติการอื่น

3.3.2. การเชื่อมต่อ Beaglebone Black กับ 6LoWPAN และอินเตอร์เน็ต



รูปที่ 3-3 การเชื่อมต่อ Beaglebone Black กับ 6LoWPAN และอินเตอร์เน็ต

Beaglebone Black ร่วมกับ บอร์ดทดสอบ 1188EM ทำหน้าที่เป็นเกตเวย์ให้กับเครือข่าย 6LoWPAN ไปยังอินเตอร์เน็ต ดังแสดงในรูปที่ 3-3 จะใช้งาน Universal asynchronous receiver/transmitter (UART) พอร์ตที่ 1 ของ Beaglebone Black เชื่อมต่อเข้ากับ CC1180EM เพื่อทำการรับส่งข้อมูล พอร์ต *MODE* ใช้สำหรับเปลี่ยนโหมดการทำงานระหว่างโหมด Bootloader และ Application โดยจะมีการเปลี่ยนโหมดเมื่อ CC1180EM ได้รับสัญญาณขอบขาลง พอร์ต *GPIO RESET_N* ใช้สำหรับสั่งการรีเซ็ตการทำงานของ CC1180EM เมื่อได้รับสัญญาณที่มี Logic Low



รูปที่ 3-4 เกทเวจที่ได้จากการออกแบบ

จากการออกแบบเกทเวย์ข้างต้นได้นำไปสร้างวงจรแล้วสังเคราะห์ออกมาเป็นแผ่นวงจรพิมพ์ดังแสดงในรูปที่ 3-4 ซึ่งเป็นบอร์ดที่แปลงพินเฮดเดอร์ของ BeagleBone Black ให้ตรงกับพินเฮดเดอร์ของ CC1180EM เกทเวย์ที่ได้ออกแบบขึ้นมานี้มีขนาดใหญ่กว่าบัตรเครดิตเพียงเล็กน้อย

3.4. การออกแบบฮาร์ดแวร์ 6LoWPAN โหนด

รูปที่ 3-5 ชุดพัฒนา CC-6LoWPAN

ฮาร์ดแวร์ของ 6LoWPAN โหนดได้นำชุดพัฒนา CC-6LoWPAN ของบริษัท Texas Instrument (TI) [12] ดังแสดงในรูปที่ 3-5 มาใช้งาน ซึ่งชุดพัฒนานี้ใช้งานมาตรฐาน 6LoWPAN ในการสื่อสาร ซึ่งมีคุณสมบัติตามหัวข้อ 3.4.1 โดยในงานวิจัยได้นำโหนดในชุดพัฒนานี้มาสร้างโหนดสำหรับบริหารจัดการพลังงานไฟฟ้าภายในอาคารจำนวน 2 โหนด คือ 1. โหนดวัดพลังงานไฟฟ้า 2. โหนดวัดสภาวะแวดล้อม ดังมีรายละเอียดดังต่อไปนี้

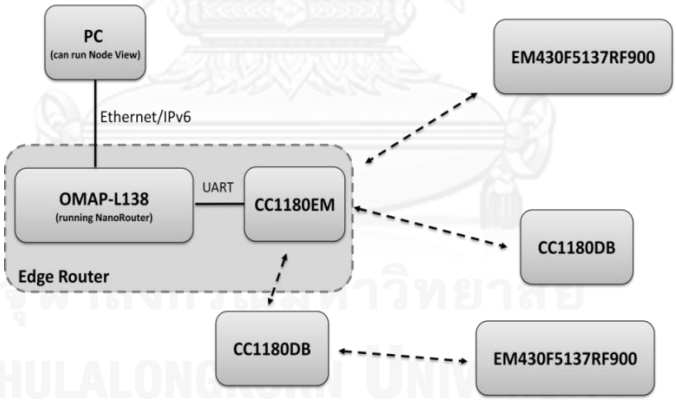
3.4.1. ชุดพัฒนา CC-6LoWPAN

3.4.1.1. คุณสมบัติของชุดพัฒนา CC-6LoWPAN

- **ระบบเครือข่ายที่มีลักษณะการทำงานแบบ IP ซึ่งทำให้ทุกอย่างสื่อสารอยู่บนอินเทอร์เน็ต “Internet of Things”**
- **CC1180 สามารถทำการอัพเกรดเฟิร์มแวร์ไร้สาย Over-Network Download (OTA)**
  - รองรับการอัพเกรดโปรแกรมประยุกต์ และเฟิร์มแวร์ระบบเครือข่าย
- **ซอฟร์ตแวร์สำหรับโพรโตคอลสแตกใช้งานหน่วยความจำขนาดเล็ก**
  - 6LoWPAN สแตกมี สำหรับบอร์ดที่ใช้ CC1180 มีขนาดเล็กกว่า 32 กิโลไบต์
  - 6LoWPAN สแตกมี สำหรับบอร์ดที่ใช้ CC430 มีขนาดประมาณ 17 กิโลไบต์
- **ซอฟต์แวร์ Sensinode 6LoWPAN สามารถที่จะทำงานได้ในทุกความถี่ที่ CC1180 และ CC430 รองรับ ซึ่งทำงานในช่วงความถี่ที่ต่ำกว่า 1 กิกะเฮิร์ทซ์ (หมายเหตุ: ชุดพัฒนา CC-6LoWPAN ได้ถูกออกแบบมาใช้งานที่ความถี่ 868/915 เมกกะเฮิร์ทซ์)**
- **ผู้ใช้งานสามารถโปรแกรมการสื่อสารแบบ IP มีความซับซ้อนในการพัฒนาต่ำ และยังลดเวลาในการพัฒนาด้วย Application Programming Interface (API) ที่ง่ายต่อการใช้งาน**
- **อินเตอร์เฟสของคลื่นวิทยุที่สามารถตั้งค่าได้**
  - กำลังส่ง -30dBm ถึง +10dBm
  - ความเร็วในการส่งข้อมูล 50, 100, 150 หรือ 200kbit/s
  - การลดทอนสัญญาณทางด้านรับ
  - รองรับการเข้ารหัสข้อมูลแบบ AES-CCM* secured IEEE802.15.4e
- **มีระบบเครือข่ายที่ประสานงานกันแบบ Mesh**
- **ใช้งานมาตรฐาน IEEE802.15.4g/e ใน PHY และ MAC เลเยอร์**
- **รองรับการบีบอัดเฮดเดอร์ของ IPv6**

- **ใช้งานโปรโทคอล ICMPv6 เพื่อค้นหาโหนดใกล้เคียง**
- **ใช้ User Datagram Protocol (UDP) ในการรับส่งข้อมูล ในชั้นเครือข่าย (Network Layer)**
- **รองรับการใช้งานแอดเดรสขนาดสั้นในการสื่อสารในชั้น link-layer ซึ่งแอดเดรสนี้จะถูกแจกให้กับโหนดโดย Edge Router มีขนาด 2 ไบต์ และไม่ซ้ำกันในระบบเครือข่าย 6LoWPAN ขนาดเล็ก**
- **มีกระบวนการ Bootstrap และค้นหาเส้นทางในการส่งข้อมูลอัตโนมัติ**
- **สามารถค้นหาเส้นทางของระบบเครือข่ายแบบ Mesh ได้เมื่อเส้นทางเดิมใช้งานไม่ได้**
- **ทุกโหนดสามารถรับโปรโทคอล ICMPv6 แล้วสามารถตอบกับได้**
- **สามารถสื่อสารแบบ P2P**
- **รองรับ Synchronous frequency hopping โดยใช้งานช่องสัญญาณ 50 FHSS**

3.4.1.2. **ระบบโดยรวมของชุดพัฒนา CC-6LoWPAN**

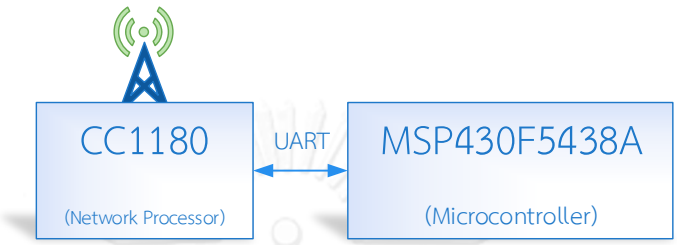


รูปที่ 3-6 ระบบของชุดพัฒนา CC-6LoWPAN

ภายในชุดพัฒนา CC-6LoWPAN ดังแสดงในรูปที่ 3-6 จะประกอบไปด้วยบอร์ดพัฒนาจำนวน 3 บอร์ด คือ บอร์ด Edge Router, บอร์ด CC1180DB, และ บอร์ด EM430F513RF900 ซึ่งแต่ละบอร์ดมีคุณสมบัติที่แตกต่างกันออกไปดังต่อไปนี้

1. บอร์ด Edge Router จะประกอบไปด้วย 2 บอร์ดย่อยด้วยกัน คือ 1. บอร์ด OMAP-L138 มีการทำงานเสมือนเครื่องคอมพิวเตอร์สำหรับระบบสมองกลฝังตัวที่ใช้งาน

ระบบปฏิบัติการ Linux รันซอฟต์แวร์ที่ชื่อว่า NanoRouter อยู่ ซึ่งซอฟต์แวร์ตัวนี้มีหน้าที่ ในการจัดการแปลงรูปแบบเฮดเดอร์ของ 6LoWPAN เป็น IPv6 อีกทั้งติดต่อโมดุล RF อินเทอร์เฟส CC1180EM ผ่านทาง UART 2. บอร์ด CC11800EM เป็นโมดุล RF ที่ทำงานในช่วงความถี่ที่ต่ำกว่า 1 กิกะเฮิร์ทซ์ ที่คอยรับส่งสัญญาณให้กับ บอร์ด CC1180EM และ EM430F5137RF900 เมื่อลงทะเบียนเข้าสู่ระบบครั้งแรก หรือเมื่อต้องการติดต่อกับระบบเครือข่ายภายนอก เป็นต้น

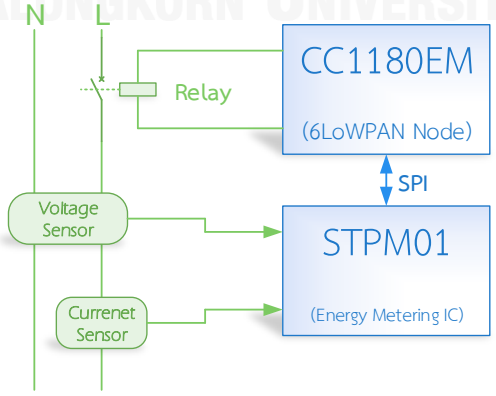


รูปที่ 3-7 ฮาร์ดแวร์ภายในบอร์ด CC1180DB

2. บอร์ด CC1180DB มีฮาร์ดแวร์ดังแสดงในรูปที่ 3-7 ทำหน้าเป็นโหนดซึ่งภายในประกอบไปด้วยไมโครคอนโทรเลอร์ MSP430F5438A ที่ติดต่อ Network Porcessor CC1180 โดยผ่านทาง UART ซึ่ง CC1180 จะทำหน้าที่เป็นตัวประมวลผลทางด้านระบบเครือข่าย 6LoWPAN ทั้งหมดส่วนไมโครคอนโทรเลอร์นั้นมีหน้าที่ในการอ่านค่าเซ็นเซอรN ควบคุม Actuator และ รองรับ Event ต่างๆที่ CC1180 ส่งมาให้

3. บอร์ด EM430F5137RF900 ทำหน้าที่เป็นโหนด ซึ่งใช้งาน System on Chip (SoC) พร้อม RF อินเทอร์เฟสเบอร์ CC430F52x1 เพียงไอซีเดียว ซึ่งทำหน้าที่ประมวลผลโปรโทคอล 6LoWPAN และทำงานอย่างอื่นพร้อมๆกันไปด้วย ทำให้บอร์ด EM430F5137RF900 นั้นใช้พลังงานและมีโปรเซสซิ่งเพาเวอร์ต่ำกว่าบอร์ด CC1180DB

3.4.2. การออกแบบโหนดวัดพลังงานไฟฟ้า



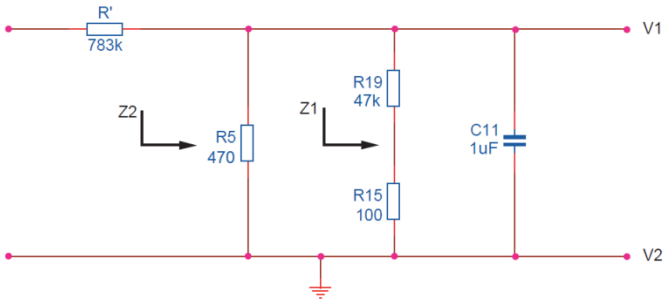
รูปที่ 3-8 บล็อกไดอะแกรมของโหนดวัดพลังงานไฟฟ้า

โหนดวัดพลังงานไฟฟ้าได้ถูกออกแบบมาเพื่อทำการวัดการใช้พลังงานไฟฟ้าของเครื่องใช้ไฟฟ้า ซึ่งใช้งานไอซี STPM01 [13] ในการวัดการใช้พลังงาน และมี Relay ที่ใช้ในการตัดต่อไฟฟ้าให้แก่เครื่องใช้ไฟฟ้า โดยไอซี STPM01 จะวัดกระแสไฟฟ้าผ่านทางหม้อแปลงกระแส ส่วนแรงดันนั้นจะถูกวัดโดยใช้ตัวต้านทางต่ออนุกรมเพื่อลดระกับแรงดันไฟฟ้า ซึ่งสื่อสารกับบอร์ด CC1180EM ผ่านทาง Serial Peripheral Interface (SPI) ดังมีรายละเอียดดังต่อไปนี้

3.4.2.1. คุณสมบัติของไอซีวัดพลังงานไฟฟ้า STPM01

- **วัดพลังงานไฟฟ้า Active, reactive, apparent และ ค่า RMS**
- **ให้สัญญาณพัลส์เอาท์พุตที่ขึ้นกับปริมาณการใช้พลังงาน**
- **สามารถตรวจสอบการลักลอบใช้พลังงานไฟฟ้าจาก Live และ neutral**
- **ง่าย และรวดเร็วในการสอบเทียบในระบบดิจิตอล ซึ่งใช้จุดสอบเทียบเพียงหนึ่งจุดเท่านั้น**
- **One Time Programming (OTP) สำหรับกำหนดค่าการทำงาน และเก็บค่าสอบเทียบ**
- **มีวงจรรักษาแรงดันแบบเชิงเส้นภายในที่จ่ายให้กับวงจรแอนะล็อก และดิจิตอล**
- **สามารถใช้งานร่วมกับวงจรกำเนิดความถี่แบบ RC หรือ Crystal**
- **รองรับการใช้งานที่ความถี่ 50 – 60 เฮิร์ทซ์ ตามมาตรฐาน IEC62052-11, IEC62053-2x**
- **มีความผิดพลาดต่ำกว่า 0.1%**
- **มีวงจรรักษาแรงดันที่เที่ยงตรง 1.23 โวลต์ ที่ 30 ppm/°C max**

3.4.2.2. การออกแบบวงจรวัดแรงดันไฟฟ้า



รูปที่ 3-9 วงจรวัดแรงดันไฟฟ้า

ตารางที่ 3-1 ย่านแรงดันอินพุตของไอซี STPM01 สำหรับวัดแรงดันไฟฟ้า

Voltage channels	
Gain	**Max Input voltage (V)**
4	±0.30

วงจรวัดแรงดันไฟฟ้าที่ได้ออกแบบจะใช้ตัวต้านทานต่อขนานกันเพื่อลดระดับแรงดันไฟฟ้าดังแสดงในรูปที่ 3-9 เพื่อให้มีระดับแรงดันอินพุตอยู่ในช่วงที่ไอซี STPM01 ได้ออกแบบไว้ตามตารางที่ 3-1 โดยความต้านทาน 783kΩ จะถูกแบ่งออกเป็นตัวต้านทาน 380kΩ, 380kΩ, และ 3kΩ ต่ออนุกรมกันเพื่อเป็นการป้องกันแรงดันไฟฟ้าสูงเกินกำหนดในช่วง transient C11, R19, และ R15 ทำหน้าที่เป็นวงจรฟิลเตอร์ที่ป้องกันคลื่นแม่เหล็กไฟฟ้ารบกวน Electromagnetic Interference (EMI) ดังนั้นจากวงจรที่ได้ออกแบบสามารถนำมาหาความสัมพันธ์ของแรงดันอินพุทและแรงดันเอาท์พุต ตามสมการต่อไปนี้

ความต้านทาน Z1

$$Z_1 = (R_{19} + R_{15}) = 47.1k\Omega \tag{3-1}$$

ความต้านทาน Z247

$$Z_2 = \frac{R_5 \cdot Z_1}{R_5 + Z_1} = 465.36\Omega \tag{3-2}$$

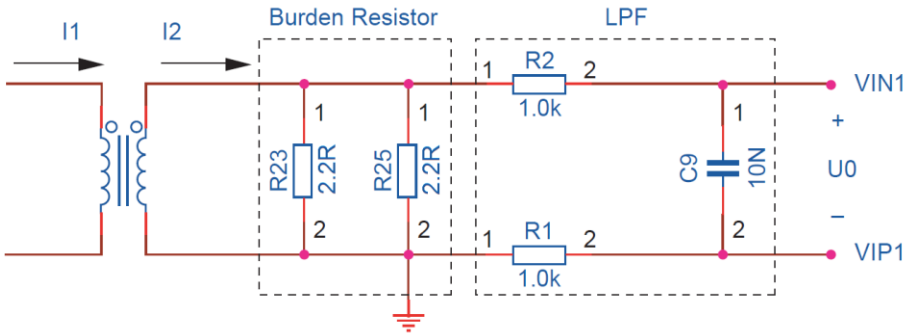
แรงดันเอาท์พุต U_0

$$U_0 = \frac{Z_2}{R' + Z_2} \cdot V_{mains} = \begin{cases} V_{mains} = 110\sqrt{2}\,V, U_1 = 0.092 \\ V_{mains} = 220\sqrt{2}V, U_1 = 0.185 \end{cases} \tag{3-3}$$

ความต้านทาน Z_1มีผลต่อแรงดันเอาท์พุต U_0น้อยมาก ดังนั้น

$$U_0 \approx \frac{R_5}{R' + R_5} \cdot V_{mains} \tag{3-4}$$

3.4.2.3. การออกแบบวงจรวัดกระแสไฟฟ้าไฟฟ้า



รูปที่ 3-10 วงจรวัดกระแสไฟฟ้า

ตารางที่ 3-2 ย่านแรงดันอินพุตของไอซี STPM01 สำหรับวัดกระแสไฟฟ้า

Currrent channels	
Gain	**Max Input voltage (V)**
8x	±0.15
16x	±0.075
24x	±0.05
32x	±0.035

วงจรวัดกระแสไฟฟ้าที่ได้ออกแบบจะใช้หม้อแปลงกระแสไฟฟ้า *Current transformer* (CT) เป็นตัวตรวจจับกระแสไฟฟ้าที่ไหลผ่านซึ่งมีจำนวนรอบ 1000/1 ซึ่งมีหมายความว่า CT มีกระแสจริงที่ขดปฐมภูมิ I_1 เท่ากับ 1000 แอมป์ จะมีกระแสที่ขดทุติยะภูมิ I_2 เท่ากับ 1 แอมป์ดังแสดงในรูปที่ 3-10 ตัวต้านทาน Burden ที่ต่อขนานกัน 2 ตัวเพื่อแปลงกระแสที่ได้รับมาจาก CT ให้เป็นแรงดันไฟฟ้า VIN1 และ VIN2 ที่มีระดับแรงดันอินพุตอยู่ในช่วงที่ไอซี STPM01 ได้ออกแบบไว้ตามตารางที่ 3-2 วงจร Low Pass Filter (LPF) มีหน้าที่ในการ Filter สัญญาณรบกวนในความถี่สูง โดยวงจร LPF ที่ออกแบบมานี้มีผลกระทบต่อแรงรันตกคล่อมที่ VIN1 และ VIN2 น้อยมาก ดังนั้นจากวงจรที่ได้ออกแบบสามารถนำมาหาความสัมพันธ์ของกระแสอินพุท และแรงดันเอาท์พุตตามสมการต่อไปนี้

ความสัมพันธ์ระหว่างกระแส I_1 และ I_2

$$I_2 = \frac{N_1}{N_2} \cdot I_1 \tag{3-5}$$

แรงดันเอาท์พุต

$$U_0 \approx U_A = I_2 \cdot \frac{R_{23} \cdot R_{25}}{R_{23} + R_{25}} = \frac{N_1}{N_2} \cdot I_1 \cdot \frac{R_{23} \cdot R_{25}}{R_{23} + R_{25}} \tag{3-6}$$

กำหนดให้ I_{1PEAK} เท่ากับ 30 แอมป์ จะได้ U_{0PEAK}เท่ากับ

$$\frac{I_{1PEAK}}{I_{2PEAK}} = \frac{N_2}{N_1} = \frac{1000}{1} \tag{3-7}$$

$$I_{2PEAK} = \frac{I_{1PEAK}}{1000} = 30mA \tag{3-8}$$

$$U_{0PEAK} = U_{APEAK} = I_{2PEAK} \frac{R_{23} \cdot R_{25}}{R_{23} + R_{23}} = 33mV \tag{3-9}$$

เนื่องจากแรงดันสูงสุดในอินพุต VIN1 และ VIN2 ขึ้นอยู่กับอัตราขยายที่สามารถตั้งค่าได้ผ่านการโปรแกรม ซึ่งในวงจรที่ออกแบบได้เลือกใช้อัตราขยาย 8 เท่าซึ่งจะมี $U_{0PEAK} = 0.15V$ ดังนั้นวงจรวัดกระแสที่ได้ออกแบบมานี้จะสามารถวัดแรงดัน I_{1PEAK} สูงสุดได้เท่ากับ

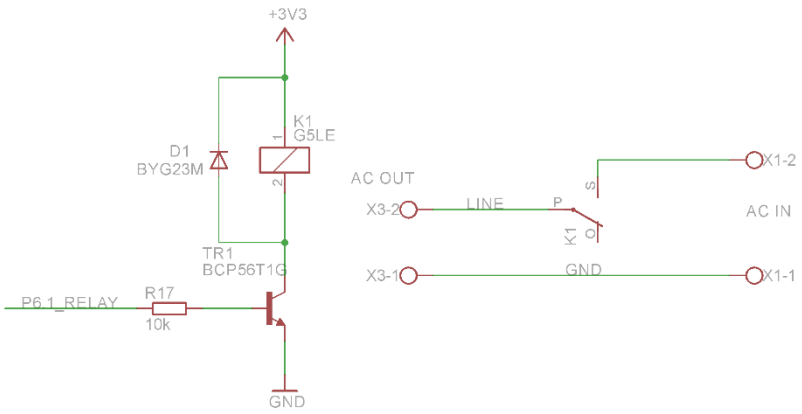
$$U_{APEAK} = U_{0PEAK} = 0.15V \tag{3-10}$$

$$I_{2PEAK} = U_{APEAK} \cdot \frac{R_{23} + R_{25}}{R_{23} \cdot R_{25}} = 165mA \tag{3-11}$$

$$I_{1PEAK} = 1000 \cdot I_{2PEAK} = 165A \tag{3-12}$$

$$I_{1RMS} = \frac{I_{1PEAK}}{\sqrt{2}} = 116.67A \tag{3-13}$$

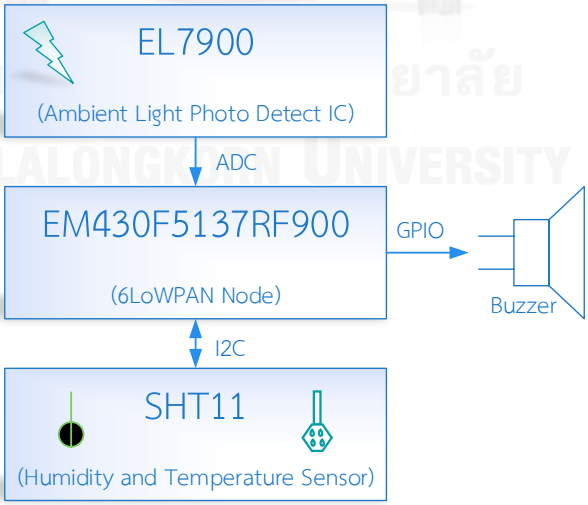
3.4.2.4. การออกแบบวงจรตัดต่อไฟฟ้า



รูปที่ 3-11 วงจรตัดต่อไฟฟ้า

วงจรตัดต่อไฟฟ้าดังแสดงในรูปที่ 3-11 ได้ใช้ Relay เป็นอุปกรณ์ที่ทำการตัดต่อไฟฟ้าให้แก่เครื่องใช้ไฟฟ้า ทรานซิสเตอร์ TR1 ทำหน้าที่เป็นตัวจ่ายกระแสไฟฟ้าไปควบคุมการทำงานของ Coil Relay โดยจะถูกควบคุมการทำงานโดยไมโครคอนโทรเลอร์ ซึ่งเมื่อแรงดันไฟฟ้าที่ขา Base มากกว่า 0.7 โวลต์ทำให้ทรานซิสเตอร์ทำงาน ปิดวงจร และเมื่อแรงดันไฟฟ้าที่ขา Base ต่ำกว่า 0.7 โวลต์ทรานซิสเตอร์จะไม่ทำงาน เปิดวงจร ส่วนไดโอด D1 มีหน้าที่ในการป้องกันการไหลกลับของกระแสไฟฟ้าที่เกิดจากสนามแม่เหล็กของ Coil Relay ยุบตัวขณะที่ทรานซิสเตอร์เปลี่ยนสถานะจากปิดวงจรไปเปิดวงจร

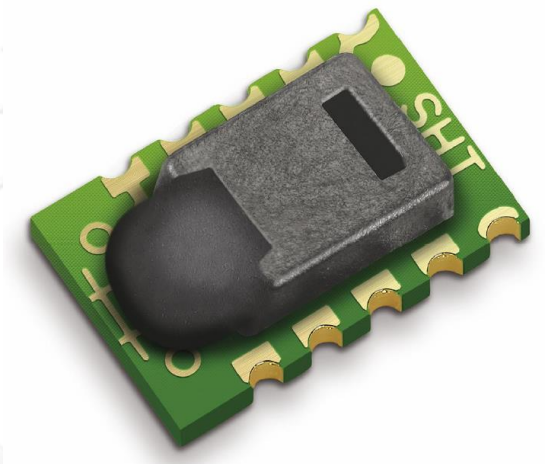
3.4.3. การออกแบบโหนดวัดสภาวะแวดล้อม



รูปที่ 3-12 บล็อกไดอะแกรมของโหนดวัดสภาวะแวดล้อม

ภายโหนดวัดสภาวะแวดล้อมดังแสดงในรูปที่ 3-12 จะประกอบไปด้วย เซ็นเซอร์วัดอุณหภูมิ, เซ็นเซอร์วัดความชื้น, เซ็นเซอร์วัดความเข้มแสง และ Buzzer ซึ่งค่าของเซ็นเซอร์เหล่านี้ความความสำคัญต่อการใช้พลังงานภายในอาคารเป็นอย่างมาก ตัวอย่างเช่น ค่าของอุณหภูมิ และความชื่น จะส่งผลต่อการใช้งานเครื่องปรับอากาศ ค่าความเข้มแสงมีผลต่อการใช้งานไฟส่องสว่างภายในอาคาร เป็นต้น ซึ่งการวัดค่าเหล่านี้แล้วนำมาทำการควบคุมการทำงานเครื่องใช้ไฟฟ้าอย่างเหมาะสมจะทำให้ลดการใช้พลังงานไฟฟ้าลงได้เป็นอย่างมาก การติดต่อระหว่าง EM430F5137RF900 กับเซ็นเซอร์วัดอุณหภูมิ และความชื้นจะใช้งานอินเตอร์เฟส Inter-Integrated Circuit (I^2C) ส่วนเซ็นเซอร์วัดความเข้มแสงจะใช้วงจรแปลงแอนะล็อกเป็นดิจิตอลในการอ่านค่าของแรงดัน และ Buzzer นั้นจะใช้ในการแจ้งเตือนผู้ใช้งานในห้องนั้นๆว่า เกิดเหตุการณ์ที่สำคัญต่างๆขึ้น

3.4.3.1. คุณสมบัติของเซ็นเซอร์วัดอุณหภูมิ และความชื้น



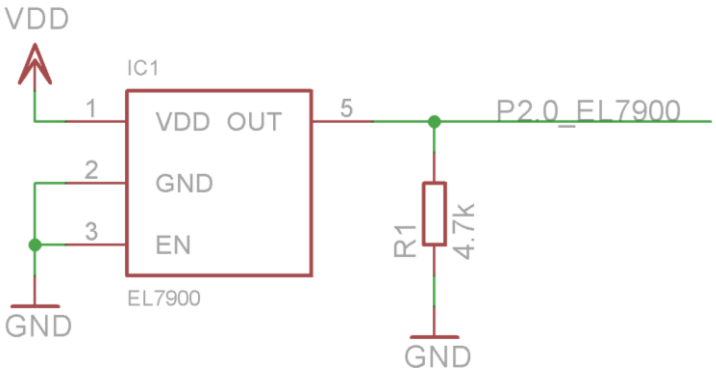
รูปที่ 3-13 เซ็นเซอร์วัดอุณหภูมิ และความชื้น SHT11

- **ทำการสอบเทียบค่ามาจากโรงงาน**
- **ใช้งานพลังงานต่ำ**
- **ยังคงมีเสถียรภาพสูงเมื่อผ่านการใช้งานที่นาวนาน**
- **มีค่าความผิดพลาดของการวัดความชื้นที่ ±3.0 %RH**
- **มีค่าความผิดพลาดของการวัดอุณหภูมิที่ ±0.4 °C**

3.4.3.2. คุณสมบัติของเซ็นเซอร์วัดความเข้มแสง และการออกแบบวงจร

รูปที่ 3-14 เซ็นเซอร์วัดความเข้มแสง EL7900

- สามารถวัดความเข้มแสงได้ 1 lux ถึง 8000 lux
- มีเอาท์พุตเป็นกระแสที่เปลี่ยนแปลงตามความเข้มแสง
- ใช้กระแสเลี้ยงต่ำกว่า 1 ไมโครแอมป์
- เวลาในการตอบสนองต่ำกว่า 200 µs
- เอาท์พุตมีความสัมพันธ์ต่อความเข้มแสงในแบบเชิงเส้น
- มีขนาดเล็ก น้ำหนักเบา และมีตัวถังเป็นแบบ surface mount



รูปที่ 3-15 วงจรของเซ็นเซอร์วัดความเข้มแสง EL7900

เนื่องจากเอาท์พุตของ EL7900 นั้นเป็นกระแสจึงต้องแปลงเป็นแรงดันก่อนเพื่อให้วงจรแปลงแอนะล็อกเป็นดิจิตอลสามารถที่จะอ่านค่าได้ โดยการต่อตัวต้านทานที่เอาท์พุตของ EL7900 ดังแสดงในรูปที่ 3-15 ซึ่งสามารถแสดงความสัมพันธ์ของแรงดันเอาท์พุตต่อความเข้มแสง ได้ดังสมการต่อไปนี้

ความสัมพันธ์ของความเข้มแสงต่อกระแสเอาท์พุต

$$I_{out} = \frac{60\mu A}{100lux} \times L_{INPUT} \tag{3-14}$$

โดย I_{out} คือ กระเอาท์พุตในหน่วย μA และ L_{INPUT} คือความเข้มแสงในหน่วย lux

ซึ่งความสัมพันธ์ต่อแรงดันไฟฟ้า

$$V_{OUT} = I_{OUT} \times R_{LOAD} = \frac{60\mu A}{100lux} \times L_{INPUT} \times R_{LOAD} \tag{3-15}$$

$$V_{OUT} = \frac{60\mu A}{100lux} \times L_{INPUT} \times R_{LOAD} = 2.82mV/lux \tag{3-16}$$

3.5. การออกแบบฮาร์ดแวร์โปรแกรมประยุกต์

รูปที่ 3-16 โทรศัพท์มือถือ Android รุ่น Galaxy Mega 6.3

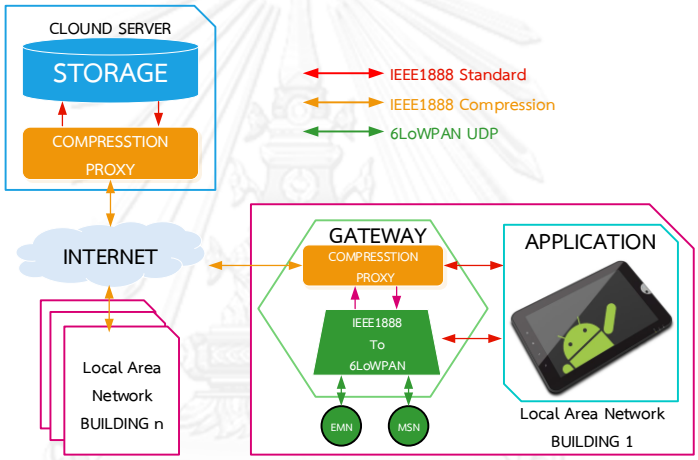
ฮาร์ดแวร์ของโปรแกรมประยุกต์ได้นำโทรศัพท์มือถือที่มีระบบปฏิบัติการ Android รุ่น Galaxy Maga 6.3 ดังแสดงในรูปที่ 3-16 มาทำการพัฒนาโปรแกรม อันเนื่องมาจากระบบปฏิบัติการ Andorid นั้นเป็น Open Source ซึ่งทำให้ไม่ต้องเสียค่าใช้จ่ายในการซื้อซอฟต์แวร์ในการพัฒนา อีกทั้งยังสามารถหาข้อมูลในการพัฒนาโปรแกรมได้ง่าย

บทที่ 4

การออกแบบซอร์ฟแวร์ระบบ

ในบทนี้ได้กล่าวถึงการออกแบบซอฟต์แวร์ระบบ ซึ่งอธิบายถึงการทำงานร่วมกันของซอฟต์แวร์ภายในระบบ และการออกแบบซอฟต์แวร์ในส่วนต่างๆ ที่ประกอบไปด้วยซอฟต์แวร์ Proxy ที่ทำการบีบอัดข้อมูลในมาตรฐาน IEEE1888 โดยใช้กระบวนการบีบอัด EXI ซอฟต์แวร์แปลงรูปแบบข้อมูล IEEE1888 ไปยัง 6LoWPAN ซอฟต์แวร์ 6LoWPAN ในโหนดวัดสภาพแวดล้อม และโหนดวัดพลังงาน รวมถึงแอพพลิเคชั่นจัดการการใช้พลังงานภายในอาคารบนระบบปฏิบัติการ Android

4.1. การออกแบบซอร์ฟแวร์ภายในระบบ



รูปที่ 4-1 ซอฟต์แวร์ภายในระบบ

ซอร์ฟแวร์ที่นำเสนอในงานวิจัยนี้ดังแสดงในรูปที่ 4-1 ได้ถูกออกแบบมาเพื่อให้ใช้งานกับข้อมูลที่แตกต่างกันไปตามรูปแบบของระบบเครือข่าย ซึ่งประกอบไปด้วยซอร์ฟแวร์ 4 ตัว คือ 1. ซอฟต์แวร์ Proxy ที่บีบอัดข้อมูลในมาตรฐาน IEEE1888 2. ซอฟต์แวร์แปลงข้อมูลในมาตรฐาน IEEE1888 ให้อยู่ในมาตรฐาน 6LoWPAN 3. ซอฟต์แวร์โหนดวัดพลังงานไฟฟ้า และโหนดวัดสภาวะแวดล้อมบนมาตรฐาน 6LoWPAN 4. โปรแกรมประยุกต์สำหรับบริหารจัดการพลังงานภายในอาคารบนระบบปฏิบัติการ Android [14] โดยซอฟต์แวร์แต่ละตัวนั้นมีหน้าที่การทำงานดังต่อไปนี้

1. ซอฟต์แวร์ Proxy ที่บีบอัดข้อมูลในมาตรฐาน IEEE1888 จะถูกติดตั้งใน Cloud สตอเรจ และเกทเวย มีหน้าที่ในการบีบอัดข้อมูล IEEE1888 โดยใช้การบวนการบีบอัด EXI เพื่อให้มีขนาดข้อมูลเล็กลง ซึ่งเหมาะแก่การส่งข้อมูลที่มีขนาดใหญ่ในระบบเครือข่ายที่มีการจราจรหนาแน่น ดังนั้นคอมโพเน้นท์ทุกตัวภายในอาคารเมื่อต้องการจะสื่อสารข้อมูลกับสตอเรจจะต้องสื่อสารข้อมูลผ่าน Proxy นี้เท่านั้น

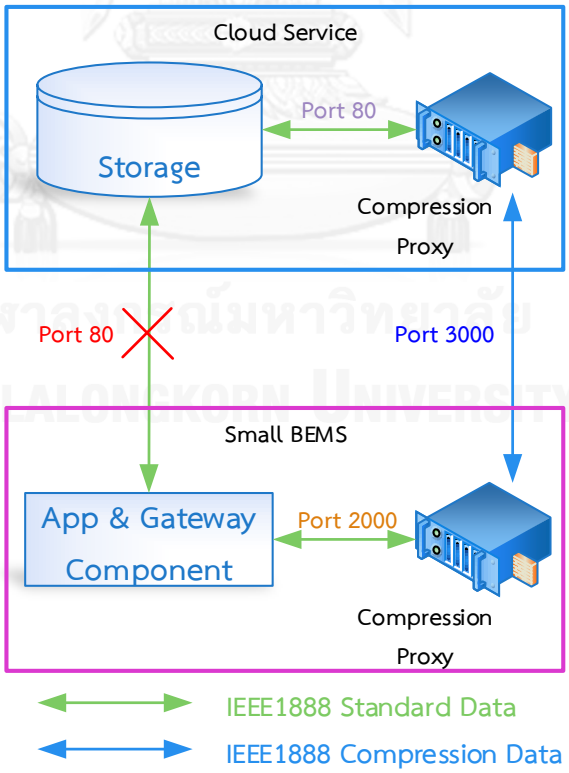
2. ซอฟต์แวร์แปลงข้อมูลในมาตรฐาน IEEE1888 ให้อยู่ในมาตรฐาน 6LoWPAN ซึ่งถูกติดตั้งอยู่ในเกทเวยจะทำการแปลงข้อมูล FETCH, WRITE, และ TRAP ในมาตรฐาน IEEE1888 ให้สามารถสื่อสารกับข้อมูลมาตรฐาน 6LoWPAN ที่ถูกออกแบบขึ้นมาใหม่ได้ ซึ่งภายในซอฟต์แวร์ตัวนี้จะเก็บค่าพารามิเตอร์ (Parameter) ที่จำเป็นต่อการแปลงรูปแบบของข้อมูลของทั้งสองมาตรฐานไว้ในไฟล์ตั้งค่า

3. ซอฟต์แวร์โหนดวัดพลังงานไฟฟ้า และโหนดวัดสภาวะแวดล้อมบนมาตรฐาน 6LoWPAN เป็นซอฟต์แวร์ที่ทำงานบนไมโครคอนโทรเลอร์ ซึ่งรับการสั่งงานจากซอฟต์แวร์ในข้อที่ 2 ซึ่งสื่อสารข้อมูลโดยโพรโทคอล UDP มายัง 6LoWPAN โหนดเพื่อทำการอ่านค่าจากเซ็นเซอร์ และควบคุมการทำงานของ Actuator

4. โปรแกรมประยุกต์สำหรับบริหารจัดการพลังงานภายในอาคารบนระบบปฏิบัติการ Android ทำหน้าที่ในการแสดงผลข้อมูลของระบบจัดการพลังงานไฟฟ้าภายในอาคาร ควบคุมการทำงานของเครื่องใช้ไฟฟ้าที่ได้มาจากโหนดวัดพลังงาน และโหนดวัดสภาพแวดล้อม อีกทั้งยังสามารถวิเคราะห์การใช้พลังงานในแบบเบื้องต้นได้

4.2. การออกแบบ Proxy ที่บีบอัดข้อมูลในมาตรฐาน IEEE1888

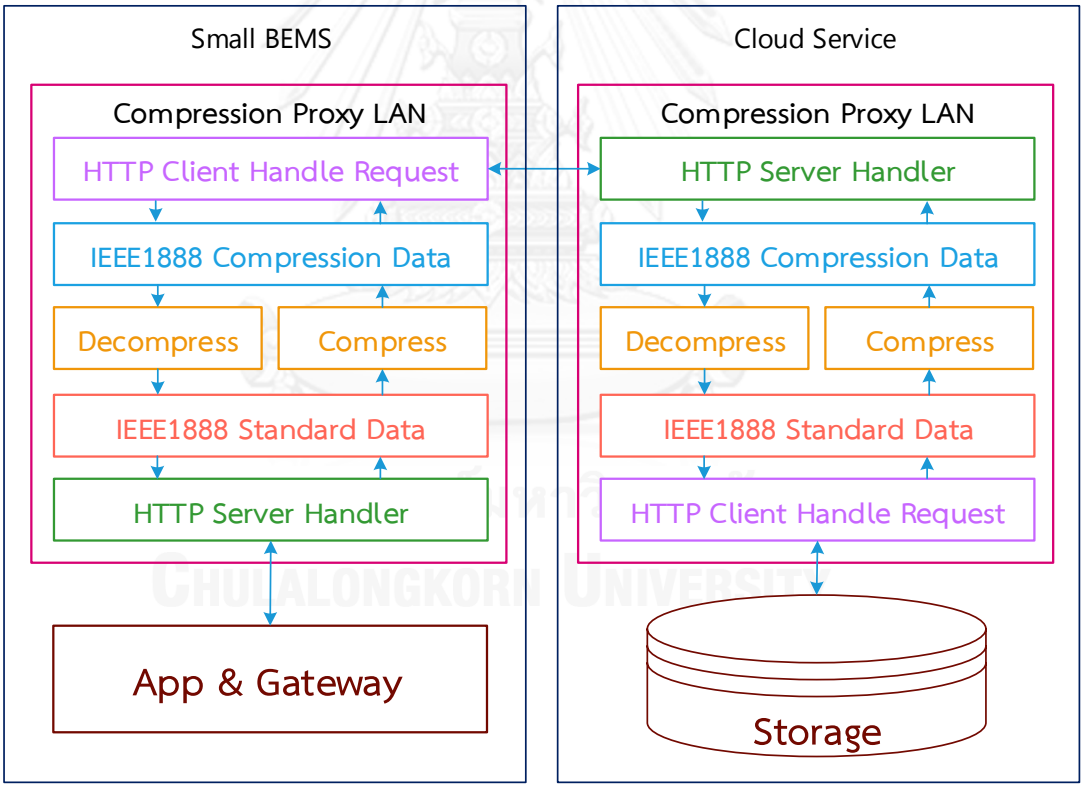
4.2.1. การออกแบบระบบเครือข่ายของ Compression Proxy



รูปที่ 4-2 การออกแบบระบบเครือข่ายของ Compression Proxy

ปกติแล้วคอมโพเน้นท์ภายในมาตรฐาน IEEE1888 จะสื่อสารข้อมูลระหว่างกันโดยใช้งานพอร์ต 80 ดังแสดงในรูปที่ 4-2 รูปแบบข้อมูลที่ใช้ในการสื่อสารนั้นจะไม่มีการบีบอัด แต่ระบบที่ได้นำเสนอในงานวิจัยนี้ได้ทำการปรับปรุงโครงสร้างของระบบ โดยทำการเพิ่ม Proxy บีบอัดข้อมูลที่ใช้งานกระบวนการบีบอัด EXI เข้ามาช่วยในการลดขนาดของข้อมูล ซึ่ง Proxy นี้จะถูกติดตั้งทั้งสองฝั่ง คือ ฝั่งผู้ให้บริการ Clound สตอเรจ และฝั่งอาคารที่ติดตั้งระบบบริหารจัดการพลังงาน ในฝั่งผู้ให้บริการ Clound สตอเรจ Proxy จะเปิดพอร์ต 3000 เพื่อให้คอมโพเน้นจากภายนอกสามารถสื่อสารข้อมูลในรูปแบบที่มีการบีบอัดไปยังสตอเรจได้ ส่วนพอร์ต 80 ก็คงสามารถสื่อสารข้อมูลได้ตามปกติ ในฝั่งอาคารที่ติดตั้งระบบบริหารจัดการพลังงานนั้นจะเปิดพอร์ต 2000 ให้คอมโพเน้นท์ภายใน LAN สามารถสื่อสารข้อมูลโดยผ่าน Proxy นี้ไปยัง Clound สตอเรจที่พอร์ต 3000 อีกทั้ง Proxy ตัวนี้ยังทำหน้าที่ป้องกันการสื่อสารข้อมูลจากภายนอกที่ไม่พึงประสงค์เข้ามายัง LAN ได้อีกด้วย ดังนั้นคอมโพเน้นภายในอาคารไม่ควรที่จะสื่อสารข้อมูลโดยตรงกับสตอเรจ ซึ่งสามารถดูได้จากรูปว่าจะมีเครื่องหมายกากบาทช่องทางปกติที่ใช้ในสื่อสารข้อมูล

4.2.2. การทำงานของ Compression Proxy ซอฟต์แวร์



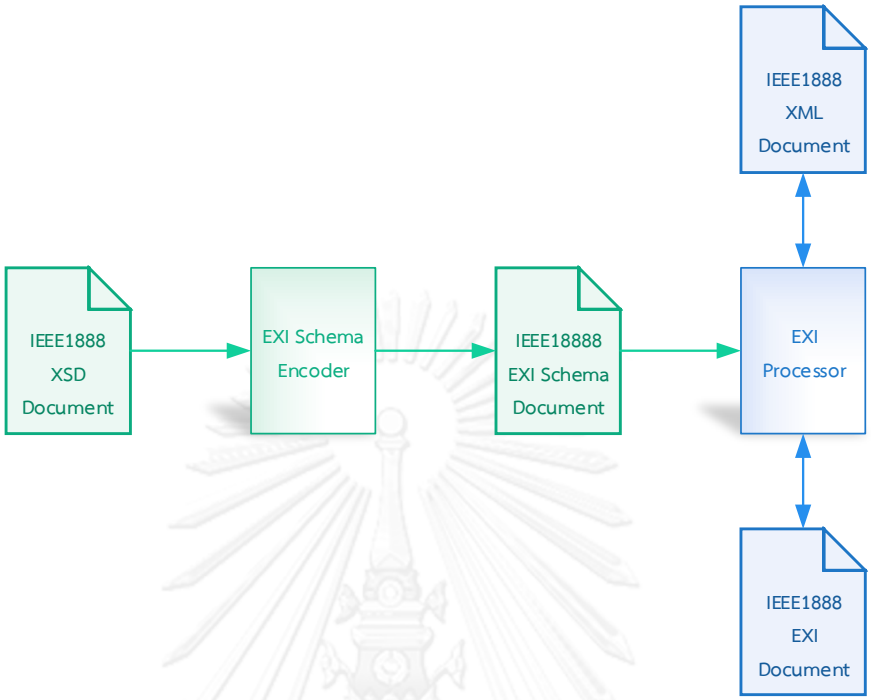
รูปที่ 4-3 การทำงานของ Compression Proxy

ซอฟต์แวร์ Compression Proxy ที่ได้ทำการออกแบบนั้นสามารถแสดงการทำงานได้ดังรูปที่ 4-3 ซึ่ง Proxy โดยประกอบไปด้วย Proxy ที่ทำงานภายในระบบเครือข่าย LAN และ Proxy ที่ทำงานในผู้ให้บริการ Cloud สตอเรจ ซึ่ง Proxy ทั้งสองนั้นมีการทำงานที่คล้ายคลึงกันเพียงแต่สลับการทำงานในส่วนของ HTTP Client และ HTTP Server เท่านั้น โดย Proxy นี้ได้นำเอาซอฟต์แวร์ Membrane Service Proxy [15] ที่เป็น Open Source มาทำการพัฒนาเพิ่มเติม

การทำงานของ Compression Proxy ในฝั่ง LAN นั้นจะมี HTTP Server (บล็อกสีเขียว) รอรับข้อมูลจากคอมโพเน้นท์แอปพลิเคชั่น และเกทเวย ที่ต้องการจะสื่อสารข้อมูลไปยังสตอเรจ จากนั้นข้อมูลที่ได้รับมาจะถูกออกเป็น 2 ส่วน คือ HTTP Header และ HTTT Body ที่เป็นข้อมูลในมาตรฐาน IEEE1888 (บล็อกสีส้ม) ข้อมูลในส่วน Body นี้จะถูกนำไปทำการบีบอัดโดยกระบวนการบีบอัด EXI แล้วถูกแปลงให้รูปแบบของ Binary Encoding (บล็อกสีเหลือง) ข้อมูลภายหลังการบีบอัดนี้จะถูกนำไปรวมกับ HTTP Header เดิมที่ถูกแยกออกไป (บล็อกสีฟ้า) แล้วทำการส่งไปยัง Proxy ที่ทำงานในผู้ให้บริการ Clound สตอเรจ โดยมี HTTP Client (บล็อกสีม่วง) ที่ทำหน้าที่เป็นตัวจัดการรับส่งข้อมูลอยู่ภายใน Proxy หลังจากนั้นเมื่อ HTTP Client ได้รับข้อมูลตอบกลับจะทำการแยกข้อมูลในส่วนของ HTTP Header และ HTTP Body ที่เป็นข้อมูลที่มีการบีบอัดโดย EXI ออกจากกัน แล้วนำข้อมูลนี้ไปทำการคลายการบีบอัดจะได้ข้อมูลในมาตรฐาน IEEE1888 ปกติ ซึ่งจะถูกนำไปรวมกับ HTTP Header เดิมที่ถูกแยกออกไป ข้อมูลสมบูรณ์นี้จะถูก HTTP Server ตอบกลับไปยังคอมโพเน้นท์ ที่ทำการส่งการร้องขอมาในตอนเริ่มต้นของกระบวนการ

การทำงานของ Compression Proxy ในฝั่งผู้ให้บริการ Cloud สตอเรจ นั้นจะมีการทำงานคล้ายคลึงกับ Compression Proxy ในฝั่ง LAN เพียงแต่มีการสลับกันในส่วนของ HTTP Server กับ HTTP Client ให้ผู้ใช้งานจากอินเทอร์เน็ตสามารถใช้งาน Proxy นี้ได้ ซึ่งการทำงานของ Proxy นี้จะมี HTTP Server (บล็อกสีเขียว) รอรับข้อมูลจาก Compression Proxy ในฝั่ง LAN ที่ต้องการจะสื่อสารข้อมูลไปยังสตอเรจ จากนั้นข้อมูลที่ได้รับมาจะถูกออกเป็น 2 ส่วน คือ HTTP Header และ HTTT Body ที่เป็นข้อมูลที่มีการบีบอัดโดย EXI (บล็อกสีฟ้า) ข้อมูลในส่วน Body นี้จะถูกนำไปคลายการบีบอัดโดยกระบวนการ EXI (บล็อกสีเหลือง) จะได้ข้อมูลออกมาจะมีรูปแบบเหมือนกับที่ส่งจากคอมโพเน้นท์ในฝั่งของ LAN ข้อมูลภายหลังการบีบอัดนี้จะถูกนำไปรวมกับ HTTP Header เดิมที่ถูกแยกออกไป (บล็อกสีฟ้า) แล้วทำการส่งไปยังสตอเรจตอเรจ โดยมี HTTP Client (บล็อกสีม่วง) ที่ทำหน้าที่เป็นตัวจัดการรับส่งข้อมูลอยู่ภายใน Proxy หลังจากนั้นเมื่อ HTTP Client ได้รับข้อมูลตอบกลับจากสตอเรจ ข้อมูลนี้จะถูกแยกส่วนของ HTTP Header และ HTTP Body ที่เป็นข้อมูลตามมาตรฐาน IEEE1888 ออกจากกัน แล้วนำข้อมูลนี้ไปทำการบีบอัดแล้วนำไปรวมกับ HTTP Header เดิมที่ถูกแยกออกไป ข้อมูลสมบูรณ์นี้จะถูก HTTP Server ตอบกลับไปยัง Proxy ที่ทำการส่งข้อมูลร้องขอในตอนเริ่มต้นของกระบวนการ

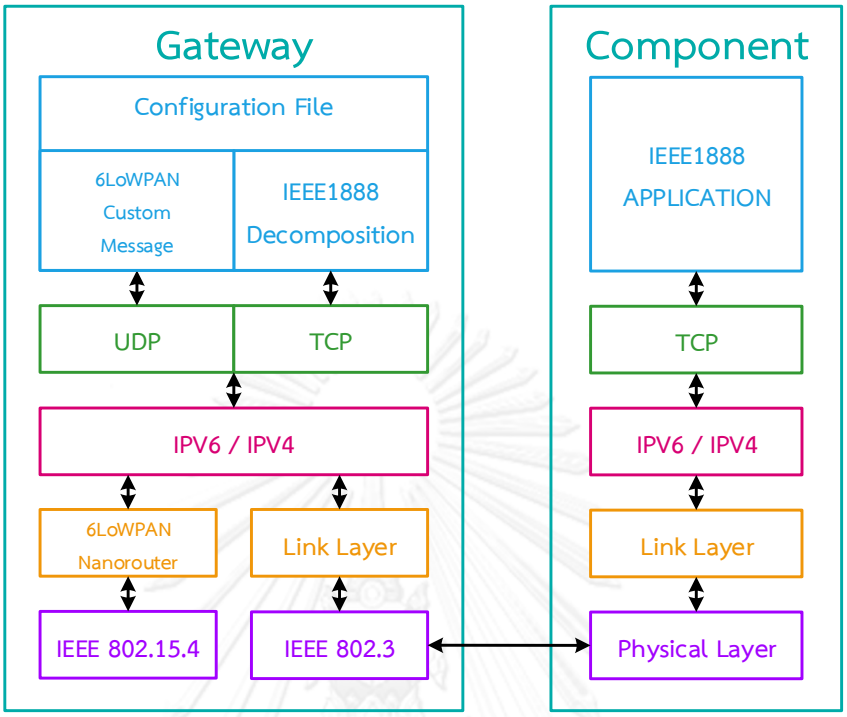
4.2.3. การทำงานของกระบวนการบีบอัด EXI



รูปที่ 4-4 การทำงานของกระบวนการบีบอัด EXI

การทำงานของกระบวนการบีบอัด EXI ดังแสดงในรูปที่ 4-4 จะเห็นได้ว่าในการทำงานของ EXI Processor [16] ที่ทำการบีบอัด หรือ คลายการบีบอัดข้อมูลในมาตรฐาน IEEE1888 นั้นจะใช้งาน XML Schema Definition (XSD) ร่วมด้วย โดย XSD นี้จะอธิบายถึงโครงสร้างข้อมูล XML ในมาตรฐาน IEEE1888 ว่าภายในข้อมูลนั้นประกอบไปด้วย Element และ Attribute ชื่ออะไรบ้าง แล้วมีการเก็บข้อมูลชนิดใดลงไป เพื่อทำให้ EXI Processor ทำการบีบอัดข้อมูลอย่างมีประสิทธิภาพ ตัวอย่างเช่น ข้อมูลตัวเลข integer ขนาด 8 บิตมีค่าสูงสุด 255 ถ้าส่งตัวเลขนี้เป็น XML ตามปกติจะใช้พื้นที่ 3 ไบต์ แต่ถ้าหาก EXI Processor รู้ถึงชนิดของข้อมูล จะสามารถทำการแปลงตัวเลขนี้ให้อยู่ในรูปของมูลตัวเลขที่มีขนาดเพียง 1 ไบต์เท่านั้น อีกทั้ง EXI Processor สามารถละเลยชื่อของ Element และ Attribute ในข้อมูลที่ผ่านการบีบอัดไปแล้ว เพราะข้อมูลเหล่านี้สามารถอ่านได้จากไฟล์ XSD ที่ปลายทางเมื่อทำการคลายการบีบอัด จากในรูปจะเห็นได้ว่าข้อมูล XSD จะถูกแปลงให้อยู่ในรูปแบบของ EXI Schema ก่อนเพื่อที่ทำให้เหมาะสมต่อการประมวลผลของ EXI Processor

4.3. การออกแบบซอฟต์แวร์เชื่อมต่อมาตรฐาน IEEE1888 กับมาตรฐาน 6LoWPAN



รูปที่ 4-5 ลำดับชั้นของซอฟต์แวร์แปลงข้อมูลระหว่างมาตรฐาน IEEE1888 – 6LoWPAN

ซอฟต์แวร์เชื่อมต่อมาตรฐาน IEEE1888 กับมาตรฐาน 6LoWPAN มีการทำงานดังแสดงในรูปที่ 4-5 ซึ่งมีการเชื่อมต่อ Physical สองชนิดเข้าด้วยกัน คือ IEEE 802.15.4 ที่ใช้งานสื่อสารกับโหนด 6LoWPAN และ IEEE 802.3 ที่เป็นระบบเครือข่าย Ethernet ที่ใช้สาย LAN ในการเชื่อมต่อตามปกติ ซึ่งในส่วนของการสื่อสาร 6LoWPAN กับ Physical IEEE 802.15.4 จะถูกจัดการโดย ซอฟต์แวร์ Nanorouter ของบริษัท TI

การออกแบบโปรแกรมในชั้นประยุกต์นั้นประกอบไปด้วยสองส่วนหลัก คือ 1. IEEE1888 Decomposition จะมีหน้าที่ในการสื่อสารข้อมูลกับมาตรฐาน IEEE1888 ผ่านการเชื่อมต่อแบบ TCP และถอดส่วนประกอบข้อมูลในมาตรฐาน IEEE1888 ที่จำเป็นต่อการสร้างข้อมูลที่จะสื่อสารไปยัง 6LoWPAN โหนด เช่น Point ID, Value, และชนิดของ Common Communication โพรโทคอล 2. 6LoWPAN Custom Message จะนำข้อมูลที่ได้จาก IEEE1888 Decomposition ไปสร้างเป็นข้อมูลที่จะใช้ในการสื่อสารกับ 6LoWPAN โหนด ผ่านการเชื่อมต่อแบบ UDP ซึ่งการแปลง

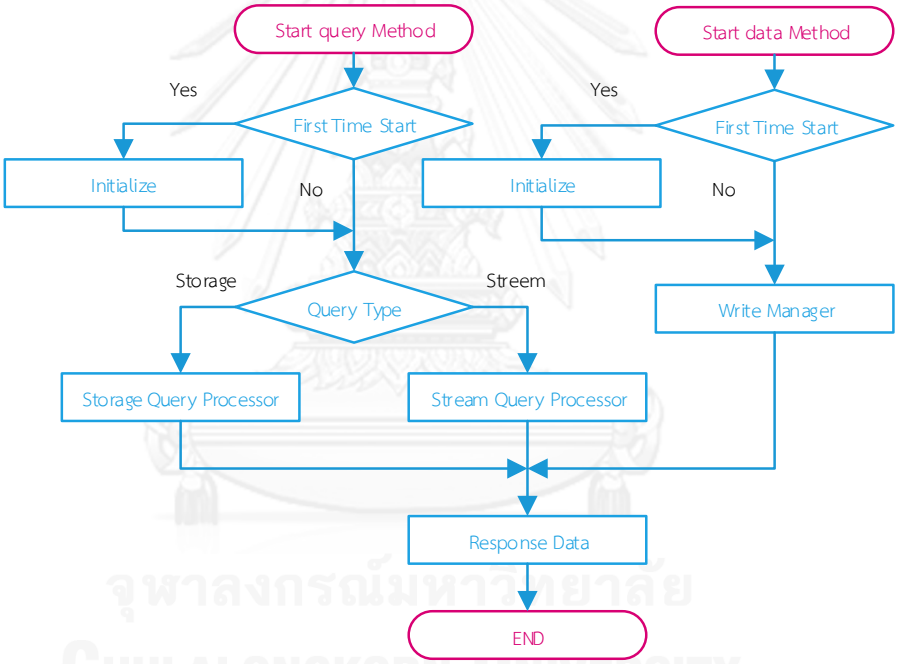
ข้อมูลระหว่างทั้งสองส่วนนั้นสามารถเชื่อมโยงกันโดยใช้ Configuration File เก็บข้อมูลที่ใช้ในการ Mapping ที่ประกอบไปด้วย

1. IEEE1888 Point ID: เป็นตัวแทนของชื่อของข้อมูล 1 ข้อมูลที่จะถูกเขียน หรืออ่าน จาก 6LoWPAN โหนด ซึ่งจะเชื่อมโยงกับข้อมูล EUI-64 และ Name

2. EUI-64: เป็นหมายเลข Media Access Control (MAC) ของแต่ละ 6LoWPAN โหนดที่ใช้เป็นตัวที่ใช้ในการสร้างความเชื่อมโยงไปยังหมายเลข IP Address ของ 6LoWPAN โหนด เพื่อใช้ในการกำหนดปลายทางในการสื่อสารข้อมูล UDP

3. Name: เป็นตัวระบุถึงข้อมูลภายใน 6LoWPAN โหนดเพียง 1 ข้อมูล เนื่องจากภายในโหนดหนึ่งๆนั้นประกอบไปด้วยข้อมูลจากเซ็นเซอร์ และ Actuators จำนวนหลายตัว ดังนั้นจึงได้ออกแบบให้ Name เป็นตัวสื่อถึงข้อมูลที่ได้จากเซ็นเซอร์ หรือ Actuators จำนวน 1 ตัว เพื่อให้ง่ายต่อการแยกแยะข้อมูล

4.3.1. การทำงานของซอฟต์แวร์เชื่อมต่อมาตรฐาน IEEE1888 กับมาตรฐาน 6LoWPAN



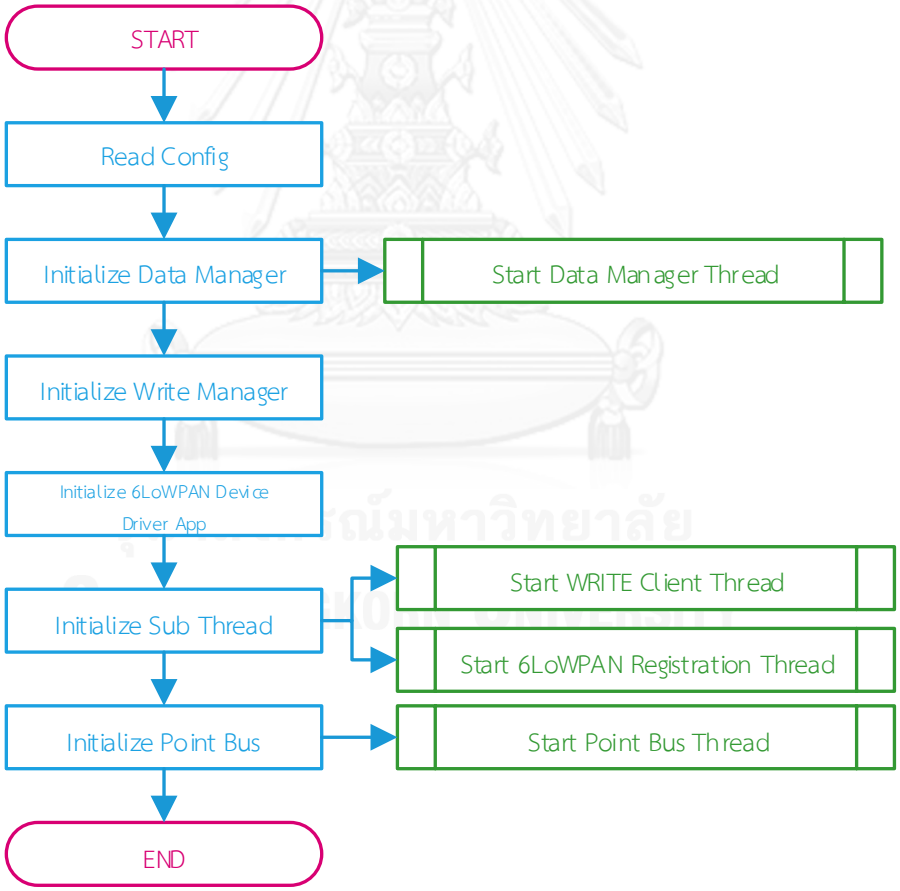
รูปที่ 4-6 โฟลว์ชาร์ตแสดงการทำงานของซอฟต์แวร์เชื่อมต่อมาตรฐาน IEEE1888 กับมาตรฐาน 6LoWPAN

การพัฒนาซอฟต์แวร์เชื่อมต่อมาตรฐาน IEEE1888 กับมาตรฐาน 6LoWPAN ได้ถูกถูกพัฒนาบน Apache Axis2™ Web Services / SOAP / WSDL engine Version 1.5.6 [17-19] ซึ่งมีการทำงานของโปรแกรมหลักดังแสดงในรูปที่ 4-6 จากโฟลว์ชาร์ตจะมี 2 ช่องทางในการเริ่มการทำงานของซอฟต์แวร์นี้ ประกอบไปด้วย ช่องทาง query เมื่อได้รับข้อมูลจาก FETCH และ TRAP โพรโทคอล ช่องทาง data เมื่อได้รับข้อมูลจาก WRITE โพรโทคอล โดยมีรายละเอียดของการทำงานของแต่ช่องทางดังนี้

เมื่อได้รับข้อมูลในช่องทาง query จากนั้นจะทำการเช็คว่าระบบเริ่มต้นการทำงานเป็นครั้งแรกหรือไม่ ถ้าหาระบบเริ่มต้นการทำงานเป็นครั้งแรกจำเป็นต้องทำการกำหนดค่าเริ่มต้นต่างๆของระบบก่อนที่จะทำงานในขั้นตอนถัดไป ขั้นตอนถัดไปจะเป็นการตรวจสอบชนิดของข้อมูล query ว่าเป็น storage หรือ stream เพื่อเป็นการแยกแยะข้อมูลว่าเป็น FETCH หรือ TRAP โพรโทคอล เพื่อส่งต่อข้อมูลไปยังส่วนประมวลผลของแต่ละโพรโทคอลได้อย่างถูกต้อง ภายหลังการประมวลผลจะทำการตอบกลับไปยัง client คอมโพเน้นท์ตามการร้องขอ

เมื่อได้รับข้อมูลในช่องทาง data จากนั้นจะทำการเช็คว่าระบบเริ่มต้นการทำงานเป็นครั้งแรกหรือไม่ ถ้าหาระบบเริ่มต้นการทำงานเป็นครั้งแรกจำเป็นต้องทำการกำหนดค่าเริ่มต้นต่างๆของระบบก่อนที่จะทำงานในขั้นตอนถัดไป ขั้นตอนถัดไปจะส่งต่อข้อมูลไปยังส่วนประมวลผลในตัวจัดการเขียนข้อมูล ภายหลังการเขียนข้อมูลจะทำการตอบกลับไปยัง client คอมโพเน้นท์ว่าสำเร็จหรือไม่

4.3.1.1. การทำงานของโปรมแกรมย่อยตั้งค่าเริ่มต้น (Initialize)



รูปที่ 4-7 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมย่อยตั้งค่าเริ่มต้น

โปรแกรมย่อยตั้งค่าเริ่มต้นจะทำการกำหนดค่าเริ่มต้นต่างๆของซอฟต์แวร์เชื่อมต่อมาตรฐาน IEEE1888 กับมาตรฐาน 6LoWPAN ทั้งหมด ซึ่งแบ่งออกเป็น 6 ส่วนดังต่อไปนี้

- **Read Config: จะทำการอ่านข้อมูลในไฟล์เก็บข้อมูลการตั้งค่าเก็บไว้ในตัวแปลภายในโปรแกรม เพื่อให้ง่ายต่อการเรียกใช้งาน**

- **Initialize Data Manager: จะทำการกำหนดค่าเริ่มต้นของ DataManager Object แล้วเริ่มต้นการทำงาน Thread ที่ทำการควบคุม QueryProcessor Object ที่ประกอบไปด้วย Object 2 ชนิดคือ StreamQueryProcessor และ StorageQueryProcessor ซึ่ง Thread จะทำการตรวจสอบ Time to Live (TTL) ของ Object ทั้งสองว่าเกินเวลาที่กำหนดแล้วหรือยัง ถ้าหากเกินเวลาที่กำหนดจะทำการลบ Object ตัวนั้นทิ้งออกจากคิวไป**

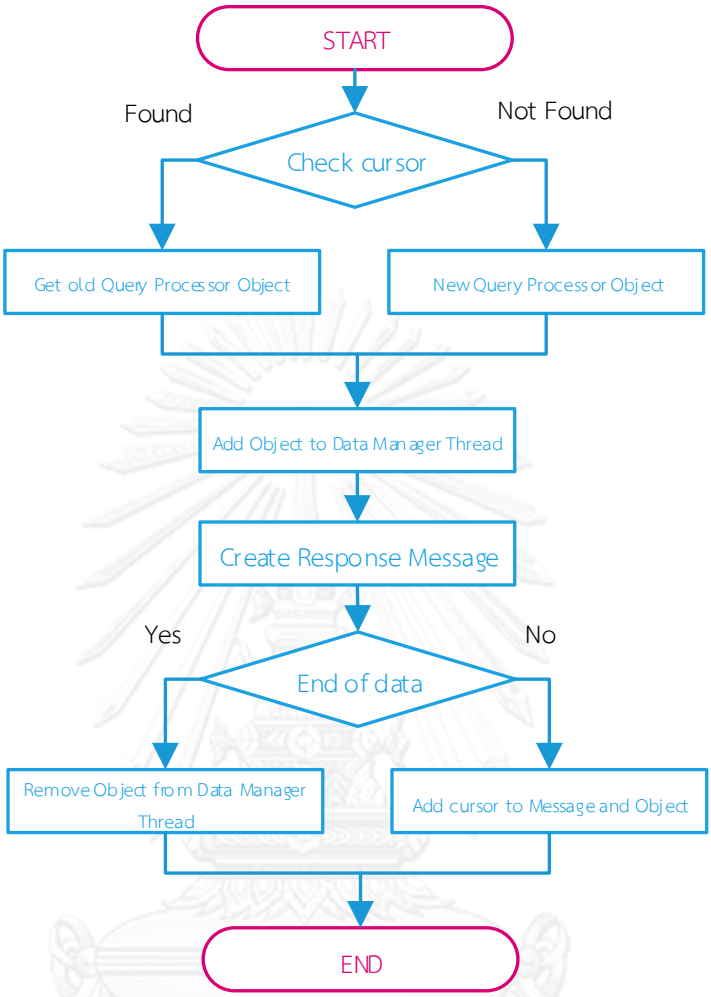
- **Initialize Write Manager: จะทำการกำหนดค่าเริ่มต้นของ Write Manager Object**

- **Initialize 6LoWPAN Device Driver App: จะทำการกำหนดค่าเริ่มต้นของ 6LoWPAN Device Driver App Object**

- **Initialize Sub Thread: จะทำการกำหนดค่าเริ่มต้นของ Object WRITE Client และ 6LoWPAN Registration แล้วเริ่มต้นการทำงาน Thread ทั้งสอง โดย WRITE Client Thread จะทำการอ่านข้อมูลจาก 6LoWPAN โหนดแล้วทำการ WRITE ไปยังสตอเรจทุกๆ 1 นาที ส่วน 6LoWPAN Registration Thread มีหน้าที่ในการรับข้อมูลลงทะเบียนที่ได้มาจาก 6LoWPAN โหนดเมื่อทำการเชื่อมต่อเข้ากับระบบเป็นครั้งแรกโดยภายในข้อมูลนี้ประกอบไปด้วย EUI-64 และหมายเลข IPv6**

- **Initialize Point Bus: จะทำการกำหนดค่าเริ่มต้นของ Point Bus Object แล้วเริ่มต้นการทำงาน Thread โดย Thread นี้จะหน้าที่ในการ Update ข้อมูล Value ที่เกี่ยวข้องกับ Point ID ทั้งหมดภายในเกทเวย ซึ่งเชื่อมต่อการทำงานกับ 3 Object ได้แก่ Data Manager, Write Manager, และ SIXLOWPAN Device Driver App โดยการเปลี่ยนแปลงข้อมูลไดข้อมูลหนึ่งภายใน Object เหล่านี้จะส่งให้ข้อมูลใน Object อื่นเปลี่ยนแปลงตามไปด้วย**

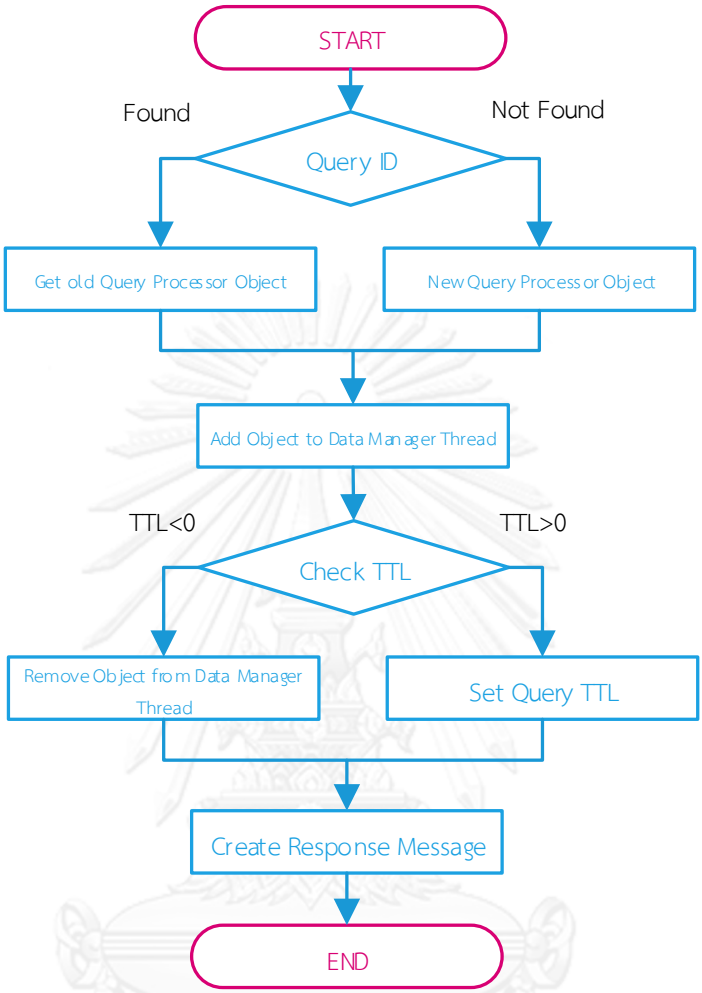
4.3.1.2. การทำงานของโปรมแกรมย่อย Storage Query Processor



รูปที่ 4-8 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมย่อย Storage Query Processor

โปรแกรมย่อย Storage Query Processor ดังแสดงในรูปที่ 4-8 จะทำการประมวลผลข้อมูล FETCH Request ที่ได้รับมาจาก remote คอมโพเน้นท์ โดยเริ่มจากการเช็ค Cursor ซึ่งเป็นตัวที่บอกถึงการสื่อสารข้อมูลที่ยังไม่แล้วเสร็จในครั้งก่อนหน้าภายในข้อมูล FETCH Request ที่ได้รับมาว่ามีหรือไม่ ถ้ามีจะทำการดึงเอา Object Storage Query Processor ในการสื่อสารก่อนหน้ามาใช้งาน แต่ถ้าไม่มีจะทำการสร้างใหม่ โดย Object ตัวนี้จะเก็บข้อมูลทั้งหมดของ FETCH Request ไว้เมื่อเกิดกรณีที่ไม่สามารถที่จะส่งข้อมูลทั้งหมดไปในครั้งเดียวได้ จากนั้น Object ตัวนี้จะถูกเก็บไว้ในตัวจัดการข้อมูล (Data Manager Thread) แล้วทำการสร้างข้อมูลตอบกลับ ซึ่งถ้าหากสามารถตอบกลับได้ในครั้งเดียวจะไม่มีการใส่ Cursor ลงไปในข้อมูลตอบกลับ และทำการลบ Object ออกจากตัวจัดการข้อมูล แต่ถ้าไม่สามารถจะมีการสร้าง Cursor แล้วทำการใส่ลงในข้อมูลตอบกลับ และ Object เพื่อใช้เป็นตัวในการแสดงถึงการสื่อสารข้อมูลยังไม่สิ้นสุด โดย Cursor นี้จะใช้ในการค้นหา Object ในการสื่อสารครั้งถัดไป

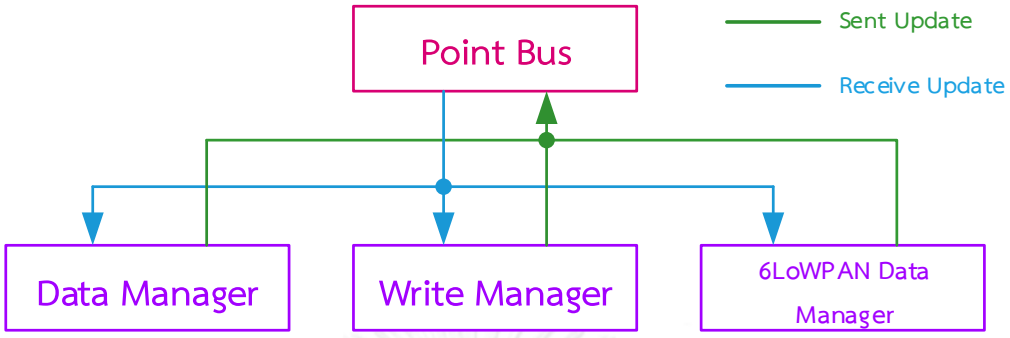
4.3.1.3. การทำงานของโปรมแกรมย่อย Stream Query Processor



รูปที่ 4-9 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมย่อย Stream Query Processor

โปรแกรมย่อย Stream Query Processor ดังแสดงในรูปที่ 4-9 จะทำการประมวลผลข้อมูล TRAP Request ที่ได้รับมาจาก remote คอมโพเน้นท์ โดยเริ่มจากการเช็ค Query ID จากข้อมูล TRAP Request ที่ได้รับมาว่าตรงกับ Object Stream Query Processor ในโปรแกรมหรือไม่ ถ้ามีจะทำการดึงเอา Object ในการสื่อสารก่อนหน้ามาใช้งาน แต่ถ้าไม่มีจะทำการสร้างใหม่ โดย Object ตัวนี้จะเก็บข้อมูลทั้งหมดของ TRAP Request ไว้ตรวจสอบเมื่อเกิดกรณีที่ข้อมูล หรือ Time Stamp มีการเปลี่ยนแปลงแล้วตรงกับเหตุการณ์ที่กำหนดไว้ใน TRAP โพรโทคอล จากนั้น Object ตัวนี้จะถูกเก็บไว้ในตัวจัดการข้อมูล (Data Manager Thread) จากนั้นจะตรวจสอบ TTL ภายในข้อมูล TRAP Request ถ้ามีค่ามากกว่า 0 จะทำการกำหนดเวลา TTL นี้ลงไปใน Object แต่ถ้าไม่จะทำการลบทำการลบ Object ออกจากตัวจัดการข้อมูล ผลที่ได้จะนำมาสร้างข้อมูลตอบกลับว่ากำหนด TRAP ได้สำเร็จหรือไม่

4.3.1.4. การทำงานของโปรมแกรมย่อย Point Bus

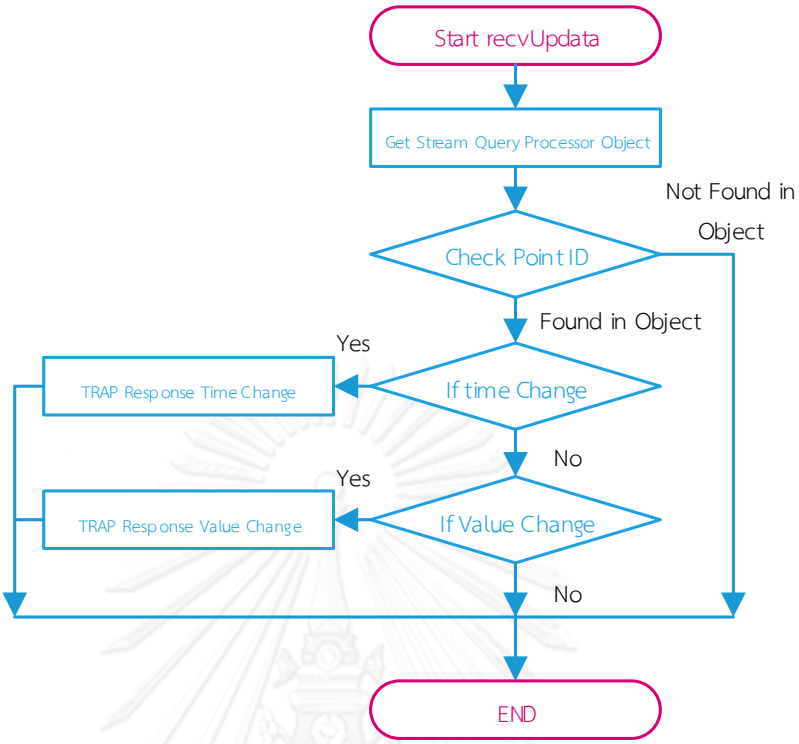


รูปที่ 4-10 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมย่อย Stream Query Processor

โปรแกรมย่อย Point Bus ดังแสดงในรูปที่ 4-10 จะเห็นได้ว่า Point Bus เป็นตัวเชื่อมโยงการทำงานระหว่าง Class Data Manager, Class Write Manager, และ Class 6LoWPAN Data Manager โดย Class เหล่านี้จะสืบทอด Class Point Bus Observer ที่ภายในประกอบไปด้วย 2 Method คือ sentUpdata และ receiveUpdata ซึ่งถ้าหาก Class ใดๆทำการเรียกใช้งาน sentUpdata เพื่อทำการกำหนดข้อมูลใหม่ลงไปใน Point Bus จะทำให้ Class อื่นๆที่เหลือได้รับข้อมูลใหม่นี้จาก Method receiveUpdata ซึ่งข้อมูลที่ถูก Update ผ่านทาง Point Bus นี้จะเป็นข้อมูล value ทั้งหมดของ Point ID ภายในเกทเวยทั้งหมด ดังนั้นโปรแกรมย่อย Point Bus จึงทำให้ Class ภายในทั้ง 3 ได้รับการ Update ข้อมูลที่เหมือนกันอยู่เสมอ

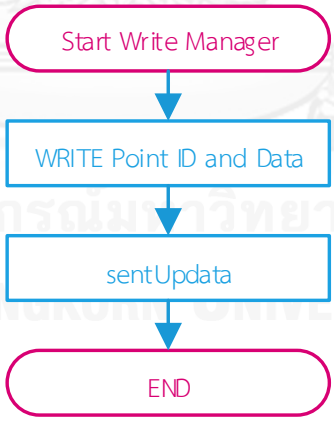
4.3.1.5. การประสานงานระหว่าง Data Manager กับ Point Bus

การประสานงานระหว่าง Data Manager กับ Point Bus ดังแสดงรูปที่ 4-11 โดยภายใน Class Data Manager จะใช้งานเฉพาะ Method recvUpdata ที่สืบทอดมาจาก Class Point Bus Observer เมื่อมีการ Update ข้อมูลใหม่จะนำเอา Stream Query Processor Object ทั้งหมดมาทำการตรวจสอบ ในขั้นตอนแรกนั้นจะทำการตรวจสอบว่ามีการกำหนด TRAP ใน Point ID ที่มีการ Update ข้อมูลมาหรือไม่ ถ้าไม่ใช่ให้สิ้นสุดการทำงาน ขั้นตอนที่สองทำการตรวจสอบว่าเป็นการกำหนด TRAP แบบเวลาเปลี่ยน หรือไม่ ถ้าเป็นจะทำการสร้างของมูล TRAP time change ตอบกลับไปยัง Callback Data ที่ตั้งไว้ ขั้นตอนที่สามทำการตรวจสอบว่าเป็นการกำหนด TRAP แบบค่าเปลี่ยน หรือไม่ ถ้าเป็นจะทำการสร้างของมูล TRAP value change ตอบกลับไปยัง Callback Data ที่ตั้งไว้ ซึ่งในการกำหนด TRAP นั้นสามารถเลือกได้เพียงรูปแบบเดียวคือ เวลาเปลี่ยน หรือ ค่าเป็น เท่านั้น



รูปที่ 4-11 โฟลว์ชาร์ตแสดงการประสานงานระหว่าง Data Manager กับ Point Bus

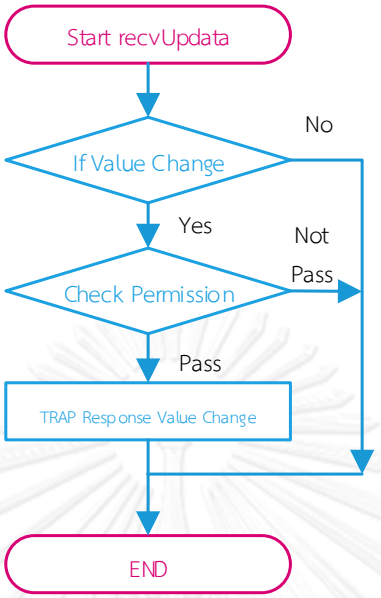
4.3.1.6. การประสานงานระหว่าง Write Manager กับ Point Bus



รูปที่ 4-12 โฟลว์ชาร์ตแสดงการประสานงานระหว่าง Write Manager กับ Point Bus

การประสานงานระหว่าง Write Manager กับ Point Bus ดังแสดงรูปที่ 4-12 โดยภายใน Class Write Manager จะใช้งานเฉพาะ Method sentUpdata ที่สืบทอดมาจาก Class Point Bus Observer ซึ่งจะได้รับข้อมูล Point ID และ Value ที่ได้มากจาก WRITE โพรโทคอล แล้วทำการ Update ข้อมูลลงไปใน Point Bus

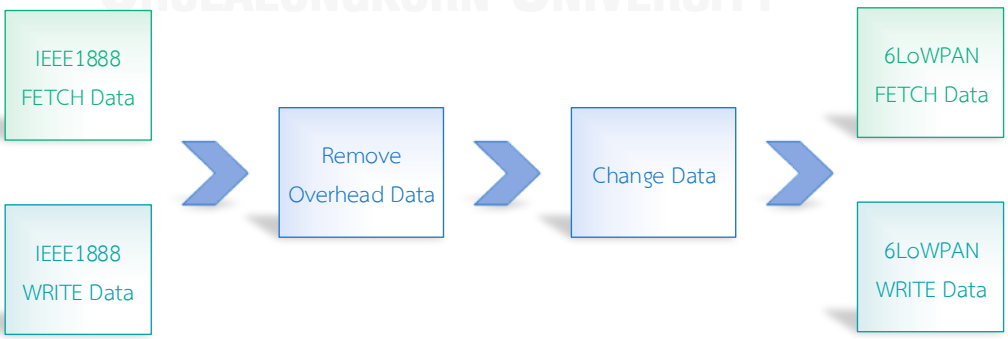
4.3.1.7. การประสานงานระหว่าง 6LoWPAN Data Manager กับ Point Bus



รูปที่ 4-13 โฟลว์ชาร์ตแสดงการประสานงานระหว่าง 6LoWPAN Data Manager กับ Point Bus

การประสานงานระหว่าง Data Manager กับ 6LoWPAN Data Manager ดังแสดงรูปที่ 4-13 โดยภายใน Class 6LoWPAN Data Manager เมื่อมีการ Update ข้อมูลใหม่จะทำการตรวจสอบ value ที่ได้รับมาใหม่กับค่าเก่าว่ามีการเปลี่ยนแปลงหรือไม่ ถ้าหากไม่มีการเปลี่ยนแปลงจะถือว่าสิ้นสุดการทำงาน แต่ถ้าหากมีการเปลี่ยนแปลงจากค่าเดิมจะทำการเช็คว่า Point ID นั้นสามารถทำการเขียนข้อมูลได้หรือไม่ ซึ่งบาง Point ID ไม่อนุญาตให้เขียนข้อมูลเนื่องจากสามารถอ่านได้อย่างเดียว เช่น อุณหภูมิ ความชื้น เป็นต้น ถ้าหาก Point ID นั้นสามารถเขียนข้อมูลได้ในขั้นตอนถัดไปจะทำการเขียนข้อมูลไปยัง 6LoWPAN โหนด

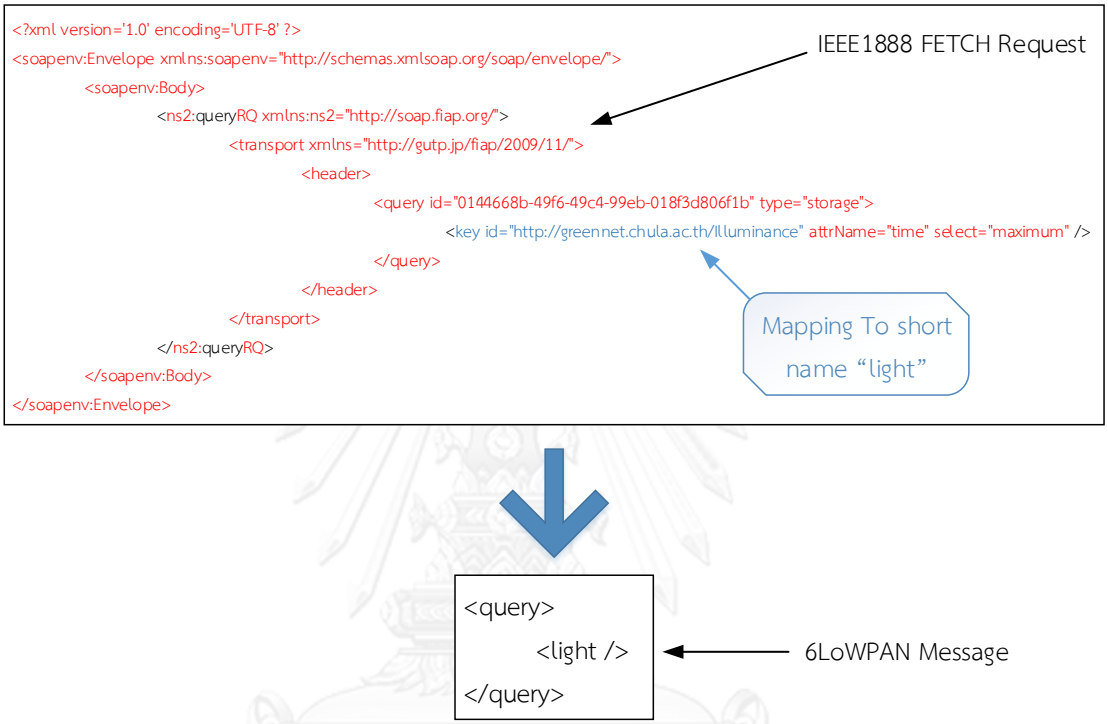
4.3.2. การออกแบบข้อมูลที่ใช้ในการสื่อสารกับ 6LoWPAN โหนด



รูปที่ 4-14 ขั้นตอนในการออกแบบข้อมูลที่ใช้ในการสื่อสารกับ 6LoWPAN โหนด

การออกแบบข้อมูลที่ใช้ในการสื่อสารกับ 6LoWPAN โหนด ได้นำเอาข้อมูลในมาตรฐาน IEEE1888 ในโพรโทคอล FETCH และ WRITE เป็นต้นแบบในการออกแบบ ซึ่งจะทำการลบข้อมูลที่ไม่จำเป็นทิ้งไป และเปลี่ยนข้อมูลบางส่วนให้มีขนาดเล็กลง เพื่อที่จะทำให้ได้ข้อมูลที่มีขนาดเล็ก เนื่องจาก 6LoWPAN Pay Load สามารถบรรจุข้อมูลได้มากสุด 128 ไบต์

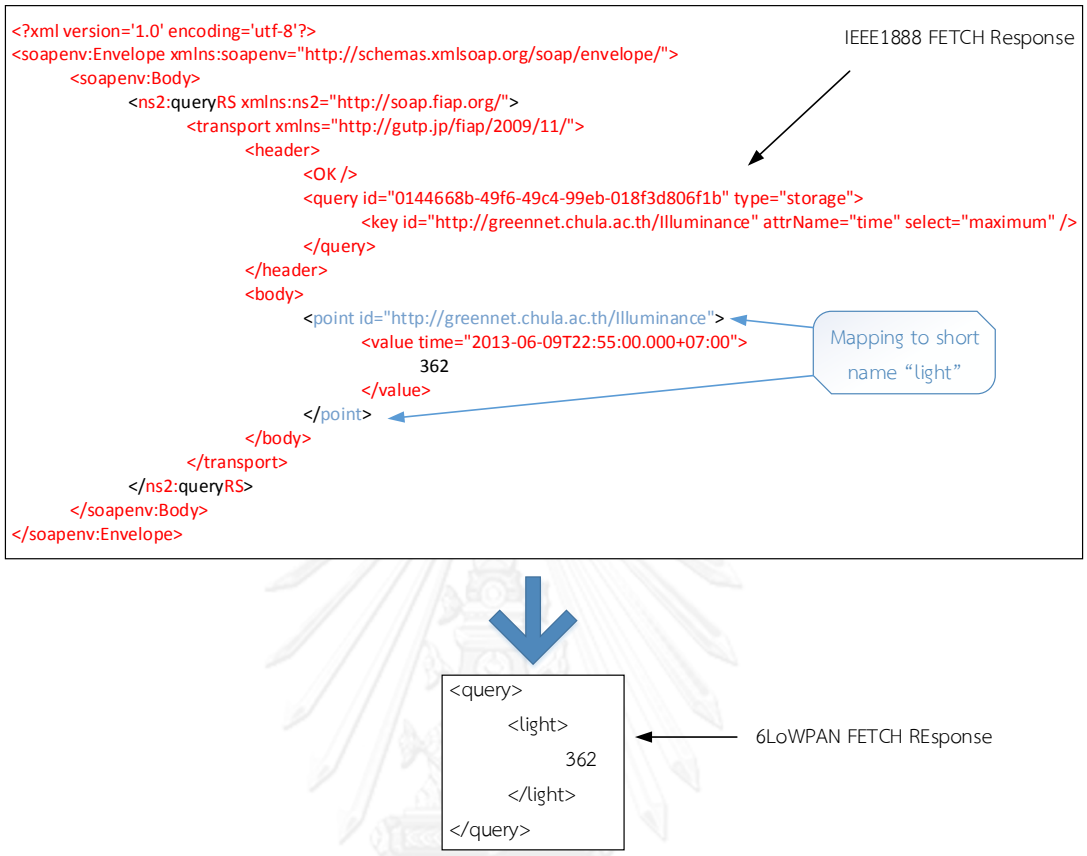
4.3.2.1. การแปลงข้อมูล FETCH Request ให้สื่อสารกับ 6LoWPAN โหนด



รูปที่ 4-15 ขั้นตอนการแปลงข้อมูล FETCH Request ให้สื่อสารกับ 6LoWPAN โหนด

การแปลงข้อมูล FETCH Request ดังแสดงในรูปที่ 4-15 จะนำข้อมูลที่เป็นโสหุ้ยในมาตรฐาน IEEE1888 (สีแดง) เก็บไว้ในส่วนประมวนผลข้อมูลของโปรแกรม จากนั้นทำการตัดข้อมูลเหล่านี้ทิ้งไปแต่ยังคงช่องทางที่ให้บริการ query ไว้ ส่วน key id (สีน้ำเงิน) จะถูกแปลงเป็นชื่อของข้อมูลภายใน 6LoWPAN โหนด และถูกใช้ในการค้นหาปลางทางในการส่งข้อมูล จากตัวอย่างข้อมูลต้นฉบับจะมีขนาด 464 ไบต์ และข้อมูลที่ใช้สื่อสารกับ 6LoWPAN โหนดมีขนาด 24 ไบต์ ซึ่งมีขนาดที่เล็กว่า 19.33 เท่า

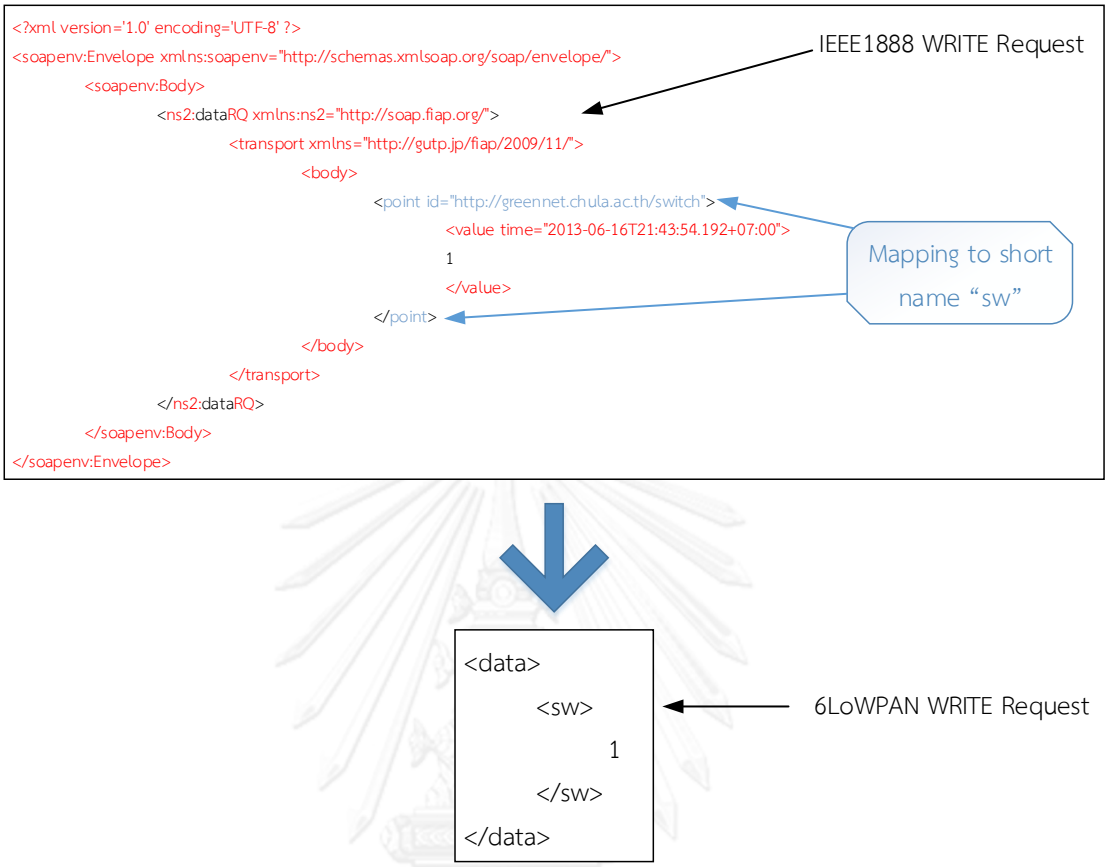
4.3.2.2. การออกแบบข้อมูลที่ใช้ในการสื่อสารกับ FETCH Response



รูปที่ 4-16 ขั้นตอนการแปลงข้อมูล FETCH Response ให้สื่อสารกับ 6LoWPAN โหนด

การแปลงข้อมูล FETCH Response ดังแสดงในรูปที่ 4-16 จะทำการตัดข้อมูลที่เป็นโสหุ้ยในมาตรฐาน IEEE1888 (สีแดง) เหล่านี้ทิ้งไปแต่ยังคงช่องทางที่ให้บริการ query ไว้ ส่วน point id (สีน้ำเงิน) จะถูกแทนที่ด้วยชื่อของข้อมูลในโหนด 6LoWPAN แบบย่อ ซึ่งความสัมพันธ์ของ point id แบบเต็ม และแบบย่อจะถูกเก็บอยู่ในไฟล์ตั้งค่า โดยค่าของข้อมูลนั้นจะถูกบรรยุอยู่ใน Element ของชื่อเซ็นเซอร์แบบสั้น จากตัวอย่างข้อมูลต้นฉบับจะมีขนาด 591 ไบต์ และข้อมูลที่ใช้สื่อสารกับ 6LoWPAN โหนดมีขนาด 33 ไบต์ ซึ่งมีขนาดที่เล็กว่า 17.90 เท่า

4.3.2.3. การออกแบบข้อมูลที่ใช้ในการสื่อสารกับ WRITE Request



รูปที่ 4-17 ขั้นตอนการแปลงข้อมูล WRITE Request ให้สื่อสารกับ 6LoWPAN โหนด

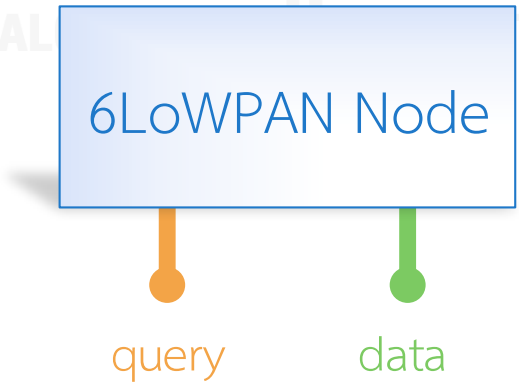
การแปลงข้อมูล WRITE Request ดังแสดงในรูปที่ 4-17 จะนำข้อมูลที่เป็นโสหุ้ยในมาตรฐาน IEEE1888 (สีแดง) เก็บไว้ในส่วนประมวนผลข้อมูลของโปรแกรม จากนั้นทำการตัดข้อมูลเหล่านี้ทิ้งไปแต่ยังคงช่องทางที่ให้บริการ data ไว้ ส่วน point id (สีน้ำเงิน) จะถูกแปลงเป็นชื่อของข้อมูลภายใน 6LoWPAN โหนด และถูกใช้ในการค้นหาปลายทางในการส่งข้อมูล ส่วน value จะเก็บอยู่ภายใน element ชื่อย่อของ Point ID ซึ่งจากตัวอย่างข้อมูลต้นฉบับจะมีขนาด 403 ไบต์ และข้อมูลที่ใช้สื่อสารกับ 6LoWPAN โหนดมีขนาด 23 ไบต์ ซึ่งมีขนาดที่เล็กว่า 17.52 เท่า

4.3.2.4. การออกแบบข้อมูลที่ใช้ในการสื่อสารกับ WRITE Response

รูปที่ 4-18 ขั้นตอนการแปลงข้อมูล WRITE Response ให้สื่อสารกับ 6LoWPAN โหนด

การแปลงข้อมูล WRITE Response ดังแสดงในรูปที่ 4-18 จะทำการตัดข้อมูลที่เป็นโสหุ้ยในมาตรฐาน IEEE1888 (สีแดง) เหล่านี้ทิ้งไปแต่ยังคงช่องทางที่ให้บริการ data ไว้ และข้อมูลตอบรับในการ WRITE ว่าสำเร็จหรือไม่ ซึ่งจากตัวอย่างข้อมูลต้นฉบับจะมีขนาด 304 ไบต์ และข้อมูลที่ใช้สื่อสารกับ 6LoWPAN โหนดมีขนาด 18 ไบต์ ซึ่งมีขนาดที่เล็กว่า 16.89 เท่า

4.4. การออกแบบซอฟต์แวร์ 6LoWPAN โนท

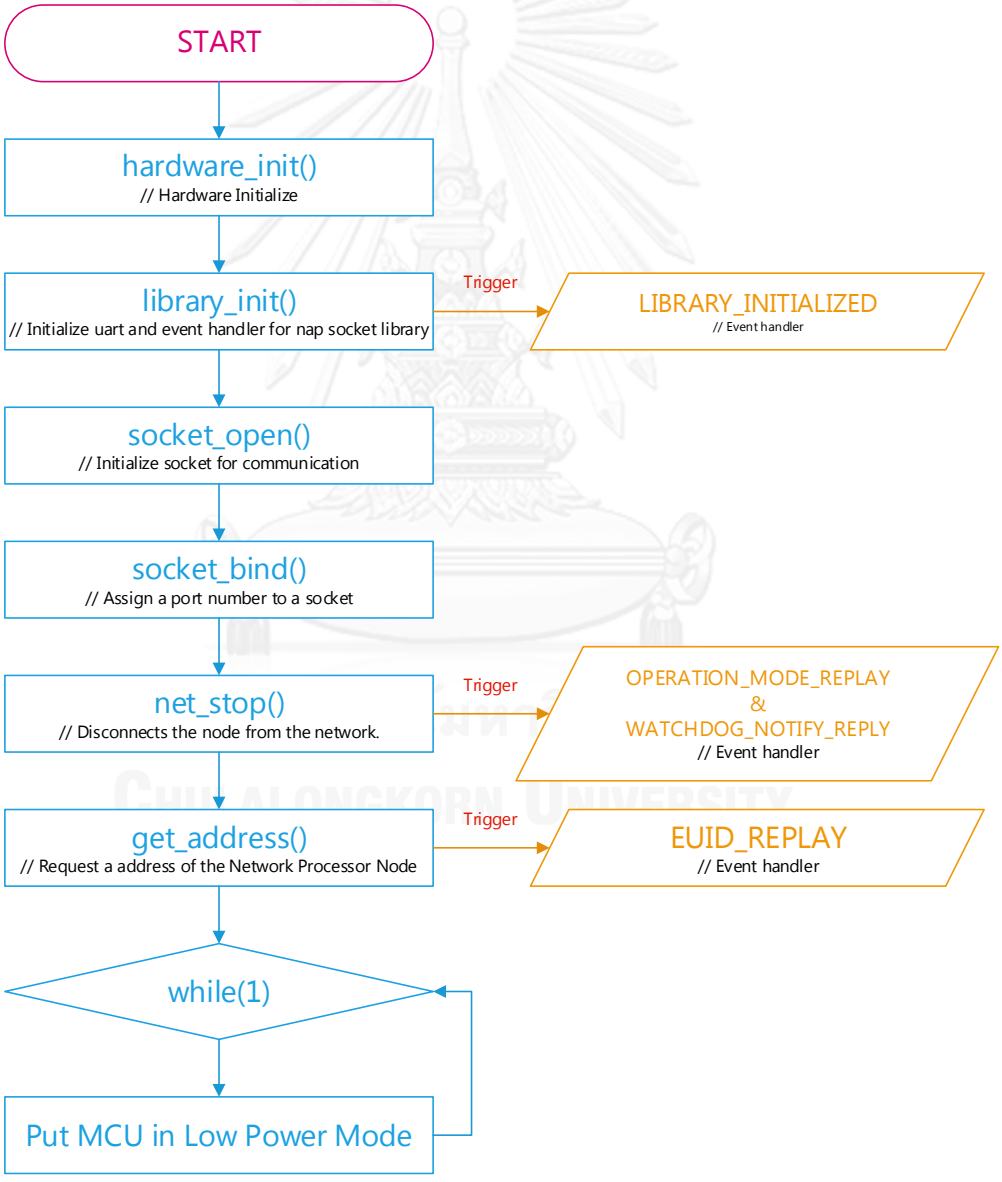


รูปที่ 4-19 Interface ของ 6LoWPAN โหนด

ซอฟต์แวร์สื่อสารของ 6LoWPAN โหนดได้ออกแบบให้มี interface ในการสื่อสารเหมือนกับมาตรฐาน IEEE1888 ประกอยไปด้วย 2 ช่องทาง คือ 1. query สำหรับการร้องขอข้อมูล 2. Data สำหรับการเขียนข้อมูล โดยมีรูปแบบข้อมูลตามหัวข้อที่ 4.3.2

ซอฟต์แวร์ 6LoWPAN Stack ที่นำมาพัฒนามีชื่อว่า NAP Socket Library ของบริษัท SENSINODE [20] มีลักษณะการทำงานเป็น Event Driven ซึ่งได้มีการกำหนดเหตุการณ์ที่เกิดขึ้นแบบตายตัว ทำให้ง่ายต่อการพัฒนาซอฟต์แวร์ และใช้งานทรัพยากรของไมโครคอนโทรเลอร์น้อย

4.4.1. ซอฟต์แวร์กำหนดค่าเริ่มต้น



รูปที่ 4-20 โฟลว์ชาร์ตกำหนดค่าเริ่มต้นของซอฟต์แวร์ 6LoWPAN

ก่อนที่ระบบจะสามารถทำงานได้นั้นจำเป็นต้องมีการกำหนดค่าเริ่มต้นต่างๆดังต่อไปนี้

- **haware_init(): เป็นฟังก์ชั่นที่ทำการกำหนดค่า Hardware ของไมโครคอรโทรเลอร์ เช่น กำหนด GPIO ให้เป็น Input หรือ Output, เปิดการทำงานของโมดุลภายใน เช่น SPI UART, และทำการกำหนดความถี่ออสซิเลเตอร์ที่ไมโครคอนโทรเลอร์ทำงาน**

- **library_init(): ทำการเชื่อมต่อการสื่อสารแบบ uart ของไมโครคอนโทรเลอร์ เข้ากับการทำงานของ CC1180 และ เริ่มต้นการทำงานของ Nap Socket Library ที่เป็นตัวจัดการ Event ของ 6LoWPAN ทั้งหมด โดยภายหลังจากการทำงานของฟังก์ชั่นนี้จะส่งผลให้ Nap Socket Library กระตุ้นการทำงานของเหตุการณ์ LIBRARY_INITIALIZED**

- **socket_open(): ทำการสร้าง socket ที่ใช้ในการเก็บค่าพารามิเตอร์ต่างๆ ที่ใช้ในการสื่อสาร**

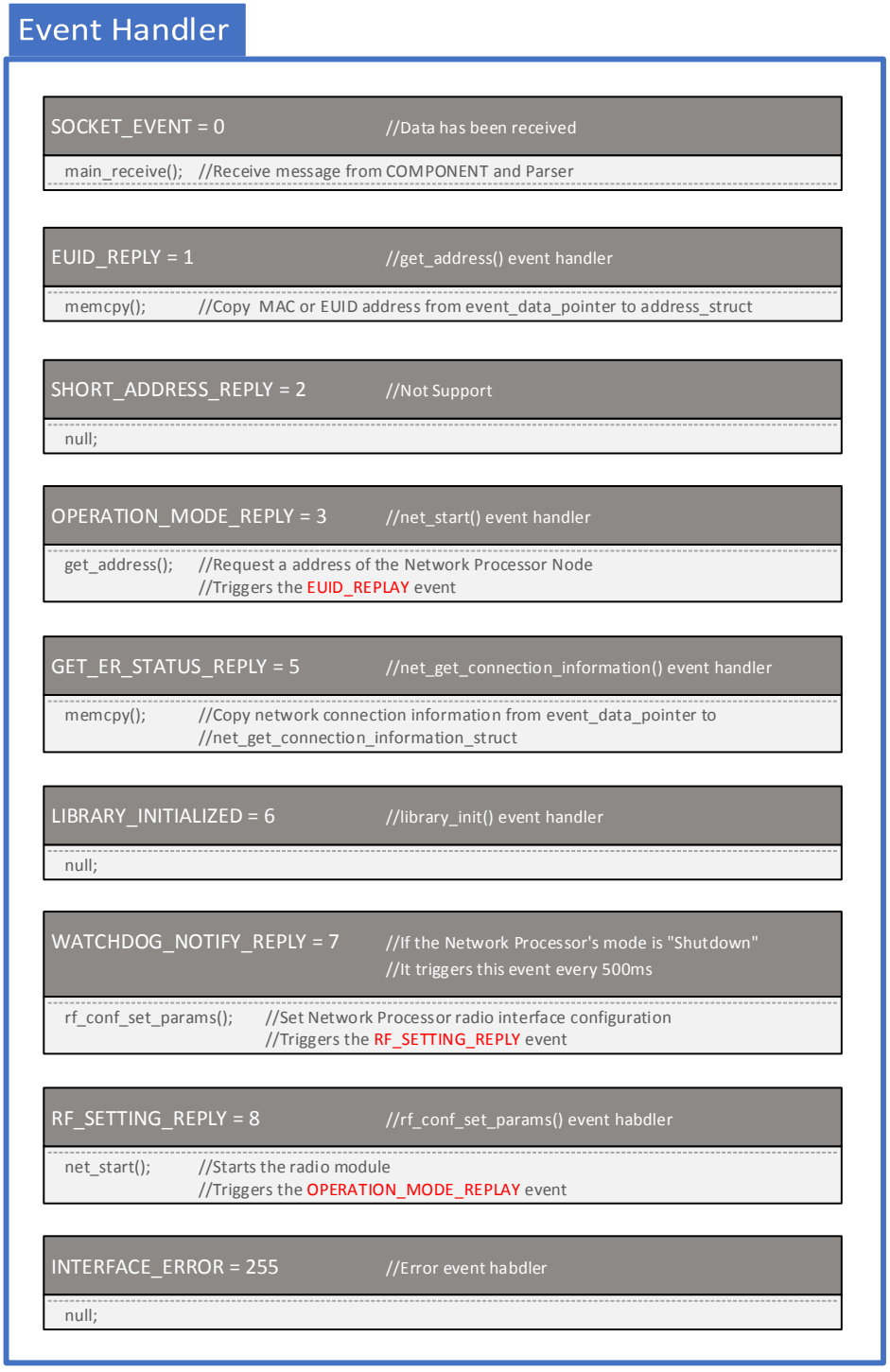
- **socket_bind(): ทำการเชื่อโยง socket ที่ได้ทำการสร้างเข้ากับ Port ที่จะใช้ในการสื่อสาร ซึ่งได้เลือกใช้ Port จำนวน 2 Port คือ 1. Port 12345 ใช้ในการสื่อสารข้อมูลกับเกทเวยผ่าน Interface query และ data 2. Port 8000 ใช้ในการลงทะเบียนหมายเลข IPv6 และ EUI-64**

- **net_stop(): ทำการรีเซ็ตการทำงานของ network processor และ ล้างข้อมูลตาราง routing ทั้งหมด โดยภายหลังจากการทำงานของฟังก์ชั่นนี้จะส่งผลให้ Nap Socket Library กระตุ้นการทำงานของเหตุการณ์ OPERATION_MODE_REPLY 1 ครั้ง และเหตุการณ์ WATCHDOG_NOTIFY_REPLY ทุกๆ 0.5 วินาที**

- **get_address(): จะใช้ในการหาหมายเลข IPv6 ของโนทนั้นๆจากเกท โดยภายหลังจากการทำงานของฟังก์ชั่นนี้จะส่งผลให้ Nap Socket Library กระตุ้นการทำงานของเหตุการณ์ EUDI_REPLAY**

ภายหลังจากการตั้งค่าต่างๆของระบบเรียบร้อยแล้วไมโครคอนโทรเลอร์ จะทำงานอยู่ให้โหมดประหยัดพลังงานไฟฟ้า และคอยรับ EVENT จาก Nap Socket Library

4.4.2. การตอบสนองต่อ Nap Socket Library Event



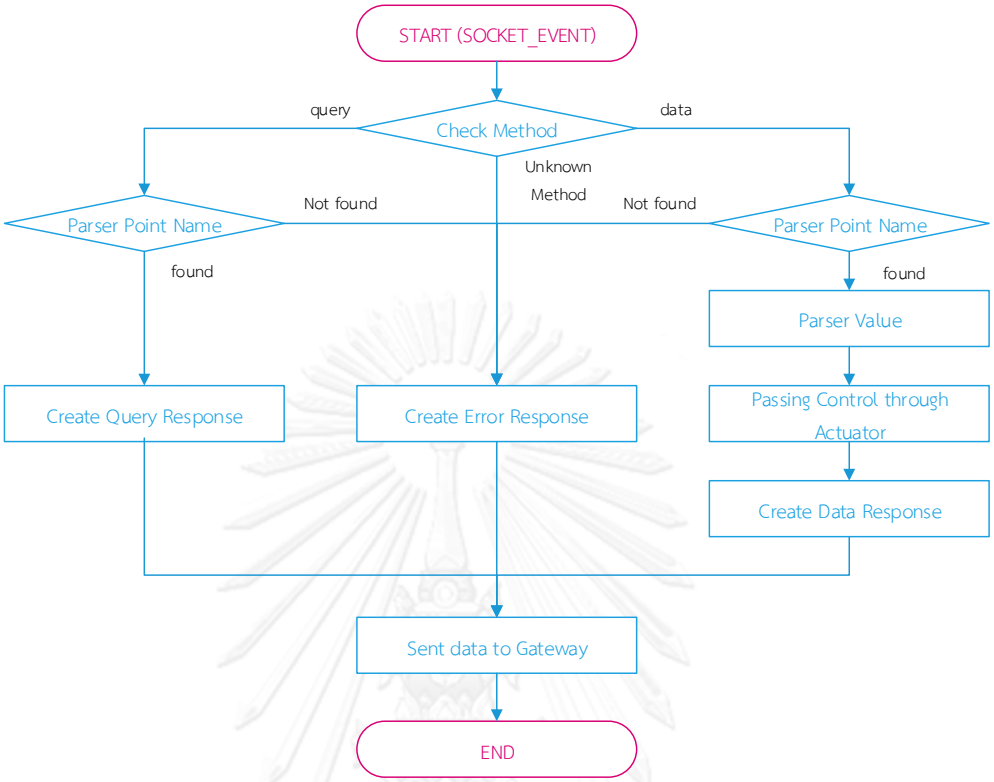
รูปที่ 4-21 โฟลว์ชาร์ตซอฟต์แวร์ตอบสนองต่อ Nap Socket Library Event

การตอบสนองต่อ Event ที่ Nap Socket Library ได้เกิดขึ้นจากรูปที่ 4-21 มีการตอบสนองของแต่ละ Event ดังแสดงในตารางที่ 4-1

ตารางที่ 4-1 การตอบสนองต่อเหตุการณ์ของ Nap Socket Library

ชื่อของ Event	การกระตุ้น Event	การตอบสนองต่อ Event
SOCKET_EVENT	ได้รับข้อมูลจากโนทอื่นๆ	นำข้อมูลที่ได้รับไปประมวลผลในฟังก์ชั่น main_receiver()
EUID_REPLY	เรียกใช้ฟังก์ชั่น get_address() เพื่อขอหมายเลข IPv6 ของโนท	เก็บหมายเลข IPv6 ลงในตัวแปลชนิด address_struct
SHORT_ADDRESS_REPLY	Nap Socket Library ไม่รองรับ	-
OPERATION_MODE_REPLY	เรียกใช้ฟังก์ชั่น net_start() เพื่อเชื่อมต่อเข้ากับระบบเครือข่าย	เรียกใช้ฟังก์ชั่น get_address()
GET_ER_STATUS_REPLY	เรียกใช้ฟังก์ชั่น net_get_connection_information() เพื่อขอข้อมูลของการเชื่อมต่อเครือข่าย เช่น ความแรงของสัญญาณ, จำนวน Hop, และสถาณะ เป็นต้น	เก็บข้อมูลลงในตัวแปลชนิด net_connection_information
LIBRARY_INITIALIZED	เรียกใช้ฟังก์ชั่น library_init() เพื่อกำหนดค่าเริ่มต้นของ Nap Socket Library	-
WATCHDOG_NOTIFY_REPLY	เมื่อ network processor ไม่เชื่อมต่อเข้ากับระบบเครือข่าย และอยู่ในโหมด Shutdown จะทำให้กระตุ้นเหตุการณ์นี้ทุกๆ 0.5 วินาที	เรียกใช้ฟังก์ชั่น rf_conf_set_params() เพื่อกำหนดกำหนดค่าพารามิเตอร์ของคลื่นความถี่วิทยุ
RF_SETTING_REPLY	เรียกใช้ฟังก์ชั่น rf_conf_set_params() เพื่อกำหนดกำหนดค่าพารามิเตอร์ของคลื่นความถี่วิทยุ	เรียกใช้ฟังก์ชั่น net_start() เพื่อเชื่อมต่อเข้ากับระบบเครือข่าย
INTERFACE_ERROR	เมื่อเกิดข้อผิดพลาดขึ้นในระบบ	-

4.4.3. การทำงานของโปรแกรมย่อย main_receiver



รูปที่ 4-22 โฟลว์ชาร์ตซอฟต์แวร์ประมวลผลข้อมูล 6LoWPAN โนท

โปรแกรมย่อย main_receiver ดังแสดงในรูปที่ 4-22 มีการทำงานเริ่มต้นจากการรับข้อมูลมาแยกวิเคราะห์ช่องทางการให้บริการ ที่ประกอบไปด้วย 2 ช่องทาง คือ data และ query แล้วส่งไปประมวลผลในแต่ละส่วน แต่ถ้าหากข้อมูลที่ได้รับมามีการใช้งานช่องทางการให้บริการที่ไม่ถูกต้องจะทำการสร้างข้อมูลที่แสดงความผิดพลาดแล้วส่งข้อมูลนั้นกลับไปยังเกทเวย

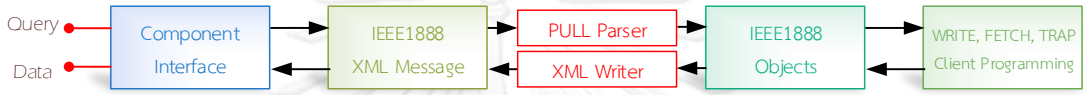
การประมวลผลข้อมูล query ในขั้นตอนแรกจะทำการตรวจสอบชื่อของ Point ID แบบสั้นว่ามีชื่อนั้นปรากฏอยู่ในโนทหรือไม่ ถ้าหากไม่มีจะทำการสร้างข้อมูลที่แสดงความผิดพลาดแล้วส่งข้อมูลนั้นกลับไปยังเกทเวย แต่ถ้าหาก Point ID ตรงกับข้อมูลภายในโนท ขั้นตอนถัดไปจะทำการสร้างข้อมูลตอบกลับที่บรรจุ Value ของ Point ID แบบสั้นที่ร้องขอมา แล้วส่งข้อมูลกลับไปยังเกทเวย

การประมวลผลข้อมูล data ในขั้นตอนแรกจะทำการตรวจสอบชื่อของ Point ID แบบสั้นว่ามีชื่อนั้นปรากฏอยู่ในโนทหรือไม่ ถ้าหากไม่มีจะทำการสร้างข้อมูลที่แสดงความผิดพลาดแล้วส่งข้อมูลนั้นกลับไปยังเกทเวย แต่ถ้าหาก Point ID ตรงกับชื่อของ Actuator ภายในโนท ขั้นตอนถัดไปจะนำข้อมูล Value ที่ได้รับมาไปควบคุมการทำงานของ Actuator แล้วสร้างข้อมูลตอบกลับไปยังเกทเวยว่าการควบคุมที่ได้สั่งการมานั้นสามารถทำงานได้สำเร็จหรือไม่

4.5. การออกแบบซอฟต์แวร์จัดการพลังงานไฟฟ้าภายในอาคาร

ซอฟต์แวร์จัดการพลังงานไฟฟ้าภายในอาคารที่ได้ทำการออกแบบขึ้นในวิทยานิพนธ์นี้ถูกพัฒนาขึ้นบนระบบปฏิบัติการ Android ที่ถูกพัฒนาขึ้นบนโปรแกรม Android Developer Tools V. 21.1.0 โดยมีแผนผังตามห้องปฏิบัติการวิจัยการออกแบบวงจรฝังตัว และวงจรรวม ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และมีวิธีการสื่อสารข้อมูลตามมาตรฐาน IEEE1888 ซึ่งสามารถแบ่งการพัฒนาออกเป็น 2 ส่วน ดังต่อไปนี้

4.5.1. การออกแบบซอฟต์แวร์สื่อสารกับมาตรฐาน IEEE1888

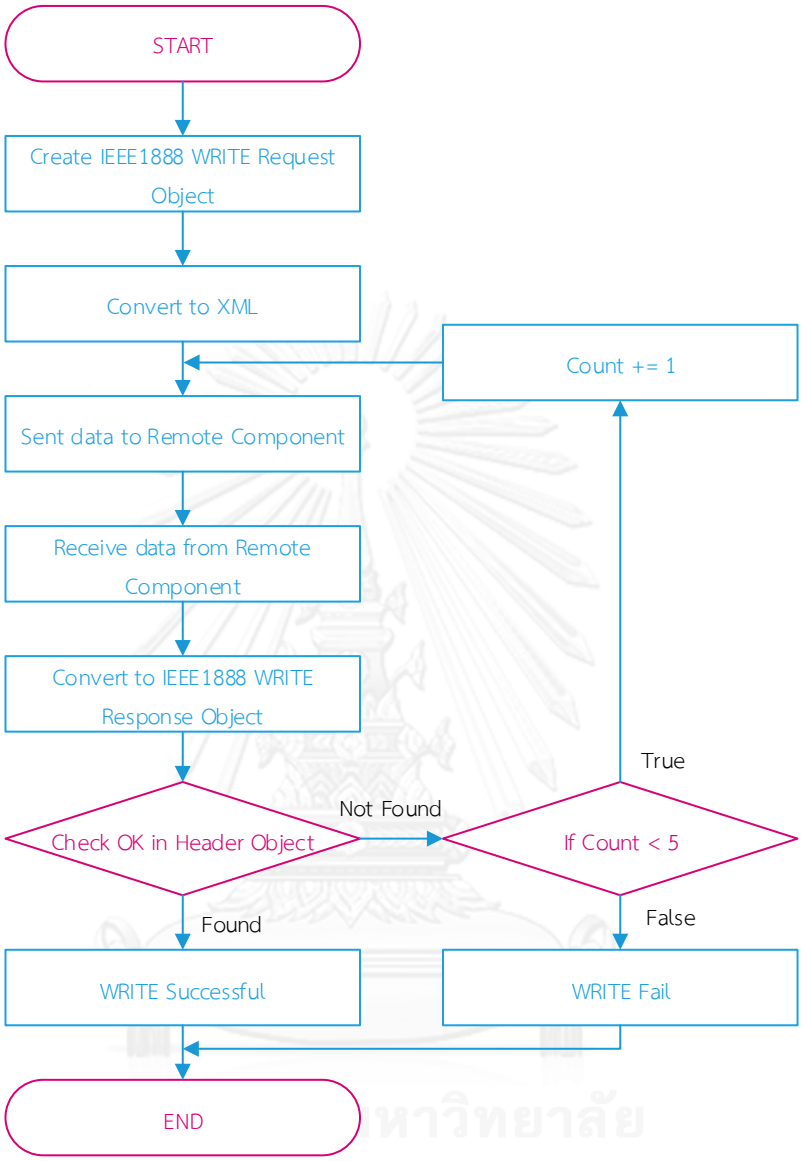


รูปที่ 4-23 กระบวนการแปลงรูปแบบข้อมูล XML ให้อยู่ในรูปแบบ Object

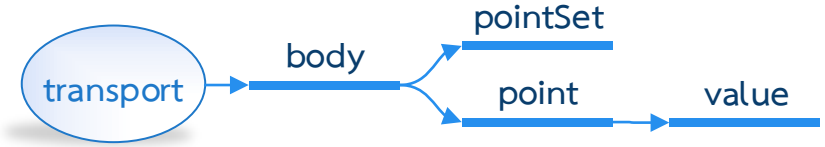
รูปที่ 4-24 ลำดับชั้นของข้อมูลในมาตรฐาน IEEE1888

ก่อนที่จะทำการเขียนโปรแกรมสื่อสารข้อมูลในมาตรฐาน IEEE1888 ได้นั้น จำเป็นต้องทำการแปลงข้อมูลในมาตรฐาน IEEE1888 ที่อยู่ในรูปแบบ XML ให้เป็น Object เสียก่อน ดังแสดงในรูปที่ 4-23 ซึ่งในแอปพลิเคชั่นที่ได้ออกแบบขึ้นมานี้ได้พัฒนาโพรโทคอลที่สื่อสารกับมาตรฐาน IEEE1888 ฝั่ง Client ที่มีการเขียนโปรแกรมในรูปแบบ IEEE1888 Object เพื่อให้ง่ายต่อการจัดการข้อมูล โดย IEEE1888 Object นี้จะมีลำดับชั้นโครงสร้างเหมือนกับข้อมูล XML ในมาตรฐาน IEEE1888 ทุกประการดังแสดงในรูปที่ 4-24 ภายใน Object แต่ละตัวจะมี 2 Method ที่สำคัญต่อการแปลงมูล คือ 1. Serialize จะทำหน้าที่ในการแปลงข้อมูลภายใน Object นั้นๆ ให้อยู่ในรูปแบบ XML โดยใช้งาน Class XML Writer 2. Parser จะทำหน้าที่ในการแปลงข้อมูลจาก XML ให้อยู่ในรูปแบบของ IEEE1888 Object โดยใช้งาน Class Pull Parser

4.5.1.1. การทำงานของโปรแกรม IEEE1888 WRITE Client



รูปที่ 4-25 โฟลว์ชาร์ตโปรแกรม IEEE1888 WRITE Client



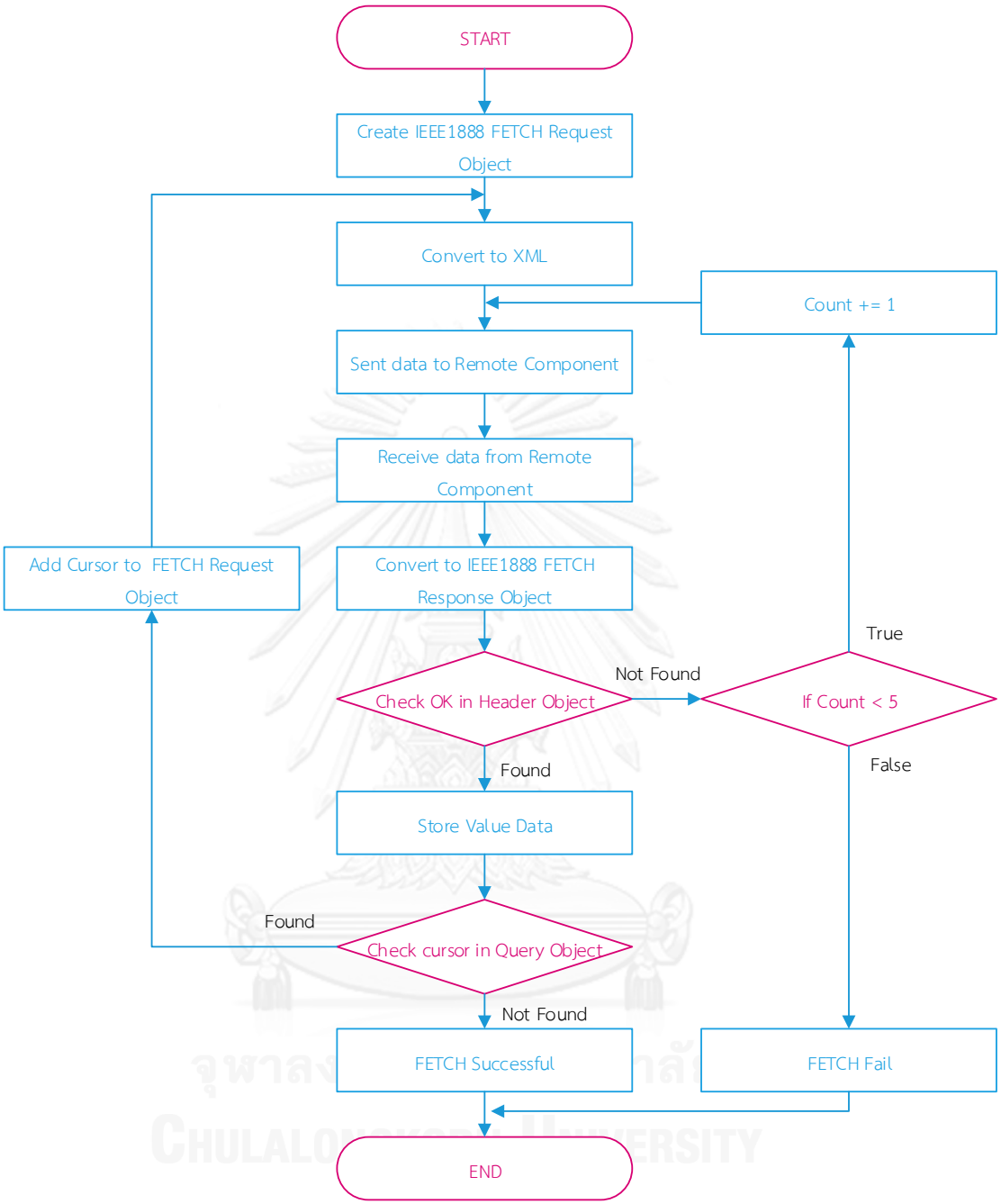
รูปที่ 4-26 ลำดับชั้นของข้อมูล WRITE Client Request Object

รูปที่ 4-27 ลำดับชั้นของข้อมูล WRITE Client Response Object

การทำงานของโปรแกรม IEEE1888 WRITE Client ดังแสดงในรูปที่ 4-25 เริ่มต้นจากการสร้างข้อมูล WRITE Client Request Object ซึ่งจะมีโครงสร้างของข้อมูลตามรูปที่ 4-26 ระบุ PointID และ Value ที่ต้องการจะเขียนข้อมูลลงไป แล้วทำการแปลงข้อมูลนี้ให้อยู่ในรูปแบบ XML ส่งไปยัง Remote คอมโพเน้นท์ จากนั้นรอรับข้อมูลตอบกลับจากรีโมทคอมโพเน้น เมื่อได้รับข้อมูลที่ครบสมบูรณ์แล้วจะทำแปลงข้อมูลนี้ให้อยู่ในรูปแบบ IEEE1888 Object ซึ่งจะมีโครงสร้างของข้อมูลตามรูปที่ 4-27 ข้อมูลนี้จะถูกนำไปตรวจสอบว่าถ้าหากภายใน Object header มี Object OK อยู่แสดงว่าการเขียนข้อมูลเสร็จสมบูรณ์ แต่ถ้าหากมี Object error อยู่แสดงว่าการเขียนข้อมูลผิดพลาดให้ และให้ส่งข้อมูลใหม่อีกครั้ง โดยได้กำหนดจำนวนครั้งในการส่งใหม่ไว้มากที่สุด 5 ครั้ง

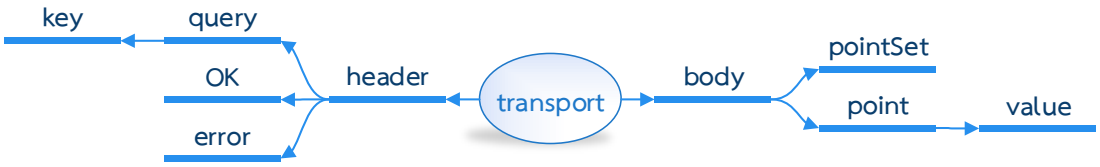
4.5.1.2. **การทำงานของโปรแกรม IEEE1888 FETCH Client**

การทำงานของโปรแกรม IEEE1888 FETCH Client ดังแสดงในรูปที่ 4-28 เริ่มต้นจากการสร้างข้อมูล FETCH Client Request Object ซึ่งจะมีโครงสร้างของข้อมูลตามรูปที่ 4-29 ระบุ key id และช่วงเวลาที่ต้องการจะอ่านข้อมูลลงไป แล้วทำการแปลงข้อมูลนี้ให้อยู่ในรูปแบบ XML ส่งไปยัง Remote คอมโพเน้นท์ จากนั้นรอรับข้อมูลตอบกลับจากรีโมทคอมโพเน้น เมื่อได้รับข้อมูลที่ครบสมบูรณ์แล้วจะทำแปลงข้อมูลนี้ให้อยู่ในรูปแบบ IEEE1888 Object ซึ่งจะมีโครงสร้างของข้อมูลตามรูปที่ 4-30 ข้อมูลนี้จะถูกนำไปตรวจสอบว่าถ้าหากภายใน Object header มี Object OK อยู่แสดงว่าการอ่านข้อมูลเสร็จสมบูรณ์ แต่ถ้าหากมี Object error อยู่แสดงว่าการอ่านข้อมูลผิดพลาดให้ และให้ส่งข้อมูลใหม่อีกครั้ง โดยได้กำหนดจำนวนครั้งในการส่งใหม่ไว้มากที่สุด 5 ครั้ง หลังจากที่ตรวจสอบความถูกต้องของข้อมูลแล้วจะนำข้อมูลใน Object value เก็บไว้ในตัวแปรของโปรแกรม แล้วทำการตรวจสอบ attribute cursor ใน object query ถ้าหากมีข้อมูลนี้อยู่แสดงว่าการสื่อสารข้อมูลยังไม่สิ้นสุดให้นำ cursor นี้เพิ่มเข้าไปใน FETCH Client Request Object ที่สร้างในขั้นตอนแรกแล้วทำการสื่อสารตามที่กล่าวข้างต้นอีกครั้ง จนกว่าจะได้รับข้อมูลที่ไม่มี cursor จึงจะถือว่าสิ้นสุดการสื่อสารข้อมูล FETCH โพรโทคอล



รูปที่ 4-28 โฟลว์ชาร์ตโปรแกรม IEEE1888 FETCH Client

รูปที่ 4-29 ลำดับชั้นของข้อมูล FETCH Client Request Object

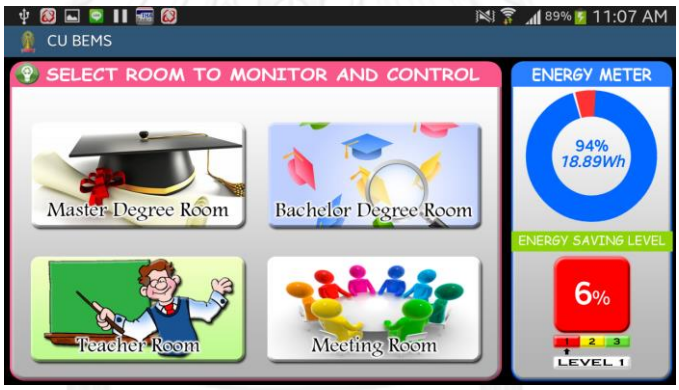


รูปที่ 4-30 ลำดับชั้นของข้อมูล FETCH Client Response Object

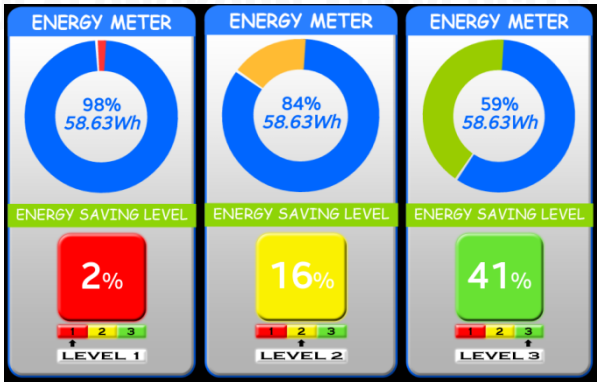
4.5.2. การออกแบบส่วนติดต่อผู้ใช้งาน

ส่วนติดต่อผู้ใช้งานที่ได้ออกแบบขึ้นมานี้รองรับกับการทำงานร่วมกับสตอเรจจำนวน 1 ตัว, เกทเวยจำนวน 1 ตัวต่อ 1 ห้อง, และ ในแต่ละห้องสามารถมีจำนวนโหนดสูงสุด 2 โนท โดยประกอบไปด้วย Activity ดังต่อไปนี้

4.5.2.1. Main Activity



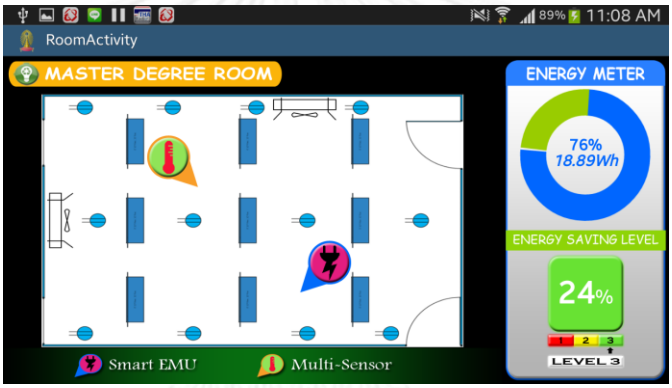
รูปที่ 4-31 ส่วนติดต่อผู้ใช้งานหน้า Main Activity



(ก.) ระดับวิกฤต (ข.) ระดับปานกลาง (ค.) ระดับปกติ
รูปที่ 4-32 มิเตอร์วัดพลังงานไฟฟ้า

จากรูปที่ 4-31 แสดงส่วนติดต่อผู้ใช้งานหน้า Main Activity โดยในด้านซ้ายผู้ใช้งานสามารถเลือกห้องที่ต้องการที่จะเรียกดูข้อมูล หรือ ทำการควบคุม ส่วนในด้านขวาจะมีมิเตอร์วัดการใช้พลังงานไฟฟ้าทั้งหมดภายในอาคาร ซึ่งประกอบไปด้วยกราฟวงกลมที่แสดงสัดส่วนการใช้พลังงานเปรียบเทียบกับพลังงานคงเหลือในแต่ละวันที่สามารถกำหนดได้ ส่วนด้านล่างจะแสดงปริมาณพลังงานไฟฟ้าคงเหลือว่าอยู่ในระดับใดแล้ว โดยประกอบไปด้วย 3 ระดับ คือ 1. ระดับวิกฤตมีปริมาณพลังงานไฟฟ้าคงเหลือน้อยกว่า 10% แถบแสดงระดับพลังงานจะปรากฏสีแดงดังแสดงในรูปที่รูปที่ 4-32 (ก.) 2. ระดับปานกลางมีปริมาณไฟฟ้าคงเหลืออยู่ในช่วง 10%-20% แถบแสดงระดับพลังงานจะปรากฏสีเหลืองดังแสดงในรูปที่รูปที่ 4-32 (ข.) 3. ระดับปกติมีปริมาณไฟฟ้าคงเหลือมากกว่า 20% แถบแสดงระดับพลังงานจะปรากฏสีเขียวดังแสดงในรูปที่รูปที่ 4-32 (ค.)

4.5.2.2. Room Activity



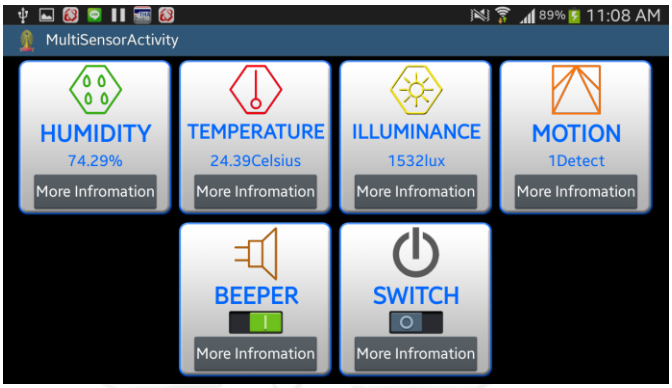
รูปที่ 4-33 ส่วนติดต่อผู้ใช้งานหน้า Room Activity

จากรูปที่ 4-33 แสดงส่วนติดต่อผู้ใช้งานหน้า Room Activity ซึ่งจะปรากฏแผนผังของห้องที่เราทำการเลือกใน Main Activity บนแผนผังจะมีลูกศรชี้ตำแหน่งที่ทำการติดตั้งโนทวัดพลังงานไฟฟ้า และโนทวัดสภาพแวดล้อม โดยเมื่อสัมผัสที่ตัวลูกศรจะทำให้เชื่อมโยงการทำงานไปยัง Energy Measurement Unit Activity และ Sensor and Actuator Summary Activity การกำหนดตำแหน่งสามารถทำได้โดยกดปุ่ม Setting และเลือก Set Node Position ในทางด้านขวาจะมีมิเตอร์วัดพลังงานไฟฟ้าที่มีการทำงานเหมือนกับ Main Activity แต่วัดพลังงานเฉพาะในห้องที่ทำการเลือกเท่านั้น

4.5.2.3. Multi-Sensor Activity

จากรูปที่ 4-34 แสดงส่วนติดต่อผู้ใช้งานหน้า Multi-Sensor Activity ซึ่งจะแสดงข้อมูลล่าสุดภายในโนทวัดสภาพแวดล้อมทั้งหมด อีกทั้งยังสามารถควบคุมการทำงานของ Buzzer ได้โดยกดปุ่มที่มีรูปสวิตซ์ใต้ชื่อ BEEPER นอกจากนั้นแล้วถ้าหากผู้ใช้งานต้องการลายละเอียดข้อมูล

เพิ่มเติมผู้ใช้งานสามารถที่จะกดปุ่ม More Information เพื่อทำการเชื่อโยงการทำงานไปยัง Sensor and Actuator Summary Activity



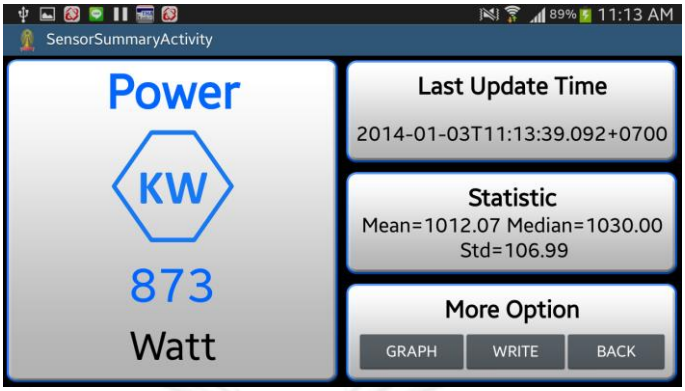
รูปที่ 4-34 ส่วนติดต่อผู้ใช้งานหน้า Multi-Sensor Activity

4.5.2.4. **Energy Measurement Unit Activity**

รูปที่ 4-35 ส่วนติดต่อผู้ใช้งานหน้า Energy Measurement Unit Activity

จากรูปที่ 4-34 แสดงส่วนติดต่อผู้ใช้งานหน้า Energy Measurement Unit Activity ซึ่งจะแสดงข้อมูลล่าสุดภายในโนทวัดพลังงานไฟฟ้า อีกทั้งยังสามารถควบคุมการทำงานของ Relay ตัดต่อไฟฟ้าได้โดยกดปุ่มที่มีรูปสวิตซ์ใต้ชื่อ SWITCH นอกจากนั้นแล้วถ้าหากผู้ใช้งานต้องการลายละเอียดข้อมูลเพิ่มเติมผู้ใช้งานสามารถที่จะกดปุ่ม More Information เพื่อทำการเชื่อโยงการทำงานไปยัง Sensor and Actuator Summary Activity

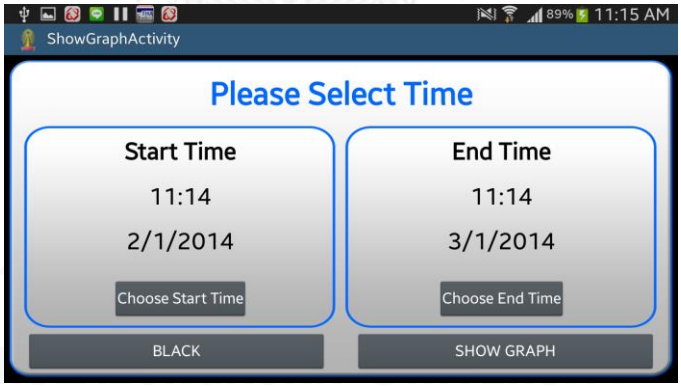
4.5.2.5. Sensor and Actuator Summary Activity



รูปที่ 4-36 ส่วนติดต่อผู้ใช้งานหน้า Sensor and Actuator Summary Activity

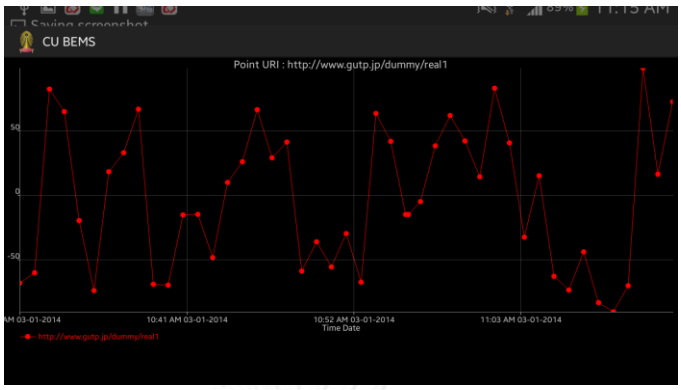
จากรูปที่ 4-36 แสดงส่วนติดต่อผู้ใช้งานหน้า Sensor and Actuator Summary Activity ซึ่งจะแสดงค่าล่าสุดข้อมูลที่ทำการเลือกในด้านซ้าย ส่วนทางด้านขวานั้นจะแบงออกเป็น 3 ส่วน คือ 1. เวลาที่มีการเปรียนแปลงค่าล่าสุด 2. สถิติค่อของข้อมูลนั้นในช่วงเวลา 24 ช่วงโมงที่ผ่านมา 3. การกระทำเพิ่มเติมกับข้อมูลนั้น เช่น พล็อตกราฟ เขียนข้อมูล และกลับสู่ Activity ก่อนหน้านี้

4.5.2.6. Graph Activity



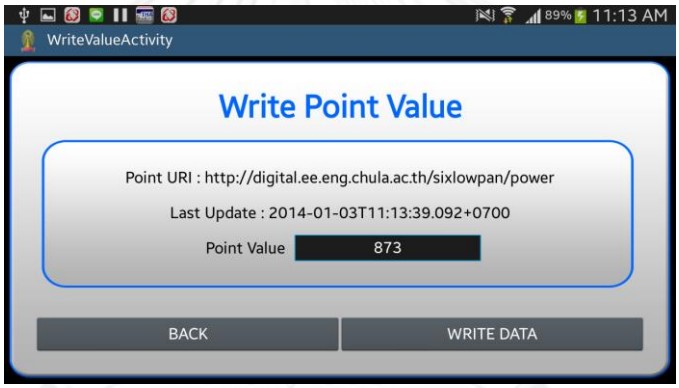
รูปที่ 4-37 ส่วนติดต่อผู้ใช้งานหน้า Graph Activity

จากรูปที่ 4-37 แสดงส่วนติดต่อผู้ใช้งานหน้า Graph Activity ผู้ใช้งานสามารถที่จะกำหนดช่วงเวลาเริ่มต้น และสิ้นสุดที่จะต้องการพล็อตข้อมูลลงบนกราฟ โดยมีระยะเวลาสูงสุดไม่เกิน 1 สัปดาห์จากนั้นกดปุ่ม SHOW GRAPH เพื่อทำการพล็อตกราฟดังแสดงในรูปที่ 4-38



รูปที่ 4-38 กราฟที่ได้ทำการพล็อต

4.5.2.7. **Write Value Activity**



รูปที่ 4-39 ส่วนติดต่อผู้ใช้งานหน้า Write Value Activity

จากรูปที่ 4-39 แสดงส่วนติดต่อผู้ใช้งานหน้า Write Value Activity ผู้ใช้งานสามรถทำการเขียนข้อมูลไปยัง Point ID ได้โดยการใส่ค่าที่ต้องการจะเขียนลงไปในช่อง Point Value จากนั้นกดปุ่ม WRITE DATA ข้อมูลจะถูดเขียนไปยังสตอเรจ แล้วมีหน้า Pop up เด้งขึ้นมาว่าการเขียนข้อมูลสำเร็จหรือไม่

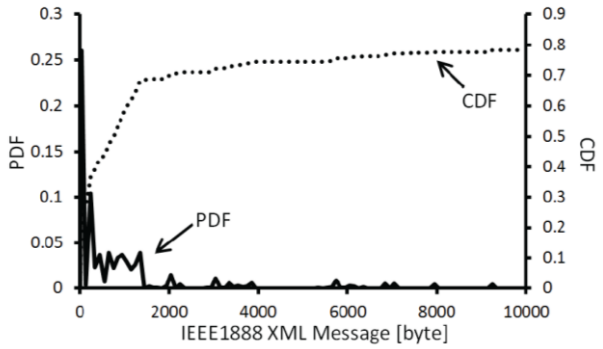
บทที่ 5

ผลการทดลอง

ในบทนี้ได้กล่าวถึง ผลการทดลองบีบอัดข้อมูล และแอพพลิเคชั่น ซึ่งจะได้ทำการเปรียบเทียบผลการบีบอัดข้อมูลในกระบวนการ EXI กับกระบวนการอื่นๆ ผลการบีบอัดข้อมูลที่ได้มาจาก Proxy ผลการทดสอบการใช้งานแอพพลิเคชั่นในการแสดงผลข้อมูล และควบคุมการทำงาน

5.1. การทดลองเปรียบเทียบกระบวนการบีบอัดข้อมูล

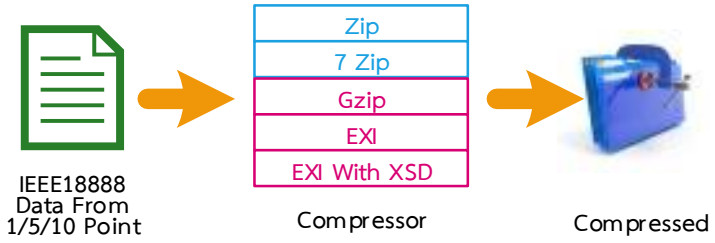
5.1.1. การเลือกช่วงขนาดข้อมูล



รูปที่ 5-1 การแจกแจงความน่าจะเป็นของขนาดข้อมูลในมาตรฐาน IEEE1888

ข้อมูลจากกราฟในรูปที่ 5-1 ได้มาจากการเก็บขนาดข้อมูลในมาตรฐาน IEEE1888 ที่ใช้สื่อสารจริง ณ University of Tokyo ประเทศญี่ปุ่น [21] แสดงให้เห็นว่าขนาดข้อมูลในช่วง 0-1500 ไบต์ มีความหนาแน่นความน่าจะเป็น (PDF) สูงกว่าในช่วงข้อมูลอื่นๆ ดังนั้นในการทดลองจึงนำข้อมูลในช่วงนี้ไปทำการทดสอบ โดยช่วงข้อมูลดังกล่าวมีความสัมพันธ์กับขนาดของ Point ID ในช่วง 1-10 จุด

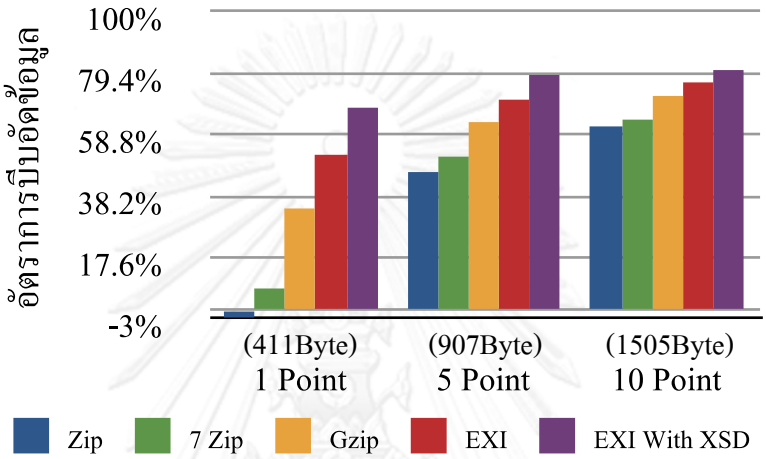
5.1.2. วิธีการทดสอบกระบวนการบีบอัดข้อมูล



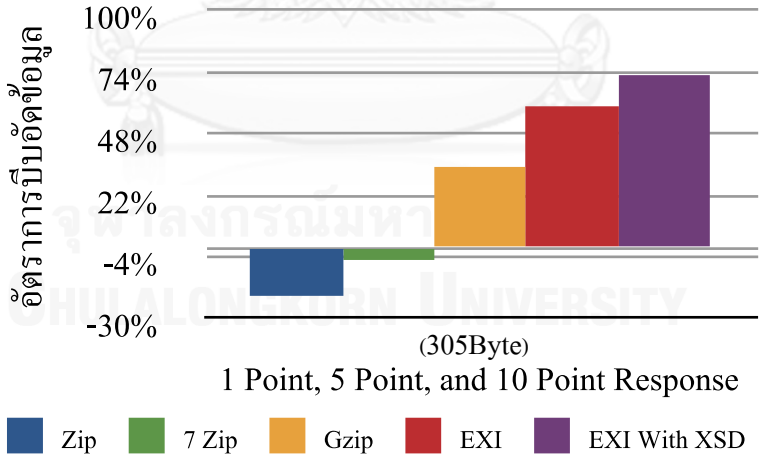
รูปที่ 5-2 การทดลองเปรียบเทียบกระบวนการบีบอัดข้อมูล

เพื่อศึกษาหากระบวนการบีบอัดข้อมูลที่เหมาะสมแก่การบีบอัดข้อมูลในมาตรฐาน IEEE1888 จึงได้ทดลองนำกระบวนการบีบอัดแบบข้อมูลแบบไม่มีการสูญเสีย 5 กระบวนการ คือ EXI, EXI ที่ทำงานร่วมกับ XSD, Gzip, Zip และ 7-Zip มาบีบอัดข้อมูล WRITE โพรโทคอล และ FETCH โพรโทคอล คอล ที่มีข้อมูลขนาด 1 จุด, 5 จุด และ 10 จุด เรียงลำดับตามขนาดข้อมูลจากเล็กไปใหญ่ ดังแสดงในรูปที่ 5-2 เพื่อหาความสามารถในการบีบอัดข้อมูลของแต่ละกระบวนการ [22]

5.1.3. การบีบอัดข้อมูล WRITE โพรโทคอล



รูปที่ 5-3 อัตราการบีบอัดข้อมูลร้องขอใน WRITE โพรโทคอล



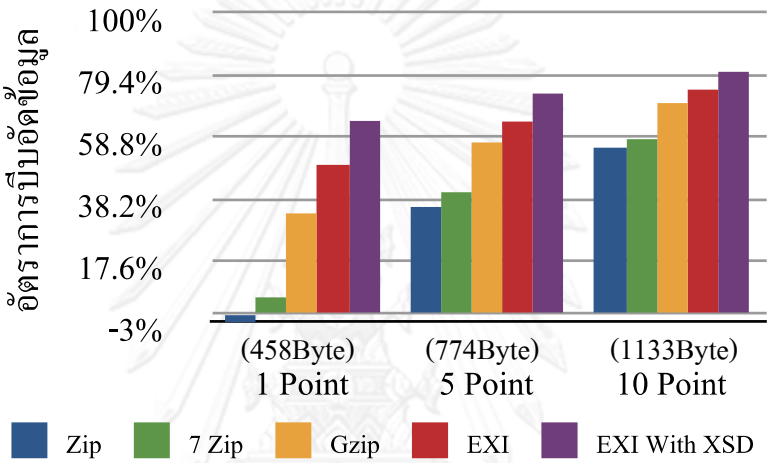
รูปที่ 5-4 อัตราการบีบอัดข้อมูลตอบรับใน WRITE โพรโทคอล

การบีบอัดข้อมูลร้องขอใน WRITE โพรโทคอล ดังแสดงในรูปที่ 5-3 พบว่าข้อมูลที่มีขนาด 1 จุด (1 Point) กระบวนการ Gzip มีประสิทธิภาพในการบีบอัดข้อมูลต่ำกว่า EXI 1.52 เท่า และ ต่ำกว่า EXI ที่ทำงานร่วมกับ XSD 1.99 เท่า แต่กระบวนการ Zip มีค่าติดลบเนื่องจากขนาด

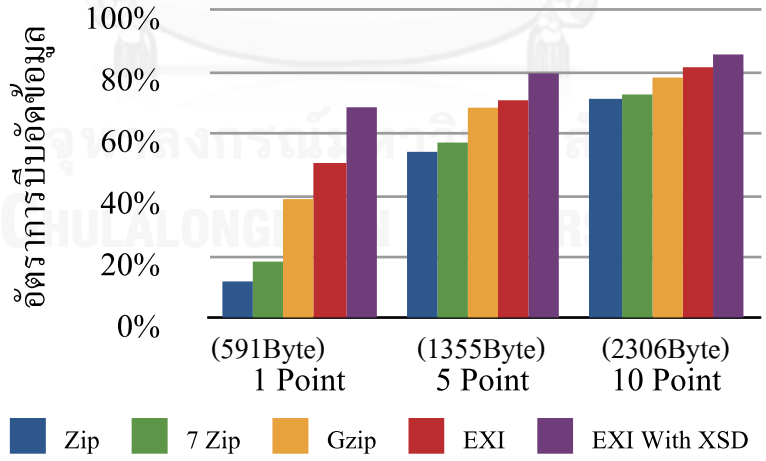
ข้อมูลใหญ่ขึ้น และ กระบวนการ 7-Zip มีประสิทธิภาพต่ำมาก สำหรับข้อมูลที่มีขนาด 5 จุด และ 10 จุด (5 Point และ10 Point) ประสิทธิภาพของกระบวนการทั้งห้าแตกต่างกันน้อยลงตามลำดับ

การบีบอัดข้อมูลตอบรับใน WRITE โพรโทคอล ดังแสดงในรูปที่ 5-4 มีข้อมูลตอบรับจากการร้องขอใน WRITE โพรโทคอลเพียงรูปแบบเดียว โดยกระบวนการ Gzip มีประสิทธิภาพในการบีบอัดข้อมูลต่ำกว่า EXI 1.75 เท่า และ ต่ำกว่า EXI ที่ทำงานร่วมกับ XSD 2.13 เท่า แต่กระบวนการ Zip และ 7-Zip มีค่าติดลบเนื่องจากขนาดข้อมูลใหญ่ขึ้น

5.1.4. การบีบอัดข้อมูล FETCH โพรโทคอล



รูปที่ 5-5 อัตราการบีบอัดข้อมูลร้องขอใน FETCH โพรโทคอล

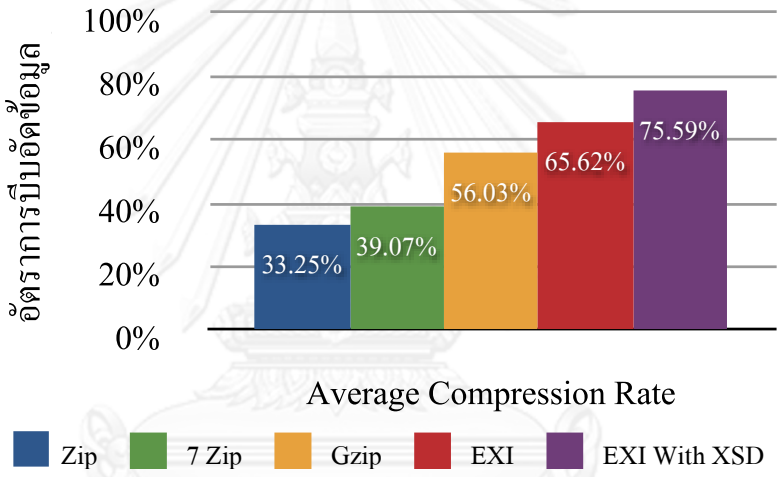


รูปที่ 5-6 อัตราการบีบอัดข้อมูลตอบรับใน FETCH โพรโทคอล

การบีบอัดข้อมูลร้องขอใน FETCH โพรโทคอล ดังแสดงในรูปที่ 5-5 พบว่าข้อมูลที่มีขนาด 1 จุดกระบวนการ Gzip มีประสิทธิภาพในการบีบอัดข้อมูลต่ำกว่า EXI 1.48 เท่า และ ต่ำกว่า EXI ที่ทำงานร่วมกับ XSD 1.92 เท่า แต่กระบวนการ Zip มีค่าติดลบเนื่องจากขนาดข้อมูลใหญ่ขึ้น และ กระบวนการ 7-Zip มีประสิทธิภาพต่ำมาก สำหรับข้อมูลที่มีขนาด 5 จุด และ 10 จุด ประสิทธิภาพของกระบวนการทั้งห้าแตกต่างกันน้อยลงตามลำดับ

การบีบอัดข้อมูลตอบรับใน FETCH โพรโทคอล ดังแสดงในรูปที่ 5-6 พบว่าข้อมูลที่มีขนาด 1 จุดกระบวนการ Gzip มีประสิทธิภาพในการบีบอัดข้อมูลต่ำกว่า EXI 1.30 เท่า และ ต่ำกว่า EXI ที่ทำงานร่วมกับ XSD 1.76 เท่า แต่กระบวนการ Zip และ 7-Zip มีประสิทธิภาพต่ำมาก สำหรับข้อมูลที่มีขนาด 5 จุด และ 10 จุดประสิทธิภาพของกระบวนการทั้งห้าแตกต่างกันน้อยลงตามลำดับ

5.1.5. ประสิทธิภาพในการบีบอัดข้อมูลโดยเฉลี่ย

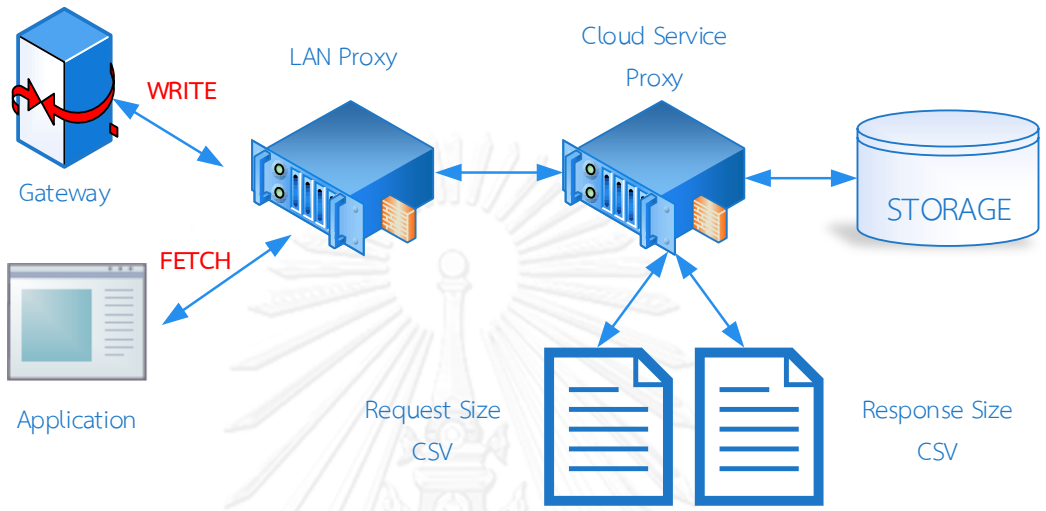


รูปที่ 5-7 ประสิทธิภาพของกระบวนการบีบอัดข้อมูลโดยเฉลี่ย

จากรูปที่ 5-7 แสดงให้เห็นประสิทธิภาพของกระบวนการบีบอัดข้อมูลเรียงลำดับจากกระบวนการที่มีประสิทธิภาพต่ำสุดไปยังกระบวนการที่มีประสิทธิภาพสูงสุด ซึ่งกระบวนการ EXI ที่ทำงานร่วมกับ XSD, EXI, และ Gzip สามารถทำการบีบอัดข้อมูลในมาตรฐาน IEEE1888 ให้มีขนาดเล็กลงได้ในทุกโพรโทคอล แต่กระบวนการ Zip และ 7-Zip นั้นมีประสิทธิภาพในการบีบอัดต่ำกว่ากระบวนการทั้งสาม และ ไม่สามารถบีบอัดข้อมูลที่มีขนาด 1 จุดให้มีขนาดเล็กลงได้มากนัก

5.2. การทดสอบ Proxy บีบอัดข้อมูล

5.2.1. วิธีการทดสอบ Proxy บีบอัดข้อมูล



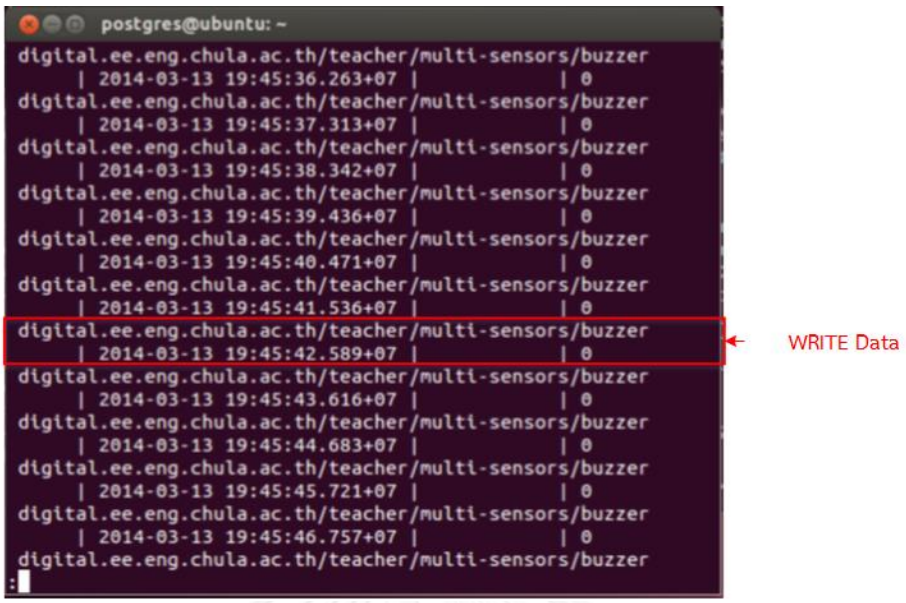
รูปที่ 5-8 การทดสอบ proxy บีบอัดข้อมูล

การทดสอบ proxy บีบอัดข้อมูลดังแสดงในรูปที่ 5-8 นั้นได้ทำการเขียนโปรแกรมภายใน proxy ทางฝั่งผู้ให้บริการ Cloud สตอเรจ ซึ่งจะทำการเก็บขนาดของข้อมูล Request ที่มีการบีบอัดเปรียบเทียบกับขนาดข้อมูลภายหลังการคลายการบีบอัดลงในไฟล์ Request.csv และ ทำการเก็บขนาดข้อมูล Response ที่ได้รับมาจากสตอเรจเปรียบเทียบกับขนาดข้อมูลภายหลังการบีบอัดลงในไฟล์ Response.csv

ข้อมูลที่นำมาทดสอบ proxy นั้นมี 2 ส่วน คือ 1. WRITE โพรโทคอล ได้มาจากเกทเวย 6LoWPAN ที่ทำการพัฒนาในงานวิจัยนี้ 2. FETCH โพรโทคอล ได้มาจากแอพพลิเคชั่นที่เขียนมาเพื่อทำการ FETCH ค่าล่าสุดไปยังสตอเรจทุกๆ 1 นาที โดยมีขนาดของข้อมูล 1จุด, 2จุด, 3จุด, 4จุด, 5จุด, และ 11จุด เรียงลำดับตามขนาดเล็กไปใหญ่

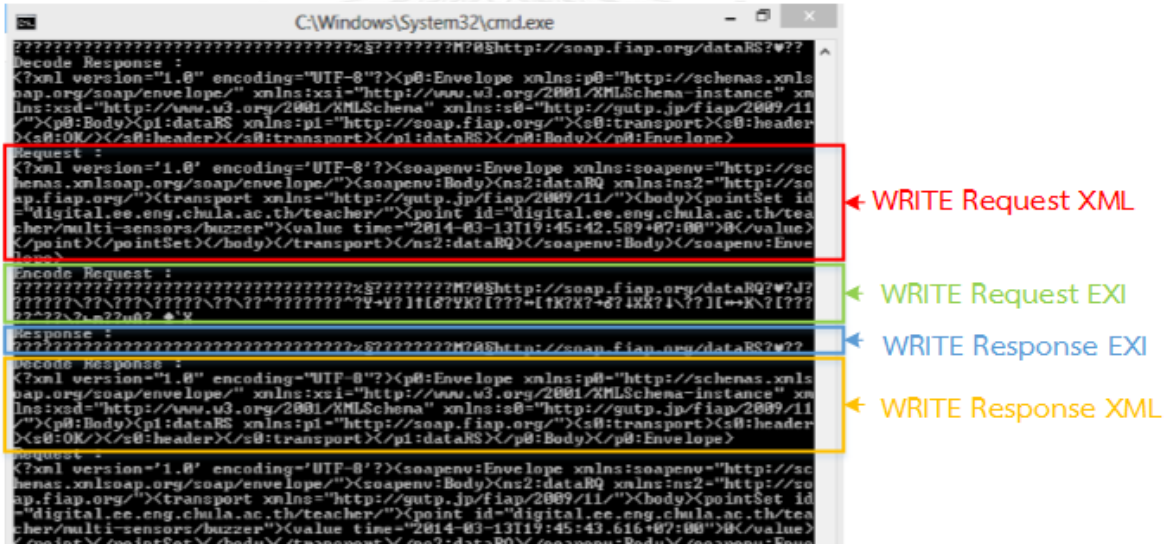
5.2.2. ผลการทดสอบ proxy ในการบีบอัดข้อมูล WRITE โพรโทคอล

5.2.2.1. การทดสอบการทำงานของ proxy กับ WRITE โพโทรคอล



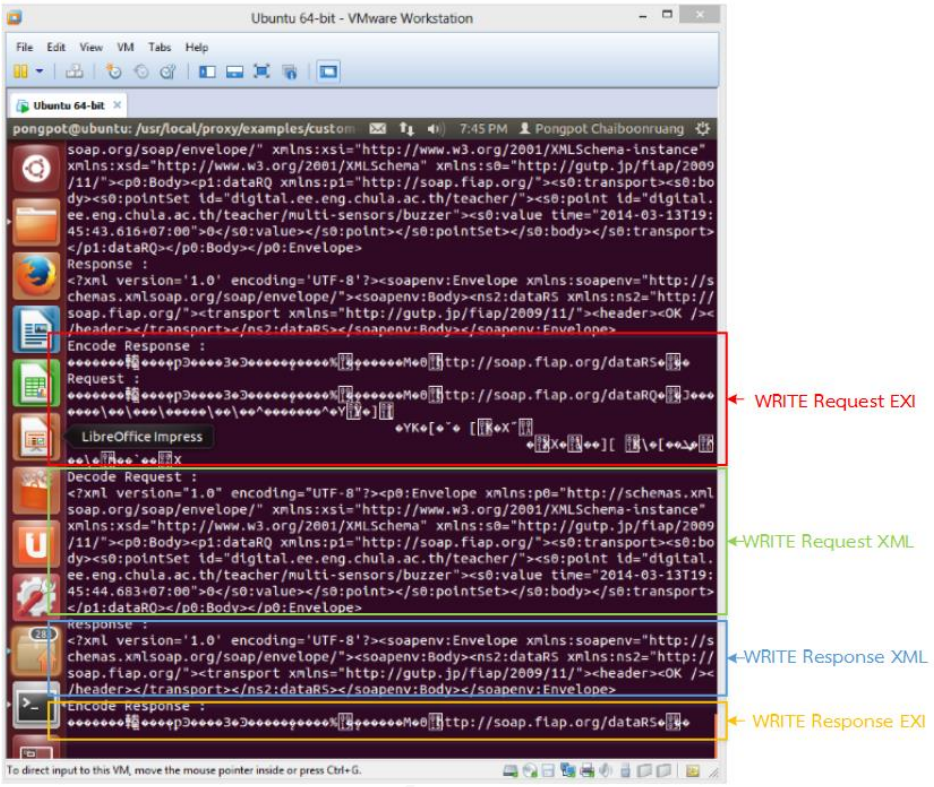
รูปที่ 5-9 ข้อมูลที่ถูกเก็บในสตอเรจ

ผลการทดสอบ proxy ในการบีบอัดข้อมูล WRITE โพรโทคอลนั้น proxy สามารถทำงานได้อย่างถูกต้อง โดยเกทเวย 6LoWPAN ที่พัฒนาขึ้นในงานวิจัยนี้สามารถส่งข้อมูล WRITE โพรโทคอลผ่าน proxy บีบอัดข้อมูลทั้งสองไปยังสตอเรจได้ดังแสดงในรูปที่ 5-14 ส่วนผลการการทำงานของ proxy ทั้งสองฝั่งนั้นมีรายละเอียดดังต่อไปนี้



รูปที่ 5-10 LAN Proxy ขณะทำการบีบอัดข้อมูล WRITE โพรโทคอล

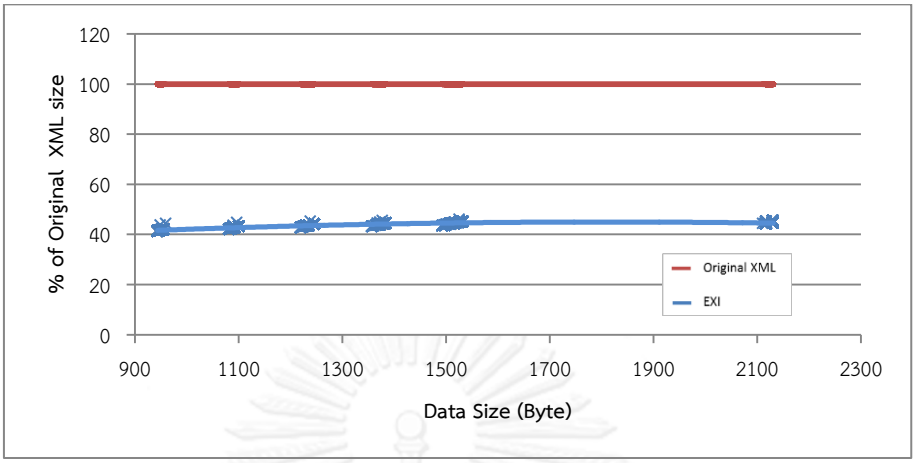
ผลการทดสอบ proxy บีบอัดข้อมูลในฝั่ง LAN ดังแสดงในรูปที่ 5-10 นั้น proxy สามารถทำงานได้อย่างถูกต้อง เริ่มต้นจากการรับข้อมูล WRITE Request XML (สีแดง) ที่ได้มาจากเกทเวย 6LoWPAN มาทำการบีบอัดซึ่งจะได้ข้อมูล WRITE Request EXI (สีเขียว) แล้วส่งไปยัง proxy บีบอัดข้อมูลฝั่งผู้ให้บริการสตอเรจ จากนั้น proxy จะได้ข้อมูลตอบรับที่ถูกบีบอัด WRITE Response EXI (สีน้ำเงิน) ข้อมูลนี้จะถูกคลายการบีบอัด เพื่อให้ได้ข้อมูล WRITE Response XML (สีเหลือง) แล้วทำการส่งกลับไปยังเกทเวย 6LoWPAN เป็นอันเสร็จการทำงานของ proxy บีบอัดข้อมูลฝั่ง LAN ในการสื่อสาร WRITE โพรโทคอลตามมาตรฐาน IEEE1888



รูปที่ 5-11 Cloud Storage Proxy ขณะทำการบีบอัดข้อมูล WRITE โพรโทคอล

ผลการทดสอบ proxy บีบอัดข้อมูลในฝั่งผู้ให้บริการ cloud สตอเรจดังแดงในรูปที่ 5-11 นั้น proxy สามารถทำงานได้อย่างถูกต้อง เริ่มต้นจากการรับข้อมูล WRITE Request EXI (สีแดง) ที่ได้มาจาก proxy บีบอัดข้อมูลในฝั่ง LAN มาทำการคลายการบีบอัด ซึ่งได้ข้อมูล WRITE Request XML (สีเขียว) ที่จะส่งต่อไปยังสตอเรจ จากนั้น proxy ได้ข้อมูลตอบรับจากสตอเรจที่อยู่ในรูปแบบ WRITE Response XML (สีน้ำเงิน) ข้อมูลนี้จะถูกทำการบีบอัด ซึ่งได้ข้อมูล WRITE Response EXI (สีเหลือง) ที่จะส่งกลับไปยัง proxy บีบอัดข้อมูลในฝั่ง LAN เป็นอันเสร็จการทำงานของ proxy บีบอัดข้อมูลในฝั่งผู้ให้บริการ cloud สตอเรจในการสื่อสาร WRITE โพรโทคอลตามมาตรฐาน IEEE1888

5.2.2.2. ผลการทดสอบประสิทธิภาพในการบีบอัดข้อมูล WRITE โพรโทคอล

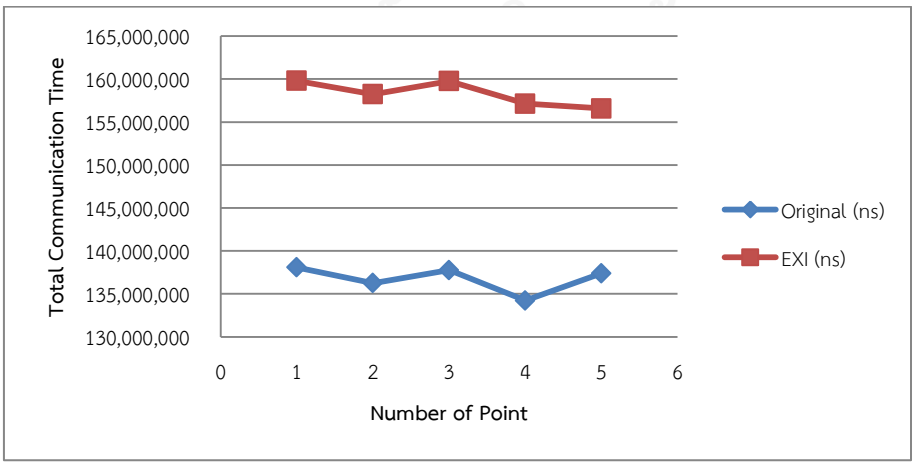


รูปที่ 5-12 ประสิทธิภาพในการบีบอัดข้อมูล WRITE Request

จากผลการทดสอบการบีบอัดข้อมูล WRITE Request ดังแสดงรูปที่ 5-12 proxy สามารถบีบอัดข้อมูลให้มีขนาด 41.31% - 46.00% และมีค่าเฉลี่ยเท่ากับ 43.47% เมื่อเทียบกับข้อมูลต้นฉบับ ซึ่งถือว่ามีการบีบอัดข้อมูลที่ไม่ค่อยมีประสิทธิภาพสูงมากนัก อันเนื่องมาจากในข้อมูล WRITE Request นั้นจะมีชื่อของ Point ID ที่ไม่ซ้ำกัน ทำให้กระบวนการ EXI ต้องเก็บชื่อของ Point ID ที่มีความแตกต่างลงในข้อมูลที่มีการบีบอัดไปด้วย

ส่วนข้อมูล WRITE Response นั้นจะมีการตอบกลับเพียงรูปแบบเดียว ซึ่ง proxy สามารถทำการบีบอัดให้ข้อมูลมีขนาด 46.52% เมื่อเทียบกับข้อมูลต้นฉบับ

5.2.2.3. ผลการทดสอบเวลาในการบีบอัดข้อมูล WRITE โพรโทคอล

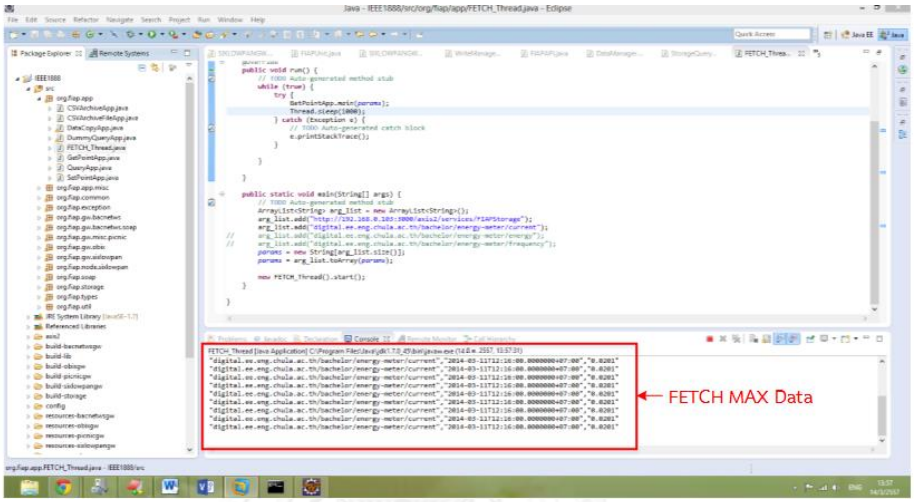


รูปที่ 5-13 เวลาในการบีบอัดข้อมูล WRITE โพรโทคอล

เวลาในการบีบอัดข้อมูล WRITE โพรโทคอลขนาด 1 – 5 Point ดังแสดงในรูปที่ 5-13 เฉลี่ยแล้วใช้เวลาในการรับส่งข้อมูลทั้งหมดมากกว่าข้อมูลในรูปแบบปกติอยู่ 20 mS โดยขนาดของข้อมูลที่ใหญ่ขึ้นนั้นไม่มีผลต่อเวลาในการบีบอัดข้อมูลมากนัก

5.2.3. ผลการทดสอบ proxy ในการบีบอัดข้อมูล FETCH โพรโทคอล

5.2.3.1. การทดสอบการทำงานของ proxy กับ FETCH โพรโทคอล

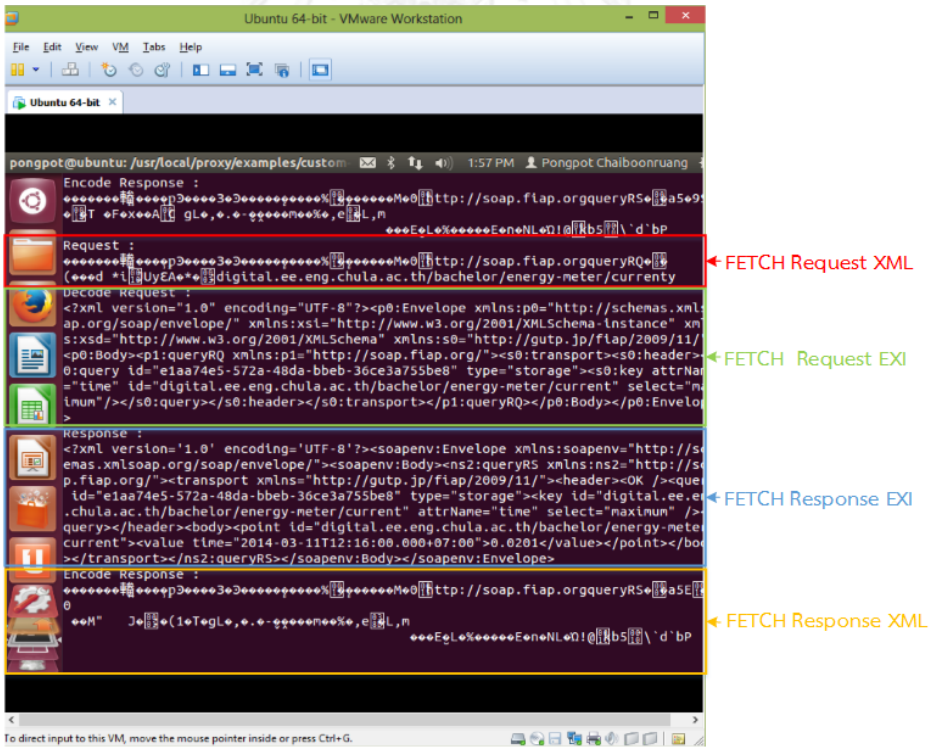


รูปที่ 5-14 โปรแกรม FETCH ข้อมูลจากสตอเรจผ่าน proxy บีบอัดข้อมูล

ผลการทดสอบ proxy ในการบีบอัดข้อมูล FETCH โพรโทคอลนั้น proxy สามารถทำงานได้อย่างถูกต้อง โดยแอพพลิเคชั่นที่เขียนขึ้นมาบน IEEE1888 Java Reference Code นั้นสามารถส่งข้อมูล FETCH โพรโทคอลผ่าน proxy บีบอัดข้อมูลทั้งสองไปยังสตอเรจได้ และได้รับข้อมูลตอบกลับที่ถูกต้องดังแสดงในรูปที่ 5-14 ส่วนผลการการทำงานของ proxy ทั้งสองฝั่งนั้นมีรายละเอียดดังต่อไปนี้

ผลการทดสอบ proxy บีบอัดข้อมูลในฝั่ง LAN ดังแสดงในรูปที่ 5-15 นั้น proxy สามารถทำงานได้อย่างถูกต้อง เริ่มต้นจากการรับข้อมูล FETCH Request XML (สีแดง) ที่ได้มาจากแอพพลิเคชั่น มาทำการบีบอัดซึ่งจะได้ข้อมูล FETCH Request EXI (สีเขียว) แล้วส่งไปยัง proxy บีบอัดข้อมูลฝั่งผู้ให้บริการสตอเรจ จากนั้น proxy จะได้ข้อมูลตอบกลับที่ถูกบีบอัด FETCH Response EXI (สีน้ำเงิน) ข้อมูลนี้จะถูกคลายการบีบอัด เพื่อให้ได้ข้อมูล FETCH Response XML (สีเหลือง) แล้วทำการส่งกลับไปยังแอพพลิเคชั่น เป็นอันเสร็จการทำงานของ proxy บีบอัดข้อมูลฝั่ง LAN ในการสื่อสาร FETCH โพรโทคอลตามมาตรฐาน IEEE1888

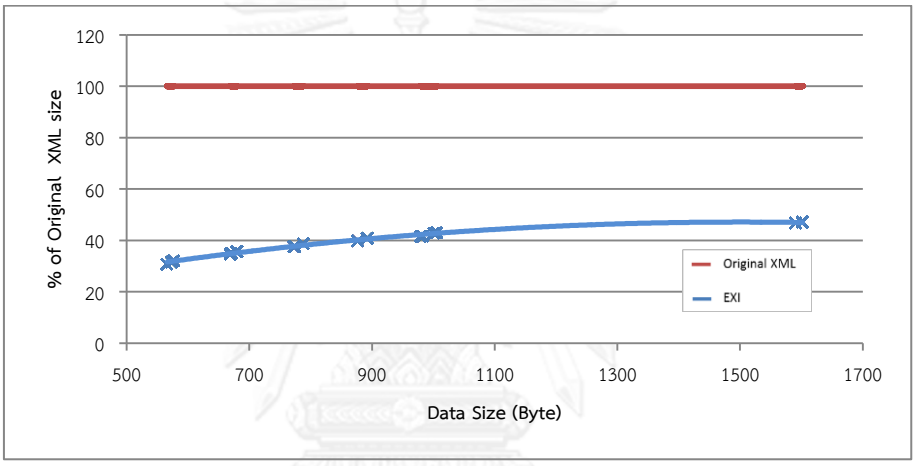
รูปที่ 5-15 LAN Proxy ขณะทำการบีบอัดข้อมูล FETCH โพรโทคอล



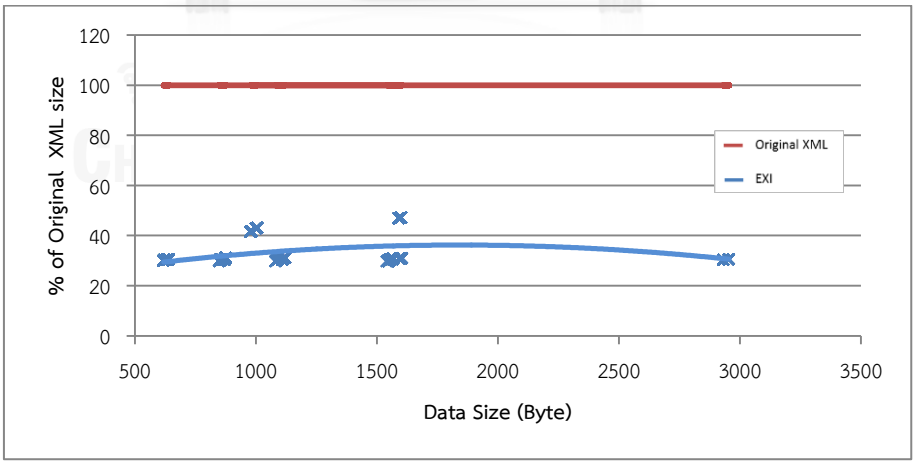
รูปที่ 5-16 Cloud Storage Proxy ขณะทำการบีบอัดข้อมูล FETCH โพรโทคอล

ผลการทดสอบ proxy บีบอัดข้อมูลในฝั่งผู้ให้บริการ cloud สตอเรจดังแสดงในรูปที่ 5-16 นั้น proxy สามารถทำงานได้อย่างถูกต้อง เริ่มต้นจากการรับข้อมูล FETCH Request EXI (สีแดง) ที่ได้มาจาก proxy บีบอัดข้อมูลในฝั่ง LAN มาทำการคลายการบีบอัด ซึ่งได้ข้อมูล FETCH Request XML (สีเขียว) เพื่อส่งต่อไปยังสตอเรจ จากนั้น proxy ได้ข้อมูลตอบรับจากสตอเรจที่อยู่ในรูปแบบ FETCH Response XML (สีน้ำเงิน) ข้อมูลนี้จะถูกทำการบีบอัด ซึ่งได้ข้อมูล FETCH Response EXI (สีเหลือง) ที่ส่งกลับไปยัง proxy บีบอัดข้อมูลในฝั่ง LAN เป็นอันเสร็จการทำงานของ proxy บีบอัดข้อมูลในฝั่งผู้ให้บริการ cloud สตอเรจในการสื่อสาร FETCH โพรโทคอลตามมาตรฐาน IEEE1888

5.2.3.2. ผลการทดสอบประสิทธิภาพในการบีบอัดข้อมูล FETCH ค่าล่าสุด



รูปที่ 5-17 ประสิทธิภาพในการบีบอัดข้อมูล FETCH MAX Request

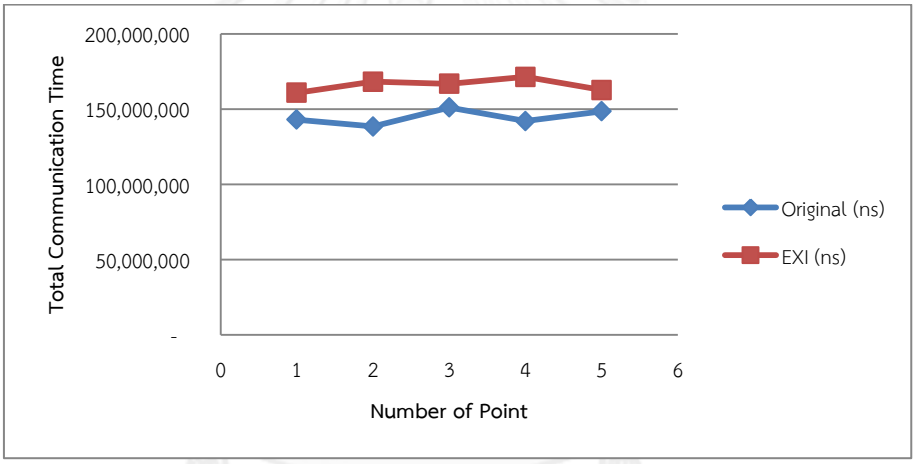


รูปที่ 5-18 ประสิทธิภาพในการบีบอัดข้อมูล FETCH MAX Response

จากผลการทดสอบการบีบอัดข้อมูล FETCH MAX Request ดังแสดงรูปที่ 5-17 proxy สามารถบีบอัดข้อมูลให้มีขนาด 30.74% - 47.19% และมีค่าเฉลี่ยเท่ากับ 38.97% เมื่อเทียบกับข้อมูลต้นฉบับ ซึ่งกระบวนการบีบอัดข้อมูลมีประสิทธิภาพสูงในการบีบอัดข้อมูลขนาดเล็กมากกว่าข้อมูลขนาดใหญ่ เพราะในข้อมูล FETCH MAX Request ที่มีขนาดใหญ่นั้นจะบรรจุชื่อ Point ID ที่มีความแตกต่างกันมากกว่าในข้อมูลขนาดเล็ก ซึ่งทำให้กระบวนการ EXI ต้องเก็บชื่อของ Point ID ที่แตกต่างนี้ลงในข้อมูลที่มีการบีบอัดไปด้วย

จากผลการทดสอบการบีบอัดข้อมูล FETCH MAX Response ดังแสดงรูปที่ 5-18 proxy สามารถบีบอัดข้อมูลให้มีขนาด 29.59% - 47.19% และมีค่าเฉลี่ยเท่ากับ 32.67% เมื่อเทียบกับข้อมูลต้นฉบับ ซึ่งจากกราฟจะเห็นว่ามีข้อมูลอยู่สองกลุ่มที่มีอัตราการบีบอัดข้อมูลที่มากกว่า 40% ซึ่งน่าจะเกิดจากการ FETCH ข้อมูลที่มีชื่อของ Point ID ที่แตกต่างกันจำนวนหลาย Point ID

5.2.3.3. **ผลการทดสอบเวลาในการบีบอัดข้อมูล FETCH ค่าล่าสุด**

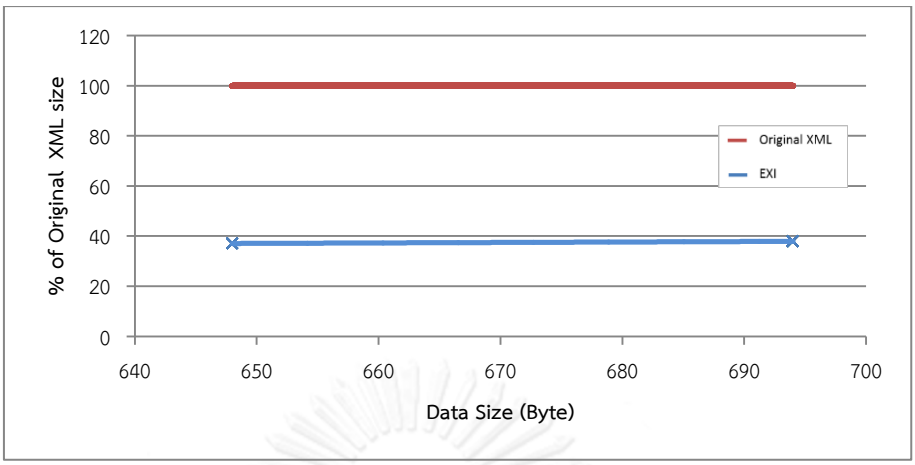


รูปที่ 5-19 เวลาในการบีบอัดข้อมูล FETCH ค่าล่าสุด

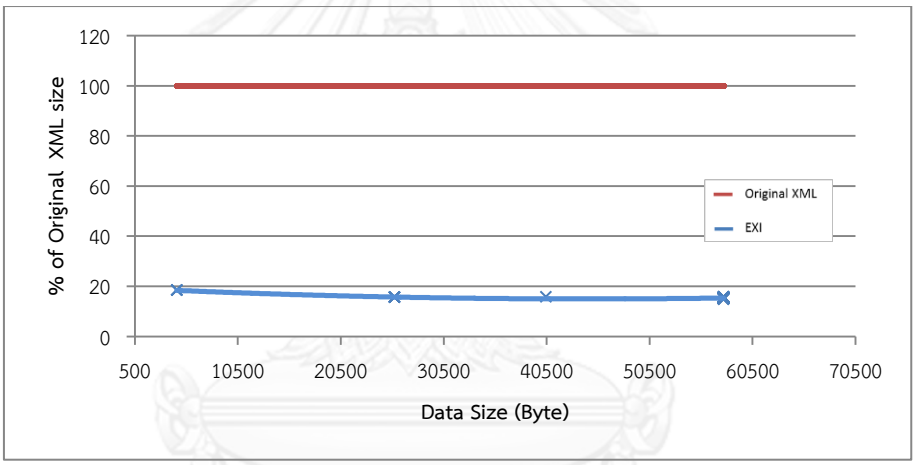
เวลาในการบีบอัดข้อมูล FETCH ค่าล่าสุดขนาด 1 – 5 Point ดังแสดงในรูปที่ 5-19 เฉลี่ยแล้วใช้เวลาในการรับส่งข้อมูลทั้งหมดมากกว่าข้อมูลในรูปแบบปกติอยู่ 20 mS โดยขนาดของข้อมูลที่ใหญ่ขึ้นนั้นไม่มีผลต่อเวลาในการบีบอัดข้อมูลมากนัก

5.2.3.4. **ผลการทดสอบประสิทธิภาพในการบีบอัดข้อมูล FETCH ค่าย้อนหลัง**

จากผลการทดสอบการบีบอัดข้อมูล FETCH Historical Request ดังแสดงรูปที่ 5-20 proxy สามารถบีบอัดข้อมูลให้มีขนาด 37.04% - 37.89% และมีค่าเฉลี่ยเท่ากับ 37.71% เมื่อเทียบกับข้อมูลต้นฉบับ



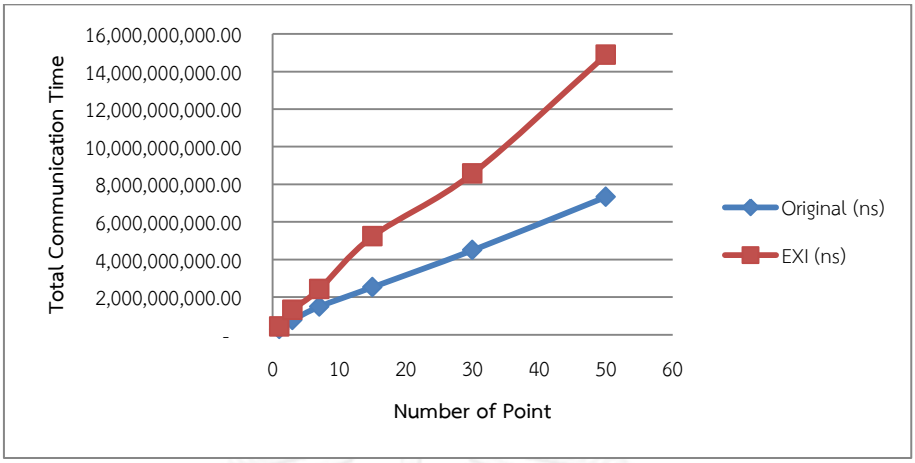
รูปที่ 5-20 ประสิทธิภาพในการบีบอัดข้อมูล FETCH Historical Request



รูปที่ 5-21 ประสิทธิภาพในการบีบอัดข้อมูล FETCH Historical Response

จากผลการทดสอบการบีบอัดข้อมูล FETCH Historical Response ดังแสดงรูปที่ 5-21 proxy สามารถบีบอัดข้อมูลให้มีขนาด 14.69% - 18.43% และมีค่าเฉลี่ยเท่ากับ 15.54% เมื่อเทียบกับข้อมูลต้นฉบับ ซึ่งจะเห็นได้ว่า proxy มีประสิทธิภาพในการบีบอัดข้อมูลที่สูงมาก เนื่องจากภายในข้อมูลตอบกลับนี้จะมีข้อมูลที่ซ้ำกันจำนวนมาก เช่น ชื่อ Element ชื่อ Attribute และ เวลาที่ Stamp กับข้อมูล เป็นต้น ทำให้กระบวนการบีบอัด EXI สามารถที่จะเก็บข้อมูลเหล่านี้ให้อยู่ในรูปแบบที่มีขนาดเล็กที่สุดได้

5.2.3.5. ผลการทดสอบเวลาในการบีบอัดข้อมูล FETCH ค่าย้อนหลัง



รูปที่ 5-22 เวลาในการบีบอัดข้อมูล FETCH ค่าย้อนหลัง

เวลาในการบีบอัดข้อมูล FETCH ค่าย้อนหลัง 1 – 60 วัน ดังแสดงในรูปที่ 5-22 จะเห็นได้ว่าการสื่อสารข้อมูลผ่าน Proxy บีบอัดข้อมูลนั้นใช้เวลาในการรับส่งข้อมูลทั้งหมดมากกว่าข้อมูลในรูปแบบปกติอยู่ 4 วินาทีต่อข้อมูลใน 30 วัน โดยจำนวนวันที่มากขึ้นส่งผลต่อการใช้เวลาในการบีบอัดที่มากขึ้นตาม

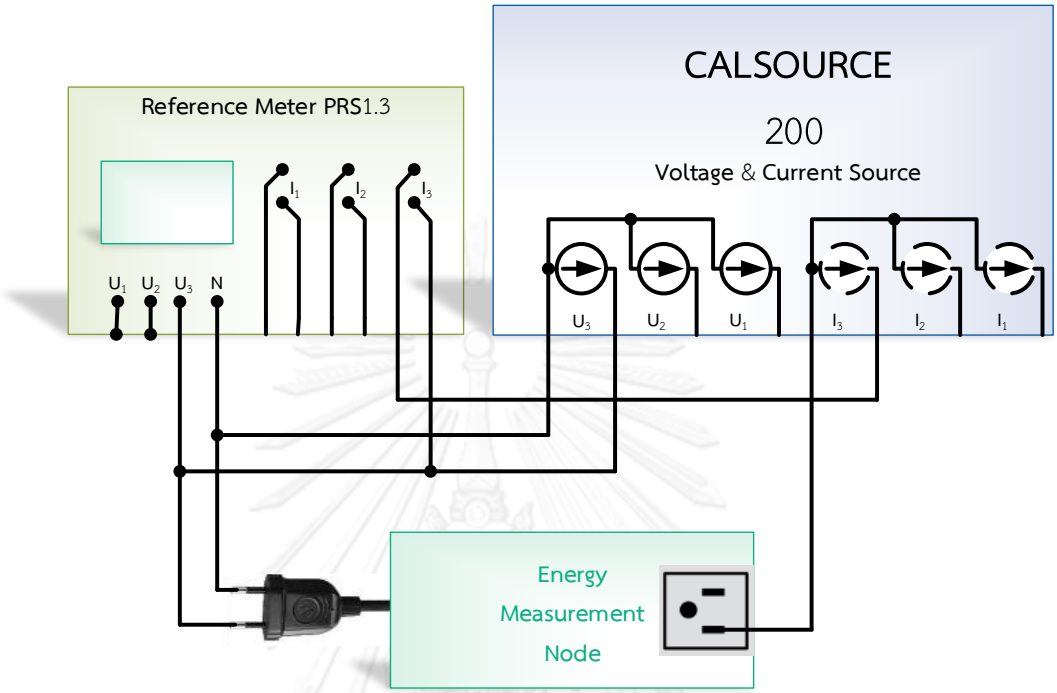
5.2.4. สรุปผลการทดสอบ proxy ในการบีบอัดข้อมูล

จากผลการทดสอบ proxy ในการบีบอัดข้อมูล WRITE และ FETCH โพรโทคอลพบว่า proxy ที่ได้ทำการพัฒนาขึ้นมานี้สามารถบีบอัดข้อมูลในมาตรฐาน IEEE1888 ให้มีขนาด 14.69 – 47.19% และมีค่าเฉลี่ยเท่ากับ 35.81% เมื่อเทียบกับข้อมูลต้นฉบับ หรือมีความสามารถในการบีบอัดข้อมูล 64.19% ซึ่งจากการทดสอบพบว่า Point ID นั้นมีนัยสำคัญต่อขนาดของข้อมูลภายหลังการบีบอัดเป็นอย่างมาก โดย Point ID ที่มีความแตกต่างกันมากภายในข้อมูล WRITE และ FETCH ทำให้ข้อมูลภายหลังการบีบอัดมีขนาดใหญ่ตามไปด้วย เมื่อเปรียบเทียบกับข้อมูล WRITE และ FETCH ที่มี Point ID ที่มี Prefix คล้ายคลึงกัน

ความล่าช้าของเวลาที่เกิดจากการบีบอัดข้อมูลทั้งสามนั้นถูกทดสอบในระบบเครือข่ายที่มีแบนวิธด์ในการรับส่งข้อมูลความเร็วสูง เพื่อทดสอบหาความล่าช้าในการบีบอัดข้อมูล แต่ในระบบที่นำไปใช้งานจริงช่องทางจะมีการจราจรที่หนาแน่น ซึ่งข้อมูลที่มีขนาดเล็กกว่าจะทำให้ได้เปรียบ และใช้เวลาในการรับส่งที่น้อยกว่าข้อมูลปกติ

5.3. การทดสอบวงจรวัดพลังงานไฟฟ้า

5.3.1. วิธีการทดสอบวงจรวัดพลังงานไฟฟ้า



รูปที่ 5-23 การทดสอบวงจรวัดพลังงานไฟฟ้า

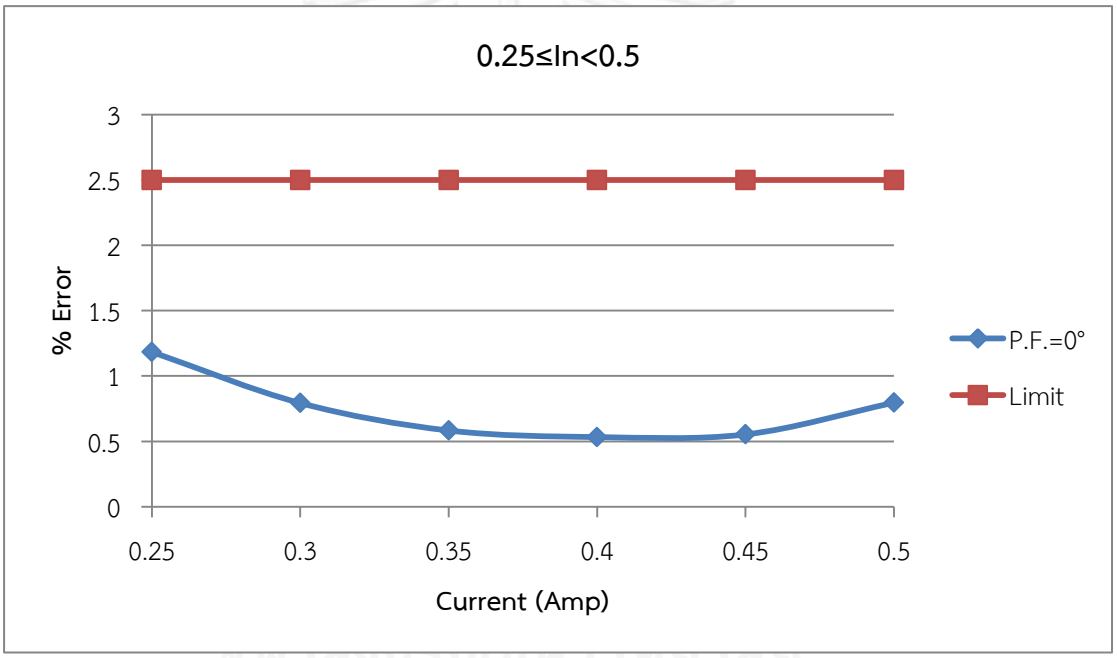
วิธีการทดสอบวงจรวัดพลังงานไฟฟ้านั้นจะใช้เครื่องจ่ายโหลดพลังงานไฟฟ้าเสมือน (Phanthom Load) รุ่น CALSOURCE 200 ของบริษัท MEH (Energie Messtechnik GmbH) จ่ายพลังงานไฟฟ้าให้แก่วงจรวัดพลังงานไฟฟ้า โดยค่าพลังงานไฟฟ้าที่อ่านได้จากวงจรวัดพลังงานไฟฟ้านั้นจะแปลงเป็นการกระพริบของ LED ซึ่งถูกตรวจจับโดย มิเตอร์อ้างอิง รุ่น PRS1.3 ที่มีความแม่นยำ 0.1% ของบริษัทเดียวกับเครื่องจ่ายโหลดพลังงานไฟฟ้าเสมือน โดยมิเตอร์นี้จะทำการแปลงความถี่ในการกระพริบของ LED เป็นปริมาณพลังงานไฟฟ้าที่วงจรวัดได้ แล้วไปทำการเปรียบเทียบกับค่าพลังงานที่ตัวมิเตอร์เองสามารถอ่านได้เพื่อหาค่าความผิดพลาดในการวัดพลังงานของวงจรวัดพลังงานไฟฟ้า

5.3.2. ผลการทดสอบวงจรวัดพลังงานไฟฟ้า

การทดสอบวงจรวัดพลังงานไฟฟ้านั้นได้ใช้มาตรฐาน IEC 62053-21 [23] เป็นข้อกำหนดในการทดสอบการวัดพลังงานไฟฟ้า ซึ่งวงจรวัดพลังงานไฟฟ้านี้ได้ออกแบบให้ใช้งานที่กระแส 5 แอมป์ Class 2 ซึ่งมีข้อกำหนดของค่าความผิดพลาดในการวัดพลังงานดังในตารางที่ 5-1 โดยแบ่งการวัดพลังงานไฟฟ้าออกเป็น 4 กลุ่มใหญ่ ดังต่อไปนี้

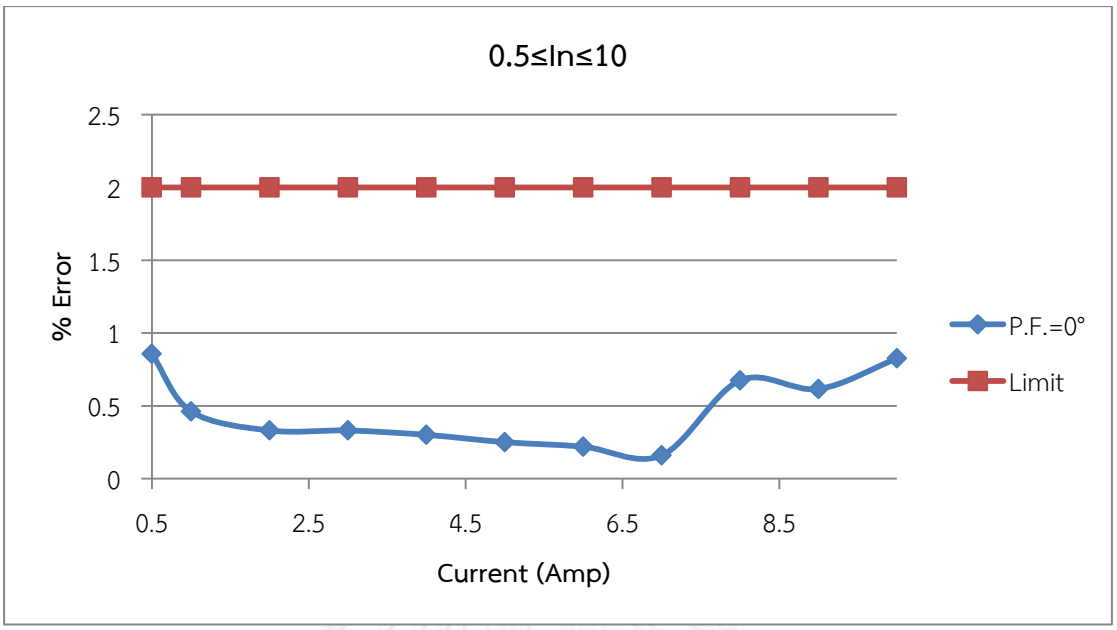
ตารางที่ 5-1 ข้อกำหนดความผิดพลาดในการวัดพลังงานไฟฟ้าตามมาตรฐาน IEC 62053-21 Class 2

Value of current	Power factor	% error limit
0.25 ≤ I < 0.5	1	±2.5
0.5 ≤ I ≤ 10	1	±2.0
0.5 ≤ I < 1	0.5 inductive 0.8 capacitive	±2.5 -
0.1 ≤ I ≤ 10	0.5 inductive 0.8 capacitive	±2.0 -



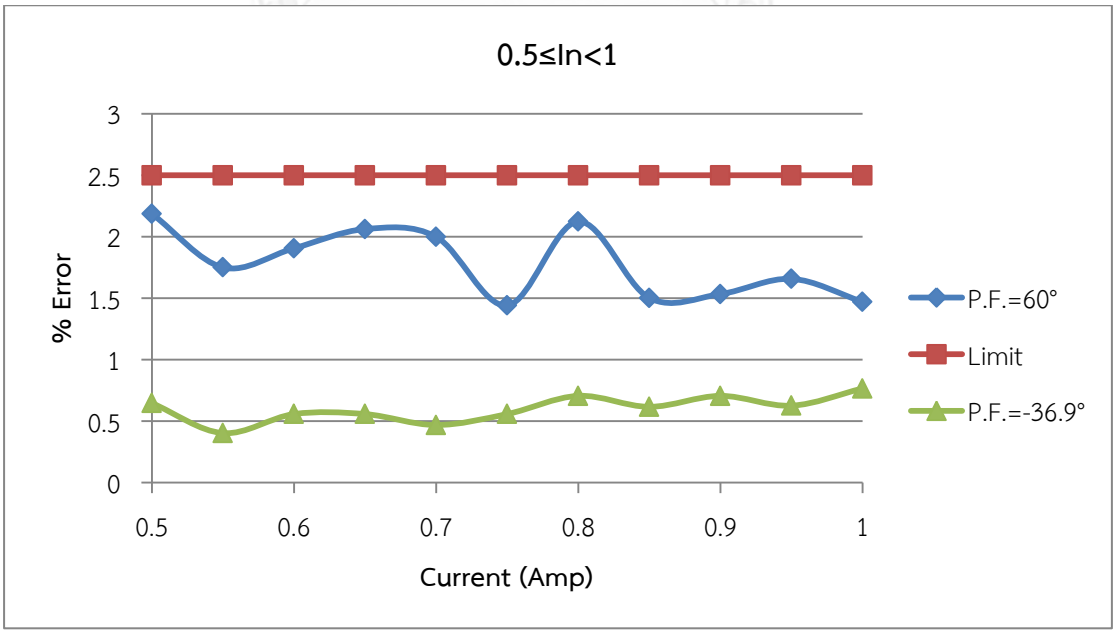
รูปที่ 5-24 ความผิดพลาดในการวัดพลังงานไฟฟ้าที่กระแสไฟฟ้า 0.25≤In<0.5 P.F.=0°

1. ผลการทดสอบการวัดพลังงานไฟฟ้าในช่วงกระแส 0.25 ≤ I < 0.5 P.F. = 1 ดังแสดงในรูปที่ 5-24 มีค่าความผิดพลาดสูงสุด 1.184% ซึ่งผ่านเกณฑ์ตามตารางที่ 5-1 ที่ห้ามมีความผิดพลาดเกิน 2.5%



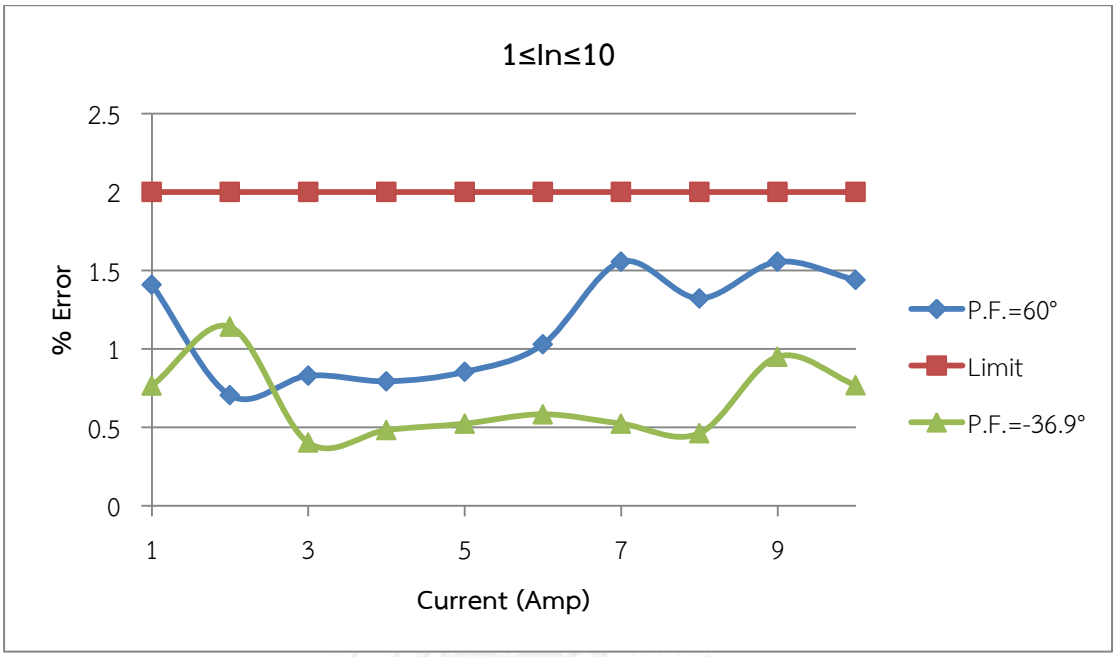
รูปที่ 5-25 ความผิดพลาดในการวัดพลังงานไฟฟ้าที่กระแสไฟฟ้า 0.5≤In<10 P.F.=0°

2. ผลการทดสอบการวัดพลังงานไฟฟ้าในช่วงกระแส 0.5 ≤ I ≤ 10 P.F. = 1 ดังแสดงในรูปที่ 5-25 มีค่าความผิดพลาดสูงสุด 0.857% ซึ่งผ่านเกณฑ์ตามตารางที่ 5-1 ที่ห้ามมีความผิดพลาดเกิน 2.0%



รูปที่ 5-26 ความผิดพลาดในการวัดพลังงานไฟฟ้าที่กระแสไฟฟ้า 0.5≤In<1 P.F.=60° และ P.F.=-36.9°

3. ผลการทดสอบการวัดพลังงานไฟฟ้าในช่วงกระแส 0.5 ≤ I < 1 P.F. = 0.5 inductive ดังแสดงในรูปที่ 5-26 มีค่าความผิดพลาดสูงสุด 2.187% ซึ่งผ่านเกณฑ์ตามตารางที่ 5-1 ที่ห้ามมีความผิดพลาดเกิน 2.5%



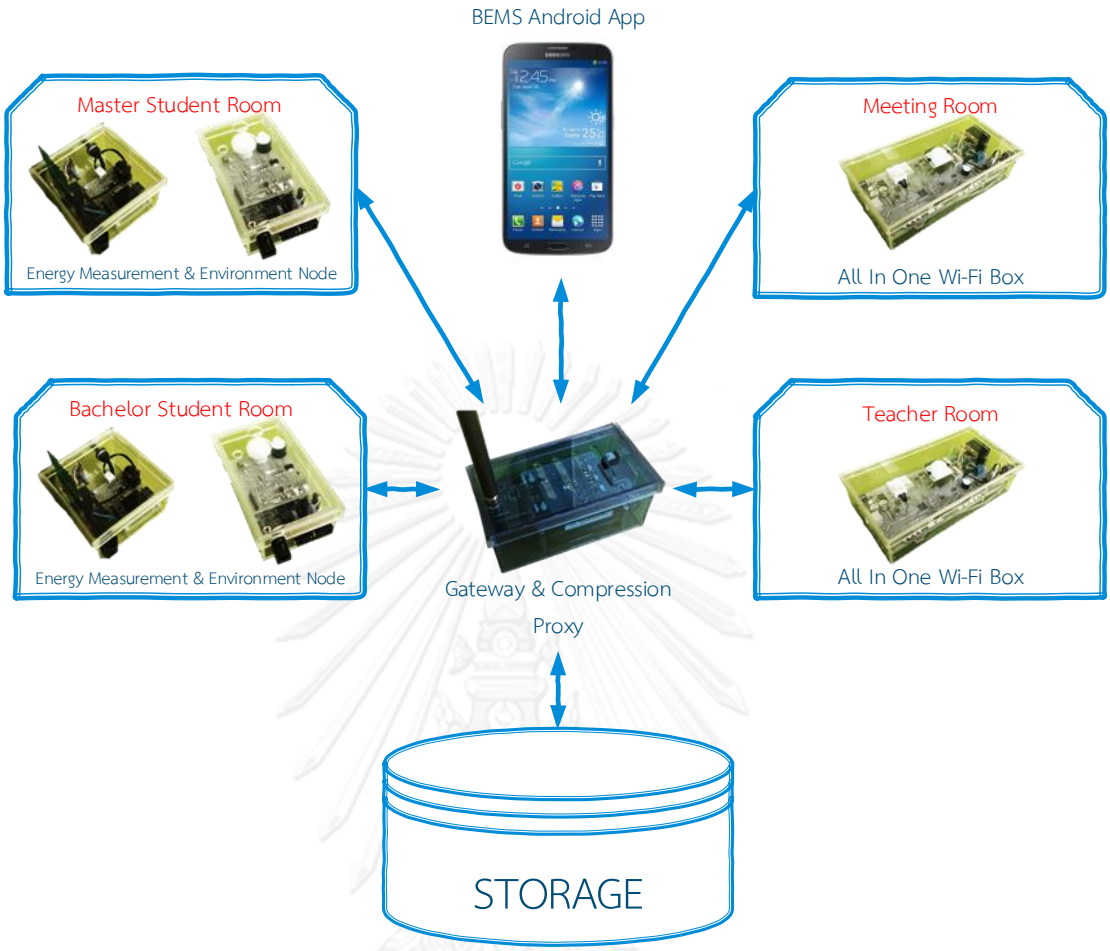
รูปที่ 5-27 ความผิดพลาดในการวัดพลังงานไฟฟ้าที่กระแสไฟฟ้า 1≤In≤10 P.F.=60° และ P.F.=-36.9°

4. ผลการทดสอบการวัดพลังงานไฟฟ้าในช่วงกระแส 0.1 ≤ I ≤ 10 P.F. = 0.5 inductive ดังแสดงในรูปที่ 5-27 มีค่าความผิดพลาดสูงสุด 1.555% ซึ่งผ่านเกณฑ์ตามตารางที่ 5-1 ที่ห้ามมีความผิดพลาดเกิน 2.0%

5.4. การทดสอบระบบจัดการพลังงานไฟฟ้าภายในอาคาร

5.4.1. วิธีการทดสอบระบบจัดการพลังงานไฟฟ้าภายในอาคาร

การทดสอบระบบจัดการพลังงานไฟฟ้าภายในอาคารได้ถูกจัดขึ้นที่ห้องปฏิบัติการวิจัยการออกแบบวงจรฝังตัวและวงจรรวม ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ดังแสดงในรูปที่ 5-28 ซึ่งประกอบไปด้วยห้องจำนวน 4 ห้อง ในแต่ละห้องจะถูกติดตั้งโหนดวัดพลังงานไฟฟ้า โหนดวัดสภาวะแวดล้อม และ All In One Wi-Fi Box เพื่อทำการวัดค่าแล้วส่งข้อมูลไปเก็บยังสตอเรจ แอพพลิเคชั่นจะนำข้อมูลที่บันทึกอยู่ในสตอเรจมาแสดงผลข้อมูลให้ผู้ใช้งานได้รับรู้ถึงค่าต่างๆภายในระบบ และสามารถสั่งการควบคุมการทำงานของเครื่องใช้ไฟฟ้า



รูปที่ 5-28 การทดสอบระบบจัดการพลังงานไฟฟ้าภายในอาคาร

5.4.2. ผลการทดสอบระบบจัดการพลังงานไฟฟ้าภายในอาคาร

จากการติดตั้งระบบจัดการพลังงานไฟฟ้าภายในอาคารแล้วทดสอบการทำงานในส่วนต่างๆของระบบดังแสดงในรูปที่ 5-29 รูปที่ 5-30 และรูปที่ 5-31 ได้ผลการทดสอบดังต่อนี้

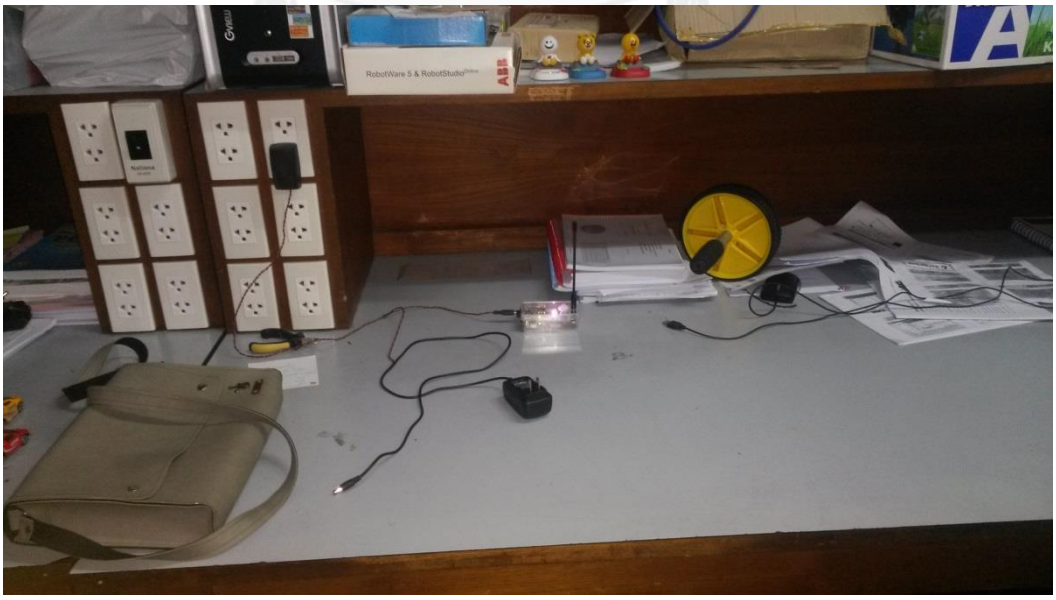
- เกทเวยสามารถสื่อสารระหว่างมาตรฐาน IEEE1888 กับ 6LoWPAN ในโพรโทคอล FETCH, WRITE, และ TRAP ได้อย่างถูกต้อง
- โหนดวัดพลังงานไฟฟ้าสามารถวัดพลังงานไฟฟ้า และควบคุม เปิด ปิด เครื่องใช้ไฟฟ้าได้
- โหนดวัดสภาวะแวดล้อมสามารถวัดอุณหภูมิ วัดความชื้น แสงสว่าง และ ความเคลื่อนไหวได้ถูกต้อง

- ในบางเวลาโหนด 6LoWPAN ขาดการติดต่อไม่สามารถสื่อสารกับเกทเวยได้ ซึ่งได้ทำการแก้ไขโดยให้โหนด 6LoWPAN พยายามทำการเชื่อมต่อกับเกทเวยใหม่อีกครั้ง

- สตอเรจเก็บบันทึกข้อมูลที่ WRITE มาจาก เกทเวย และ All In One Wi-Fi Box ได้

- แอพพลิเคชั่นสามารถ FETCH ข้อมูลจาก เกทเวย และ สตอเรจ มาแสดงผลได้ถูกต้อง อีกทั้งยัง WRITE ไปยังเกทเวยเพื่อควบคุมการทำงานเของเครื่องใช้ไฟฟ้าได้

รูปที่ 5-29 การทดสอบโหนดวัดพลังงานไฟฟ้า



รูปที่ 5-30 การทดสอบโหนดวัดสภาพแวดล้อม

รูปที่ 5-31 การทดสอบแอพพลิเคชั่น

บทที่ 6

ข้อสรุป และข้อเสนอแนะ

ในบทสุดท้ายนี้ได้กล่างถึงสรุปผลการวิจัย ว่าสามารถนำงานวิจัยนี้ไปพัฒนาต่อได้อย่างไรบ้าง มีสิ่งใดที่ทำแล้วไม่ประสบผลสำเร็จบ้าง และข้อเสนอแนะ จากการทำงานวิจัยนี้

6.1. สรุป

วิทยานิพนธ์ฉบับนี้นำเสนอรายละเอียดการออกแบบ และพัฒนาเทคนิคการบีบอัดข้อมูลสำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคารบนพื้นฐานของมาตรฐาน IEEE1888 โดยศึกษาค้นคว้าหากระบวนการบีบอัดข้อมูลที่เหมาะสมแก่การนำมาบีบอัดข้อมูลในมาตรฐาน IEEE1888 แล้วนำกระบวนการบีบอัดนั้นไปสร้างระบบจัดการพลังงานไฟฟ้าภายในอาคารที่รองรับข้อมูลที่มีการบีบอัด ซึ่งจะต้องมีการเปลี่ยนแปลงระบบเดิมตามมาตรฐาน IEEE1888 ให้น้อยที่สุด

จากระบบที่ได้นำเสนอ ผู้วิจัยได้เริ่มต้นจากการนำกระบวนการบีบอัดที่ไม่มีการสูญเสีย 4 กระบวนการได้แก่ EXI, Gzip, Zip และ 7 Zip มาทดลองหาประสิทธิภาพในการบีบอัดข้อมูลตามมาตรฐาน IEEE1888 ในโพรโทคอล FETCH และ WRITE จากการทดลองพบว่ากระบวนการบีบอัดข้อมูลที่มีประสิทธิภาพสูงสุดคือ EXI ซึ่งมีอัตราการบีบอัดข้อมูลเฉลี่ย 75.59% จากนั้นได้นำกระบวนการ EXI ไปพัฒนาเป็น Proxy ที่มีความสามารถในการบีบอัดข้อมูลในมาตรฐาน IEEE1888 ร่วมกับการพัฒนาระบบจัดการพลังงานขึ้นด้วย

ระบบจัดการพลังงานที่ได้พัฒนาขึ้นมาใหม่นี้ได้ออกแบบให้ระบบจัดการพลังงานภายในหลายๆอาคารสามารถใช้งานสตอเรจในการเก็บข้อมูลร่วมกัน อีกทั้งถูกออกแบบให้รองรับข้อมูลที่มีการบีบอัดโดยกระบวนการบีบอัด EXI ภายในระบบได้ทำการพัฒนาฮาร์ดแวร์ที่ประกอบไปด้วย เกทเวยเพื่อเชื่อมต่อมาตรฐาน IEEE1888 กับมาตรฐาน 6LoWPAN อีกทั้งทำการพัฒนาโหนดในมาตรฐาน 6LoWPAN จำนวน 2 โหนด ประกอบไปด้วยโหนดวัดพลังงาน และโหนดวัดสภาวะแวดล้อม ที่ทำการวัดค่าต่างๆที่จำเป็นต่อการตัดสินใจของระบบจัดการพลังงาน ในส่วนของซอฟต์แวร์นั้นได้ทำการพัฒนา Proxy ที่สามารถบีบอัดข้อมูลในมาตรฐาน IEEE1888 โดยเฉพาะที่เหมาะแก่การใช้งานในระบบสื่อสารที่มีการจราจรหนาแน่น หรือค่าใช้จ่ายสูง ซอฟต์แวร์แปลงมาตรฐาน IEEE1888 กับ มาตรฐาน 6LoWPAN ซอฟต์แวร์ 6LoWPAN โหนด และแอพพลิเคชั่นจัดการพลังงานภายในอาคารบนระบบปฏิบัติการ Android

จากการทดสอบพบว่า Proxy ที่ได้ทำการพัฒนาในวิทยานิพนธ์นี้มีความสามารถในการบีบอัดข้อมูลโดยเฉลี่ย 64.18% ซึ่งอยู่ในช่วงที่น่าพึงพอใจ การทดสอบความผิดพลาดในการวัดพลังงานไฟฟ้าของโหนดวัดพลังงานอยู่ใน Class2 ตามมาตรฐาน IEC 62053-21 การทำงานของเกท

เวย 6LoWPAN โหนด และแอพพลิเคชั่นจัดการพลังงานภายในอาคารสามารถทำงานได้ตามที่ได้ออกแบบไว้

6.2. ประโยชน์ที่คาดว่าจะได้รับ

1. ได้แอนดรอยด์แอพพลิเคชั่นที่สามารถจัดการพลังงานไฟฟ้าภายในอาคารบนมาตรฐาน IEEE1888
2. ได้เกทเวย์ที่ทำการเชื่อมต่อการสื่อสารระหว่างมาตรฐาน IEEE1888 กับ 6LoWPAN และสามารถบีบอัดข้อมูล

6.3. ข้อเสนอแนะ

1. ชุดข้อมูลที่นำมาทดสอบประสิทธิภาพ proxy บีบอัดข้อมูลนั้นยังไม่มีความหลากหลาย ถ้าหากนำ proxy บีบอัดข้อมูลไปทดลองกับระบบที่ติดตั้งใช้งานจริงน่าจะเห็นประสิทธิภาพในการทำงานได้อย่างแท้จริง
2. สามารถทำการพัฒนาเพิ่มเติมให้ proxy สามารถเก็บข้อมูลย้อนหลังบางส่วนไว้ เพื่อเป็นการลดภาระการทำงานของ Cloud สตอเรจ อีกทั้งยังช่วงเพิ่มความเร็วในการสื่อสารข้อมูลในระบบเครือข่ายท้องถิ่นเป็นอย่างมาก
3. ระบบจัดการพลังงานภายในอาคารที่ได้นำเสนอนี้ยังไม่มีการควบคุมอัตโนมัติ ซึ่งสามารถพัฒนาเพิ่มเติมโดยการนำข้อมูลที่ได้มาจากโหนดทั้งสองไปประมวลผล แล้วควบคุมการทำงานของเครื่องใช้ไฟฟ้าอย่างเหมาะสม
4. ชุดพัฒนา CC-6LoWPAN นั้นไม่สามารถทำการเปลี่ยนแปลงการทำงานในส่วนของ 6LoWPAN Software Stack อีกทั้งยังไม่สามารถกำหนด Static IP Address ได้ซึ่งทำให้ยากต่อการพัฒนาถ้าหากระบบมีขนาดใหญ่ขึ้น

รายการอ้างอิง

1. อรรจน์ เศรษฐบุตร, ขั้นตอนการบริหารจัดการพลังงานในอาคาร. วารสารวิชาการ "สถาปัตนกรรม", 2546. ฉบับที่ **2/2546**: p. 60-79.
2. Ochiai, H., et al. *FIAP: Facility information access protocol for data-centric building automation systems*. in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*. 2011.
3. *IEEE Standard for Ubiquitous Green Community Control Network Protocol.* IEEE Std 1888-2011, 2011: p. 1-65.
4. Gudgin, M., et al. *Simple Object Access Protocol 1.1*. 2003; Available from: http://www.w3.org/TR/ SOAP.
5. Adams, J.T. *An introduction to IEEE STD 802.15.4*. in *Aerospace Conference, 2006 IEEE*. 2006.
6. Kushalnagar, N., G. Motenegro, and C. Schumache, *"IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals,"RFC 4919*. 2007.
7. Bray, T., et al. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. 2008; Available from: http://www.w3.org/TR/2008/REC-xml-20081126/.
8. Schneider, J., et al. *Efficient XML Interchange (EXI) Format 1.0 (Second Edition)*. 2014; Available from: http://www.w3.org/TR/2014/REC-exi-20140211/.
9. *DEFLATE Compressed Data Format Specification version 1.3*, in *RFC 1951*. 1996. p. 1-17.
10. Ochiai, H. *Power data management on the Internet space: Green ICT projects in Japan*. in *Communications Conference (COLCOM), 2012 IEEE Colombian*. 2012.
11. Foundation, B.o. *BeagleBone Black*. 2013; Available from: http://beagleboard.org/Products/BeagleBone+Black.
12. Texas Instruments. *CC-6LoWPAN*. 2013; Available from: http://processors.wiki.ti.com/index.php/CC-6LoWPAN.
13. STMICROELECTRONICS. *STPM01 PROGRAMMABLE SINGLE PHASE ENERGY METERING IC WITH TAMPER DETECTIO*. 2013; Available from: http://www.st.com/web/en/catalog/sense_power/FM1963/SC397/SS1214/PF81930.
14. พงษ์พจน์ ชัยบุญเรือง, et al. *Small Buildings Energy Management System Based on IEEE1888 Standard with Data Compression*. in *ECTI-CON*. 2014. Sima Thani Hotel , Nakhon Ratchasima, Thailand.

15. predic8 GmbH. *Membrane SOAP/HTTP Monitor and Proxy*. 2013; Available from: http://www.membrane-soa.org/.
16. Fujitsu Laboratories of America. *OpenEXI*. Available from: http://openexi.sourceforge.net/.
17. The Apache Software Foundation. *Apache Axis2*. 2011; Available from: http://axis.apache.org/axis2/java/core/.
18. Jayasinghe, D., *Quickstart Apache Axis2*. 2008. 180.
19. Jayasinghe, D. and A. Azeez, *Apache Axis2 Web Services*. 2 ed. 2011. 308.
20. Sensinode Ltd. *Texas Instrument - CC-6LoWPAN kit*. Available from: http://www.sensinode.com/EN/products/software/ti-cc-6lowpan-kit.html.
21. OCHIAI, H., Y. DOI, and H. ESAKI. *CA-EXI: Context-Aware Efficient XML Interchange for IEEE1888 Message Compression*. in *IEICE TRANSACTIONS on Communications (Japanese Edition)*. 2013.
22. พงษ์พจน์ ชัยบุญเรือง, et al. เทคนิคการบีบอัดสำหรับข้อมูลที่เป็นไปตามมาตรฐาน *IEEE1888*. in การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ *36 (EECON 36)*. 2556. FELIX RIVER KWAI RESORT.
23. *Electricity metering equipment (a.c.) - Particular requirements - Part 21: Static meters for active energy (classes 1 and 2)*, in *IEC 62053-21*. 2003. p. 1-45.

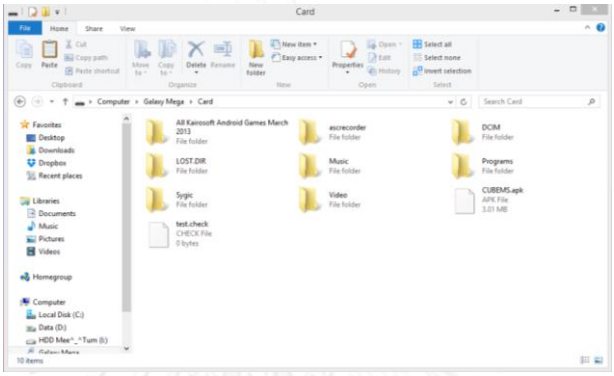
ภาคผนวก

ภาคผนวก ก

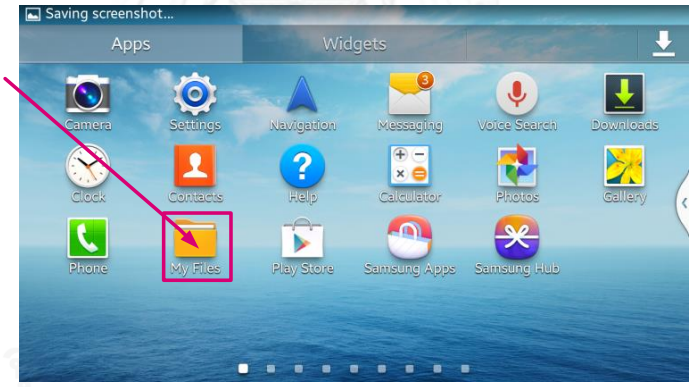
คู่มือการใช้งานโปรแกรมบริหารจัดการพลังงานไฟฟ้าภายในอาคาร

<u>วิธีการลงโปรแกรม</u>

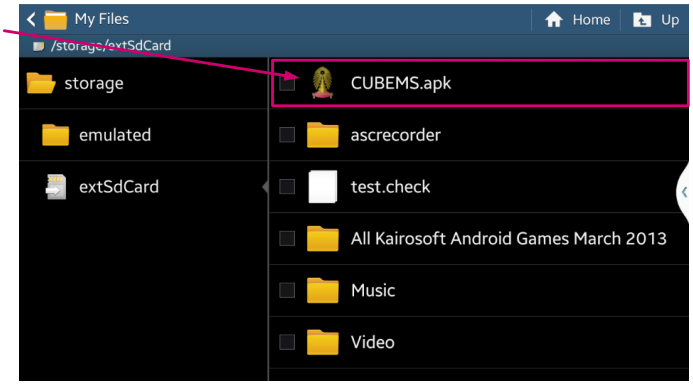
1. นำไฟล์ CUBEMS.apk ที่ดาวน์โหลดมาไว้ใน sd card ของ tablet หรือ smart phone



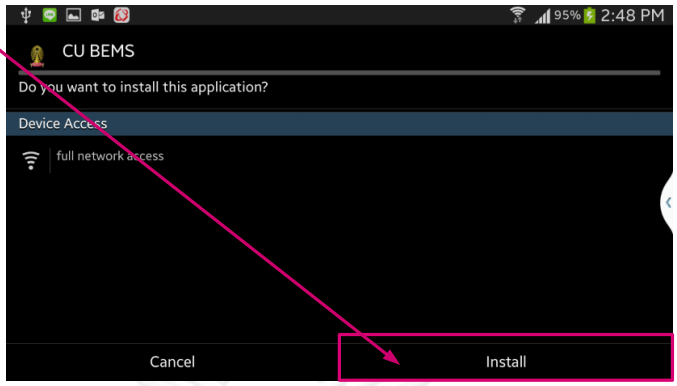
2. เปิดแอพพลิเคชั่น My files



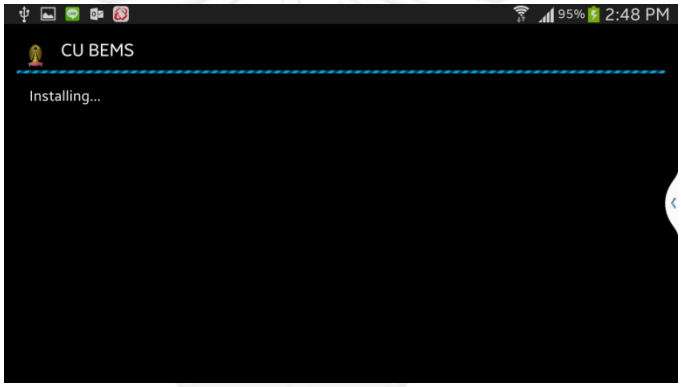
3. เลือกไฟล์แอพพลิเคชั่น CUBEMS.apk ตาม Directory ที่ได้วางไว้ในข้อ 1



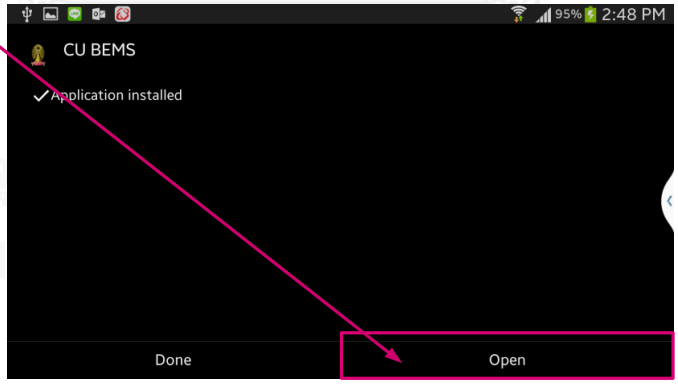
4. เลือก Install



5. ระบบจะทำการติดตั้งแอพพลิเคชั่น กรุณารอจนกว่าจะเสร็จ

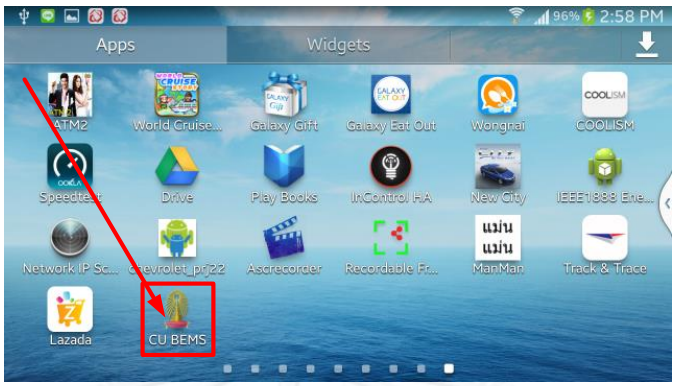


6. สามารถเริ่มการทำงานของแอพพลิเคชั่นได้ โดยเลือก Open

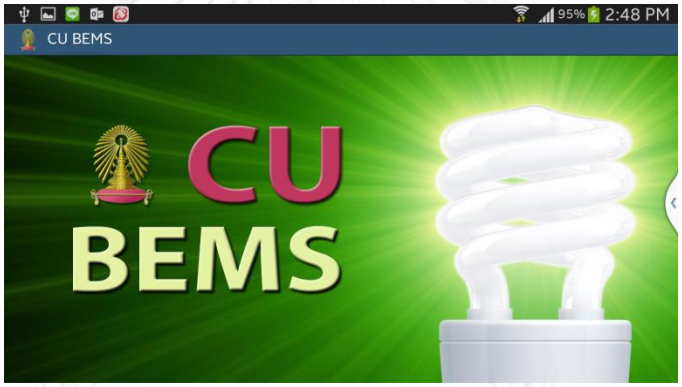


<u>วิธีการตั้งค่า เกทเวย สตอเรจ และ PointID</u>

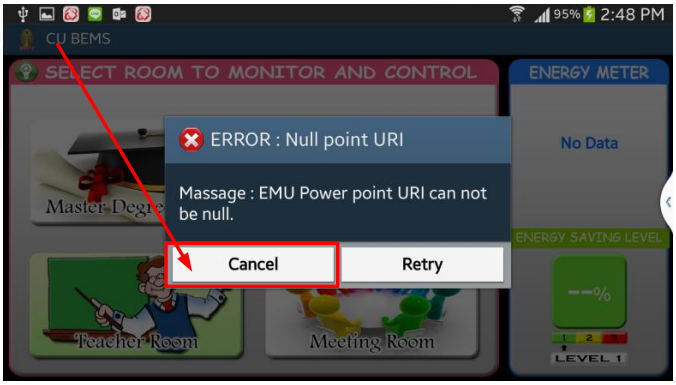
1. เปิดโปรแกรม CU BEMS



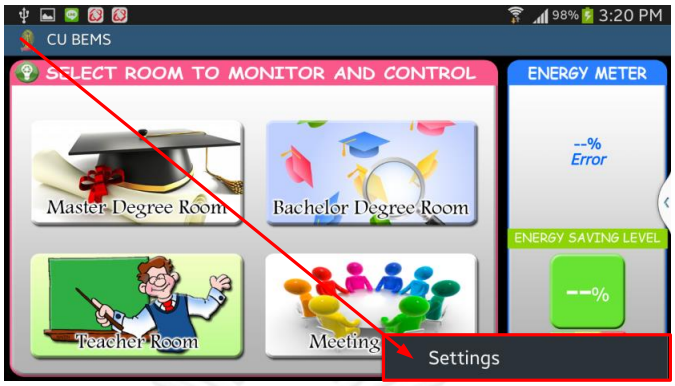
2. จะปรากฏหน้าต้อนรับดังรูป



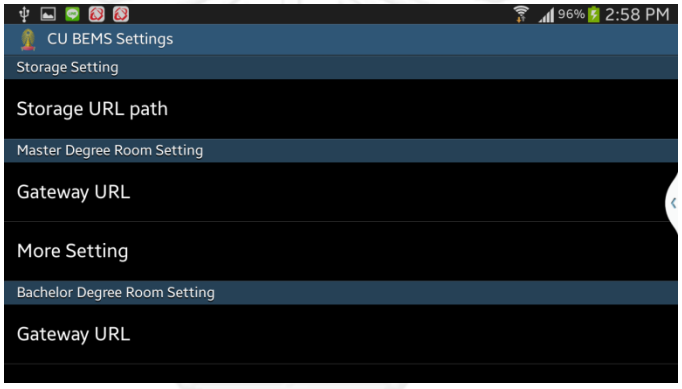
3. เมื่อเข้าโปรแกรมมาแล้วจะพบกับกล่องแจ้งเตือนว่า PointID ที่เก็บค่าพลังงานไฟฟ้าไม่สามารถที่จะทิ้งให้ว่างเปล่าได้ จากนั้นให้เลือก Cancel เพื่อรับทราบ



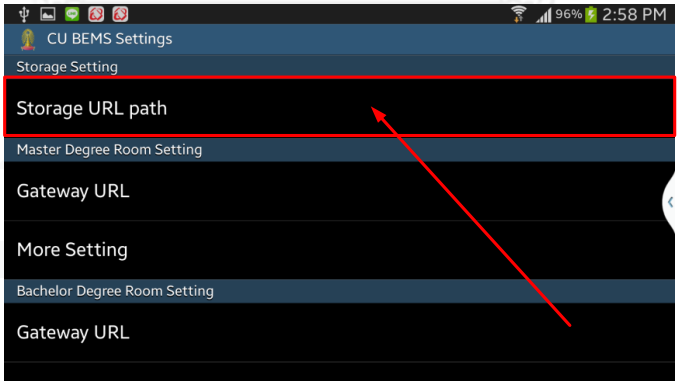
4. กดปุ่มตั้งค่าที่ตัวเครื่อง จะปรากฏเมนูทางด้านขวาล่างของจอ จากนั้นเลือก Setting



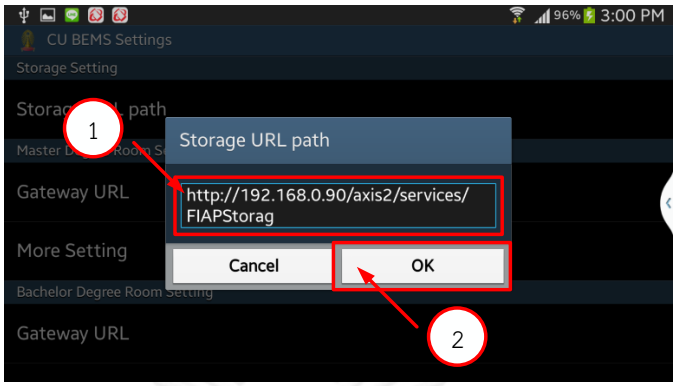
5. จะประกฏหน้า CU BEMS Settings ที่สามารถทำการตั้งค่าต่างๆของโปรแกรมได้ทั้งหมด



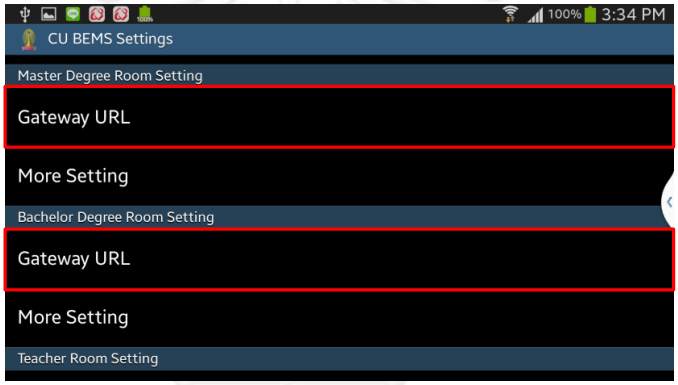
6. การตั้งค่า Path ของสตอเรจ ให้เลือกที่แถบ Storage URL path



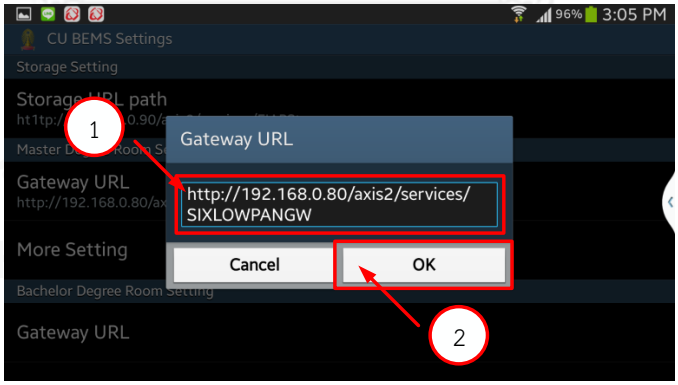
7. จะมีกล่องข้อความปรากฏขึ้นมาให้ใส่ Path ของสตอเรจ แล้วกด OK



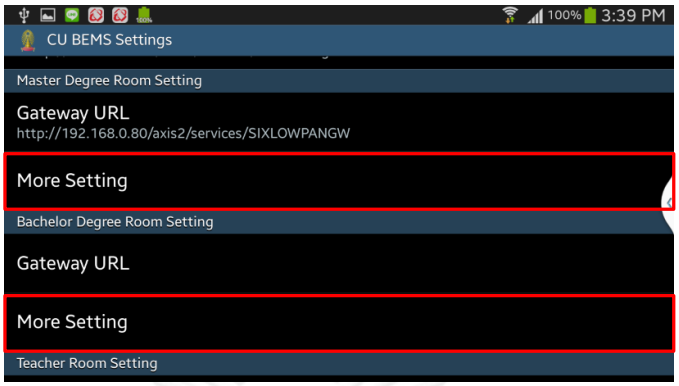
8. การตั้งค่า Path ของเกทเวยให้เลือกที่แถบ Gateway URL ของห้องที่ต้องการจะตั้งค่า



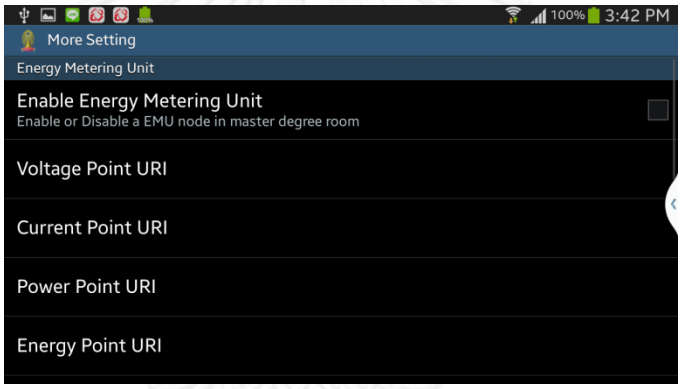
9. จะมีกล่องข้อความปรากฏขึ้นมาให้ใส่ Path ของเกทเวยในห้องนั้นๆ แล้วกด OK



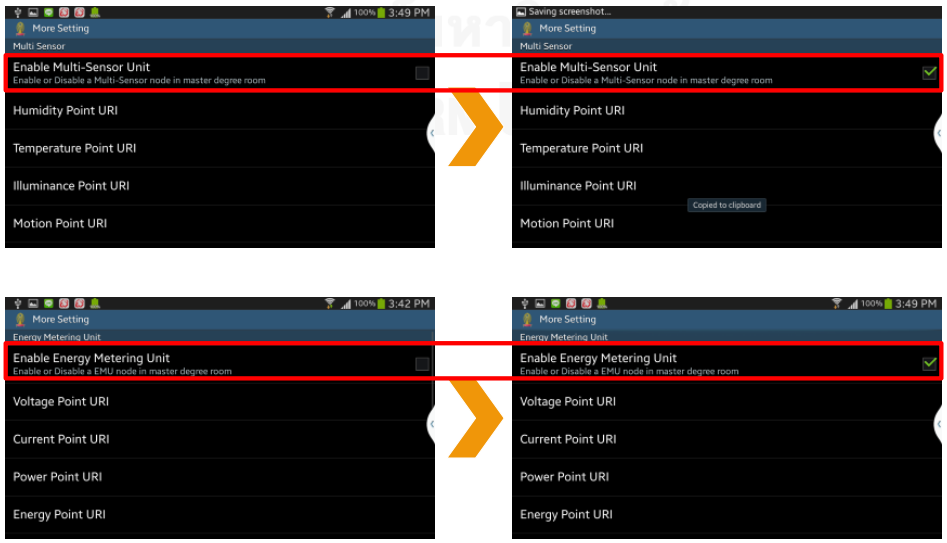
10. การตั้งค่า PointID ของเกทเวยให้เลือกที่แถบ More Setting ของห้องที่ต้องการจะตั้งค่า



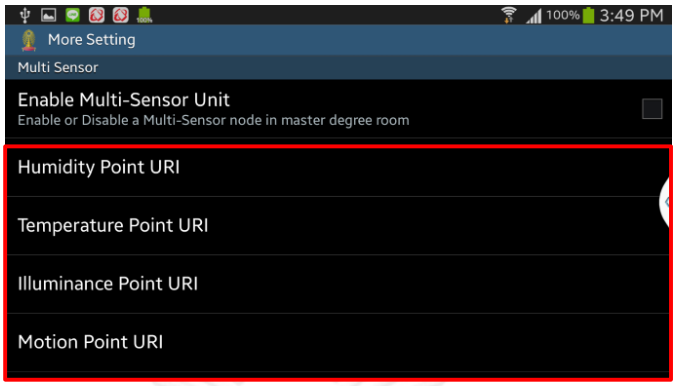
11. จะปรากฏหน้า More Setting ที่ใช้ในการตั้งค่า PointID ทั้งหมดภายในห้องนั้นๆ และสามารถทำการ เปิด/ปิด การทำงานของโนทวัดพลังงาน หรือโนทวัดสภาพแวดล้อมได้



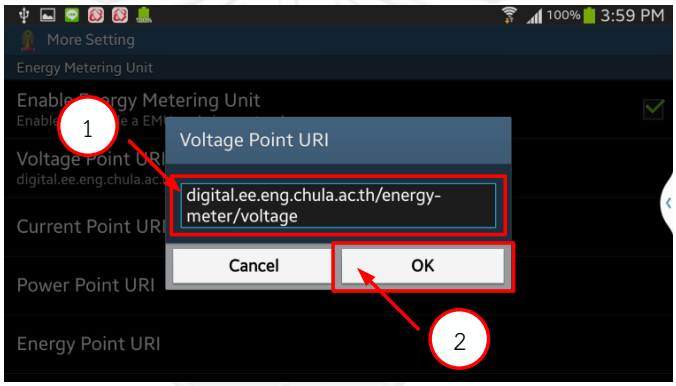
12. ผู้ใช้งานสามารถเลือกที่แถบ Enable Energy Metering Unit หรือ Enable Multi-Sensor Unit เพื่อ เปิด/ปิด การทำงาน ซึ่งเมื่อเปิดจะมีสัญลักษณ์เช็คถูกอยู่ด้านหลังของแถบ



13. การตั้งค่า Point ID สามารถทำได้โดยเลือกแถบของ Point ID URI ที่ต้องการ

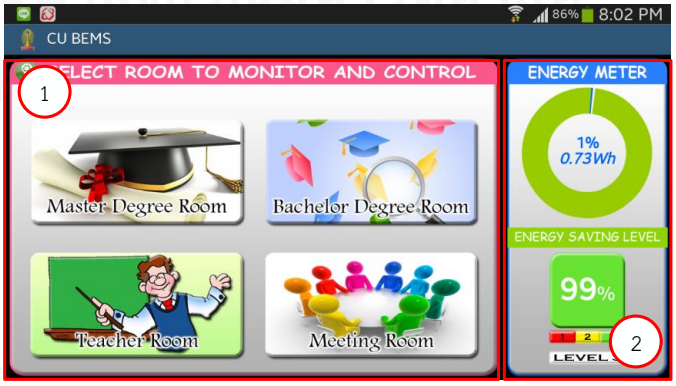


14. จะมีกล่องข้อความปรากฏขึ้นมาให้ใส่ PointID ของข้อมูลในแถบนั้นๆลงไป แล้วกด OK

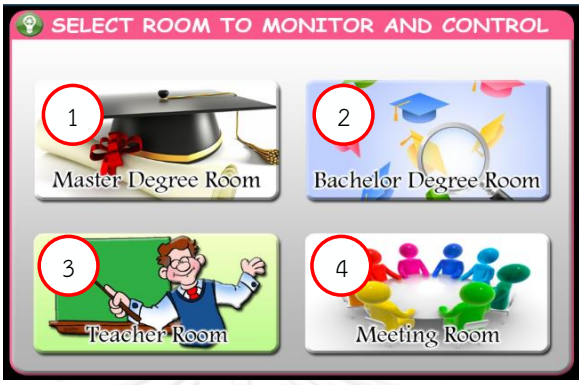


การใช้งานหน้า CU BEMS

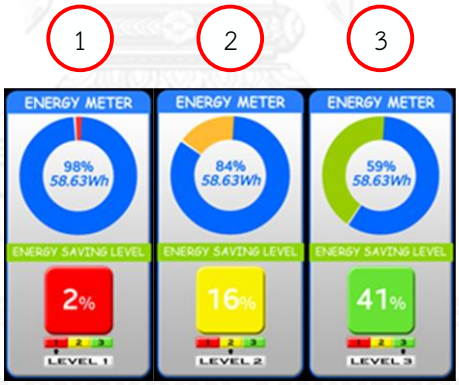
1. หน้า CU BEMS เป็นหน้าแรกของโปรแกรมที่เมื่อเริ่มการทำงานจะเข้ามาหน้าที่ก่อนเสมอทุกครั้ง ซึ่งประกอยไปด้วย 2 ส่วน คือ 1. ส่วนเลือกห้องที่ต้องการจะควบคุม 2. ส่วนแสดงผลการพลังงานไฟฟ้าทั้งหมดภายในอาคาร



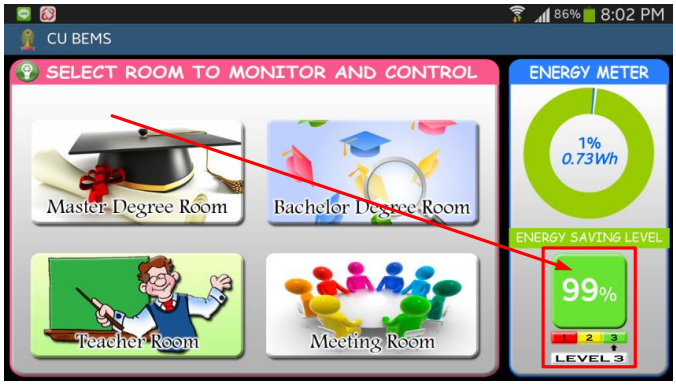
2. ส่วนเลือกห้องนั้นจะประกอบไปด้วยปุ่มจำนวน 4 ปุ่ม โดยแต่ละปุ่มจะเชื่อมโยงการทำงานไปยัง Room Activity ประกอบไปด้วย 4 ห้อง คือ 1. ห้องนิสิต ป.โท 2. ห้องนักศึกษา ป.ตรี 3. ห้องพักอาจารย์ และ 4. ห้องประชุม



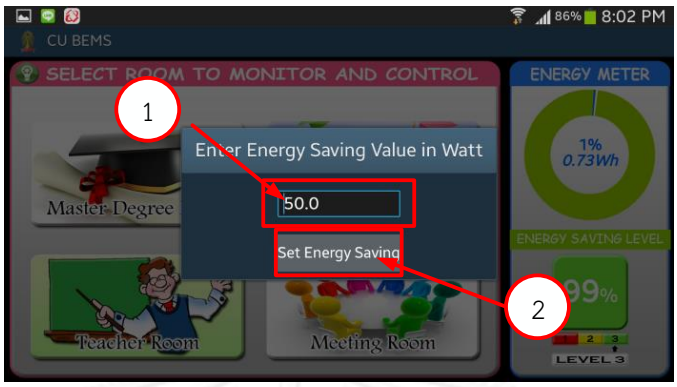
3. ส่วนแสดงผลพลังงานไฟฟ้าจะประกอบไปด้วย 2 ส่วน คือ ด้านบนจะเป็นมิเตอร์แสดงการใช้พลังงานไฟฟ้า และ ด้านล่างแสดงระดับการใช้พลังงานไฟฟ้า ซึ่งประกอบไปด้วย 3 ระดับ คือ 1. ระดับวิกฤตมีปริมาณพลังงานไฟฟ้าคงเหลือน้อยกว่า 10% แถบแสดงระดับพลังงานจะปรากฏสีแดง 2. ระดับปานกลางมีปริมาณไฟฟ้าคงเหลืออยู่ในช่วง 10%-20% แถบแสดงระดับพลังงานจะปรากฏสีเหลือง) 3. ระดับปกติมีปริมาณไฟฟ้าคงเหลือมากกว่า 20% แถบแสดงระดับพลังงานจะปรากฏสีเขียว



4. การตั้งค่าขอบเขตพลังงานสามารถทำได้โดยกดที่ตัวเลขแสดงระดับพลังงาน

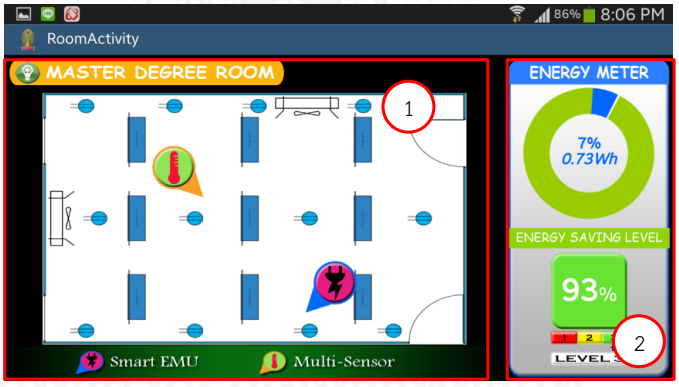


5. จะมีกล่องข้อความปรากฏขึ้นมาให้ใส่ปริมาณของพลังงานสูงสุดในหน่วย Wh จากนั้นกดปุ่ม Set Energy Saving เพื่อบันทึกข้อมูล

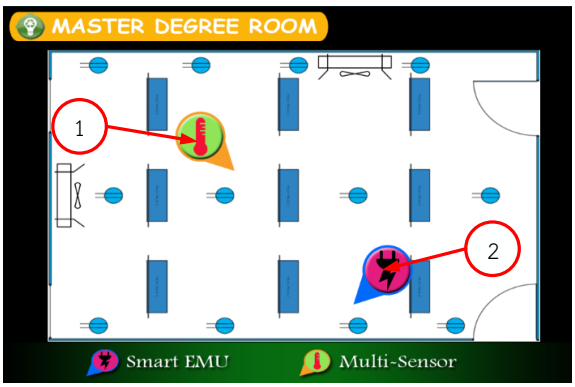


<u>การใช้งานหน้า Room Activity</u>

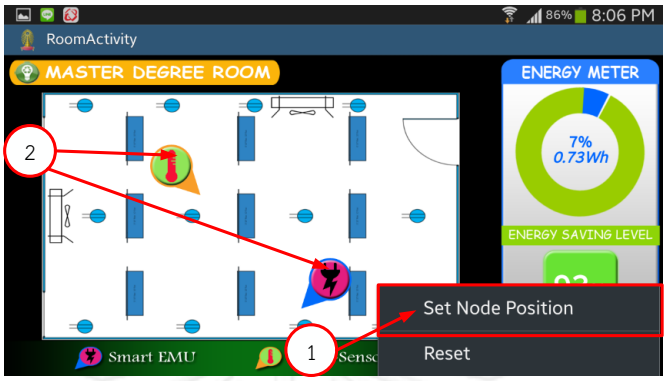
1. ในหน้า Room Activity จะประกอบไปด้วย 2 ส่วน คือ ส่วนแผนผังห้อง และ ส่วนแสดงผลการพลังงานไฟฟ้าทั้งหมดภายในอาคารที่มีการทำงานเช่นเดียวกับในหน้า CU BEMS



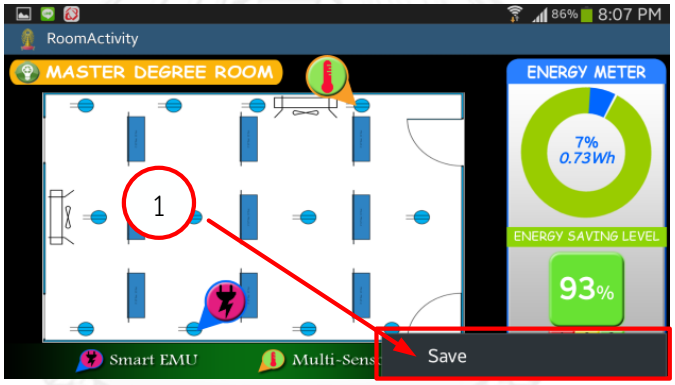
2. ส่วนแผนผังห้องจะมีไอคอนจำนวน 2 ชนิดคือ 1. โนทวัดสภาพแวดล้อม 2. โนทวัดพลังงานไฟฟ้า ซึ่งผู้ใช้งานสามารถดูข้อมูลภายในโนทได้โดยการสัมผัสที่ไอค่อนที่ต้องการ



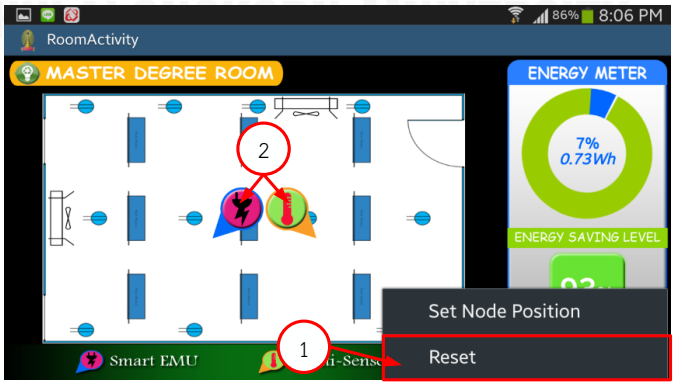
3. การเปลี่ยนตำแหน่งของไอค่อนสามารถทำได้โดยการกดปุ่มตั้งค่าที่ตัวเครื่องจากนั้นจะมีเมนูทางด้านขวาล่างของหน้าจอปรากฏ ให้เลือกที่ Set Node Position จากนั้นผู้ใช้งานสามารถย้ายตำแหน่งของโนทได้ตามต้องการ



4. เมื่อได้ตำแหน่งที่ต้องการแล้วให้ทำการบันทึกตำแหน่งของโนทโดยกดปุ่มตั้งค่าจากนั้นจะมีเมนูทางด้านขวาล่างของหน้าจอปรากฏ ให้เลือกที่ Save

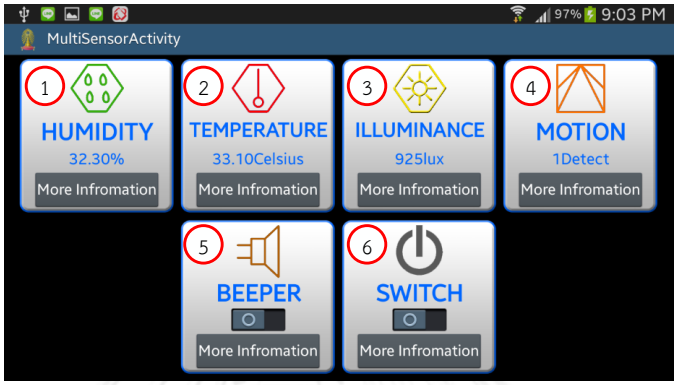


5. ผู้ใช้งานสามารถทำการ Reset ตำแหน่งของโนทได้โดยการกดปุ่มตั้งค่าจากนั้นจะมีเมนูทางด้านขวาล่างของหน้าจอปรากฏแล้วเลือก Reset จากนั้นโนททั้งสองจะย้ายตำแหน่งไปตรงกลางของแผนผังห้องโดยอัตโนมัติ

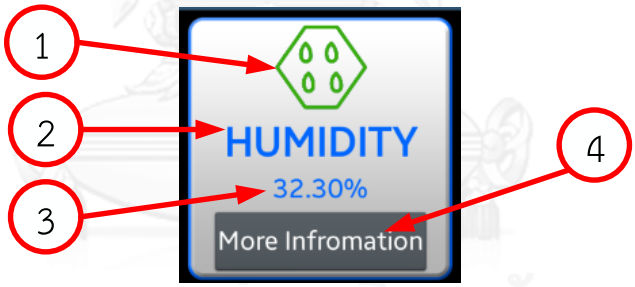


การใช้งานหน้า Multi Sensor Activity

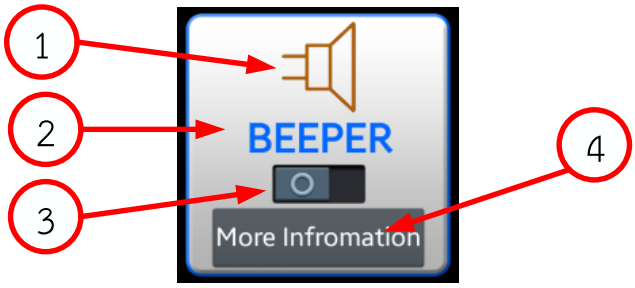
1. เมื่อทำการเลือกโหนดวัดสภาพแวดล้อมในหน้า Room Activity จะทำการเชื่อมโยงการทำงานมายังหน้านี้ โดยภายในจะประกอบไปด้วยข้อมูลจากเซ็นเซอร์ และ Actuator จำนวน 6 ข้อมูล คือ 1. ความชื้น 2. อุณหภูมิ 3. แสงสว่าง 4. ความเคลื่อนไหว 5. Buzzer และ 6. สวิตซ์



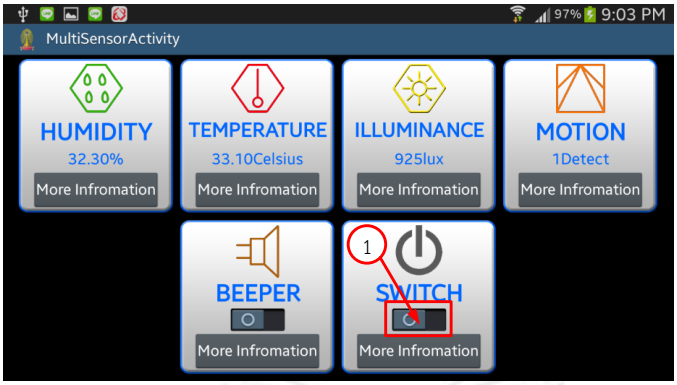
2. ข้อมูลที่ไม่สามารถควบคุมได้จะประกอบไปด้วย 4 ส่วนคือ 1. ไอค่อนของข้อมูล 2. ชื่อของข้อมูล 3. ค่าล่าสุดของข้อมูล 4. ปุ่มกดเมื่อต้องการข้อมูลเพิ่มเติมซึ่งจะเชื่อมโยงการทำงานไปยังหน้า Sensor Summary Activity



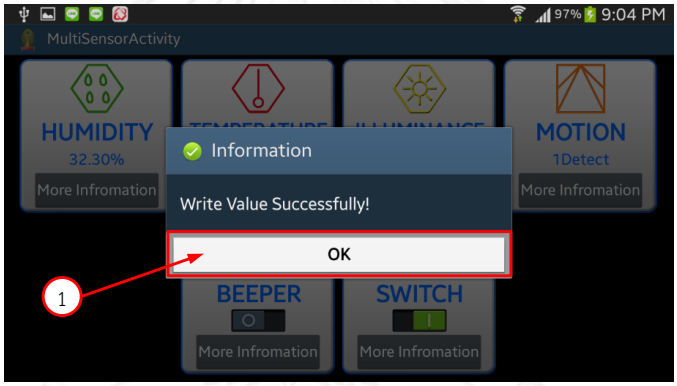
3. ข้อมูลที่สามารถควบคุมได้จะประกอบไปด้วย 4 ส่วนคือ 1. ไอค่อนของข้อมูล 2. ชื่อของข้อมูล 3. สวิตซ์ควบคุมการ เปิด-ปิด 4. ปุ่มกดเมื่อต้องการข้อมูลเพิ่มเติมซึ่งจะเชื่อมโยงการทำงานไปยังหน้า Sensor Summary Activity



4. การควบคุมการทำงานสามารถทำได้โดยกดสวิตซ์ในกรอบของข้อมูลที่ต้องการ

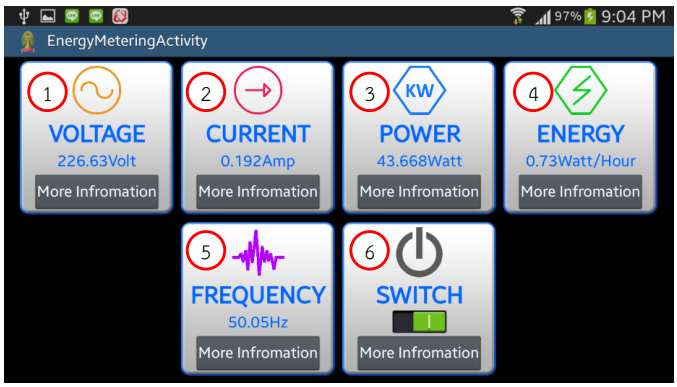


5. จากนั้นจะปรากฏกล่องข้อความที่แสดงผลการควบคุมการทำงาน ให้เลือก OK เพื่อรับทราบ



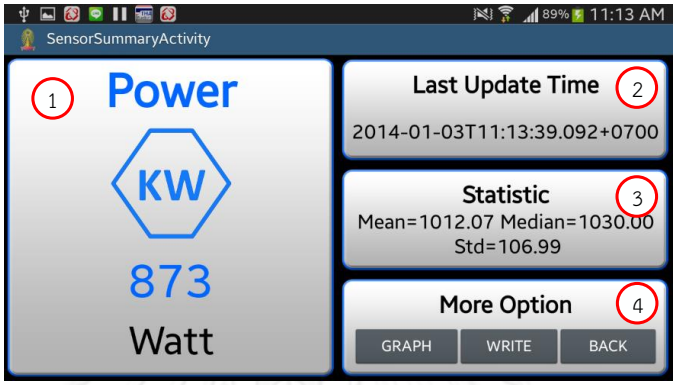
การใช้งานหน้า Energy Metering Activity

1. เมื่อทำการเลือกโหนดวัดพลังงานในหน้า Room Activity จะทำการเชื่อมโยงการทำงานมายังหน้านี้ โดยภายในจะประกอบไปด้วยข้อมูลจากเซ็นเซอร์ และ Actuator จำนวน 6 ข้อมูล คือ 1. แรงดันไฟฟ้า 2. กระแสไฟฟ้า 3. กำลังไฟฟ้า 4. พลังงานไฟฟ้า 5. ความถี่ไฟฟ้า และ 6. สวิตซ์ ที่มีการทำงานเช่นเดียวกับ Multi Sensor Activity



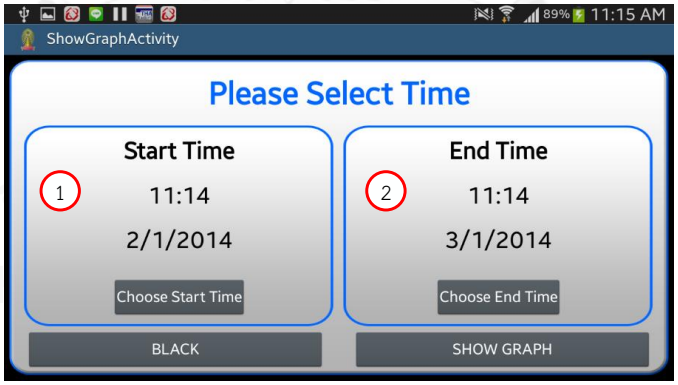
การใช้งานหน้า Sensor Summary Activity

1. ภายในหน้านี้จะแสดงรายละเอียดของข้อมูลที่ได้ทำการเลือก โดยประกอบไปด้วย 4 ส่วนย่อย คือ 1. ชื่อข้อมูล และ ค่าล่าสุด 2. เวลาที่ข้อมูลมีการเปลี่ยนแปลงล่าสุด 3. ค่าสถิติของข้อมูลนั้นในช่วง 24 ชั่วโมงที่ผ่านมา 4. ปุ่ม GRAPH ที่เชื่อมโยงการทำงานไปยังหน้า Show Graph Activity และ ปุ่ม WRITE ที่เชื่อมโยงการทำงานไปยังหน้า Write Value Activity

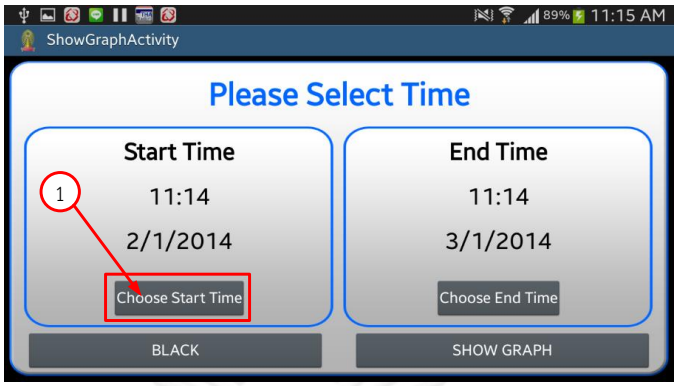


การใช้งานหน้า Show Graph Activity

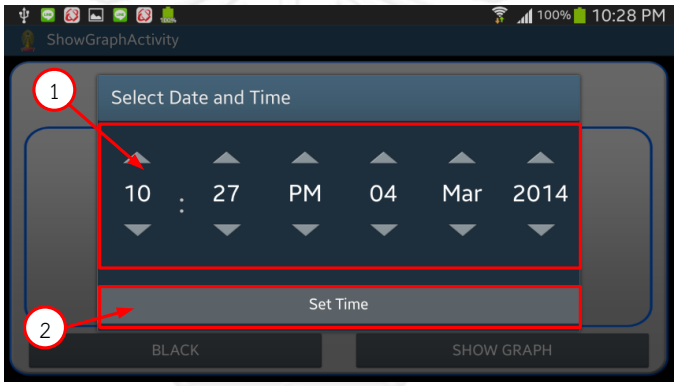
1. หน้า Show Graph Activity จะใช้ในการกำหนดช่วงเวลาของข้อมูลที่ต้องการจะพล็อตกราฟโดยสามารถกำหนดช่วงเวลาที่สุดได้ 1 สัปดาห์ ซึ่งประกอบไปด้วย 2 ส่วน คือ 1. ส่วนกำหนดเวลาเริ่มต้น 2.ส่วนกำหนดเวลาสิ้นสุด



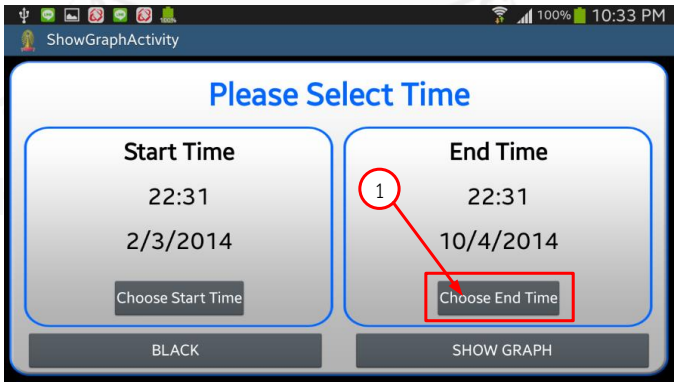
2. การกำหนดเวลาเริ่มต้นสามารถทำได้โดยกดปุ่ม Choose Start Time



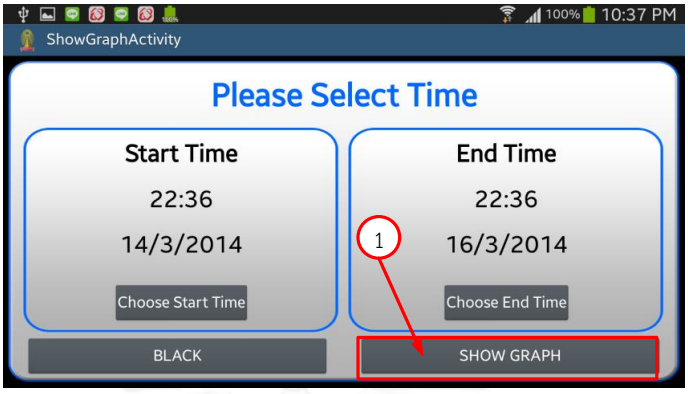
3. เลือกเวลาเริ่มต้นที่ต้องการ จากนั้นกดปุ่ม Set Time



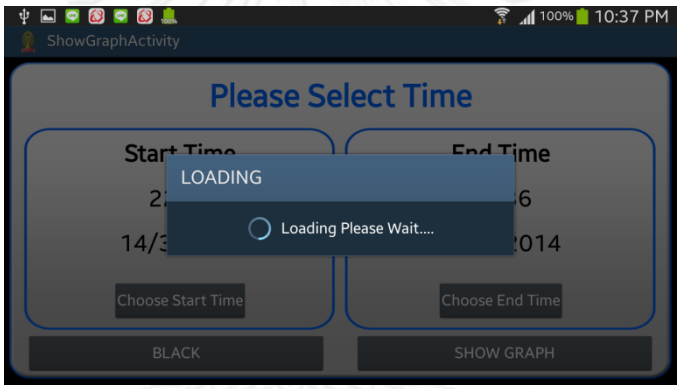
4. การกำหนดเวลาสิ้นสุดสามารถทำได้โดยกดปุ่ม Choose End Time แล้วทำการเลือกเวลาเช่นเดียวกับในข้อที่ 3



5. เมื่อได้ช่วงเวลาที่ต้องการแล้วให้กดปุ่ม SHOW GRAPH เพื่อทำการดึงข้อมูลจากสตอเรจมาพล็อตกราฟ



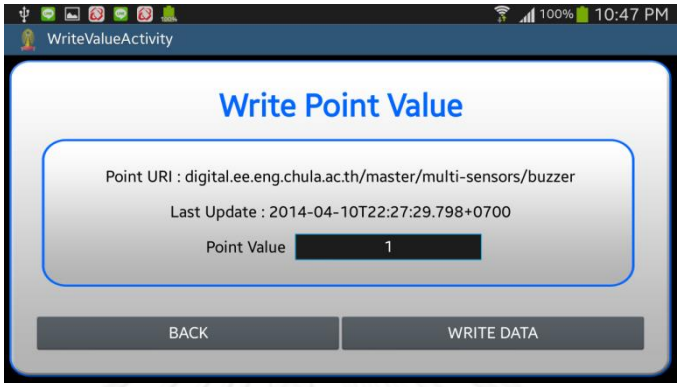
6. โปรแกรมจะทำการประมวลผลข้อมูลเพื่อนำมาสร้างกราฟ รอจนกว่าหน้าต่างนี้จะหายไป



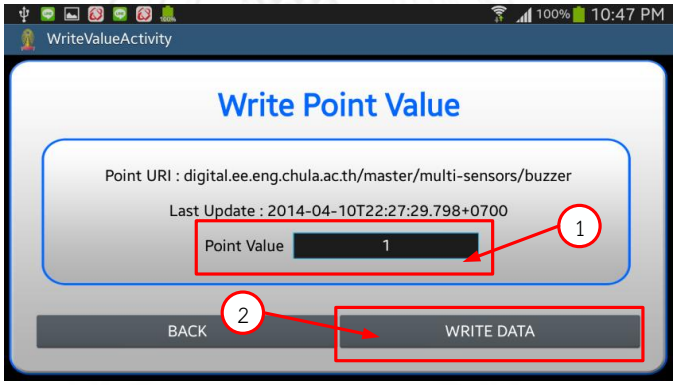
7. ได้กราฟของข้อมูลในช่วงเวลาดังกล่าวปรากฏขึ้น

การใช้งานหน้า Write Value Activity

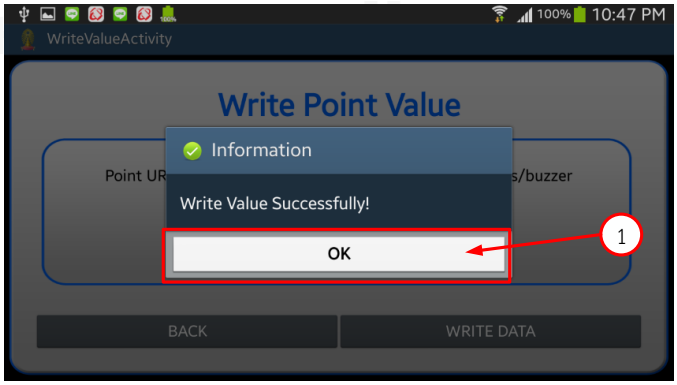
1. หน้า Write Value Activity ใช้ในการเขียนข้อมูลไปยัง PointID ที่ต้องการ ซึ่งเมื่อเปิดหน้านี้ขึ้นมาครั้งแรกจะแสดง ชื่อของ PointID เวลาล่าสุดที่มีการเปลี่ยนแปลงของข้อมูล และ ค่าล่าสุด



2. ผู้ใช้งานสามารถเขียนข้อมูล โดยป้อนข้อมูลลงในช่อง Point Value จากนั้นกดปุ่ม WRITE DATA เพื่อทำการเขียนข้อมูล



3. จากนั้นจะปรากฏกล่องข้อความที่แสดงผลการเขียนข้อมูล ให้เลือก OK เพื่อรับทราบ



ประวัติผู้เขียนวิทยานิพนธ์

นายพงษ์พจน์ ชัยบุญเรือง เกิดเมื่อวันที่ 30 มกราคม พ.ศ. 2532 ที่จังหวัดเชียงใหม่ สำเร็จการศึกษาหลักสูตรหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาเทคโนโลยีวิศวกรรมอิเล็กทรอนิกส์ จากภาควิชาเทคโนโลยีวิศวกรรมอิเล็กทรอนิกส์ วิทยาลัยเทคโนโลยีอุตสาหกรรมมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ในปีการศึกษา 2554 ต่อมาได้เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า แขนงวิชาการออกแบบและประยุกต์วงจรรวม ที่คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยในปีการศึกษา 2555