การพัฒนาการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite สำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคาร

นายชญานนท์ แสงอำไพ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2559

DEVELOPMENT OF REAL-TIME INTERWORKING BETWEEN IEEE1888 AND ECHONET LITE STANDARDS FOR BUILDING ENERGY MANAGEMENT SYSTEM

MR. CHAYANON SANGUMPAI

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 2016

หัวข้อวิทยานิพนธ์	การพัฒนาการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite สำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคาร
โดย	นายชญานนท์ แสงอำไพ
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.เชาวน์ดิศ อัศวกุล

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รองศาสตราจารย์ ดร.แนบบุญ หุนเจริญ)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.เชาวน์ดิศ อัศวกุล)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ชัยเชษฐ์ สายวิจิตร)

.....กรรมการ
(อาจารย์ ดร.ภาณุวัฒน์ จันทร์ภักดี)

.....กรรมการภายนอกมหาวิทยาลัย
(รองศาสตราจารย์ ดร.ภูมิพัฒ แสงอุดมเลิศ)

ชญานนท์ แสงอำไพ : การพัฒนาการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite สำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคาร (DEVELOPMENT OF REAL-TIME INTERWORKING BETWEEN IEEE1888 AND ECHONET LITE STANDARDS FOR BUILDING ENERGY MANAGEMENT SYSTEM) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ. ดร. เชาวน์ดิศ อัศวกุล, 81 หน้า.

วิทยานิพนธ์นี้นำเสนอการพัฒนาการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite สำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคาร มาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ถูกพัฒนามาในรูปแบบมาตรฐานแบบเปิด จึงรองรับการพัฒนาโปรแกรมประยุกต์และอุปกรณ์ชนิดต่าง ๆ ได้ มาตรฐาน IEEE1888 ใช้รูปแบบข้อความ XML และเหมาะสำหรับการสื่อสารระหว่างอุปกรณ์ที่มีความหลากหลาย มาตรฐาน ECHONET Lite ได้การยอมรับอย่างกว้างขวางในประเทศญี่ปุ่นพร้อมกับการสนับสนุนที่เกิดขึ้นใหม่จากผู้ผลิตหลายราย เช่น อุปกรณ์เครื่องปรับอากาศ เป็นต้น การประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite เริ่มจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งการร้องขอการแจ้งเตือน TRAP ตามมาตรฐาน IEEE1888 ไปยังหน่วยเก็บข้อมูล BEMS เมื่อข้อมูลเกิดการเปลี่ยนแปลง การตอบกลับ TRAP callback จะถูกส่งต่อไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ทันทีพร้อมกับค่าของข้อมูลที่เปลี่ยนแปลง ต่อมาจึงส่งการร้องขอ WRITE ตามมาตรฐาน ECHONET Lite ไปยังโนด ECHONET Lite เป้าหมาย การประสานข้อมูลแบบทันทีนี้เกิดข้อความขนาดรวมกันทั้งหมด 2,410 ไบต์ การทดสอบนี้ถูกพิจารณาในที่ซึ่งตัวรับรู้ IEEE1888 ชนิด PIR กระตุ้นอุปกรณ์เครื่องปรับอากาศ ECHONET Lite ที่พร้อมใช้งานภายในห้องปฏิบัติการคอมพิวเตอร์คลัสเตอร์ และการประสานข้อมูลแบบทันทีในทิศทางกลับกันจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 เริ่มจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งการร้องขอ NOTIFY ไปยังโนด ECHONET Lite ตัวอย่างเช่น เพื่อตรวจจับสถานะของอุปกรณ์เครื่องปรับอากาศ โนด ECHONET Lite จะส่งข้อความตอบกลับพร้อมกับค่าของสถานะที่เปลี่ยนแปลงไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ จากนั้นจึงส่งข้อมูลสถานะไปเก็บยังหน่วยเก็บข้อมูล BEMS ด้วยโพรโทคอล WRITE ตามมาตรฐาน IEEE1888 จากการทดสอบภายในห้องคอมพิวเตอร์คลัสเตอร์ การประสานข้อมูลแบบทันทีนี้เกิดข้อความขนาดรวมกันทั้งหมด 860 ไบต์ นอกจากการประเมินเพย์โหลด การประสานข้อมูลแบบทันทีระหว่างสองมาตรฐานถูกทดสอบในแง่ของสมรรถนะการหน่วงเวลา เมื่อเพิ่มความถี่ในการร้องขอ WRITE ต่อนาที จากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ไปยังหน่วยเก็บข้อมูลสำหรับการเปลี่ยนแปลงค่าของข้อมูล ผลลัพธ์แสดงให้เห็นว่าเวลาที่ใช้ตลอดการประสานข้อมูลแปรผันจาก 178 ถึง 424 มิลลิวินาที เมื่อความถี่ของการส่ง WRITE เพิ่มขึ้นจาก 1 ถึง 50 ครั้งต่อนาที ในตอนท้ายได้มีการตรวจสอบการใช้งานโหมดทุกโหมดของตัวควบคุม ECHONET Lite ซึ่งถูกติดตั้งไว้สำหรับอุปกรณ์เครื่องปรับอากาศภายในห้องปฏิบัติการ จากผลการทดสอบโหมดซึ่งสามารถใช้งานได้ในขณะนี้ คือ สถานะการเปิดปิด การตั้งค่าโหมดการใช้งาน การตั้งค่าอุณหภูมิ ค่าอุณหภูมิที่วัดได้ภายในห้อง การตั้งค่าระดับพัดลม การตั้งค่าส่ายลมอัตโนมัติของพัดลม และการตั้งค่าทิศทางของพัดลมตามแนวแกนตั้ง

ภาควิชา วิศวกรรมไฟฟ้า **ลายมือชื่อนิสิต**.....
สาขาวิชา วิศวกรรมไฟฟ้า **ลายมือชื่อ อ.ที่ปรึกษาหลัก**.....
ปีการศึกษา 2559

5770143621 : MAJOR ELECTRICAL ENGINEERING
KEYWORDS: INTERWORKING/ IEEE1888/ ECHONET LITE/ BUILDING ENERGY MANAGEMENT.

CHAYANON SANGUMPAI : DEVELOPMENT OF REAL-TIME INTERWORKING BETWEEN IEEE1888 AND ECHONET LITE STANDARDS FOR BUILDING ENERGY MANAGEMENT SYSTEM. ADVISOR: ASSOC. PROF. CHAODIT ASWAKUL, Ph.D., 81 pp.

This thesis presents the development of real-time interworking between IEEE1888 and ECHONET Lite standards for a building energy management system (BEMS). Both IEEE1888 and ECHONET Lite have been developed as open standards; hence, a support for application and equipment development. IEEE1888 standard uses an XML message format and is suitable for a wide range of machine-to-machine communications. ECHONET Lite standard is widely accepted in Japan with emerging supports by equipment manufacturers such as for air conditioners. The real-time data synchronization from IEEE1888 to ECHONET Lite standards starts with the interworking proxy gateway sending an IEEE1888 TRAP alert query to the BEMS storage. When the trapped data changes, a TRAP callback can then be forwarded immediately with the changed data value to the interworking proxy gateway, which subsequently sends an ECHONET Lite WRITE request to a target ECHONET Lite node. This real-time synchronization generates messages with the total size of 2,410 bytes in the considered testbed where a PIR sensor from IEEE1888 triggers an ECHONET Lite compliant air conditioner inside the laboratory's computer cluster room. In the reversed data synchronization direction from ECHONET Lite to IEEE1888 standards, the process starts with the interworking proxy gateway sending a NOTIFY request to ECHONET Lite node e.g. to detect the air conditioner status. With a changed value of the status, the ECHONET Lite node sends a response message to the interworking proxy gateway, which subsequently pushes that status data into the BEMS storage by using the IEEE1888 WRITE protocol. In the testbed at the computer cluster room, this real-time synchronization from ECHONET Lite to IEEE1888 standards generates messages with the total size of 860 bytes. In addition to payload evaluation, the real-time data synchronization between the two standards has been tested in terms of time delay performance. When increasing the frequency of the WRITE queries per minute from the interworking proxy gateway to the storage for changing data values and causing subsequent synchronization messages, the results show that the round-trip time delay varies from 178 to 424 milliseconds when the WRITE frequency is increased from 1 to 50 times per minute. Finally, all modes of the ECHONET Lite controller installed for the in-laboratory air conditioner have been verified. From the testing results, the currently functional ECHONET Lite modes are the operation status, operation mode setting, set temperature value, measured value of room temperature, air flow rate setting, automatic swing of air flow setting and air flow direction (vertical) setting.

Department	**:** Electrical Engineering	**Student's Signature**	.....
Field of Study	**:** Electrical Engineering	**Advisor's Signature**	.....
Academic Year	**:** 2016		

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยความช่วยเหลือจากอาจารย์ที่ปรึกษาวิทยานิพนธ์ รองศาสตราจารย์ ดร.เชาวน์ดิศ อัศวกุล ผู้ที่ช่วยถ่ายทอดความรู้ทั้งทางด้านวิจัยและด้านวิชาการ ให้คำปรึกษาและคำแนะนำสำหรับเป็นแนวทางในงานวิจัย รวมถึงช่วยตรวจทานวิทยานิพนธ์ฉบับนี้อย่างดีเสมอมา ผู้วิจัยจึงขอกราบขอบพระคุณมา ณ ที่นี้ และขอขอบพระคุณ รองศาสตราจารย์ ดร.แนบบุญ หุนเจริญ ประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.ชัยเชษฐ์ สายวิจิตร อาจารย์ ดร. ภาณุวัฒน์ จันทร์ภักดี และรองศาสตราจารย์ ดร.ภูมิพัฒ แสงอุดมเลิศ กรรมการสอบวิทยานิพนธ์ ที่สละเวลาตรวจทานและให้คำแนะนำสำหรับวิทยานิพนธ์ฉบับนี้เพื่อความสมบูรณ์ยิ่งขึ้น

งานวิจัยนี้ได้รับการสนับสนุนจากหน่วยปฏิบัติการวิจัยโครงข่ายไร้สาย และอินเทอร์เน็ตอนาคต (Wireless Network and Future Internet Research Unit) กองทุนรัชดาภิเษกสมโภช จุฬาลงกรณ์มหาวิทยาลัย

ขอขอบคุณกลุ่มวิจัยโครงข่ายไร้สายและอินเทอร์เน็ตอนาคตภายใต้การดูแลโดย รองศาสตราจารย์ ดร. เชาวน์ดิศ อัศวกุล และผู้ช่วยศาสตราจารย์ ดร. ชัยเชษฐ์ สายวิจิตร ที่จัดกิจกรรมเพื่อส่งเสริมการเรียนรู้ การทำงาน และทักษะเพื่อเป็นผู้วิจัยที่มีประสิทธิภาพที่ดียิ่งขึ้น

ขอขอบคุณโครงการวิจัยและพัฒนาเทคโนโลยีระบบโครงข่ายไฟฟ้าอัจฉริยะเพื่อบริหารจัดการการใช้พลังงานไฟฟ้าของอาคาร จุฬาลงกรณ์มหาวิทยาลัย (CU-BEMS) และโครงการมหาวิทยาลัยสำหรับอินเทอร์เน็ตในอนาคต (Universities for Future Internet, UNIFI) ที่ทำให้ผู้วิจัยมีองค์ความรู้และได้รับคำแนะนำจากคณาจารย์และทีมงานเป็นอย่างดียิ่งซึ่งเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์ฉบับนี้เกิดขึ้นได้

ขอขอบคุณทุนวิจัยจากโครงการ Seeds for The Future โดยบริษัทหัวเว่ยเทคโนโลยี (ประเทศไทย) จำกัด (Huawei Technology, Thailand) สำหรับการสนับสนุนทุนการศึกษาในหลักสูตรมหาบัณฑิต

ขอขอบคุณเพื่อน พี่ น้องนักวิจัย เจ้าหน้าที่ บุคลากร และคณาจารย์ ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่คอยให้คำแนะนำ ความช่วยเหลือเรื่องต่าง ๆ และการสนับสนุนที่ดีเสมอมา

สุดท้ายนี้ขอขอบคุณครอบครัวของผู้วิจัย ซึ่งได้ให้การสนับสนุนและเป็นกำลังใจให้แก่ผู้วิจัยเสมอมาจนสำเร็จการศึกษา

สารบัญ

บทที่ หน้า

บทที่ หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

หน้า

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันเทคโนโลยีการสื่อสารข้อมูลถูกพัฒนาอย่างรวดเร็ว จุดประสงค์เพื่อการเข้าถึงข้อมูลต่าง ๆ ในโครงข่ายการสื่อสารได้อย่างมีประสิทธิภาพ ตลอดจนบทบาทในการเข้าถึงข้อมูลของอุปกรณ์ต่าง ๆ จากผู้ใช้งานผ่านแนวความคิดเมืองสมาร์ต (smart cities) [1] มีมากยิ่งขึ้น ส่งผลทำให้ผู้ใช้งานมีความจำเป็นในการเข้าถึงการให้บริการการใช้งานของอุปกรณ์ที่มีความหลากหลาย และอุปกรณ์ที่ต่างกันต้องสามารถสื่อสารหรือทำงานร่วมกันได้ [2] เพื่อให้สามารถรองรับสิ่งที่คาดว่าจะเกิดขึ้นในโครงข่ายต่อไปในอนาคต หนึ่งในนั้นคือ อินเทอร์เน็ตของสรรพสิ่ง (internet of things, IoT) [3] ซึ่งเป็นการเชื่อมโยงสิ่งต่าง ๆ ไปยังโครงข่ายอินเทอร์เน็ต และมีการสื่อสารเพื่อแลกเปลี่ยนข้อมูลระหว่างสิ่งต่าง ๆ เช่น ตัวรับรู้ (sensor) ตัวกระตุ้น (actuator) หรือ อุปกรณ์เฝ้าสังเกตการณ์ (monitoring device) เป็นต้น โดยคาดว่าอินเทอร์เน็ตของสรรพสิ่งจะได้รับความสนใจอย่างยิ่ง เนื่องจากอินเทอร์เน็ตของสรรพสิ่งมาจากโครงข่ายการสื่อสารข้อมูลที่มีขนาดใหญ่จึงมีการใช้งานหลากหลายมาตรฐาน ทำให้มีความจำเป็นที่จะต้องพึ่งพาอุปกรณ์ที่มีความสามารถในการแลกเปลี่ยนข้อมูลระหว่างมาตรฐานในโครงข่ายอินเทอร์เน็ตของสรรพสิ่ง

ภายใต้แนวความคิดและการริเริ่มโครงข่ายอินเทอร์เน็ตของสรรพสิ่งยังมีความจำเป็นต้องนำเทคโนโลยีการสื่อสารระหว่างอุปกรณ์หรือเรียกว่า การสื่อสารจากเครื่องถึงเครื่อง (machine to machine communication, M2M) [4] มาใช้ในรูปแบบการสื่อสารแบบมีสายหรือไร้สายเพื่อให้สามารถแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์ที่ใช้งานทั้งในมาตรฐานเดียวกันและมาตรฐานที่แตกต่างกัน อีกทั้งภายในโครงข่ายอินเทอร์เน็ตของสรรพสิ่งมีการเชื่อมต่อจากอุปกรณ์ไปยังโครงข่ายอินเทอร์เน็ตทำให้ผู้ใช้งานสามารถเข้าถึงการให้บริการโปรแกรมประยุกต์ได้ [5] จึงง่ายต่อการใช้งาน ตรวจสอบ และรองรับการใช้งานจากผู้ให้บริการที่แตกต่างกัน

ในปี ค.ศ. 2011 สถาบันวิชาชีพวิศวกรไฟฟ้าและอิเล็กทรอนิกส์ (institute of electrical and electronics engineers, IEEE) ได้กำหนดมาตรฐานที่ใช้ในเทคโนโลยีการสื่อสารระหว่างอุปกรณ์สำหรับระบบจัดการพลังงานเพื่อการใช้พลังงานไฟฟ้าอย่างมีประสิทธิภาพในระดับอาคาร เมือง หรือ ภูมิภาค โดยมีการพัฒนาการสื่อสารแบบเปิดตามมาตรฐาน IEEE1888 ในโครงการชื่อ Ubiquitous Green Community Control Network (UGCCNet) [6] มาตรฐาน IEEE1888 เป็นมาตรฐานการสื่อสารระหว่างอุปกรณ์ที่สามารถประยุกต์ใช้กับระบบจัดการพลังงานภายในอาคาร ซึ่งมีวัตถุประสงค์หลัก [6] คือ การเฝ้าระวังสภาพแวดล้อมโดยใช้อุปกรณ์ตัวรับรู้ และการควบคุมอุปกรณ์ตัวกระตุ้นเพื่อลดปริมาณการใช้พลังงานอย่างมีประสิทธิภาพ ผู้ใช้สามารถเข้าถึงการสื่อสารระหว่างอุปกรณ์ระยะไกลผ่านทางอินเทอร์เน็ตด้วยโพรโทคอล TCP/IP โดยตัวรับรู้หรือตัวกระตุ้นถูกเข้าถึงและควบคุมผ่านเกตเวย์ (gateway) ข้อมูลที่ได้สามารถถูกส่งไปเก็บยังหน่วยเก็บข้อมูล (storage) และถูกเรียกเพื่อแจ้งไปยังโปรแกรมประยุกต์ (application) ได้ มาตรฐาน IEEE1888 มีการกำหนดองค์ประกอบ คือ เกตเวย์ หน่วยเก็บข้อมูล และโปรแกรมประยุกต์ สำหรับใช้ในการสื่อสารตามมาตรฐาน และมีการกำหนดโพรโทคอลที่ใช้ภายในโครงข่ายให้เป็นมาตรฐานเดียวกันเพื่อให้องค์ประกอบแต่ละองค์ประกอบในโครงข่ายการสื่อสารระหว่างอุปกรณ์สามารถทำงานร่วมกันได้

มาตรฐาน IEEE1888 ได้ถูกนำมาใช้ในโครงการ CU Building Energy Management

System (CU-BEMS) [7] ผ่านความร่วมมือจากภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย มีวัตถุประสงค์เพื่อเฝ้าสังเกตปริมาณการใช้พลังงานและสภาพแวดล้อมภายในอาคาร ให้นิสิตและบุคคลากรภายในคณะวิศวกรรมศาสตร์ได้รู้ถึงปริมาณการใช้พลังงานไฟฟ้าในส่วนต่าง ๆ ของอาคาร ตลอดจนสามารถตระหนักถึงการใช้พลังงานไฟฟ้าอย่างมีประสิทธิภาพ โครงการ CU-BEMS ประกอบด้วย มาตรวัดอัจฉริยะ [8] ตัวรับรู้สภาพแวดล้อมแบบไร้สาย [9] และจอแสดงผลภายในอาคาร [10] เป็นต้น ซึ่งมีส่วนประกอบหลักจากมาตรฐานการสื่อสาร IEEE1888 ที่ถูกนำมาใช้ในโครงการ CU-BEMS ประกอบด้วย เกตเวย์ หน่วยเก็บข้อมูล และโปรแกรมประยุกต์ โดยการทำงานของระบบ CU-BEMS ผ่านความสัมพันธ์ระหว่างส่วนประกอบหลักทั้ง 3 คือ ข้อมูลที่วัดได้จากตัวรับรู้สภาพแวดล้อม (CU-BEMS sensor) จะถูกส่งไปยังเกตเวย์ในระบบ CU-BEMS (CU-BEMS gateway) แบบไร้สาย หลังจากนั้นเกตเวย์จึงทำการแปลงข้อมูลก่อนที่จะส่งข้อมูลไปเก็บยังหน่วยเก็บข้อมูลในระบบ CU-BEMS ด้วยโพรโทคอล WRITE ตามมาตรฐาน IEEE1888 ในขณะเดียวกันที่โปรแกรมประยุกต์สามารถเรียกใช้ข้อมูลแบบคาบเวลาจากหน่วยเก็บข้อมูลด้วยโพรโทคอล FETCH ตามมาตรฐาน IEEE1888

อีกหนึ่งมาตรฐานการสื่อสารระหว่างอุปกรณ์ที่จะถูกนำมาใช้ในวิทยานิพนธ์ฉบับนี้คือมาตรฐานการสื่อสารระหว่างอุปกรณ์เพื่อการจัดการปริมาณการใช้พลังงานภายในบ้าน หรือเรียกว่า มาตรฐาน ECHONET Lite [11], [12] โดยมาตรฐาน ECHONET Lite มีวัตถุประสงค์ [11] คือ โพรโทคอลตามมาตรฐานถูกออกแบบมาเพื่อง่ายต่อการติดตั้งโครงข่ายการสื่อสารระหว่างอุปกรณ์ภายในบ้าน อุปกรณ์ที่ใช้งานภายในโครงข่ายเดียวกันรองรับการใช้งานจากผู้ผลิตและผู้ให้บริการที่แตกต่างกันได้ โมเดลของสถาปัตยกรรมถูกออกแบบมาเพื่อการใช้งานรายบุคคลสำหรับการพัฒนาระบบแอพพลิเคชั่น มิดเดิลแวร์การสื่อสาร (ECHONET Lite communication middleware) รองรับการพัฒนาอุปกรณ์ที่ใช้งานในโครงข่าย และรองรับบริการแอพพลิเคชั่นจากการพัฒนาแอพพลิเคชั่น มาตรฐาน ECHONET Lite ถูกพัฒนามาในรูปแบบมาตรฐานแบบเปิด โดยมีระดับชั้นมิดเดิลแวร์การสื่อสารอยู่ระหว่างระดับชั้นแอพพลิเคชัน (application) และการสื่อสารระดับล่าง (lower communication) มาตรฐาน ECHONET Lite ได้มีการกำหนดโครงข่ายการสื่อสารที่ใช้โพรโทคอลตามมาตรฐาน ECHONET Lite เรียกว่า โครงข่าย ECHONET Lite (ECHONET Lite network) ซึ่งสามารถเชื่อมต่อออกไปยังนอกโครงข่ายผ่านเกตเวย์ของมาตรฐาน ECHONET Lite (ECHONET Lite gateway)

จากงานวิจัย [13] และ [14] ที่ผ่านมาได้มีการนำเสนอการพัฒนาโพรโทคอลที่ใช้ในเกตเวย์สำหรับการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M ทั้งแบบคาบเวลาและแบบทันทีเพื่อประสานข้อมูลระหว่างสองมาตรฐานนี้ในโครงการ CU-BEMS โดยเป็นการประสานข้อมูลระหว่างหน่วยเก็บข้อมูล (storage) ตามมาตรฐาน IEEE1888 และคลังเก็บข้อมูล (repository) ตามมาตรฐาน ETSI M2M เกตเวย์ที่ทำหน้าที่เป็นตัวประสานข้อมูลนี้เรียกว่า พร็อกซีเกตเวย์ (proxy gateway) ซึ่งมีการเรียกข้อมูลจากหน่วยเก็บข้อมูลตามมาตรฐาน IEEE1888 ด้วยโพรโทคอล FETCH แบบคาบเวลา หรือ TRAP แบบทันที และเรียกข้อมูลจากคลังเก็บข้อมูลตามมาตรฐาน ETSI M2M ด้วยวิธีการ RETRIEVE แบบคาบเวลา หรือ NOTIFY แบบทันที สำหรับในส่วนของการบันทึกข้อมูลนั้นเกตเวย์จะใช้โพรโทคอล WRITE ในการบันทึกข้อมูลไปยังหน่วยเก็บข้อมูลตามมาตรฐาน IEEE1888 และใช้วิธีการ CREATE ในการบันทึกข้อมูลไปยังคลังเก็บข้อมูลตามมาตรฐาน ETSI M2M

เพื่อเพิ่มความสามารถและความยืดหยุ่นในการใช้งานระบบเฝ้าระวังสภาพแวดล้อมและตรวจจับปริมาณการใช้พลังงานไฟฟ้าภายในอาคารตามโครงการ CU-BEMS ซึ่งใช้การสื่อสารระหว่างอุปกรณ์

ตามมาตรฐาน IEEE1888 ให้สามารถรองรับการใช้งานมาตรฐาน ECHONET Lite ได้ วิทยานิพนธ์ฉบับนี้จึงนำเสนอการพัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ (interworking proxy gateway) สำหรับการประสานข้อมูลแบบทันทีในการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite เริ่มจากกระบวนการค้นหาโนด ECHONET Lite (ECHONET Lite node) และอุปกรณ์ ECHONET Lite (ECHONET Lite equipment) ที่ถูกเชื่อมต่ออยู่ภายในโครงข่าย ECHONET Lite โดยส่งการร้องขอแบบแพร่สัญญาณ (broadcast) จากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ไปยังทุกโนดที่อยู่ภายในโครงข่าย จากนั้นจึงรอการตอบกลับจากโนดต่าง ๆ พร้อมกับข้อมูลของโนดนั้น ๆ (node profile) เช่น หมายเลขที่อยู่ไอพี (IP address) ซึ่งใช้สำหรับระบุโนดภายในโครงข่าย ECHONET Lite เดียวกัน เป็นต้น เมื่ออินเตอร์เวิร์กกิงพร็อกซีเกตเวย์สามารถสื่อสารกับโนดภายในโครงข่าย ECHONET Lite ได้แล้วจึงสามารถทำหน้าที่เป็นตัวกลางในการสื่อสารข้อมูลระหว่างโนดภายในโครงข่าย ECHONET Lite และโครงข่ายภายนอกในที่นี้คือระบบ CU-BEMS ได้ ในโดเมนของมาตรฐาน ECHONET Lite ใช้การร้องขอแบบ WRITE เพื่อเขียนชุดคำสั่งไปยังอุปกรณ์ ECHONET Lite และแบบ NOTIFY เพื่อแจ้งเตือนข้อมูลของอุปกรณ์ ECHONET Lite ที่เกิดการเปลี่ยนแปลง ส่วนในโดเมนของมาตรฐาน IEEE1888 นั้นใช้โพรโทคอล WRITE เพื่อจัดเก็บข้อมูลไปยังหน่วยเก็บข้อมูล และใช้โพรโทคอล TRAP เพื่อแจ้งเตือนข้อมูลที่เกิดการเปลี่ยนแปลงในหน่วยเก็บข้อมูล สำหรับการสื่อสารในระบบ CU-BEMS มีรูปแบบของข้อมูลที่ใช้ในการสื่อสารแบบ XML ตามมาตรฐาน IEEE1888 และมี point ID เป็นตัวระบุตัวรับรู้สภาพแวดล้อมไร้สายแต่ละตัว

1.2 วัตถุประสงค์ของงานวิทยานิพนธ์

1. พัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์สำหรับการประสานข้อมูลแบบทันทีในระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ในระบบ CU-BEMS
2. ประยุกต์ใช้ระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite เพื่อรับการแจ้งเตือนการเปลี่ยนแปลงข้อมูลที่ได้จากตัวรับรู้สภาพแวดล้อมไร้สายในระบบ CU-BEMS และนำไปใช้ควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศที่รองรับมาตรฐาน ECHONET Lite
3. ประยุกต์ใช้ระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite เพื่อรับการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite และส่งข้อมูลที่เปลี่ยนแปลงไปเก็บยังหน่วยเก็บข้อมูลระบบ CU-BEMS

1.3 ขอบเขตวิทยานิพนธ์

1. พัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์สำหรับการประสานข้อมูลในระบบการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite โดยการโปรแกรมลงบนภาษา javascript ที่มีการดำเนินการด้วยแพลตฟอร์ม node.js
2. อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์สามารถเรียกหรือร้องขอการแจ้งเตือนการเปลี่ยนแปลงข้อมูลตัวรับรู้สภาพแวดล้อมไร้สายจากหน่วยเก็บข้อมูลระบบ CU-BEMS ด้วยโพรโทคอล TRAP

ตามมาตรฐาน IEEE1888 เพื่อใช้ควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศด้วยการร้องขอแบบ WRITE ตามมาตรฐาน ECHONET Lite

3. อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์สามารถเรียกหรือร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศที่เชื่อมต่อกับโนด ECHONET Lite ด้วยการร้องขอแบบ NOTIFY ตามมาตรฐาน ECHONET Lite เพื่อส่งข้อมูลที่เปลี่ยนแปลงไปเก็บยังหน่วยเก็บข้อมูลระบบ CU-BEMS ด้วยโพรโทคอล WRITE ตามมาตรฐาน IEEE1888
4. ทดสอบ วิเคราะห์ และประเมินผลการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite

1.4 ขั้นตอนและวิธีการดำเนินงาน

1. ศึกษามาตรฐาน ECHONET Lite และมาตรฐาน IEEE1888 ที่ใช้งานในระบบ CU-BEMS
2. ทบทวนบทความและงานวิจัยที่เกี่ยวข้องเพื่อนำไปประยุกต์ใช้ในวิทยานิพนธ์
3. พัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์สำหรับควบคุมอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ในโหมดต่าง ๆ เช่น การเปิดปิด การตั้งค่าอุณหภูมิ การวัดค่าอุณหภูมิจากตัวรับรู้ เป็นต้น
4. พัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์สำหรับการประสานข้อมูลแบบทันทีในการแจ้งเตือนการเปลี่ยนแปลงข้อมูลของตัวรับรู้ไร้สายในระบบ CU-BEMS เพื่อใช้ควบคุมอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite
5. ทดสอบ วิเคราะห์ และสรุปผลการทำงานของระบบที่ถูกพัฒนาขึ้นในวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ผลการศึกษากระบวนการที่จำเป็นสำหรับการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite
2. ตัวรับรู้สภาพแวดล้อมในระบบ CU-BEMS ตามมาตรฐาน IEEE1888 และเครื่องปรับอากาศที่รองรับมาตรฐาน ECHONET Lite สามารถทำงานสอดคล้องกันได้

1.6 ประมวลวิทยานิพนธ์

บทที่ 1 บทนำ: กล่าวถึงความเป็นมาและความสำคัญของการพัฒนาการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite วัตถุประสงค์ของงานวิทยานิพนธ์ ขอบเขตวิทยานิพนธ์ ขั้นตอนและวิธีการดำเนินงาน และกล่าวถึงประโยชน์ที่คาดว่าจะได้รับ

บทที่ 2 ทฤษฎีพื้นฐาน: กล่าวถึงทฤษฎีเบื้องต้นและการประยุกต์ใช้งานของมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ในการทำงานร่วมกันระหว่างสองมาตรฐาน

บทที่ 3 โครงสร้างและการทำงานของระบบการทำงานร่วมกันแบบทันที: กล่าวถึงโครงสร้างส่วนประกอบ และการทำงานของระบบการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ที่นำเสนอในวิทยานิพนธ์

บทที่ 4 การทดสอบและการประเมินผลของระบบการทำงานร่วมกันแบบทันที: กล่าวถึงการทดสอบและการประเมินผลการทดสอบของระบบการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite โดยการพัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เพื่อประสานข้อมูลแบบทันที

บทที่ 5 บทสรุปและข้อเสนอแนะ: กล่าวถึงบทสรุปของงานวิจัยทั้งหมดในวิทยานิพนธ์ฉบับนี้ และแนวทางการพัฒนางานวิจัยต่อไป

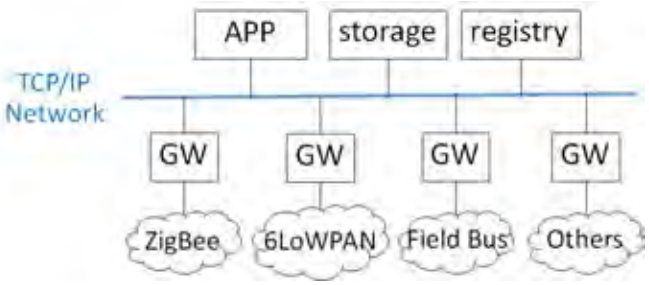
บทที่ 2
ทฤษฎีพื้นฐาน

2.1 มาตรฐาน IEEE1888 [6]

สถาปัตยกรรมโครงข่ายมาตรฐาน IEEE1888 หรือ สถาปัตยกรรมโครงข่ายการควบคุมชุมชนสีเขียวอย่างแพร่หลาย (Ubiquitous Green Community Control Network, UGCCNet) ใช้โพรโทคอลที่ถูกออกแบบมาเพื่อการสื่อสารบนพื้นฐานของสถาปัตยกรรมโครงข่าย TCP/IP โดยมีจุดประสงค์เพื่อสามารถทำงานร่วมกันระหว่างองค์ประกอบต่าง ๆ (component) ที่อยู่ภายในโครงข่ายประกอบด้วย เกตเวย์ หน่วยเก็บข้อมูล และโปรแกรมประยุกต์ ซึ่งมีส่วนต่อประสาน (interface) ทั่วไปที่คล้ายกัน และรีจิสทรี (registry) ซึ่งมีส่วนต่อประสานที่แตกต่างออกไป

2.1.1 สถาปัตยกรรมโครงข่ายมาตรฐาน IEEE1888

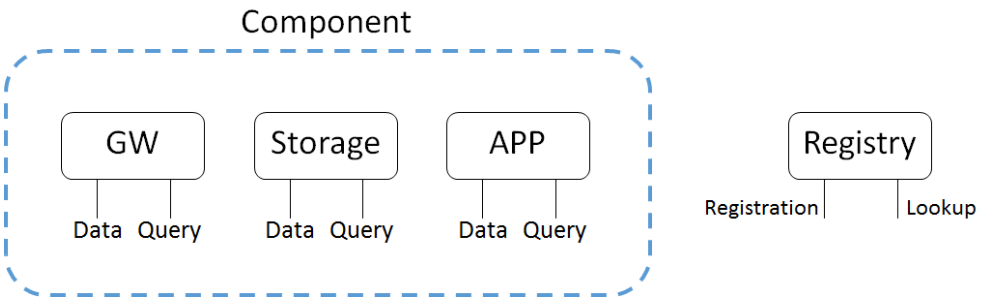
สถาปัตยกรรมโครงข่ายมาตรฐาน IEEE1888 ถูกแบ่งออกเป็น 2 ระนาบ ประกอบด้วย ระนาบควบคุม (control plane) คือ ส่วนของรีจิสทรี และระนาบข้อมูล (data plane) คือ ส่วนขององค์ประกอบทั้งสาม (เกตเวย์ หน่วยเก็บข้อมูล และโปรแกรมประยุกต์) โดยส่วนประกอบทั้งสามและรีจิสทรีถูกเชื่อมต่อเข้ากับโครงข่าย TCP/IP ดังรูปที่ 2.1



รูปที่ 2.1: สถาปัตยกรรมโครงข่าย IEEE1888 [6]

1. เกตเวย์: เป็นตัวกำหนดข้อมูลการเข้าถึงอุปกรณ์ทางกายภาพ เช่น ตัวรับรู้ หรือ ตัวกระตุ้น เป็นต้น จากโครงข่าย TCP/IP เกตเวย์เป็นเสมือนตัวกลางในการสั่งการไปยังตัวกระตุ้นตามข้อมูลที่ถูกเขียนจากองค์ประกอบอื่น หรือส่งผ่านข้อมูลที่อ่านได้จากตัวรับรู้ไปยังองค์ประกอบอื่น
2. หน่วยเก็บข้อมูล: ทำหน้าที่เก็บข้อมูลตามลำดับข้อมูลที่ถูกบันทึกจากองค์ประกอบอื่นที่ร้องขอ และข้อมูลในหน่วยเก็บข้อมูลสามารถถูกเรียกใช้ได้จากองค์ประกอบอื่นที่ร้องขอ
3. โปรแกรมประยุกต์: ทำหน้าที่แสดงข้อมูลที่อ่านได้จากตัวรับรู้ หรือสั่งการไปยังตัวกระตุ้นผ่านหน้าอินเทอร์เฟส

4. รีจิสทรี: เป็นเสมือนศูนย์กลางควบคุมการทำงานระหว่างองค์ประกอบทั้งสาม ทำงานอยู่ในส่วนของระนาบควบคุม และไม่ทำงานโดยตรงกับข้อมูลที่อ่านได้จากตัวรับรู้หรือข้อมูลที่ถูกตั้งค่าไปยังตัวกระตุ้น



รูปที่ 2.2: แบบจำลองการทำงานขององค์ประกอบในมาตรฐาน IEEE1888 [6]

รูปที่ 2.2 แสดงแบบจำลองการทำงานของเกตเวย์ หน่วยเก็บข้อมูล โปรแกรมประยุกต์ และรีจิสทรี ตามมาตรฐาน IEEE1888 องค์ประกอบสามองค์ประกอบแรกนั้นมีส่วนต่อประสาน (interface) ที่เหมือนกัน ซึ่งมีวิธีที่เรียกใช้งาน คือ data และ query แต่ในรีจิสทรีนั้นมีส่วนต่อประสานที่เรียกใช้งานวิธี registration และ lookup วิธีแต่ละวิธีถูกอธิบายได้ดังนี้

1. Data คือ วิธีที่ถูกใช้สำหรับการดึงข้อมูลจากองค์ประกอบนั้น
2. Query คือ วิธีที่ถูกใช้สำหรับการดันข้อมูลเข้าไปยังองค์ประกอบนั้น
3. Registration คือ วิธีที่ถูกใช้สำหรับการลงทะเบียนองค์ประกอบ
4. Lookup คือ วิธีที่ถูกใช้สำหรับการค้นหาองค์ประกอบที่กำลังถูกเรียกใช้งาน

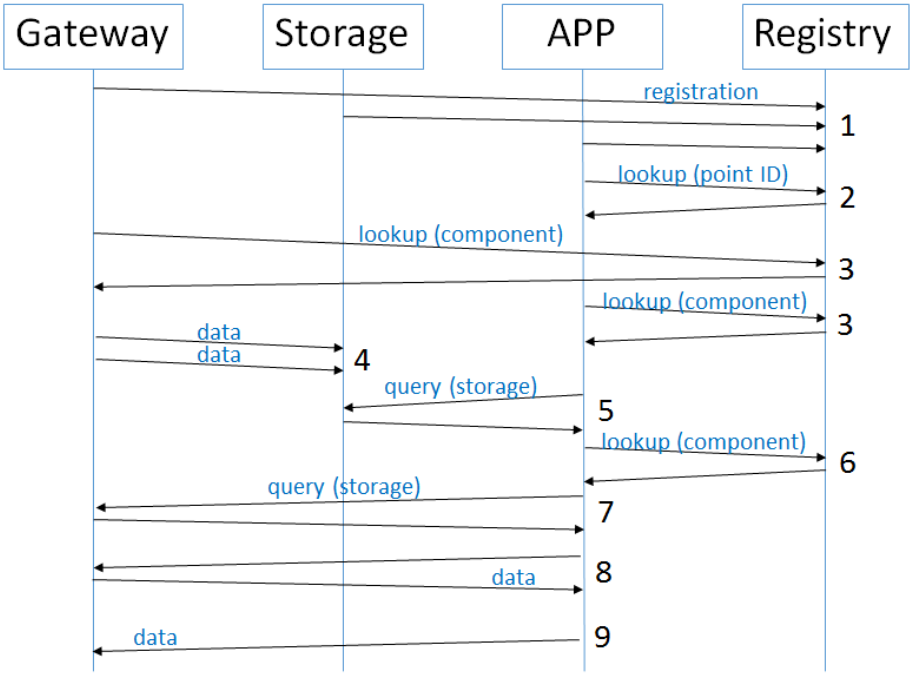
2.1.2 การระบุตัวรับรู้หรือตัวกระตุ้นด้วย URI

มาตรฐาน IEEE1888 ได้กำหนด point สำหรับการระบุตัวรับรู้หรือตัวกระตุ้นที่ใช้งานตามมาตรฐาน IEEE1888 เพื่อประโยชน์ในการจำแนกอุปกรณ์แต่ละตัวที่ทำงานอยู่ในโครงข่ายเดียวกัน และเพื่อจัดการ point ตามตำแหน่งการติดตั้งอุปกรณ์นั้น จึงง่ายต่อการเรียกใช้งานอุปกรณ์และซ่อมบำรุง ทั้งนี้เนื่องจากอุปกรณ์ตัวรับรู้และตัวกระตุ้นภายในระบบนั้นมีจำนวนมากและหลากหลายชนิด โดย point แต่ละตัวถูกระบุตามรูปแบบของ URI มีการระบุ point ID เป็นอย่างแรก และมีการระบุ point set เพื่อรวมกลุ่ม point ID ย่อยเป็นลำดับชั้น ในการระบุ URI ของ point ID แต่ละตัวนั้น ส่วนแรกของ URI มักถูกระบุเป็นชื่อของเกตเวย์ เนื่องจากอุปกรณ์ตัวรับรู้และตัวกระตุ้นถูกเชื่อมต่อกับเกตเวย์ ดังตัวอย่างของการระบุ URI นี้

point ID = “http://<GW host name>/<any unique format to be defined>”

2.1.3 ลำดับการสื่อสารของโพรโทคอลตามมาตรฐาน IEEE1888

ลำดับการติดต่อสื่อสารระหว่างองค์ประกอบต่าง ๆ ทั้งสามส่วนโดยมีรีจิสทรีเป็นเสมือนศูนย์กลางควบคุมการทำงานร่วมกันในระนาบข้อมูลถูกแสดงในรูปแบบของลำดับซึ่งเป็นแบบอย่าง (typical sequence) ดังรูปที่ 2.3



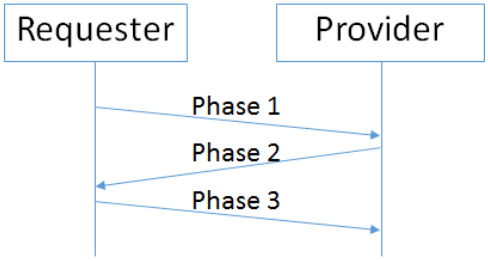
รูปที่ 2.3: ลำดับการสื่อสารของโพรโทคอลตามมาตรฐาน IEEE1888 [6]

1. การลงทะเบียนของส่วนประกอบต่าง ๆ ทั้งสามโดยใช้ข้อมูล point ID
2. การค้นหา point โดยใช้โพรโทคอลสอบถาม (query protocol)
3. การค้นหาหน่วยเก็บข้อมูลโดยระบุ point ซึ่งจะมีการตอบกลับเป็น URI
4. เกตเวย์ส่งข้อมูลที่อ่านได้จากตัวรับรู้ไปเก็บยังหน่วยเก็บข้อมูล
5. โปรแกรมประยุกต์ร้องขอข้อมูลจากหน่วยเก็บข้อมูล
6. การค้นหาเกตเวย์โดยระบุ point ซึ่งจะมีการตอบกลับเป็น URI
7. โปรแกรมประยุกต์รับข้อมูลจากเกตเวย์
8. เกตเวย์ตอบกลับการอัพเดทข้อมูลจากการร้องขอโดยโปรแกรมประยุกต์
9. โปรแกรมประยุกต์เขียนคำสั่งไปยังตัวกระตุ้นผ่านเกตเวย์

2.1.4 โพรโทคอลการสื่อสารหลักระหว่างองค์ประกอบ

โพรโทคอลหลักที่ใช้ในการติดต่อสื่อสารระหว่างองค์ประกอบทั้งสาม (เกตเวย์ หน่วยเก็บข้อมูล และโปรแกรมประยุกต์) ตามมาตรฐาน IEEE1888 คือ โพรโทคอล FETCH WRITE และ TRAP ซึ่งใช้ในการเรียก บันทึก และแจ้งเตือนข้อมูลระหว่างองค์ประกอบ และมีโพรโทคอลของข้อความที่ใช้ในการติดต่อสื่อสาร คือ simple object access protocol (SOAP)

1. โพรโทคอล FETCH คือ โพรโทคอลที่ใช้ในการร้องขอข้อมูลจากองค์ประกอบหนึ่งไปยังอีกองค์ประกอบหนึ่งในการสื่อสารระหว่างองค์ประกอบ โดยมีองค์ประกอบที่ทำการร้องขอเรียกว่า ผู้ร้องขอ (requester) และองค์ประกอบที่ถูกร้องขอเรียกว่า ผู้ให้บริการ (provider) มีลำดับการสื่อสารดังรูปที่ 2.4



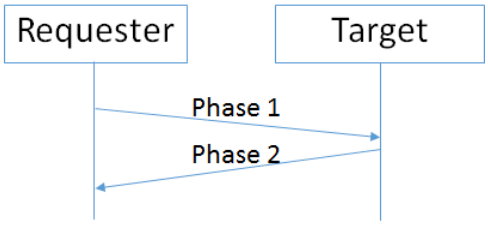
รูปที่ 2.4: โพรโทคอล FETCH [6]

   (a) เฟส 1 (phase 1): ผู้ร้องขอส่งการร้องขอไปยังผู้ให้บริการพร้อมกับกำหนดขนาดของข้อมูลที่รับได้

   (b) เฟส 2 (phase 2): ผู้ให้บริการตอบกลับข้อมูลทั้งหมดจากการร้องขอ ถ้าข้อมูลมีขนาดใหญ่เกินขนาดที่กำหนดไว้ ข้อมูลจะถูกแบ่งออกเป็นส่วน ๆ และจึงถูกส่งทีละส่วน

   (c) เฟส 3 (phase 3): ผู้ร้องขอทำการร้องขอข้อมูลไปยังผู้ให้บริการอีกครั้งในกรณีที่ยังมีข้อมูลส่วนที่เหลือจนกระทั่งครบทุกส่วน

2. โพรโทคอล WRITE คือ โพรโทคอลที่ใช้ในการบันทึกข้อมูลจากองค์ประกอบหนึ่งไปยังอีกองค์ประกอบหนึ่งในการสื่อสารระหว่างองค์ประกอบ โดยมีองค์ประกอบที่ร้องขอการบันทึกข้อมูลเรียกว่า ผู้ร้องขอ และองค์ประกอบที่เป็นเป้าหมายในการร้องขอการบันทึกข้อมูลเรียกว่า องค์ประกอบเป้าหมาย (target) มีลำดับการสื่อสารดังรูปที่ 2.5

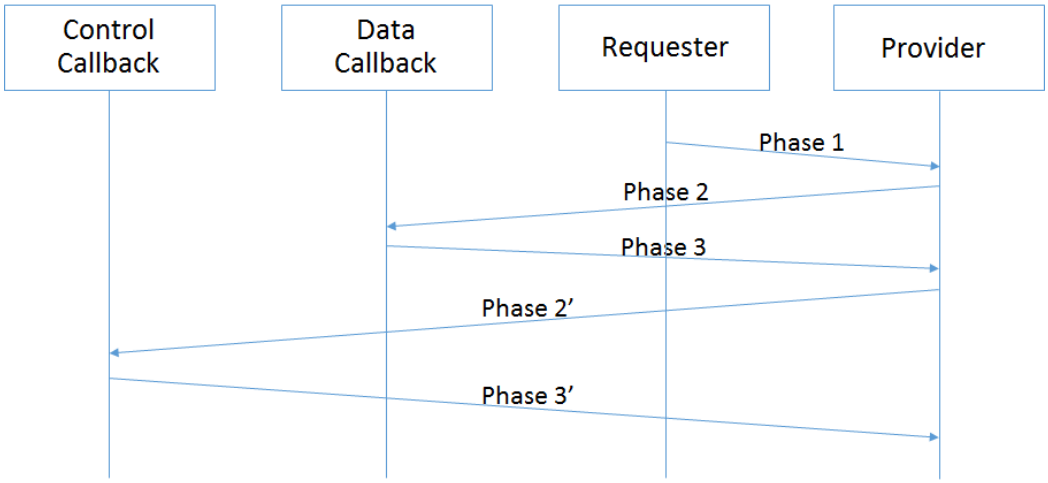


รูปที่ 2.5: โพรโทคอล WRITE [6]

   (a) เฟส 1 (phase 1): ผู้ร้องขอส่งข้อความร้องขอการบันทึกข้อมูลไปยังองค์ประกอบเป้าหมาย

   (b) เฟส 2 (phase 2): องค์ประกอบเป้าหมายตอบกลับไปยังผู้ร้องขอเพื่อยืนยันการบันทึกข้อมูลว่าเสร็จสมบูรณ์หรือมีข้อผิดพลาด

3. โพรโทคอล TRAP คือ โพรโทคอลที่ใช้ในการร้องขอการแจ้งเตือนข้อมูลจากองค์ประกอบหนึ่งไปยังอีกองค์ประกอบหนึ่งเฉพาะกรณี โดยมีการกำหนดเงื่อนไขการแจ้งเตือนข้อมูลล่วงหน้า และส่งข้อมูลกลับมาในกรณีที่ตรงตามเงื่อนไขที่กำหนดไว้ ประกอบด้วย ผู้ร้องขอ ผู้ให้บริการ ผู้เรียกกลับข้อมูล (data callback) และผู้เรียกกลับการควบคุม (control callback) ซึ่งแบ่งการทำงานออกเป็นลำดับต่าง ๆ ดังรูปที่ 2.6



รูปที่ 2.6: โพรโทคอล TRAP [6]

(a) เฟส 1 (phase 1): ผู้ร้องขอส่งเงื่อนไขการแจ้งเตือนข้อมูลรวมถึงระยะเวลาในเงื่อนไขนั้นพร้อมกับระบุ URI ของผู้รับข้อมูล

(b) เฟส 2 (phase 2): ผู้ให้บริการส่งข้อมูลไปยังผู้เรียกกลับข้อมูลตาม URI ที่ได้ระบุไว้ในกรณีที่ตรงตามเงื่อนไขการแจ้งเตือน

(c) เฟส 3 (phase 3): การตอบกลับจากผู้เรียกกลับข้อมูลไปยังผู้ให้บริการเพื่อยืนยันว่าการส่งสมบูรณ์หรือมีข้อผิดพลาด

(d) เฟส 2' (phase 2'): กรณีที่มีข้อผิดพลาดในเฟสที่ 3 ผู้ให้บริการจะแจ้งไปยัง URI ของผู้เรียกกลับการควบคุม

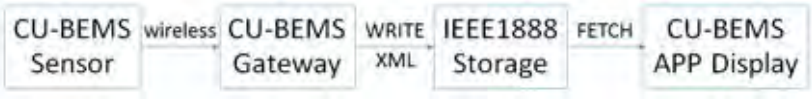
(e) เฟส 3' (phase 3'): ผู้เรียกกลับการควบคุมตอบกลับไปยังผู้ให้บริการเพื่อยืนยันว่าการส่งสมบูรณ์หรือมีข้อผิดพลาด

2.1.5 การใช้งานมาตรฐาน IEEE1888 ในโครงการ CU-BEMS

โครงการ CU-BEMS [7] คือ โครงการของระบบเฝ้าระวังสภาพแวดล้อมและจัดการปริมาณการใช้พลังงานภายในตัวอาคารแบบไร้สาย ถูกพัฒนาขึ้นโดยภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย มีวัตถุประสงค์เพื่อสาธิตระบบจัดการปริมาณการใช้พลังงานภายในอาคารที่จะถูกนำไปประยุกต์ใช้ในอนาคต และเปรียบเทียบพฤติกรรมการใช้พลังงานไฟฟ้าก่อนและหลังติดตั้งระบบ CU-BEMS ระบบ CU-BEMS ได้มีการติดตั้งตัวรับรู้สภาพแวดล้อมไร้สาย 4 ชนิด คือ ตัวรับรู้อุณหภูมิ ตัวรับรู้ความชื้นสัมพัทธ์ ตัวรับรู้แสง และตัวรับรู้การเคลื่อนไหวของคน ตัวรับรู้สภาพแวดล้อมไร้สายนี้มีจำนวนทั้งสิ้น 200 ตัว เพื่อเฝ้าสังเกตุสภาพแวดล้อมส่วนต่าง ๆ ภายในอาคาร มีการ

ติดตั้งสมาร์ตมิเตอร์จำนวน 20 ตัว เพื่อวัดปริมาณพลังงานไฟฟ้าที่ผลิตได้และใช้ไป และระบบเครื่องปรับอากาศจำนวน 10 ตัว นอกจากนี้ยังมีสถานีประจุไฟฟ้าสำหรับพาหนะ แผงเซลล์สุริยะด้วยความจุในการผลิต 40 kW. และกังหันลมผลิตไฟฟ้าขนาดเล็ก ในส่วนของการแสดงผลในระบบ CU-BEMS นั้นผู้ใช้งานสามารถเข้าถึงข้อมูลผ่านหน้าเว็บไซต์ [15] หรือแอพพลิเคชั่นบนระบบปฏิบัติการแอนดรอยด์ [16] ตลอดจนมีการติดตั้งจอแสดงผล IDaaS ไว้ที่ส่วนหลักภายในอาคาร 3 ส่วน [17] เพื่อเพิ่มความสนใจในการเข้าถึงข้อมูลของผู้ใช้งานและผู้ใช้งานสามารถโต้ตอบกับจอแสดงผลได้

จากรูปที่ 2.7 แสดงตัวอย่างการประยุกต์ใช้งานมาตรฐาน IEEE1888 ในระบบ CU-BEMS เพื่อนำไปใช้ในงานวิจัยนี้ โดยข้อมูลที่วัดได้จากตัวรับรู้สภาพแวดล้อมไร้สายถูกส่งไปยังเกตเวย์ของระบบ CU-BEMS จากนั้นเกตเวย์จึงแปลงข้อมูลดิบที่ได้รับมาให้อยู่ในรูปแบบ XML ตามที่ใช้งานในมาตรฐาน IEEE1888 เพื่อส่งข้อมูลไปเก็บยังหน่วยเก็บข้อมูลด้วยโพรโทคอล WRITE ตามมาตรฐาน และผู้ใช้งานสามารถดึงข้อมูลจากหน่วยเก็บข้อมูลเพื่อนำไปใช้งานหรือแสดงผลบนโปรแกรมประยุกต์ด้วยโพรโทคอล FETCH สำหรับดึงข้อมูลแบบคาบเวลา



รูปที่ 2.7: ตัวอย่างบล็อกการทำงานระบบ CU-BEMS

ระบบ CU-BEMS ที่ใช้งานตามมาตรฐาน IEEE1888 นี้ยังสามารถถูกนำไปประยุกต์ใช้สำหรับการประสานข้อมูลแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ตามที่ได้กล่าวไว้ในวิทยานิพนธ์ฉบับนี้ โดยการประสานข้อมูลจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ใช้โพรโทคอล TRAP เพื่อร้องขอการแจ้งเตือนการเปลี่ยนแปลงข้อมูลในหน่วยเก็บข้อมูลแบบทันที หรือใช้โพรโทคอล FETCH เพื่อดึงข้อมูลจากหน่วยเก็บข้อมูลแบบคาบเวลา จากนั้นข้อมูลจะถูกนำไปแสดงผล หรือใช้งานโดยการส่งการร้องขอแบบ WRITE ตามมาตรฐาน ECHONET Lite เพื่อควบคุมไปยังอุปกรณ์ไฟฟ้าที่รองรับการใช้งานมาตรฐาน ECHONET Lite เช่น อุปกรณ์เครื่องปรับอากาศ เป็นต้น และในทิศทางกลับกันสำหรับการประสานข้อมูลจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 นั้นอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จะส่งการร้องขอแบบ NOTIFY เพื่อร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะของอุปกรณ์มาตรฐาน ECHONET Lite แบบทันที หรือดึงข้อมูลจากอุปกรณ์มาตรฐาน ECHONET Lite แบบคาบเวลาด้วยการร้องขอแบบ READ ก่อนที่จะส่งข้อมูลไปเก็บยังหน่วยเก็บข้อมูลด้วยโพรโทคอล WRITE ตามมาตรฐาน IEEE1888

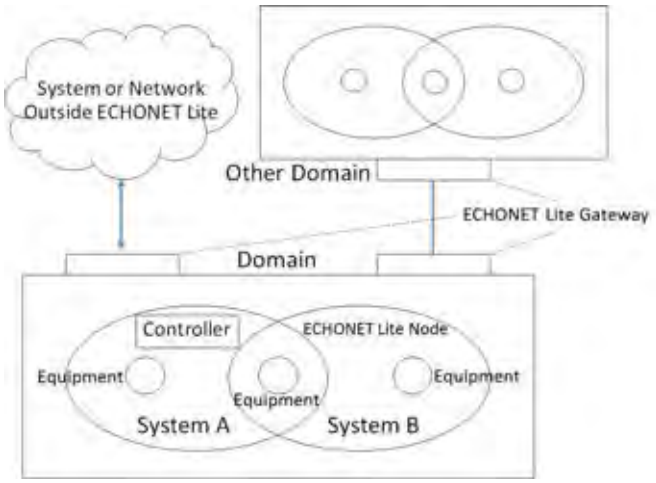
2.2 มาตรฐาน ECHONET Lite [11]

มาตรฐาน ECHONET Lite เป็นมาตรฐานที่ถูกออกแบบมาเพื่อใช้งานในระบบจัดการพลังงานภายในบ้าน และถูกออกแบบมาในรูปแบบมาตรฐานแบบเปิด จึงง่ายต่อการเชื่อมต่อระหว่างอุปกรณ์ไฟฟ้าภายในบ้านที่มาจากผู้ผลิตรายต่าง ๆ เข้ากับโครงสร้างการสื่อสารพื้นฐานของระบบที่มีอยู่เดิมจึงสะดวกในการติดตั้งระบบใหม่ ผู้ใช้สามารถควบคุมอุปกรณ์ไฟฟ้าและจัดการปริมาณพลังงานไฟฟ้าที่ใช้ภายในบ้าน ตลอดจนสามารถเชื่อมต่อออกไปยังระบบหรือโครงข่ายอื่นได้ มาตรฐาน ECHONET Lite มีจุดประสงค์ [11] คือ เพื่อให้ง่ายต่อการพัฒนาระบบโครงข่ายภายในบ้านจากผู้ให้บริการ

ที่ต่างกัน รองรับการใช้งานระบบโครงข่ายภายในบ้านในระยะยาวเนื่องจากสามารถติดตั้งเพิ่มเติมได้ง่าย ติดตั้งหรือเคลื่อนย้ายอุปกรณ์ภายในระบบได้ง่ายเนื่องจากมีการใช้งานแบบ plug-and-play และเชื่อมต่อไปยังระบบหรือโครงข่ายภายนอกได้ตามการใช้งานมาตรฐานทั่วไป

2.2.1 สถาปัตยกรรมมาตรฐาน ECHONET Lite

โครงข่าย ECHONET Lite (ECHONET Lite network) เป็นโครงข่ายการสื่อสารข้อมูลระหว่างอุปกรณ์ เช่น อุปกรณ์ไฟฟ้า ตัวรับรู้ หรือ ตัวกระตุ้น เป็นต้น โดยขอบเขตของระบบการทำงานและการจัดการที่มีคุณลักษณะเดียวกันในโครงข่ายถูกเรียกว่า โดเมน (domain) ในโดเมนแต่ละโดเมนประกอบด้วยตัวควบคุม (controller) ทำหน้าที่คอยเฝ้าระวังหรือสั่งการไปยังอุปกรณ์ นอกจากนี้ยังมีการติดต่อสื่อสารระหว่างโดเมนแต่ละโดเมนโดยมีเกตเวย์ (ECHONET Lite gateway) เป็นเสมือนส่วนต่อประสานสำหรับติดต่อสื่อสารไปยังโดเมนอื่น หรือติดต่อสื่อสารออกไปยังระบบอื่นภายนอกโครงข่าย ECHONET Lite ดังรูปที่ 2.8



รูปที่ 2.8: สถาปัตยกรรมมาตรฐาน ECHONET Lite [11]

จากรูปที่ 2.8 จะเห็นได้ว่าระบบ (system) แต่ละระบบนั้นครอบคลุมอุปกรณ์ได้แค่ภายในโดเมนเดียวกัน ไม่สามารถครอบคลุมออกไปยังโดเมนอื่นได้ และภายในโดเมนเดียวกันอาจมีระบบหรือตัวควบคุมได้มากกว่าหนึ่งตัวที่ครอบคลุมอุปกรณ์ภายในโดเมนนั้น ดังนั้นอุปกรณ์ตัวเดียวกันสามารถถูกครอบคลุมโดยระบบหรือตัวควบคุมที่แตกต่างกันได้

2.2.2 ส่วนประกอบในมาตรฐาน ECHONET Lite

การสื่อสารตามมาตรฐาน ECHONET Lite ภายในโครงข่าย ECHONET Lite ได้มีการกำหนดส่วนประกอบที่ใช้ในการสื่อสารประกอบด้วย โนด ECHONET Lite อุปกรณ์ ECHONET Lite ตัวปรับต่อมิดเดิลแวร์การสื่อสารของ ECHONET Lite และเกตเวย์ ECHONET Lite ซึ่งอธิบายคุณลักษณะการทำงานได้ดังนี้

1. โนด ECHONET Lite: เป็นตัวเชื่อมการสื่อสารระหว่างโครงข่าย ECHONET Lite และอุปกรณ์ที่อยู่ภายในโครงข่าย โดยมีหมายเลขที่อยู่ไอพีเป็นตัวระบุโนดนั้น ๆ โนด ECHONET

Lite มีหน้าที่ส่งหรือรับข้อมูลระหว่างโครงข่ายกับอุปกรณ์ (general node) หรือโนด ECHONET Lite มีหน้าที่แจ้งค่าข้อมูลหรือส่งอย่างเดียว (transmission-only node)

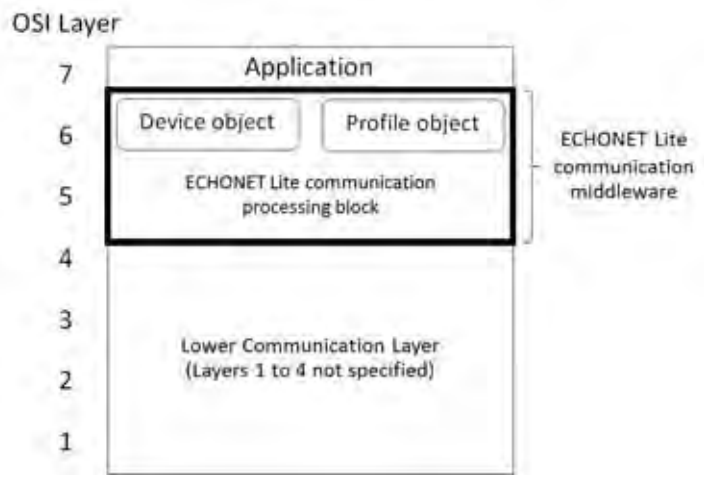
2. อุปกรณ์ ECHONET Lite: เป็นอุปกรณ์ที่ทำงานตามค่าของข้อมูลที่รับมาจากโนด ECHONET Lite หรือส่งค่าของข้อมูลที่ได้ไปยังโนด ECHONET Lite เพื่อนำข้อมูลไปใช้ เช่น ตัวกระตุ้น หรือ ตัวรับรู้ เป็นต้น อีกทั้งยังสามารถเป็นอุปกรณ์เสมือนเป็นศูนย์กลางการควบคุมอุปกรณ์อื่น ๆ ภายในโครงข่าย

3. ตัวปรับต่อมิดเดิลแวร์การสื่อสารของ ECHONET Lite: ทำหน้าที่เป็นตัวแปลงการเชื่อมต่ออุปกรณ์ไปยังโครงข่าย ECHONET Lite มีส่วนต่อประสานระหว่างอุปกรณ์และตัวปรับต่อมิดเดิลแวร์การสื่อสารของ ECHONET Lite อยู่แยกกัน

4. เกตเวย์ ECHONET Lite: เป็นตัวเชื่อมต่อจากโดเมนภายในโครงข่าย ECHONET Lite ไปยังระบบอื่น ๆ ภายนอกโครงข่าย หรือเชื่อมต่อระหว่างโดเมนกับโดเมนภายในโครงข่าย ECHONET Lite โดยจำนวนของเกตเวย์ภายในโดเมนอาจมีมากกว่าหนึ่งตัวขึ้นอยู่กับการเชื่อมต่อไปยังระบบอื่นภายนอกโครงข่ายหรือระหว่างโดเมนภายในโครงข่าย

2.2.3 ระดับชั้นการสื่อสารมาตรฐาน ECHONET Lite

ระดับชั้นในการสื่อสารตามมาตรฐาน ECHONET Lite แบ่งออกเป็น 3 ระดับหลัก คือ ซอฟต์แวร์ประยุกต์ (application software) มิดเดิลแวร์การสื่อสาร (communication middleware) และซอฟต์แวร์การสื่อสารระดับล่าง (lower-layer communication software) แสดงดังรูปที่ 2.9



รูปที่ 2.9: ระดับชั้นการสื่อสารมาตรฐาน ECHONET Lite [11]

1. ซอฟต์แวร์ประยุกต์: เป็นส่วนที่ติดต่อกับผู้ใช้งาน ให้บริการในการเข้าถึงการควบคุมอุปกรณ์และกำหนดหน้าที่การทำงานทางกายภาพของอุปกรณ์ภายในโครงข่าย ECHONET Lite

2. มิดเดิลแวร์การสื่อสาร: ให้บริการอยู่ระหว่างระดับชั้นซอฟต์แวร์ประยุกต์และซอฟต์แวร์การสื่อสารระดับล่าง และควบคุมกระบวนการสื่อสารตามมาตรฐาน ECHONET Lite

3. ซอฟต์แวร์การสื่อสารระดับล่าง: เปรียบได้กับหน้าที่การทำงานในระดับชั้นล่างของโมเดล OSI (OSI model) ครอบคลุมการทำงานเกี่ยวกับการประมวลผลข้อมูล (data processing) ตลอดจนโพรโทคอลตามมาตรฐาน ECHONET Lite ที่ใช้ในการส่งข้อมูลไปในตัวกลางการสื่อสาร

2.2.4 การประมวลผลอ็อบเจกต์ (object processing)

การประมวลผลการทำงานภายในมาตรฐาน ECHONET Lite เป็นการประมวลผลแบบอ็อบเจกต์ ถูกจำแนกออกเป็น 2 ชนิด คือ อ็อบเจกต์อุปกรณ์มาตรฐาน ECHONET Lite (ECHONET Lite device object) และอ็อบเจกต์โพรไฟล์มาตรฐาน ECHONET Lite (ECHONET Lite profile object)

1. อ็อบเจกต์อุปกรณ์: เป็นอ็อบเจกต์สำหรับนำมาเชื่อมต่อในโครงข่าย ECHONET Lite แทนตัวอุปกรณ์ เช่น ตัวรับรู้ อุปกรณ์ไฟฟ้าภายในบ้าน หรือ อุปกรณ์เครื่องปรับอากาศ เป็นต้น เพื่อบ่งบอกถึงข้อมูลรายละเอียดของอุปกรณ์ โดยข้อมูลรายละเอียดและการควบคุมอุปกรณ์จะถูกระบุในสมบัติ (property) ของอ็อบเจกต์ ส่วนรูปแบบการทำงานจะถูกระบุในบริการ (service)

2. อ็อบเจกต์โพรไฟล์: เป็นอ็อบเจกต์สำหรับบ่งบอกข้อมูลรายละเอียดของโนด ECHONET Lite เช่น ข้อมูลจากโรงงานผลิต รายการ หรือ จำนวนอ็อบเจกต์อุปกรณ์ที่ถูกเชื่อมต่อกับโนด ECHONET Lite เป็นต้น โดยข้อมูลรายละเอียด การควบคุม และรูปแบบการทำงานของอ็อบเจกต์โพรไฟล์ถูกแสดงในรูปสมบัติของอ็อบเจกต์

2.2.5 รูปแบบเฟรมมาตรฐาน ECHONET Lite (ECHONET Lite frame format)

รูปแบบของเฟรมที่ใช้ในการสื่อสารตามมาตรฐาน ECHONET Lite ถูกกำหนดไว้ในข้อกำหนดคุณลักษณะ (specification) ในมาตรฐาน ECHONET Lite จากรูปที่ 2.10 สามารถอธิบายรูปแบบของเฟรมและคุณลักษณะหน้าที่ของเฟรมแต่ละเฟรมได้ดังนี้

1. EHD1: เฟรมส่วนหัวลำดับที่ 1 ของข้อความ ECHONET Lite ตามข้อกำหนดในมาตรฐาน

2. EHD2: เฟรมส่วนหัวลำดับที่ 2 ของข้อความ ECHONET Lite ตามข้อกำหนดในมาตรฐาน

3. TID: เฟรมที่บ่งบอกถึงค่าของการเปลี่ยนแปลงสถานะระหว่างข้อความการร้องขอและข้อความการตอบกลับ

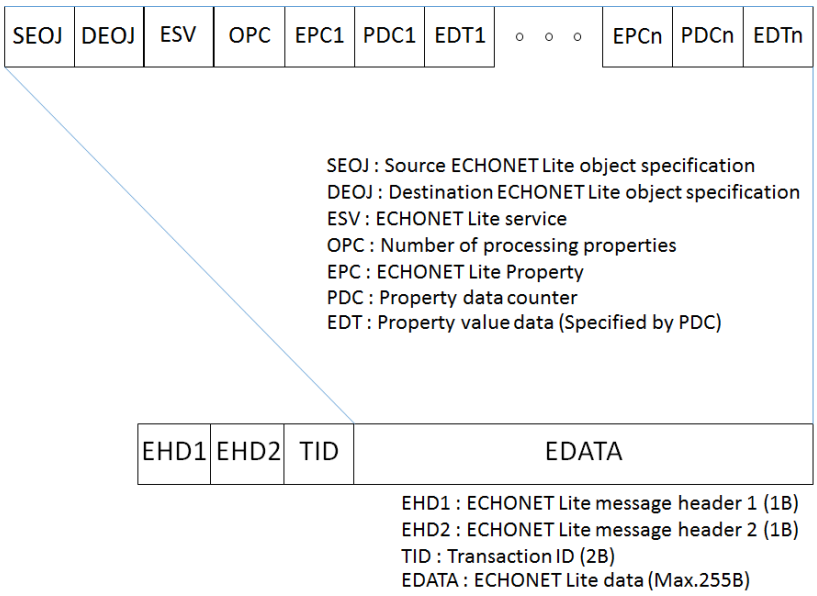
4. EDATA: เฟรมที่เป็นข้อมูลเพื่อใช้แลกเปลี่ยนกันในระดับชั้นมิดเดิลแวร์ของการสื่อสารตามมาตรฐาน ECHONET Lite

5. SEOJ: เฟรมที่ระบุถึงต้นทางการสื่อสารตามข้อกำหนดมาตรฐาน ECHONET Lite

6. DEOJ: เฟรมที่ระบุถึงปลายทางการสื่อสารตามข้อกำหนดมาตรฐาน ECHONET Lite

7. ESV: เฟรมที่ระบุถึงประเภทของบริการตามมาตรฐาน ECHONET Lite เช่น WRITE READ หรือ NOTIFY เป็นต้น

8. OPC: เฟรมที่ระบุถึงจำนวนของบริการประเภทต่าง ๆ ตัวอย่างเช่น ถ้าจำนวนบริการมีถึง EPC3 PDC3 และ EDT3 ค่าของ OPC ที่ได้จะเป็น 0x03 เป็นต้น



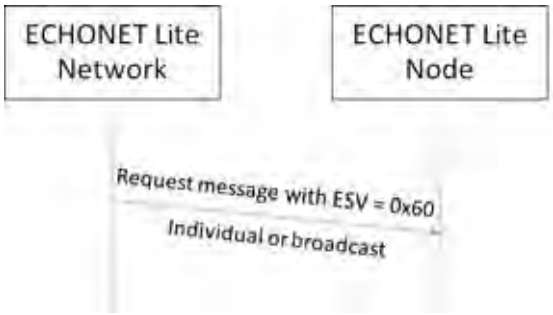
รูปที่ 2.10: รูปแบบเฟรมตามมาตรฐาน ECHONET Lite [11]

9. EPC: เฟรมที่ระบุถึงคุณลักษณะของการทำงานตามประเภทของบริการนั้น ๆ
10. PDC: เฟรมที่บ่งบอกถึงขนาด (ไบต์) ของเฟรม EDT ตัวอย่างเช่น ถ้าขนาดของเฟรม EDT เท่ากับ 2 ไบต์ ค่าของ PDC จะเท่ากับ 0x02
11. EDT: เฟรมที่แสดงถึงค่าที่จะถูกดำเนินการตามฟังก์ชันการทำงานนั้น ๆ

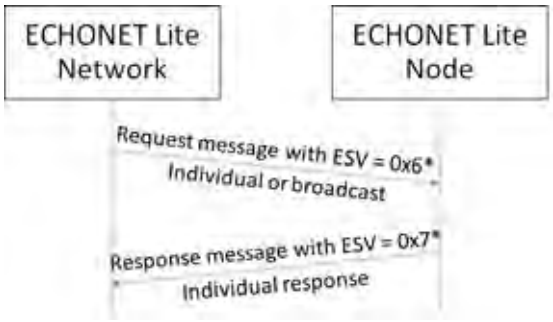
2.2.6 ลำดับพื้นฐานของการสื่อสารตามมาตรฐาน ECHONET Lite

ลำดับการสื่อสารหลักในการสื่อสารระหว่างมิดเดิลแวร์การสื่อสารมาตรฐาน ECHONET Lite และโครงข่าย ECHONET Lite หรือเรียกว่า ลำดับพื้นฐาน (basic sequence) ถูกใช้กำหนดลำดับข้อความการสื่อสารเพื่อเชื่อมต่อไปยังโครงข่าย ECHONET Lite และควบคุมอุปกรณ์ภายในโครงข่าย แบ่งตามกรณีหลัก ๆ ดังนี้

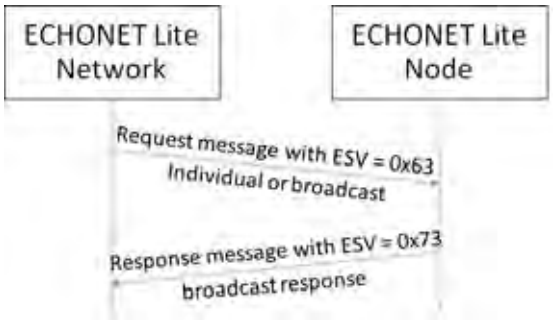
1. ลำดับพื้นฐานกรณีที่โนด ECHONET Lite รับข้อความการร้องขอจากการร้องขอส่วนบุคคล (individual request) หรือการร้องขอแบบแพร่สัญญาณ (broadcast request) โดยไม่จำเป็นต้องมีข้อความตอบกลับ (ESV = 0x60) แสดงดังรูปที่ 2.11
2. ลำดับพื้นฐานกรณีที่โนด ECHONET Lite รับข้อความการร้องขอจากการร้องขอส่วนบุคคลหรือการร้องขอแบบแพร่สัญญาณโดยจำเป็นต้องมีข้อความตอบกลับ (ESV = 0x61 ESV = 0x62 หรือ ESV = 0x6E) แสดงดังรูปที่ 2.12
3. ลำดับพื้นฐานกรณีที่โนด ECHONET Lite รับข้อความการร้องขอการแจ้งเตือน (notification) จากการร้องขอส่วนบุคคล หรือการร้องขอแบบแพร่สัญญาณโดยจำเป็นต้องมีข้อความตอบกลับ (ESV = 0x63) แสดงดังรูปที่ 2.13



รูปที่ 2.11: ลำดับพื้นฐานกรณีรับข้อความการร้องขอแบบไม่มีการตอบกลับ (ESV = 0x60) [11]

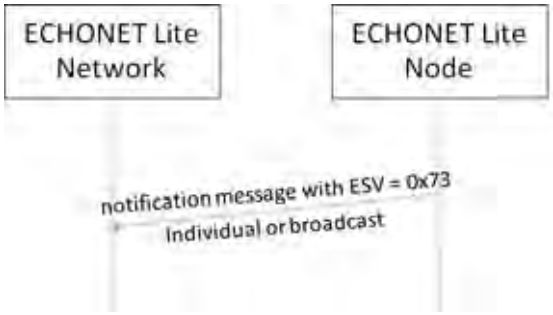


รูปที่ 2.12: ลำดับพื้นฐานกรณีรับข้อความการร้องขอแบบมีการตอบกลับ (ESV = 0x6*) (*: 1 2 หรือ E) [11]



รูปที่ 2.13: ลำดับพื้นฐานกรณีรับข้อความการร้องขอการแจ้งเตือนแบบมีการตอบกลับ (ESV = 0x63) [11]

4. ลำดับพื้นฐานสำหรับการแจ้งเตือนอัตโนมัติ (autonomous notification) โดยมีการแจ้งเตือนไปยังโครงข่าย ECHONET Lite แบบส่วนบุคคล หรือแบบแพร่สัญญาณแสดงดังรูปที่ 2.14



รูปที่ 2.14: ลำดับพื้นฐานกรณีรับข้อความการแจ้งเตือนอัตโนมัติ (ESV = 0x73) [11]

2.2.7 การกำหนดค่าในคลาสอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite (home air conditioner class) [18]

มาตรฐาน ECHONET Lite ได้มีการกำหนดคลาสของอุปกรณ์ชนิดต่าง ๆ เพื่อให้ง่ายต่อการจัดการและควบคุมอุปกรณ์ภายในโครงข่าย ECHONET Lite ซึ่งคลาสที่ถูกนำมาใช้ในงานวิจัยนี้คือคลาส home air conditioner สำหรับใช้ในการควบคุมอุปกรณ์เครื่องปรับอากาศที่รองรับมาตรฐาน ECHONET Lite โดยมีการระบุค่าต่าง ๆ ตามโพรโทคอล ECHONET Lite ที่ใช้ในการสื่อสารภายในโครงข่าย ECHONET Lite ดังนี้

1. class group code: 0x01
2. class code: 0x30
3. instance code: 0x01–0x7F

และค่าอื่น ๆ ที่ถูกใช้งานตามโพรโทคอล ECHONET Lite เพื่อควบคุมอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ในโหมดการใช้งานแต่ละโหมดถูกอธิบายดังรูปที่ 2.15-2.20

Property name	EPC	Contents of property Value range (decimal notation)	Data type	Data size	Unit	Access rule	Mandatory	Announcement at status change	Remark
Operation status	0x80	This property indicates the ON/OFF status. ON=0x30, OFF=0x31	unsigned char	1 byte	—	Set Get	○ ○	○	
Operation power-saving	0x8F	Used to specify the power-saving operation mode and to acquire the current setting. power saving mode = 0x41 normal mode = 0x42	unsigned char	1 byte	–	Set/Get	○	○	
Operation mode setting	0xB0	Used to specify the operation mode ("automatic," "cooling," "heating," "dehumidification," "air circulator" or "other"), and to acquire the current setting. The following values shall be used: Automatic: 0x41 Cooling: 0x42 Heating: 0x43 Dehumidification: 0x44 Air circulator: 0x45 Other: 0x40	unsigned char	1 byte	–	Set/Get	○	○	
Automatic temperature control setting	0xB1	Used to specify whether or not to use the automatic temperature control function, and to acquire the current setting. Automatic = 0x41 Non-automatic = 0x42	unsigned char	1 byte	–	Set/Get			
Normal/high-speed/silent operation setting	0xB2	Used to specify the type of operation ("normal," "high-speed" or "silent"), and to acquire the current setting. Normal operation: 0x41 High-speed operation: 0x42 Silent operation: 0x43	unsigned char	1 byte	–	Set/Get			
Set temperature value	0xB3	Used to set the temperature and to acquire the current setting. 0x00–0x32 (0–50°C)	unsigned char	1 byte	°C	Set/Get	○		
Set value of relative humidity in dehumidifying mode	0xB4	Used to set the relative humidity for the "dehumidification" mode and to acquire the current setting. 0x00–0x64 (0–100%)	unsigned char	1 byte	%	Set/Get			
Set temperature value in cooling mode	0xB5	Used to set the temperature for the "cooling" mode and to acquire the current setting. 0x00–0x32 (0–50°C)	unsigned char	1 byte	°C	Set/Get			
Set temperature value in heating mode	0xB6	Used to set the temperature for the "heating" mode and to acquire the current setting.	unsigned char	1 byte	°C	Set/Get			

รูปที่ 2.15: การกำหนดค่าต่าง ๆ ในโหมดการใช้งานแต่ละโหมดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite [18]

		0x00–0x32 (0–50°C)							
Set temperature value in dehumidifying mode	0xB7	Used to set the temperature for the "dehumidification" mode and to acquire the current setting.	unsigned char	1 byte	°C	Set/Get			
		0x00–0x32 (0–50°C)							
Rated power consumption	0xB8	Rated power consumption in each operation mode of cooling/heating/dehumidifying/blast	unsigned short × 4	8 bytes	W	Get			
		0x0000–0xFFFD (0–65533W) Cooling: heating: dehumidifying: blast							
Measured value of current consumption	0xB9	Measured value of current consumption	unsigned short	2 bytes	0.1A	Get			
		0x0000–0xFFFD (0–6553.3A)							
Measured value of room relative humidity	0xBA	Measured value of room relative humidity	unsigned char	1 byte	%	Get			
		0x00–0x64 (0–100°C)							
Measured value of room temperature	0xBB	Measured value of room temperature	signed char	1 byte	°C	Get	○		
		0x80–0x7D (-127–125°C)							
Set temperature value of user remote control	0xBC	Set temperature value of user remote control	unsigned char	1 byte	°C	Get			
		0x00–0x32 (0–50°C)							
Measured cooled air temperature	0xBD	This property indicates the measured cooled air temperature at the outlet.	signed char	1 byte	°C	Get			
		0x81–0x7D (-127–125°C)							
Measured outdoor air temperature	0xBE	This property indicates the measured outdoor air temperature.	signed char	1 byte	°C	Get			
		0x81–0x7D (-127–125°C)							
Relative temperature setting	0xBF	Used to set the relative temperature relative to the target temperature for an air conditioner operation mode, and to acquire the current setting.	unsigned char	1 byte	0.1 °C	Set/Get			
		0x81–0x7D (-12.7°C–12.5°C)							
Air flow rate setting	0xA0	Used to specify the air flow rate or use the function to automatically control the air flow rate, and to acquire the current setting. The air flow rate shall be selected from among the 8 predefined levels.	unsigned char	1 byte	–	Set/Get	○	○	
		Automatic air flow rate control function used = 0x41 Air flow rate = 0x31–0x38							
Automatic control of air flow direction setting	0xA1	Used to specify whether or not to use the automatic air flow direction control function, to specify the plane(s) (vertical and/or horizontal) in which the automatic air flow direction control function is to be used, and to acquire the current setting.	unsigned char	1 byte	–	Set/Get			
		Automatic = 0x41, non-automatic = 0x42, automatic (vertical) = 0x43, automatic (horizontal) = 0x44							
Automatic swing of air flow setting	0xA3	Used to specify whether or not to use the automatic air flow swing function, to specify the plane(s) (vertical and/or horizontal) in which the automatic air flow swing function is to be used, and to acquire the current setting.	unsigned char	1 byte	–	Set/Get			

รูปที่ 2.16: การกำหนดค่าต่าง ๆ ในโหมดการใช้งานแต่ละโหมดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite (ต่อ) [18]

		Automatic air flow swing function not used = 0x31, used (vertical) = 0x41, used (horizontal) = 0x42, used (vertical and horizontal) = 0x43							
Air flow direction (vertical) setting	0xA4	Used to specify the air flow direction in the vertical plane by selecting a pattern from among the 5 predefined patterns, and to acquire the current setting.	unsigned char	1 byte	–	Set/Get			
		Uppermost = 0x41, lowermost = 0x42, central = 0x43, midpoint between uppermost and central = 0x44, midpoint between lowermost and central = 0x45							
Air flow direction (horizontal) setting	0xA5	Used to specify the air flow direction(s) in the horizontal plane by selecting a pattern from among the 31 predefined patterns, and to acquire the current setting.	unsigned char	1 byte	–	Set/Get			
		Rightward = 0x41, leftward = 0x42, central = 0x43, rightward and leftward = 0x44 (for a full list of the predefined patterns, see the table in the subsection defining the detailed requirements for this property.							
Special state	0xAA	This property indicates if the air conditioner is in a "special" state (i.e. the "defrosting," "preheating," or "heat removal" state).	unsigned char	1 byte	–	Get			
		"Normal operation" state = 0x40, "Defrosting" state = 0x41, "Preheating" state = 0x42, "Heat removal" state = 0x43							
Non-priority state	0xAB	Used to indicate when the air conditioner is in a "non-priority" state.	unsigned char	1 byte	–	Get			
		"Normal operation" state = 0x40, "Non-priority" state = 0x41							
Ventilation function setting	0xC0	Used to specify whether or not to use the ventilation function, to specify the ventilation direction, and to acquire the current setting.	unsigned char	1 byte	–	Set/Get			
		Ventilation function ON (outlet direction) = 0x41, ventilation function OFF = 0x42, ventilation function ON (intake direction) = 0x43							
Humidifier function setting	0xC1	Used to specify whether or not to use the humidifier function, and to acquire the current setting.	unsigned char	1 byte	–	Set/Get			
		Humidifier function ON = 0x41, Humidifier function OFF = 0x42							
Ventilation air flow rate setting	0xC2	Used to specify the ventilation air flow rate by selecting a level from among the predefined levels, and to acquire the current setting.	unsigned char	1 byte	–	Set/Get			
		Automatic control of ventilation air flow rate = 0x41, ventilation air flow rate = 0x31–0x38							
Degree of humidification setting	0xC4	Used to specify the degree of humidification to achieve by selecting a level from among the predefined levels, and to acquire the current setting.	unsigned char	1 byte	–	Set /Get			

รูปที่ 2.17: การกำหนดค่าต่าง ๆ ในโหมดการใช้งานแต่ละโหมดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite (ต่อ) [18]

		Automatic control of the degree of humidification = 0x41 Degree of humidification = 0x31–0x38							
Mounted air cleaning method	0xC6	A bitmap indicates mounted method of exercising air cleaning function. Bit 0: Information about electrical dust collection method mounting 0 - Not mounted 1 - Mounted Bit 1: Information about cluster ion method mounting 0 - Not mounted 1 - Mounted	unsigned char	1 byte	–	Get			
Air purifier function setting	0xC7	An 8-byte array used to specify, for each type of air purifier function, whether or not to use the air purifier function and the degree of air purification to achieve with the air purifier function, and to acquire the current settings. Element 0: Indicates whether or not to use the electrical dust collection-based air purifier function. Element 1: Indicates whether or not to use the cluster ion-based air purifier function. Elements 2 to 7: Reserved for future use.	unsigned char × 8	1 byte × 8	–	SetM /GetM Set /Get			
Mounted air refresh method	0xC8	A bitmap indicates mounted method for exercising refresh function. Bit 0: Information about minus ion method mounting 0 - Not mounted 1 - Mounted Bit 1: Information about cluster ion method mounting 0 - Not mounted 1 - Mounted	unsigned char	1 byte	–	Get			
Air refresher function setting	0xC9	An 8-byte array used to specify, for each type of air refresher function, whether or not to use the air refresher function and the degree of air refreshing to achieve with the air refresher function, and to acquire the current settings. Element 0: Indicates whether or not to use the minus ion-based air refresher function. Element 1: Indicates whether or not to use the cluster ion-based air refresher function. Elements 2–7: Reserved for future use.	unsigned char × 8	1 byte × 8	–	SetM /GetM Set /Get			
Mounted self-cleaning method	0xCA	A bitmap indicates mounted method for exercising self-cleaning function. Bit 0: Information about ozone cleaning method mounting 0 - Not mounted 1 - Mounted Bit 1: Information about drying method mounting 0 - Not mounted 1 - Mounted	unsigned char	1 byte	–	Get			

รูปที่ 2.18: การกำหนดค่าต่าง ๆ ในโหมดการใช้งานแต่ละโหมดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite (ต่อ) [18]

Self-cleaning function setting	0xCB	An 8-byte array used to specify, for each type of self-cleaning function, whether or not to use the self-cleaning function and the degree of self-cleaning to achieve with the self-cleaning function, and to acquire the current settings. Element 0: Indicates whether or not to use the ozone-based self-cleaning function. Element 1: Indicates whether or not to use the drying-based self-cleaning function. Elements 2–7: Reserved for future use.	unsigned char × 8	1 byte × 8	–	SetM /GetM Set /Get			
Special function setting	0xCC	Used to specify the "special function" to use, and to acquire the current setting. No setting: 0x40, clothes dryer function: 0x41, condensation suppressor function: 0x42, mite and mold control function: 0x43, active defrosting function: 0x44 0x45–: Reserved for future use.	unsigned char	1 byte	–	Set /Get			
Operation status of components	0xCD	This property indicates the operation status of components of the air conditioner in a bitmap format. Bit 0: Operation status of the compressor: 0: Not operating 1: In operation Bit 1: Operation status of the thermostat: 0: Thermostat OFF 1: Thermostat ON Bits 2–7: Reserved for future use.	unsigned char	1 byte	–	Get			
Thermostat setting override function	0xCE	Used to specify whether or not to allow the air conditioner to operate ignoring its thermostat setting. Normal setting = 0x40, thermostat setting override function ON = 0x41, thermostat setting override function OFF = 0x42	unsigned char	1 byte	–	Set/Get			
Air purification mode setting	0xCF	Used to set the air purification mode setting ON/OFF and to acquire the current setting. Air purification ON=0x41,OFF=0x42	unsigned char	1 byte	–	Set/Get			
ON timer-based reservation setting	0x90	Used to specify whether or not to use the ON timer (time-based reservation function, relative time-based reservation function or both), and to acquire the current setting. Both the time- and relative time-based reservation functions are ON = 0x41, both reservation functions are OFF = 0x42, time-based reservation function is ON = 0x43, relative time-based reservation function is ON = 0x44	unsigned char	1 byte	–	Set/Get			
ON timer setting (time)	0x91	Used to specify the time for the time-based reservation function in the HH:MM format and to acquire the current setting. 0–0x17; 0–0x3B (= 0–23); (= 0–59)	unsigned char × 2	2 bytes	–	Set/Get			

รูปที่ 2.19: การกำหนดค่าต่าง ๆ ในโหมดการใช้งานแต่ละโหมดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite (ต่อ) [18]

ON timer setting (relative time)	0x92	Used to specify the relative time for the relative time-based reservation function in the HH:MM format and to acquire the current setting. 0–0xFF: 0–0x3B (= 0–255): (= 0–59)	unsigned char × 2	2 bytes	–	Set/Get			
OFF timer-based reservation setting	0x94	Used to specify whether or not to use the OFF timer (time-based reservation function, relative time-based reservation function or both), and to acquire the current setting. Both the time- and relative time-based reservation functions are ON = 0x41, both reservation functions are OFF = 0x42, time-based reservation function is ON = 0x43, relative time-based reservation function is ON = 0x44	unsigned char	1 byte	–	Set/Get			
OFF timer setting (time)	0x95	Used to specify the time for the time-based reservation function in the HH:MM format and to acquire the current setting. 0–0x17: 0–0x3B (= 0–23): (= 0–59)	unsigned char × 2	2 bytes	–	Set/Get			
OFF timer setting (relative time)	0x96	Used to specify the relative time for the relative time-based reservation function in the HH:MM format and to acquire the current setting. 0–0xFF: 0–0x3B (= 0–255): (= 0–59)	unsigned char × 2	2 bytes	–	Set/Get			

รูปที่ 2.20: การกำหนดค่าต่าง ๆ ในโหมดการใช้งานแต่ละโหมดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite (ต่อ) [18]

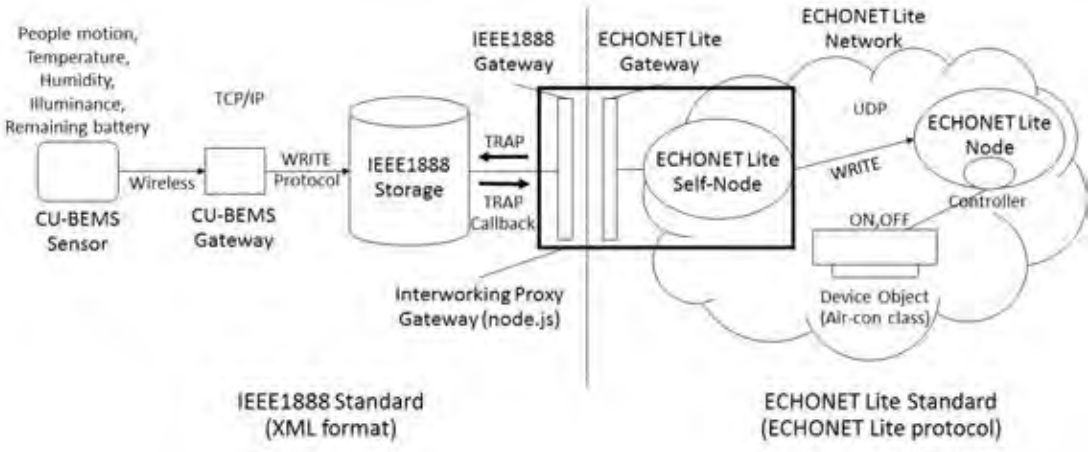
บทที่ 3

โครงสร้างและการทำงานของระบบการทำงานร่วมกันแบบทันที

วิทยานิพนธ์ฉบับนี้นำเสนอการพัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ในระบบการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite โดยมีโครงสร้างการทำงานของระบบดังรูปที่ 3.1 ซึ่งเป็นการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite และรูปที่ 3.2 ซึ่งเป็นการประสานข้อมูลแบบทันทีจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888

3.1 โครงสร้างการทำงานของระบบ

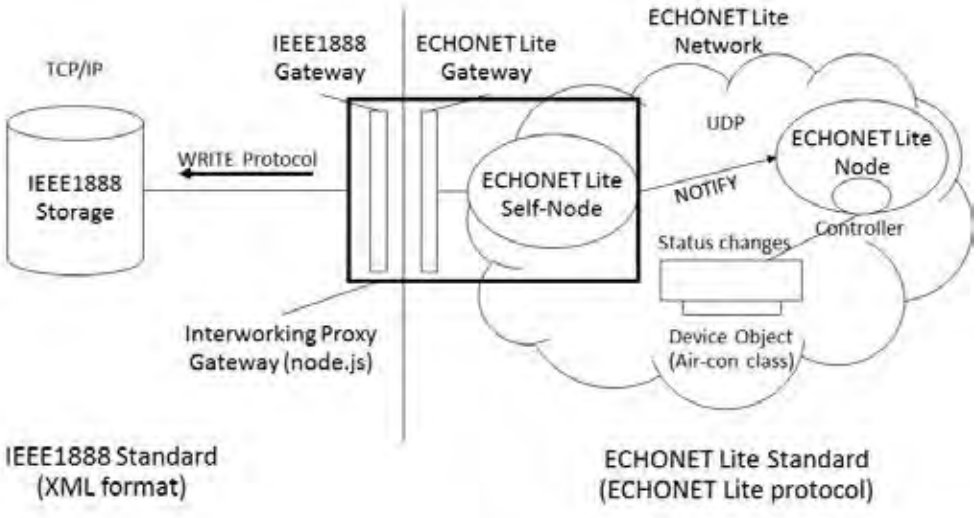
โครงสร้างการทำงานของระบบที่นำเสนอแบ่งออกเป็น 2 โดเมนการทำงาน คือ โดเมนการทำงานในระบบ CU-BEMS ตามมาตรฐาน IEEE1888 และโดเมนการทำงานในระบบจัดการพลังงานภายในบ้านตามมาตรฐาน ECHONET Lite โดยการทำงานร่วมกันระหว่างสองมาตรฐานนี้จำเป็นต้องมีอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เพื่อทำหน้าที่ประสานข้อมูลแบบทันทีระหว่างมาตรฐานทั้งสอง



รูปที่ 3.1: โครงสร้างของระบบที่นำเสนอสำหรับการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite

จากรูปที่ 3.1 การทำงานเริ่มจากโดเมนของมาตรฐาน IEEE1888 ข้อมูลที่ได้จากการตรวจจับโดยตัวรับรู้สภาพแวดล้อมในระบบ CU-BEMS ถูกส่งไปยังเกตเวย์ในระบบ CU-BEMS แบบไร้สายเพื่อแปลงข้อมูลที่ได้มาให้อยู่ในรูปแบบ XML จากนั้นข้อมูลที่อยู่ในรูปแบบ XML จะถูกส่งไปเก็บในหน่วยเก็บข้อมูลด้วยโพรโทคอล WRITE ตามมาตรฐาน IEEE1888 บนโครงข่ายการสื่อสารชนิด TCP/IP ในขณะเดียวกันอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จะส่งการร้องขอ TRAP ซึ่งเป็นโพรโทคอลตามมาตรฐาน IEEE1888 ที่ใช้สำหรับแจ้งเตือนการเปลี่ยนแปลงข้อมูลไปยังหน่วยเก็บข้อมูล และเมื่อข้อมูลในหน่วยเก็บข้อมูลเกิดการเปลี่ยนแปลงจึงมีการตอบกลับข้อความ TRAP callback พร้อมกับค่าการเปลี่ยนแปลงไปยังโดเมนของมาตรฐาน ECHONET Lite ต่อไป

ในการทำงานลำดับถัดไปหลังจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ได้รับข้อความตอบกลับ TRAP callback พร้อมกับค่าการเปลี่ยนแปลงจากหน่วยเก็บข้อมูลระบบ CU-BEMS แล้วจึงทำการแจกแจงข้อมูล (data parsing) ที่ได้รับมาเนื่องจากข้อมูลอยู่ในรูปแบบ XML ก่อนที่จะแปลงรูปแบบของข้อมูลให้อยู่ในรูปแบบ ECHONET Lite เพื่อนำไปใช้ควบคุมอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite โดยการส่งการร้องขอแบบ WRITE ตามมาตรฐาน ECHONET Lite บนโครงข่ายการสื่อสารชนิด UDP ไปยังโนดในโครงข่าย ECHONET Lite ที่พบว่ามีอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite เชื่อมต่ออยู่ และทำการควบคุมการเปิดปิด (ON, OFF) หรือในโหมดอื่น ๆ ไปยังคลาสของอุปกรณ์เครื่องปรับอากาศ (home air conditioner class) ผ่านตัวควบคุม (ECHONET Lite controller) ภายในโนด ECHONET Lite นั้น

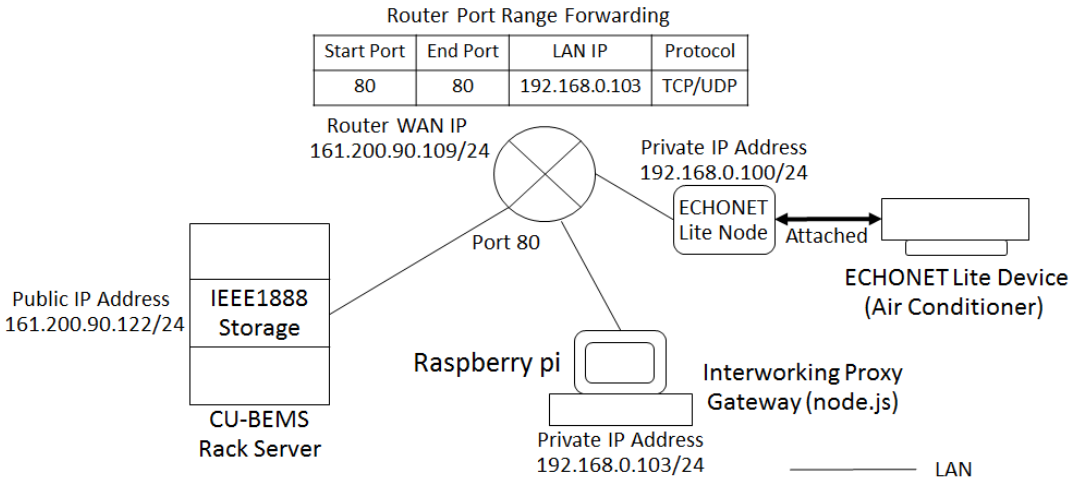


รูปที่ 3.2: โครงสร้างของระบบที่นำเสนอสำหรับการประสานข้อมูลแบบทันทีจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888

และจากรูปที่ 3.2 การทำงานเริ่มจากโดเมนของมาตรฐาน ECHONET Lite โดยอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศไปยังโนดภายในโครงข่าย ECHONET Lite ที่มีอุปกรณ์เครื่องปรับอากาศเชื่อมต่ออยู่บนโครงข่ายการสื่อสารชนิด UDP ด้วยการร้องขอแบบ NOTIFY ตามมาตรฐาน ECHONET Lite เมื่อสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศเกิดการเปลี่ยนแปลง โนด ECHONET Lite จึงตอบกลับไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์พร้อมกับค่าการเปลี่ยนแปลงสถานะ จากนั้นอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จึงทำการแปลงรูปแบบของข้อมูลที่ได้รับมาจากรูปแบบ ECHONET Lite ให้อยู่ในรูปแบบ XML เพื่อส่งไปเก็บยังหน่วยเก็บข้อมูลด้วยโพรโทคอล WRITE ตามมาตรฐาน IEEE1888 บนโครงข่ายการสื่อสารชนิด TCP/IP ซึ่งการร้องขอการแจ้งเตือนแบบ NOTIFY ตามมาตรฐาน ECHONET Lite นั้นสามารถร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะได้ทั้งแบบการร้องขอส่วนบุคคล (individual request) หรือ การร้องขอแบบแพร่สัญญาณ (broadcast request) ภายในโครงข่าย ECHONET Lite นั้น

3.2 การเชื่อมต่อส่วนประกอบในระบบที่นำเสนอ

ส่วนประกอบทางกายภาพของระบบการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite โดยใช้อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เป็นตัวประสานข้อมูลถูกเชื่อมต่อกันตามรูปที่ 3.3



รูปที่ 3.3: การเชื่อมต่อระหว่างส่วนประกอบในระบบที่นำเสนอ

1. หน่วยเก็บข้อมูลในระบบ CU-BEMS: หน่วยเก็บข้อมูลตามมาตรฐาน IEEE1888 ในระบบ CU-BEMS มีหน้าที่เก็บข้อมูลที่ได้จากตัวรับรู้สภาพแวดล้อมในระบบ CU-BEMS เพื่อนำไปแสดงผลหรือแจ้งเตือนข้อมูลด้วยโพรโทคอล TRAP ตามมาตรฐาน IEEE1888 หน่วยเก็บข้อมูลในระบบ CU-BEMS มีเลขที่อยู่ไอพี คือ 161.200.90.122 แบบสาธารณะ (public) และมีช่องทางเข้าออก (port) คือ 80 เนื่องจากหน่วยเก็บข้อมูลใช้โพรโทคอล http ในการส่งข้อความบนโครงข่ายอินเทอร์เน็ต

2. อุปกรณ์จัดเส้นทาง: อุปกรณ์จัดเส้นทาง (router) มีหน้าที่กำหนดเส้นทางภายในโครงข่ายบริเวณเฉพาะที่ (local area network, LAN) มีช่องทางขาเข้าจากโครงข่ายบริเวณกว้าง (wide area network, WAN) ด้วยเลขที่อยู่ไอพี 161.200.90.109 แบบสาธารณะ และมีการกำหนดการส่งต่อช่องทางเข้าออก (port forwarding) ตามรูปที่ 4.2 เนื่องจากหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 ในระบบ CU-BEMS มีเลขที่อยู่ไอพีแบบสาธารณะ แต่โนดในมาตรฐาน ECHONET Lite มีเลขที่อยู่ไอพีแบบส่วนบุคคล (private) ในการกำหนดการส่งต่อช่องทางเข้าออกนี้เพื่อให้โครงข่ายบริเวณกว้างสามารถสื่อสารกับโครงข่ายบริเวณเฉพาะที่ที่มีเลขที่อยู่ไอพีเฉพาะเจาะจง (specified IP address) ผ่านช่องทางเข้าออก 80 ได้

3. อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์: อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เป็นเกตเวย์ซึ่งในงานวิจัยนี้ใช้ระบบสมองกลฝังตัว (raspberry pi) ทำหน้าที่ประสานข้อมูลระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เชื่อมต่อกับอุปกรณ์จัดเส้นทางแบบอีเทอร์เน็ตด้วยเลขที่อยู่ไอพี คือ 192.168.0.103 แบบส่วนบุคคล สาเหตุที่เลขที่อยู่

ไอพีเป็นแบบส่วนบุคคลเนื่องจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ได้มีการสร้างโนดมาตรฐาน ECHONET Lite ขึ้นมาอีกหนึ่งโนด (ECHONET Lite self-node) ซึ่งต้องอยู่ภายในโครงข่ายบริเวณเฉพาะที่เดียวกันจึงสามารถสื่อสารภายในโครงข่าย ECHONET Lite เดียวกันด้วยเลขที่อยู่ไอพีแบบส่วนบุคคลได้ โดยอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ทำงานด้วยโปรแกรม node.js [19] เพื่อการทำงานในโครงข่ายการสื่อสารอย่างมีประสิทธิภาพ

4. โนดมาตรฐาน ECHONET Lite: โนดมาตรฐาน ECHONET Lite มีหน้าที่สื่อสารกันระหว่างโนดภายในโครงข่าย ECHONET Lite และควบคุมอุปกรณ์มาตรฐาน ECHONET Lite ที่ถูกเชื่อมต่ออยู่ โนด ECHONET Lite มีเลขที่อยู่ไอพี คือ 192.168.0.100 แบบส่วนบุคคลและถูกเชื่อมต่อไปยังอุปกรณ์จัดเส้นทาง โดยอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite และการเชื่อมต่อระหว่างโนด ECHONET Lite กับอุปกรณ์เครื่องปรับอากาศนี้ได้รับการสนับสนุนผ่านความร่วมมือจาก ECHONET Consortium ประเทศญี่ปุ่น

5. อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite: อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ถูกเชื่อมต่อแบบแนบติดกับโนด ECHONET Lite ซึ่งภายในโนด ECHONET Lite มีตัวควบคุมทำหน้าที่ควบคุมอุปกรณ์เครื่องปรับอากาศ ECHONET Lite ที่เชื่อมต่ออยู่ อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ทำงานในโหมดต่าง ๆ ตามชุดคำสั่งที่ถูกกำหนดไว้ในคู่มือการใช้งานมาตรฐาน ECHONET Lite คลาสอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite (home air conditioner class)

3.3 แผนภาพทางเวลาสำหรับการทดสอบระบบในเบื้องต้น

อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ทำหน้าที่ประสานข้อมูลระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite และใช้ node.js เป็นตัวโปรแกรมการทำงานบนภาษาจาวาสคริปต์ (javascript) โดยในการประสานข้อมูลระหว่างสองมาตรฐานมีลำดับการสื่อสารระหว่าง 3 ส่วนประกอบ คือ หน่วยเก็บข้อมูลมาตรฐาน IEEE1888 อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ และโนดมาตรฐาน ECHONET Lite แสดงดังรูปที่ 3.4 และรูปที่ 3.5

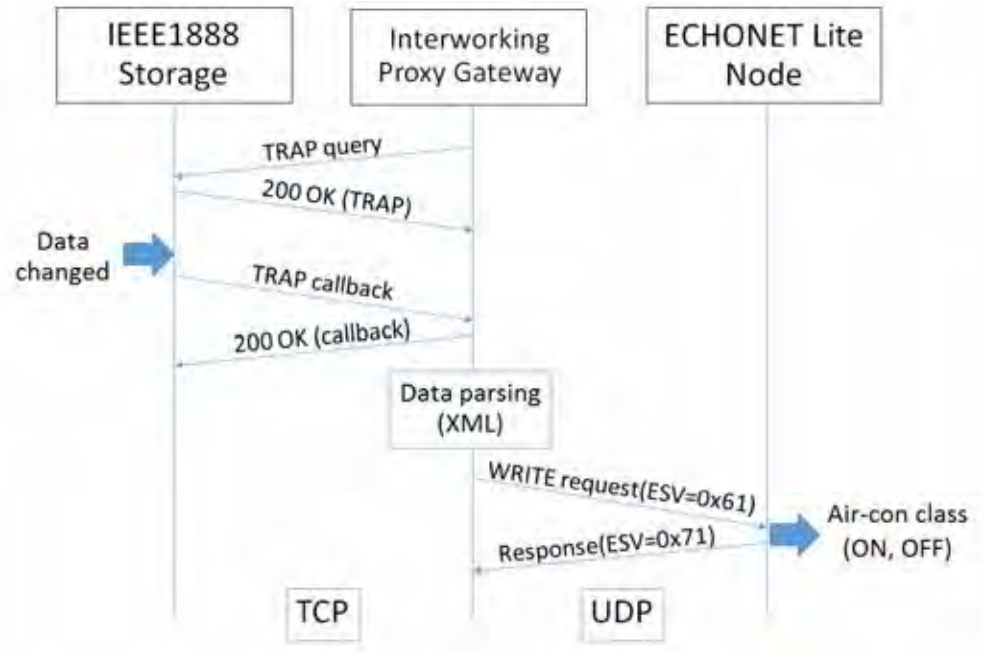
ลำดับการสื่อสารดังรูปที่ 3.4 เป็นลำดับการสื่อสารสำหรับการประสานข้อมูลจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite แบบทันทีของระบบที่นำเสนอเพื่อแจ้งเตือนการเปลี่ยนแปลงข้อมูลในหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 และควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite มีลำดับการสื่อสารระหว่าง 3 ส่วนประกอบ อธิบายได้ดังนี้

1. อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งข้อความ TRAP query ไปยังหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 เพื่อร้องขอการแจ้งเตือนการเปลี่ยนแปลงข้อมูลด้วยการกำหนด point ID

2. เมื่อหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 ได้รับข้อความการร้องขอ TRAP แล้ว หน่วยเก็บข้อมูลจะส่งข้อความตอบกลับ TRAP 200 OK ไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เพื่อยืนยันการรับข้อความ

3. ถ้าข้อมูลในหน่วยเก็บข้อมูลจากการร้องขอ TRAP เกิดการเปลี่ยนแปลง หน่วยเก็บข้อมูลจะส่งข้อความ TRAP callback ไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์พร้อมกับค่าการเปลี่ยนแปลง

4. อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งข้อความตอบกลับ TRAP callback 200 OK ไปยังหน่วยเก็บข้อมูลเพื่อยืนยันว่าได้รับข้อความแล้ว

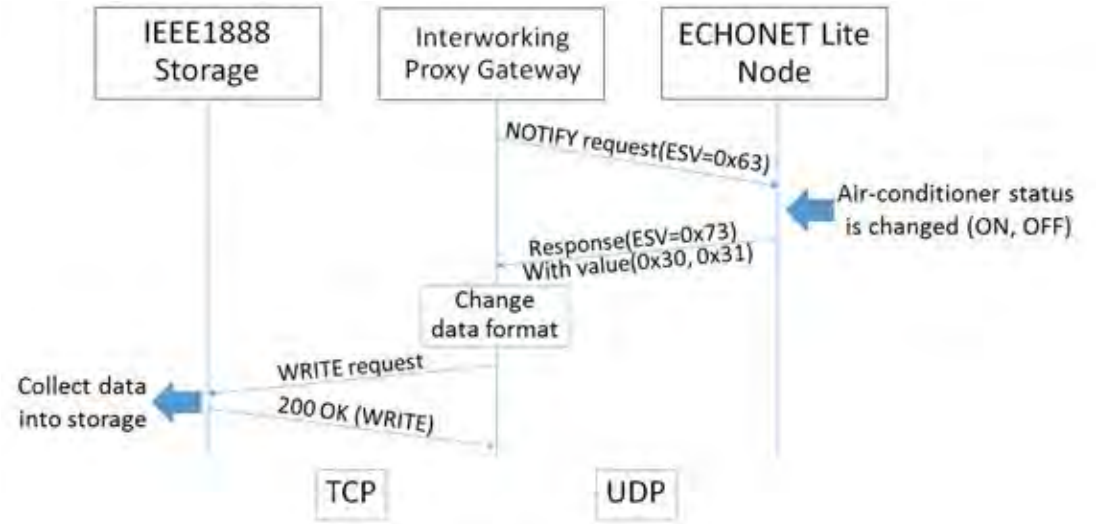


รูปที่ 3.4: ลำดับการสื่อสารสำหรับการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite

5. เนื่องจากข้อมูลที่ได้รับมาอยู่ในรูปแบบ XML ดังนั้นอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จึงต้องทำการแจกแจงข้อมูล (data parsing) และแปลงรูปแบบของข้อมูลให้อยู่ในรูปแบบ ECHONET Lite ก่อนที่จะนำข้อมูลไปใช้
6. อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ตรวจสอบข้อมูลที่ถูกแปลงมาแล้ว และนำไปใช้โดยส่งข้อความการร้องขอแบบ WRITE พร้อมกับระบุค่า ESV เท่ากับ 0x61 ไปยังโนดมาตรฐาน ECHONET Lite
7. จากนั้นโนด ECHONET Lite จึงทำการควบคุมการเปิดปิดไปยังอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ที่เชื่อมต่ออยู่ตามค่าของข้อมูลที่ได้รับมา (ESV = 0x61)
8. โนด ECHONET Lite ส่งข้อความตอบกลับไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์โดยกำหนดค่า ESV เท่ากับ 0x71 เป็นอันสิ้นสุดกระบวนการสื่อสาร

และลำดับการสื่อสารสำหรับการประสานข้อมูลจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 แบบทันทีเพื่อร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่อง-ปรับอากาศมาตรฐาน ECHONET Lite และส่งข้อมูลไปเก็บยังหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 แสดงดังรูปที่ 3.5

1. อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งข้อความร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศแบบ NOTIFY ไปยังโนด ECHONET Lite ด้วยการระบุค่า ESV เท่ากับ 0x63
2. เมื่อสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศเกิดการเปลี่ยนแปลง โนด ECHONET



รูปที่ 3.5: ลำดับการสื่อสารสำหรับการประสานข้อมูลแบบทันทีจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888

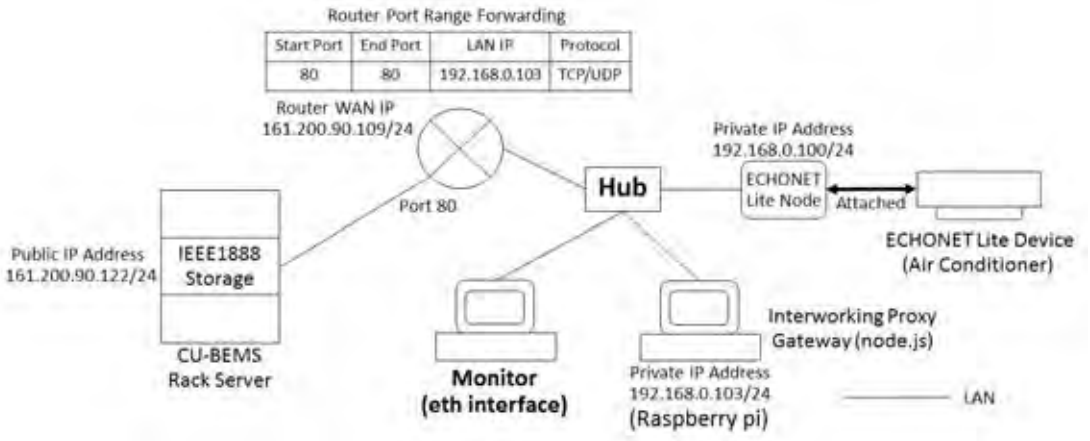
Lite จะตอบกลับข้อความด้วยค่า ESV เท่ากับ 0x73 ไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์พร้อมกับค่าการเปลี่ยนแปลง

3. หลังจากที่อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ได้รับข้อมูลแล้ว จึงทำการแปลงรูปแบบของข้อมูลจากรูปแบบของ ECHONET Lite เป็นรูปแบบ XML ตามมาตรฐาน IEEE1888
4. อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ใช้ข้อมูลที่อยู่ในรูปแบบ XML เพื่อส่งไปเก็บยังหน่วยเก็บข้อมูลด้วยโพรโทคอล WRITE ตามมาตรฐาน IEEE1888
5. หน่วยเก็บข้อมูล IEEE1888 ตอบกลับข้อความ 200 OK ไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เพื่อยืนยันการรับข้อความ

บทที่ 4

การทดสอบและการประเมินผลของระบบการทำงานร่วมกันแบบทันที

การทดสอบเบื้องต้นที่ถูกนำเสนอในวิทยานิพนธ์ คือ การทดสอบการประสานข้อมูลแบบทันทีในระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ทั้งการประสานข้อมูลจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite และการประสานข้อมูลจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 โดยการพัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ให้สามารถแจ้งเตือนการเปลี่ยนแปลงข้อมูลที่ได้จากตัวรับรู้ระบบ CU-BEMS ในหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 เพื่อส่งไปควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite และสามารถแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite เพื่อส่งค่าไปเก็บยังหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 ในระบบ CU-BEMS ได้ ในการทดสอบนี้ยังมีการเชื่อมต่ออุปกรณ์รวมสายสัญญาณ (hub) เพิ่มหลังจากอุปกรณ์จัดเส้นทางเพื่อสามารถตรวจจับข้อความที่อยู่บนโครงข่ายบริเวณเฉพาะที่เดียวกันจากอุปกรณ์ตรวจจับ (monitor) ด้วยอินเตอร์เฟสอีเทอร์เน็ต (Ethernet) ตามรูปที่ 4.1



รูปที่ 4.1: การเชื่อมต่อของอุปกรณ์รวมสายสัญญาณที่ถูกเชื่อมต่อเพิ่มในระบบทดสอบ

4.1 ส่วนประกอบที่ใช้ในการทดสอบระบบ

ส่วนประกอบในการทดสอบระบบการทำงานร่วมกันระหว่างสองมาตรฐานนี้ประกอบด้วย อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ หน่วยเก็บข้อมูลมาตรฐาน IEEE1888 ตัวรับรู้ CU-BEMS โนด ECHONET Lite อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite และอุปกรณ์จัดเส้นทาง ซึ่งหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 และตัวรับรู้ในระบบ CU-BEMS ได้มีการติดตั้งจริงไว้ในห้องเซิร์ฟเวอร์สำหรับใช้ทดสอบระบบ CU-BEMS ตามวัตถุประสงค์ของโครงการอยู่แล้ว และได้มีการติดตั้งโนด ECHONET Lite พร้อมกับอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite

เพิ่มไว้สำหรับการทดสอบในห้องเดียวกัน คุณสมบัติของส่วนประกอบแต่ละส่วนที่ใช้ทดสอบในระบบถูกแสดงดังตารางที่ 4.1

ตารางที่ 4.1: รายละเอียดส่วนประกอบแต่ละส่วนที่ใช้ในการทดสอบ

ส่วนประกอบ	รายละเอียด
อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ รูปที่ 4.2	ตราสินค้า Raspberry Pi Model B+ Version 1.2 คุณลักษณะ CPU 700 MHz SDRAM 512 MB ARM1176JZF-S Core ระบบปฏิบัติการ RASPBIAN JESSIE LITE ซอฟต์แวร์ node.js เลขที่อยู่ไอพี 192.168.0.103 (private)
โนด ECHONET Lite รูปที่ 4.3	ตราสินค้า Mitsubishi Electric (MAC-894IF) เลขที่อยู่ไอพี 192.168.0.100 (private)
หน่วยเก็บข้อมูล CU-BEMS รูปที่ 4.4	ตราสินค้า IBM คุณลักษณะ Intel Xeon (virtual machine) dual core 2.40 GHz x 2 processor, 8 GB RAM ระบบปฏิบัติการ Ubuntu 12.04 LTS, 64-bit ซอฟต์แวร์ Apache เลขที่อยู่ไอพี 161.200.90.122 (public)
ตัวรับรู้ CU-BEMS รูปที่ 4.5	คุณลักษณะ รองรับมาตรฐาน IEEE1888 เลขที่อยู่ไอพี ไม่มี
อุปกรณ์จัดเส้นทาง รูปที่ 4.6	ตราสินค้า Tenda คุณลักษณะ 300Mbps 2.4GHz เลขที่อยู่ไอพี 192.168.0.1/24
อุปกรณ์เครื่องปรับอากาศ มาตรฐาน ECHONET Lite รูปที่ 4.7	ตราสินค้า Mitsubishi Electric คุณลักษณะ มาตรฐาน ECHONET Lite เลขที่อยู่ไอพี ไม่มี

4.2 การทดสอบการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite

กระบวนการทดสอบระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite สำหรับการประสานข้อมูลแบบทันทีโดยการพัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ถูกทดสอบอย่างเป็นลำดับขั้นตอน เพื่อนำไปสู่ผลการทดสอบที่สามารถตรวจสอบได้ เริ่มจากการทดสอบการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite โดยอิน

รูปที่ 4.2: อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์

รูปที่ 4.3: โนด ECHONET Lite

รูปที่ 4.4: หน่วยเก็บข้อมูล CU-BEMS

รูปที่ 4.5: ตัวรับรู้ CU-BEMS

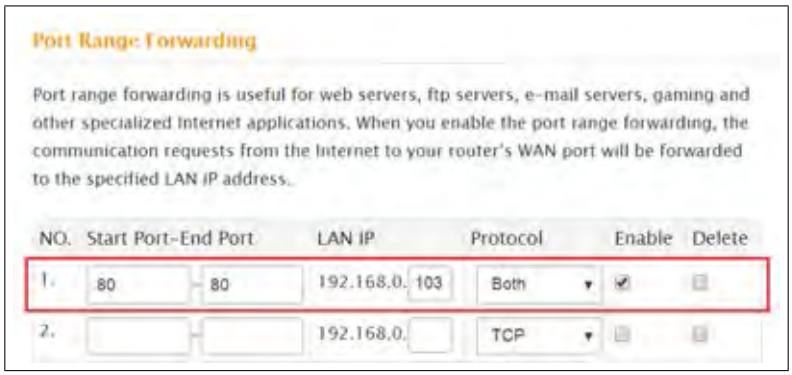
รูปที่ 4.6: อุปกรณ์จัดเส้นทาง

รูปที่ 4.7: อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite

เตอร์เวิร์กกิงพร็อกซีเกตเวย์ร้องขอการแจ้งเตือนการเปลี่ยนแปลงข้อมูล TRAP ไปยังหน่วยเก็บข้อมูลจากมาตรฐาน IEEE1888 และนำข้อมูลไปใช้ควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite

4.2.1 การทดสอบการเชื่อมต่อระหว่างส่วนประกอบที่ใช้ในระบบทดสอบ

การทดสอบการประสานข้อมูลนี้เริ่มจากการตั้งค่าและเชื่อมต่อส่วนประกอบต่าง ๆ เข้าด้วยกัน โดยใช้อุปกรณ์จัดเส้นทางกำหนดโครงข่ายให้เป็นโครงข่ายบริเวณเฉพาะที่ จากนั้นเชื่อมต่อและตั้งค่าส่วนประกอบให้อยู่ภายในโครงข่ายเดียวกันเพื่อให้ส่วนประกอบแต่ละส่วนสามารถสื่อสารถึงกันได้ดังรูปที่ 4.8



รูปที่ 4.8: การกำหนดการส่งต่อช่องทางเข้าออกบนอุปกรณ์จัดเส้นทาง

ตั้งค่าเลขที่อยู่ไอพีของอุปกรณ์จัดเส้นทางเป็น 192.168.0.1 และกำหนดการส่งต่อช่องทางเข้า-ออกเป็นเลขที่อยู่ไอพี 192.168.0.103 ช่องทางเข้าออก 80 เพื่อให้สามารถสื่อสารถึงกันได้ระหว่างเลขที่อยู่ไอพีแบบสาธารณะและเลขที่อยู่ไอพีแบบส่วนบุคคลดังรูปที่ 4.8

4.2.2 การทดสอบการร้องขอเพื่อแจ้งเตือนการเปลี่ยนแปลงข้อมูลด้วยโพรโทคอล TRAP

หลังจากส่วนประกอบแต่ละส่วนในระบบถูกเชื่อมต่อถึงกันภายในโครงข่ายแล้ว จึงสามารถทดสอบส่งการร้องขอการแจ้งเตือนการเปลี่ยนแปลงข้อมูลด้วยโพรโทคอล TRAP จากอินเตอร์เวิร์กกิง-พร็อกซีเกตเวย์ด้วยโปรแกรม trap_query.js ดังภาคผนวก ก ไปยังหน่วยเก็บข้อมูล IEEE1888 โดยระบุค่า ttl ซึ่งเป็นอายุของข้อความ TRAP เท่ากับ 5 นาที ที่ point ID คือ http://bems.ee.eng.chula.ac.th/eng4/fl13/north/room_server/z1/sensor1/monitor/pir (รูปที่ 4.9) เพื่อให้หน่วยเก็บข้อมูลตอบกลับด้วยข้อความ TRAP callback ไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เมื่อข้อมูลในหน่วยเก็บข้อมูลเกิดการเปลี่ยนแปลง โดยกำหนดเลขที่อยู่ไอพีของผู้เรียกกลับข้อมูลและผู้เรียกกลับการควบคุมเป็น 161.200.90.109 จากผลการทดสอบได้รูปแบบของข้อความการร้องขอ TRAP จากการตรวจจับด้วยโปรแกรม wireshark บนอุปกรณ์ตรวจจับเป็นไปดังรูปที่ 4.10

รูปที่ 4.10 แสดงข้อความการร้องขอการแจ้งเตือน TRAP จากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ไปยังหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 ในระบบ CU-BEMS ด้วยการระบุ point ID เป็น http://bems.ee.eng.chula.ac.th/eng4/fl13/north/room_server/z1/sensor1/monitor/pir สำหรับการแจ้งเตือนการเปลี่ยนแปลงข้อมูลตัวรับรู้การเคลื่อนไหวของคน ณ จุดที่ตัวรับรู้ระบบ CU-BEMS ถูกติดตั้งไว้ภายในห้องทดสอบ เมื่อเกิดการเปลี่ยนแปลงของข้อมูลตัวรับรู้ในหน่วยเก็บข้อมูล

http://bems.ee.eng.chula.ac.th/eng4/fl13/north/room_server/z1/sensor1/monitor/humidity	2017-03-27T17:46:01.000+07:00	62.9
http://bems.ee.eng.chula.ac.th/eng4/fl13/north/room_server/z1/sensor1/monitor/illuminance	2017-03-27T17:46:01.000+07:00	0.0
http://bems.ee.eng.chula.ac.th/eng4/fl13/north/room_server/z1/sensor1/monitor/pir	2017-03-27T17:46:01.000+07:00	OFF
http://bems.ee.eng.chula.ac.th/eng4/fl13/north/room_server/z1/sensor1/monitor/temperature	2017-03-27T17:46:01.000+07:00	23.85

รูปที่ 4.9: point ID สำหรับเก็บค่าที่ได้จากตัวรับรู้ชนิด pir ของระบบ CU-BEMS ในห้องทดสอบ

```
Frame 239: 838 bytes on wire (6704 bits), 838 bytes captured (6704 bits)
Ethernet II, Src: IntelCor_c3:6e:2e (00:21:6a:c3:6e:2e), Dst: TendaTec_2f:9d:80 (c8:3a:35:2f:9d:80)
Internet Protocol Version 4, Src: 192.168.0.103 (192.168.0.103), Dst: 161.200.90.122 (161.200.90.122)
Transmission Control Protocol, Src Port: 52999 (52999), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 772
Hypertext Transfer Protocol
eXtensible Markup Language
  <?xml
  <soapenv:Envelope
      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body>
      <ns2:queryRQ
          xmlns:ns2="http://soap.fiap.org/">
        <transport
            xmlns="http://gutp.jp/fiap/2009/11/">
          <header>
            <query
                id="89cefb00-aa4e-4602-bc58-9ce4fc1cb0e3"
                type="stream"
                ttl="300"
                callbackData="http://161.200.90.109"
                callbackControl="http://161.200.90.109">
              <key
                id="http://bems.ee.eng.chula.ac.th/eng4/fl13/north/room_server/z1/sensor1/monitor/pir"
                attrName="value"
                trap="changed"/>
            </query>
          </header>
        </transport>
      </ns2:queryRQ>
    </soapenv:Body>
  </soapenv:Envelope>
```

รูปที่ 4.10: รูปแบบของข้อความการร้องขอ TRAP ที่ถูกตรวจจับโดยโปรแกรม wireshark

ระบบ CU-BEMS หน่วยเก็บข้อมูลจะตอบกลับข้อมูลที่เปลี่ยนแปลงนั้นไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ จากรูปที่ 4.10 ข้อความการร้องขอการแจ้งเตือน TRAP ตามมาตรฐาน IEEE1888 มีรูปแบบของข้อความตามรูปแบบ XML บนการสื่อสารแบบ TCP/IP หน่วยเก็บข้อมูลระบบ CU-BEMS จึงต้องมีการตอบกลับด้วยข้อความ 200 OK ไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์หลังจากที่ได้รับข้อความการร้องขอ TRAP แล้วดังรูปที่ 4.11 และรูปที่ 4.12 แสดงข้อความตอบกลับ 200 OK ที่อยู่ในรูปแบบ XML

	Time	Source	Destination	Protocol	Length	Info
17	2017-05-04 23:29:48.942945000	192.168.0.103	161.200.90.122	TCP	74	42948→80 [SYN]
18	2017-05-04 23:29:48.943838000	161.200.90.122	192.168.0.103	TCP	74	80→42948 [SYN,
19	2017-05-04 23:29:48.944305000	192.168.0.103	161.200.90.122	TCP	66	42948→80 [ACK]
22	2017-05-04 23:29:49.048105000	192.168.0.103	161.200.90.122	HTTP/X	838	POST /axis2/ser
23	2017-05-04 23:29:49.049186000	161.200.90.122	192.168.0.103	TCP	66	80→42948 [ACK]
24	2017-05-04 23:29:49.053055000	161.200.90.122	192.168.0.103	TCP	1080	[TCP segment of
25	2017-05-04 23:29:49.053056000	161.200.90.122	192.168.0.103	HTTP/X	66	HTTP/1.0 200 OK
26	2017-05-04 23:29:49.053552000	192.168.0.103	161.200.90.122	TCP	66	42948→80 [ACK]
27	2017-05-04 23:29:49.093777000	192.168.0.103	161.200.90.122	TCP	66	42948→80 [ACK]
28	2017-05-04 23:29:49.159485000	192.168.0.103	161.200.90.122	TCP	66	42948→80 [FIN,

รูปที่ 4.11: ข้อความตอบกลับการร้องขอ TRAP ด้วยข้อความ 200 OK ที่ถูกตรวจจับโดยโปรแกรม wireshark

```
<?xml
  version='1.0'
  encoding='UTF-8'
  ?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <ns2:queryRS
      xmlns:ns2="http://soap.fiap.org/">
      <transport
        xmlns="http://gutp.jp/fiap/2009/11/">
        <header>
          <OK/>
          <query
            id="a08c3a89-fcaa-4d6b-a455-52d39b792cb9"
            type="stream"
            ttl="300"
            callbackData="http://161.200.90.109"
            callbackControl="http://161.200.90.109">
            <key
              id="http://bems.ee.eng.chula.ac.th/eng4/fl13/north/room_server/z1/sensor1/monitor/pir"
              attrName="value"
              trap="changed"/>
            </query>
          </header>
        <body/>
        </transport>
      </ns2:queryRS>
    </soapenv:Body>
  </soapenv:Envelope>
```

รูปที่ 4.12: รูปแบบของข้อความตอบกลับการร้องขอ TRAP ด้วยข้อความ 200 OK ที่อยู่ในรูปแบบ XML

4.2.3 การทดสอบการตอบกลับ TRAP callback เมื่อข้อมูลเกิดการเปลี่ยนแปลง

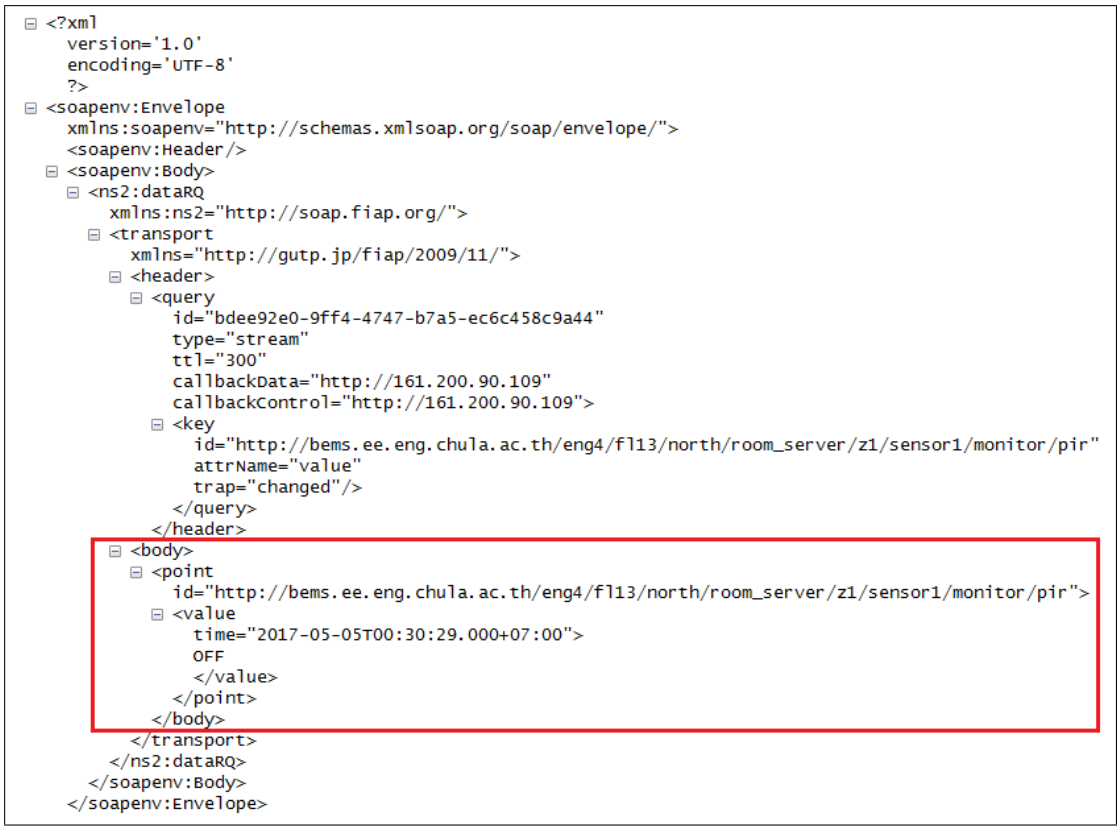
หลังจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งการร้องขอการแจ้งเตือน TRAP ไปยังหน่วยเก็บข้อมูลแล้ว ถ้าข้อมูลที่ได้จากตัวรับรู้ CU-BEMS ในหน่วยเก็บข้อมูลเกิดการเปลี่ยนแปลงที่

point ID สำหรับเก็บค่าที่ได้จากตัวรับรู้การเคลื่อนไหวของคน http://bems.ee.eng.chula.ac.th /eng4/fl13/north/room_server/z1/sensor1/monitor/pir หน่วยเก็บข้อมูลจะตอบกลับข้อความการแจ้งเตือนข้อมูล TRAP callback พร้อมกับค่าการเปลี่ยนแปลงไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ตามโปรแกรม trap_callback_pi.js ดังภาคผนวก ข จากนั้นอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จึงทำการแจกแจงข้อมูลที่ได้รับมาจากรูปแบบ XML ก่อนที่จะเปลี่ยนรูปแบบของข้อมูลเพื่อนำไปใช้ในกระบวนการต่อไป โดยรูปแบบของข้อความตอบกลับการแจ้งเตือน TRAP callback จากการตรวจจับด้วยโปรแกรม wireshark บนอุปกรณ์ตรวจจับเป็นไปดังรูปที่ 4.13 และรูปที่ 4.14

```
<?xml
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Header/>
  <soapenv:Body>
    <ns2:dataRQ
        xmlns:ns2="http://soap.fiap.org/">
      <transport
          xmlns="http://gutp.jp/fiap/2009/11/">
        <header>
          <query
              id="89cefb00-aa4e-4602-bc58-9ce4fc1cb0e3"
              type="stream"
              ttl="300"
              callbackData="http://161.200.90.109"
              callbackControl="http://161.200.90.109">
            <key
                id="http://bems.ee.eng.chula.ac.th/eng4/fl13/north/room_server/z1/sensor1/monitor/pir"
                attrName="value"
                trap="changed"/>
            </query>
          </header>
        <body>
          <point
              id="http://bems.ee.eng.chula.ac.th/eng4/fl13/north/room_server/z1/sensor1/monitor/pir">
            <value
                time="2016-05-19T17:08:55.000+07:00">
                ON
                </value>
            </point>
          </body>
        </transport>
      </ns2:dataRQ>
    </soapenv:Body>
  </soapenv:Envelope>
```

รูปที่ 4.13: รูปแบบของข้อความตอบกลับการแจ้งเตือน TRAP callback เมื่อตัวรับรู้ CU-BEMS ตรวจจับการเคลื่อนไหวของคนได้

จากรูปที่ 4.13 จะเห็นได้ว่ารูปแบบของข้อความตอบกลับการแจ้งเตือน TRAP callback เป็นการตอบกลับการแจ้งเตือนจากหน่วยเก็บข้อมูลไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์พร้อมกับข้อมูลการเปลี่ยนแปลงที่ได้จากตัวรับรู้การเคลื่อนไหวของคนด้วยค่า “ON” หมายถึง ตัวรับรู้การเคลื่อนไหวของคนตรวจจับการเคลื่อนไหวได้ หรือ “OFF” ในรูปที่ 4.14 หมายถึง ไม่มีการตรวจจับการเคลื่อนไหวของคน นอกจากข้อความตอบกลับ TRAP callback ตอบกลับมาพร้อมกับค่าการเปลี่ยนแปลงแล้วยังมีเวลาที่ข้อมูลเกิดการเปลี่ยนแปลงในหน่วยเก็บข้อมูล (time stamp) อีกด้วย และอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จะตอบกลับข้อความ 200 OK ไปยังหน่วยเก็บข้อมูลระบบ CU-BEMS เพื่อยืนยันการรับข้อความ TRAP callback ดังรูปที่ 4.15 จากนั้นอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จึงทำการแจกแจงข้อมูลจากรูปแบบ XML และแปลงเป็นรูปแบบ ECHONET Lite เพื่อนำไปใช้ในการส่งข้อความควบคุมไปยังโนด ECHONET Lite ที่มีอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite เชื่อมต่ออยู่



รูปที่ 4.14: รูปแบบของข้อความตอบกลับการแจ้งเตือน TRAP callback เมื่อตัวรับรู้ CU-BEMS ไม่มีการตรวจจับการเคลื่อนไหวของคน

รูปที่ 4.15: รูปแบบของข้อความตอบกลับ 200 OK เพื่อยืนยันการรับข้อความ TRAP callback

4.2.4 การทดสอบการส่งข้อความควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศด้วยการร้องขอแบบ WRITE

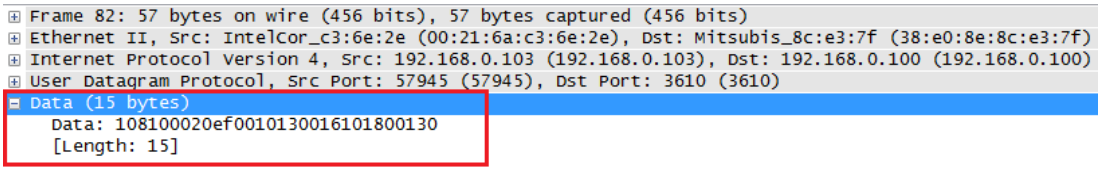
อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ทำการแปลงรูปแบบของข้อมูลให้เป็นรูปแบบ ECHONET Lite หลังจากผ่านกระบวนการแจกแจงข้อมูลจากรูปแบบ XML เพื่อนำข้อมูลเหล่านั้นไปใช้ส่งข้อความการร้องขอในการควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน (ON, OFF) แบบ WRITE ตามมาตรฐาน ECHONET Lite ด้วยโปรแกรม node.js ที่เรียกใช้มอดูล echonet-lite [20] ดังชุดคำสั่งสำหรับการควบคุมการเปิดและปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite โดยระบุค่า ESV เท่ากับ 0x61 สำหรับการร้องขอแบบ WRITE และระบุค่า EPC เท่ากับ 0x80 สำหรับการควบคุมในโหมดการเปิดปิด ในชุดคำสั่งยังมีการระบุเลขที่อยู่ไอพี คือ 192.168.0.100 ซึ่งเป็นเลขที่อยู่ไอพีของโนด ECHONET Lite ที่มีอุปกรณ์เครื่องปรับอากาศเชื่อมต่ออยู่

ชุดคำสั่งสำหรับการควบคุมการเปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ในโปรแกรม trap_callback_pi.js ดังภาคผนวก ข

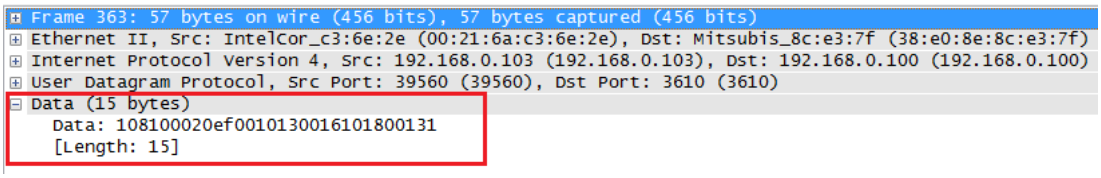
```
//EL.sendOPC1 = function( ip, seoj, deoj, esv, epc, edt)
EL.sendOPC1('192.168.0.100',[0x0e,0xf0,0x01],[0x01,0x30,0x01],0x61,0x80,0x30);
```

ชุดคำสั่งสำหรับการควบคุมการปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ในโปรแกรม trap_callback_pi.js ดังภาคผนวก ข

```
//EL.sendOPC1 = function( ip, seoj, deoj, esv, epc, edt)
EL.sendOPC1('192.168.0.100',[0x0e,0xf0,0x01],[0x01,0x30,0x01],0x61,0x80,0x31);
```



รูปที่ 4.16: ข้อความการร้องขอเพื่อควบคุมการเปิดเครื่องปรับอากาศ (ESV=0x61, EDT=0x30)



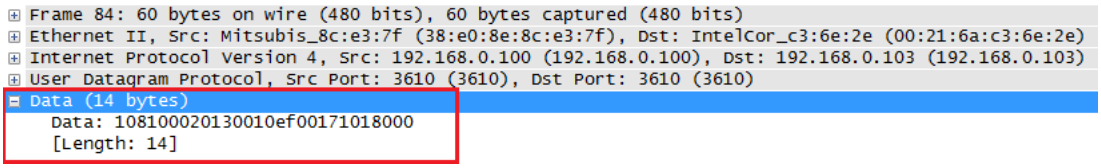
รูปที่ 4.17: ข้อความการร้องขอเพื่อควบคุมการปิดเครื่องปรับอากาศ (ESV=0x61, EDT=0x31)

จากการทดสอบได้ข้อความที่ถูกส่งจากโนดในอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ (ECHONET Lite self-node) เพื่อไปควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศที่เชื่อมต่ออยู่กับโนดอีกโนดในโครงข่าย ECHONET Lite จากการตรวจจับด้วยโปรแกรม wireshark บนอุปกรณ์ตรวจจับแสดงดังรูป

ที่ 4.16 และ 4.17 ในรูปที่ 4.16 คือ ข้อความการร้องขอเพื่อควบคุมเครื่องปรับอากาศด้วยการระบุค่า EDT เท่ากับ 0x30 หมายถึง การควบคุมการเปิดเครื่องปรับอากาศ และในรูปที่ 4.17 ระบุค่า EDT เท่ากับ 0x31 หมายถึง การควบคุมการปิดเครื่องปรับอากาศ ข้อความที่ถูกตรวจจับได้นี้ยังแสดงให้เห็นว่าในการสื่อสารด้วยโพรโทคอลตามมาตรฐาน ECHONET Lite นั้นใช้การสื่อสารแบบ UDP (user datagram protocol) ขนาดของข้อความที่ใช้จึงน้อยกว่าการสื่อสารตามมาตรฐาน IEEE1888 ที่ใช้การสื่อสารแบบ TCP/IP ซึ่งขนาดของข้อความตามมาตรฐาน ECHONET Lite (ECHONET Lite data) ที่ใช้ในการควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite มีขนาดเท่ากับ 15 ไบต์

4.2.5 การทดสอบการตอบกลับข้อความจากการร้องขอแบบ WRITE

เมื่อโนด ECHONET Lite ได้รับข้อความการร้องขอแบบ WRITE จากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เพื่อนำไปควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศแล้ว โนด ECHONET Lite จะทำการควบคุมไปยังตัวควบคุม (controller) ที่เชื่อมต่ออยู่กับอุปกรณ์เครื่องปรับอากาศพร้อมกับตอบกลับข้อความไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ด้วยการกำหนดค่า ESV เท่ากับ 0x71 และค่า EDT เท่ากับ 0x00 เพื่อเป็นการยืนยันการร้องขอว่าการร้องขอนั้นเสร็จสมบูรณ์และถูกต้องตามที่มาตรฐานกำหนด ในกรณีที่คำสั่งในการควบคุมไม่ถูกต้องหรือไม่ตรงตามที่กำหนดในมาตรฐาน โนด ECHONET Lite จะทำการตอบกลับข้อความโดยระบุค่า ESV เท่ากับ 0x51 เพื่อบ่งบอกว่ามีข้อผิดพลาดในการร้องขอแบบ WRITE จากการทดสอบได้ข้อความตอบกลับจากการตรวจจับด้วยโปรแกรม wireshark บนอุปกรณ์ตรวจจับดังรูปที่ 4.18 โดยข้อความตอบกลับที่ได้นี้มีขนาดข้อความที่ใช้ตามมาตรฐาน ECHONET Lite เท่ากับ 14 ไบต์



รูปที่ 4.18: ข้อความตอบกลับจากโนด ECHONET Lite ด้วยการระบุค่า ESV เท่ากับ 0x71

4.3 การทดสอบการประสานข้อมูลแบบทันทีจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888

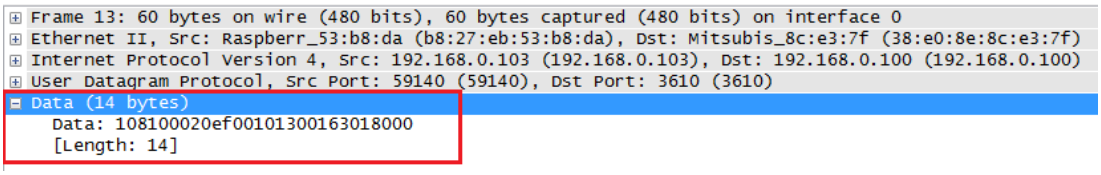
กระบวนการทดสอบการประสานข้อมูลแบบทันทีจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 มีความคล้ายคลึงกับกระบวนการทดสอบการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite แต่การทดสอบไปในทิศทางกลับกัน โดยเริ่มจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite และนำค่าการเปลี่ยนแปลงไปเก็บยังหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 ในระบบ CU-BEMS

4.3.1 การทดสอบการร้องขอเพื่อแจ้งเตือนการเปลี่ยนแปลงสถานะของอุปกรณ์เครื่องปรับอากาศด้วยการร้องขอแบบ NOTIFY

สำหรับการประสานข้อมูลแบบทันทีจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 นั้นอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งข้อความร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะของอุปกรณ์เครื่องปรับอากาศแบบ NOTIFY ตามมาตรฐาน ECHONET Lite ด้วยโปรแกรม echonet_notify_write.js ดังภาคผนวก ค ซึ่งเรียกใช้มอดูล echonet-lite [20] แสดงดังชุดคำสั่งสำหรับการร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite แบบ NOTIFY ในการร้องขอแบบ NOTIFY นี้มีการกำหนดค่า ESV เท่ากับ 0x63 ใช้สำหรับร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะ ค่า EPC เท่ากับ 0x80 สำหรับการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิด (ON, OFF) ของอุปกรณ์เครื่องปรับอากาศ และกำหนดเลขที่อยู่ไอพีของโนด ECHONET Lite ที่มีอุปกรณ์เครื่องปรับอากาศเชื่อมต่ออยู่ คือ 192.168.0.100 ซึ่งในการทดสอบนี้สามารถตรวจจับข้อความการร้องขอ NOTIFY ด้วยโปรแกรม wireshark บนอุปกรณ์ตรวจจับได้ดังรูปที่ 4.19 ซึ่งข้อความการร้องขอ NOTIFY นี้มีขนาดของข้อความตามมาตรฐาน ECHONET Lite เท่ากับ 14 ไบต์

ชุดคำสั่งสำหรับการร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite แบบ NOTIFY ในโปรแกรม echonet_notify_write.js ดังภาคผนวก ค

```
//EL.sendOPC1 = function( ip, seoj, deoj, esv, epc, edt)
EL.sendOPC1('192.168.0.100',[0x0e,0xf0,0x01],[0x01,0x30,0x01],0x63,0x80,0x00);
```

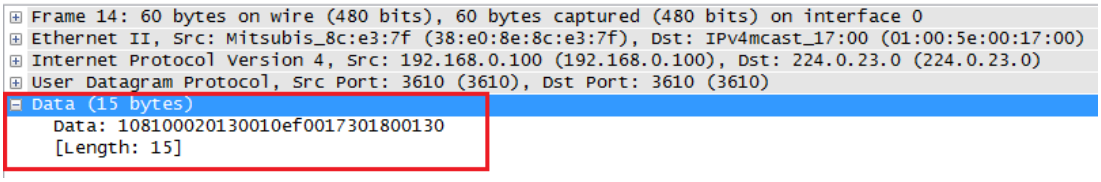


รูปที่ 4.19: ข้อความการร้องขอ NOTIFY ที่ถูกตรวจจับโดยโปรแกรม wireshark

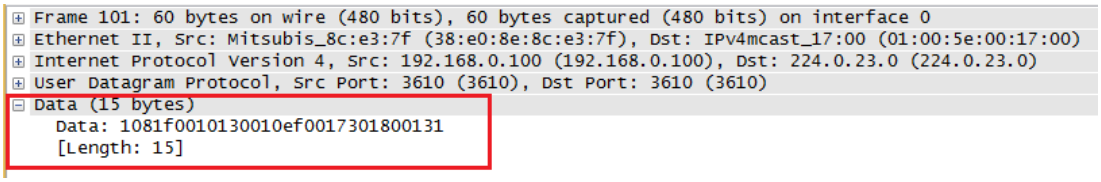
4.3.2 การทดสอบการรับข้อความตอบกลับ NOTIFY เมื่อสถานะของอุปกรณ์เครื่องปรับอากาศเกิดการเปลี่ยนแปลง

หลังจากที่อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งข้อความร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ไปยังโนด ECHONET Lite ที่เชื่อมต่ออยู่ โนด ECHONET Lite จะทำการตอบกลับข้อความ NOTIFY พร้อมกับค่าสถานะที่เปลี่ยนแปลงเมื่อสถานะการเปิดปิดเกิดการเปลี่ยนแปลง ข้อความตอบกลับ NOTIFY นี้ถูกตรวจจับได้โดยโปรแกรม wireshark บนอุปกรณ์ตรวจจับสำหรับแจ้งเตือนสถานะการเปิดหรือปิดของอุปกรณ์เครื่องปรับอากาศแสดงดังรูปที่ 4.20 และ 4.21 ข้อความตอบกลับการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดมีการกำหนดค่า ESV เท่ากับ 0x73 ซึ่งเป็นการกำหนดค่าตามมาตรฐานสำหรับการตอบกลับข้อความ NOTIFY และเป็นการตอบกลับแบบแพร่สัญญาณ (broadcast) ไปยังทุกโนดที่อยู่ภายในโครงข่าย ECHONET Lite เดียวกันเมื่อสถานะเกิดการเปลี่ยนแปลง นอกจากนี้ยังมีการกำหนด

ค่า EDT เท่ากับ 0x30 ซึ่งหมายถึงการแจ้งเตือนสถานะการเปิดของอุปกรณ์เครื่องปรับอากาศ หรือค่า EDT เท่ากับ 0x31 หมายถึงการแจ้งเตือนสถานะการปิดของอุปกรณ์เครื่องปรับอากาศ ส่วนค่า EPC นั้นถูกกำหนดเหมือนกันทั้งข้อความร้องขอและข้อความตอบกลับ NOTIFY คือ 0x80 เพื่อบ่งบอกว่าเป็นโหมดการเปิดปิดอุปกรณ์เครื่องปรับอากาศตามมาตรฐาน ECHONET Lite การทดสอบนี้ได้ขนาดของข้อความตอบกลับ NOTIFY ที่ใช้ในการสื่อสารตามมาตรฐาน ECHONET Lite เท่ากับ 15 ไบต์ ในกรณีที่การร้องขอการแจ้งเตือน NOTIFY ไม่ถูกต้องตามที่กำหนดในมาตรฐานหรือเกิดข้อผิดพลาดในการตอบกลับการแจ้งเตือน โนด ECHONET Lite จะตอบกลับข้อความด้วยการระบุค่า ESV เท่ากับ 0x53



รูปที่ 4.20: ข้อความแจ้งเตือนการเปิดของอุปกรณ์เครื่องปรับอากาศที่ถูกตรวจจับโดยโปรแกรม wireshark



รูปที่ 4.21: ข้อความแจ้งเตือนการปิดของอุปกรณ์เครื่องปรับอากาศที่ถูกตรวจจับโดยโปรแกรม wireshark

4.3.3 การทดสอบการร้องขอเพื่อจัดเก็บข้อมูลสถานะของอุปกรณ์เครื่องปรับอากาศด้วยโพรโทคอล WRITE

เมื่ออินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ได้รับข้อความตอบกลับ NOTIFY พร้อมกับค่าการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศแล้ว อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จึงทำการแปลงรูปแบบของข้อมูลจากรูปแบบ ECHONET Lite ไปเป็นรูปแบบ XML ตามที่ใช้ในการสื่อสารมาตรฐาน IEEE1888 เพื่อส่งข้อมูลไปเก็บยังหน่วยเก็บข้อมูลด้วยโพรโทคอล WRITE ตามมาตรฐาน IEEE1888 ในระบบ CU-BEMS จากรูปที่ 4.22 แสดงข้อความการร้องขอ WRITE สำหรับเก็บค่าสถานะการเปิดของอุปกรณ์เครื่องปรับอากาศจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ไปยังหน่วยเก็บข้อมูลที่ถูกตรวจจับโดยโปรแกรม wireshark บนอุปกรณ์ตรวจจับ และรูปที่ 4.23 แสดงข้อความการร้องขอ WRITE สำหรับเก็บค่าสถานะการปิดของอุปกรณ์เครื่องปรับอากาศ ค่าสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศถูกส่งไปเก็บยังหน่วยเก็บข้อมูลที่ point ID คือ http://chayanon.chula.ac.th/echonet/air-conditioner/status พร้อมกับค่าเวลาที่ถูกส่ง (time stamp) ดังรูปที่ 4.24

รูปที่ 4.22: รูปแบบของข้อความการร้องขอ WRITE เพื่อเก็บข้อมูลสถานะการเปิดของอุปกรณ์เครื่องปรับอากาศที่ถูกตรวจจับโดยโปรแกรม wireshark

รูปที่ 4.23: รูปแบบของข้อความการร้องขอ WRITE เพื่อเก็บข้อมูลสถานะการปิดของอุปกรณ์เครื่องปรับอากาศที่ถูกตรวจจับโดยโปรแกรม wireshark

http://boss/eng4/fl12/corridor/elevatorfront/kinect/num_user	2560-04-29T14:52:19.000+07:00	2
http://chayanon.ac.th/eng4/fl13/corridor/elevatorfront/kinect/cumm_swipe		
http://chayanon.ac.th/eng4/fl13/corridor/elevatorfront/kinect/num_user		
http://chayanon.chula.ac.th/echonet/air-conditioner/status	2017-05-05T01:04:50.000+07:00	ON
http://chayanon.chula.ac.th/test/echonet/sensor/temperature		
http://chayanon.chula.ac.th/test/echonet/write	2017-03-14T15:24:36.000+07:00	24

รูปที่ 4.24: point ID สำหรับเก็บค่าสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ในหน่วยเก็บข้อมูล CU-BEMS

4.3.4 การทดสอบการตอบกลับโพรโทคอล WRITE ด้วยข้อความ 200 OK

เนื่องจากการสื่อสารระหว่างอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์และหน่วยเก็บข้อมูลในระบบ CU-BEMS ด้วยโพรโทคอล WRITE นั้นใช้การสื่อสารชนิด TCP/IP หน่วยเก็บข้อมูลจึงต้องยืนยันการรับข้อมูลโดยการตอบกลับข้อความ 200 OK ไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์หลังจากที่หน่วยเก็บข้อมูลได้รับข้อมูลแล้ว ดังรูปที่ 4.25 และในรูปที่ 4.26 แสดงข้อความการตอบกลับ 200 OK ในรูปแบบ XML ที่ถูกตรวจจับโดยโปรแกรม wireshark บนอุปกรณ์ตรวจจับ

Time	Source	Destination	Protocol	Length	Info
69 2017-03-31 18:04:14.174270000	192.168.0.103	161.200.90.122	TCP	74	50574→80 [SYN]
70 2017-03-31 18:04:14.175153000	161.200.90.122	192.168.0.103	TCP	74	80→50574 [SYN,
71 2017-03-31 18:04:14.175597000	192.168.0.103	161.200.90.122	TCP	66	50574→80 [ACK]
72 2017-03-31 18:04:14.250289000	192.168.0.103	161.200.90.122	HTTP/XM	673	POST /axis2/ser
73 2017-03-31 18:04:14.251295000	161.200.90.122	192.168.0.103	TCP	66	80→50574 [ACK]
74 2017-03-31 18:04:14.254095000	161.200.90.122	192.168.0.103	TCP	788	[TCP segment of
75 2017-03-31 18:04:14.254096000	161.200.90.122	192.168.0.103	HTTP/XM	66	HTTP/1.0 200 OK
76 2017-03-31 18:04:14.254580000	192.168.0.103	161.200.90.122	TCP	66	50574→80 [ACK]
77 2017-03-31 18:04:14.285148000	192.168.0.103	161.200.90.122	TCP	66	50574→80 [ACK]
82 2017-03-31 18:04:14.374899000	192.168.0.103	161.200.90.122	TCP	66	50574→80 [FIN,
83 2017-03-31 18:04:14.375631000	161.200.90.122	192.168.0.103	TCP	66	80→50574 [ACK]

รูปที่ 4.25: รูปแบบของข้อความการร้องขอด้วยโพรโทคอล WRITE บนโครงข่ายการสื่อสารชนิด TCP/IP

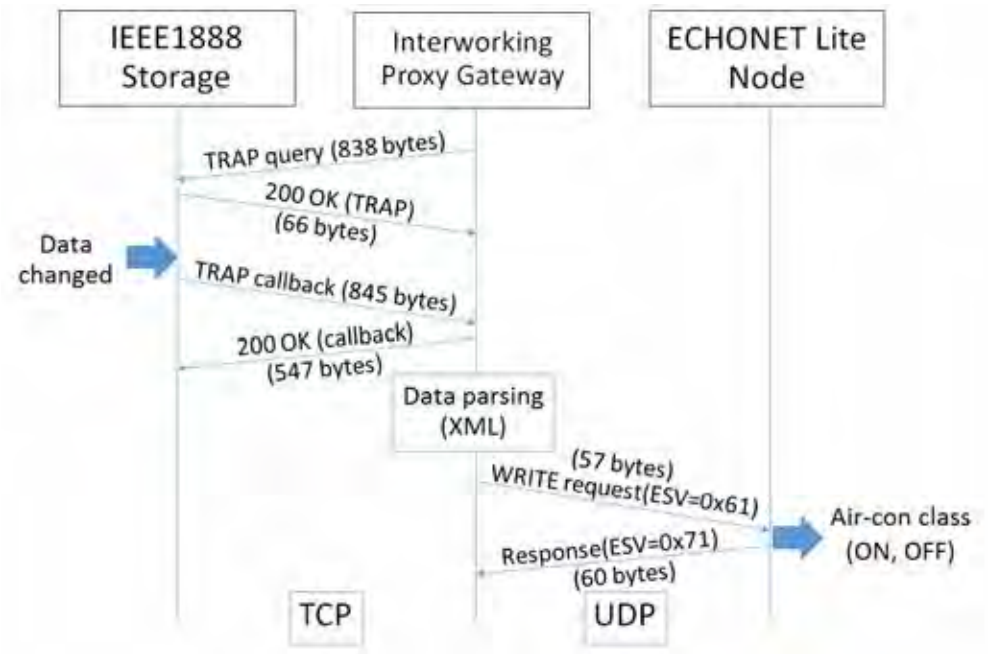
4.4 รายละเอียดและลำดับการสื่อสารในการประสานข้อมูลระหว่างสองมาตรฐาน

จากการทดสอบการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite และจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 ในระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite โดยการพัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์มีลำดับการสื่อสารและรายละเอียดแสดงดังรูปที่ 4.27 และรูป 4.28

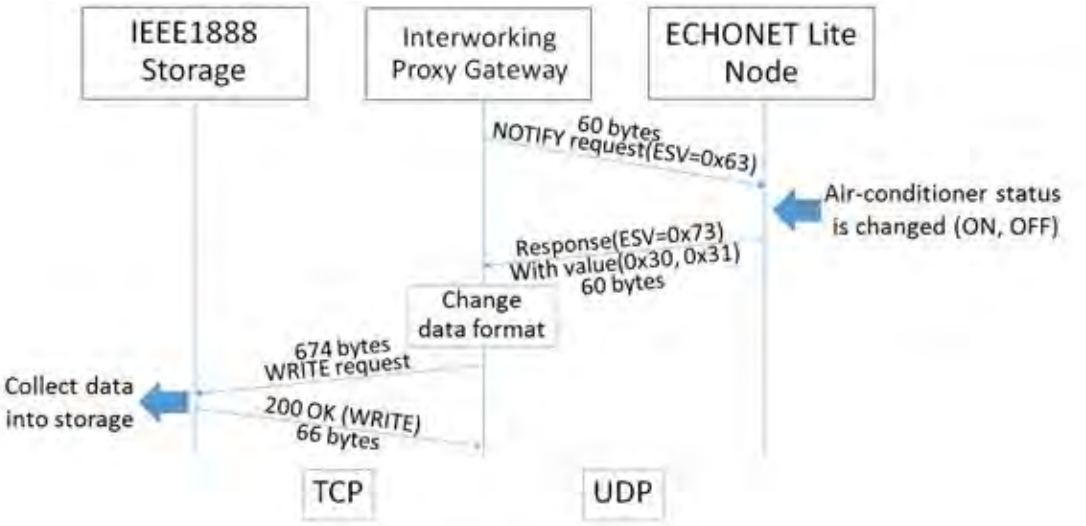
จากรูป 4.27 และ 4.28 เป็นการสรุปลำดับทั้งหมดในการสื่อสารระหว่าง 3 ส่วนประกอบหลักของการประสานข้อมูลในระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ในรูปที่ 4.27 แสดงรายละเอียดและลำดับการสื่อสารและขนาดของข้อความที่ใช้ในการประสานข้อมูลแบบทันทีตั้งแต่อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งข้อความร้องขอ TRAP ไปยังหน่วยเก็บข้อมูล IEEE1888 จนกระทั่งข้อมูลในหน่วยเก็บข้อมูลเกิดการเปลี่ยนแปลง อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จึงแปลงข้อมูลที่ได้มาและใช้ส่งข้อความร้องขอแบบ WRITE ไปยังโนด ECHONET

```
<?xml
  version='1.0'
  encoding='UTF-8'
  ?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <ns2:dataRS
      xmlns:ns2="http://soap.fiap.org/">
      <transport
        xmlns="http://gutp.jp/fiap/2009/11/">
        <header>
          <OK/>
          </header>
        </transport>
      </ns2:dataRS>
    </soapenv:Body>
  </soapenv:Envelope>
```

รูปที่ 4.26: รูปแบบของข้อความตอบกลับ 200 OK จากหน่วยเก็บข้อมูลไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์



รูปที่ 4.27: รายละเอียดการสื่อสารในการทดสอบการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite



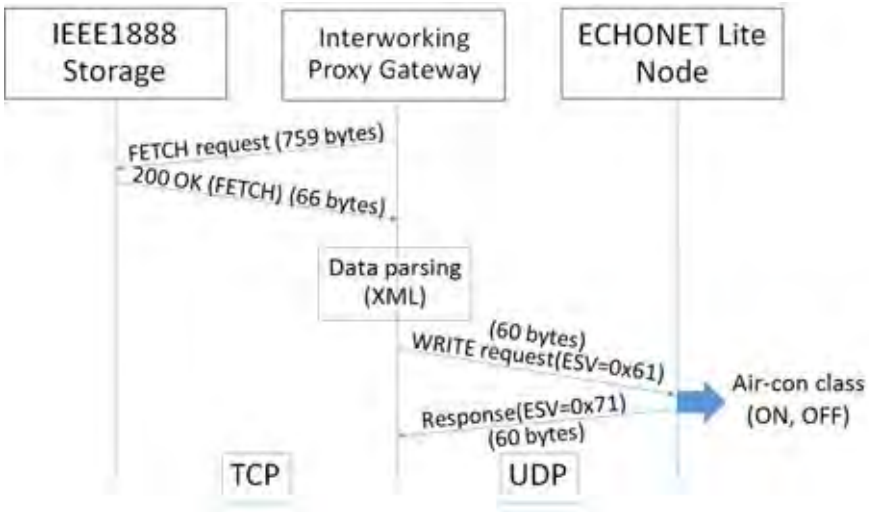
รูปที่ 4.28: รายละเอียดการสื่อสารในการทดสอบการประสานข้อมูลแบบทันทีจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888

Lite เพื่อควบคุมการเปิดปิดเครื่องรับอากาศ และในรูปที่ 4.28 แสดงรายละเอียดและขนาดของข้อความที่ใช้ในการประสานข้อมูลแบบทันทีในทิศทางตรงกันข้ามโดยเริ่มจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งข้อความร้องขอแบบ NOTIFY ไปยังโนด ECHONET Lite จนกระทั่งสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศเกิดการเปลี่ยนแปลง อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จึงแปลงข้อมูลและส่งข้อความร้องขอ WRITE ไปยังหน่วยเก็บข้อมูล IEEE1888

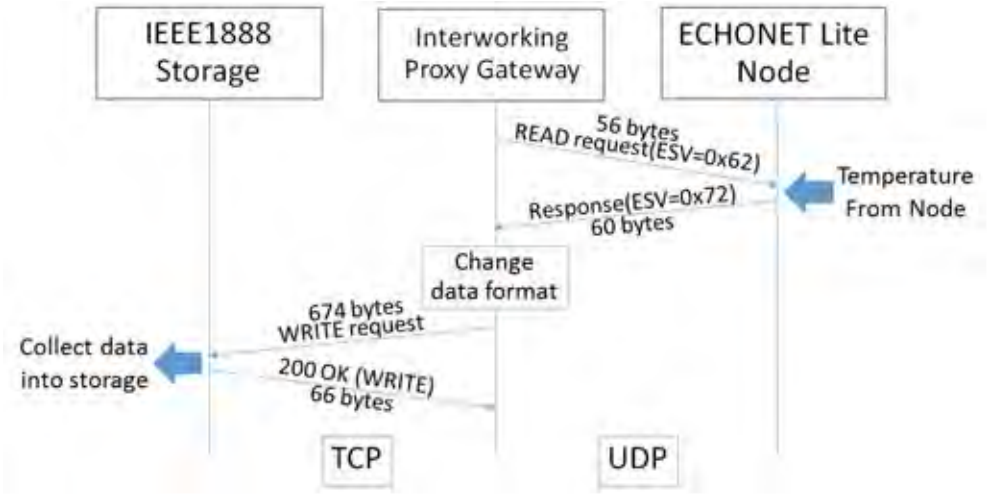
นอกจากการทดสอบการประสานข้อมูลแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ในทั้งสองทิศทางแล้ว วิทยานิพนธ์นี้มีการทดสอบการประสานข้อมูลแบบคาบเวลาระหว่างสองมาตรฐานนี้ในลักษณะการใช้งานที่จำเป็น จากการทดสอบจึงได้รายละเอียดและขนาดของข้อความที่ใช้ในการประสานข้อมูลแบบคาบเวลาทั้งจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite และจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 ดังรูปที่ 4.29 และ 4.30

จากการทดสอบได้สรุปขนาดของแพ็กเกตที่ใช้ในการสื่อสารแต่ละลำดับสำหรับการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite และจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 จากการตรวจจับด้วยโปรแกรม wireshark บนอุปกรณ์ตรวจจับ ดังรูปที่ 4.31 และ 4.32

จากรูปที่ 4.31 และ 4.32 เห็นได้ว่าเมื่อเปรียบเทียบการประสานข้อมูลแบบทันทีเพื่อให้เกิดการแจ้งเตือนข้อมูลเมื่อข้อมูลมีการเปลี่ยนแปลงระหว่างทั้งสองมาตรฐานแล้ว การประสานข้อมูลตามมาตรฐาน IEEE1888 ในระบบ CU-BEMS ใช้ขนาดของข้อความทั้งหมด 2,296 ไบต์ ดังรูปที่ 4.31 และ 740 ไบต์ ดังรูปที่ 4.32 ซึ่งมากกว่าในการประสานข้อมูลตามมาตรฐาน ECHONET Lite ที่ใช้ขนาดของข้อความทั้งหมด 117 ไบต์ ในรูปที่ 4.31 และ 120 ไบต์ ในรูปที่ 4.32 นั่นหมายถึงการประสานข้อมูลตามมาตรฐาน IEEE1888 นั้นจำเป็นต้องใช้แบนด์วิดท์ (bandwidth) มากกว่าในการประสานข้อมูลตามมาตรฐาน ECHONET Lite สำหรับการแจ้งเตือนหรืออัพเดทข้อมูล point ID แต่ละครั้ง ทั้งนี้เนื่องมาจากมาตรฐาน IEEE1888 ใช้รูปแบบข้อความ XML บนการสื่อสารชนิด TCP/IP ในขณะที่มาตรฐาน ECHONET Lite นั้นใช้รูปแบบข้อความเป็นแบบการเข้ารหัสบิตที่มีการกำหนดไว้



รูปที่ 4.29: รายละเอียดการสื่อสารในการทดสอบการประสานข้อมูลแบบคาบเวลาจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite



รูปที่ 4.30: รายละเอียดการสื่อสารในการทดสอบการประสานข้อมูลแบบคาบเวลาจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888

	Message	Avg. Packet Size(bytes)
IEEE1888	TRAP query	838
	200 OK (TRAP)	66
	TRAP callback	845
	200 OK (TRAP callback)	547
ECHONET Lite	WRITE request (ESV=0x61)	57
	Response (ESV=0x71)	60
	Overall	**2,413**

รูปที่ 4.31: ขนาดของแพ็กเกตที่วัดได้จากการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite

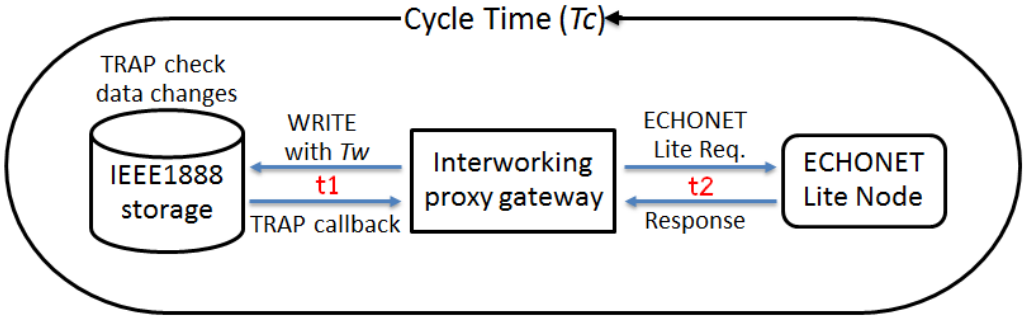
	Message	Avg. Packet Size(bytes)
ECHONET Lite	NOTIFY request (ESV=0x63)	60
	Broadcast Response (ESV=0x73)	60
IEEE1888	WRITE Protocol	674
	200 OK (WRITE)	66
	Overall	**860**

รูปที่ 4.32: ขนาดของแพ็กเกตที่วัดได้จากการประสานข้อมูลแบบทันทีจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888

ในมาตรฐานบนการสื่อสารชนิด UDP โดยส่วนใหญ่ อย่างไรก็ตามรูปแบบข้อความ XML ตามมาตรฐาน IEEE1888 มีข้อได้เปรียบมากกว่ารูปแบบข้อความตามมาตรฐาน ECHONET Lite ในเรื่องของความยืดหยุ่นทางการใช้งานระหว่างมาตรฐานอื่น ๆ ด้วยเช่นกัน เนื่องจากรูปแบบข้อความ XML มีรูปแบบที่ใกล้เคียงกับภาษามนุษย์มากกว่ารูปแบบข้อความ ECHONET Lite ที่มีการเข้ารหัสบิตตามมาตรฐาน สำหรับเรื่องความปลอดภัยในการสื่อสารของทั้งสองมาตรฐานนั้น มาตรฐาน IEEE1888 สามารถใช้โพรโทคอลการสื่อสาร HTTPS สำหรับการเข้ารหัสข้อความ และมาตรฐาน ECHONET Lite มีการเข้ารหัสบิตของข้อความตามมาตรฐาน ECHONET Lite ซึ่งไม่สามารถอ่านได้ด้วยการดักข้อความแบบธรรมดา

4.5 การทดสอบสมรรถนะของอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ในการประสานข้อมูลแบบทันทีระหว่างสองมาตรฐาน

เนื่องจากระบบการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite โดยการพัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์นั้นได้มีการติดตั้งเพื่อใช้ทดสอบจริง การทดสอบสมรรถนะของอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จึงมีความจำเป็นอย่างยิ่งในการประเมินระบบที่ถูกพัฒนาขึ้นในวิทยานิพนธ์ฉบับนี้ การทดสอบนี้เป็นการทดสอบการประสานข้อมูลแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite โดยมีการจำลองข้อมูลเพิ่มเข้ามาสำหรับใช้ในการทดสอบตามรูปที่ 4.33



รูปที่ 4.33: การทดสอบสมรรถนะของอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์

การทดสอบนี้ถูกทดสอบโดยการให้อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งโพรโทคอล TRAP ไปยังหน่วยเก็บข้อมูลเพื่อตรวจสอบการเปลี่ยนแปลงของข้อมูลด้วยโปรแกรม node.js ดังภาคผนวก ก และ ข จากนั้นอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เริ่มส่งโพรโทคอล WRITE เพื่อเปลี่ยนแปลงข้อมูลในหน่วยเก็บข้อมูลด้วยโปรแกรม node.js ดังภาคผนวก ง และ จ อย่างต่อเนื่องด้วยคาบเวลาที่แตกต่างกัน คือ Tw ซึ่งหาได้จาก 60 / จำนวนครั้งที่ส่งในหนึ่งนาที ข้อมูลที่เปลี่ยนแปลงจึงถูกส่งกลับมาพร้อมกับการตอบกลับ TRAP callback ไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เพื่อแปลงรูปแบบของข้อมูล และส่งไปยังโนด ECHONET Lite เพื่อควบคุมอุปกรณ์เครื่องปรับอากาศที่เชื่อมต่ออยู่ หลังจากที่โนด ECHONET Lite ได้รับข้อความการร้องขอแล้วจะตอบกลับข้อความไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ ในการทดสอบได้มีการวัดช่วงเวลาตั้งแต่อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งโพรโทคอล

WRITE จนกระทั่งอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ได้รับข้อความตอบกลับจากโนด ECHONET Lite คือ Tc ตามรูปที่ 4.33 และได้ผลจากการทดสอบส่งโพรโทคอล WRITE ด้วยจำนวนครั้งในการส่งต่อนาทีที่แตกต่างกัน ดังตารางที่ 4.2

ตารางที่ 4.2: ระยะเวลาเฉลี่ยที่ได้ในการประสานข้อมูลแบบครบรอบสมบูรณ์จากการเพิ่มความถี่ในการส่งการร้องขอ WRITE จากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ไปยังหน่วยเก็บข้อมูลระบบ CU-BEMS

WRITE (times/minute)	Average Cycle Time, Tc (millisecond)
1	178
10	180
20	219
30	367
50	424

ตารางที่ 4.2 แสดงให้เห็นว่าเมื่อทดสอบโดยการเพิ่มความถี่ในการส่งการร้องขอ WRITE จากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ไปยังหน่วยเก็บข้อมูลระบบ CU-BEMS ให้มากขึ้นจาก 1 ถึง 50 ครั้งต่อนาทีแล้ว ช่วงเวลาที่ใช้ในการประสานข้อมูลจนครบรอบสมบูรณ์ตั้งแต่ข้อความร้องขอออกจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์จนกระทั่งอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ได้รับข้อความตอบกลับ (Tc) ตามรูปที่ 4.33 มีค่ายาวนานยิ่งขึ้นจาก 178 ถึง 424 มิลลิวินาที จึงสรุปได้ว่าเมื่อระบบมีการอัพเดทค่าไปยัง point ID มากขึ้น ระบบต้องใช้เวลาในการประสานข้อมูลระหว่างสองมาตรฐานยาวนานขึ้น และเมื่อพิจารณาเพื่อหาค่าเวลาที่ใช้ตั้งแต่อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งการร้องขอ WRITE จนกระทั่งรับข้อความตอบกลับ TRAP callback หรือ t1 และเวลาที่ใช้ตั้งแต่อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ส่งการร้องขอเพื่อควบคุมอุปกรณ์เครื่องปรับอากาศ ECHONET Lite จนกระทั่งรับข้อความตอบกลับ หรือ t2 ตามรูปที่ 4.33 จึงได้ค่าเวลา t1 และ t2 จากการทดสอบเพิ่มความถี่ในการส่งการร้องขอ WRITE ดังตารางที่ 4.3

จากตารางที่ 4.3 เมื่อเพิ่มความถี่ในการส่งการร้องขอ WRITE จาก 1 ถึง 50 ครั้งต่อนาที พบว่าระยะเวลาที่ใช้ t1 มีค่ายาวนานขึ้นจาก 113 ถึง 355 มิลลิวินาที ในขณะที่ระยะเวลาที่ใช้ t2 มีค่าใกล้เคียงกันจากการทดสอบเพิ่มความถี่ในการส่งการร้องขอ WRITE แสดงให้เห็นว่าระยะเวลาที่ใช้ในการประสานข้อมูลของโดเมนมาตรฐาน IEEE1888 มีค่ายาวนานกว่าระยะเวลาที่ใช้ในการประสานข้อมูลของโดเมน ECHONET Lite ทั้งนี้ผลการทดสอบที่ได้เกิดจากการทดสอบการประสานข้อมูลแบบทันทีแค่ทิศทางเดียว คือ ทิศทางจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite เนื่องจากการทดสอบในทิศทางตรงกันข้ามนั้นมีลักษณะการสื่อสารที่เหมือนกันกับทิศทางที่ถูกทดสอบ จึงทำให้ผลที่ได้จากการทดสอบตรงกันทั้งสองทิศทาง

4.6 การทดสอบการทำงานในโหมดอื่น ๆ สำหรับควบคุมอุปกรณ์เครื่องปรับอากาศตามมาตรฐาน ECHONET Lite

นอกจากการทดสอบในเบื้องต้นที่กล่าวถึงการทดสอบโดยการควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite แล้วนั้น ยังมีการทดสอบการทำงานในโหมดอื่น ๆ ตามที่

ตารางที่ 4.3: ระยะเวลาเฉลี่ยที่ได้ของโดเมนมาตรฐาน IEEE1888 และโดเมนมาตรฐาน ECHONET Lite ในการประสานข้อมูลแบบครบรอบสมบูรณ์จากการเพิ่มความถี่ในการส่งการร้องขอ WRITE จากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ไปยังหน่วยเก็บข้อมูลระบบ CU-BEMS

WRITE (times/minute)	Avg. t1 (millisecond)	Avg. t2 (millisecond)
1	113	65
10	116	64
20	154	65
30	300	67
50	355	69

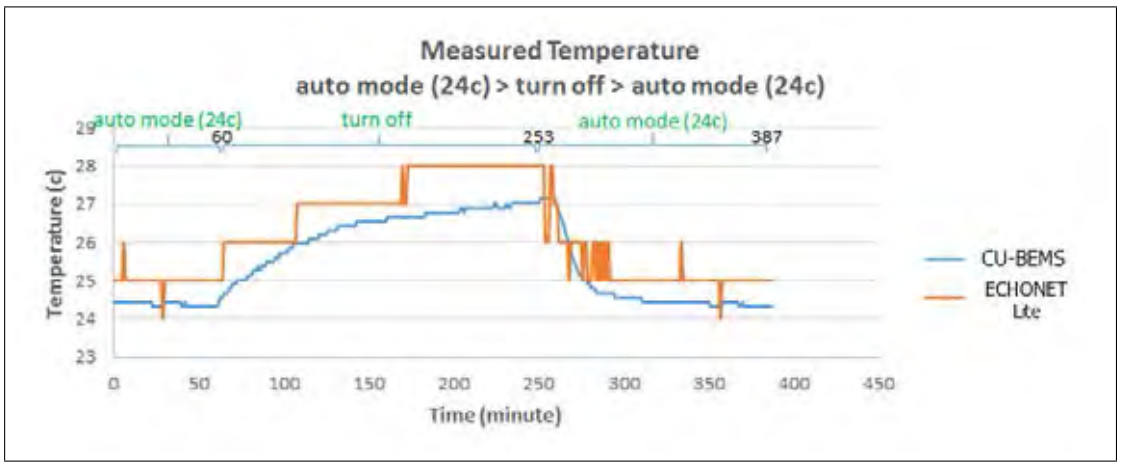
กำหนดในคู่มือของมาตรฐาน ECHONET Lite อย่างไรก็ตามการทดสอบในโหมดอื่น ๆ นี้ไม่สามารถทดสอบได้ทุกโหมดที่มีกำหนดในคู่มือการใช้งาน เนื่องจากตัวควบคุม (controller) ที่เชื่อมต่ออยู่กับอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ที่ติดตั้งจริงในห้องทดสอบนั้นมีข้อจำกัดในการควบคุมได้ในบางโหมดของการใช้งานเท่านั้น ซึ่งการควบคุมอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ในโหมดอื่น ๆ ที่สามารถใช้งานได้มีดังนี้

- Operation status (EPC = 0x80)
- Operation mode setting (EPC = 0xB0)
- Set temperature value (EPC = 0xB3)
- Measured value of room temperature (EPC = 0xBB)
- Air flow rate setting (EPC = 0xA0)
- Automatic swing of air flow setting (EPC = 0xA3)
- Air flow direction (vertical) setting (EPC = 0xA4)

ในการทดสอบระบบการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ในระบบ CU-BEMS โดยการพัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์นี้ อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ถูกติดตั้งจริงในห้องเซิร์ฟเวอร์เพื่อใช้ทดสอบในสภาพแวดล้อมจริง ห้องเซิร์ฟเวอร์มีการติดตั้งเครื่องคอมพิวเตอร์เซิร์ฟเวอร์เพื่อใช้เป็นหน่วยเก็บข้อมูล IEEE1888 อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite 1 ตัว โนด ECHONET Lite 1 ตัว ซึ่งเชื่อมต่อแบบแนบติดกับอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ตัวรับรู้ไร้สายในระบบ CU-BEMS 1 ตัว และตัวรับรู้ ECHONET Lite 1 ตัว อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ถูกทดสอบโดยปรับเปลี่ยนการควบคุมในรูปแบบต่าง ๆ ตามลำดับอย่างต่อเนื่อง และมีการวัดค่าอุณหภูมิและความชื้นสัมพัทธ์จากตัวรับรู้ CU-BEMS และตัวรับรู้ ECHONET Lite ด้วยคาบเวลาในการวัดที่เท่ากัน เท่ากับ 1 นาที ทั้งนี้อุปกรณ์เครื่องปรับอากาศเครื่องอื่น ๆ ในห้องทดสอบถูกปิดตลอดระยะเวลาการทดสอบ เพื่อให้สามารถวิเคราะห์ค่าอุณหภูมิและความชื้นสัมพัทธ์ที่เปลี่ยนแปลงไปได้อย่างชัดเจนจากการปรับการควบคุมอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite การทดสอบถูกทดสอบแยกเป็นกรณีดังนี้

4.6.1 การทดสอบการเปิดและปิดอุปกรณ์เครื่องปรับอากาศตามมาตรฐาน ECHONET Lite (EPC = 0x80)

ในการทดสอบการเปิดและปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite เริ่มจากการปล่อยให้อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ทำงานในสภาวะปกติภายในห้องทดสอบ ซึ่งอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite นี้ทำงานอยู่ในโหมดอัตโนมัติ (auto mode) ที่การตั้งค่าอุณหภูมิเท่ากับ 24 องศาเซลเซียส จากนั้นทดสอบปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite และปล่อยให้อุณหภูมิภายในห้องทดสอบเปลี่ยนแปลงอย่างต่อเนื่องจนกระทั่งคงที่ ทำการเปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite อีกครั้งในโหมดเดิม ได้ผลการเปลี่ยนแปลงอุณหภูมิภายในห้องทดสอบแสดงดังรูปที่ 4.34 พบว่าเมื่ออุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ทำงานในโหมดที่ได้ตั้งค่าไว้ อุณหภูมิภายในห้องทดสอบรักษาระดับอยู่ที่ประมาณ 24 องศาเซลเซียส แต่หลังจากทดสอบปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite อุณหภูมิที่วัดได้ภายในห้องทดสอบมีค่าเพิ่มขึ้นอย่างต่อเนื่องจนกระทั่งอยู่ที่ระดับประมาณ 27 องศาเซลเซียส และเมื่อทดสอบเปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite อีกครั้งในโหมดเดิมจึงทำให้อุณหภูมิภายในห้องทดสอบลดต่ำลงจนกระทั่งอยู่ในระดับเดิมกับช่วงแรกที่ถูกเปิดอยู่โดยประมาณ

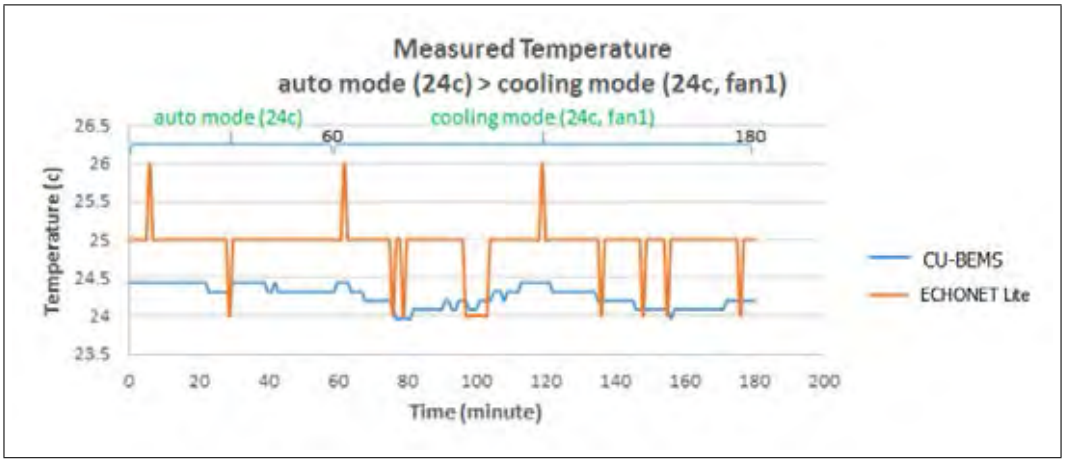


รูปที่ 4.34: อุณหภูมิที่ได้จากตัวรับรู้ CU-BEMS และตัวรับรู้ ECHONET Lite จากการทดสอบเปิดและปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite

4.6.2 การทดสอบเปลี่ยนโหมดการทำงานของอุปกรณ์เครื่องปรับอากาศตามมาตรฐาน ECHONET Lite (EPC = 0xB0)

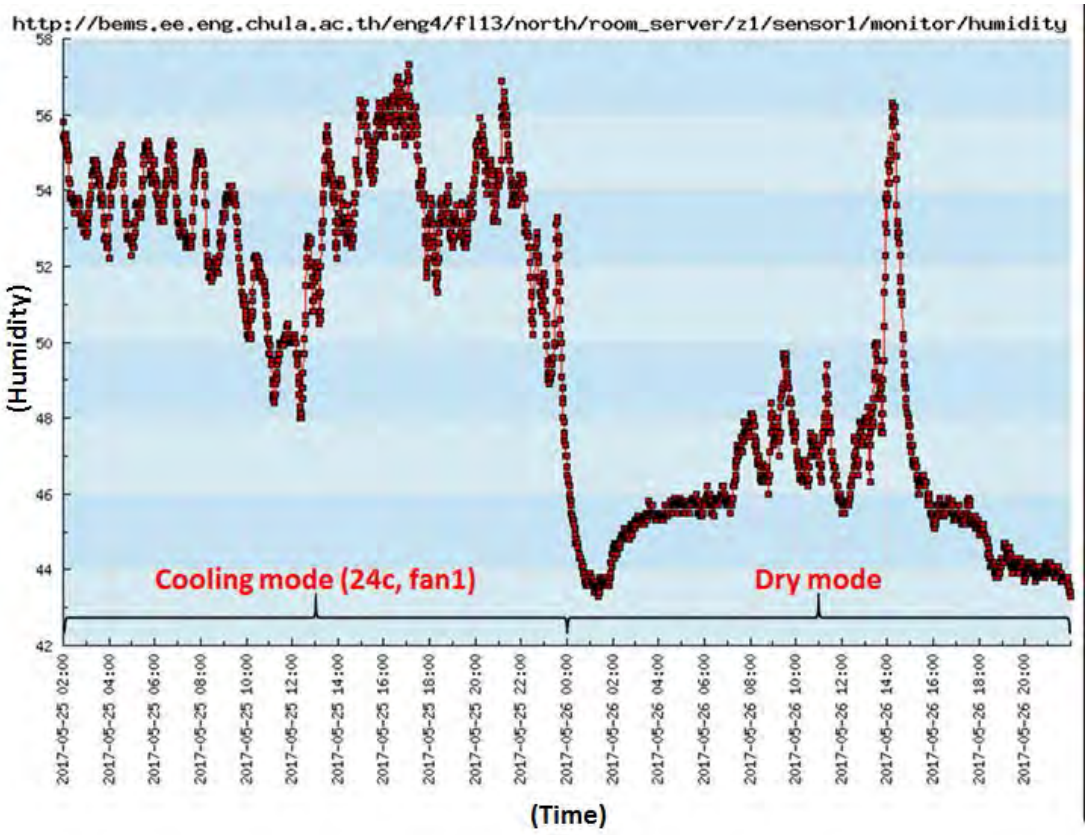
การทดสอบในลำดับต่อจากหัวข้อ 4.6.1 คือ การเปลี่ยนโหมดการทำงานของอุปกรณ์เครื่องปรับอากาศจากเดิมซึ่งอยู่ในโหมดอัตโนมัติ (auto mode) ที่อุณหภูมิ 24 องศาเซลเซียส วัดค่าอุณหภูมิภายในห้องทดสอบได้ประมาณ 24.5 องศาเซลเซียส ไปเป็นโหมดการทำความเย็น (cooling mode) ที่อุณหภูมิ 24 องศาเซลเซียส อุณหภูมิภายในห้องทดสอบจึงลดต่ำลงมาอีกเล็กน้อยอยู่ที่ประมาณ 24 องศาเซลเซียส และระดับพัดลมถูกปรับให้อยู่ในระดับ 1 จากเดิมที่ระดับพัดลมถูกปรับแบบอัตโนมัติซึ่งเป็นการทำงานปกติในโหมดอัตโนมัติ ผลการเปลี่ยนแปลงอุณหภูมิที่วัดได้ภายในห้องทดสอบ

สำหรับการทดสอบการเปลี่ยนโหมดการทำงานของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite นี้ถูกแสดงดังรูปที่ 4.35

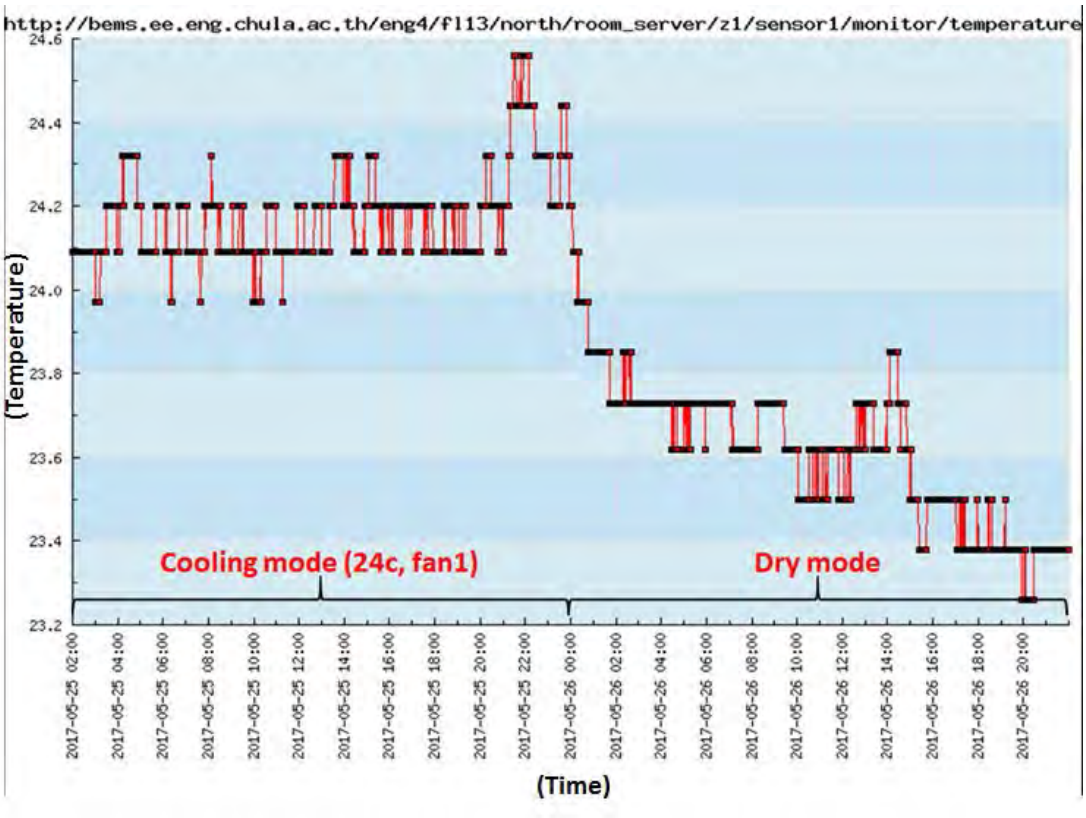


รูปที่ 4.35: อุณหภูมิที่ได้จากตัวรับรู้ CU-BEMS และตัวรับรู้ ECHONET Lite จากการทดสอบเปลี่ยนโหมดการทำงานเป็นโหมดการทำความเย็น (cooling mode) ของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite

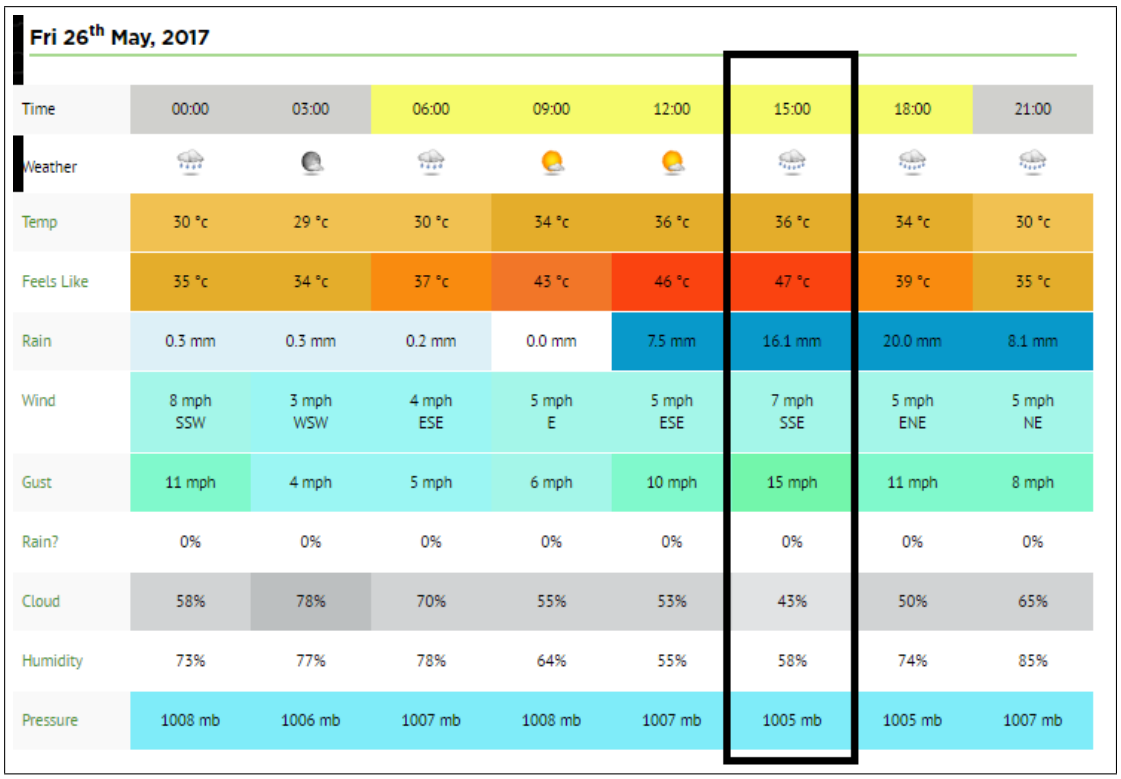
จากนั้นลำดับต่อไป คือ การทดสอบการเปลี่ยนโหมดเป็นโหมดลดความชื้น (dry mode) โดยเริ่มจากปล่อยให้อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ทำงานอยู่ในโหมดการทำความเย็น (cooling mode) ที่การตั้งค่าอุณหภูมิ 24 องศาเซลเซียส ตามปกติในช่วงแรกของวันที่ 25 เดือนพฤษภาคม พ.ศ. 2560 ต่อมาทดสอบเปลี่ยนโหมดการทำงานเป็นโหมดลดความชื้นในช่วงเวลา 00.00 น. ของวันที่ 26 เดือนพฤษภาคม พ.ศ. 2560 (วันถัดไป) จากนั้นปล่อยให้อุณหภูมิและความชื้นสัมพัทธ์ภายในห้องทดสอบเข้าสู่สภาวะคงตัว ได้ความสัมพันธ์ของการเปลี่ยนแปลงความชื้นสัมพัทธ์ ดังรูปที่ 4.36 จากตัวรับรู้ CU-BEMS เพียงแค่ตัวเดียว เนื่องจากตัวควบคุมภายในโนด ECHONET Lite มีข้อจำกัดในการวัดค่าความชื้นสัมพัทธ์ และได้ความสัมพันธ์ของการเปลี่ยนแปลงอุณหภูมิ ดังรูปที่ 4.37 จากตัวรับรู้ CU-BEMS เพียงแค่ตัวเดียว เพื่อเปรียบเทียบค่าอุณหภูมิและความชื้นสัมพัทธ์ ณ ช่วงเวลาเดียวกัน พบว่าหลังจากเปลี่ยนโหมดการทำงานเป็นโหมดลดความชื้น อุณหภูมิภายในห้องทดสอบมีค่าลดลงเล็กน้อยจนอยู่ที่ระดับประมาณ 23.4 องศาเซลเซียส จากเดิมที่ระดับประมาณ 24.2 องศาเซลเซียส และความชื้นสัมพัทธ์ภายในห้องทดสอบมีค่าลดต่ำลงอย่างเห็นได้ชัดในช่วงเวลาตั้งแต่เปลี่ยนเป็นโหมดลดความชื้น (เวลา 00.00 น. ของวันที่ 26 เดือนพฤษภาคม พ.ศ. 2560) แต่ยังมีช่วงเวลาสั้น ๆ ที่ความชื้นสัมพัทธ์มีค่าสูงขึ้นมาอีกครั้งที่ช่วงเวลาประมาณ 14.00 น. ของวันที่ 26 เดือนพฤษภาคม พ.ศ. 2560 อันเนื่องมาจากช่วงเวลาดังกล่าวมีความชื้นจากอากาศภายนอกสูงจากฝนที่ตกอย่างหนักดังการพยากรณ์อากาศวันที่ 26 เดือนพฤษภาคม พ.ศ. 2560 ในรูปที่ 4.38 ก่อนที่จะสามารถกลับไปควบคุมความชื้นสัมพัทธ์ให้มีค่าต่ำลงเหมือนเดิมได้ภายหลังฝนเริ่มตกน้อยลงในช่วงเวลา 18.00 น. เป็นต้นไป



รูปที่ 4.36: ความชื้นสัมพัทธ์ที่ได้จากตัวรับรู้ CU-BEMS ในการทดสอบเปลี่ยนโหมดการทำงานเป็นโหมดลดความชื้น (dry mode) ของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite



รูปที่ 4.37: อุณหภูมิที่ได้จากตัวรับรู้ CU-BEMS ในการทดสอบเปลี่ยนโหมดการทำงานเป็นโหมดลดความชื้น (dry mode) ของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite



Fri 26th May, 2017

Time	00:00	03:00	06:00	09:00	12:00	15:00	18:00	21:00
Weather								
Temp	30 °c	29 °c	30 °c	34 °c	36 °c	36 °c	34 °c	30 °c
Feels Like	35 °c	34 °c	37 °c	43 °c	46 °c	47 °c	39 °c	35 °c
Rain	0.3 mm	0.3 mm	0.2 mm	0.0 mm	7.5 mm	16.1 mm	20.0 mm	8.1 mm
Wind	8 mph SSW	3 mph WSW	4 mph ESE	5 mph E	5 mph ESE	7 mph SSE	5 mph ENE	5 mph NE
Gust	11 mph	4 mph	5 mph	6 mph	10 mph	15 mph	11 mph	8 mph
Rain?	0%	0%	0%	0%	0%	0%	0%	0%
Cloud	58%	78%	70%	55%	53%	43%	50%	65%
Humidity	73%	77%	78%	64%	55%	58%	74%	85%
Pressure	1008 mb	1006 mb	1007 mb	1008 mb	1007 mb	1005 mb	1005 mb	1007 mb

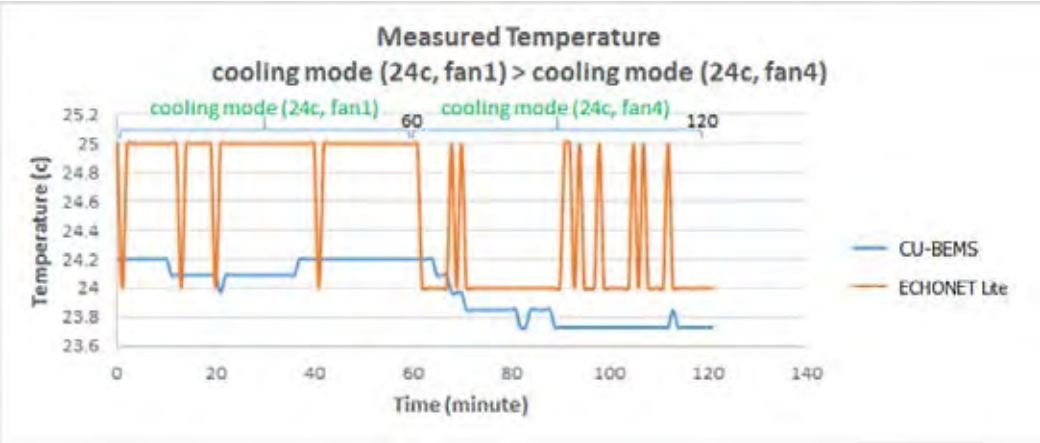
รูปที่ 4.38: การพยากรณ์อากาศวันที่ 26 เดือนพฤษภาคม พ.ศ. 2560 [21]

4.6.3 การทดสอบปรับระดับพัดลมในโหมดการทำความเย็น (cooling mode) ของอุปกรณ์เครื่องปรับอากาศตามมาตรฐาน ECHONET Lite (EPC = 0xA0)

สำหรับการทดสอบปรับระดับพัดลมในโหมดการทำงานโหมดทำความเย็น (cooling mode) จากระดับต่ำสุด (ระดับ 1) ไปเป็นระดับสูงสุด (ระดับ 4) ที่การตั้งค่าอุณหภูมิ 24 องศาเซลเซียส ได้ผลของอุณหภูมิที่วัดได้จากตัวรับรู้ทั้งสองมาตรฐานภายในห้องทดสอบดังรูปที่ 4.39 จะเห็นได้ว่าเมื่อตั้งค่าระดับพัดลมอยู่ที่ระดับต่ำสุด (ระดับ 1) ไว้สักพักจะได้ค่าอุณหภูมิภายในห้องทดสอบรักษาระดับคงที่ที่ประมาณ 24.2 องศาเซลเซียส หลังจากนั้นปรับการตั้งค่าระดับพัดลมไปที่ระดับสูงสุด (ระดับ 4) ทำให้อุณหภูมิที่วัดได้ภายในห้องทดสอบมีค่าลดลงจากเดิมเล็กน้อยอยู่ที่ระดับประมาณ 23.8 องศาเซลเซียส สำหรับค่าที่วัดได้จากตัวรับรู้ CU-BEMS และลดลงอยู่ที่ระดับประมาณ 24 องศาเซลเซียส จากเดิมที่ระดับประมาณ 25 องศาเซลเซียส สำหรับค่าที่วัดได้จากตัวรับรู้ ECHONET Lite

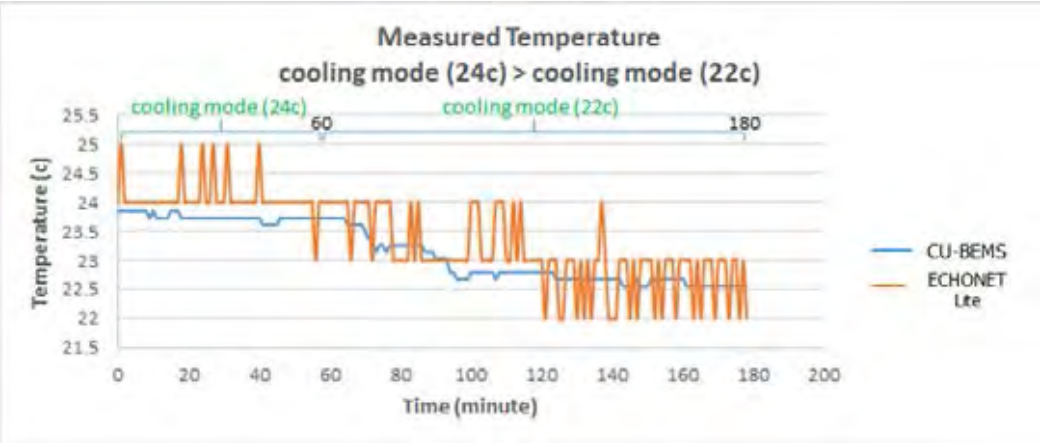
4.6.4 การทดสอบปรับการตั้งค่าอุณหภูมิในโหมดการทำความเย็น (cooling mode) ของอุปกรณ์เครื่องปรับอากาศตามมาตรฐาน ECHONET Lite (EPC = 0xB3)

จากผลการทดสอบในการปรับการตั้งค่าอุณหภูมิในโหมดการทำงานโหมดการทำความเย็น (cooling mode) วัดค่าอุณหภูมิที่เปลี่ยนแปลงภายในห้องทดสอบได้ดังรูปที่ 4.40 แสดงให้เห็นว่าเมื่อปรับการตั้งค่าอุณหภูมิในโหมดการทำความเย็นจากเดิม 24 องศาเซลเซียส ลดลงเหลือ 22 องศาเซลเซียส แล้ว ส่งผลให้อุณหภูมิที่วัดได้จากตัวรับรู้ทั้งสองมาตรฐานภายในห้องทดสอบลดลงอย่างต่อเนื่องจากเดิมที่ระดับประมาณ 24 องศาเซลเซียส จนกระทั่งอยู่ที่ระดับประมาณ 22.5 องศาเซล-



รูปที่ 4.39: อุณหภูมิที่ได้จากตัวรับรู้ CU-BEMS และตัวรับรู้ ECHONET Lite จากการทดสอบปรับระดับพัดลมในโหมดการทำความเย็น (cooling mode)

เซียส สำหรับค่าที่วัดได้จากตัวรับรู้ CU-BEMS และจากระดับประมาณ 24 องศาเซลเซียส ลดลงจนกระทั่งระดับประมาณ 23 องศาเซลเซียส สำหรับค่าที่วัดได้จากตัวรับรู้ ECHONET Lite สาเหตุที่อุณหภูมิที่วัดได้จากตัวรับรู้ CU-BEMS และตัวรับรู้ ECHONET Lite ภายในห้องทดสอบมีค่าแตกต่างกันบ้างเนื่องมาจากตัวรับรู้ ECHONET Lite นั้นถูกติดตั้งแบบแนบติดกับอุปกรณ์เครื่อง-ปรับอากาศมาตรฐาน ECHONET Lite ต่างจากตัวรับรู้ CU-BEMS ซึ่งถูกติดตั้งในตำแหน่งที่ห่างจากอุปกรณ์เครื่องปรับอากาศออกไป จึงทำให้อุณหภูมิที่วัดได้จากตัวรับรู้ ECHONET Lite มีค่าต่ำกว่าอุณหภูมิที่วัดได้จากตัวรับรู้ CU-BEMS อยู่เล็กน้อย



รูปที่ 4.40: อุณหภูมิที่ได้จากตัวรับรู้ CU-BEMS และตัวรับรู้ ECHONET Lite จากการทดสอบปรับการตั้งค่าอุณหภูมิในโหมดการทำความเย็น (cooling mode) ของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite

4.6.5 การทดสอบการตั้งค่าส่ายลมอัตโนมัติของพัดลม (automatic swing of air flow setting) ตามมาตรฐาน ECHONET Lite (EPC = 0xA3)

เมื่อทำการควมคุมอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite โดยการส่งชุดคำสั่งจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เพื่อตั้งค่าในโหมดการส่ายลมอัตโนมัติของพัดลม (automatic swing of air flow setting) พบว่าใบพัดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ทำงานโดยการส่ายขึ้นและลงตามแนวแกนตั้ง (vertical swing) อย่างต่อเนื่องแบบเดียวกันกับการตั้งค่าการส่ายลมโดยตรงจากรีโมตคอนโทรล (remote control) แสดงให้เห็นว่าการทำงานในโหมดส่ายลมอัตโนมัติของพัดลมสามารถถูกปรับตั้งได้ผ่านทางมาตรฐาน ECHONET Lite

4.6.6 การทดสอบการตั้งค่าทิศทางของพัดลมตามแนวแกนตั้ง (vertical) ตามมาตรฐาน ECHONET Lite (EPC = 0xA4)

หลังจากทดสอบอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite โดยการส่งชุดคำสั่งจากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เพื่อตั้งค่าทิศทางของพัดลมตามแนวแกนตั้ง (vertical) ให้อยู่ในระดับต่าง ๆ เช่น ระดับบน ระดับกลาง ระดับล่าง เป็นต้น พบว่าใบพัดของอุปกรณ์เครื่องปรับอากาศถูกปรับให้อยู่ในระดับบน ระดับกลาง หรือระดับล่าง ตามการตั้งค่าทิศทางของพัดลมผ่านมาตรฐาน ECHONET Lite ได้

จากการทดสอบการทำงานในโหมดต่าง ๆ ของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ภายในห้องทดสอบจะสังเกตุว่าอุณหภูมิที่วัดได้จากตัวรับรู้มาตรฐาน ECHONET Lite มีค่าเป็นจำนวนเต็มเท่านั้นซึ่งไม่ละเอียดเท่ากับอุณหภูมิที่วัดได้จากตัวรับรู้ไร้สายในระบบ CU-BEMS เนื่องจากตัวรับรู้ ECHONET Lite ส่งค่าอุณหภูมิที่วัดได้ไปยังโนด ECHONET Lite เป็นจำนวนเต็มในรูปฐาน 16 ก่อนที่โนด ECHONET Lite จะส่งค่าต่อไปยังอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ และค่าของอุณหภูมิที่วัดได้จากตัวรับรู้มาตรฐาน ECHONET Lite มีค่าต่ำกว่าค่าของอุณหภูมิที่วัดได้จากตัวรับรู้ CU-BEMS อยู่เล็กน้อย เนื่องจากตัวรับรู้มาตรฐานฐาน ECHONET Lite ถูกติดตั้งแบบแนบติดกับอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ซึ่งอยู่ในตำแหน่งที่ใกล้กับอุปกรณ์เครื่องปรับอากาศมากกว่าตัวรับรู้ CU-BEMS ที่ถูกติดตั้งอยู่ไกลออกไป

บทที่ 5

บทสรุปและข้อเสนอแนะ

5.1 บทสรุป

วิทยานิพนธ์ฉบับนี้นำเสนอระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ในระบบจัดการพลังงานภายในอาคารสำหรับการประสานข้อมูลแบบทันที เพื่อให้อุปกรณ์ที่ใช้งานต่างมาตรฐานกันสามารถทำงานร่วมกันได้ ตัวอย่างเช่น ตัวรับรู้การเคลื่อนไหวของคน ตัวรับรู้อุณหภูมิ อุปกรณ์เครื่องปรับอากาศ เป็นต้น โดยการพัฒนาอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ซึ่งทำหน้าที่เป็นเสมือนตัวกลางในการประสานข้อมูลระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ในระบบทดสอบ CU-BEMS อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์นี้ถูกพัฒนาขึ้นบนบอร์ดสำเร็จรูป Raspberry Pi B+ ที่ใช้งานแพลตฟอร์ม node.js ซึ่งเป็นแพลตฟอร์มที่เหมาะสมในการใช้งานสำหรับการแจ้งเตือนข้อมูลแบบทันที กรณีแรกในการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite นั้นอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ใช้โพรโทคอล TRAP เพื่อร้องขอการแจ้งเตือนการเปลี่ยนแปลงข้อมูลในหน่วยเก็บข้อมูลระบบ CU-BEMS ที่ใช้งานตามมาตรฐาน IEEE1888 และใช้การร้องขอแบบ WRITE หรือ READ ในการควบคุมอุปกรณ์เครื่องปรับอากาศตามมาตรฐาน ECHONET Lite กรณีที่สองในการประสานข้อมูลแบบทันทีจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 อินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ใช้การร้องขอแบบ NOTIFY เพื่อร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite และใช้โพรโทคอล WRITE ตามมาตรฐาน IEEE1888 เพื่อเก็บข้อมูลการเปลี่ยนแปลงไปยังหน่วยเก็บข้อมูลระบบ CU-BEMS

การทดสอบการประสานข้อมูลแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite มีอุปกรณ์ที่ใช้ในการทดสอบประกอบด้วย หน่วยเก็บข้อมูลระบบ CU-BEMS ตัวรับรู้ไร้สายในระบบ CU-BEMS อุปกรณ์จัดเส้นทาง โนด ECHONET Lite อุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite และอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์ จากการทดสอบพบว่าในการประสานข้อมูลแบบทันทีจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite ใช้ขนาดของข้อความในการแจ้งเตือนการเปลี่ยนแปลงข้อมูลด้วยโพรโทคอล TRAP ตามมาตรฐาน IEEE1888 ทั้งหมดเท่ากับ 2,296 ไบต์ และใช้ขนาดของข้อความในการส่งไปควบคุมอุปกรณ์เครื่องปรับอากาศด้วยการร้องขอแบบ WRITE ตามมาตรฐาน ECHONET Lite ทั้งหมดเท่ากับ 117 ไบต์ ส่วนในการประสานข้อมูลแบบทันทีจากมาตรฐาน ECHONET Lite ไปยังมาตรฐาน IEEE1888 นั้นใช้ขนาดของข้อความในการแจ้งเตือนการเปลี่ยนแปลงสถานะของอุปกรณ์เครื่องปรับอากาศด้วยการร้องขอแบบ NOTIFY ตามมาตรฐาน ECHONET Lite ทั้งหมดเท่ากับ 120 ไบต์ และใช้ขนาดของข้อความในการส่งข้อมูลการเปลี่ยนแปลงไปเก็บยังหน่วยเก็บข้อมูลด้วยโพรโทคอล WRITE ตามมาตรฐาน IEEE1888 ทั้งหมดเท่ากับ 740 ไบต์ ดังนั้นเมื่อเปรียบเทียบขนาดของข้อความที่ใช้ในการแจ้งเตือนข้อมูลแบบทันทีระหว่างมาตรฐานแล้ว พบว่ามาตรฐาน IEEE1888 ใช้ขนาดของข้อความมากกว่าขนาดที่ใช้ในมาตรฐาน ECHONET Lite ต่อการแจ้งเตือนข้อมูล 1 ครั้ง จึงทำให้ค่าแบนด์วิดท์ที่ใช้ในการสื่อสารมาตรฐาน IEEE1888 มากกว่าในการสื่อสารมาตรฐาน ECHONET Lite จากผลการทดสอบยังได้ขนาดของข้อความที่ใช้ในการประสานข้อมูลแบบคาบเวลาจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ECHONET Lite ทั้งหมดเท่ากับ 945 ไบต์ และจากมาตรฐาน ECHONET Lite ไปยัง

มาตรฐาน IEEE1888 ทั้งหมดเท่ากับ 856 ไบต์ และเมื่อทำการทดสอบการประสานข้อมูลแบบทันที โดยการเพิ่มความถี่ในการแจ้งเตือนการเปลี่ยนแปลงข้อมูลต่อหนึ่งนาที พบว่าช่วงเวลาที่ใช้ในการประสานข้อมูลแบบทันทีแบบครบรอบสมบูรณ์มีค่ายาวนานขึ้นเป็น 424 มิลลิวินาที สำหรับการประสานข้อมูล 50 ครั้งต่อนาที เมื่อเปรียบเทียบกับช่วงเวลาที่ใช้ในการประสานข้อมูล 1 ครั้งต่อนาที ซึ่งใช้เวลา 178 มิลลิวินาที เนื่องจากปริมาณทราฟฟิกในระบบที่เพิ่มมากขึ้น

นอกจากนี้วิทยานิพนธ์ฉบับนี้ได้ทดสอบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite เพื่อควบคุมอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ในโหมดการใช้งานต่าง ๆ ได้แก่ การตั้งค่าการเปิดปิด การตั้งค่าโหมดการทำงาน การตั้งค่าอุณหภูมิ การตั้งค่าระดับหรือทิศทางของพัดลม การตั้งค่าส่ายลมอัตโนมัติของพัดลม และการตั้งค่าทิศทางของพัดลมตามแนวแกนตั้ง การทดสอบนี้ถูกทดสอบในห้องเซิร์ฟเวอร์ที่มีเครื่องคอมพิวเตอร์เซิร์ฟเวอร์และอุปกรณ์อื่น ๆ ซึ่งทำงานอยู่จริง จากผลการทดสอบพบว่าการควบคุมการทำงานของอุปกรณ์เครื่องปรับอากาศในโหมดต่าง ๆ ดังกล่าว สามารถถูกควบคุมผ่านมาตรฐาน ECHONET Lite ได้ การพัฒนาการทำงานร่วมกันแบบทันทีระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite สำหรับระบบจัดการพลังงานภายในอาคารในวิทยานิพนธ์ฉบับนี้คาดว่าจะมีประโยชน์อย่างยิ่งในการเพิ่มขีดความสามารถและความยืดหยุ่นของการสื่อสารระหว่างอุปกรณ์ที่ใช้งานต่างมาตรฐานกันซึ่งมีความจำเป็นสำหรับทิศทางการพัฒนาโครงข่ายอินเทอร์เน็ตของสรรพสิ่งต่อไปในอนาคต

5.2 ข้อเสนอแนะ

งานวิจัยในวิทยานิพนธ์ฉบับนี้มีแนวทางในการวิจัยต่อไปโดยการพัฒนาแอพพลิเคชันสำหรับผู้ใช้งานในการควบคุมการทำงานของตัวกระตุ้นหรืออุปกรณ์ไฟฟ้าชนิดต่าง ๆ ที่ถูกเชื่อมต่ออยู่ภายในระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ECHONET Lite ตามความเหมาะสมของลักษณะการใช้งาน ตลอดจนสามารถเข้าถึงข้อมูลตัวรับรู้ของทั้งสองมาตรฐานและนำข้อมูลไปวิเคราะห์ให้เกิดประโยชน์ต่อระบบจัดการพลังงาน

การวิเคราะห์ผลกระทบในการขยายโครงข่ายของระบบการทำงานร่วมกันระหว่างมาตรฐานสามารถวิจัยต่อได้โดยการติดตั้งโนดและอุปกรณ์มาตรฐาน ECHONET Lite เพิ่มขึ้นจากเดิมในวิทยานิพนธ์นี้ซึ่งพิจารณาโนด ECHONET Lite จำนวน 1 โนดเท่านั้นเพื่อควบคุมอุปกรณ์เครื่องปรับอากาศซึ่งรองรับมาตรฐาน ECHONET Lite ดังนั้นจึงมีความน่าสนใจในการพัฒนาและทดสอบอุปกรณ์ชนิดอื่น ๆ ซึ่งรองรับมาตรฐาน ECHONET Lite เช่น ตัวรับรู้ความชื้นสัมพัทธ์ ตัวกระตุ้นชนิดหลอดไฟ อุปกรณ์ไฟฟ้าชนิดต่าง ๆ เป็นต้น ซึ่งจะมีความยุ่งยากเพิ่มมากขึ้นสำหรับการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 ในระบบ CU-BEMS และมาตรฐาน ECHONET Lite ในเรื่องการจัดการข้อมูลที่ได้จากตัวรับรู้ชนิดต่าง ๆ และการควบคุมการทำงานไปยังอุปกรณ์ไฟฟ้าหรือตัวกระตุ้นแต่ละชนิด

นอกจากนี้ในงานวิจัยที่ผ่านมาจนถึงปัจจุบันได้มีการพัฒนาการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 ที่ใช้งานในระบบ CU-BEMS กับมาตรฐาน ETSI M2M [13], [14] และมาตรฐาน ECHONET Lite [22] ในงานวิจัยนี้ แต่มีโพรโทคอลอื่น ๆ ที่น่าสนใจและกำลังได้รับความนิยมซึ่งยังไม่ได้พิจารณา เช่น โพรโทคอล MQTT [23] เป็นต้น ดังนั้นงานวิจัยเพื่อเพิ่มขีดความสามารถในการสื่อสารระหว่างอุปกรณ์ที่มีหลากหลายชนิดและใช้งานต่างมาตรฐานกันให้สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพในระบบจัดการพลังงาน จึงเป็นงานวิจัยในอนาคตที่สำคัญ ทั้งนี้เพื่อสามารถพัฒนาขึ้นเป็นต้นแบบของการสื่อสารระหว่างอุปกรณ์ภายในโครงข่ายอินเทอร์เน็ตของสรรพสิ่งได้ต่อไปในอนาคต

รายการอ้างอิง

[1] Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., and Oliveira, A. Smart cities and the future internet: Towards cooperation frameworks for open innovation. The Future Internet, Lect. Notes Comput. Sci. 6656 (2011): 431–446.

[2] Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. Internet of Things for smart cities. IEEE Internet of Things Journal 1 (2014): 22-32.

[3] Lowe, G. Driving the internet of things. IEEE Design and Test 31 (2014): 22-27.

[4] Weyrich, M., Schmidt, J. P., and Ebert, C. Machine-to-Machine communication. IEEE Software 31 (2014): 19-23.

[5] Datta, S. K., and Bonnet, C. Internet of Things and M2M communications as enablers of smart city initiatives. The 9th International Conference on Next Generation Mobile Applications, Services and Technologies, (2015): 393-398.

[6] IEEE. IEEE standard for ubiquitous green community control network protocol. IEEE Std 1888, 2011.

[7] รายงานฉบับสิ้นสุดโครงการ โครงการวิจัยและพัฒนาเทคโนโลยีระบบโครงข่ายไฟฟ้าอัจฉริยะเพื่อบริหารจัดการการใช้พลังงานไฟฟ้าของอาคาร จุฬาลงกรณ์มหาวิทยาลัย (สถาบันวิจัยพลังงาน) เสนอกองทุนเพื่อส่งเสริมการอนุรักษ์พลังงานสำนักงานนโยบายและแผนพลังงาน, กันยายน 2557.

[8] Le, D. H., and Pora, W. Development of smart meter for building energy management system based on the IEEE 1888 standard with Wi-Fi communication. The 2014 International Conference Electronics Information and Communication (ICEIC 2014), 2014.

[9] Inthasut, T., and Aswakul, C. ZigBee wireless sensor network with IEEE1888 gateway for building energy management system. The 2014 International Conference on Electronics, Information and Communications (ICEIC 2014), 2014.

[10] Khawsa-ard, P., and Aswakul, C. Application of simple computer board game with gesture sensor input for increasing awareness in electrical energy consumption. The 29th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2014), 2014.

[11] ECHONET Lite specification, version 1.11. Energy Conservation and Homecare Network, 2014.

[12] Murakami, T., Sugimura, H., and Isshiki, M. Application of ECHONET Lite which is open standard into energy management system. The IEEE International Conference on Consumer Electronics (ICCE 2016), (2016): 455-458.

[13] Klinpratum, T., Saivichit, C., Elmangoush, A., and Magedanz, T. Toward interconnecting M2M/IoT standards: Interworking proxy for IEEE1888 standard at ETSI M2M platform. The 29th International Technical Conference on Circuit/Systems Computers and Communications (ITC-CSCC 2014), 2014.

[14] Kosolworrawattanakul, N., Elmangoush, A., Magedanz, T., and Aswakul, C. Development of real-time data synchronization for IEEE1888 and ETSI M2M standards. The IEICE Technical Report, 2014.

[15] CU-BEMS website [Online]. Available: http://www.bems.ee.eng.chula.ac.th:9061/bems.web [Accessed: 14 February 2017].

[16] CU-BEMS android application [Online]. Available: https://play.google.com/store/apps /details?id=com.app.cuee_bems [Accessed: 14 February 2017].

[17] Khawsa-ard, P., and Aswakul, C. IEEE1888 interactive display as a service (IDaaS): Example in building energy management system. COMPSAC 2015: The 39th Annual International Computers, Software and Applications Conference, 2015.

[18] English version of appendix. Detailed Requirements for ECHONET Device Objects, Release H, 2017.

[19] Tilkov, S., and Vinoski, S. Node.js: Using JavaScript to build high-performance network programs. Internet Computing 14 (2010): 80-83.

[20] Node module echonet-lite [Online]. Available: https://www.npmjs.com/package/echonet-lite [Accessed: 14 February 2017].

[21] Bangkok weather history [Online]. Available: https://www.worldweatheronline.com/bangkok-weather-history/krung-thep/th.aspx [Accessed: 28 May 2017].

[22] Sangumpai, C., and Aswakul, C. Development of real-time interworking between IEEE1888 and ECHONET Lite standards for building energy management system. The 39th Electrical Engineering Conference (EECON-39), Phetchaburi City, Thailand, 2016.

[23] Chen, H. W., and Lin, F. J. Converging MQTT resources in ETSI standards based M2M platform. The 2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom), (2014): 292-295.

ภาคผนวก

ก โปรแกรม node.js สำหรับส่งการร้องขอการแจ้งเตือน TRAP (TRAP query) ตามเงื่อนไขการแจ้งเตือนการเปลี่ยนแปลงข้อมูลในหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 (trap_query.js)

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 1-4: การประกาศตัวแปร http และ uuid สำหรับอ้างอิงถึงมอดูล http และ node-uuid ตามลำดับ
- บรรทัดที่ 7-12: การประกาศตัวแปร trap_body ชนิดอักขระ (string) เพื่อเป็นเพย์โหลดสำหรับกำหนดเงื่อนไขการแจ้งเตือนการเปลี่ยนแปลงข้อมูลตามโพรโทคอล TRAP โดยใช้โพรโทคอล SOAP ในการเข้าถึงอ็อบเจกต์ และรูปแบบข้อความ XML ภายในเพย์โหลดมีการระบุเลขที่อยู่ไอพีของผู้เรียกกลับข้อมูล (data callback) และผู้เรียกกลับการควบคุม (control callback) นอกจากนี้ยังมีการระบุ point ID ที่ต้องการแจ้งเตือนข้อมูล และอายุของข้อความ TRAP query (ttl)
- บรรทัดที่ 15-25: การประกาศตัวแปร postRequest ชนิดอ็อบเจกต์เพื่อเป็นส่วนหัว (header) ของการร้องขอ http ภายในอ็อบเจกต์มีการระบุเลขที่อยู่ไอพีแม่ข่าย (host) คือ หน่วยเก็บข้อมูล IEEE1888 มีการระบุวิถี (path) ช่องทาง (port) วิธีการร้องขอ http แบบ post และยังมีการกำหนดอ็อบเจกต์ headers ซึ่งเป็นการกำหนดตามรูปแบบ XML ที่มีอักขระแบบ UTF-8
- บรรทัดที่ 28-33: การประกาศตัวแปร trap_req เพื่อเป็นส่วนย่อย (element) สำหรับเรียกใช้ฟังก์ชันเรียกกลับ request ในอ็อบเจกต์ http โดยมีอินพุตอาร์กิวเมนต์ 2 ตัว คือ postRequest และฟังก์ชันที่มีอินพุตอาร์กิวเมนต์เป็นตัวแปร res สำหรับการรับข้อมูลตอบกลับจากหน่วยเก็บข้อมูล IEEE1888
- บรรทัดที่ 29: การแสดงผลของสถานะการร้องขอ http จากส่วนย่อย statusCode ในตัวแปร res
- บรรทัดที่ 30: การประกาศตัวแปร buffer ชนิดอักขระว่างสำหรับพักข้อมูลอักขระที่ได้รับจากการตอบกลับของหน่วยเก็บข้อมูล IEEE1888
- บรรทัดที่ 31: การกำหนดฟังก์ชัน on ของส่วนย่อย res ที่มีการเรียกใช้ฟังก์ชันเรียกกลับสำหรับการรับข้อมูลอักขระ data ไปยังตัวแปร buffer เมื่อมีเหตุการณ์ที่เป็นอ็อบเจกต์ data เข้ามา
- บรรทัดที่ 32: การกำหนดฟังก์ชัน on ของส่วนย่อย res ที่มีการเรียกใช้ฟังก์ชันเรียกกลับสำหรับแสดงข้อมูลอักขระที่ได้รับมาจากตัวแปร buffer เมื่อมีเหตุการณ์ที่เป็นอ็อบเจกต์ end เข้ามา แสดงให้เห็นว่าการรับข้อมูลอักขระเสร็จสิ้นแล้ว
- บรรทัดที่ 36-38: การกำหนดฟังก์ชัน on ของส่วนย่อย trap_req ที่มีการเรียกใช้ฟังก์ชันเรียกกลับสำหรับแสดงข้อผิดพลาดจากส่วนย่อย message ของอินพุตอาร์กิวเมนต์ที่เป็นอ็อบเจกต์ของเหตุการณ์ e เมื่อเกิดข้อผิดพลาดในการร้องขอ http

- บรรทัดที่ 41: การเรียกใช้ฟังก์ชัน write ของส่วนย่อย trap_req สำหรับส่งการร้องขอ http ด้วยข้อมูลเพย์โหลด trap_body ไปยังหน่วยเก็บข้อมูล IEEE1888
- บรรทัดที่ 42: การเรียกใช้ฟังก์ชัน end ของส่วนย่อย trap_req สำหรับการสิ้นสุดการร้องขอ http
- บรรทัดที่ 45: การกำหนดตัวแปร minutes และ time_interval สำหรับใช้เป็นเวลาหน่วงในฟังก์ชัน setInterval
- บรรทัดที่ 46-55: การเรียกใช้ฟังก์ชัน setInterval สำหรับการร้องขอการแจ้งเตือนตามเงื่อนไขการแจ้งเตือนการเปลี่ยนแปลงข้อมูลด้วยการระบุอินพุตอาร์กิวเมนต์ 2 ตัว คือ ฟังก์ชันที่ใช้สำหรับส่งการร้องขอ http แบบเดียวกันกับบรรทัดที่ 28-42 และตัวแปร time_interval ซึ่งเป็นค่าบเวลาที่ใช้ในการหน่วง (มิลลิวินาที)

```
1  // require 'http' module.
2  var http = require('http');
3  // require 'node-uuid' module for generating query id.
4  var uuid = require('node-uuid');
5
6  // iniatial trap query body.
7  var trap_body = '<?xml version="1.0" encoding="utf-8"?>' +
8  '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
       envelope/">'+
9  '<soapenv:Body><ns2:queryRQ xmlns:ns2="http://soap.fiap.org/"><
       transport xmlns="http://gutp.jp/fiap/2009/11/">'+
10 '<header><query id="'+uuid.v4()+'" type="stream" ttl="300"
       callbackData="http://161.200.90.103"  callbackControl="http
       ://161.200.90.103">'
11 +'<key id="http://bems.ee.eng.chula.ac.th/eng4/fl13/north/
       room_server/z1/sensor1/monitor/pir"  attrName="value" trap="
       changed" />'
12 +'</query></header></transport></ns2:queryRQ>'+'</soapenv:Body></
       soapenv:Envelope>';
13
14 // iniatial header of http request.
15 var postRequest = {
16   host: "161.200.90.122",
17   path: "/axis2/services/FIAPStorage",
18   port: 80,
19   method: "POST",
20   headers: {
21     'Content-Type': 'text/xml charset=UTF-8',
22     'SOAPAction': 'http://soap.fiap.org/query',
23     'Content-Length': Buffer.byteLength(trap_body)
24   }
25 };
```

```
26
27 // iniatial http request for receiving data.
28 var trap_req = http.request( postRequest, function ( res ) {
29   console.log( res.statusCode );
30   var buffer = "";
31   res.on( "data", function( data ) { buffer = buffer + data; } );
32   res.on( "end", function( data ) { console.log( buffer ); } );
33 });
34
35 // show error message for http request when any errors occur.
36 trap_req.on('error', function(e) {
37   console.log('problem with request: ' + e.message);
38  });
39
40 // send http request.
41 trap_req.write( trap_body );
42 trap_req.end();
43
44 // set loop function to send http request again after previous trap
       time out.
45 var minutes = 5, time_interval = minutes * 60 * 1000;
46 setInterval(function() {
47   var trap_req = http.request( postRequest, function ( res ) {
48     console.log( res.statusCode );
49       var buffer = "";
50       res.on( "data", function( data ) { buffer = buffer + data; } )
             ;
51       res.on( "end", function( data ) { console.log( buffer ); } );
52   });
53   trap_req.write( trap_body );
54   trap_req.end();
55 }, time_interval);
```

ข โปรแกรม node.js สำหรับการรับการตอบกลับการเปลี่ยนแปลงข้อมูลตัวรับรู้จากหน่วยเก็บข้อมูล IEEE1888 ตามโพรโทคอล TRAP เพื่อใช้ควบคุมการเปิดปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ด้วยวิธี WRITE (trap_callback_pi.js)

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 2-5: การประกาศตัวแปร EL http parseString และ moment สำหรับอ้างอิงถึงมอดูล echonet-lite http xml2js และ moment ตามลำดับ
- บรรทัดที่ 8: การประกาศตัวแปร trap_query อ้างอิงถึงโปรแกรม trap_query.js สำหรับส่งการร้องขอการแจ้งเตือน TRAP ตามเงื่อนไขการแจ้งเตือนการเปลี่ยนแปลงข้อมูลในหน่วยเก็บข้อมูลมาตรฐาน IEEE1888 ดังภาคผนวก ก
- บรรทัดที่ 11: การประกาศตัวแปร objList ชนิดแถวลำดับของอักขระที่อยู่ในรูปเลขฐาน 16 สำหรับใช้เป็นออบเจกต์ของตัวควบคุม (controller) ตามการสื่อสารมาตรฐาน ECHONET Lite
- บรรทัดที่ 14-31: การประกาศตัวแปร elsocket เพื่อเรียกใช้ฟังก์ชัน initialize ในมอดูล EL โดยมีอินพุตอาร์กิวเมนต์ 2 ตัว ประกอบด้วย ออบเจกต์ของตัวควบคุม objList และฟังก์ชันเรียกกลับที่มีตัวแปร rinfo และ els ฟังก์ชันเรียกกลับนี้จะถูกเรียกใช้เมื่อมีข้อมูล ECHONET Lite เข้ามา เพื่อแสดงข้อมูลที่ได้รับมาเหล่านั้น
- บรรทัดที่ 34: การประกาศตัวแปร ok_body ชนิดอักขระสำหรับใช้เป็นข้อความตอบกลับ OK ตามโพรโทคอล TRAP ในมาตรฐาน IEEE1888 ซึ่งเป็นเพย์โหลดสายอักขระ โดยใช้โพรโทคอล SOAP และมีรูปแบบภาษาเป็น XML
- บรรทัดที่ 37: การประกาศตัวแปร server เป็นส่วนย่อยของฟังก์ชัน createServer ในอ็อบเจกต์ http ภายในฟังก์ชัน createServer มีอินพุตอาร์กิวเมนต์คือฟังก์ชันเรียกกลับซึ่งทำหน้าที่รับและส่งเมื่อมีการร้องขอหรือการตอบกลับเข้ามาในการสื่อสาร http โดยมีอินพุตอาร์กิวเมนต์ประกอบด้วย ส่วนย่อย req และ res การทำงานของฟังก์ชันแสดงบรรทัดที่ 37-97
- บรรทัดที่ 38: การประกาศตัวแปร body เป็นอักขระว่างสำหรับการบันทึกอักขระที่ได้รับจากการตอบกลับข้อมูลจากหน่วยเก็บข้อมูล IEEE1888
- บรรทัดที่ 41: การเรียกใช้ฟังก์ชัน setEncoding ในส่วนย่อย req ด้วยอินพุตอาร์กิวเมนต์ คือ 'utf8' สำหรับการแปลงข้อมูลการร้องขอให้อยู่ในรูปแบบอักขระ utf8
- บรรทัดที่ 44: การเรียกใช้ฟังก์ชัน on ในส่วนย่อย req โดยจะกระทำฟังก์ชันเรียกกลับที่มีอินพุตอาร์กิวเมนต์ คือ ตัวแปร chunk เมื่อมีเหตุการณ์ data เข้ามา สำหรับการรับข้อมูลจากหน่วยเก็บข้อมูล IEEE1888 และเข้าสู่กระบวนการแจกแจงข้อมูล (data parsing) ดังบรรทัดที่ 44-86
- บรรทัดที่ 45: การบันทึกข้อมูลที่ได้รับมาจากส่วนย่อย chunk ไปเก็บยังตัวแปรอักขระ body อย่างต่อเนื่อง

- บรรทัดที่ 48-51: การเรียกใช้ฟังก์ชันของส่วนย่อย parseString ที่มีอินพุตอาร์กิวเมนต์ 2 ตัว คือ ตัวแปรอักขระ body และฟังก์ชันเรียกกลับสำหรับการแปลงรูปแบบข้อมูลที่ได้รับมาในตัวแปรอักขระ body จากรูปแบบ XML เป็น JSON ด้วยอินพุตอาร์กิวเมนต์ err และ result แล้วจึงบันทึกไปยังตัวแปร jsonbody ก่อนที่จะถูกนำไปใช้ในการแจกแจงข้อมูล
- บรรทัดที่ 49: การกำหนดตัวแปร jsonbody เป็นอ็อบเจกต์ที่ได้จากฟังก์ชัน stringify ในส่วนย่อย JSON ที่มีอินพุตอาร์กิวเมนต์ คือ result
- บรรทัดที่ 53-55: การประกาศตัวแปรแถวลำดับ (array) คือ idc timec และ valuec สำหรับการเก็บข้อมูลของ point ID ตราเวลา และค่าที่ได้จากการแจกแจงข้อมูล ตามลำดับ
- บรรทัดที่ 58-68: การแจกแจงข้อมูลของตัวแปรชนิดอ็อบเจกต์ jsonbody เป็น key และ value จากนั้นทำการตรวจสอบ key และ value เพื่อบันทึกข้อมูล point id ตราเวลา และค่าในตัวแปรแถวลำดับ idc timec และ valuec ตามลำดับ
- บรรทัดที่ 71-74: การนำข้อมูลจากแถวลำดับ idc เฉพาะข้อมูลที่มีการแจ้งเตือนบันทึกในแถวลำดับ id
- บรรทัดที่ 78-81: การสร้างเงื่อนไข เมื่อค่าในแถวลำดับ valuec มีค่าเท่ากับ 'ON' แล้วจึงส่งชุดคำสั่งเพื่อควบคุมการเปิดอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ไปยังโนด ECHONET Lite ที่มีเลขที่อยู่ไอพี คือ 192.168.0.100 ด้วยการระบุค่า ESV เท่ากับ 0x61 EPC เท่ากับ 0x80 และค่า EDT เท่ากับ 0x30
- บรรทัดที่ 89: การกำหนดฟังก์ชัน on ของส่วนย่อย req ให้กระทำฟังก์ชันในบรรทัดที่ 89-96 เมื่อมีเหตุการณ์ end เข้ามา
- บรรทัดที่ 90-94: การกำหนดฟังก์ชัน writeHead ของส่วนย่อย res สำหรับเป็นส่วนหัวในการร้องขอ http ที่มีอินพุตอาร์กิวเมนต์ 2 ตัว ประกอบด้วย รหัสสถานะ 200 และออบเจกต์สำหรับเป็นส่วนหัวซึ่งมีการระบุเนื้อหาตามรูปแบบ XML ที่เป็นอักขระแบบ UTF-8 มีการระบุการกระทำ SOAP และขนาดของเพย์โหลด
- บรรทัดที่ 95: การสิ้นสุดการทำงานของส่วนย่อย res โดยการส่งข้อมูลเพย์โหลด ok_body สำหรับการตอบกลับ ok ไปยังหน่วยเก็บข้อมูล IEEE1888
- บรรทัดที่ 100: การเรียกใช้ฟังก์ชัน listen ของส่วนย่อย server ที่ช่องทาง 80 สำหรับส่งการร้องขอและรอการตอบกลับ http ผ่านช่องทางการสื่อสารของตัวบริการ

```
1 // require 'echonet lite' npm module, 'http' module, 'xml2js' module
    , and 'moment' module.
2 var EL = require('echonet-lite');
3 var http = require('http');
4 var parseString = require('xml2js').parseString; // converting xml
    to json.
5 var moment = require('moment');
6
7 // execute 'trap_query.js' program to send trap query.
```

```
8  var trap_query = require('./trap_query');
9
10 // initialize ECHONET Lite object of controller.
11 var objList = ['05ff01'];
12
13 // initialize ECHONET Lite socket of self-node, which is called back
       by received package.
14 var elsocket = EL.initialize( objList, function( rinfo, els ) {
15   console.log('==============================');
16   console.log('Get ECHONET Lite data');
17   console.log('rinfo is ');
18   console.dir(rinfo);
19
20   console.log('----');
21   console.log('els is ');
22   console.dir(els);
23
24   console.log('----');
25   console.log( 'ECHONET Lite data array is ' );
26   console.log( EL.ELDATA2Array( els ) );
27
28   console.log('----');
29   console.log( 'Found facilities are ' );
30   console.dir( EL.facilities );
31 });
32
33 // initialize OK body message in xml string format to response from
       gateway to storage.
34 var ok_body = '<?xml version="1.0" encoding="UTF-8"?><soapenv:
       Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope
       /"><soapenv:Body><ns2:dataRS xmlns:ns2="http://soap.fiap.org/"><
       transport xmlns="http://gutp.jp/fiap/2009/11/"><header><OK/></
       header></transport></ns2:dataRS></soapenv:Body></soapenv:Envelope
       >';
35
36 // callback function to request or response from storage.
37 var server = http.createServer(function (req, res) {
38   var body = '';
39
40   // set request message to utf8 string format.
41   req.setEncoding('utf8');
42
43   // Readable streams emit 'data' events once a listener is added.
44   req.on('data', function (chunk) {
45     body += chunk;
46
```

```
47      // convert data format from XML to JSON before parsing.
48      parseString(body, function (err, result) {
49        jsonbody = JSON.stringify(result);
50      return jsonbody;
51      })
52      try {
53        var idc=[];
54        var timec=[];
55        var valuec=[];
56
57    /*----------------------parsing json: start----------------------*/
58        JSON.parse(jsonbody, function (key, value) {
59          if (key===’id’&& value[0]===’h’){
60            idc.push(value);
61          }
62          else if (key===’time’){
63            timec.push(value);
64          }
65          else if (key===’_’){
66            valuec.push(value);
67          }
68        });
69
70        // loop for collecting any last idc for each valuec.
71        var id=[];
72        for (var i=idc.length - valuec.length; i < idc.length; i++){
73          id.push(idc[i]);
74        }
75    /*----------------------parsing json: end----------------------*/
76
77        // the condition: if pir sensor can detect motion then air
              conditioner is turned on.
78        if (valuec[0]==’ON’){
79          // EL.sendOPC1 = function( ip, seoj, deoj, esv, epc, edt);
80          EL.sendOPC1( ’192.168.0.100’, [0x01,0x30,0x01], [0x0e,0xf0,0
                x01], 0x61, 0x80, 0x30);
81        }
82      }
83      catch (er) {
84        console.log(er);
85      }
86    });
87
88    // send response with 200 OK message to storage at the end.
89    req.on(’end’, function () {
90      res.writeHead(200, {
```

```
91        'Content-Type': 'text/xml charset=UTF-8',
92        'SOAPAction': 'http://soap.fiap.org/data',
93        'Content-Length': Buffer.byteLength(ok_body)
94      });
95      res.end(ok_body);
96    });
97  });
98
99  // server is running at port 80.
100 server.listen(80);
101 console.log('Server running at http://127.0.0.1:80/');
```

ค โปรแกรม node.js สำหรับการแจ้งเตือนการเปลี่ยนแปลงสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ด้วยวิธี NOTIFY และส่งข้อมูลสถานะการเปลี่ยนแปลงไปเก็บยังหน่วยเก็บข้อมูล IEEE1888 ด้วยโพรโทคอล WRITE (echonet_notify_write.js)

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 2-4: การประกาศตัวแปร EL http และ moment สำหรับอ้างอิงถึงมอดูล echonet-lite http และ moment ตามลำดับ
- บรรทัดที่ 7: การประกาศตัวแปร objList ชนิดแถวลำดับของอักขระที่อยู่ในรูปเลขฐาน 16 สำหรับใช้เป็นออบเจกต์ของตัวควบคุม (controller) ตามการสื่อสารมาตรฐาน ECHONET Lite
- บรรทัดที่ 10-69: การประกาศตัวแปร elsocket เพื่อเรียกใช้ฟังก์ชัน initialize ในมอดูล EL โดยมีอินพุตอาร์กิวเมนต์ 2 ตัว ประกอบด้วย ออบเจกต์ของตัวควบคุม objList และฟังก์ชันเรียกกลับที่มีตัวแปร rinfo และ els เป็นอินพุตอาร์กิวเมนต์ ฟังก์ชันเรียกกลับนี้จะถูกเรียกใช้เมื่อมีข้อมูล ECHONET Lite เข้ามา เพื่อแสดงข้อมูลที่ได้รับมาเหล่านั้นหรือนำข้อมูลเหล่านั้นไปใช้
- บรรทัดที่ 29: การประกาศและกำหนดตัวแปรชนิดแถวลำดับ temp เท่ากับฟังก์ชัน ELDATA2Array ในมอดูล EL โดยมีอินพุตอาร์กิวเมนต์ คือ ส่วนย่อย els
- บรรทัดที่ 30-37: การกำหนดเงื่อนไข ถ้าค่าของตัวแปรแถวลำดับ temp ตำแหน่งที่ 14 เท่ากับ 48 แล้วจึงกำหนดให้ตัวแปร status เท่ากับ 'ON' และถ้าค่าของตัวแปรแถวลำดับ temp ตำแหน่งที่ 14 เท่ากับ 49 แล้วจึงกำหนดให้ตัวแปร status เท่ากับ 'OFF'
- บรรทัดที่ 40: การประกาศตัวแปร write_body ชนิดอักขระ (string) เพื่อเป็นเพย์โหลดสำหรับบันทึกข้อมูลสถานะการเปิดปิดของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite จากตัวแปร status ตามโพรโทคอล WRITE โดยใช้โพรโทคอล SOAP ในการเข้าถึงอ็อบเจกต์ และรูปแบบข้อความ XML ภายในเพย์โหลดมีการระบุ point ID ที่ต้องการบันทึกข้อมูล
- บรรทัดที่ 43-53: การประกาศตัวแปร postRequest ชนิดอ็อบเจกต์เพื่อเป็นส่วนหัว (header) ของการร้องขอ http ภายในอ็อบเจกต์มีการระบุเลขที่อยู่ไอพีแม่ข่าย (host) คือ หน่วยเก็บข้อมูล IEEE1888 มีการระบุวิถี (path) ช่องทาง (port) วิธีการร้องขอ http แบบ post และยังมีการกำหนดอ็อบเจกต์ headers ซึ่งเป็นการกำหนดตามรูปแบบ XML ที่มีอักขระแบบ UTF-8
- บรรทัดที่ 57-62: การประกาศตัวแปร write_req เพื่อเป็นส่วนย่อย (element) สำหรับเรียกใช้ฟังก์ชันเรียกกลับ request ในอ็อบเจกต์ http โดยมีอินพุตอาร์กิวเมนต์ 2 ตัว คือ postRequest และฟังก์ชันที่มีอินพุตอาร์กิวเมนต์เป็นตัวแปร res สำหรับการรับข้อมูลตอบกลับจากหน่วยเก็บข้อมูล IEEE1888
- บรรทัดที่ 65: การเรียกใช้ฟังก์ชัน write ของส่วนย่อย write_req สำหรับส่งการร้องขอ http ด้วยข้อมูลเพย์โหลด write_body ไปยังหน่วยเก็บข้อมูล IEEE1888

- บรรทัดที่ 66: การเรียกใช้ฟังก์ชัน end ของส่วนย่อย write_req สำหรับการสิ้นสุดการร้องขอ http
- บรรทัดที่ 72: การส่งชุดคำสั่งเพื่อร้องขอการแจ้งเตือนการเปลี่ยนแปลงสถานะของอุปกรณ์เครื่องปรับอากาศมาตรฐาน ECHONET Lite ไปยังโนด ECHONET Lite ที่มีเลขที่อยู่ไอพี คือ 192.168.0.100 ด้วยการระบุค่า ESV เท่ากับ 0x63 และค่า EPC เท่ากับ 0x80

```
1  // require 'echonet lite' npm module, 'http' module, and 'moment'
       module.
2  var EL = require('echonet-lite');
3  var http = require('http');
4  var moment = require('moment');
5
6  // initialize ECHONET Lite object of controller.
7  var objList = ['05ff01'];
8
9  // initialize ECHONET Lite socket of self-node, which is called back
       by received package.
10 var elsocket = EL.initialize( objList, function( rinfo, els ) {
11   console.log('==============================');
12   console.log('Get ECHONET Lite data');
13   console.log('rinfo is ');
14   console.dir(rinfo);
15
16   console.log('----');
17   console.log('els is ');
18   console.dir(els);
19
20   console.log('----');
21   console.log( 'ECHONET Lite data array is ' );
22   console.log( EL.ELDATA2Array( els ) );
23
24   console.log('----');
25   console.log( 'Found facilities are ' );
26   console.dir( EL.facilities );
27
28   // the condition: if temp[14]=48 then status='ON', if temp[14]=49
         then status='OFF'.
29   var temp = EL.ELDATA2Array( els );
30   if (temp[14]!=undefined){
31     var status = '';
32     if (temp[14]==48){
33       status = 'ON';
34     }
35     else if (temp[14]==49){
36       status = 'OFF';
```

```
37      }
38
39      // initialize write body.
40      var write_body = '<?xml version="1.0" encoding="UTF-8"?><soapenv
            :Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
            envelope/"><soapenv:write_body><ns2:dataRQ xmlns:ns2="http://
            soap.fiap.org/"><transport xmlns="http://gutp.jp/fiap
            /2009/11/"><write_body><point id="http://chayanon.chula.ac.th
            /echonet/air-conditioner/status"><value time="'+moment().
            format()+'">'+status+'</value></point></write_body></
            transport></ns2:dataRQ></soapenv:write_body></soapenv:
            Envelope>';
41
42      // initialize header of http request.
43      var postRequest = {
44        host: "161.200.90.122",
45        path: "/axis2/services/FIAPStorage",
46        port: 80,
47        method: "POST",
48        headers: {
49          'Content-Type': 'text/xml charset=UTF-8',
50          'SOAPAction': 'http://soap.fiap.org/data',
51          'Content-Length': Buffer.byteLength(write_body),
52        }
53      };
54      var buffer = "";
55
56      // initialize http request for receiving data.
57      var write_req = http.request( postRequest, function ( res ) {
58        console.log( res.statusCode );
59        var buffer = "";
60        res.on( "data", function( data ) { buffer = buffer + data; } )
              ;
61        res.on( "end", function( data ) { console.log( buffer ); } );
62      });
63
64      // send http request.
65      write_req.write( write_body );
66      write_req.end();
67
68    } // end of condition if.
69  });
70
71  // EL.sendOPC1 = function( ip, seoj, deoj, esv, epc, edt);
72  EL.sendOPC1( '192.168.0.100', [0x0e,0xf0,0x01], [0x01,0x30,0x01], 0
        x63, 0x80, 0x00);
```

ง โปรแกรม node.js สำหรับส่งการร้องขอด้วยโพรโทคอล WRITE จากอินเตอร์เวิร์กกิงพร็อกซีเกตเวย์เพื่อเปลี่ยนแปลงข้อมูลในหน่วยเก็บข้อมูล CU-BEMS โดยการดึงข้อมูลที่ถูกจำลองจากไฟล์ข้อมูลที่กำหนด (write_dumpdata.js)

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 2-4: การประกาศตัวแปร argv สำหรับอ้างอิงถึงมอดูล optimist เพื่อใช้รับค่าอินพุตอาร์กิวเมนต์ delay และ id จากชุดคำสั่งบนหน้าจอเทอร์มินัล
- บรรทัดที่ 7-10: การประกาศตัวแปร moment uuid fs และ lazy สำหรับอ้างอิงถึงมอดูล moment node-uuid fs และ lazy ตามลำดับ
- บรรทัดที่ 13: การกำหนดตัวแปร export_write สำหรับเรียกใช้โปรแกรม export_write.js
- บรรทัดที่ 15-18: การกำหนดค่าต่าง ๆ ที่จำเป็นในโปรแกรม
- บรรทัดที่ 20-32: การกำหนดฟังก์ชัน main และ createAPP ตามลำดับ
- บรรทัดที่ 34-42: การกำหนดฟังก์ชัน sendLines สำหรับเลื่อนการอ่านข้อมูลทีละชุด โดยมีอินพุตอาร์กิวเมนต์ คือ containerId และ lines
- บรรทัดที่ 44-51: การกำหนดฟังก์ชัน buildContainer สำหรับเรียกใช้มอดูล lazy เพื่อแบ่งข้อมูลและประมวลผลข้อมูลทีละชุดเนื่องจากไม่รู้ขนาดของชุดข้อมูลทั้งหมด
- บรรทัดที่ 54-60: การกำหนดฟังก์ชัน buildJSON สำหรับแจกแจงข้อมูลอักขระให้อยู่ในรูปแบบอ็อบเจกต์ของ JSON และส่งค่าของอ็อบเจกต์นี้เพื่อนำไปใช้ต่อไป
- บรรทัดที่ 63-83: การกำหนดฟังก์ชัน readData สำหรับอ่านข้อมูลทีละไฟล์ในสารบบ (directory)
- บรรทัดที่ 86-95: การกำหนดฟังก์ชัน sendData สำหรับเรียกใช้โปรแกรม export_write.js ด้วยการกำหนดอินพุตอาร์กิวเมนต์ คือ data.point.setid และ fiap_element
- บรรทัดที่ 98: การเรียกใช้งานฟังก์ชัน main

```
1 // require 'optimist' module to export an object that contains the
    parsed command line arguments (delay, id).
2 var argv = require('optimist')
3     .usage('Usage: $0 --delay [num] --id [num]')
4     .argv;
5
6 // require 'moment', 'node-uuid', 'fs' and 'lazy' module.
7 var moment = require('moment');
8 var uuid = require('node-uuid');
9 var fs = require('fs');
```

```
10 var lazy = require('lazy');
11
12 // import export_write.js
13 var export_write = require('./export_write');
14
15 var data = [];
16 var registered = false;
17 var delay, dataID;
18 var dataDIR = __dirname + '/data'; // define string of directory.
19
20 function main() {
21   delay = parseFloat(argv.delay) || 0;
22   dataID = argv.id;
23   delay *= 1000;
24
25   // execute 'createAPP' function.
26   createAPP();
27 }
28
29 function createAPP() {
30   // execute 'readData' function.
31   readData();
32 };
33
34 function sendLines(containerId, lines) {
35   if (lines.length == 0)
36     return;
37
38   // shift data reading line by line.
39   sendData(containerId, lines.shift(), function() {
40     sendLines(containerId, lines);
41   });
42 }
43
44 function buildContainer(containerID) {
45   var i = 0;
46
47   // asynchronous function to treat a stream of event list.
48     new lazy(fs.createReadStream(dataDIR + "/" + containerID)).lines
          .map(buildJSON).join(function(lines) {
49     sendLines(containerID, lines)
50   });
51 }
52
53 // function to build JSON object by string parsing.
54 function buildJSON(line) {
```

```
55    var codes = line.toString().split(" ");
56    var id = codes[0];
57    var consumed = parseInt(codes[1], 16);
58    var json_object = { 'point':{'setid' : 'http://chayanon.chula.ac.
          th/test/echonet/write', 'id' : id},'data': { 'timestamp':
          moment().format(), 'consumed': consumed} };
59    return json_object;
60  };
61
62  // read data and build json objects.
63  function readData() {
64    if (dataID) {
65      buildContainer(dataID);
66      return;
67    }
68    console.log("reading all data from " + dataDIR);
69    'use strict';
70
71    // define number of files in directory.
72    var numFiles = fs.readdirSync(dataDIR).length;
73    console.log(numFiles + " files found");
74    var x = 0;
75
76    // read file for each in directory.
77    fs.readdirSync(dataDIR).forEach(function (filename) {
78      if (++x > 1)
79      return;
80      console.log("reading " + x + " - "  +  dataDIR + "/" + filename)
            ;
81      buildContainer(filename);
82    });
83  }
84
85  // send data to 'export_write.js' with setid and fiap_element
86  function sendData(containerID, data, onSuccess) {
87    fiap_element = [[data.point.setid + data.point.id, data.data.
          consumed, moment().format()]];
88    export_write.write(data.point.setid, fiap_element);
89
90    // in case: there is delay argument.
91      if (delay) {
92      setTimeout(onSuccess, delay);
93      } else
94      onSuccess();
95  };
96
```

```
97 // execute 'main' function.
98 main();
```

จ โปรแกรม node.js สำหรับเรียกใช้ฟังก์ชัน 'export_write' เพื่อส่งการร้องขอ WRITE ไปยังหน่วยเก็บข้อมูล CU-BEMS ด้วยการระบุค่า point ID และอ็อบเจกต์ของ JSON (export_write.js)

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 2-3: การประกาศตัวแปร fs และ http สำหรับอ้างอิงถึงมอดูล fs และ http
- บรรทัดที่ 6-42: การเรียกใช้ฟังก์ชัน export_write สำหรับส่งการร้อง WRITE ตามมาตรฐาน IEEE1888 เพื่อเปลี่ยนแปลงข้อมูลในหน่วยเก็บข้อมูล CU-BEMS ด้วยการระบุค่า point ID และค่าของข้อมูลที่ถูกจำลอง
- บรรทัดที่ 9-12: การผสานตัวแปรชนิดอักขระสำหรับเก็บค่าอักขระของทุก point ID
- บรรทัดที่ 15: การประกาศตัวแปร body ชนิดอักขระ (string) เพื่อเป็นเพย์โหลดสำหรับเปลี่ยนแปลงข้อมูลจากข้อมูลที่ถูกจำลองในหน่วยเก็บข้อมูล CU-BEMS ตามโพรโทคอล WRITE โดยใช้โพรโทคอล SOAP ในการเข้าถึงอ็อบเจกต์ และรูปแบบข้อความ XML ภายในเพย์โหลดมีการระบุ point ID ที่ต้องการเปลี่ยนแปลงข้อมูล
- บรรทัดที่ 18-28: การประกาศตัวแปร postRequest ชนิดอ็อบเจกต์เพื่อเป็นส่วนหัว (header) ของการร้องขอ http ภายในอ็อบเจกต์มีการระบุเลขที่อยู่ไอพีแม่ข่าย (host) คือ หน่วยเก็บข้อมูล IEEE1888 มีการระบุวิถี (path) ช่องทาง (port) วิธีการร้องขอ http แบบ post และยังมีการกำหนดอ็อบเจกต์ headers ซึ่งเป็นการกำหนดตามรูปแบบ XML ที่มีอักขระแบบ UTF-8
- บรรทัดที่ 32-37: การประกาศตัวแปร req เพื่อเป็นส่วนย่อย (element) สำหรับเรียกใช้ฟังก์ชันเรียกกลับ request ในอ็อบเจกต์ http โดยมีอินพุตอาร์กิวเมนต์ 2 ตัว คือ postRequest และฟังก์ชันที่มีอินพุตอาร์กิวเมนต์เป็นตัวแปร res สำหรับการรับข้อมูลตอบกลับจากหน่วยเก็บข้อมูล IEEE1888
- บรรทัดที่ 40: การเรียกใช้ฟังก์ชัน write ของส่วนย่อย req สำหรับส่งการร้องขอ http ด้วยข้อมูลเพย์โหลด body ไปยังหน่วยเก็บข้อมูล IEEE1888
- บรรทัดที่ 41: การเรียกใช้ฟังก์ชัน end ของส่วนย่อย req สำหรับการสิ้นสุดการร้องขอ http

```
1  // require the 'fs' and 'http' module.
2  var fs = require('fs');
3  var http = require('http');
4
5  // execute 'export_write' function.
6  exports.write = function (pointset, fiap){
7
8    // merge any point IDs in case that there are more than 1 point ID
       .
9    var point = [];
10   for (var i = 0; i < fiap.length; i++) {
```

```
11      point = point + '<point id='+'"'+fiap[i][0]+'"'+'><value time="'
            +fiap[i][2]+'">'+fiap[i][1]+'</value></point>'
12    }
13
14    // initialize write body with point ID.
15    var body = '<?xml version="1.0" encoding="UTF-8"?><soapenv:
          Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
          envelope/"><soapenv:Body><ns2:dataRQ xmlns:ns2="http://soap.
          fiap.org/"><transport xmlns="http://gutp.jp/fiap/2009/11/"><
          body><pointSet id="'+pointset+'">'+point+'</pointSet></body></
          transport></ns2:dataRQ></soapenv:Body></soapenv:Envelope>';
16
17    // initialize header of http request.
18    var postRequest = {
19      host: "161.200.90.122",
20      path: "/axis2/services/FIAPStorage",
21      port: 80,
22      method: "POST",
23      headers: {
24        'Content-Type': 'text/xml charset=UTF-8',
25        'SOAPAction': 'http://soap.fiap.org/data',
26        'Content-Length': Buffer.byteLength(body),
27      }
28    };
29    var buffer = "";
30
31    // initialize http request for receiving data.
32    var req = http.request( postRequest, function ( res ) {
33      console.log( res.statusCode );
34      var buffer = "";
35      res.on( "data", function( data ) { buffer = buffer + data; } );
36      res.on( "end", function( data ) { console.log( buffer ); } );
37    });
38
39    // send http request.
40    req.write( body );
41    req.end();
42  };
```

ประวัติผู้เขียนวิทยานิพนธ์

นายชญานนท์ แสงอำไพ เกิดเมื่อวันอาทิตย์ที่ 21 กรกฎาคม พ.ศ. 2534 จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิตจากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง คณะวิศวกรรมศาสตร์ สาขาโทรคมนาคม ปีการศึกษา 2556 และได้ศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ที่จุฬาลงกรณ์มหาวิทยาลัย คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมไฟฟ้า กลุ่มวิจัยโครงข่ายโทรคมนาคมและสารสนเทศ ปีการศึกษา 2557

บทความทางวิชาการจากวิทยานิพนธ์

[1] Sangumpai, C., and Aswakul, C. Development of real-time interworking between IEEE1888 and ECHONET Lite standards for building energy management system. The 39th Electrical Engineering Conference (EECON-39), Phetchaburi City, Thailand, 2016.

[2] Sangumpai, C., and Aswakul, C. Development of real-time interworking between IEEE1888 and ECHONET Lite standards for building energy management system. Acceptance Recommendation Subject to A Minor Revision in Engineering Journal (EJ).