การพัฒนาระบบการทำงานร่วมกันแบบเวลาจริงระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M สำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคาร

นายณภัทร โกศลวรวัฒนกุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2558

DEVELOPMENT OF REAL-TIME INTERWORKING SYSTEM BETWEEN IEEE1888 AND ETSI M2M STANDARDS FOR BUILDING ENERGY MANAGEMENT SYSTEM

MR. NAPAT KOSOLWORRAWATTANAKUL

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 2015

หัวข้อวิทยานิพนธ์	การพัฒนาระบบการทำงานร่วมกันแบบเวลาจริง ระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M สำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคาร
โดย	นายณภัทร โกศลวรวัฒนกุล
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.เชาวน์ดิศ อัศวกุล

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร. บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.แนบบุญ หุนเจริญ)

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

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

..... กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.กิตติพันธุ์ เตชะกิตติโรจน์)

ณภัทร โกศลวรวัฒนกุล : การพัฒนาระบบการทำงานร่วมกันแบบเวลาจริงระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M สำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคาร (DEVELOPMENT OF REAL-TIME INTERWORKING SYSTEM BETWEEN IEEE1888 AND ETSI M2M STANDARDS FOR BUILDING ENERGY MANAGEMENT SYSTEM) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ. ดร. เชาวน์ดิศ อัศวกุล, 90 หน้า.

วิทยานิพนธ์ฉบับนี้เสนอระบบการทำงานร่วมกันแบบเวลาจริงระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M ในระบบจัดการพลังงานไฟฟ้าในอาคาร ระบบนี้นำเสนอการประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M ในเวลาจริงเพื่อทดสอบความสามารถการทำงานร่วมกันของอุปกรณ์ที่สื่อสารตามมาตรฐานที่ต่างกัน เช่น ตัวรับรู้ และตัวกระตุ้น เอ็นไอพี (เน็ทเวิร์คอินเตอร์เวิร์กกิงพร็อกซี) ได้รับการพัฒนาในการประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M ในเวลาจริง เริ่มจากเอ็นไอพีส่งการร้องขอแทรปโดยโพรโทคอล TRAP ไปยังหน่วยเก็บข้อมูลของ IEEE1888 และส่งสารติดตามโดยวิธี CREATE ไปยังคลังเก็บข้อมูลของ ETSI M2M เมื่อมีการเปลี่ยนแปลงใด ๆ ในค่าข้อมูลหน่วยเก็บข้อมูลของ IEEE1888 จะส่งข้อมูลตอบกลับ และคลังเก็บข้อมูลของ ETSI M2M จะส่งสาร NOTIFY มายังเอ็นไอพี หลังจากนั้นเอ็นไอพีส่งค่าข้อมูลไปยังอีกฐานข้อมูลโดยโพรโทคอล WRITE สำหรับมาตรฐาน IEEE1888 และวิธี CREATE สำหรับมาตรฐาน ETSI M2M ในส่วนของโครงการ CU-BEMS ที่มีโนดตัวรับรู้ทั้งหมด 688 โนด วิทยานิพนธ์ฉบับนี้ประมาณค่าสูงสุดของปริมาณงานเฉลี่ยรวมในการประสานค่าตัวรับรู้ทั้งหมดได้เป็น 0.36 และ 0.41 เมกะบิตต่อวินาที สำหรับกรณีของการประสานข้อมูลแบบคาบเวลาหนึ่งนาที และแบบเวลาจริง ตามลำดับ นอกจากนี้เกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M ได้รับการพัฒนาเพื่อเชื่อมต่อกับตัวรับรู้ และตัวกระตุ้น ผลการทดสอบถ่ายโอนข้อมูลระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M โดยข้อมูลเลียนแบบพบว่าระบบสามารถรองรับโนดตัวรับรู้ได้สูงที่สุดเป็น 500 โนดด้วยเวลาประวิงเฉลี่ยเท่ากับ 305 มิลลิวินาทีสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M และ 115 โนดด้วยเวลาประวิงเฉลี่ยเวลาประวิงเฉลี่ยเท่ากับ 3,561 มิลลิวินาทีสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 การทดสอบสุดท้ายใช้ตัวรับรู้จริงในโครงการ CU-BEMS ที่ติดตั้ง ณ ห้องปฏิบัติการวิจัยระบบโทรคมนาคมจำนวนตัวรับรู้ 28 ตัว มีโนดตัวรับรู้รวมทั้งหมด 112 โนด ซึ่งบริเวณดังกล่าวมีความหนาแน่นของตัวรับรู้ที่ติดต่อกับเกตเวย์หนึ่งตัวสูงสุดในโครงการ CU-BEMS พบว่าเวลาประวิงเฉลี่ยของการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 เท่ากับ 2,306 มิลลิวินาที ซึ่งสูงกว่าเวลาประวิงเฉลี่ยที่ 114 มิลลิวินาทีของการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M จากระบบดังกล่าวในวิทยานิพนธ์นี้พบว่าคอขวดของการถ่ายโอนข้อมูลอยู่ที่โครงข่ายของ ETSI M2M และเกตเวย์ของ IEEE1888 ที่ทดสอบการดำเนินการบนคอมพิวเตอร์ส่วนบุคคล และแพลตฟอร์ม Raspberry Pi B+ ตามลำดับ โดยจำนวนโนดตัวรับรู้สูงสุดที่ระบบรองรับได้นั้นขึ้นกับฮาร์ดแวร์ของส่วนประกอบในระบบ

ภาควิชา วิศวกรรมไฟฟ้า **ลายมือชื่อนิสิต**.....
สาขาวิชา วิศวกรรมไฟฟ้า **ลายมือชื่อ อ.ที่ปรึกษาหลัก**.....
ปีการศึกษา ....2558.....

5670181321 : MAJOR ELECTRICAL ENGINEERING
KEYWORDS: INTERWORKING/ IEEE1888/ ETSI M2M/ BUILDING ENERGY MANAGEMENT.

NAPAT KOSOLWORRAWATTANAKUL : DEVELOPMENT OF REAL-TIME INTERWORKING SYSTEM BETWEEN IEEE1888 AND ETSI M2M STANDARDS FOR BUILDING ENERGY MANAGEMENT SYSTEM. ADVISOR: ASSOC. PROF. CHAODIT ASWAKUL, Ph.D., 90 pp.

This thesis presents a real-time interworking system between IEEE1888 and ETSI M2M standards in building energy management system. This system proposes data synchronization between IEEE1888 storage and ETSI M2M repository in real-time to test the interoperability of devices communicating with the different standards, namely, sensors and actuators. NIP (network interworking proxy) has been developed to synchronize data between IEEE1888 storage and ETSI M2M repository in real-time. NIP sends trap queries by TRAP protocol to an IEEE1888 storage and sends a subscription message by CREATE method to an ETSI M2M repository. Upon any change in data values, the IEEE1888 storage then returns the callback data and the ETSI M2M repository returns NOTIFY messages to NIP. After that, NIP further relays the data values to the other database by IEEE1888 WRITE protocol and ETSI M2M CREATE method. Based on the CU-BEMS project with 688 sensor nodes, this thesis has estimated the total average throughput involved in synchronizing all sensor values to be at most 0.36 and 0.41 Mbits/sec for the case with updating period of one minute and for the real-time data synchronization case, respectively. Moreover, the IEEE1888 and ETSI M2M gateways have been developed to connect with sensors and actuators. For the data transfer between the IEEE1888 and ETSI M2M gateways by emulated data, the results have shown that the system can support up to 500 sensor nodes with the average delay of 305 milliseconds for the data transfer from the IEEE1888 gateway to ETSI M2M gateway and 115 nodes with the average delay of 3,561 milliseconds for the data transfer from the ETSI M2M gateway to the IEEE1888 gateway. The last experiment is based on the actual CU-BEMS sensors installed at the Telecommunication System Research Laboratory. There are 28 sensors with 112 sensor nodes in the laboratory, which is the area with the highest sensor density connected to one gateway in the CU-BEMS project. The results have shown that the average delay of data transfer from the ETSI M2M gateway to the IEEE1888 gateway is 2,306 milliseconds, which is higher than the average delay of 114 milliseconds from IEEE1888 gateway to ETSI M2M gateway. Based on the current implementation in this thesis, the performance bottleneck of data transfer is found at the ETSI M2M network and the IEEE1888 gateway running with the personal computer and the Raspberry Pi B+ platform, respectively. The maximum number of sensor nodes that can be supported in the system depends on the hardware components of the system.

Department : Electrical Engineering . **Student's Signature** . . . . . .
Field of Study : Electrical Engineering . **Advisor's Signature** . . . . . .
Academic Year : . . . . . 2015 . . . . . .

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ด้วยความช่วยเหลือจากอาจารย์ที่ปรึกษาวิทยานิพนธ์ รองศาสตราจารย์ ดร.เชาวน์ดิศ อัศวกุล ที่ช่วยถ่ายทอดความรู้ทั้งทางด้านวิจัยและด้านวิชาการ ให้คำปรึกษาและคำแนะนำ รวมถึงช่วยตรวจทานวิทยานิพนธ์ฉบับนี้อย่างดีเสมอมา ผู้วิจัยจึงขอกราบขอบพระคุณมา ณ ที่นี้ และขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.แนบบุญ หุนเจริญ ประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.ชัยเชษฐ์ สายวิจิตร และผู้ช่วยศาสตราจารย์ ดร.กิตติพันธุ์ เตชะกิตติโรจน์ กรรมการสอบวิทยานิพนธ์ ที่สละเวลาตรวจสอบและให้คำแนะนำสำหรับวิทยานิพนธ์ฉบับนี้เพื่อความสมบูรณ์ยิ่งขึ้น

งานวิจัยชิ้นนี้ได้รับทุนสนับสนุนจากโครงการขับเคลื่อนการวิจัย กองทุนรัชดาภิเษกสมโภช (Special Task Force for Activating Research, STAR) ภายใต้กลุ่มวิจัยโครงข่ายไร้สาย และอินเทอร์เน็ตอนาคต (Wireless Network and Future Internet Research Group, WI-FUN) จุฬาลงกรณ์มหาวิทยาลัย

ขอขอบคุณกลุ่มวิจัยโครงข่ายไร้สายและอินเทอร์เน็ตอนาคตดูแลโดย รองศาสตราจารย์ ดร. เชาวน์ดิศ อัศวกุล และผู้ช่วยศาสตราจารย์ ดร. ชัยเชษฐ์ สายวิจิตร ที่จัดกิจกรรมเพื่อส่งเสริมการเรียนรู้ การทำงานและทักษะเพื่อเป็นผู้วิจัยที่มีประสิทธิภาพที่ดียิ่งขึ้น

ขอขอบคุณโครงการวิจัย และพัฒนาเทคโนโลยีระบบโครงข่ายไฟฟ้าอัจฉริยะเพื่อบริหารจัดการการใช้พลังงานไฟฟ้าของอาคาร จุฬาลงกรณ์มหาวิทยาลัย (CU-BEMS) และโครงการมหาวิทยาลัยสำหรับอินเทอร์เน็ตในอนาคต (Universities for Future Internet, UNIFI) ที่ทำให้ผู้วิจัยมีองค์ความรู้ และได้รับคำแนะนำจากคณาจารย์และทีมงานเป็นอย่างดียิ่งซึ่งเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์ฉบับนี้เกิดขึ้นได้

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

สุดท้ายนี้ขอขอบคุณครอบครัวของผู้วิจัย ซึ่งได้ให้การสนับสนุนและเป็นกำลังใจให้แก่ผู้วิจัยเสมอมาจนสำเร็จการศึกษา

สารบัญ

บทที่ หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

หน้า

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันการพัฒนาระบบฝังตัว (embedded system) และเทคโนโลยีการสื่อสารสำหรับการเชื่อมโยงสิ่งของต่าง ๆ เข้ากับโครงข่ายอินเทอร์เน็ตที่เรียกว่า อินเทอร์เน็ตของสรรพสิ่ง (internet of things, IoT) [1] ได้รับความสนใจมากจากผู้คนทั่วโลก โดยทางบริษัทไอทียักษ์ใหญ่ ซิสโก้ (cisco) คาดการณ์ว่าภายในปี ค.ศ. 2020 สิ่งของต่าง ๆ จะเชื่อมต่อกันผ่านอินเทอร์เน็ตจำนวนสูงถึง 50,000 ล้านอุปกรณ์ [2] โดยอาศัยการสื่อสารระหว่างอุปกรณ์ หรือ ที่เรียกว่า การสื่อสารแบบแมชชีน-ทู-แมชชีน (machine-to-machine communication, M2M) [3] คือการสื่อสารระหว่างอุปกรณ์ผ่านทางเทคโนโลยีทั้งมีสาย และไร้สาย เช่น xDSL, cable, ZigBee, bluetooth, 6LoWPAN, WiFi, WiMax, LTE เป็นต้น

การสื่อสารแบบแมชชีน-ทู-แมชชีนถูกนำมาใช้อย่างกว้างขวางสำหรับระบบจัดการพลังงานในโครงข่ายไฟฟ้าอัจฉริยะเพื่อตอบสนองการใช้พลังงานอย่างมีประสิทธิภาพ ในปี ค.ศ. 2011 สถาบันวิชาชีพวิศวกรไฟฟ้าและอิเล็กทรอนิกส์ (institute of electrical and electronics engineers, IEEE) ได้ออกแบบมาตรฐานเปิดสำหรับระบบจัดการพลังงานโดยอาศัยการสื่อสารระหว่างอุปกรณ์ที่เรียกว่า IEEE1888 หรือ โพรโทคอลโครงข่ายการควบคุมชุมชนสีเขียวอย่างแพร่หลาย (ubiquitous green community control network, UGCCNet) [4] ซึ่งสามารถเฝ้าสังเกตการใช้พลังงานและสภาพแวดล้อมโดยสื่อสารผ่านโครงข่ายทีซีพี/ไอพี (TCP/IP) กับอุปกรณ์ระยะไกลเพื่อรับข้อมูล และควบคุม โดยมาตรฐาน IEEE1888 ถูกนำไปใช้ในระบบจัดการพลังงานในโครงการมหาวิทยาลัยโตเกียวสีเขียว (green university of tokyo project, GUTP) ประเทศญี่ปุ่น [5] มีวัตถุประสงค์เพื่อลดการใช้พลังงานไฟฟ้าจากการเฝ้าสังเกตการใช้พลังงานไฟฟ้าภายในอาคาร และเปรียบเทียบการใช้พลังงานไฟฟ้าในแต่ละวันทำให้ผู้ใช้ตระหนักถึงการใช้พลังงานไฟฟ้าทำให้มีการใช้พลังงานไฟฟ้าที่ลดลง นอกจากนี้ได้มีการติดตั้งระบบควบคุมหลอดไฟ และเครื่องปรับอากาศอัตโนมัติตามระยะเวลาการใช้งานในห้องเรียน ทำให้มีการใช้พลังงานไฟฟ้าอย่างมีประสิทธิภาพ จากแนวคิดโครงการมหาวิทยาลัยโตเกียวสีเขียว [5] ทางภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ได้ดำเนินโครงการวิจัยและพัฒนาเทคโนโลยีระบบโครงข่ายไฟฟ้าอัจฉริยะเพื่อบริหารจัดการการใช้พลังงานไฟฟ้าของอาคาร (Chulalongkorn university building energy management system, CU-BEMS) [6]

โครงการ CU-BEMS มีจุดประสงค์เพื่อเฝ้าสังเกตการใช้พลังงาน และสภาพแวดล้อมภายในอาคาร ซึ่งประกอบด้วย มาตรวัดอัจฉริยะ [7] ตัวรับรู้วัดสภาพแวดล้อมแบบไร้สาย [8] [9] [10] จอแสดงผลภายในอาคาร [11] [12] เป็นต้น โดยโครงการ CU-BEMS ได้นำสถาปัตยกรรม และโพรโทคอลการสื่อสารตามมาตรฐาน IEEE1888 มาใช้ประกอบด้วย เกตเวย์ (GW) หน่วยเก็บข้อมูล (storage) และโปรแกรมประยุกต์ (APP) ซึ่งไม่ได้นำ รีจิสทรี (registry) ตามมาตรฐาน IEEE1888 มาใช้ ตัวรับรู้ (sensor) วัดสภาพแวดล้อมในโครงการ CU-BEMS มีด้วยกัน 4 ชนิด คือ ตัวรับรู้อุณหภูมิ ตัวรับรู้ความชื้นสัมพัทธ์ ตัวรับรู้แสง และตัวรับรู้การเคลื่อนไหวของคน เชื่อมต่อด้วยโครงข่ายเทคโนโลยีไร้สาย ZigBee ส่งข้อมูลตัวรับรู้มายังเกตเวย์ หลังจากนั้นเกตเวย์ใช้โพรโทคอล WRITE ในการส่งข้อมูลตัวรับรู้ไปยังหน่วยเก็บข้อมูล และโปรแกรมประยุกต์ใช้โพรโทคอล FETCH เรียกข้อมูลจาก

หน่วยเก็บข้อมูล

นอกเหนือจากมาตรฐาน IEEE1888 ในปี ค.ศ. 2011 สถาบันมาตรฐานโทรคมนาคมยุโรป (european telecommunications standards institute, ETSI) [13] ได้ออกมาตรฐานที่เรียกว่า ETSI M2M เพื่อให้การสื่อสารระหว่างอุปกรณ์เป็นแพลตฟอร์มมาตรฐานที่มีระบบประยุกต์ สภาพแวดล้อม และองค์ประกอบของโครงข่ายร่วมกัน ต่อมาในปี ค.ศ. 2012 สถาบันมาตรฐานโทรคมนาคมยุโรปได้ร่วมมือกับองค์กรทางด้านไอซีทียักษ์ใหญ่ทั่วโลกในการสร้างมาตรฐานการสื่อสารระหว่างอุปกรณ์โดยพัฒนาต่อจากมาตรฐาน ETSI M2M ภายใต้ชื่อว่า oneM2M เพื่อเป็นมาตรฐานการสื่อสารระหว่างอุปกรณ์ และอินเทอร์เน็ตของสรรพสิ่งของโลก [14]

เนื่องจากทางจุฬาลงกรณ์มหาวิทยาลัยเป็นส่วนหนึ่งของโครงการมหาวิทยาลัยสำหรับอินเทอร์เน็ตในอนาคต (universities for future internet, UNIFI) [15] ได้รับความร่วมมือจากมหาวิทยาลัยเทคนิคแห่งเบอร์ลิน (technical university of Berlin, TUB) [16] และสถาบัน Fraunhofer FOKUS [17] ในการสนับสนุนทางด้านเทคนิค และวิชาการสำหรับแพลตฟอร์มที่เรียกว่า OpenMTC [18] โดยแพลตฟอร์ม OpenMTC เป็นมิดเดิลแวร์แพลตฟอร์มที่สอดคล้องตามมาตรฐาน ETSI M2M เพื่อการวิจัย และการพัฒนาการสื่อสารระหว่างอุปกรณ์รวมถึงอินเทอร์เน็ตของสรรพสิ่งสำหรับโครงข่ายไฟฟ้าอัจฉริยะ อาคารอัจฉริยะ เป็นต้น งานวิจัยที่ผ่านมา [19] และ [20] กล่าวถึงสถาปัตยกรรม ฟังก์ชันการทำงาน และการสาธิตการนำแพลตฟอร์ม OpenMTC ควบคุมอุปกรณ์เครื่องใช้ไฟฟ้าเช่น พัดลม โคมไฟ เป็นต้น และรับข้อมูลจากตัวรับรู้สำหรับวัดอุณหภูมิ ความชื้น และความสว่าง งานวิจัยดังกล่าวไม่ได้คำนึงถึงการทำงานร่วมกับแพลตฟอร์มมาตรฐานอื่น ซึ่งมาตรฐาน ETSI M2M มีความสามารถในการทำงานร่วมกับมาตรฐานอื่นโดยอาศัยอินเตอร์เวิร์กกิงพร็อกซี (interworking proxy)

การพัฒนาการทำงานร่วมกันระหว่างโครงการ CU-BEMS และแพลตฟอร์ม OpenMTC จะช่วยเพิ่มประสิทธิภาพ และความยืดหยุ่นในการทำงานของโครงการ CU-BEMS ซึ่งสามารถพิจารณาได้ดังนี้ รูปแบบของข้อมูลในมาตรฐาน IEEE1888 ใช้ XML เมื่อพิจารณาในส่วนของมาตรฐาน ETSI M2M สามารถรองรับการใช้รูปแบบของข้อมูลได้ทั้ง XML, JSON, EXI และ fast infoset [21] ซึ่งมีความหลากหลายของรูปแบบข้อมูลมากกว่ามาตรฐาน IEEE1888 นอกจากนี้โครงสร้างสำหรับการจัดการชื่อของข้อมูลตามมาตรฐาน IEEE1888 ใช้ point id โดยรูปแบบของ point id เป็นยูอาร์ไอที่มีค่ากำหนดตามตำแหน่งของตัวรับรู้ และตัวกระตุ้น [4] ในส่วนของมาตรฐาน ETSI M2M สามารถระบุตำแหน่งได้หลายรูปแบบ เช่น group, container, searchstring, locationcontainer เป็นต้น [23] โดยสามารถจัดเก็บข้อมูลได้หลายรูปแบบเพื่อง่ายต่อการเรียกข้อมูล สุดท้ายโพรโทคอลในการสื่อสารหลักของมาตรฐาน IEEE1888 ประกอบด้วย โพรโทคอล WRITE, FETCH และ TRAP ในส่วนของวิธีสื่อสารของมาตรฐาน ETSI M2M ประกอบด้วยวิธี CREATE, RETRIEVE, UPDATE และ DELETE นอกจากนี้ยังมีวิธี NOTIFY และ EXECUTE [21] เห็นได้ว่ามาตรฐาน ETSI M2M มีรูปแบบในการสื่อสารมากกว่ามาตรฐาน IEEE1888 โดยการพัฒนาการทำงานร่วมกันของโครงการ CU-BEMS และแพลตฟอร์ม OpenMTC เป็นการริเริ่มการทำงานร่วมกันระหว่างมาตรฐานการสื่อสารระหว่างอุปกรณ์ที่จะมีบทบาทความสำคัญในอนาคต ด้วยเหตุนี้จึงเกิดแนวคิดในการพัฒนาเอ็นไอพี (network interworking proxy, NIP) ในมาตรฐาน ETSI M2M สำหรับการทำงานร่วมกับมาตรฐาน IEEE1888

งานวิจัย [21] และ [22] ได้นำเสนอการพัฒนาโพรโทคอลของเกตเวย์สำหรับการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M เพื่อการประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูล (repository) ของ ETSI M2M ตัวกลางใน

การทำงานร่วมกันนี้เรียกว่า พร็อกซีเกตเวย์ (proxy gateway) โดยเกตเวย์จำลองร้องขอข้อมูล และรับข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M ด้วยโพรโทคอล FETCH และวิธี RETRIEVE ตามลำดับ หลังจากนั้นเกตเวย์จำลองส่งข้อมูลด้วยวิธี CREATE ไปเก็บในคลังเก็บข้อมูลของ ETSI M2M และโพรโทคอล WRITE ไปเก็บในหน่วยเก็บข้อมูลของ IEEE1888 งานวิจัยดังกล่าวนี้ไม่ได้เป็นการประสานข้อมูลแบบเวลาจริงเพราะเกตเวย์จำลองต้องร้องขอข้อมูลเมื่อต้องการประสานข้อมูลทุกครั้งจึงมีข้อจำกัดในกรณีการเฝ้าสังเกตข้อมูลตัวรับรู้แบบเวลาจริง การแจ้งเตือนฉุกเฉิน และการควบคุมอุปกรณ์อัตโนมัติแบบเวลาจริง

วิทยานิพนธ์ฉบับนี้จึงนำเสนอการพัฒนาเอ็นไอพีในการประสานข้อมูลแบบเวลาจริงระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M ในส่วนของการประสานข้อมูลจากมาตรฐาน IEEE1888 ไปยังมาตรฐาน ETSI M2M เอ็นไอพีมีการใช้โพรโทคอล TRAP แทนการใช้โพรโทคอล FETCH สำหรับการตอบกลับข้อมูลแบบไม่ประสานเวลา (asynchronous) จากหน่วยเก็บข้อมูลของ IEEE1888 โดยการจดทะเบียนเงื่อนไขในแจ้งเตือนข้อมูลที่หน่วยเก็บข้อมูลของ IEEE1888 ซึ่งโพรโทคอล TRAP ยังไม่ได้ถูกนำมาใช้ในโครงการ CU-BEMS หลังจากนั้นเอ็นไอพีส่งข้อมูลไปเก็บยังคลังเก็บข้อมูลของ ETSI M2M โดยใช้วิธี CREATE ในส่วนของการประสานข้อมูลจากมาตรฐาน ETSI M2M ไปยังมาตรฐาน IEEE1888 เอ็นไอพีรับข้อมูลด้วยวิธี NOTIFY แทนวิธี RETRIEVE ในการตอบกลับข้อมูลโดยส่ง subscription สำหรับการจดทะเบียนการติดตามข้อมูลด้วยวิธี CREATE ที่คลังเก็บข้อมูลของ ETSI M2M เมื่อคลังเก็บข้อมูลของ ETSI M2M ได้รับข้อมูลจะตอบกลับข้อมูลไปยังเอ็นไอพีทันที หลังจากนั้นเอ็นไอพีมีการใช้โพรโทคอล WRITE ส่งข้อมูลไปเก็บยังหน่วยเก็บข้อมูลของ IEEE1888 นอกจากนี้วิทยานิพนธ์ฉบับนี้มีการพัฒนาเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M เพื่อให้ตัวรับรู้ และตัวกระตุ้นตามมาตรฐานที่ต่างกันสามารถทำงานร่วมกันได้แบบเวลาจริง

1.2 วัตถุประสงค์ของงานวิทยานิพนธ์

1. พัฒนาเอ็นไอพีสำหรับประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M แบบเวลาจริง
2. พัฒนาเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M เพื่อให้ตัวรับรู้ และตัวกระตุ้นระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M สามารถทำงานร่วมกันได้แบบเวลาจริง

1.3 ขอบเขตวิทยานิพนธ์

1. เอ็นไอพีสามารถประสานข้อมูลแบบเวลาจริงระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M โดยใช้โพรโทคอล TRAP และวิธี NOTIFY ในการแจ้งเตือนข้อมูลตัวรับรู้จากหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M หลังจากนั้นใช้วิธี CREATE และโพรโทคอล WRITE ในการส่งข้อมูลตัวรับรู้ไปเก็บยังคลังเก็บข้อมูลของ ETSI M2M และหน่วยเก็บข้อมูลของ IEEE1888 ตามลำดับ
2. พัฒนาเกตเวย์ของ IEEE1888 บนบอร์ดสำเร็จรูป Raspberry Pi B+ โดยใช้โพรโทคอล

WRITE ในการส่งข้อมูลตัวรับรู้ไปเก็บยังหน่วยเก็บข้อมูลของ IEEE1888 และใช้โพรโทคอล TRAP ในการแจ้งเตือนข้อมูลตัวรับรู้จากหน่วยเก็บข้อมูลของ IEEE1888

3. พัฒนาเกตเวย์ของ ETSI M2M บนบอร์ดสำเร็จรูป Raspberry Pi B+ โดยใช้วิธี CREATE ในการส่งข้อมูลตัวรับรู้ไปเก็บยังคลังเก็บข้อมูลของ ETSI M2M และใช้วิธี NOTIFY ในการแจ้งเตือนข้อมูลตัวรับรู้จากคลังเก็บข้อมูลของ ETSI M2M
4. ทดสอบ และประเมินการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M

1.4 ขั้นตอนและวิธีการดำเนินงาน

1. ศึกษามาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M
2. พัฒนาเอ็นไอพีประสานข้อมูลตัวรับรู้ระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M แบบเวลาจริง
3. พัฒนาเกตเวย์ของ IEEE1888 เกตเวย์ของ ETSI M2M และตัวกระตุ้นไร้สาย
4. ทดสอบ วิเคราะห์ และสรุปผลจากการทำงานของระบบที่พัฒนาขึ้น
5. เผยแพร่ผลการวิจัยลงในบทความ และจัดทำเล่มวิทยานิพนธ์ฉบับสมบูรณ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. หน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M สามารถประสานข้อมูลแบบเวลาจริง
2. ตัวรับรู้ และตัวกระตุ้นที่ทำงานตามมาตรฐานที่ต่างกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M สามารถทำงานร่วมกันได้แบบเวลาจริง

1.6 ประมวลวิทยานิพนธ์

บทที่ 1 บทนำ: กล่าวถึงความเป็นมา และความสำคัญของการพัฒนาระบบการทำงานร่วมกันแบบเวลาจริงระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M วัตถุประสงค์ของงานวิทยานิพนธ์ ขอบเขตวิทยานิพนธ์ ขั้นตอนและวิธีการดำเนินงาน สุดท้ายประโยชน์ที่คาดว่าจะได้รับ

บทที่ 2 ทฤษฎีพื้นฐาน: กล่าวถึงทฤษฎี และการประยุกต์ใช้งานของมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M

บทที่ 3 โครงสร้างและการทำงานของระบบ: กล่าวถึงโครงสร้าง ส่วนประกอบ และการทำงานของระบบที่พัฒนาขึ้น

บทที่ 4 การทดสอบการประสานข้อมูลแบบเวลาจริง: กล่าวถึงการทดสอบ และการประเมินการประสานข้อมูลแบบเวลาจริงระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M

บทที่ 5 การทดสอบระบบการทำงานร่วมกันแบบเวลาจริงระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M สำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคาร: กล่าวถึงการทดสอบการประเมินระบบที่พัฒนาขึ้นด้วยข้อมูลตัวรับรู้เลียนแบบ และข้อมูลตัวรับรู้จริงที่ติดตั้งในโครงการ CU-BEMS

บทที่ 6 บทสรุปและข้อเสนอแนะ: สรุปงานวิจัยทั้งหมดในวิทยานิพนธ์ฉบับนี้ และแนวทางการพัฒนางานวิจัยต่อไป

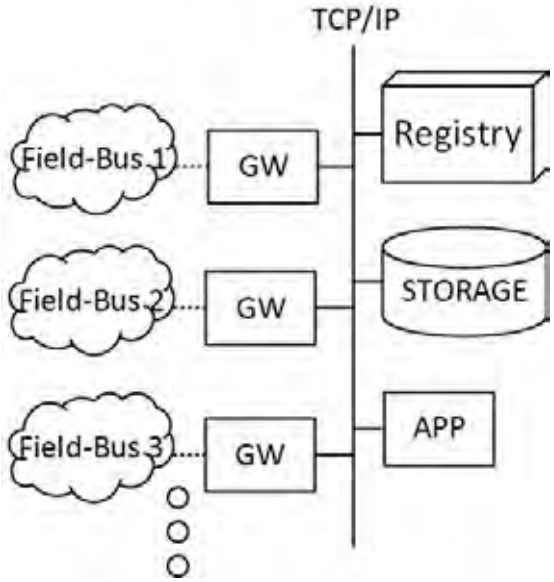
บทที่ 2
ทฤษฎีพื้นฐาน

2.1 มาตรฐาน IEEE1888 [4]

โพรโทคอลโครงข่ายการควบคุมชุมชนสีเขียวอย่างแพร่หลาย (ubiquitous green community control network, UGCCNet) หรือที่เรียกว่า มาตรฐาน IEEE1888 เป็นมาตรฐานเปิดในการสื่อสารภายในโครงข่ายทีซีพี/ไอพี (TCP/IP) โดยการเชื่อมโครงข่ายตัวรับรู้ และตัวกระตุ้นมาตรฐานต่าง ๆ ให้ทำงานร่วมกันในระบบจัดการพลังงานได้

2.1.1 สถาปัตยกรรมมาตรฐาน IEEE1888

สถาปัตยกรรมมาตรฐาน IEEE1888 ประกอบด้วยเกตเวย์ (GW) หน่วยเก็บข้อมูล (storage) และโปรแกรมประยุกต์ (APP) ที่เรียกว่า ส่วนประกอบ (component) และรีจิสทรี (registry) โดยเชื่อมต่อผ่านโครงข่ายทีซีพี/ไอพี แสดงได้ดังรูปที่ 2.1



รูปที่ 2.1: สถาปัตยกรรมมาตรฐาน IEEE1888 [4]

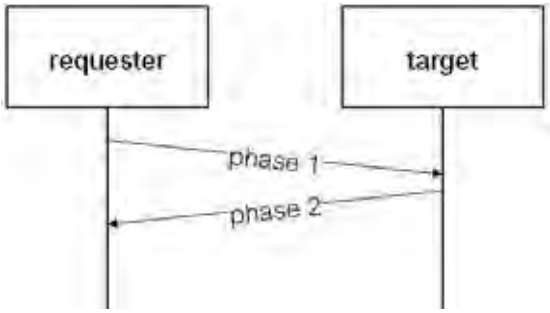
1. เกตเวย์: ทำหน้าที่ถ่ายโอนข้อมูลระหว่างโครงข่ายทีซีพี/ไอพีตามมาตรฐาน IEEE1888 และโครงข่ายฟิลด์บัส (Field-Bus) ซึ่งเชื่อมต่อตัวรับรู้ หรือ ตัวกระตุ้น
2. หน่วยเก็บข้อมูล: ทำหน้าที่บันทึกประวัติข้อมูลที่ได้รับจากเกตเวย์ โปรแกรมประยุกต์ และส่งประวัติข้อมูลไปยังเกตเวย์ โปรแกรมประยุกต์ เมื่อได้รับการร้องขอ
3. โปรแกรมประยุกต์: ทำหน้าที่ดึงประวัติข้อมูลจากหน่วยเก็บข้อมูลเพื่อแสดงผล

4. รีจิสทรี: ทำหน้าที่เป็นตัวกลางในการเชื่อมโยงระหว่างเกตเวย์ หน่วยเก็บข้อมูล และโปรแกรมประยุกต์

2.1.2 โพรโทคอลการสื่อสารหลัก

โพรโทคอลในการสื่อสารระหว่างเกตเวย์ หน่วยเก็บข้อมูล และโปรแกรมประยุกต์ โดยโพรโทคอลของข้อความที่ใช้ในการสื่อสารคือ simple object access protocol (SOAP) ประกอบด้วยโพรโทคอลย่อยดังนี้

1. โพรโทคอล WRITE ใช้สำหรับการเขียนข้อมูลไปยังส่วนประกอบอื่น โดยประกอบด้วย ผู้ร้องขอส่งข้อมูล (requester) และส่วนประกอบเป้าหมาย (target) มีขั้นตอนการทำงานดังรูปที่ 2.2

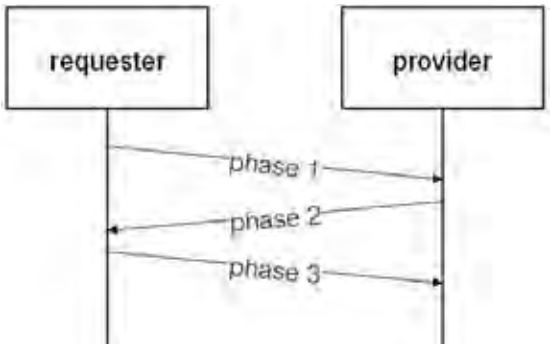


รูปที่ 2.2: โพรโทคอล WRITE [4]

   (a) เฟส 1 (phase 1): ผู้ร้องขอส่งข้อมูลไปยังส่วนประกอบเป้าหมาย

   (b) เฟส 2 (phase 2): ส่วนประกอบเป้าหมายตอบกลับไปยังผู้ร้องขอว่าข้อมูลที่รับมาถูกต้องหรือ มีข้อผิดพลาด

2. โพรโทคอล FETCH ใช้สำหรับดึงข้อมูลจากส่วนประกอบอื่น โดยประกอบด้วย ผู้ร้องขอ (requester) และผู้ให้บริการ (provider) มีขั้นตอนการทำงานดังรูปที่ 2.3



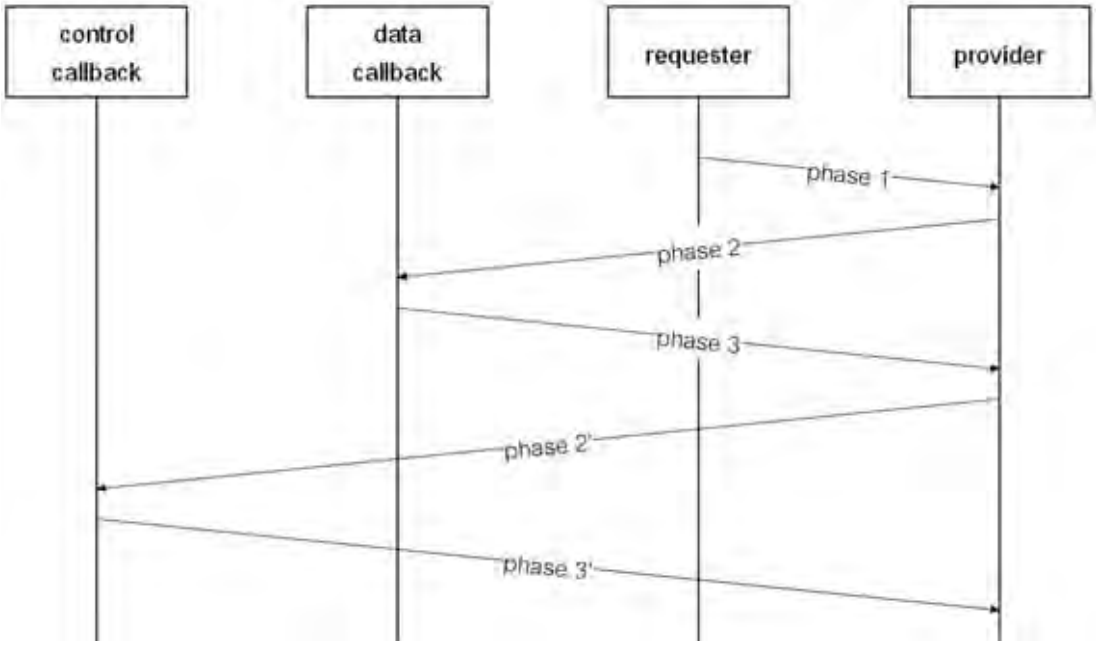
รูปที่ 2.3: โพรโทคอล FETCH [4]

(a) เฟส 1 (phase 1): ผู้ร้องขอส่งการร้องขอข้อมูลไปยังผู้ให้บริการข้อมูล

(b) เฟส 2 (phase 2): ผู้ให้บริการส่งข้อมูลไปยังผู้ร้องขอเมื่อข้อมูลมีปริมาณมากจะแบ่งข้อมูลก่อนส่ง

(c) เฟส 3 (phase 3): ผู้ร้องขอจะส่งการร้องขอข้อมูลไปยังผู้ให้บริการข้อมูลอีกครั้งเมื่อผู้ร้องขอรับข้อมูลยังไม่สิ้นสุด

3. โพรโทคอล TRAP ใช้สำหรับการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลล่วงหน้า และแจ้งเตือนข้อมูลเมื่อข้อมูลตรงกับเหตุการณ์ที่บันทึกไว้ประกอบด้วย ผู้ร้องขอ (requester) ผู้ให้บริการ (provider) เดตาคอลล์แบ็ก (data callback) และคอนโทรลคอลล์แบ็ก (control callback) มีขั้นตอนการทำงานดังรูปที่ 2.4



รูปที่ 2.4: โพรโทคอล TRAP [4]

(a) เฟส 1 (phase 1): ผู้ร้องขอส่งการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลไปยังผู้ให้บริการพร้อมกับระบุอายุการร้องขอ ยูอาร์ไอของเดตาคอลล์แบ็ก และยูอาร์ไอของคอนโทรลคอลล์แบ็กเพื่อรับข้อมูล

(b) เฟส 2 (phase 2): ผู้ให้บริการส่งข้อมูลไปยังยูอาร์ไอของเดตาคอลล์แบ็กเมื่อมีข้อมูลตรงกับเงื่อนไขที่จดทะเบียนไว้ล่วงหน้า

(c) เฟส 3 (phase 3): เดตาคอลล์แบ็กตอบกลับไปยังผู้ให้บริการว่าข้อมูลที่รับมาถูกต้อง หรือ มีข้อผิดพลาด

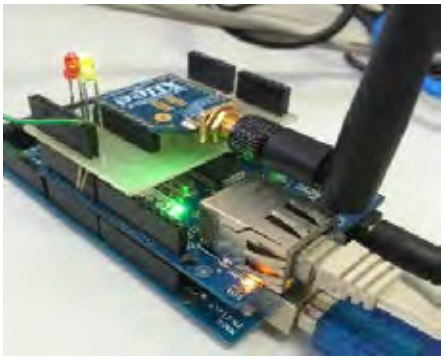
(d) เฟส 2' (phase 2'): เมื่อมีข้อผิดพลาดในเฟส 3 ผู้ให้บริการจะส่งข้อผิดพลาดที่เกิดขึ้นไปยังยูอาร์ไอของคอนโทรลคอลล์แบ็ก

(e) เฟส 3' (phase 3'): คอนโทรลคอลล์แบ็กตอบกลับไปยังผู้ให้บริการว่าข้อมูลที่รับมาถูกต้อง หรือ มีข้อผิดพลาด

2.1.3 การประยุกต์ใช้มาตรฐาน IEEE1888 ในโครงการ CU-BEMS [6]

สถาปัตยกรรม และโพรโทคอลการสื่อสารตามมาตรฐาน IEEE1888 ได้ถูกดำเนินการในโครงการ CU-BEMS สำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคาร โดยตัวรับรู้วัดสภาพแวดล้อมไร้สายมีตัวรับรู้ด้วยกัน 4 ชนิด คือ ตัวรับรู้อุณหภูมิ ตัวรับรู้ความชื้นสัมพัทธ์ ตัวรับรู้แสง และตัวรับรู้การเคลื่อนไหวของคนดังรูปที่ 2.5(a) ซึ่งตัวรับรู้วัดสภาพแวดล้อมไร้สายส่งข้อมูลตัวรับรู้ไปยังเกตเวย์ในรูปที่ 2.5(b) โดยชื่อของตัวรับรู้ถูกระบุในรูปแบบของยูอาร์ไอ ที่เรียกว่า point id หลังจากนั้นเกตเวย์ใช้โพรโทคอล WRITE ในการส่งข้อมูลตัวรับรู้วัดสภาพแวดล้อมไปยังหน่วยเก็บข้อมูลในรูปที่ 2.5(c) และโปรแกรมประยุกต์ในรูปที่ 2.5(d) ใช้โพรโทคอล FETCH เรียกข้อมูลจากหน่วยเก็บข้อมูลเพื่อแสดงผล ซึ่งโครงการ CU-BEMS ไม่ได้ดำเนินการในส่วนของรีจิสทรี และไม่ได้ใช้โพรโทคอล TRAP ในการสื่อสารข้อมูล ซึ่งงานวิจัยนี้ได้พัฒนาการใช้โพรโทคอล TRAP ในโครงการ CU-BEMS สำหรับแจ้งเตือนข้อมูลตัวรับรู้วัดสภาพแวดล้อมจากหน่วยเก็บข้อมูลแทนการใช้โพรโทคอล FETCH ในการดึงข้อมูลเพื่อให้ได้ข้อมูลที่มีความแม่นยำในการควบคุมอุปกรณ์ เช่น หลอดไฟ เครื่องใช้ไฟฟ้า เป็นต้น

(a) ตัวรับรู้วัดสภาพแวดล้อมไร้สาย



(b) เกตเวย์

(c) หน่วยเก็บข้อมูล

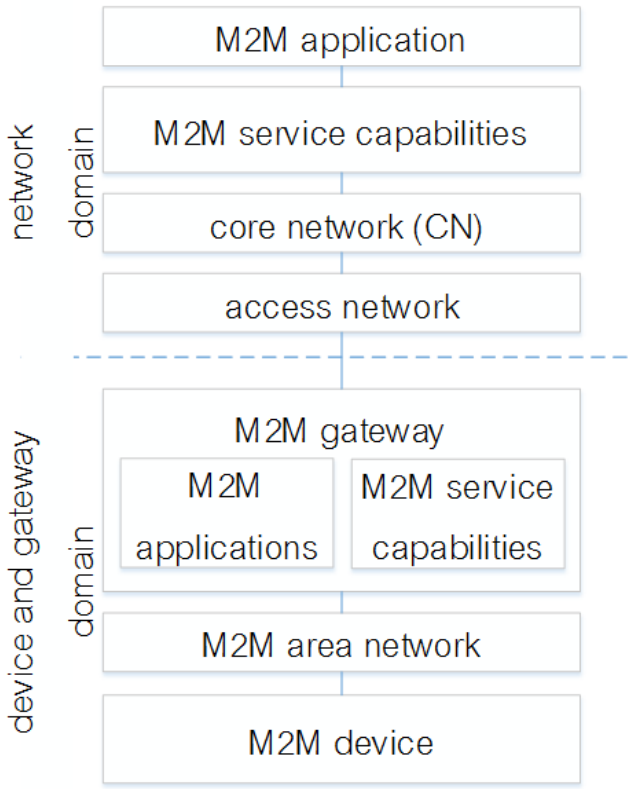
(d) ตัวอย่างโปรแกรมประยุกต์

รูปที่ 2.5: ส่วนประกอบที่ใช้ในโครงการ CU-BEMS [6]

2.2 มาตรฐาน ETSI M2M [23]

มาตรฐาน ETSI M2M เป็นมาตรฐานการสื่อสารระหว่างอุปกรณ์สำหรับเป็นแพลตฟอร์มมาตรฐาน เพื่อให้มีระบบประยุกต์ สภาพแวดล้อม และองค์ประกอบของโครงข่ายร่วมกัน

2.2.1 สถาปัตยกรรมมาตรฐาน ETSI M2M



รูปที่ 2.6: สถาปัตยกรรมการทำงานมาตรฐาน ETSI M2M [23]

สถาปัตยกรรมมาตรฐาน ETSI M2M ประกอบด้วยโดเมน 2 โดเมนคือ โดเมนโครงข่าย (network domain) และโดเมนอุปกรณ์และเกตเวย์ (device and gateway domain) ดังรูปที่ 2.6 มีรายละเอียดดังนี้

1. โดเมนโครงข่าย มีองค์ประกอบดังนี้

   (a) โครงข่ายการเข้าถึง (access network): โครงข่ายที่ช่วยให้โดเมนอุปกรณ์ และเกตเวย์เอ็มทูเอ็มสื่อสารกับโครงข่ายหลัก เช่น xDSL, W-LAN, WiMAX เป็นต้น

   (b) โครงข่ายหลัก (core network): ให้ฟังก์ชันการควบคุมโครงข่าย และการบริการ ให้การเชื่อมต่อโครงข่ายอื่น ๆ ให้การบริการข้ามแดนอัตโนมัติ เป็นต้น

(c) ความสามารถบริการเอ็มทูเอ็ม (M2M service capabilities): ให้ฟังก์ชันเอ็มทูเอ็มสำหรับการประยุกต์ใช้ที่ต่างกัน แสดงฟังก์ชันผ่านชุดของส่วนต่อประสานแบบเปิด เป็นต้น

(d) โปรแกรมประยุกต์เอ็มทูเอ็ม (M2M applications): โปรแกรมประยุกต์ที่ดำเนินงานตรรกะการบริการ และใช้ความสามารถบริการเอ็มทูเอ็มที่เข้าถึงผ่านส่วนต่อประสานแบบเปิด

2. โดเมนอุปกรณ์ และเกตเวย์ มีองค์ประกอบดังนี้

(a) อุปกรณ์เอ็มทูเอ็ม (M2M device): อุปกรณ์ที่ดำเนินงานโปรแกรมประยุกต์ซึ่งใช้ความสามารถบริการเอ็มทูเอ็ม โดยเชื่อมต่อกับโดเมนโครงข่ายทางตรง และผ่านทางเกตเวย์เอ็มทูเอ็ม

(b) โครงข่ายพื้นที่เอ็มทูเอ็ม (M2M area network): ให้การเชื่อมต่อระหว่างอุปกรณ์เอ็มทูเอ็ม และเกตเวย์เอ็มทูเอ็ม

(c) เกตเวย์เอ็มทูเอ็ม (M2M gateway): เกตเวย์ที่ดำเนินการโปรแกรมประยุกต์เอ็มทูเอ็มซึ่งใช้ความสามารถบริการเอ็มทูเอ็ม ทำหน้าที่เป็นตัวกลางระหว่างอุปกรณ์เอ็มทูเอ็ม และโดเมนโครงข่าย

2.2.2 วิธีสื่อสารหลัก

การสื่อสารระหว่างกันของโปรแกรมประยุกต์ และชั้นความสามารถบริการเอ็มทูเอ็ม (M2M SCL) ใช้สถาปัตยกรรมในรูปแบบของ RESTful สำหรับการถ่ายโอนข้อมูล ข้อมูลจะแสดงในรูปแบบของทรัพยากร (resource) ที่เป็นแบบต้นไม้ (tree) เช่น application, container, contentInstance, subscription เป็นต้น ซึ่งโครงสร้างทรัพยากรอยู่บนชั้นความสามารถบริการ

สถาปัตยกรรม RESTful ประกอบด้วยวิธีขั้นพื้นฐาน 4 วิธีซึ่งเรียกว่าวิธี CRUD ดังต่อไปนี้

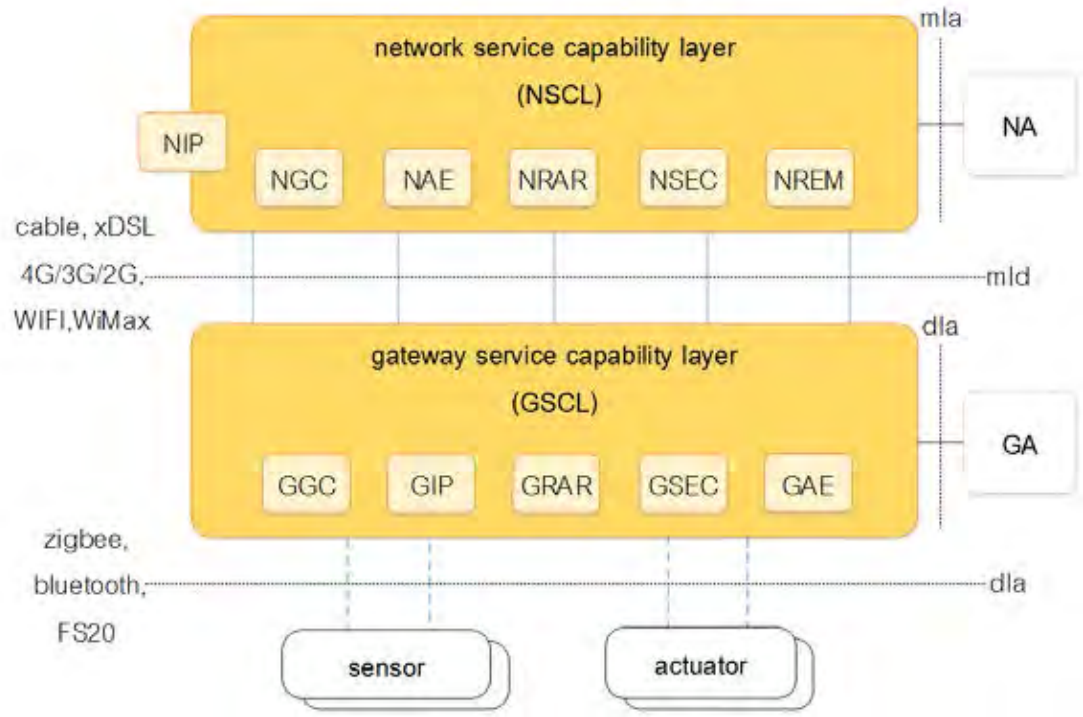
1. **CREATE**: สำหรับการสร้างทรัพยากร
2. **RETRIEVE**: สำหรับการดึงข้อมูลของทรัพยากร
3. **UPDATE**: สำหรับการแก้ไขข้อมูลของทรัพยากร
4. **DELETE**: สำหรับการลบทรัพยากร

นอกจากนี้มีวิธีเพิ่มเติมดังต่อไปนี้

1. **NOTIFY**: สำหรับการแจ้งเตือนการเปลี่ยนแปลงที่เกิดขึ้นที่ข้อมูลของทรัพยากรเป็นผลมาจากการจดทะเบียน subscription ล่วงหน้า
2. **EXECUTE**: สำหรับการดำเนินการตามคำสั่งการจัดการแสดงโดยทรัพยากร

2.2.3 การประยุกต์ใช้มาตรฐาน ETSI M2M ในแพลตฟอร์ม OpenMTC [19]

แพลตฟอร์ม OpenMTC เป็นมิดเดิลแวร์แพลตฟอร์มที่ดำเนินการโดยใช้มาตรฐาน ETSI M2M สำหรับโครงข่ายไฟฟ้าอัจฉริยะ เมืองอัจฉริยะ อาคารอัจฉริยะ เป็นต้น เพื่อการวิจัย และพัฒนาทางด้านการสื่อสารระหว่างอุปกรณ์รวมถึงอินเทอร์เน็ตของสรรพสิ่ง ซึ่งพัฒนาโดยมหาวิทยาลัยเทคนิคแห่งเบอร์ลิน (technical university of Berlin, TUB) [16] และสถาบัน Fraunhofer FOKUS [17] โดยมีสถาปัตยกรรม และจุดอ้างอิงของแพลตฟอร์ม OpenMTC ดังรูปที่ 2.7 มีรายละเอียดดังนี้



รูปที่ 2.7: สถาปัตยกรรม และจุดอ้างของแพลตฟอร์ม OpenMTC [19]

1. ชั้นความสามารถบริการเอ็มทูเอ็ม (M2M service capability layer, M2M SCL) ประกอบด้วย
   (a) ชั้นความสามารถบริการโครงข่าย (network service capability layer, NSCL)
   (b) ชั้นความสามารถบริการเกตเวย์ (gateway service capability layer, GSCL)
2. ความสามารถบริการเอ็มทูเอ็ม มีความสามารถหลัก ๆ ดังต่อไปนี้
   (a) เออี (application Enablement, xAE)
   (b) จีซี (generic communication, xGC)
   (c) อาร์เออาร์ (reachability, addressing and repository, xRAR)
   (d) อาร์อีเอ็ม (remote entity management, xREM)

(e) เอสอีซี (security, xSEC)

(f) ไอพี (interworking proxy, xIP)

โดยที่ x คือ N สำหรับ โครงข่าย (network), G สำหรับ เกตเวย์ (gateway) และ D สำหรับ อุปกรณ์ (device)

3. โปรแกรมประยุกต์ประกอบด้วย

(a) โปรแกรมประยุกต์โครงข่าย (network application, NA)

(b) โปรแกรมประยุกต์เกตเวย์ (gateway application, GA)

(c) โปรแกรมประยุกต์อุปกรณ์ (device application, DA)

4. จุดอ้างอิง

(a) จุดอ้างอิง mIa: ให้โปรแกรมประยุกต์โครงข่ายเข้าถึงความสามารถบริการเอ็มทูเอ็ม

(b) จุดอ้างอิง dIa: ให้โปรแกรมประยุกต์อุปกรณ์เข้าถึงความสามารถบริการเอ็มทูเอ็มในอุปกรณ์เอ็มทูเอ็มเดียวกัน หรือ เกตเวย์เอ็มทูเอ็ม และให้โปรแกรมประยุกต์เกตเวย์เข้าถึงความสามารถบริการเอ็มทูเอ็มในเกตเวย์เอ็มทูเอ็มเดียวกัน

(c) จุดอ้างอิง mId: ให้ความสามารถบริการเอ็มทูเอ็มที่อยู่ในอุปกรณ์เอ็มทูเอ็ม หรือ เกตเวย์เอ็มทูเอ็มสื่อสารกับความสามารถบริการเอ็มทูเอ็มในโดเมนโครงข่าย

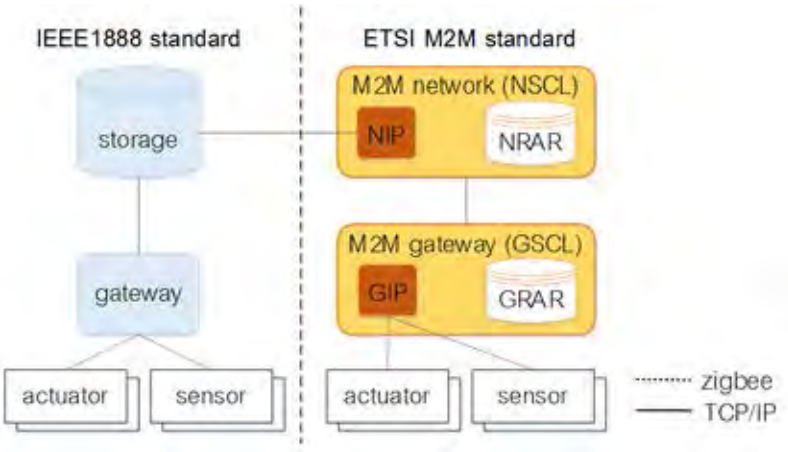
บทที่ 3

โครงสร้างและการทำงานของระบบ

ในบทนี้นำเสนอระบบการทำงานร่วมกันแบบเวลาจริงระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M โดยการพัฒนาเอ็นไอพีในการประสานข้อมูลแบบเวลาจริงและพัฒนาเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M เพื่อให้ตัวรับรู้ และตัวกระตุ้นตามมาตรฐานที่ต่างกันสามารถทำงานร่วมกันได้แบบเวลาจริง ซึ่งมีโครงสร้างของระบบ ส่วนประกอบของระบบ และการทำงานของระบบ ดังนี้

3.1 โครงสร้างของระบบที่นำเสนอ

การออกแบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETI M2M โดยการพัฒนาไอพีตามมาตรฐาน ETSI M2M มีหน้าที่ช่วยให้เกิดการทำงานร่วมกับมาตรฐานอื่นซึ่งมีไอพีให้พิจารณา 2 แบบ คือ เอ็นไอพี (network interworking proxy, NIP) และจีไอพี (gateway interworking proxy, GIP) เนื่องจากเกตเวย์ถูกออกแบบบนบอร์ดไมโครคอนโทรลเลอร์สำเร็จรูปซึ่งมีข้อจำกัดที่ความเร็วการประมวลผล และหน่วยความจำ ยกตัวอย่างเช่น เกตเวย์ของโครงการ CU-BEMS ที่ใช้บอร์ดไมโครคอนโทรลเลอร์สำเร็จรูปที่เรียกว่า arduino mega 2560 มีหน้าที่ถ่ายโอนข้อมูลตัวรับรู้ไปยังหน่วยเก็บข้อมูลของ IEEE1888 โดยบอร์ด arduino mega 2560 มีความเร็วการประมวลผล 16 เมกะเฮิรตซ์ และหน่วยความจำ 256 กิโลไบต์ และบอร์ด Raspberry Pi B+ ที่ทำหน้าที่เป็นเกตเวย์ในระบบที่นำเสนอมีความเร็วการประมวลผล 700 เมกะเฮิรตซ์ และหน่วยความจำ 8 กิกะไบต์ ซึ่งมีสมรรถนะที่ไม่สูงมากนักจึงมีข้อจำกัดเมื่อต้องรับ ส่ง และบันทึกข้อมูลจำนวนมาก ดังนั้นระบบที่นำเสนอได้เลือกพัฒนาเอ็นไอพีที่ดำเนินการบนตัวบริการ (server) ที่มีสมรรถนะที่สูง เช่น คอมพิวเตอร์ส่วนบุคคล และตัวบริการที่ให้บริการหน่วยเก็บข้อมูล IEEE1888 ในโครงการ CU-BEMS เป็นต้น



รูปที่ 3.1: โครงสร้างของระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M

นอกจากการพัฒนาเอ็นไอพีแล้วได้มีการพัฒนาเกตเวย์ตามมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M เพื่อให้ตัวรับรู้ และตัวกระตุ้นตามมาตรฐานที่ต่างกันสามารถทำงานร่วมกันได้แบบเวลาจริง โดยโครงสร้างของระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M ที่นำเสนอดังรูปที่ 3.1 แบ่งเป็น 2 ส่วนดังนี้

1. มาตรฐาน IEEE1888 ประกอบด้วย
   (a) หน่วยเก็บข้อมูล: ทำหน้าที่บันทึกข้อมูล และส่งข้อมูลเมื่อได้รับการร้องขอ
   (b) เกตเวย์: ทำหน้าที่ถ่ายโอนข้อมูลระหว่างโครงข่ายตัวรับรู้และตัวกระตุ้นไร้สาย ZigBee และหน่วยเก็บข้อมูล
2. มาตรฐาน ETSI M2M ประกอบด้วย
   (a) ชั้นความสามารถบริการโครงข่าย (network service capability layer, NSCL) มีความสามารถหลักดังนี้
      i. เอ็นไอพี: ทำหน้าที่ช่วยให้เกิดการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M โดยการประสานข้อมูลแบบเวลาจริง
      ii. เอ็นอาร์เออาร์ (network reachability, addressing and repository, NRAR): ทำหน้าที่เป็นคลังเก็บข้อมูลหลักของโครงข่ายในการจัดเก็บข้อมูล บันทึกการจดทะเบียนติดตามข้อมูล และแจ้งเตือนข้อมูล
   (b) ชั้นความสามารถบริการเกตเวย์ (gateway service capability layer, GSCL) มีความสามารถหลักดังนี้
      i. จีไอพี: ทำหน้าที่ถ่ายโอนข้อมูลระหว่างโครงข่ายตัวรับรู้และตัวกระตุ้นไร้สาย ZigBee และแพลตฟอร์มมาตรฐาน ETSI M2M
      ii. จีอาร์เออาร์ (gateway reachability, addressing and repository, GRAR): ทำหน้าที่เป็นคลังเก็บข้อมูลของเกตเวย์ในการจัดเก็บข้อมูล บันทึกการจดทะเบียนติดตามข้อมูล และแจ้งเตือนข้อมูล

3.2 ส่วนประกอบของระบบ

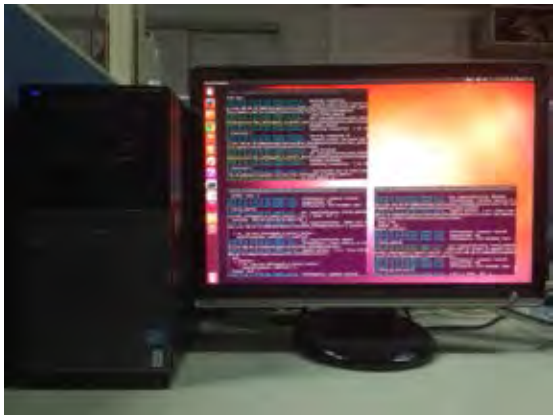
ระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M ที่นำเสนอมีรายละเอียดของส่วนประกอบหลักดังตารางที่ 3.1 โดยใช้หน่วยเก็บข้อมูลของ IEEE1888 ในโครงการ CU-BEMS ซึ่งใช้ฐานข้อมูลที่เรียกว่า โพสต์เกรสคิวเอล (PostgreSQL) ในส่วนของโครงข่ายของ ETSI M2M และเกตเวย์ของ ETSI M2M เป็นแพลตฟอร์ม OpenMTC ที่ใช้ฐานข้อมูลที่เรียกว่า มองโกดีบี (MongoDB) ดำเนินการโดยแพลตฟอร์มที่เรียกว่า node.js [24] ซึ่งเขียนโปรแกรมด้วยภาษา JavaScript มีคุณสมบัติเด่นคือ เป็นรูปแบบ non-blocking i/o หรือ ไม่ประสานเวลา (asynchronous) โดยมีความเร็วในการดำเนินงาน และมีมอดูล (module) สำเร็จรูปให้ใช้มากมาย ซึ่งแพลตฟอร์ม node.js ได้ถูกใช้บนเกตเวย์ของ IEEE1888 ด้วย

งานวิจัยนี้พัฒนาเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M ดำเนินการบนบอร์ด Raspberry Pi B+ เนื่องจากสามารถรองรับแพลตฟอร์ม node.js และเป็นบอร์ดที่ได้รับความนิยมในปัจจุบัน โดยหน่วยเก็บข้อมูลของ IEEE1888, โครงข่ายของ ETSI M2M, เกตเวย์ของ IEEE1888

และเกตเวย์ของ ETSI M2M เชื่อมต่อกันด้วยข่ายงานบริเวณเฉพาะที่ (local area network, LAN) นอกจากนี้เกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M เชื่อมต่อโมดูล ZigBee ผ่านบอร์ดที่เรียกว่า mini xbee usb dongle V2 จากรูปที่ 3.3 และ 3.4 แสดงแผนภาพบล็อกการเชื่อมต่ออุปกรณ์ และภาพอุปกรณ์ที่ใช้งาน ตามลำดับ โดยในส่วนของตัวรับรู้ได้ใช้ตัวรับรู้ไร้สาย ZigBee ในโครงการ CU-BEMS และงานวิจัยนี้พัฒนาตัวกระตุ้นไร้สาย ZigBee มีส่วนประกอบ ประกอบด้วย โมดูล ZigBee เชื่อมต่อกับบอร์ดสำเร็จรูปที่เรียกว่า arduino uno R3 ผ่านบอร์ดส่วนต่อขยาย ZigBee ซึ่งบอร์ด arduino uno R3 มีความเร็วนาฬิกา 16 เมกะเฮิรตซ์ ใช้งานง่าย และมีราคาไม่สูง โดยบอร์ด arduino ต่อกับบอร์ดสำเร็จรูปรีเลย์ในการเปิดปิดอุปกรณ์ แสดงแผนภาพบล็อกการเชื่อมต่ออุปกรณ์ดังรูปที่ 3.5 และภาพอุปกรณ์ที่ใช้งานดังรูปที่ 3.6

ตารางที่ 3.1: รายละเอียดของส่วนประกอบหลักของระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M

ส่วนประกอบหลัก	รายละเอียด
หน่วยเก็บข้อมูลของ IEEE1888 รูปที่ 2.5(c)	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 โปรแกรม Apache
โครงข่ายของ ETSI M2M รูปที่ 3.2	ตราสินค้า ASUS คุณลักษณะ Intel core 2 duo 2.66 GHz x 2 processor, 4 GB RAM ระบบปฏิบัติการ Ubuntu 12.04 LTS, 64-bit โปรแกรม node.js
เกตเวย์ของ IEEE1888 รูปที่ 3.4(a)	ตราสินค้า Raspberry Pi B+ คุณลักษณะ arm1176jzf-s core, 700 MHz processor, 512 MB RAM ระบบปฏิบัติการ Raspbian โปรแกรม node.js
เกตเวย์ของ ETSI M2M รูปที่ 3.4(b)	ตราสินค้า Raspberry Pi B+ คุณลักษณะ arm1176jzf-s core, 700 MHz processor, 512 MB RAM ระบบปฏิบัติการ Raspbian โปรแกรม node.js



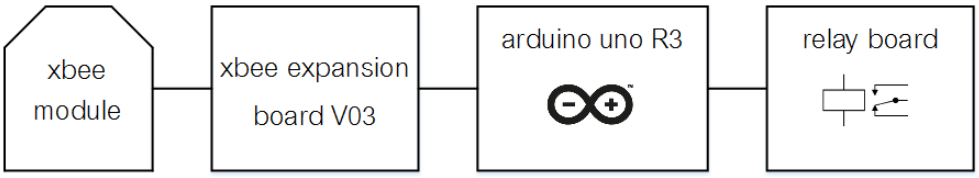
รูปที่ 3.2: ส่วนประกอบสำหรับทำหน้าที่โครงข่ายของ ETSI M2M

รูปที่ 3.3: แผนภาพบล็อกการเชื่อมต่ออุปกรณ์ของเกตเวย์

(a) ส่วนประกอบสำหรับทำหน้าที่เกตเวย์ของ IEEE1888

(b) ส่วนประกอบสำหรับทำหน้าที่เกตเวย์ของ ETSI M2M

รูปที่ 3.4: ส่วนประกอบสำหรับทำหน้าที่เกตเวย์



รูปที่ 3.5: แผนภาพบล็อกการเชื่อมต่ออุปกรณ์ของตัวกระตุ้นไร้สาย ZigBee

รูปที่ 3.6: ตัวกระตุ้นไร้สาย ZigBee

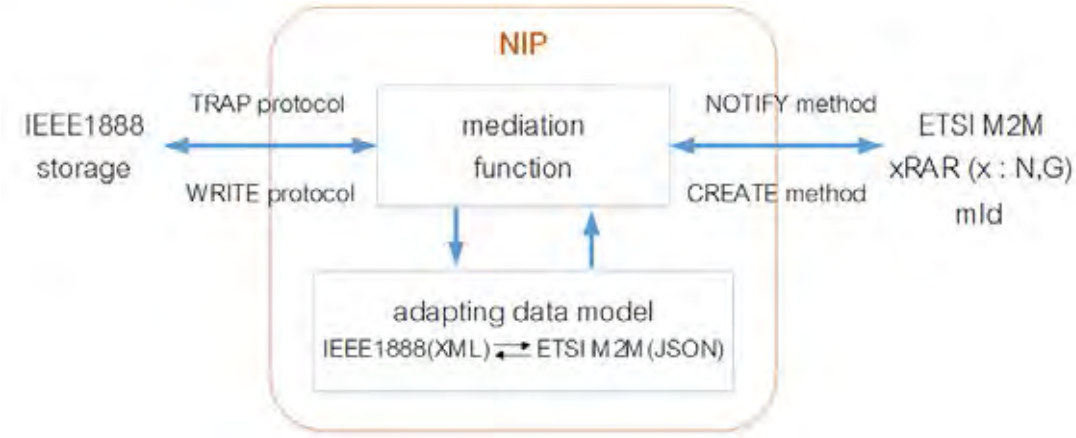
3.3 การทำงานของระบบ

งานวิจัยที่ [21] และ [22] ได้พัฒนาเอ็นไอพีใช้โพรโทคอล FETCH กับโพรโทคอล WRITE ตามมาตรฐาน IEEE1888 และวิธี RETRIEVE กับวิธี CREATE ตามมาตรฐาน ETSI M2M ซึ่งมีข้อจำกัดที่ไม่ได้ประสานข้อมูลแบบเวลาจริง ดังนั้นงานวิจัยนี้ได้นำเสนอการประสานข้อมูลแบบเวลาจริงโดยการใช้โพรโทคอล TRAP และวิธี NOTIFY แทนโพรโทคอล FETCH และวิธี RETRIEVE ตามลำดับ เอ็นไอพีทำหน้าที่เป็นตัวกลางในการประสานข้อมูลระว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M ตามแผนภาพบล็อกการทำงานดังรูปที่ 3.7 โดยทำหน้าที่เป็นฟังก์ชันการเป็นสื่อกลาง (mediation function) ในการเรียกข้อมูลจากมาตรฐานหนึ่ง และปรับตัวแบบข้อมูล (adapting data model) หลังจากนั้นส่งข้อมูลไปยังอีกมาตรฐานหนึ่ง

วิทยานิพนธ์นี้ได้พัฒนาเอ็นไอพีบนคอมพิวเตอร์ส่วนบุคคล ดังรูปที่ 3.2 สำหรับการประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และอาร์เออาร์ของ ETSI M2M แบบเวลาจริงด้วยแพลตฟอร์ม node.js ซึ่งเป็นแพ็กเกจสำหรับพัฒนาตัวบริการด้วยภาษา JavaScript โดยโปรแกรมของเอ็นไอพีที่ได้พัฒนาประกอบด้วยดังนี้

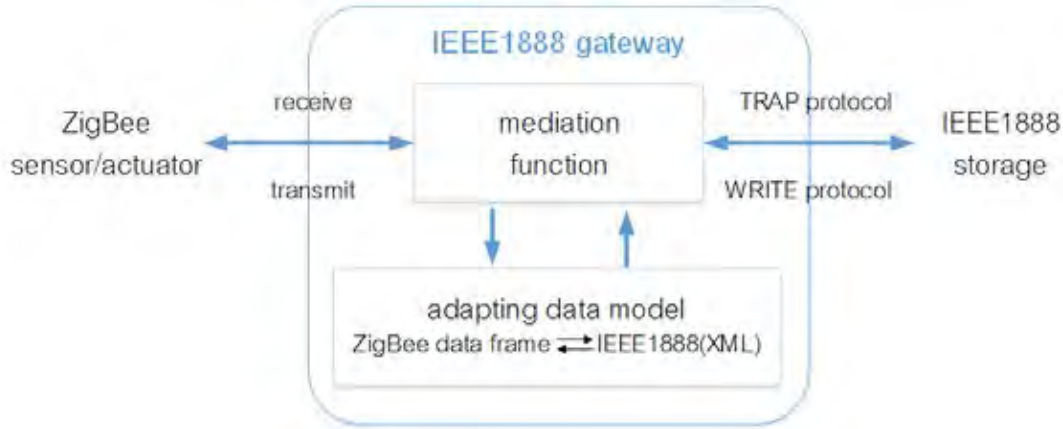
1. โปรแกรมร้องขอการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลตามโพรโทคอล TRAP ที่หน่วยเก็บข้อมูลของ IEEE1888 ดังภาคผนวก ก.
2. โปรแกรมรับข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 และแจงข้อมูลรูปแบบ XML พร้อมกับปรับข้อมูลเป็นตามโครงสร้างมาตรฐาน ETSI M2M ที่เป็นรูปแบบ JSON ดังภาคผนวก ข.
3. โปรแกรมส่งข้อมูลไปยังเอ็นอาร์เออาร์ตามวิธี CREATE ดังภาคผนวก ค.

4. โปรแกรมส่งติดตามข้อมูลตามวิธี CREATE และรับการแจ้งเตือนข้อมูลตามวิธี NOTIFY จากจีอาร์เออาร์ พร้อมกับแจงข้อมูลรูปแบบ JSON และปรับโครงสร้างของข้อมูลที่ได้รับมาเป็นตามโครงสร้างของข้อมูลแบบ XML ของมาตรฐาน IEEE1888 ดังภาคผนวก ง.
5. โปรแกรมส่งข้อมูลไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล WRITE ดังภาคผนวก จ.



รูปที่ 3.7: แผนภาพบล็อกการทำงานของเอ็นไอพี

นอกจากนี้วิทยานิพนธ์นี้ได้พัฒนาเกตเวย์ของ IEEE1888 และจีไอพีด้วยแพลตฟอร์ม node.js บนบอร์ด Raspberry Pi B+ แสดงดังรูปที่ 3.4(a) และ 3.4(b) เพื่อสื่อสารกับตัวรับรู้ และตัวกระตุ้นไร้สาย ZigBee โดยโปรแกรมที่เกตเวย์ของ IEEE1888 มีการทำงานตามรูปที่ 3.8 ประกอบด้วยดังนี้

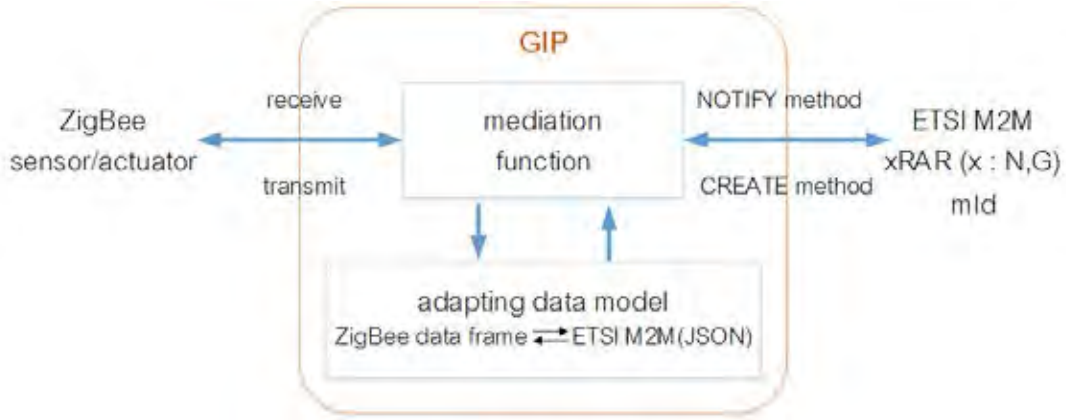


รูปที่ 3.8: แผนภาพบล็อกการทำงานของเกตเวย์ของ IEEE1888

1. โปรแกรมรับข้อมูลจากตัวรับรู้ไร้สาย ZigBee พร้อมกับตั้งชื่อโนดตัวรับรู้ และคำนวณค่าตัวรับรู้ดังภาคผนวก ฉ.
2. โปรแกรมปรับข้อมูลเป็นตามโครงสร้างของข้อมูลแบบ XML ของมาตรฐาน IEEE1888 และส่งข้อมูลไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล WRITE ดังภาคผนวก จ.

3. โปรแกรมร้องขอการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลตามโพรโทคอล TRAP ที่หน่วยเก็บข้อมูลของ IEEE1888 ดังภาคผนวก ก.
4. โปรแกรมรับข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 และแจงข้อมูลรูปแบบ XML ดังภาคผนวก ข.
5. โปรแกรมปรับข้อมูลเป็นข้อมูลควบคุม และส่งข้อมูลไปยังตัวกระตุ้นไร้สาย ZigBee ดังภาคผนวก ฉ.

ในส่วนของโปรแกรมที่จีไอพีมีการทำงานตามรูปที่ 3.9 ประกอบด้วยดังนี้



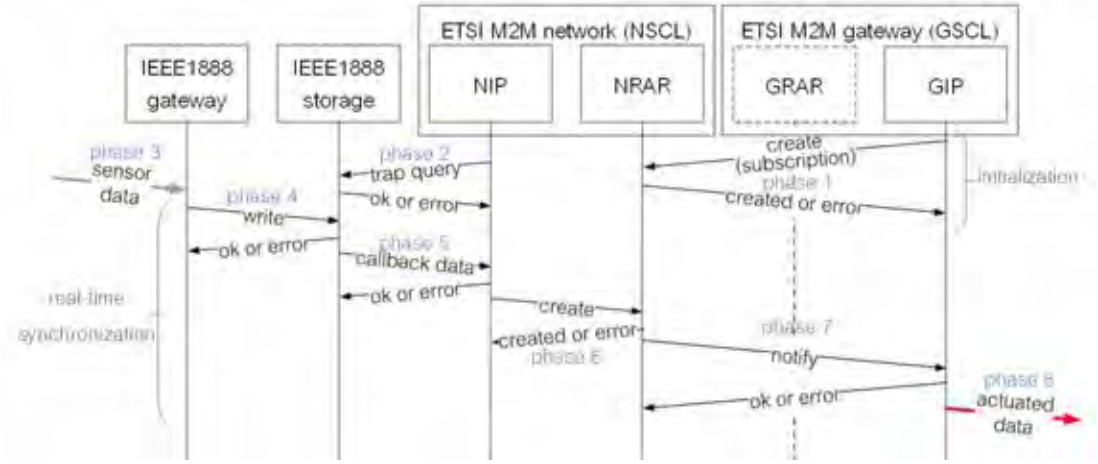
รูปที่ 3.9: แผนภาพบล็อกการทำงานของจีไอพี

1. โปรแกรมรับข้อมูลจากตัวรับรู้ไร้สาย ZigBee พร้อมกับตั้งชื่อโนดตัวรับรู้ และคำนวณค่าตัวรับรู้ดังภาคผนวก ฉ.
2. โปรแกรมปรับข้อมูลเป็นตามโครงสร้างของมาตรฐาน ETSI M2M และส่งข้อมูลไปยังจีอาร์เออาร์ตามวิธี CREATE ดังภาคผนวก ค.
3. โปรแกรมการส่งติดตามข้อมูลตามวิธี CREATE และรับการแจ้งเตือนข้อมูลตามวิธี NOTIFY จากเอ็นอาร์เออาร์ พร้อมกับแจงข้อมูลรูปแบบ JSON ดังภาคผนวก ง.
4. โปรแกรมปรับข้อมูลเป็นข้อมูลควบคุม และส่งข้อมูลไปยังตัวกระตุ้นไร้สาย ZigBee ดังภาคผนวก ฉ.

จากการพัฒนาเอ็นไอพี จีไอพี และเกตเวย์ของ IEEE1888 วิทยานิพนธ์นี้ได้นำเสนอระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M โดยการถ่ายโอนข้อมูลระหว่างเกตเวย์ทั้ง 2 มาตรฐานแบบเวลาจริง ซึ่งมีการสื่อสารของระบบดังนี้

3.3.1 การสื่อสารสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M แบบเวลาจริง

การถ่ายโอนข้อมูลตัวรับรู้จากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M มีการสื่อสารดังรูปที่ 3.10 และมีรายละเอียดดังนี้



รูปที่ 3.10: ผังเวลาการถ่ายโอนข้อมูลตัวรับรู้มาตรฐาน IEEE1888 ไปยังเกตเวย์ของ ETSI M2M แบบเวลาจริง

1. เฟส 1 (phase 1): จีไอพีรับข้อมูลตัวรับรู้โดยการส่งข้อมูล subscription สำหรับการติดตามข้อมูลตัวรับรู้ไปยังเอ็นอาร์เออาร์ตามวิธี CREATE หลังจากนั้นเกตเวย์อินเตอร์เวิร์กกิงพร็อกซีรับการตอบกลับว่าข้อมูลที่ส่งไปถูกต้อง หรือ มีข้อผิดพลาด

2. เฟส 2 (phase 2): เอ็นไอพีทำหน้าที่ผู้ร้องขอส่งข้อมูลการร้องขอแทรป (trap query) สำหรับจดทะเบียนเงื่อนไขในการแจ้งเตือนข้อมูลตัวรับรู้ไปยังผู้ให้บริการคือ หน่วยเก็บข้อมูลของ IEEE1888 พร้อมกับระบุอายุของเงื่อนไข โดยกำหนดยูอาร์ไอของเดตาคอลล์แบ็ก และคอนโทรลคอลล์แบ็กเป็นยูอาร์ไอของเอ็นไอพีเพื่อรับข้อมูลตัวรับรู้ตามโพรโทคอล TRAP หลังจากนั้นเอ็นไอพีรับการตอบกลับว่าข้อมูลที่ส่งไปถูกต้อง หรือ มีข้อผิดพลาด โดยในเฟส 1 และ 2 เป็นในส่วนของการเริ่มต้น (initialization) ของการถ่ายโอนข้อมูล

3. เฟส 3 (phase 3): เกตเวย์ของ IEEE1888 รับข้อมูลตัวรับรู้จากโครงข่ายตัวรับรู้ไร้สาย ZigBee

4. เฟส 4 (phase 4): เกตเวย์ของ IEEE1888 ทำหน้าที่ผู้ร้องขอส่งข้อมูลตัวรับรู้ไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล WRITE หลังจากนั้นหน่วยเก็บข้อมูลของ IEEE1888 ตอบกลับไปยังเกตเวย์ของ IEEE1888 ว่าข้อมูลที่รับมาถูกต้อง หรือ มีข้อผิดพลาด

5. เฟส 5 (phase 5): หน่วยเก็บข้อมูลของ IEEE1888 ส่งข้อมูลตัวรับรู้ไปยังเอ็นไอพีทันทีเมื่อมีข้อมูลตรงกับเงื่อนไขในเฟส 2 เมื่อเอ็นไอพีรับข้อมูล เอ็นไอพีแจงข้อมูลรูปแบบ XML และส่งการตอบกลับไปยังหน่วยเก็บข้อมูลของ IEEE1888 ว่าข้อมูลที่รับมาถูกต้อง หรือ มีข้อผิดพลาด

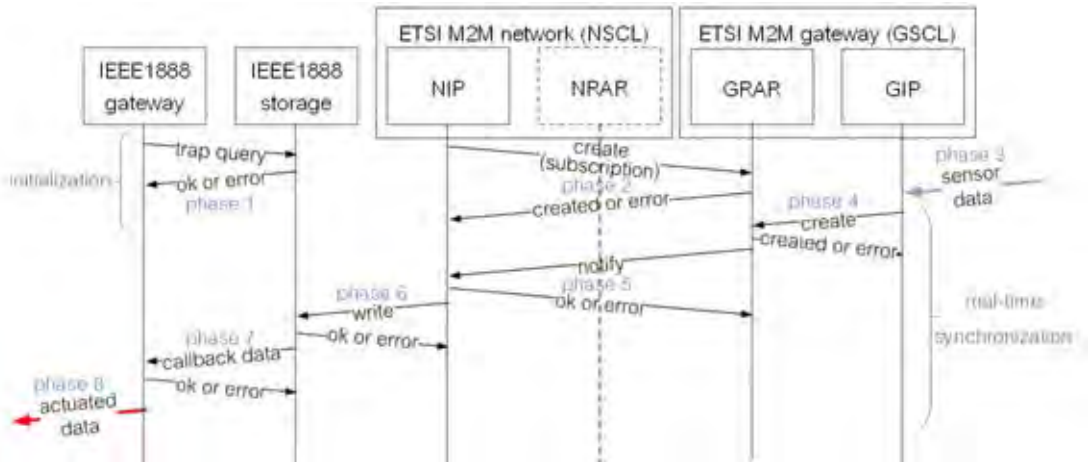
6. เฟส 6 (phase 6): เมื่อข้อมูลที่เอ็นไอพีรับมาถูกต้อง เอ็นไอพีจะปรับโครงสร้างของข้อมูลที่ได้รับมานั้นให้เป็นตามโครงสร้างของข้อมูลแบบ JSON ที่กำหนดในมาตรฐาน ETSI M2M และส่งข้อมูลไปยังเอ็นอาร์เออาร์ตามวิธี CREATE หลังจากนั้นเอ็นไอพีรับการตอบกลับว่าข้อมูลที่ส่งไปถูกต้อง หรือ มีข้อผิดพลาด

7. เฟส 7 (phase 7): เมื่อข้อมูลที่เอ็นไอพีส่งไปถูกต้อง เอ็นอาร์เออาร์ตอบกลับข้อมูลตัวรับรู้ตามที่ติดตามข้อมูลไว้ในเฟส 1 มายังจีไอพีทันทีด้วยวิธี NOTIFY หลังจากนั้นจีไอพีแจงข้อมูลรูปแบบ JSON และตอบกลับว่าข้อมูลที่รับมาถูกต้อง หรือ มีข้อผิดพลาด

8. เฟส 8 (phase 8): เมื่อข้อมูลที่จีไอพีรับมาถูกต้อง จีไอพีปรับข้อมูลตัวรับรู้เป็นข้อมูลกระตุ้นสำหรับการควบคุม และส่งข้อมูลกระตุ้นไปยังโครงข่ายตัวกระตุ้นไร้สาย ZigBee ต่อไป

3.3.2 การสื่อสารสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 แบบเวลาจริง

การถ่ายโอนข้อมูลตัวรับรู้จากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 มีการสื่อสารดังรูปที่ 3.11 และมีรายละเอียดดังนี้



รูปที่ 3.11: ผังเวลาการถ่ายโอนข้อมูลตัวรับรู้มาตรฐาน ETSI M2M ไปยังเกตเวย์ของ IEEE1888 แบบเวลาจริง

1. เฟส 1 (phase 1): เกตเวย์ของ IEEE1888 ทำหน้าที่ผู้ร้องขอส่งข้อมูลการร้องขอแทรปสำหรับเงื่อนไขในการแจ้งเตือนข้อมูลตัวรับรู้ไปยังผู้ให้บริการคือ หน่วยเก็บข้อมูลของ IEEE1888 พร้อมกับระบุอายุของเงื่อนไข โดยกำหนดยูอาร์ไอของเดตาคอลล์แบ็ก และคอนโทรลคอลล์แบ็กเป็นยูอาร์ไอของเกตเวย์ของ IEEE1888 เพื่อรับข้อมูลตัวรับรู้ตามโพรโทคอล TRAP หลังจากนั้นเกตเวย์ของ IEEE1888 รับการตอบกลับว่าข้อมูลที่ส่งไปถูกต้อง หรือ มีข้อผิดพลาด

2. เฟส 2 (phase 2): เอ็นไอพีรับข้อมูลตัวรับรู้โดยการส่งข้อมูล subscription สำหรับการติดตามข้อมูลตัวรับรู้ไปยังจีอาร์เออาร์ด้วยวิธี CREATE หลังจากนั้นเน็ทเวิร์คอินเตอร์เวิร์กกิงพร็อกซีรับการตอบกลับว่าข้อมูลที่ส่งไปถูกต้อง หรือ มีข้อผิดพลาด โดยในเฟส 1 และ 2 เป็นในส่วนของการเริ่มต้นของการถ่ายโอนข้อมูล

3. เฟส 3 (phase 3): จีไอพีรับข้อมูลตัวรับรู้จากโครงข่ายตัวรับรู้ไร้สาย ZigBee

4. เฟส 4 (phase 4): จีไอพีส่งข้อมูลไปยังจีอาร์เออาร์ด้วยวิธี CREATE หลังจากนั้นจีไอพีรับการตอบกลับว่าข้อมูลที่ส่งไปถูกต้อง หรือ มีข้อผิดพลาด

5. เฟส 5 (phase 5): จีอาร์เออาร์ตอบกลับข้อมูลตัวรับรู้ตามที่ติดตามข้อมูลไว้ในเฟส 2 มายังเอ็นไอพีทันทีด้วยวิธี NOTIFY หลังจากนั้นเอ็นไอพีแจงข้อมูลรูปแบบ JSON และตอบกลับว่าข้อมูลที่รับมาถูกต้อง หรือ มีข้อผิดพลาด

6. เฟส 6 (phase 6): เมื่อข้อมูลที่เอ็นไอพีรับมาถูกต้อง เอ็นไอพีจะปรับโครงสร้างของข้อมูลที่ได้รับมานั้นให้เป็นตามโครงสร้างของข้อมูลแบบ XML ที่กำหนดในมาตรฐาน IEEE1888 และส่งข้อมูลตัวรับรู้ไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล WRITE หลังจากนั้นหน่วยเก็บข้อมูลของ IEEE1888 ตอบกลับไปยังเอ็นไอพีว่าข้อมูลที่รับมาถูกต้อง หรือ มีข้อผิดพลาด

7. เฟส 7 (phase 7): หน่วยเก็บข้อมูลของ IEEE1888 ส่งข้อมูลตัวรับรู้ไปยังเกตเวย์ของ IEEE1888 ทันทีเมื่อมีข้อมูลตรงกับเงื่อนไขในเฟส 1 หลังจากนั้นเกตเวย์ของ IEEE1888 แจงข้อมูลรูปแบบ XML และตอบกลับไปยังหน่วยเก็บข้อมูลของ IEEE1888 ว่าข้อมูลที่รับมาถูกต้อง หรือ มีข้อผิดพลาด

8. เฟส 8 (phase 8): เมื่อข้อมูลที่เกตเวย์ของ IEEE1888 รับมาถูกต้อง เกตเวย์ของ IEEE1888 ปรับข้อมูลตัวรับรู้เป็นข้อมูลกระตุ้นสำหรับการควบคุม หลังจากนั้นเกตเวย์ของ IEEE1888 ส่งข้อมูลกระตุ้นไปยังโครงข่ายตัวกระตุ้นไร้สาย ZigBee ต่อไป

บทที่ 4

การทดสอบการประสานข้อมูลแบบเวลาจริง

ในบทที่ผ่านมาได้นำเสนอภาพรวมของระบบ การทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETI M2M โดยการถ่ายโอนข้อมูลระหว่างเกตเวย์ทั้ง 2 มาตรฐานแบบเวลาจริง จากการพัฒนาเอ็นไอพี จีไอพี และเกตเวย์ของ IEEE1888

บทนี้กล่าวถึงการทดสอบเอ็นไอพีในการประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M แบบเวลาจริง โดยเอ็นไอพีใช้โพรโทคอล TRAP และวิธี NOTIFY ในการแจ้งเตือนข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M ตามลำดับ นอกจากนี้เอ็นไอพีใช้โพรโทคอล WRITE และวิธี CREATE ในการส่งข้อมูลไปเก็บยังหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M ตามลำดับ เพื่อพิจารณารูปแบบการสื่อสาร และปริมาณงาน ซึ่งส่วนประกอบสำหรับการทดสอบประกอบด้วย เอ็นไอพี เอ็นอาร์เออาร์ และหน่วยเก็บข้อมูลของ IEEE1888 แสดงรายละเอียดดังตารางที่ 4.1

ตารางที่ 4.1: รายละเอียดของส่วนประกอบที่ใช้ในการทดสอบ

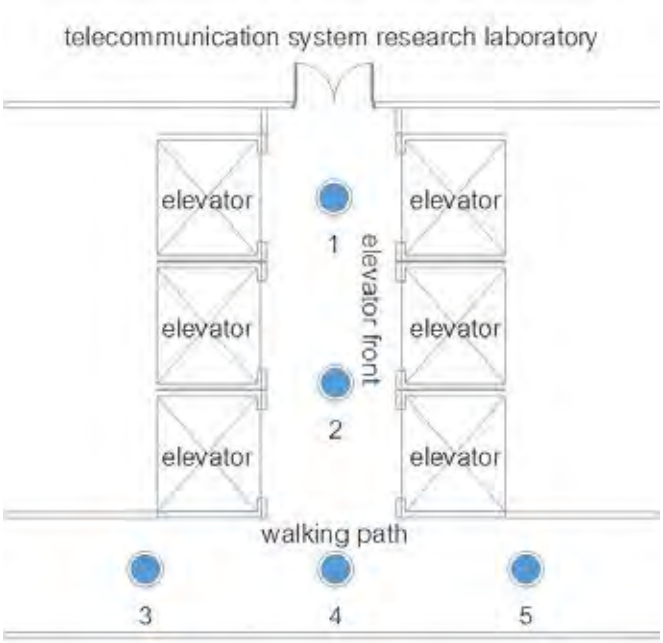
ส่วนประกอบ	รายละเอียด
เอ็นไอพีของ ETSI M2M รูปที่ 4.1(a)	ตราสินค้า Dell latitude E6400 คุณลักษณะ Intel core 2 duo 2.80 GHz x 2 processor, 2 GB RAM ระบบปฏิบัติการ Ubuntu 12.04 LTS, 64-bit โปรแกรม node.js เลขที่อยู่ไอพี 161.200.90.85
เอ็นอาร์เออาร์ของ ETSI M2M รูปที่ 4.1(b)	ตราสินค้า ASUS คุณลักษณะ Intel core 2 duo 2.66 GHz x 2 processor, 4 GB RAM ระบบปฏิบัติการ Ubuntu 12.04 LTS, 64-bit โปรแกรม node.js เลขที่อยู่ไอพี 161.200.90.78
หน่วยเก็บข้อมูลของ IEEE1888 รูปที่ 2.5(c)	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 โปรแกรม Apache เลขที่อยู่ไอพี 161.200.90.122

(a) เอ็นไอพี (b) เอ็นอาร์เออาร์

รูปที่ 4.1: ส่วนประกอบสำหรับทำหน้าที่เอ็นไอพี และเอ็นอาร์เออาร์ของ ETSI M2M

การทดสอบใช้ข้อมูลตัวรับรู้การเคลื่อนไหวของคนเพียงชนิดเดียวของตัวรับรู้วัดสภาพแวดล้อมในโครงการ CU-BEMS บริเวณทางเดินหน้าห้องปฏิบัติการวิจัยระบบโทรคมนาคมจำนวน 5 ตำแหน่ง แสดงดังรูปที่ 4.2 โดยหน่วยเก็บข้อมูลของ IEEE1888 รับข้อมูลตัวรับรู้การเคลื่อนไหวของคนแต่ละโนดด้วยคาบเวลาทุก ๆ นาที แบบไม่ประสานเวลา โดยสถานะ ON แสดงถึงมีการเคลื่อนไหวของคน และสถานะ OFF แสดงถึงไม่มีการเคลื่อนไหวของคน นอกจากนี้หน่วยเก็บข้อมูลของ IEEE1888 รับข้อมูลการเคลื่อนไหวของคนแบบไม่ประสานเวลาเมื่อมีการเปลี่ยนแปลงสถานะของการเคลื่อนไหวของคนขึ้น เช่น เมื่อตัวรับรู้การเคลื่อนไหวของคนสามารถตรวจจับการเคลื่อนไหวในขณะที่คนเดินผ่าน หน่วยเก็บข้อมูลของ IEEE1888 จะรับข้อมูลตัวรับรู้การเคลื่อนไหวของคนขณะนั้นทันที และหน่วยเก็บข้อมูลของ IEEE1888 จะรับข้อมูลตัวรับรู้การเคลื่อนไหวของคนอีกเมื่อครบคาบเวลาการรับข้อมูล ซึ่ง point id ของตัวรับรู้การเคลื่อนไหวของคนทั้ง 5 โนด สำหรับการทดสอบมีดังนี้

- ตำแหน่งที่ 1:
  http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/elevetorfront/z1/sensor1/monitor/pir
- ตำแหน่งที่ 2:
  http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/elevetorfront/z1/sensor2/monitor/pir
- ตำแหน่งที่ 3:
  http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/walkingpath/z1/sensor1/monitor/pir
- ตำแหน่งที่ 4:
  http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/walkingpath/z1/sensor2/monitor/pir
- ตำแหน่งที่ 5:
  http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/walkingpath/z1/sensor3/monitor/pir

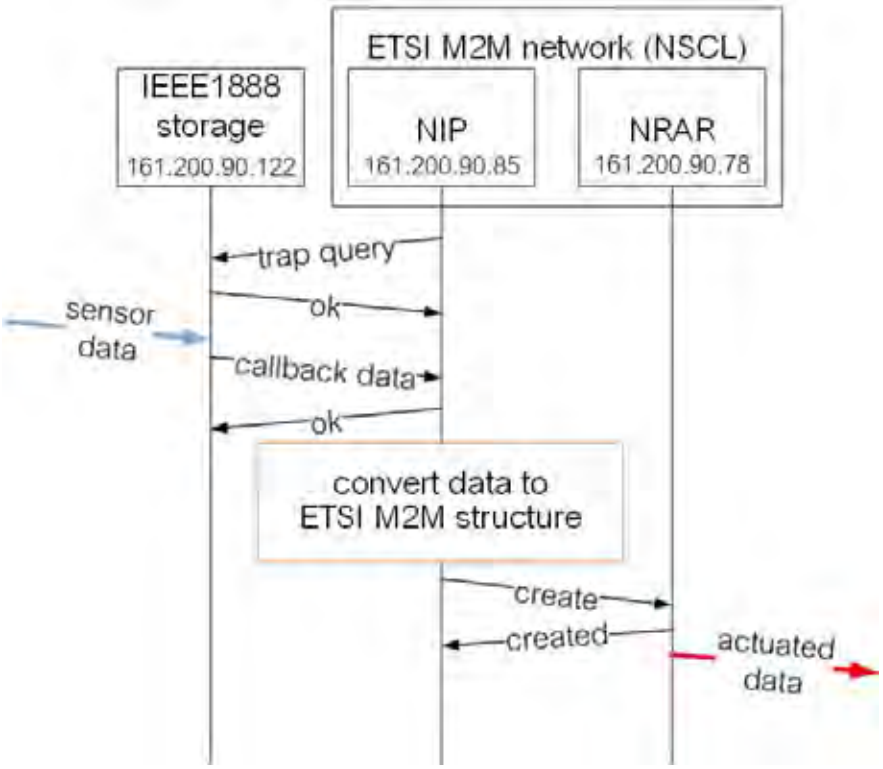


รูปที่ 4.2: แผนผังตำแหน่งตัวรับรู้วัดสภาพแวดล้อมบริเวณทางเดินหน้าห้องปฏิบัติการวิจัยระบบโทรคมนาคม

การทดสอบการประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และเอ็นอาร์เออาร์ของ ETSI M2M แบ่งเป็น 2 ขั้นตอนดังต่อไปนี้

1. ขั้นตอนการประสานข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 ไปยังเอ็นอาร์เออาร์ของ ETSI M2M แบบเวลาจริง

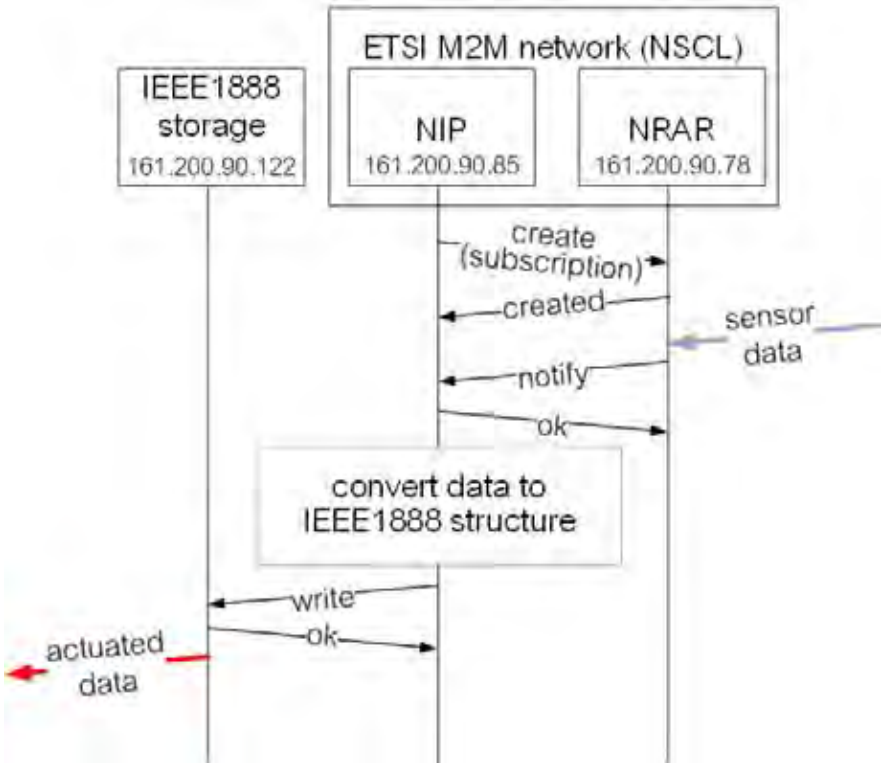
   เริ่มจากเอ็นไอพีส่งข้อมูลการร้องขอแทรปไปยังหน่วยเก็บข้อมูลของ IEEE1888 หลังจากนั้นเอ็นไอพีรับการตอบกลับ ok แสดงว่าข้อมูลที่ส่งไปถูกต้อง เมื่อหน่วยเก็บข้อมูลของ IEEE1888 รับข้อมูลตัวรับรู้ หน่วยเก็บข้อมูลของ IEEE1888 จะส่งข้อมูลตัวรับรู้มายังยังเอ็นไอพีทันที และเอ็นไอพีแจงข้อมูลรูปแบบ XML หลังจากนั้นส่งการตอบกลับ ok ไปยังหน่วยเก็บข้อมูลของ IEEE1888 แสดงว่าข้อมูลที่รับมาถูกต้อง เอ็นไอพีจะปรับข้อมูลเป็นตามโครงสร้างมาตรฐาน ETSI M2M ที่เป็นรูปแบบ JSON สุดท้ายเอ็นไอพีส่งข้อมูลไปยังเอ็นอาร์เออาร์ตามวิธี CREATE และรับการตอบกลับ created แสดงข้อมูลที่ส่งไปถูกต้องซึ่งแสดงขั้นตอนดังรูปที่ 4.3



รูปที่ 4.3: ผังเวลาการประสานข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 ไปยังเอ็นอาร์เออาร์ของ ETSI M2M แบบเวลาจริง

2. ขั้นตอนการประสานข้อมูลจากเอ็นอาร์เออาร์ของ ETSI M2M ไปยังหน่วยเก็บข้อมูลของ IEEE1888 แบบเวลาจริง

   เริ่มจากเอ็นไอพีรับข้อมูลตัวรับรู้โดยการส่งข้อมูล subscription สำหรับการติดตามข้อมูลตัวรับรู้ไปยังเอ็นอาร์เออาร์ด้วยวิธี CREATE หลังจากนั้นเอ็นไอพีรับการตอบกลับ created แสดงข้อมูลที่ส่งไปถูกต้อง เมื่อเอ็นอาร์เออาร์รับข้อมูลตัวรับรู้ที่ติดตามข้อมูลไว้ เอ็นอาร์เออาร์จะตอบกลับข้อมูลตัวรับรู้มายังเอ็นไอพีทันทีด้วยวิธี NOTIFY หลังจากนั้นเอ็นไอพีแจงข้อมูลรูปแบบ JSON และตอบกลับ ok แสดงข้อมูลที่รับมาถูกต้อง เอ็นไอพีจะปรับข้อมูลตามโครงสร้างมาตรฐาน IEEE1888 ที่เป็นรูปแบบ XML สุดท้ายเอ็นไอพีส่งข้อมูลตัวรับรู้ไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล WRITE หลังจากนั้นหน่วยเก็บข้อมูลของ IEEE1888 ตอบกลับ ok มายังเอ็นไอพีแสดงข้อมูลที่รับมาถูกต้องซึ่งแสดงขั้นตอนดังรูปที่ 4.4



รูปที่ 4.4: ผังเวลาการประสานข้อมูลจากคลังเก็บข้อมูลของ ETSI M2M ไปยังหน่วยเก็บข้อมูลของ IEEE1888 แบบเวลาจริง

4.1 การทดสอบการสื่อสารของเอ็นไอพีสำหรับการประสานข้อมูลแบบเวลาจริง

การทดสอบการสื่อสารของเอ็นไอพีในการประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และเอ็นอาร์เออาร์ของ ETSI M2M แบบเวลาจริง โดยการใช้โปรแกรม wireshark ในการตรวจวิเคราะห์ข้อมูลที่ส่งในข่ายงานบริเวณเฉพาะที่ การทดสอบใช้ข้อมูลตัวรับรู้การเคลื่อนไหวของคนจำนวน 5 โนดที่หน่วยเก็บข้อมูลของ IEEE1888 ในโครงการ CU-BEMS โดยข้อมูลตัวรับรู้การเคลื่อนไหวของคนจะถูกประสานข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 ไปยังเอ็นอาร์เออาร์ของ ETSI M2M นอกจากนี้ข้อมูลดังกล่าวในเอ็นอาร์เออาร์ของ ETSI M2M จะถูกประสานกลับไปยังหน่วยเก็บข้อมูลของ IEEE1888 เพื่อทดสอบว่ามีการประสานข้อมูลแบบเวลาจริงโดยใช้ข้อมูลชุดเดียวกัน

4.1.1 การทดสอบประสานข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 ไปยังเอ็นอาร์เออาร์ของ ETSI M2M แบบเวลาจริง

การทดสอบการสื่อสารของเอ็นไอพีในการใช้โพรโทคอล TRAP และวิธี CREATE ที่มีลำดับขั้นตอนสำหรับการประสานข้อมูลดังรูปที่ 4.3 จากรูปที่ 4.5 เป็นการร้องขอการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลจากเอ็นไอพีส่งไปยังหน่วยเก็บข้อมูลของ IEEE1888 ประกอบด้วย ไอดีการจดทะเบียน การจดทะเบียนชนิด stream อายุการร้องขอเท่ากับ 600 วินาที ยูอาร์ไอของเดตา

คอลล์แบ็กและคอนโทรลคอลล์แบ็กที่กำหนดเป็นยูอาร์ไอของเอ็นไอพีพร้อมกับ point id ของตัวรับรู้การเคลื่อนไหวของคนจำนวน 5 โนดที่ใช้สำหรับการประสานข้อมูล แต่ละโนดตัวรับรู้มีการระบุ attrName เท่ากับ time และ trap เท่ากับ changed แสดงถึงการแจ้งเตือนข้อมูลเมื่อมีการเปลี่ยนแปลงค่าของเวลาโดยมีโครงสร้างของข้อมูลเป็นแบบ XML ตามโพรโทคอล TRAP และการตอบกลับ ok พร้อมกับเหตุการณ์ที่ได้จดทะเบียนที่หน่วยเก็บข้อมูลของ IEEE1888 ส่งมายังเอ็นไอพีแสดงดังรูปที่ 4.6 และจากรูปที่ 4.7 แสดงข้อมูลที่หน่วยเก็บข้อมูลของ IEEE1888 ส่งข้อมูลตัวรับรู้มายังเอ็นไอพีประกอบด้วย เหตุการณ์ที่ได้จดทะเบียน และข้อมูลที่แจ้งเตือนประกอบด้วย point id คือ http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/elevetorfront/z1/sensor1/monitor/pir ตราเวลาเท่ากับ 2014-10-14T19:52:45.000+07:00 และค่าของข้อมูลเท่ากับ OFF แสดงว่าไม่มีการเคลื่อนไหวของคนเกิดขึ้น ณ เวลาดังกล่าวที่บริเวณตัวรับรู้ตำแหน่งที่ 1 ของแผงผังดังรูปที่ 4.2 หลังจากนั้นเอ็นไอพีตอบกลับ ok ไปยังหน่วยเก็บข้อมูลของ IEEE1888 เพื่อยืนยันข้อมูลถูกต้องแสดงดังรูปที่ 4.8

เมื่อเอ็นไอพีปรับโครงสร้างของข้อมูลที่ได้รับมานั้นให้เป็นตามโครงสร้างของข้อมูลแบบ JSON ที่กำหนดในมาตรฐาน ETSI M2M เรียบร้อยแล้ว จากรูปที่ 4.9 เป็นการส่งข้อมูลจากเอ็นไอพีไปยังเอ็นอาร์เออาร์ด้วยวิธี CREATE โดยมีการระบุยูอาร์ไอปลายทางสำหรับบันทึกข้อมูลซึ่งบันทึกในชื่อทรัพยากร application ว่า NIPA_IEEE1888 และชื่อทรัพยากร container ว่า elevetorfront_z1_sensor1_monitor_pir ข้อมูลจะอยู่ในรูปแบบ JSON โดยข้อมูลตัวรับรู้ระบุที่คีย์ชื่อ $t ซึ่งเข้ารหัสเป็นแบบ base64 ที่เป็นอ็อบเจกต์อยู่ในคีย์ชื่อ contentInstance และการตอบกลับ created จากเอ็นอาร์เออาร์มายังเอ็นไอพีดังรูปที่ 4.10 ซึ่งแสดงยูอาร์ไอของข้อมูลที่บันทึกเรียบร้อยแล้วข้อมูลอยู่ในทรัพยากร contentInstance ที่ชื่อว่า contentInstance4866819 และจากรูปที่ 4.11 แสดงข้อมูลที่บันทึกในทรัพยากร containers ที่เอ็นอาร์เออาร์ที่ได้จากการประสานข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 ประกอบด้วย ชื่อทรัพยากร container อ้างอิงถึงตัวรับรู้การเคลื่อนไหวของคนแต่ละตำแหน่งที่ระบุในคีย์ชื่อ id และแสดงพาท (path) ที่เข้าถึงแต่ละ container ที่ระบุในคีย์ $t ซึ่งมี container ทั้งหมด 5 container เช่นเดียวกับตัวรับรู้ที่มีทั้งหมด 5 โนด และในแต่ละทรัพยากร container มีการบันทึกค่าตัวรับรู้การเคลื่อนไหวของคนในทรัพยากร contentInstances แบบเวลาจริงซึ่งค่าของข้อมูลระบุที่คีย์ $t ที่อยู่ในรูปแบบ base64 แสดงดังรูปที่ 4.12

```
<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="9eed9de4-1c48-4b08-a41d-dac067fc1c0d"
              type="stream"
              ttl="600"
              callbackData="http://161.200.90.85"
              callbackControl="http://161.200.90.85">
            <key
                id="http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/elevetorfront/z1/sensor1/monitor/pir"
                attrName="time"
                trap="changed"/>
            <key
                id="http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/elevetorfront/z1/sensor2/monitor/pir"
                attrName="time"
                trap="changed"/>
            <key
                id="http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/walkingpath/z1/sensor1/monitor/pir"
                attrName="time"
                trap="changed"/>
            <key
                id="http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/walkingpath/z1/sensor2/monitor/pir"
                attrName="time"
                trap="changed"/>
            <key
                id="http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/walkingpath/z1/sensor3/monitor/pir"
                attrName="time"
                trap="changed"/>
          </query>
        </header>
      </transport>
    </ns2:queryRQ>
  </soapenv:Body>
</soapenv:Envelope>
```

trap query message

รูปที่ 4.5: การร้องขอการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลจากเอ็นไอพีส่งไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล TRAP

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns2:queryRS
        xmlns:ns2="http://soap.fiap.org/">
      <transport
          xmlns="http://gutp.jp/fiap/2009/11/">
        <header>
          <OK/>
          <query
              id="9eed9de4-1c48-4b08-a41d-dac067fc1c0d"
              type="stream"
              ttl="600"
              callbackData="http://161.200.90.85"
              callbackControl="http://161.200.90.85">
            <key
                id="http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/elevetorfront/z1/sensor1/monitor/pir"
                attrName="time"
                trap="changed"/>
            <key />
            <key />
            <key />
            <key />
          </query>
        </header>
        <body/>
      </transport>
    </ns2:queryRS>
  </soapenv:Body>
</soapenv:Envelope>
```

ok (trap query) message

รูปที่ 4.6: การตอบกลับการร้องขอการจดทะเบียนจากหน่วยเก็บข้อมูลของ IEEE1888 ไปยังเอ็นไอพีตามโพรโทคอล TRAP

```
<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/">
    <header>
     <query
       id="9eed9de4-1c48-4b08-a41d-dac067fc1c0d"
       type="stream"
       ttl="600"
       callbackData="http://161.200.90.85"
       callbackControl="http://161.200.90.85">
      <key
        id="http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/elevetorfront/z1/sensor1/monitor/pir"
        attrName="time"
        trap="changed"/>
      <key
      <key
      <key
      <key
       </query>
     </header>
    <body>
     <point
       id="http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/elevetorfront/z1/sensor1/monitor/pir">
      <value
        time="2014-10-14T19:52:45.000+07:00">
        OFF
       </value>
     </point>
    </body>
   </transport>
  </ns2:dataRQ>
 </soapenv:Body>
</soapenv:Envelope>
```

callback message

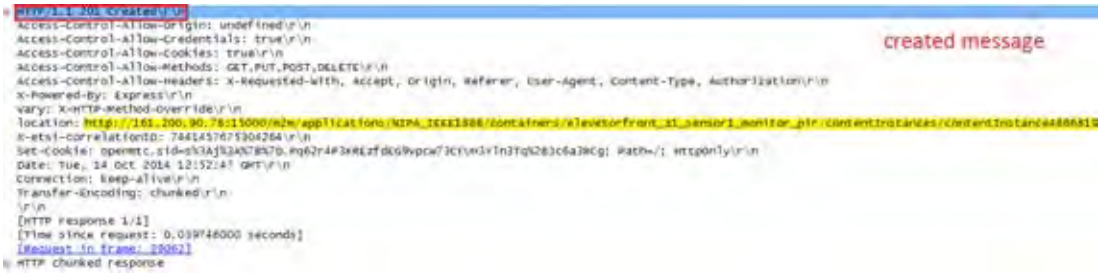
รูปที่ 4.7: การส่งข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 ไปยังเอ็นไอพีตามโพรโทคอล TRAP

```
<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>
```

ok (callback) message

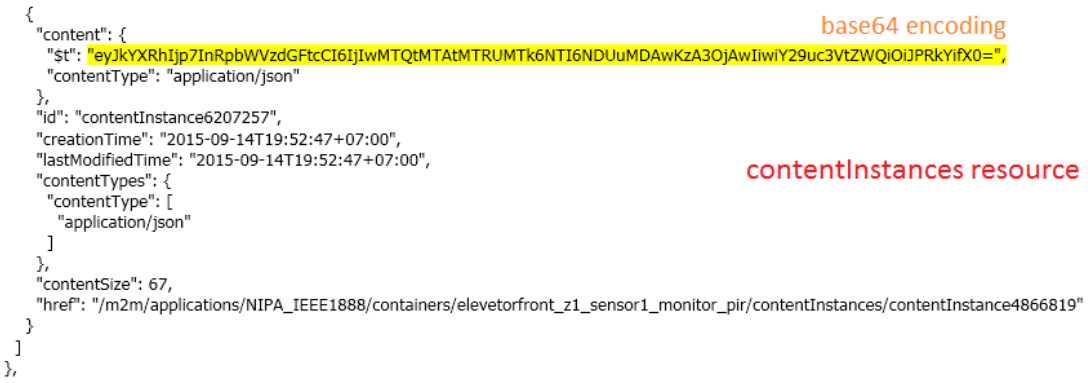
รูปที่ 4.8: การตอบกลับจากเอ็นไอพีไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล TRAP

รูปที่ 4.9: การส่งข้อมูลจากเอ็นไอพีไปยังเอ็นอาร์เออาร์ตามวิธี CREATE



รูปที่ 4.10: การตอบกลับจากเอ็นอาร์เออาร์ไปยังเอ็นไอพีตามวิธี CREATE

รูปที่ 4.11: ข้อมูลที่บันทึกในทรัพยากร containers ที่เอ็นอาร์เออาร์




```
{
  "content": {
    "$t": "eyJkYXRhIjp7InRpbWVzdGFtcCI6IjIwMTQtMTAtMTRUMTk6NTI6NDUuMDAwKzA3OjAwIiwiY29uc3VtZWQiOiJPRkYifX0=",
    "contentType": "application/json"
  },
  "id": "contentInstance6207257",
  "creationTime": "2015-09-14T19:52:47+07:00",
  "lastModifiedTime": "2015-09-14T19:52:47+07:00",
  "contentTypes": {
    "contentType": [
      "application/json"
    ]
  },
  "contentSize": 67,
  "href": "/m2m/applications/NIPA_IEEE1888/containers/elevetorfront_z1_sensor1_monitor_pir/contentInstances/contentInstance4866819"
 }
]
},
```

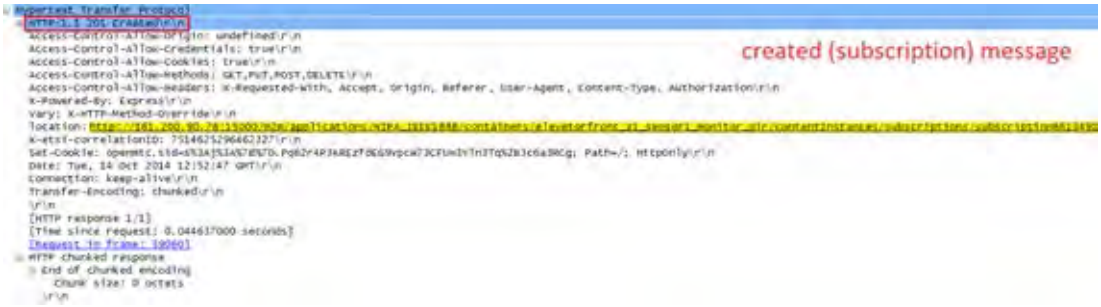

รูปที่ 4.12: ค่าตัวรับรู้การเคลื่อนไหวของคนในทรัพยากร contentInstances ที่เอ็นอาร์เออาร์

4.1.2 การทดสอบประสานข้อมูลจากเอ็นอาร์เออาร์ของ ETSI M2M ไปยังหน่วยเก็บข้อมูลของ IEEE1888 แบบเวลาจริง

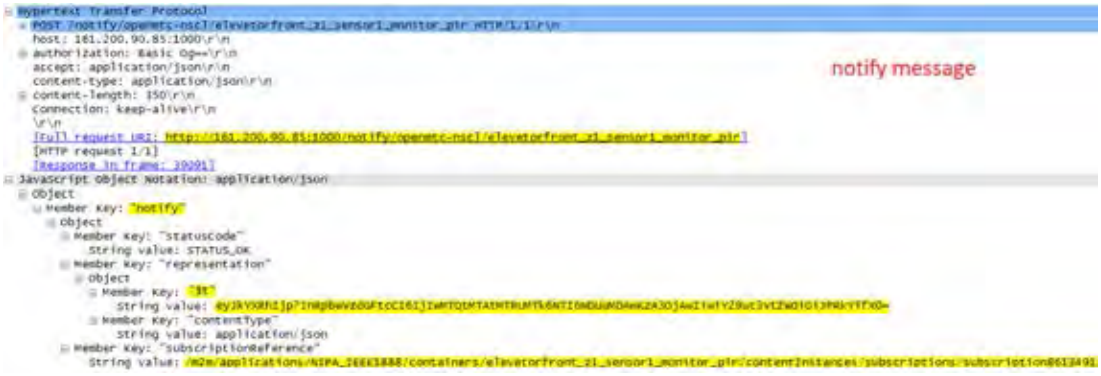
การทดสอบการสื่อสารของเอ็นไอพีในการใช้โพรโทคอล WRITE, วิธี CREATE (subscription) และวิธี NOTIFY ที่มีลำดับขั้นตอนสำหรับการประสานข้อมูลดังรูปที่ 4.4 จากรูปที่ 4.13 เป็นการส่งติดตามข้อมูลจากเอ็นไอพีที่เอ็นอาร์เออาร์ โดยมีการระบุยูอาร์ไอปลายทางสำหรับบันทึกการติดตามข้อมูลซึ่งบันทึกในทรัพยากร application ที่ชื่อว่า NIPA_IEEE1888 และทรัพยากร container ที่ชื่อว่า elevetorfront_z1_sensor1_monitor_pir ข้อมูลจะอยู่ในรูปแบบ JSON โดยระบุยูอาร์ไอสำหรับรับการแจ้งเตือนข้อมูลคือเอ็นไอพีที่คีย์ชื่อว่า contact เป็นอ็อบเจกต์อยู่ในคีย์ชื่อ subscription โดยมีการระบุ latest/content หมายถึงส่งข้อมูลล่าสุดเท่านั้น และเอ็นอาร์เออาร์ตอบกลับ created พร้อมกับยูอาร์ไอของทรัพยากรที่ติดตามข้อมูลที่มีการสร้างทรัพยากร subscription ที่ชื่อว่า subscription8613491 ดังรูปที่ 4.14 และจากรูปที่ 4.15 การแจ้งเตือนข้อมูลจากเอ็นอาร์เออาร์ส่งไปยังเอ็นไอพีด้วยวิธี NOTIFY มีการระบุยูอาร์ไอรับการแจ้งเตือนข้อมูลซึ่งคือเอ็นไอพี ข้อมูลจะอยู่ในรูปแบบ JSON โดยข้อมูลตัวรับรู้ระบุที่คีย์ชื่อ $t ซึ่งเข้ารหัสเป็นแบบ base64 ที่อยู่ในอ็อบเจกต์ที่มีคีย์ชื่อ notify โดยภายในอ็อบเจกต์มีระบุพาทที่อ้างอิงถึงการติดตามข้อมูลที่สร้างไว้ และเอ็นไอพีตอบกลับ ok ยืนยันการรับข้อมูลแสดงดังรูปที่ 4.16

เมื่อเอ็นไอพีปรับโครงสร้างของข้อมูลที่ได้รับมานั้นให้เป็นตามโครงสร้างของข้อมูลแบบ XML ที่กำหนดในมาตรฐาน IEEE1888 เรียบร้อยแล้ว จากรูปที่ 4.17 แสดงข้อมูลที่เอ็นไอพีส่งไปยังหน่วยเก็บข้อมูลของ IEEE1888 ด้วยโพรโทคอล WRITE ประกอบด้วย point id คือ http://napat.test.chula.ac.th/applications/NIP_IEEE1888/containers/elevetorfront_z1_sensor1_monitor_pir ตราเวลาเท่ากับ 2014-10-14T19:52:40+07:00 และค่าของข้อมูลเท่ากับ OFF และการตอบกลับ ok จากหน่วยเก็บข้อมูลของ IEEE1888 มายังเอ็นไอพียืนยันการส่งข้อมูลถูกต้องแสดงดังรูปที่ 4.18 สุดท้ายจากรูปที่ 4.19 แสดงข้อมูลที่บันทึกในหน่วยเก็บข้อมูลของ IEEE1888 ประกอบด้วย point id จำนวน 5 โนดอ้างอิงถึงตัวรับรู้การเคลื่อนไหวของคน 5 ตำแหน่ง ซึ่งแต่ละโนดประกอบไปด้วย ตราเวลา และค่าของตัวรับรู้ที่ได้จากการประสานข้อมูลจากเอ็นอาร์เออาร์แบบเวลาจริง

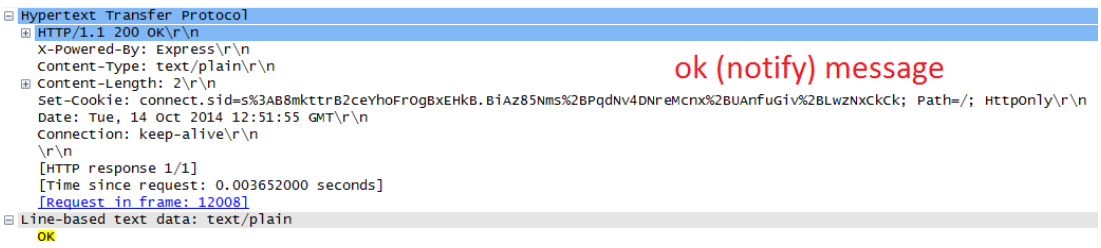
รูปที่ 4.13: การส่งติดตามข้อมูลจากเอ็นไอพีไปยังเอ็นอาร์เออาร์ตามวิธี CREATE



รูปที่ 4.14: การตอบกลับการติดตามข้อมูลจากเอ็นอาร์เออาร์ไปยังเอ็นไอพีตามวิธี CREATE



รูปที่ 4.15: การส่งข้อมูลจากเอ็นอาร์เออาร์ไปยังเอ็นไอพีตามวิธี NOTIFY



รูปที่ 4.16: การตอบกลับจากเอ็นไอพีไปยังเอ็นอาร์เออาร์ตามวิธี NOTIFY

```
<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>
          <point
              id="http://napat.test.chula.ac.th/applications/NIP_IEEE1888/containers/elevetorfront_z1_sensor1_monitor_pir">
            <value
                time="2014-10-14T19:52:40+07:00">
              OFF
            </value>
          </point>
        </body>
      </transport>
    </ns2:dataRQ>
  </soapenv:Body>
</soapenv:Envelope>
```

write message

รูปที่ 4.17: การส่งข้อมูลจากเอ็นไอพีไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล WRITE

```
<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>
```

ok (write) message

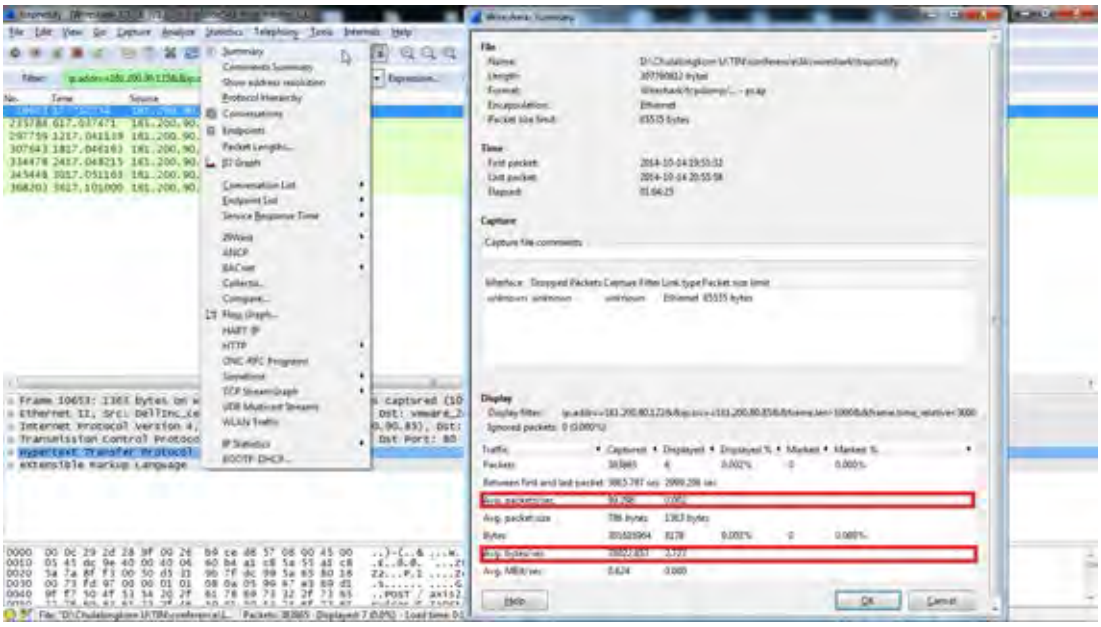
รูปที่ 4.18: การตอบกลับจากหน่วยเก็บข้อมูลของ IEEE1888 ไปยังเอ็นไอพีตามโพรโทคอล WRITE

Point ID	Time	Value
http://napat.test.chula.ac.th/applications/NIP_IEEE1888/containers/elevetorfront_z1_sensor1_monitor_pir	2014-10-14T 19:44:44.000+07:00	OFF
http://napat.test.chula.ac.th/applications/NIP_IEEE1888/containers/elevetorfront_z1_sensor2_monitor_pir	2014-10-14T 19:44:50.000+07:00	OFF
http://napat.test.chula.ac.th/applications/NIP_IEEE1888/containers/walkingpath_z1_sensor1_monitor_pir	2014-10-14T 19:44:42.000+07:00	OFF
http://napat.test.chula.ac.th/applications/NIP_IEEE1888/containers/walkingpath_z1_sensor2_monitor_pir	2014-10-14T 19:44:49.000+07:00	ON
http://napat.test.chula.ac.th/applications/NIP_IEEE1888/containers/walkingpath_z1_sensor3_monitor_pir	2014-10-14T 19:44:08.000+07:00	ON

รูปที่ 4.19: ข้อมูลตัวรับรู้การเคลื่อนไหวของคนที่หน่วยเก็บข้อมูลของ IEEE1888

4.2 การทดสอบปริมาณงานสำหรับการประสานข้อมูลแบบคาบเวลา [21] [22] และแบบเวลาจริง

การทดสอบปริมาณงานสำหรับการประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M แบบเวลาจริง โดยเวลาทดสอบ 1 ชั่วโมง ช่วงเวลา 18.30 - 19.30 นาฬิกา เพื่อเปรียบเทียบกับการประสานข้อมูลแบบคาบเวลาโดยอาศัยโพรโทคอล FETCH และวิธี RETRIEVE ในงานวิจัย [21] และ [22] ที่มีคาบเวลาในการประสานข้อมูลทุก ๆ นาที ใช้เวลาทดสอบ 1 ชั่วโมงในช่วงเวลาที่ต่างกันกับการประสานข้อมูลแบบเวลาจริงเพื่อไม่เกิดการทับซ้อนข้อมูลระหว่างแบบคาบเวลา และแบบเวลาจริงสำหรับการหาปริมาณงานเฉลี่ย การทดสอบใช้ข้อมูลตัวรับรู้การเคลื่อนไหวของคนจำนวน 5 โนดของโครงการ CU-BEMS ดังรูปที่ 4.2 ที่บันทึกในหน่วยเก็บข้อมูลของ IEEE1888 การทดสอบใช้โปรแกรม wireshark บนแล็ปท็อปคอมพิวเตอร์ที่ทำหน้าที่เน็ทเวิร์คอินเตอร์เวิร์กกิงพร็อกซี ในการตรวจสอบปริมาณงานเฉลี่ยของสาร (message) ในการประสานข้อมูลตามขั้นตอนดังรูปที่ 4.3 และ 4.4 เริ่มจากการใช้ตัวกรอง (filter) ในการกรองข้อมูลของสารที่ต้องการวิเคราะห์จากการกำหนด ip.src สำหรับเลขที่อยู่ไอพีต้นทาง, ip.dst สำหรับเลขที่อยู่ไอพีปลายทาง, frame.len สำหรับความยาวเฟรม, frame.time_relative สำหรับเวลา เป็นต้น หลังจากนั้นเข้าหน้าต่างที่เรียกว่า ข้อสรุป (summary) จะแสดงปริมาณการใช้ (traffic) ที่เกิดขึ้น ดังรูปที่ 4.20 ซึ่งเป็นตัวอย่างการหาปริมาณงานเฉลี่ยของการส่งสารการร้องขอแทรปที่มีคาบเวลาทุก ๆ 10 นาที โดยนำปริมาณงานเฉลี่ยในแถบ avg. packets/sec และ avg. bytes/sec มาวิเคราะห์ และเปรียบเทียบกับสารรูปแบบอื่น ๆ



รูปที่ 4.20: ข้อสรุปปริมาณงานเฉลี่ยจากโปรแกรม wireshark

ผลการทดสอบการประสานข้อมูลแบบคาบเวลาเปรียบเทียบกับการประสานข้อมูลแบบเวลาจริงในงานวิจัยที่นำเสนอ จากตารางที่ 4.2 และ 4.3 แสดงปริมาณงานเปรียบเทียบการประสานข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 ไปยังเอ็นอาร์เออาร์ของ ETSI M2M ระหว่างแบบคาบเวลา และ

แบบเวลาจริง พบว่าในส่วนของการตอบกลับข้อมูลตามโพรโทคอล TRAP มีการแจ้งเตือนข้อมูลตัวรับรู้พร้อมกับเหตุการณ์ที่ได้จดทะเบียนแจ้งเตือนข้อมูลไว้ที่ส่งแบบคาบเวลา และแบบไม่ประสานเวลาทำให้มีปริมาณงานเฉลี่ยสูงกว่าการดึงข้อมูลตามโพรโทคอล FETCH ที่มีการรับข้อมูลตัวรับรู้ทุกโนดพร้อมกันตามที่ได้ทำการร้องขอแบบคาบเวลาทุก ๆ นาที ในส่วนของตารางที่ 4.4 และ 4.5 แสดงปริมาณงานเปรียบเทียบการประสานข้อมูลจากเอ็นอาร์เออาร์ของ ETSI M2M ไปยังหน่วยเก็บข้อมูลของ IEEE1888 ระหว่างแบบคาบเวลา และแบบเวลาจริง พบว่าในส่วนของการรับข้อมูลตามวิธี RETRIEVE มีการดึงข้อมูลในส่วนของทรัพยากร application, container และ contentInstance ทำให้มีปริมาณงานเฉลี่ยสูงกว่าการแจ้งเตือนข้อมูลตามวิธี NOTIFY ที่มีการแจ้งเตือนข้อมูลเฉพาะทรัพยากร contentInstance ล่าสุดเท่านั้น นอกจากนี้พบว่าการแจ้งเตือนตามวิธี NOTIFY มีปริมาณงานเฉลี่ยที่ต่ำกว่าการตอบกลับข้อมูลตามโพรโทคอล TRAP โดยได้ภาระปริมาณงานเฉลี่ยทั้งหมดเท่ากับ 0.30 - 0.53 กิโลบิตต่อวินาที และ 0.56 - 0.60 กิโลบิตต่อวินาทีต่อโนดตัวรับรู้ สำหรับการประสานข้อมูลแบบคาบเวลา และแบบเวลาจริง ตามลำดับ ซึ่งสามารถประมาณการภาระปริมาณงานเฉลี่ยทั้งหมดในการประสานข้อมูลของตัวรับรู้ทั้งหมด 688 โนดของโครงการ CU-BEMS เท่ากับ 0.36 และ 0.41 เมกะบิตต่อวินาทีสำหรับการประสานข้อมูลทั้ง 2 กรณี ตามลำดับ ซึ่งอยู่ในช่วงที่ยอมรับได้ ดังนั้นไม่ต้องกังวลมากนักเกี่ยวกับการใช้ปริมาณงานในการประสานข้อมูลของทั้ง 2 แบบ

ตารางที่ 4.2: ปริมาณงานในการประสานข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 ไปยังคลังเก็บข้อมูลของ ETSI M2M แบบคาบเวลา

สาร	กลุ่มข้อมูลเฉลี่ยต่อวินาที	ไบต์เฉลี่ยต่อวินาที
fetch	0.017	21.930
ok (fetch)	0.035	40.508
create	0.170	46.404
created	0.085	77.618
ผลรวม	**0.307**	**186.46**

ตารางที่ 4.3: ปริมาณงานในการประสานข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 ไปยังคลังเก็บข้อมูลของ ETSI M2M แบบเวลาจริง

สาร	กลุ่มข้อมูลเฉลี่ยต่อวินาที	ไบต์เฉลี่ยต่อวินาที
trap query	0.002	2.727
ok (trap query)	0.006	3.421
callback	0.195	154.442
ok (callback)	0.098	53.862
create	0.197	53.642
created	0.102	83.039
ผลรวม	**0.600**	**351.133**

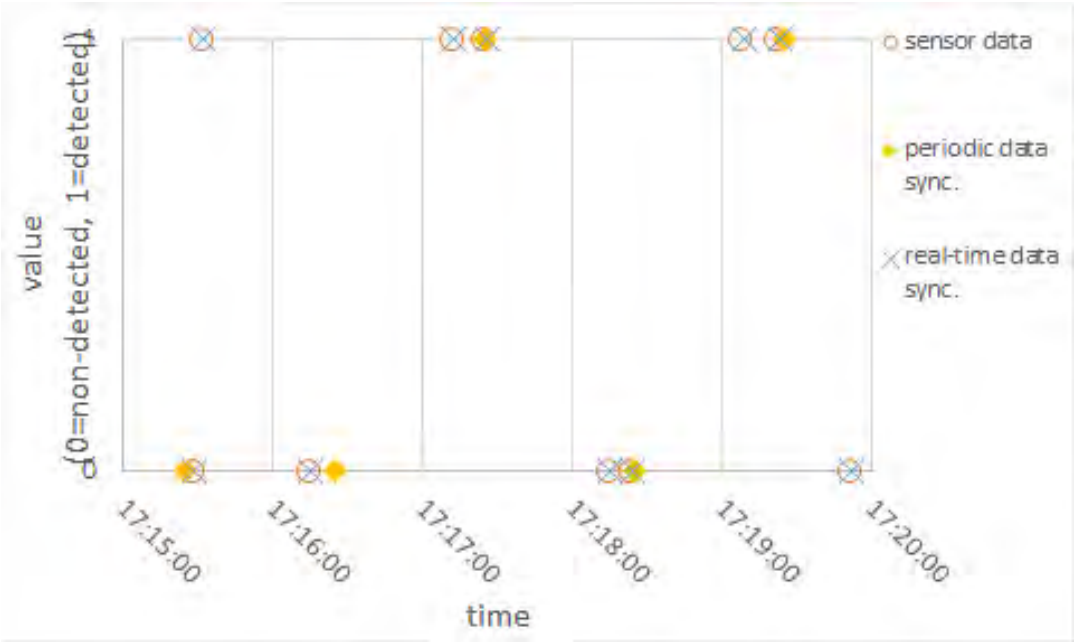
ตารางที่ 4.4: ปริมาณงานในการประสานข้อมูลจากคลังเก็บข้อมูลของ ETSI M2M ไปยังหน่วยเก็บข้อมูลของ IEEE1888 แบบคาบเวลา

สาร	**กลุ่มข้อมูลเฉลี่ยต่อวินาที**	**ไบต์เฉลี่ยต่อวินาที**
retrieve	0.117	30.695
ok (retrieve)	0.119	163.916
write	0.084	60.590
ok (write)	0.212	73.789
ผลรวม	**0.532**	**328.990**

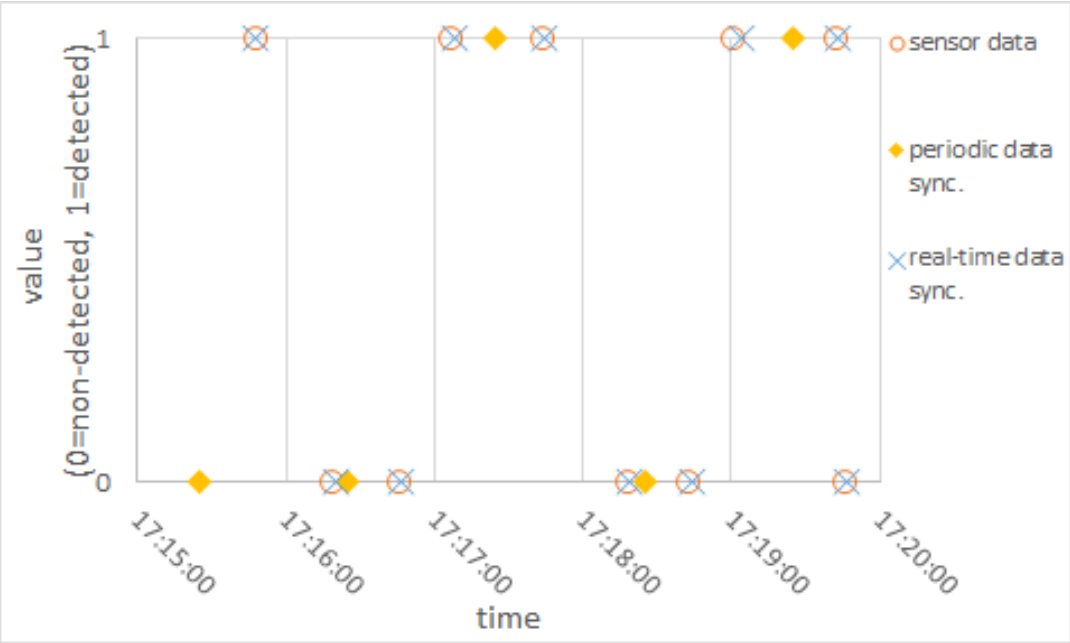
ตารางที่ 4.5: ปริมาณงานในการประสานข้อมูลจากคลังเก็บข้อมูลของ ETSI M2M ไปยังหน่วยเก็บข้อมูลของ IEEE1888 แบบเวลาจริง

สาร	**กลุ่มข้อมูลเฉลี่ยต่อวินาที**	**ไบต์เฉลี่ยต่อวินาที**
create (subscription)	0.128	26.643
created (subscription)	0.128	103.711
notify	0.201	72.597
ok (notify)	0.086	28.721
write	0.098	53.862
ok (write)	0.268	88.094
ผลรวม	**0.909**	**373.628**

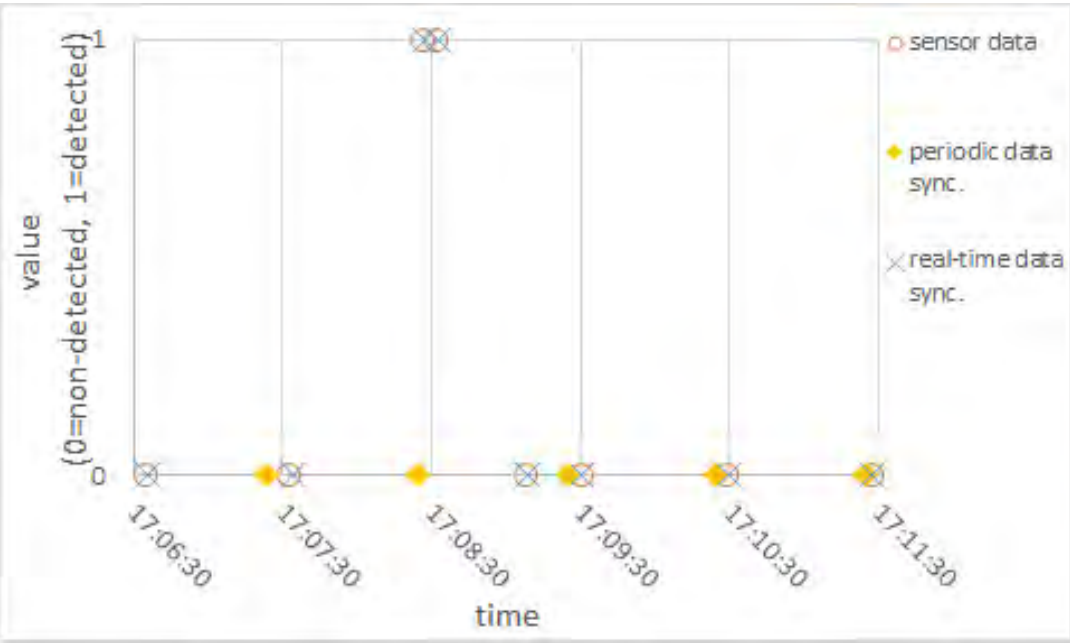
นอกจากนี้จากรูปที่ 4.21 - 4.25 แสดงการเปรียบเทียบข้อมูลตัวรับรู้การเคลื่อนไหวของคนจำนวน 5 โนดที่หน่วยเก็บข้อมูลของ IEEE1888 ระหว่างการประสานข้อมูลแบบคาบเวลา และแบบเวลาจริงซึ่งใช้ช่วงเวลาทดสอบเดียวกัน พบว่าข้อมูลที่ได้จากการประสานข้อมูลแบบเวลาจริงมีการปรับข้อมูลตามข้อมูลตัวรับรู้การเคลื่อนไหวของคนในหน่วยเก็บข้อมูลของ IEEE1888 ทำให้ได้ข้อมูลที่แม่นยำ ในส่วนของการประสานข้อมูลแบบคาบเวลาทุกนาทีพบว่ามีการปรับข้อมูลแบบคาบเวลาทำให้ได้ข้อมูลล่าช้า หรือ ไม่ได้ข้อมูลในบางค่าจึงเป็นข้อจำกัดในการนำข้อมูลไปใช้สำหรับการเฝ้าสังเกตข้อมูลตัวรับรู้แบบเวลาจริง การแจ้งเตือนฉุกเฉิน การควบคุมอุปกรณ์ เป็นต้น



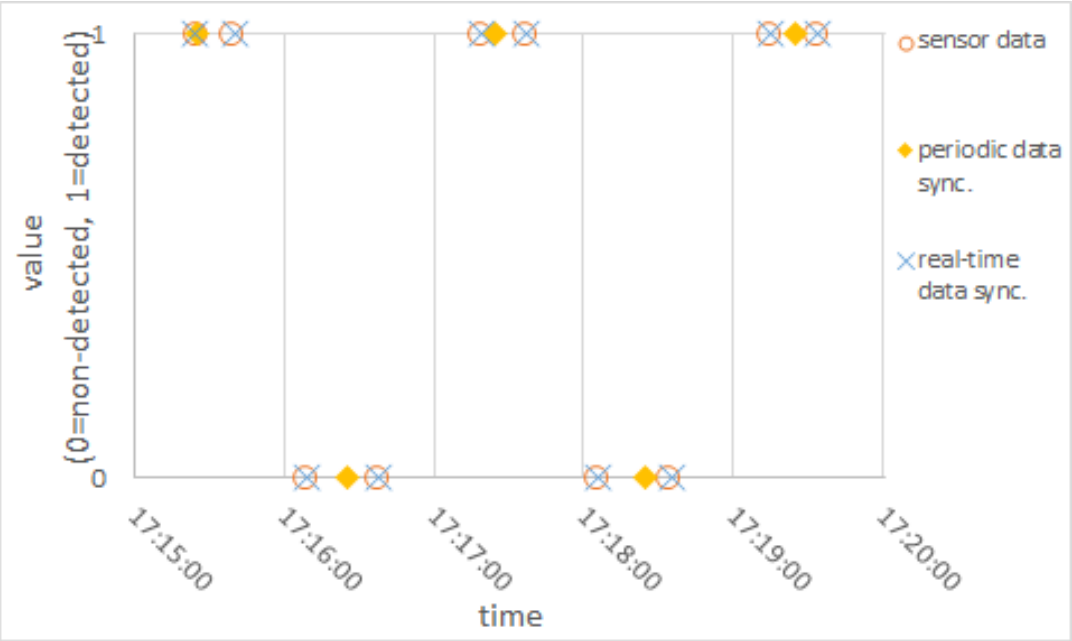
รูปที่ 4.21: เปรียบเทียบข้อมูลจากการประสานข้อมูลระหว่างแบบคาบเวลา และแบบเวลาจริงด้วยข้อมูลตัวรับรู้การเคลื่อนไหวของคนตำแหน่งที่ 1



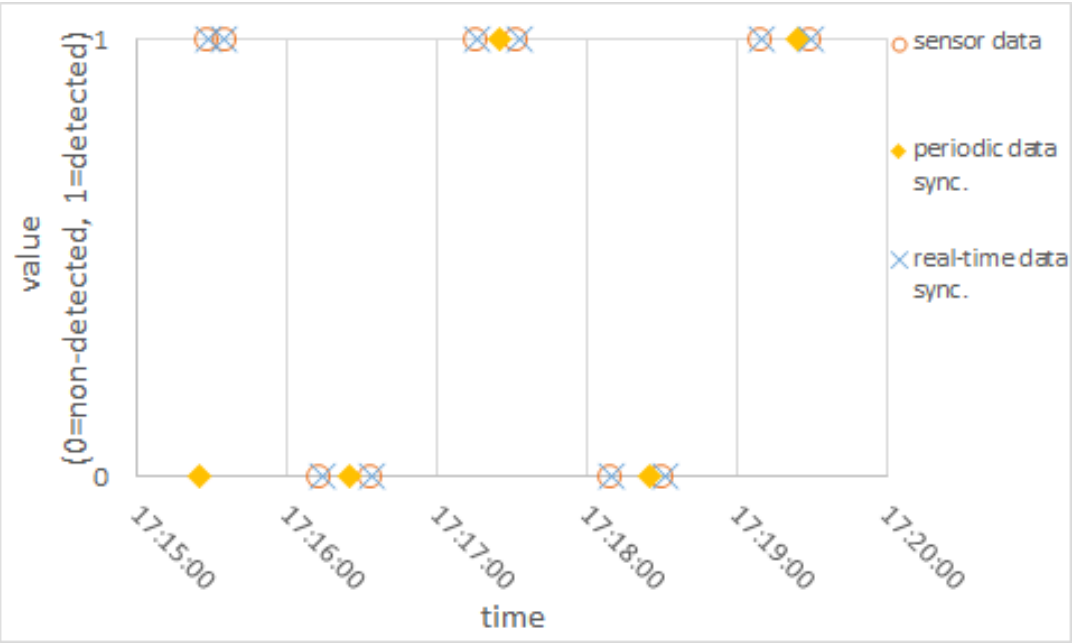
รูปที่ 4.22: เปรียบเทียบข้อมูลจากการประสานข้อมูลระหว่างแบบคาบเวลา และแบบเวลาจริงด้วยข้อมูลตัวรับรู้การเคลื่อนไหวของคนตำแหน่งที่ 2



รูปที่ 4.23: เปรียบเทียบข้อมูลจากการประสานข้อมูลระหว่างแบบคาบเวลา และแบบเวลาจริงด้วยข้อมูลตัวรับรู้การเคลื่อนไหวของคนตำแหน่งที่ 3



รูปที่ 4.24: เปรียบเทียบข้อมูลจากการประสานข้อมูลระหว่างแบบคาบเวลา และแบบเวลาจริงด้วยข้อมูลตัวรับรู้การเคลื่อนไหวของคนตำแหน่งที่ 4



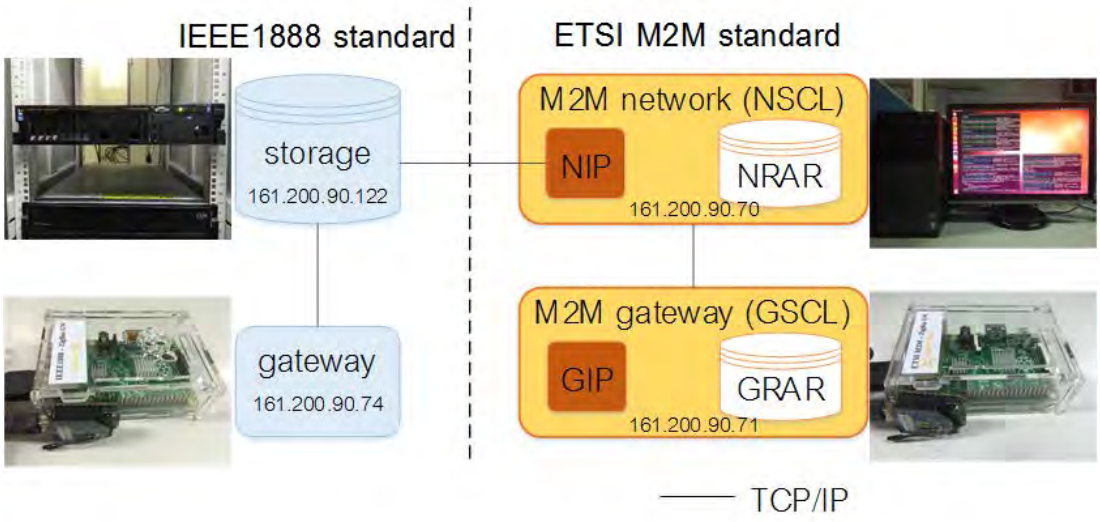
รูปที่ 4.25: เปรียบเทียบข้อมูลจากการประสานข้อมูลระหว่างแบบคาบเวลา และแบบเวลาจริงด้วยข้อมูลตัวรับรู้การเคลื่อนไหวของคนตำแหน่งที่ 5

บทที่ 5

การทดสอบระบบการทำงานร่วมกันแบบเวลาจริงระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M สำหรับระบบจัดการพลังงานไฟฟ้าภายในอาคาร

ในบทที่ผ่านมาได้กล่าวถึงโครงสร้างและการทำงานของระบบ รวมถึงการทดสอบเบื้องต้นสำหรับการประเมินรูปแบบการสื่อสาร และปริมาณงานที่ใช้ในการประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M เพื่อเตรียมสำหรับการพัฒนาระบบการทำงานร่วมกันแบบเวลาจริงในส่วนของเกตเวย์ต่อไปเพื่อใช้ในกรณีที่ตัวรับรู้ และตัวกระตุ้นตามมาตรฐานที่ต่างกันสามารถทำงานร่วมกันได้ บทนี้กล่าวถึงการทดสอบระบบการทำงานร่วมกันแบบเวลาจริงระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M เพื่อประเมินสมรรถนะของระบบ และการทำงานร่วมกับโครงการ CU-BEMS

การทดสอบระบบโดยการถ่ายโอนข้อมูลแบบเวลาจริงระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M ตามผังเวลาแสดงดังรูปที่ 4.3 - 4.4 โดยโครงสร้างของระบบ และเลขที่อยู่ไอพีสำหรับการทดสอบระบบการทำงานร่วมกันแบบเวลาจริงแสดงดังรูปที่ 5.1 ที่มีรายละเอียดส่วนประกอบของระบบดังตารางที่ 3.1

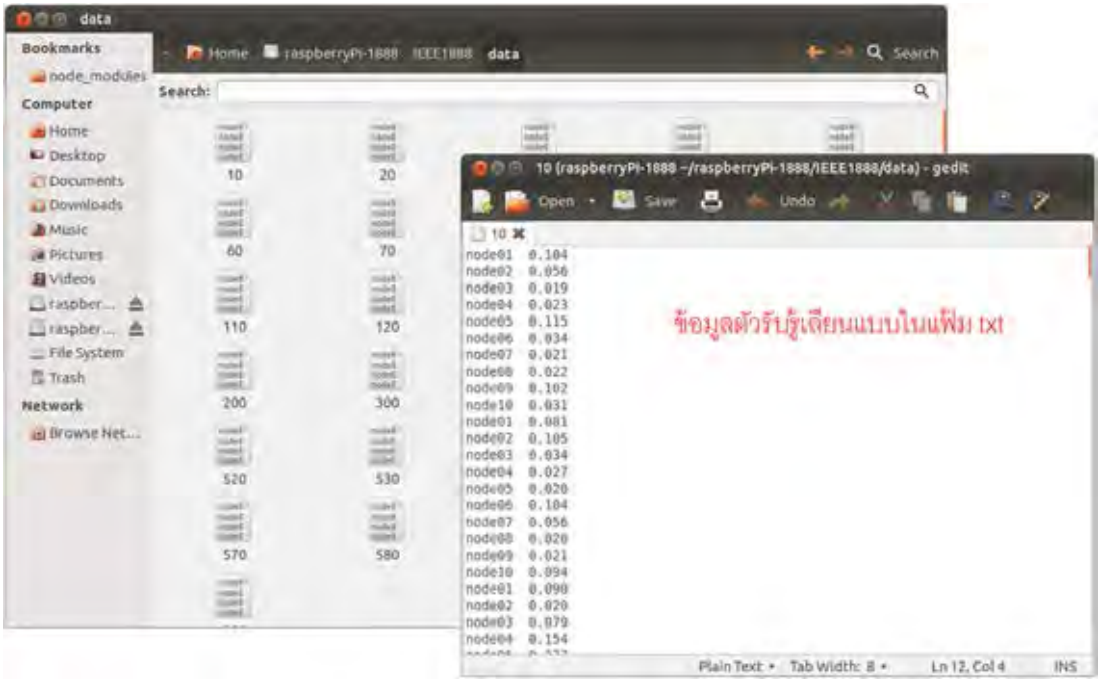


รูปที่ 5.1: โครงสร้างของระบบ และเลขที่อยู่ไอพีสำหรับการทดสอบระบบการทำงานร่วมกันแบบเวลาจริง

การทดสอบระบบการทำงานร่วมกันแบบเวลาจริงระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M ได้แบ่งการทดสอบเป็น 2 กรณี กรณีแรกเป็นการทดสอบด้วยข้อมูลตัวรับรู้เลียนแบบ (emulated sensor data) เพื่อหาแนวโน้มค่าตัวชี้วัดในการทดสอบแทนข้อมูลที่ได้จากตัวรับรู้จริงซึ่งมีข้อจำกัดในส่วนของจำนวนตัวรับรู้ ในกรณีที่ 2 เป็นการทดสอบด้วยข้อมูลตัวรับรู้ในโครงการ CU-BEMS เพื่อประเมินสมรรถนะของระบบ และเปรียบเทียบความแตกต่างกับการทดสอบด้วยข้อมูลตัวรับรู้เลียนแบบ ซึ่งตัวชี้วัดในการทดสอบประกอบด้วย เพย์โหลด (payload), แบนด์วิดท์ (bandwidth), การใช้ซีพียู (CPU usage) และเวลาประวิง (delay) มีรายละเอียดของข้อมูลที่ใช้ทดสอบดังนี้

1. **ข้อมูลตัวรับรู้เลียนแบบ**

   ชุดข้อมูลตัวรับรู้เลียนแบบในแฟ้มข้อความ (txt) ที่กำหนดขึ้นประกอบด้วย ชื่อโนดตัวรับรู้ และค่าตัวรับรู้ ดังรูปที่ 5.2 โปรแกรมที่เกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M อ่านข้อมูลในแฟ้มข้อความทีละบรรทัดซึ่งสามารถกำหนดอัตราการอ่านข้อมูลได้ หลังจากนั้นเกตเวย์ของ IEEE1888 จะปรับข้อมูลตามมาตรฐาน IEEE1888 และเกตเวย์ของ ETSI M2M จะปรับข้อมูลตามมาตรฐาน ETSI M2M



รูปที่ 5.2: ตัวอย่างข้อมูลตัวรับรู้เลียนแบบในแฟ้มข้อความ

(a) ตัวอย่างข้อมูลตัวรับรู้เลียนแบบที่เกตเวย์ของ IEEE1888 ส่งไปยังหน่วยเก็บข้อมูลของ IEEE1888 ประกอบด้วย ชื่อ point id ตราเวลา และค่าของตัวรับรู้ ที่มีขนาดเท่ากับ 99 ไบต์

- http://napat.test.chula.ac.th/IEEE1888_ETSIM2M/experiment/node01, 2015-09-05T14:01:52.000+07:00, 0.639

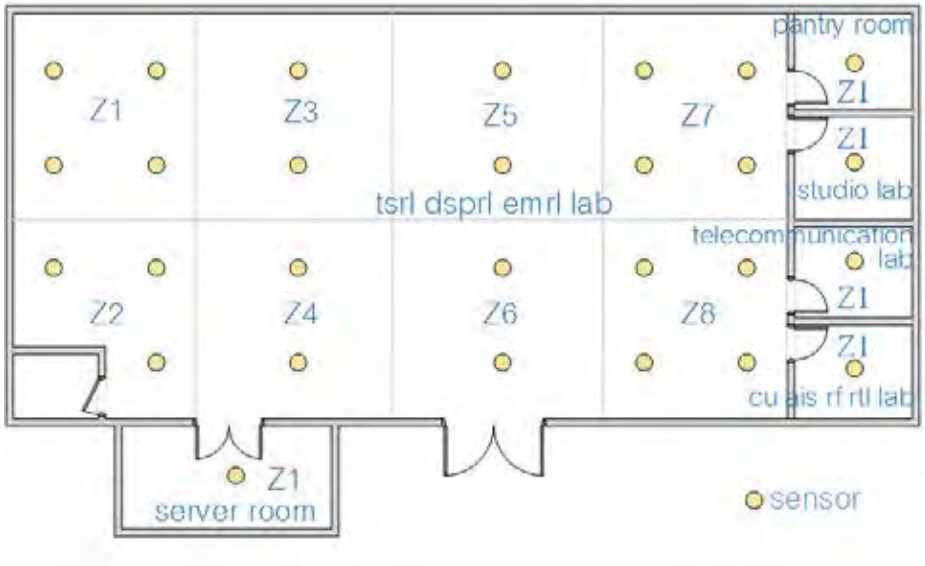
- http://napat.test.chula.ac.th/IEEE1888_ETSIM2M/experiment/node02, 2015-09-05T14:01:52.000+07:00, 0.686

(b) ตัวอย่างข้อมูลตัวรับรู้เลียนแบบที่จีไอพีส่งไปยังจีอาร์เออาร์ ประกอบด้วย ชื่อ container ตราเวลา และค่าของตัวรับรู้อยู่ในรูปแบบ JSON ที่มีขนาดเท่ากับ 88 ไบต์

- experiment_node01, {"data":{"timestamp":"2015-09-05T14:01:52.000+07:00", "consumed":"0.639"}}
- experiment_node02, {"data":{"timestamp":"2015-09-05T14:01:52.000+07:00", "consumed":"0.686"}}

การทดสอบแต่ละครั้งแสดงจำนวนโนดตัวรับรู้ต่อเกตเวย์ที่มีคาบเวลาในการส่งข้อมูลตัวรับรู้แต่ละโนดเท่ากับ 1 นาที แบบประสานเวลากันด้วยเวลาทดสอบครั้งละ 30 นาที

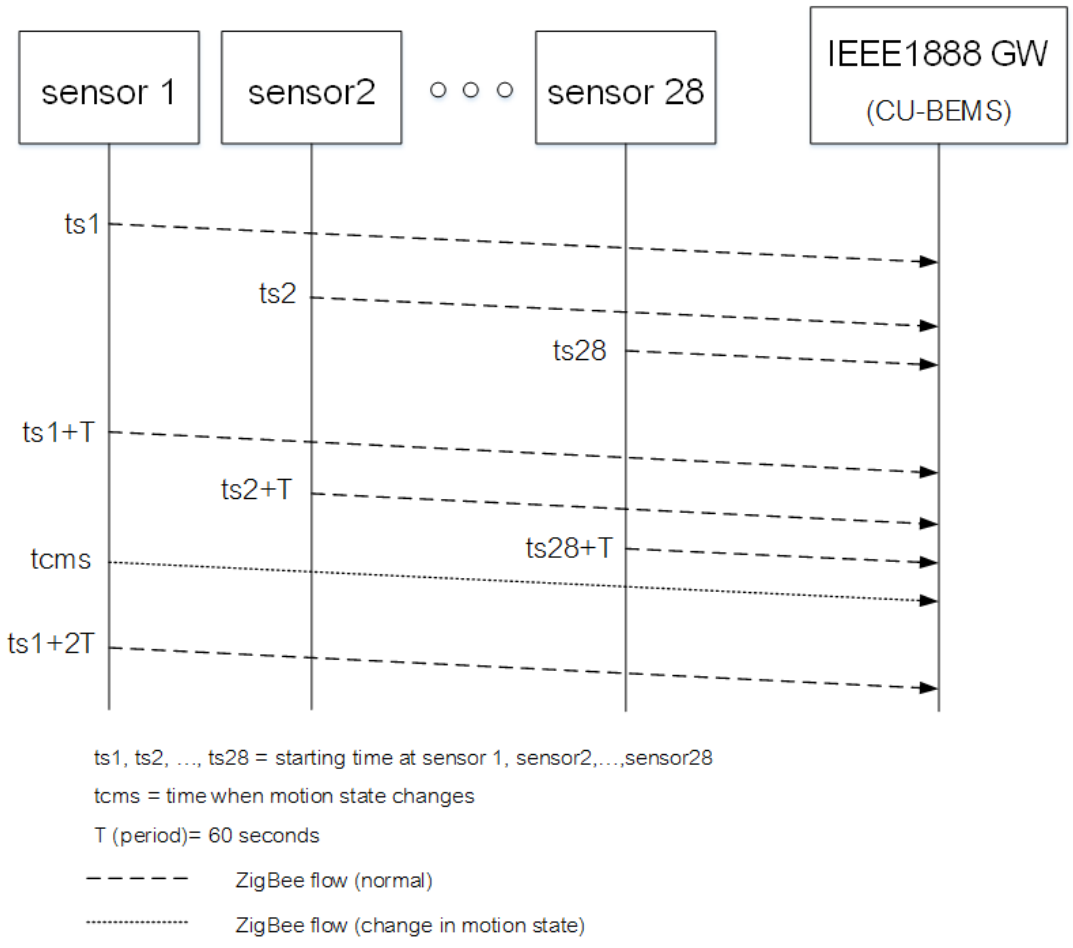
2. **ข้อมูลตัวรับรู้ในโครงการ CU-BEMS**



รูปที่ 5.3: แผนผังตำแหน่งตัวรับรู้วัดสภาพแวดล้อมทั้งหมด 28 ตำแหน่ง ที่ติดตั้งในห้องปฏิบัติการวิจัยระบบโทรคมนาคม

การทดสอบด้วยข้อมูลตัวรับรู้ในโครงการ CU-BEMS ที่ติดตั้งในห้องปฏิบัติการวิจัยระบบโทรคมนาคมที่มีตัวรับรู้วัดสภาพแวดล้อมไร้สาย ZigBee ทั้งหมด 28 ตำแหน่ง ดังแผนผังรูปที่ 5.3 ตัวรับรู้วัดสภาพแวดล้อม 1 ตำแหน่งมีตัวรับรู้ 4 โนด ประกอบด้วย อุณหภูมิ ความชื้นสัมพัทธ์ แสง และการเคลื่อนไหวของคน โดยบริเวณดังกล่าวมีจำนวนโนดตัวรับรู้รวมทั้งหมด 112 โนดซึ่งมีจำนวนโนดตัวรับรู้ต่อเกตเวย์มากที่สุดในโครงการ CU-BEMS เพื่อหาสมรรถนะของระบบสูงสุดเมื่อต้องการนำระบบนี้มาใช้กับโครงการ CU-BEMS ระยะเวลาในการทดสอบ 3 ชั่วโมง จากรูปที่ 5.4 แสดงผังเวลาการสื่อสารระหว่างตัวรับรู้วัด

สภาพแวดล้อม และเกตเวย์ในโครงการ CU-BEMS โดยตัวรับรู้วัดสภาพแวดล้อมไร้สาย ZigBee แต่ละตำแหน่งส่งข้อมูลตัวรับรู้ไม่ประสานเวลากัน โดยส่งข้อมูลตัวรับรู้ที่วัดได้ 4 โนดมายังเกตเวย์ที่เชื่อมต่อ ZigBee ภาครับ ด้วยขนาดของข้อมูล 28 ไบต์ แบบคาบเวลาทุก ๆ นาที และส่งข้อมูลตัวรับรู้การเคลื่อนไหวของคนแบบไม่ประสานเวลาเมื่อมีการเปลี่ยนสถานะลอจิกของการเคลื่อนไหวของคนขึ้น เช่น เมื่อตัวรับรู้การเคลื่อนไหวของคนสามารถตรวจจับการเคลื่อนไหวในขณะที่คนเดินผ่าน ตัวรับรู้จะเปลี่ยนสถานะลอจิกจาก 0 เป็น 1 และตัวรับรู้วัดสภาพแวดล้อมไร้สาย ZigBee จะส่งข้อมูลสถานะลอจิกของการเคลื่อนไหวของคนขณะนั้นด้วยขนาด 22 ไบต์ทันที การทดสอบได้นำ ZigBee ภาครับของเกตเวย์ในโครงการ CU-BEMS ดังกล่าวมาใช้กับเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M ที่พัฒนาขึ้น โดยเกตเวย์ของ IEEE1888 จะปรับข้อมูลตัวรับรู้ตามมาตรฐาน IEEE1888 และในส่วนของเกตเวย์ของ ETSI M2M จะปรับข้อมูลตัวรับรู้ตามมาตรฐาน ETSI M2M



รูปที่ 5.4: ตัวอย่างผังเวลาการสื่อสารระหว่างตัวรับรู้วัดสภาพแวดล้อม และเกตเวย์ของ IEEE1888 ในโครงการ CU-BEMS

(a) ตัวอย่างข้อมูลตัวรับรู้ในโครงการ CU-BEMS ที่เกตเวย์ของ IEEE1888 ส่งไปยังหน่วยเก็บข้อมูลของ IEEE1888 ประกอบด้วย point id ซึ่งระบุตำแหน่ง และชนิดของ

ตัวรับรู้ นอกจากนี้มีการระบุตราเวลา และค่าของตัวรับรู้ ที่มีขนาดรวมของข้อมูลตัวรับรู้ทั้ง 4 ชนิดเท่ากับ 509 ไบต์

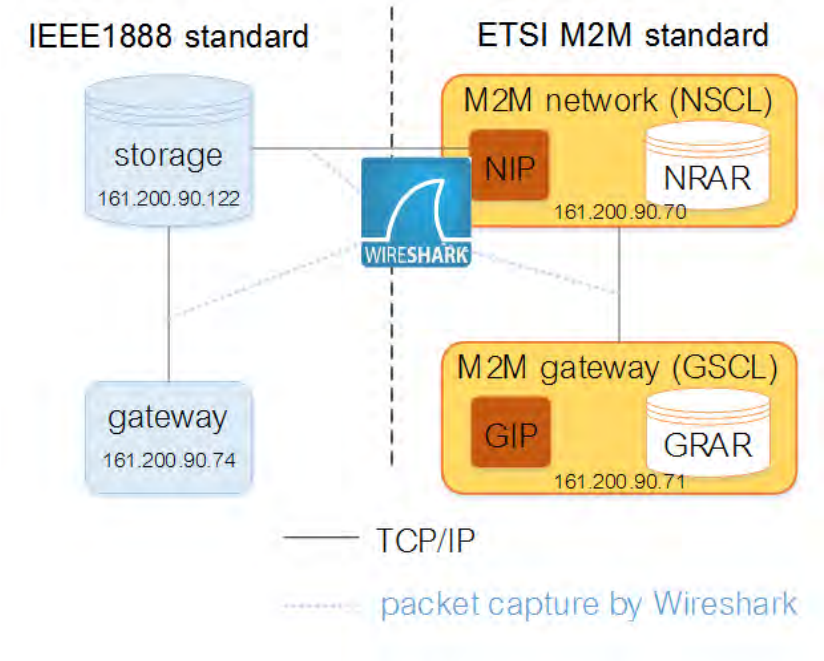
- http://bems.ee.eng.chula.ac.th/eng4/fl13/north/lab_tsrl_dsprl_emrl/z1/sensor1/monitor/humidity, 2015-09-20T14:31:23.000+07:00, 57.4 มีขนาดเท่ากับ 127 ไบต์
- http://bems.ee.eng.chula.ac.th/eng4/fl13/north/lab_tsrl_dsprl_emrl/z1/sensor1/monitor/illuminance, 2015-09-20T14:31:23.000+07:00, 23.4 มีขนาดเท่ากับ 130 ไบต์
- http://bems.ee.eng.chula.ac.th/eng4/fl13/north/lab_tsrl_dsprl_emrl/z1/sensor1/monitor/pir, 2015-09-20T14:31:23.000+07:00, OFF มีขนาดเท่ากับ 121 ไบต์
- http://bems.ee.eng.chula.ac.th/eng4/fl13/north/lab_tsrl_dsprl_emrl/z1/sensor1/monitor/temperature, 2015-09-20T14:31:23.000+07:00, 26.90 มีขนาดเท่ากับ 131 ไบต์

(b) ตัวอย่างข้อมูลตัวรับรู้ในโครงการ CU-BEMS ที่จีไอพีส่งไปยังจีอาร์เออาร์ ประกอบด้วยชื่อ container สำหรับระบุตำแหน่งของตัวรับรู้ ซึ่งข้อมูลของค่าของตัวรับรู้แต่ละชนิดและตราเวลาอยู่ในรูปแบบ JSON มีด้วยกัน 2 แบบ คือ ข้อมูลตัวรับรู้การเคลื่อนไหวของคนมีขนาดเท่ากับ 107 ไบต์ และข้อมูลตัวรับรู้ทั้ง 4 ชนิดมีขนาดเท่ากับ 206 ไบต์

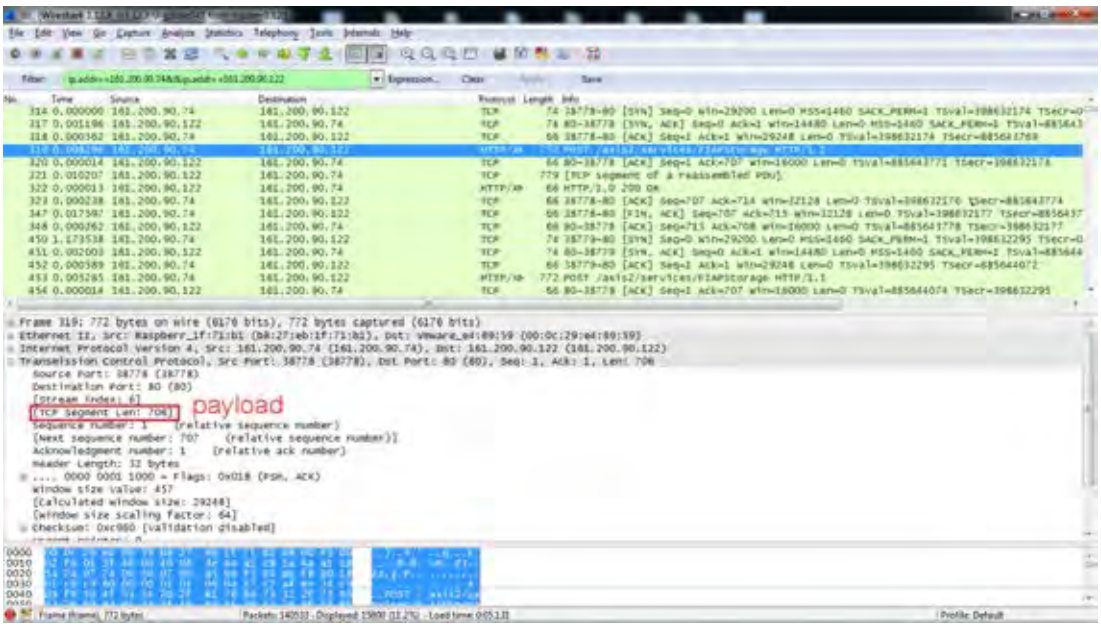
- north_lab_tsrl_dsprl_emrl_z1_sensor1,
  {"pir": {"value": "OFF"},
  "timestamp": "2015-09-20T14:31:23.000+07:00"} มีขนาดเท่ากับ 107 ไบต์
- north_lab_tsrl_dsprl_emrl_z1_sensor1,
  {"illuminance": {"value": "23.4"}, "temperature": {"value": "26.90"},
  "humidity": {"value": "57.4"}, "pir": {"value": "OFF"},
  "timestamp": "2015-09-20T14:31:23.000+07:00"} มีขนาดเท่ากับ 206 ไบต์

5.1 การทดสอบหาเพย์โหลดในการส่งข้อมูล

ข้อมูลตัวรับรู้เลียนแบบ และข้อมูลตัวรับรู้ในโครงการ CU-BEMS จะถูกถ่ายโอนข้อมูลระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M โดยข้อมูลตัวรับรู้ถูกส่งตามโพรโทคอลสื่อสารมาตรฐาน IEEE1888 และวิธีสื่อสารมาตรฐาน ETSI M2M ทำให้มีขนาดของข้อมูลที่มากขึ้น การทดสอบหาเพย์โหลดสำหรับการส่งข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888, โครงข่ายของ ETSI M2M, เกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M โดยใช้โปรแกรม wireshark บนคอมพิวเตอร์ส่วนบุคคลที่ทำหน้าที่เป็นโครงข่ายของ ETSI M2M สำหรับตรวจจับแพ็กเก็ตที่ส่งหากันระหว่างส่วนประกอบของระบบ ดังรูปที่ 5.5 และสามารถตรวจสอบเพย์โหลดที่ใช้ในแต่ละสารจาก TCP segment length ตัวอย่างดังรูปที่ 5.6 ที่มีเพย์โหลดเท่ากับ 706 ไบต์เป็นเพย์โหลดของสาร write



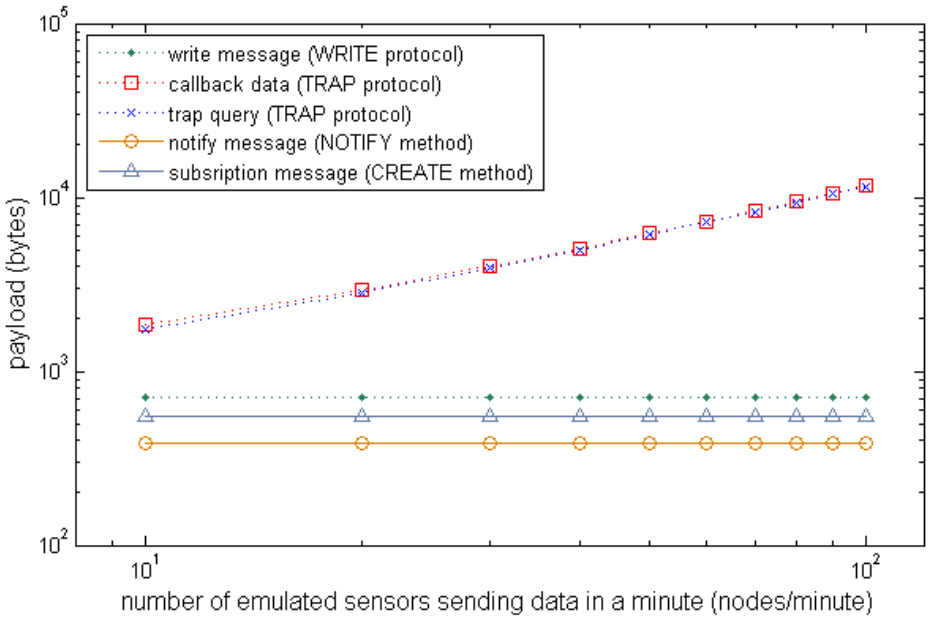
รูปที่ 5.5: การทดสอบโดยใช้โปรแกรม wireshark ตรวจจับแพ็กเก็ต



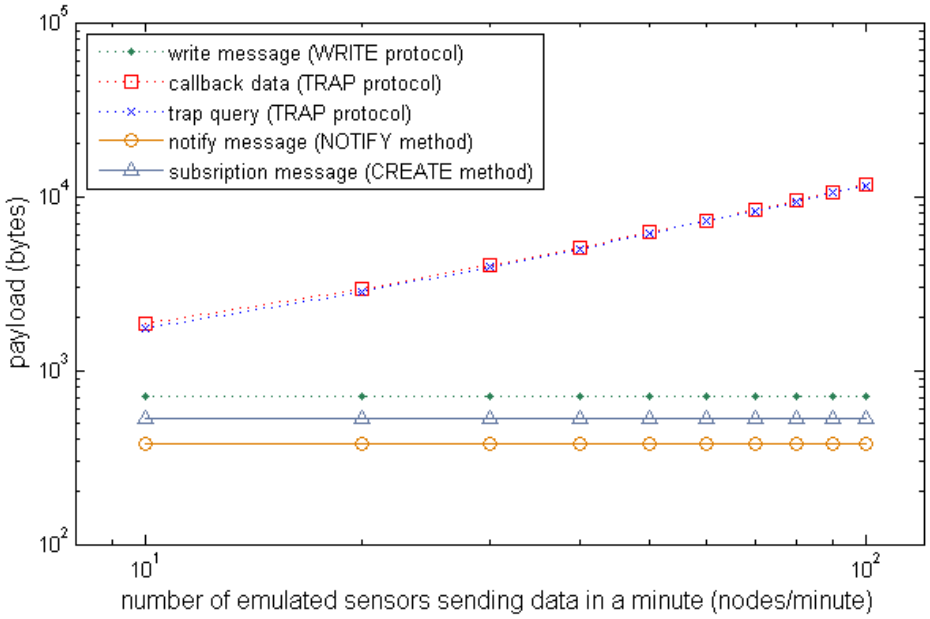
รูปที่ 5.6: ตัวอย่างการหาเพย์โหลดจากโปรแกรม wireshark

การทดสอบได้เพิ่มจำนวนตัวรับรู้เลียนแบบที่ส่งข้อมูลต่อนาทีจนถึง 100 โนดต่อนาที เพื่อหาแนวโน้มเพย์โหลดของสารที่ทำการส่งสำหรับการถ่ายโอนข้อมูลแบบเวลาจริง พบว่าเพย์โหลดของการส่งเงื่อนไขในการแจ้งเตือนข้อมูลตัวรับรู้ และการตอบกลับข้อมูลตัวรับรู้ของโพรโทคอล TRAP แปรผันตามจำนวนของตัวรับรู้เลียนแบบ ซึ่งแตกต่างกับเพย์โหลดของสารรูปแบบอื่นที่มีเพย์โหลดคงที่แสดงดังรูปที่ 5.7 - 5.8 เนื่องจากการส่งเงื่อนไขในการแจ้งเตือนข้อมูลตัวรับรู้ และการตอบกลับข้อมูลตัวรับรู้มีการระบุตัวรับรู้ทั้งหมดที่ร้องขอในการแจ้งเตือนข้อมูลตามโพรโทคอล TRAP

ข้อมูลที่ได้จากตัวรับรู้ไร้สาย ZigBee ในโครงการ CU-BEMS ขนาด 28 ไบต์ และ 22 ไบต์ จะถูกปรับข้อมูลตามโครงสร้างของการสื่อสารตามมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M จากการทดสอบหาเพย์โหลดด้วยข้อมูลตัวรับรู้ในโครงการ CU-BEMS จำนวนโนดตัวรับรู้รวมทั้งหมด 112 โนด ที่มีการส่งข้อมูลตัวรับรู้ 4 โนดที่เป็นแบบคาบเวลา และการส่งเพียงข้อมูลการเคลื่อนไหวของคนที่เป็นแบบไม่ประสานเวลา พบว่าเพย์โหลดของสารสูงขึ้นมากเมื่อเทียบกับขนาดของข้อมูลที่ได้จากตัวรับรู้ไร้สาย ZigBee โดยเฉพาะการส่งเงื่อนไขในการแจ้งเตือนข้อมูลตัวรับรู้ และการตอบกลับข้อมูลตัวรับรู้ตามโพรโทคอล TRAP มีการใช้เพย์โหลดที่สูงกว่าการส่งสารรูปแบบอื่นดังตารางที่ 5.1



รูปที่ 5.7: การเปรียบเทียบเพย์โหลดสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M ด้วยข้อมูลตัวรับรู้เลียนแบบ



รูปที่ 5.8: การเปรียบเทียบเพย์โหลดสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 ด้วยข้อมูลตัวรับรู้เลียนแบบ

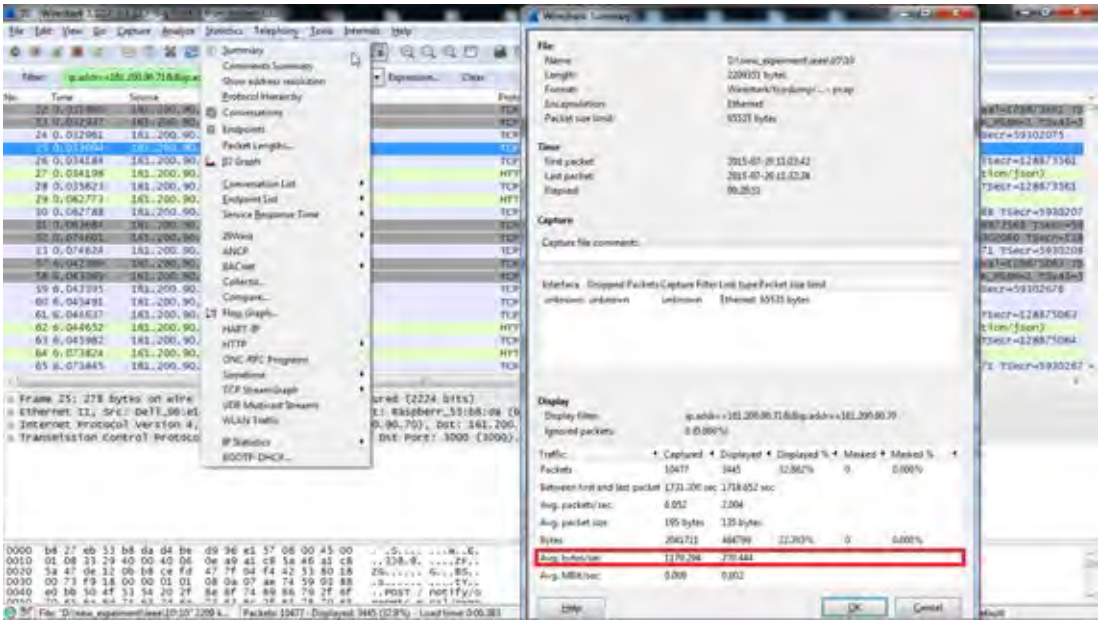
ตารางที่ 5.1: การเปรียบเทียบเพย์โหลดสำหรับการถ่ายโอนข้อมูลระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ ETSI M2M ด้วยข้อมูลตัวรับรู้ 112 โนดในโครงการ CU-BEMS

สาร	เพย์โหลดของการถ่ายโอนข้อมูล (ไบต์)			
	IEEE1888 ไปยัง ETSI M2M		ETSI M2M ไปยัง IEEE1888	
	คาบเวลา	ไม่ประสานเวลา	คาบเวลา	ไม่ประสานเวลา
write	1,257	748	1,257	748
trap query	16,145		16,145	
callback	16,742	16,221	16,742	16,221
subscription	443		437	
notify	705	346	699	340

5.2 การทดสอบหาแบนด์วิดท์

การทดสอบหาแบนด์วิดท์ที่หน่วยเก็บข้อมูลของ IEEE1888, โครงข่ายของ ETSI M2M, เกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M ในการถ่ายโอนข้อมูลระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M โดยการใช้โปรแกรม wireshark บนคอมพิวเตอร์ส่วนบุคคลที่ทำหน้าที่โครงข่ายของ ETSI M2M ดังรูปที่ 5.5 ในการตรวจสอบปริมาณงานเฉลี่ยจากการกรองเฉพาะข้อมูลที่ส่วนประกอบของแต่ละระบบใช้ในการถ่ายโอนข้อมูลโดยกำหนด ip.addr สำหรับเลขที่อยู่ไอพีที่ต้องการวิเคราะห์ หลังจากนั้นเข้าหน้าต่างข้อสรุป โดยปริมาณงานแสดงในแถบ Avg. bytes/sec แสดงดังรูปที่รูปที่ 5.9 ซึ่งกำหนดการกรองที่ใช้สำหรับหาปริมาณงานเฉลี่ยของส่วนประกอบแต่ละระบบ ดังนี้

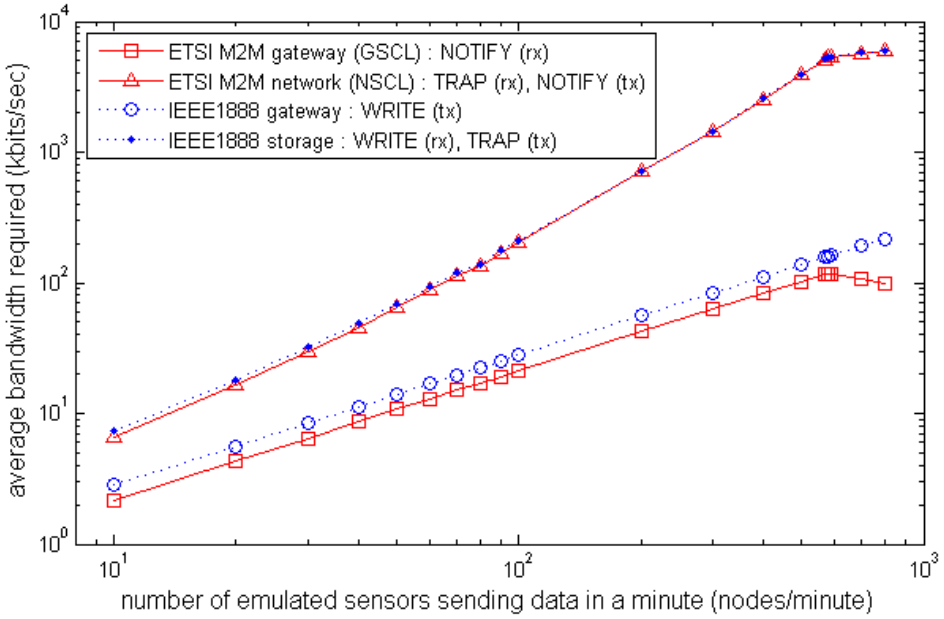
- หน่วยเก็บข้อมูลของ IEEE1888:
  ip.addr==161.200.90.122&&(ip.addr==161.200.90.74||ip.addr==161.200.90.70)
- โครงข่ายของ ETSI M2M:
  ip.addr==161.200.90.70&&(ip.addr==161.200.90.122||ip.addr==161.200.90.71)
- เกตเวย์ของ ETSI M2M:
  ip.addr==161.200.90.71&&ip.addr==161.200.90.70
- เกตเวย์ของ IEEE1888:
  ip.addr==161.200.90.74&&ip.addr==161.200.90.122



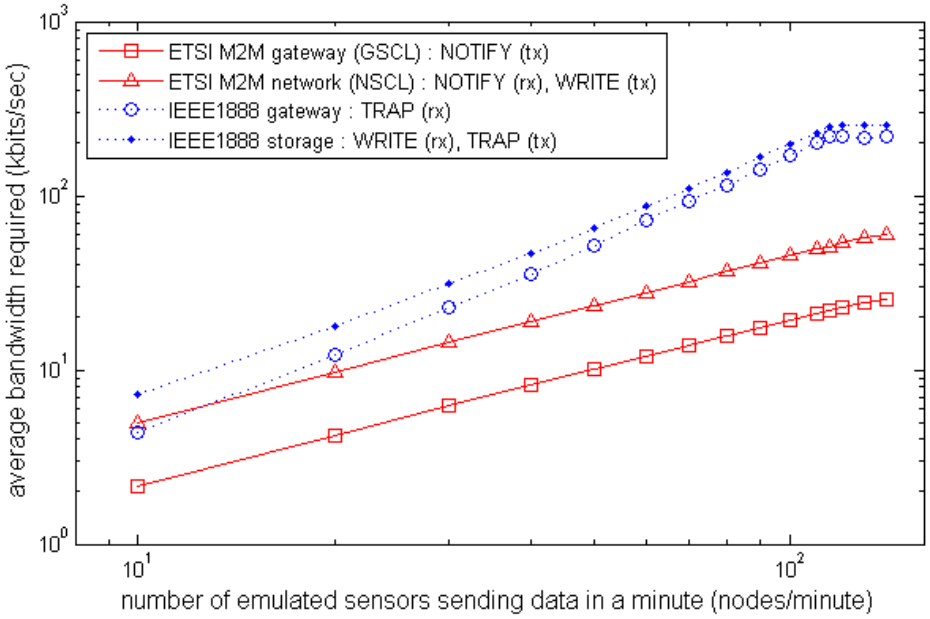
รูปที่ 5.9: การหาปริมาณงานเฉลี่ยของแต่ละส่วนประกอบจากโปรแกรม wireshark

การทดสอบได้เพิ่มจำนวนตัวรับรู้เลียนแบบที่ส่งข้อมูลต่อนาทีจนกระทั่งส่วนประกอบของระบบเกิดการอิ่มตัวของปริมาณงานเฉลี่ยขึ้นเพื่อหาคอขวด (bottleneck) ของการถ่ายโอนข้อมูล จากการทดสอบการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M พบว่าระบบเกิดการอิ่มตัวของปริมาณงานเฉลี่ยขึ้นที่หน่วยเก็บข้อมูลของ IEEE1888 และโครงข่ายของ ETSI M2M ด้วยปริมาณงานเฉลี่ย 5,319 และ 5,277 กิโลบิตต่อวินาที ตามลำดับ ที่จำนวนตัวรับรู้ 580 โนด และในส่วนของเกตเวย์ของ ETSI M2M มีการใช้ปริมาณงานเฉลี่ยที่ลดลงเมื่อเพิ่มจำนวนตัวรับรู้มากกว่า 580 โนด เนื่องจากอัตราการรับข้อมูลที่ช้าลงเป็นผลมาจากการอิ่มตัวของปริมาณงานเฉลี่ยที่หน่วยเก็บข้อมูลของ IEEE1888 และโครงข่ายของ ETSI M2M ดังรูปที่ 5.10 โดยพบว่าส่วนประกอบที่เป็นคอขวดของการถ่ายโอนข้อมูลอยู่ที่โครงข่ายของ ETSI M2M ที่ดำเนินการทดสอบบนคอมพิวเตอร์ส่วนบุคคล

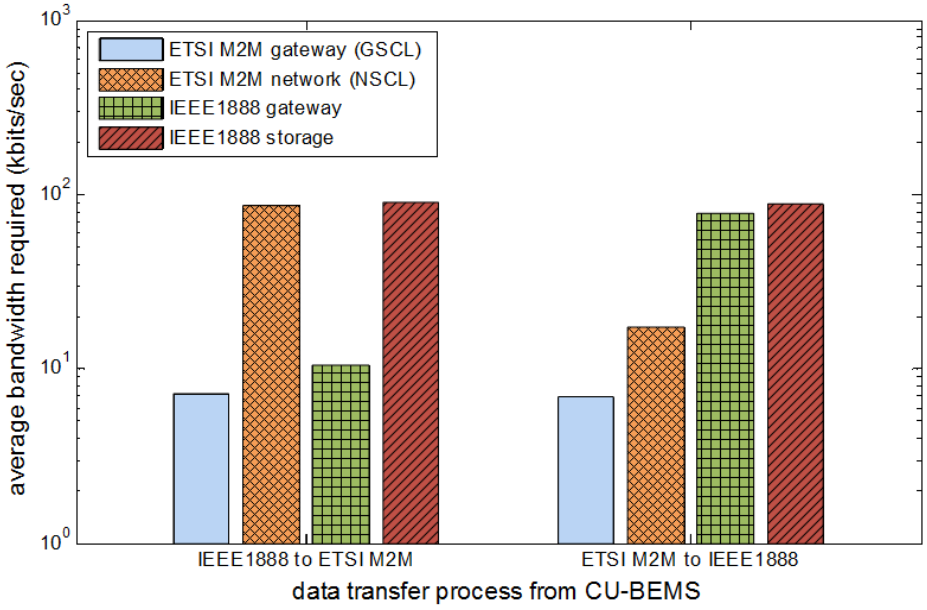
นอกจากนี้การถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 พบว่าระบบเกิดการอิ่มตัวของปริมาณงานเฉลี่ยขึ้นที่หน่วยเก็บข้อมูลของ IEEE1888 และเกตเวย์ของ IEEE1888 ด้วยปริมาณงานเฉลี่ย 247 และ 218 กิโลบิตต่อวินาที ตามลำดับ ที่จำนวนตัวรับรู้ 115 โนด ดังรูปที่ 5.11 โดยส่วนประกอบที่เป็นคอขวดของการถ่ายโอนข้อมูลอยู่ที่เกตเวย์ของ IEEE1888 ที่ดำเนินการทดสอบบนบอร์ด Raspberry Pi B+ การอิ่มตัวของปริมาณงานเฉลี่ยเกิดจากการประวิงในการตอบกลับข้อมูลของโพรโทคอล TRAP เนื่องจากเมื่อจำนวนตัวรับรู้มากขึ้นจะเพิ่มเพย์โหลดของการตอบกลับข้อมูลตามโพรโทคอล TRAP ดังที่ได้อธิบายในหัวข้อการทดสอบเพย์โหลดที่ 5.1 และส่งผลต่อเวลาการเดินทางครบรอบ (round-trip time) เมื่อถึงจุดที่อัตราการรับข้อมูลของหน่วยเก็บข้อมูลของ IEEE1888 สูงกว่าอัตราการตอบกลับการแจ้งเตือนข้อมูลทำให้เกิดการประวิงในการตอบกลับการแจ้งเตือนข้อมูลขึ้น



รูปที่ 5.10: แบนด์วิดท์เฉลี่ยของส่วนประกอบแต่ละระบบสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M ด้วยข้อมูลตัวรับรู้เลียนแบบ



รูปที่ 5.11: แบนด์วิดท์เฉลี่ยของส่วนประกอบแต่ละระบบสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 ด้วยข้อมูลตัวรับรู้เลียนแบบ

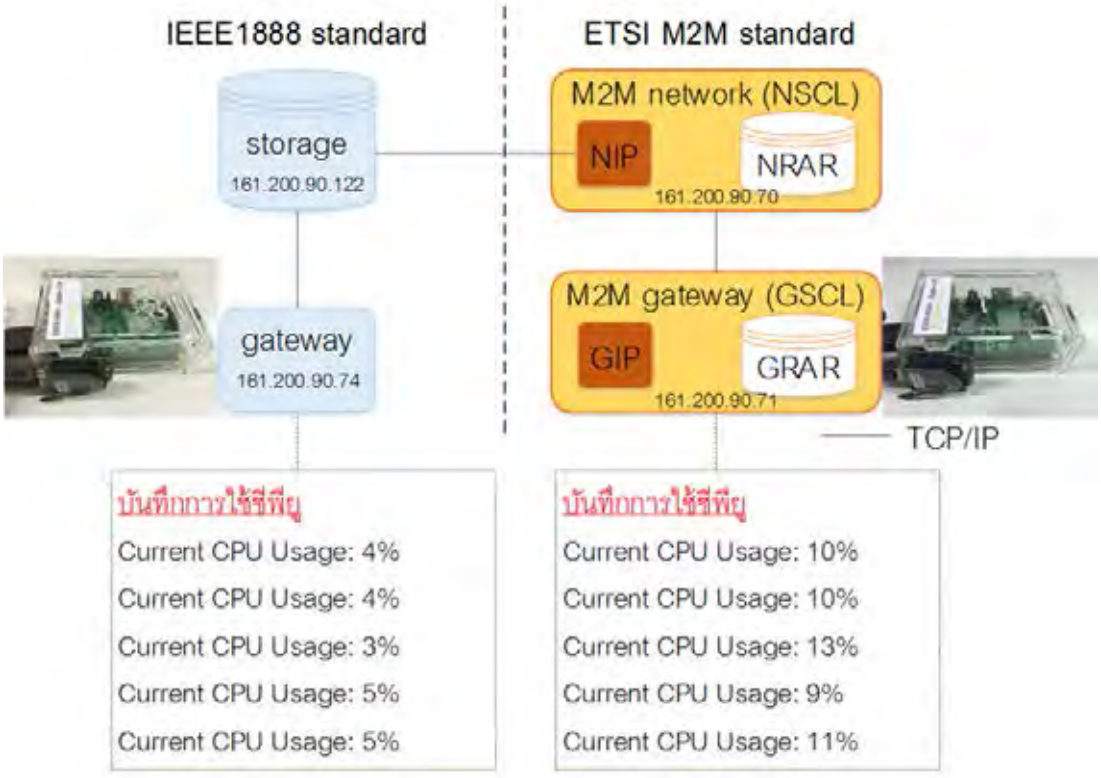


รูปที่ 5.12: แบนด์วิดท์เฉลี่ยสำหรับการถ่ายโอนข้อมูลระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M ด้วยข้อมูลตัวรับรู้ 112 โนดในโครงการ CU-BEMS

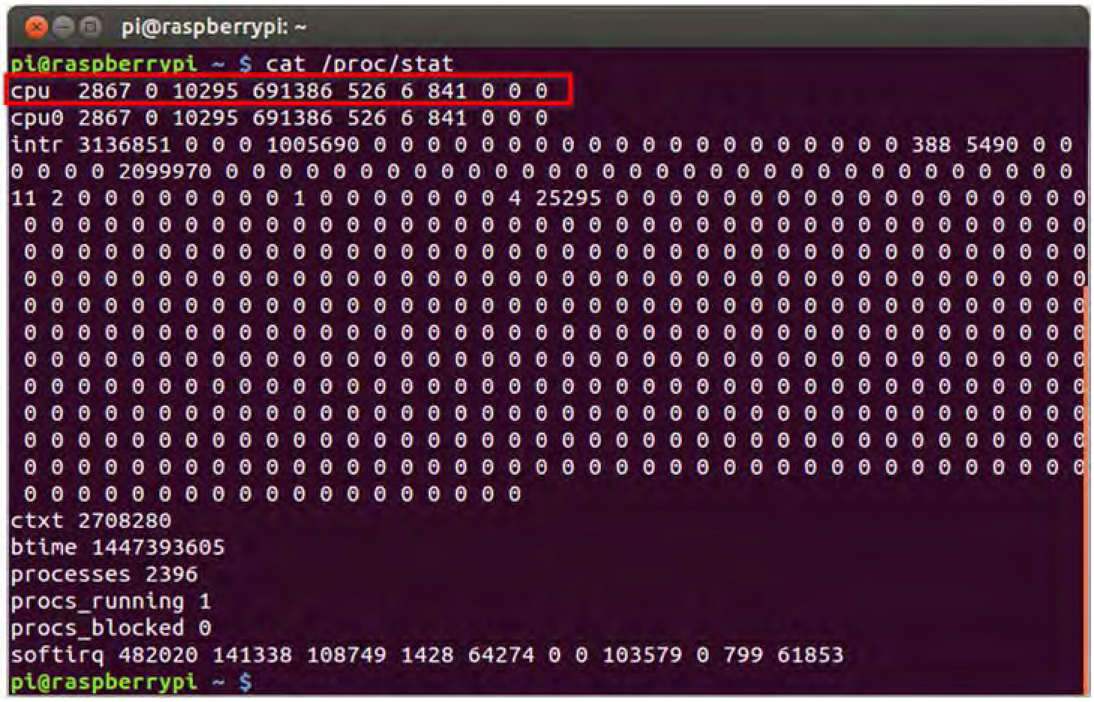
จากรูปที่ 5.12 แสดงการใช้ปริมาณงานเฉลี่ยสำหรับการทดสอบด้วยข้อมูลตัวรับรู้ในโครงการ CU-BEMS บริเวณที่มีจำนวนโนดตัวรับรู้ทั้งหมด 112 โนดที่ส่งข้อมูลต่อนาที พบว่าโครงข่ายของ ETSI M2M และหน่วยเก็บข้อมูลของ IEEE1888 สำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M มีการใช้ปริมาณงานสูงที่ 86.15 และ 89.46 กิโลบิตต่อวินาที ตามลำดับ เนื่องจากผลของการใช้โพรโทคอล TRAP เช่นเดียวกันกับเกตเวย์ของ IEEE1888 และหน่วยเก็บข้อมูลของ IEEE1888 สำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 มีการใช้ปริมาณงานสูงที่ 77.12 และ 87.43 กิโลบิตต่อวินาที ตามลำดับ

5.3 การทดสอบหาการใช้ซีพียูของเกตเวย์

การทดสอบการใช้ซีพียูที่เกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M เพื่อพิจารณาภาระการทำงานของเกตเวย์ทั้ง 2 มาตรฐาน ซึ่งดำเนินการบนบอร์ด Raspberry Pi B+ ที่มีคุณลักษณะเหมือนกันตามที่ได้อธิบายในส่วนประกอบของระบบหัวข้อที่ 3.2 โดยบันทึกการใช้ซีพียูหน่วยเปอร์เซ็นต์ในแฟ้มข้อความที่เกตเวย์ทั้ง 2 มาตรฐาน ดังรูปที่ 5.13



รูปที่ 5.13: ตัวอย่างการบันทึกข้อมูลการใช้ซีพียู



รูปที่ 5.14: ตัวอย่างข้อมูลซีพียูจากแฟ้ม /proc/stat บนระบบปฏิบัติการลินุกซ์

โปรแกรมอ่านข้อมูลจากแฟ้ม /proc/stat สำหรับแสดงข้อมูลซีพียูบนระบบปฏิบัติการลินุกซ์ (Linux) ดังรูปที่ 5.14 พร้อมกับคำนวณ และบันทึกการใช้ซีพียูทุก ๆ วินาที การคำนวณการใช้ซีพียูจากบรรทัดที่ชื่อ CPU ประกอบด้วยเวลาของซีพียูในส่วนต่าง ๆ มีหน่วยเวลาที่เรียกว่า จิฟฟี (jiffy) เท่ากับ 10^{-2} วินาที ซึ่งมีเวลาของซีพียูในแต่ละส่วนดังนี้

	user	nice	system	idle	iowait	irq	softirq	steal	guest	guest_nice
cpu	2867	0	10295	691386	526	6	841	0	0	0

user	คือ	ซีพียูในโมดผู้ใช้ (user mode)
nice	คือ	ซีพียูในโมดผู้ใช้ที่ลำดับความสำคัญต่ำ (low priority)
system	คือ	ซีพียูในโมดระบบ (system mode)
idle	คือ	ซีพียูที่ยังไม่ได้ใช้งาน
iowait	คือ	การรอคอยการทำงานของ I/O
irq	คือ	ซีพียูที่รับการทำงานของการขัดจังหวะฮาร์ดแวร์ (hardware interrupt)
softirq	คือ	ซีพียูที่รับการทำงานของการขัดจังหวะซอฟต์แวร์ (software interrupt)
steal	คือ	ซีพียูที่นำไปใช้ในการทำงานของเครื่องเสมือน (virtual machine) สำหรับระบบปฏิบัติการอื่น ๆ
guest	คือ	การทำงานของซีพียูเสมือนสำหรับผู้อื่นที่ใช้ระบบปฏิบัติการภายใต้การควบคุมของลินุกซ์ที่เคอร์เนล
guest_nice	คือ	การทำงานของซีพียูเสมือนสำหรับผู้อื่นที่ใช้ระบบปฏิบัติการภายใต้การควบคุมของลินุกซ์ที่เคอร์เนลในโมดผู้ใช้ที่ลำดับความสำคัญต่ำ

ตัวแปรสำหรับสูตรคำนวณการใช้ซีพียูประกอบด้วยดังนี้

$CPU\ usage$	คือ	ค่าการใช้ซีพียู (เปอร์เซ็นต์)
$CPU\ total$	คือ	เวลาทั้งหมดของซีพียู (จิฟฟี)
$CPU\ active$	คือ	เวลาของซีพียูที่ใช้งาน (จิฟฟี)
$\Delta CPU\ total$	คือ	ผลต่างระหว่างเวลาทั้งหมดของซีพียูปัจจุบัน และเวลาทั้งหมดของซีพียูก่อนหน้า (จิฟฟี)
$\Delta CPU\ active$	คือ	ผลต่างระหว่างเวลาของซีพียูที่ใช้งานปัจจุบัน และเวลาของซีพียูที่ใช้งานก่อนหน้า (จิฟฟี)

การคำนวณค่าการใช้ซีพียู [25] มีสูตรดังนี้

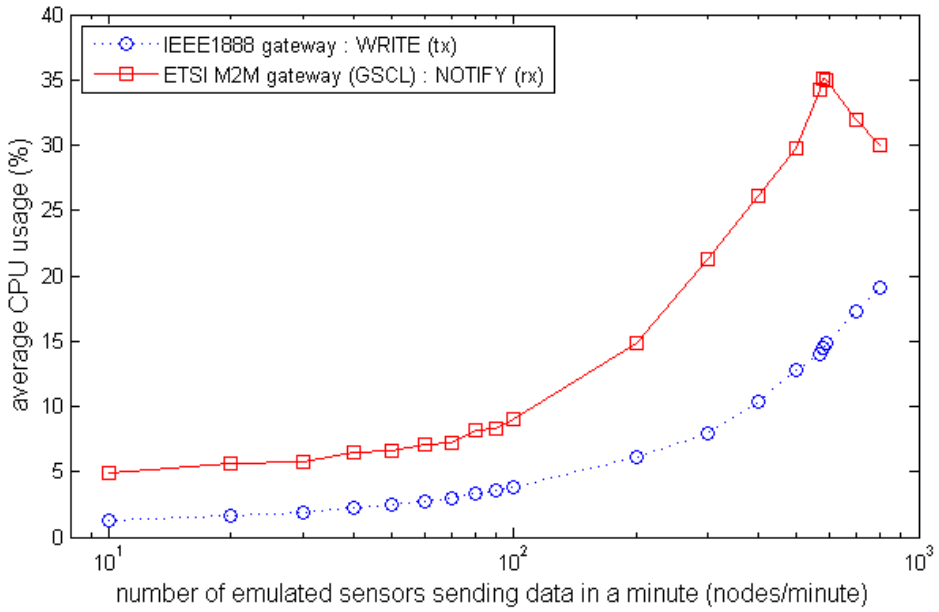
$$CPU\ usage = \frac{\Delta CPU\ active}{\Delta CPU\ total} \times 100\% \tag{5.1}$$

โดยที่

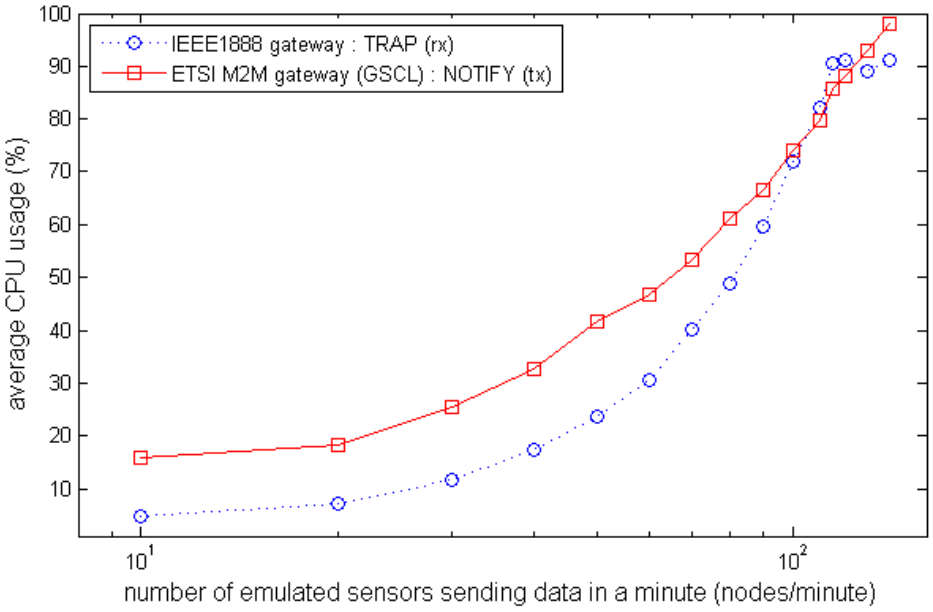
$$CPU\ total = user + nice + system + idle + iowait + irq + softirq + steal + guest + guest\_nice \tag{5.2}$$

$$CPU\ active = CPU\ total - (idle + iowait) \tag{5.3}$$

หลังจากนั้นนำค่าการใช้ซีพียูที่บันทึกไว้มาหาค่าเฉลี่ย การทดสอบได้เพิ่มจำนวนตัวรับรู้เลียนแบบที่ส่งข้อมูลต่อนาทีเช่นเดียวกับการทดสอบแบนด์วิดท์ในหัวข้อที่ 5.2 เพื่อหาแนวโน้มการใช้ซีพียูที่เกตเวย์ทั้ง 2 มาตรฐาน จากรูปที่ 5.15 - 5.16 พบว่าแนวโน้มการใช้ซีพียูของเกตเวย์ทั้ง 2 มาตรฐานมีความสอดคล้องกับการทดสอบหาแบนด์วิดท์ในหัวข้อที่ 5.2 โดยเกตเวย์ทั้ง 2 มาตรฐานสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M มีการใช้ซีพียูที่ไม่สูงซึ่งเกตเวย์ของ ETSI M2M มีการใช้ซีพียูสูงสุดอยู่ที่ 35.5% ที่จำนวนตัวรับรู้ 580 โนด หลังจากนั้นการใช้ซีพียูจะลดลงเนื่องจากเกตเวย์ของ ETSI M2M รับข้อมูลได้น้อยลงจากการอิ่มตัวของปริมาณงานเฉลี่ยขึ้นที่หน่วยเก็บข้อมูลของ IEEE1888 และโครงข่ายของ ETSI M2M โดยส่วนประกอบที่เป็นคอขวดของการถ่ายโอนข้อมูลอยู่ที่โครงข่ายของ ETSI M2M ดำเนินการทดสอบบนคอมพิวเตอร์ส่วนบุคคลที่ทำหน้าที่เอ็นไอพีในการใช้โพรโทคอล TRAP ที่มีเพย์โหลดไม่คงที่ และการทำหน้าที่เอ็นอาร์เออาร์ ส่วนในกรณีของการถ่ายโอนข้อมูลจากเกตเวย์ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 พบว่าเกตเวย์ทั้ง 2 มาตรฐานมีการใช้ซีพียูที่สูงเนื่องจากต้องรองรับภาระส่วนใหญ่ของการถ่ายโอนข้อมูล โดยเกตเวย์ของ ETSI M2M มีการทำหน้าที่บันทึกข้อมูลด้วยวิธี CREATE และการทำหน้าที่ของจีอาร์เออาร์ ในส่วนของเกตเวย์ของ IEEE1888 ดำเนินการทดสอบบนบอร์ด Raspberry Pi B+ ซึ่งเป็นส่วนประกอบที่เป็นคอขวดของการถ่ายโอนข้อมูลทำหน้าที่รับการตอบกลับการแจ้งเตือนข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล TRAP มีการใช้ซีพียูอิ่มตัวที่ 90.5% ที่จำนวนตัวรับรู้ 115 โนด

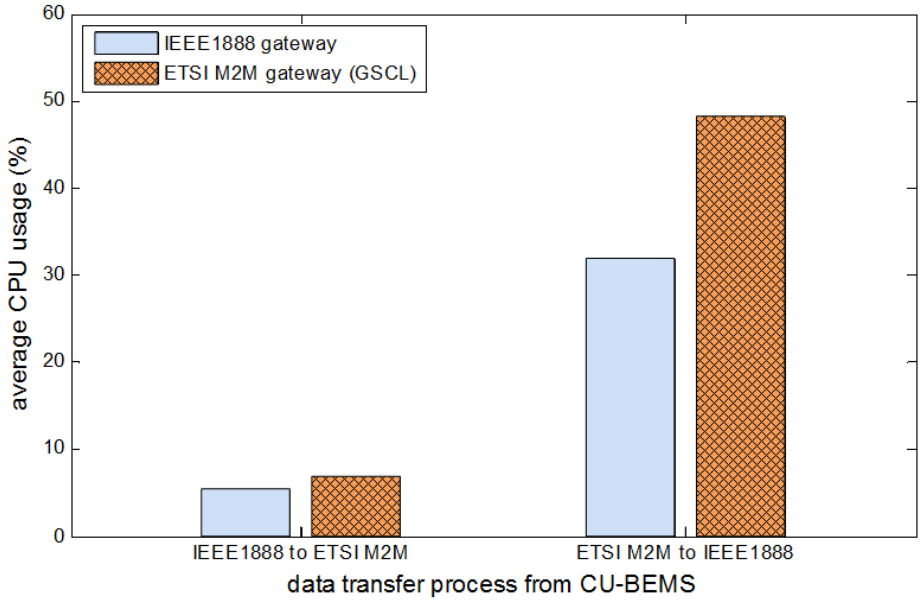


รูปที่ 5.15: การใช้ซีพียูเฉลี่ยของเกตเวย์สำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M ด้วยข้อมูลตัวรับรู้เลียนแบบ



รูปที่ 5.16: การใช้ซีพียูเฉลี่ยของเกตเวย์สำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 ด้วยข้อมูลตัวรับรู้เลียนแบบ

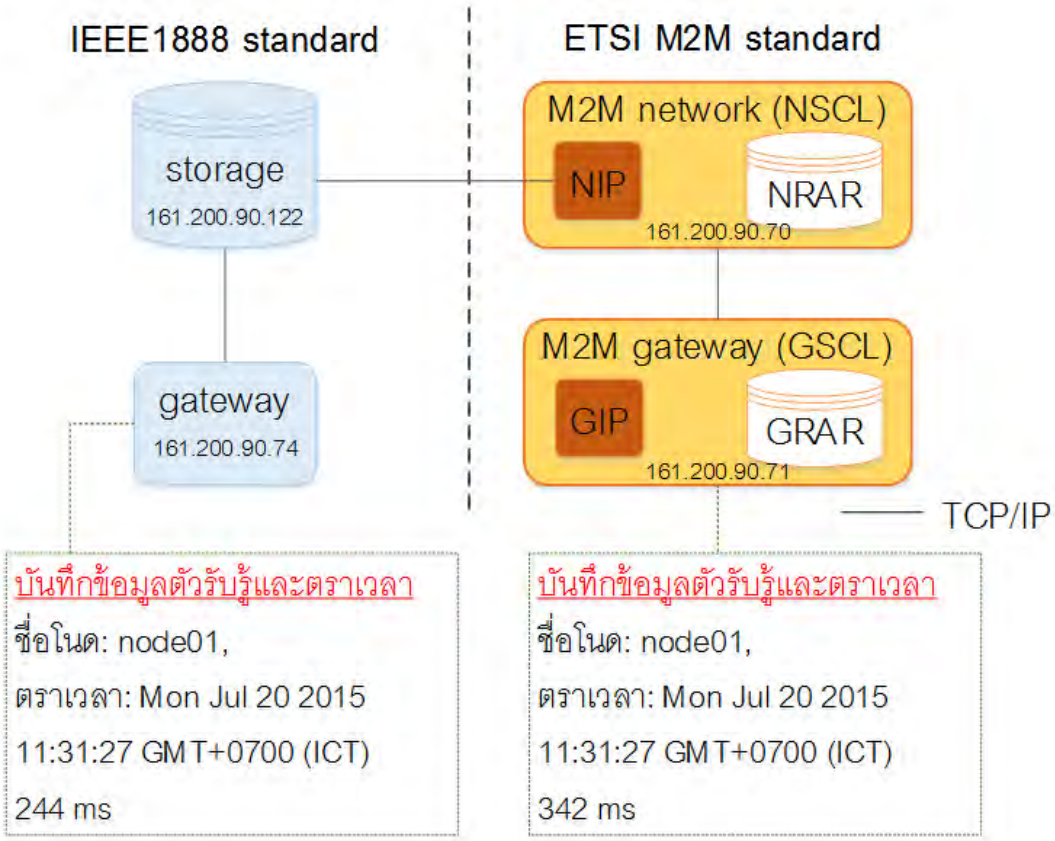
การทดสอบด้วยข้อมูลตัวรับรู้ในโครงการ CU-BEMS บริเวณที่มีจำนวนโนดตัวรับรู้ต่อเกตเวย์มากที่สุดในโครงการ CU-BEMS มีจำนวนโนดตัวรับรู้ทั้งหมด 112 โนดที่ส่งข้อมูลต่อนาที เพื่อหาการใช้ซีพียูสูงสุดของเกตเวย์ทั้ง 2 มาตรฐานสำหรับระบบการทำงานร่วมกันกับโครงการ CU-BEMS พบว่าการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M ที่เกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M มีการใช้ซีพียูเพียง 5.4% และ 6.8% ตามลำดับ โดยมีการใช้ซีพียูน้อยกว่าการถ่ายโอนข้อมูลจากเกตเวย์ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 ที่เกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M มีการใช้ซีพียูอยู่ที่ 31.9% และ 48.2% ตามลำดับ



รูปที่ 5.17: การใช้ซีพียูของเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M สำหรับการถ่ายโอนข้อมูลระหว่างเกตเวย์ทั้ง 2 มาตรฐานด้วยข้อมูลตัวรับรู้ 112 โนดในโครงการ CU-BEMS

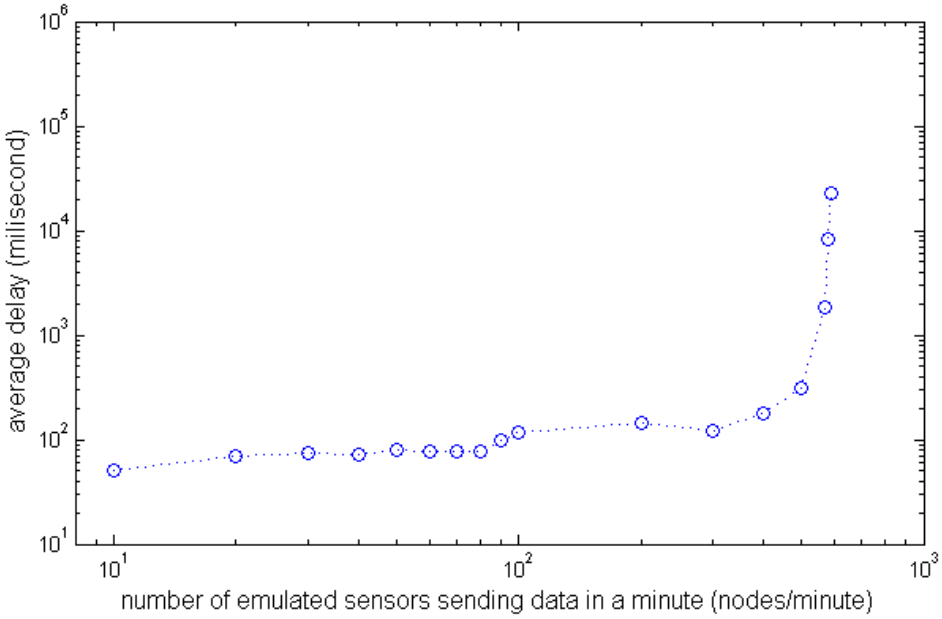
5.4 การทดสอบหาเวลาประวิงในการถ่ายโอนข้อมูล

การทดสอบเวลาประวิงในการถ่ายโอนข้อมูลระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M โดยเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M สร้างตราเวลาโดยใช้ NTP (network time protocol) สำหรับประสานเวลาเพื่อให้ส่วนประกอบในระบบใช้เวลาเดียวกันที่มีความแม่นยำอยู่ที่ ± 1 มิลลิวินาที [26] ซึ่งตัวบริการ NTP อยู่ที่หน่วยเก็บข้อมูลของ IEEE1888 จากรูปที่ 5.18 แสดงรูปแบบในการบันทึกข้อมูลที่เกตเวย์ของ IEEE1888 และจีไอพีสำหรับการถ่ายโอนข้อมูลระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M ทั้ง 2 แบบ ข้อมูลที่บันทึกประกอบด้วย ชื่อโนดตัวรับรู้ และตราเวลา ได้แก่ วัน เดือน วันที่ ปี (คริสต์ศักราช) ชั่วโมง นาที วินาที โซนเวลา และมิลลิวินาที ซึ่งบันทึกในแฟ้มข้อความ หลังจากนั้นนำข้อมูลจากแฟ้มข้อความของเกตเวย์ทั้ง 2 มาตรฐานมาจับคู่ชื่อโนดตัวรับรู้ และตราเวลา เพื่อหาเวลาประวิงจากความแตกต่างในช่วงเวลาพร้อมกับหาค่าเฉลี่ยของเวลาประวิงในหน่วยของมิลลิวินาที

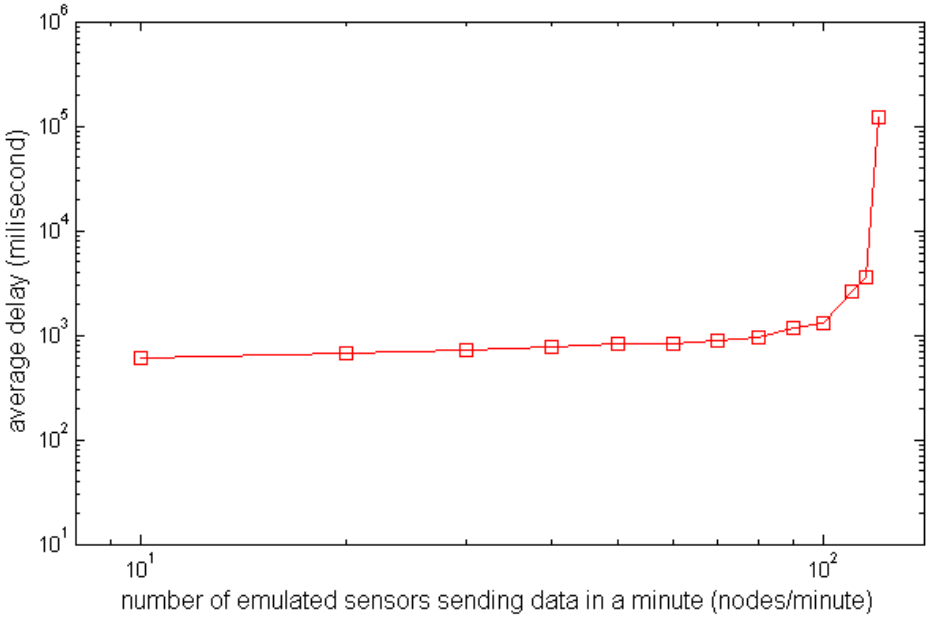


รูปที่ 5.18: ตัวอย่างการบันทึกการบันทึกชื่อโนดตัวรับรู้ และตราเวลาของข้อมูล

การทดสอบได้เพิ่มจำนวนตัวรับรู้เลียนแบบที่ส่งข้อมูลต่อนาทีเพื่อหาเวลาประวิงเฉลี่ยมากที่สุดที่ระบบสามารถรองรับได้ จากรูปที่ 5.19 - 5.20 แสดงเวลาประวิงในการถ่ายโอนข้อมูลระหว่างเกตเวย์ทั้ง 2 มาตรฐาน พบว่าการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M สามารถรองรับตัวรับรู้มากที่สุด 500 โนดด้วยเวลาประวิงเฉลี่ย 305 มิลลิวินาที ซึ่งรองรับตัวรับรู้ได้มากกว่าการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 ที่สามารถรองรับตัวรับรู้มากที่สุด 115 โนดด้วยเวลาประวิงเฉลี่ย 3,561 มิลลิวินาที และพบว่าการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 มีเวลาประวิงสูงกว่าการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M เป็นผลมาจากภาระการถ่ายโอนข้อมูลส่วนใหญ่อยู่ที่เกตเวย์ทั้ง 2 มาตรฐานที่มีข้อจำกัดของระบบประมวลผลที่ล่าช้า

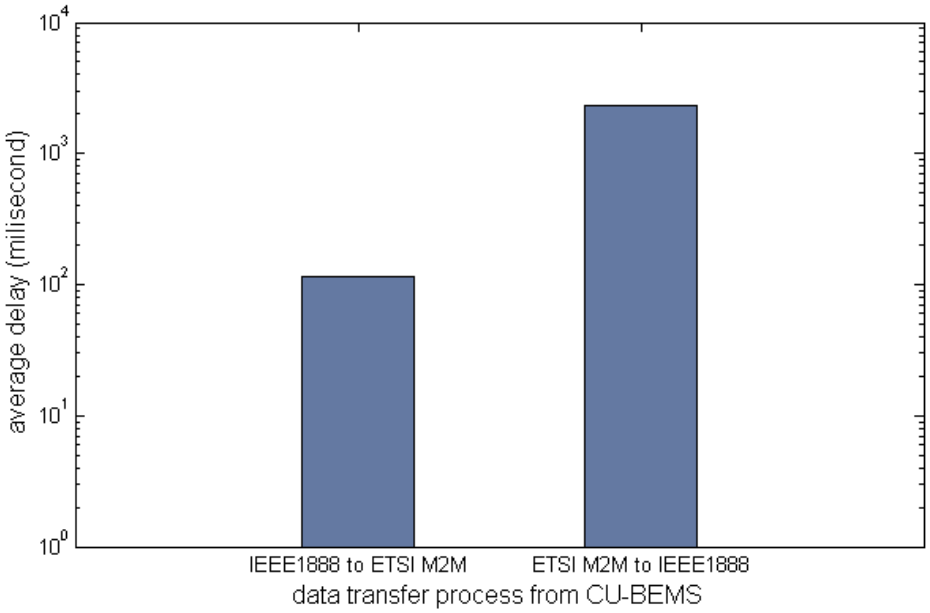


รูปที่ 5.19: เวลาประวิงเฉลี่ยสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M ด้วยข้อมูลตัวรับรู้เลียนแบบ



รูปที่ 5.20: เวลาประวิงเฉลี่ยสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 ด้วยข้อมูลตัวรับรู้เลียนแบบ

จากรูปที่ 5.21 แสดงเวลาประวิงเฉลี่ยของการทดสอบด้วยข้อมูลตัวรับรู้ในโครงการ CU-BEMS บริเวณที่มีจำนวนโนดตัวรับรู้ต่อเกตเวย์มากที่สุดในโครงการ CU-BEMS มีจำนวนโนดตัวรับรู้ทั้งหมด 112 โนดที่ส่งข้อมูลต่อนาที พบว่าเวลาประวิงเฉลี่ยสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M มีค่าเท่ากับ 114 มิลลิวินาที และเมื่อพิจารณาเวลาประวิงเฉลี่ยสำหรับการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 มีค่าสูงกว่าถึง 2,306 มิลลิวินาที ซึ่งระบบยังสามารถทำงานได้



รูปที่ 5.21: เวลาประวิงเฉลี่ยสำหรับการถ่ายโอนข้อมูลระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M ด้วยข้อมูลตัวรับรู้ 112 โนดในโครงการ CU-BEMS

บทที่ 6

บทสรุปและข้อเสนอแนะ

6.1 บทสรุป

วิทยานิพนธ์ฉบับนี้นำเสนอระบบการทำงานร่วมกันแบบเวลาจริงระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M เพื่อการประสานข้อมูลแบบเวลาจริงระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M และให้อุปกรณ์ที่ทำงานตามมาตรฐานที่ต่างกันสามารถทำงานร่วมกันได้ ยกตัวอย่างเช่น ตัวรับรู้ ตัวกระตุ้น เป็นต้น โดยการพัฒนาเอ็นไอพีเป็นตัวกลางในการประสานข้อมูลระหว่างหน่วยเก็บข้อมูลของ IEEE1888 และคลังเก็บข้อมูลของ ETSI M2M แบบเวลาจริงโดยการใช้โพรโทคอล TRAP และวิธี NOTIFY ในการรับข้อมูลโดยการจดทะเบียนการแจ้งเตือนข้อมูล และการใช้โพรโทคอล WRITE และวิธี CREATE ในการส่งข้อมูลตามมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M ตามลำดับ นอกจากนี้ได้มีการพัฒนาเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M บนบอร์ดสำเร็จรูป Raspberry Pi B+ โดยใช้แพลตฟอร์ม node.js ซึ่งสื่อสารกับตัวรับรู้ และตัวกระตุ้นผ่านโมดูลไร้สาย ZigBee

การทดสอบการประสานข้อมูลแบบคาบเวลาเปรียบเทียบกับการประสานข้อมูลแบบเวลาจริงโดยใช้ข้อมูลตัวรับรู้การเคลื่อนไหวของคนจำนวน 5 โนด พบว่า ภาระปริมาณงานทั้งหมดเท่ากับ 0.30 - 0.53 กิโลบิตต่อวินาที และ 0.56 - 0.60 กิโลบิตต่อวินาทีต่อโนดตัวรับรู้ ตามลำดับ โดยสามารถประมาณภาระปริมาณงานทั้งหมดในการประสานข้อมูลของตัวรับรู้ทั้งหมด 688 โนดของโครงการ CU-BEMS เท่ากับ 0.36 และ 0.41 เมกะบิตต่อวินาที สำหรับการประสานข้อมูลทั้ง 2 กรณี ตามลำดับ ซึ่งอยู่ในช่วงที่ยอมรับได้

การทดสอบระบบการทำงานร่วมกันแบบเวลาจริงระหว่างเกตเวย์ทั้ง 2 มาตรฐาน โดยมีส่วนประกอบของระบบประกอบด้วย หน่วยเก็บข้อมูลของ IEEE1888, เกตเวย์ของ IEEE1888, โครงข่ายของ ETSI M2M และเกตเวย์ของ ETSI M2M ผลการทดสอบยืนยันว่าระบบสามารถถ่ายโอนข้อมูลระหว่างเกตเวย์ของ IEEE1888 และเกตเวย์ของ ETSI M2M แบบเวลาจริงได้ ในกรณีของการทดสอบด้วยข้อมูลตัวรับรู้เลียนแบบพบว่าการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M สามารถรองรับตัวรับรู้มากที่สุด 500 โนดที่เวลาประวิงเฉลี่ย 305 มิลลิวินาที มีคอขวดของการถ่ายโอนข้อมูลอยู่ที่โครงข่ายของ ETSI M2M ที่ทดสอบดำเนินการบนคอมพิวเตอร์ส่วนบุคคล และการถ่ายโอนข้อมูลจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 สามารถรองรับตัวรับรู้มากที่สุด 115 โนดที่เวลาประวิงเฉลี่ย 3,561 มิลลิวินาที มีคอขวดของการถ่ายโอนข้อมูลอยู่ที่เกตเวย์ของ IEEE1888 ที่ทดสอบดำเนินการบนแพลตฟอร์ม Raspberry Pi B+ โดยจำนวนโนดตัวรับรู้สูงสุดที่ระบบรองรับได้นั้นขึ้นกับฮาร์ดแวร์ของส่วนประกอบในระบบ ในส่วนของการทดสอบด้วยข้อมูลตัวรับรู้ในโครงการ CU-BEMS บริเวณห้องปฏิบัติการวิจัยสาขาวิชาวิศวกรรมไฟฟ้าสื่อสารที่มีจำนวนโนดตัวรับรู้ต่อเกตเวย์มากที่สุดในโครงการ โดยมีจำนวนโนดตัวรับรู้ทั้งหมด 112 โนด พบว่าการถ่ายโอนข้อมูลจากเกตเวย์ของ IEEE1888 ไปยังเกตเวย์ของ ETSI M2M และจากเกตเวย์ของ ETSI M2M ไปยังเกตเวย์ของ IEEE1888 ใช้เวลาประวิงเฉลี่ยเท่ากับ 114 มิลลิวินาที และ 2,306 มิลลิวินาที ตามลำดับ ซึ่งเป็นเวลาประวิงเฉลี่ยสูงสุดเมื่อต้องการนำระบบนี้มาใช้กับโครงการ CU-BEMS โดยระบบที่นำเสนอสามารถนำไปประยุกต์ใช้กับการควบคุมอุปกรณ์ตามเหตุการณ์ (event-based) เช่น การเปิด

ปิดหลอดไฟ สัญญาณเตือนฉุกเฉิน เป็นต้น ซึ่งความสามารถในการถ่ายโอนข้อมูลระหว่างเกตเวย์ทั้ง 2 มาตรฐานขึ้นกับจำนวนตัวรับรู้ที่ส่งข้อมูลในระบบ

6.2 ข้อเสนอแนะ

การจำลองโปรแกรมที่พัฒนาขึ้นให้อยู่ในรูปแบบของฟังก์ชันโครงข่ายเสมือน (virtual network function, VNF) ที่ใช้เอ็นไอพี โดยทดลองติดตั้งที่แพลตฟอร์มคลาวด์บนพื้นฐาน Openstack เพื่อสามารถใช้เน็ทเวิร์คอินเตอร์เวิร์กกิงพร็อกซีที่พัฒนาขึ้นร่วมกับระบบของคลาวด์ IoT ได้ต่อไปเพื่อเพิ่มความสามารถในการรองรับการขยายระบบได้ในอนาคต นอกจากนี้พัฒนาระบบการทำงานร่วมกันระหว่างมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M ให้สามารถทำงานร่วมกับโพรโทคอลอื่น ๆ เช่น Echonet-lite, MQTT เป็นต้น เพื่อเพิ่มความสามารถในการรองรับการทำงานร่วมกับมาตรฐานหรือ โพรโทคอลอื่น ๆ ได้นอกเหนือจากมาตรฐาน IEEE1888 และมาตรฐาน ETSI M2M

รายการอ้างอิง

[1] Lowe, G. Driving the Internet of Things. IEEE Design and Test 31 (2014): 22-27.

[2] Evans, D. The Internet of Things : How the next evolution of the Internet is changing everything. Cisco Internet Business Solutions Group (IBSG), 2011.

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

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

[5] Esaki, H., and Ochiai, H. GUTP and IEEE1888 for smart facility system using internet architecture framework. The 1st IEEE Workshop on Holistic Building Intelligence through Sensing Systems (HOBSENSE), 2011.

[6] จุฬาลงกรณ์มหาวิทยาลัย (สถาบันวิจัยพลังงาน). โครงการวิจัยและพัฒนาเทคโนโลยีระบบโครงข่ายไฟฟ้าอัจฉริยะเพื่อบริหารจัดการการใช้พลังงานไฟฟ้าของอาคาร. ใน กองทุนเพื่อส่งเสริมการอนุรักษ์พลังงานสำนักงานนโยบายและแผนพลังงาน, 2557.

[7] 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.

[8] ธนากร อินทสุทธิ์, เชาวน์ดิศ อัศวกุล. การทดสอบระบบการเฝ้าสังเกตสภาพอากาศตามมาตรฐาน IEEE1888 สำหรับระบบการจัดการพลังงานของอาคาร. ใน รายงานประชุมวิชาการงานวิจัย และพัฒนาเชิงประยุกต์ ครั้งที่ 5, 2013.

[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] Inthasut, T., and Aswakul, C. Development and reliability testing of IEEE1888 gateway for ZigBee wireless sensor network in Chulalongkorn university's building energy management system. The 8th International Collaboration Symposium on Information, Production and Systems (ISIPS 2014), 2014.

[11] 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.

[12] 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.

[13] ETSI [Online]. Avialable from : http://www.etsi.org [2014, February].

[14] oneM2M [Online]. Avialable from : http://www.onem2m.org [2014, February].

[15] UNIFI [Online]. Avialable from : http://daad-unifi.org [2014, February].

[16] Technical University of Berlin [Online]. Avialable from : http://www.tu-berlin.de/menue/home/parameter/en [2014, February].

[17] Fraunhofer FOKUS [Online]. Avialable from : http://www.fokus.fraunhofer.de/en [2014, February].

[18] OpenMTC [Online]. Avialable from : http://www.open-mtc.org [2014, February].

[19] Corici, M., Coskun, H., Elmangoush, A., Kurniawan, A., Mao, T., Magedanz, T., and Wahle, S. OpenMTC: Prototyping machine type communication in carrier grade operator networks. The 4th International IEEE Workshop on Open NGN and IMS Testbeds (ONIT 2012), pp.1735–1740, 2012.

[20] Wahle, S., Magedanz, T., and Schulze, F. The OpenMTC framework – M2M solutions for smart cities and the Internet of Things. Demonstration at the 13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2012), 2012.

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

[22] Klinpratum, T., Saivichit, C., Elmangoush, A., and Magedanz, T. Performance of interworking proxy for interconnecting IEEE1888 standard at ETSI M2M platforms. The 2015 International Electrical Engineering Congress (iEECON2015), 2015.

[23] ETSI. Machine-to-Machine communications (M2M): Functional architecture. ETSI TS 102 690 V2.1.1, 2013.

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

[25] CPU Usage on the Raspberry Pi [Online]. Avialable from : http://www.microcasts.tv/episodes/2014/03/20/cpu-usage-on-the-raspberry-pi/ [2015, November].

[26] Mills, D.L. Computer network time synchronization: The network time protocol on earth and in Space. Broken Sound Parkway, NW : CRC Press, 2011.

ภาคผนวก

ก โปรแกรมการร้องขอการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล TRAP

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 1-2: การประกาศตัวแปร http และ uuid เป็นอ็อบเจกต์สำหรับอ้างอิงมอดูลของ http และ node-uuid ตามลำดับ
- บรรทัดที่ 4-19: การประกาศตัวแปร body_trap สำหรับจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลที่เป็นเพย์โหลดสายอักขระ (string) โดยใช้โพรโทคอล SOAP มีรูปแบบภาษาเป็น XML ตามโพรโทคอล TRAP ของมาตรฐาน IEEE1888 ภายในเพย์โหลดมีการระบุไอดีการจดทะเบียน แบบชนิดที่เป็น stream อายุการร้องขอ ยูอาร์ไอของเอ็นไอพีที่ callbackData และ callbackControl นอกจากนี้มีการระบุ point id ที่ต้องการแจ้งเตือนข้อมูล และรูปแบบการแจ้งเตือนเมื่อมีการเปลี่ยนแปลงข้อมูลเวลา
- บรรทัดที่ 21-31: การประกาศตัวแปร postRequest เป็นอ็อบเจกต์สำหรับเป็นส่วนหัว (header) ของ http ภายในอ็อบเจกต์มีการระบุแม่ข่าย (host) เป็นเลขที่อยู่ไอพีปลายทาง คือ หน่วยเก็บข้อมูลของ IEEE1888 พร้อม วิถี (path) และช่องทาง (port) มีการระบุวิธี post สำหรับการส่งข้อมูลของ http และอ็อบเจกต์ headers สำหรับการระบุเนื้อหาเป็นภาษา XML ที่เป็นอักขระแบบ UTF-8 การระบุการกระทำของ SOAP เป็นแบบการสอบถาม และขนาดของเพย์โหลด
- บรรทัดที่ 33: การประกาศตัวแปร req สำหรับเป็นส่วนย่อย (element) ของฟังก์ชัน request ในอ็อบเจกต์ http มีอินพุตอาร์กิวเมนต์ 2 ตัว คือ postRequest และฟังก์ชันที่มีอินพุตอาร์กิวเมนต์เป็นส่วนย่อย res โดยการทำงานของฟังก์ชันแสดงบรรทัดที่ 34-36 สำหรับแสดงข้อมูลที่ได้รับการตอบกลับจากหน่วยเก็บข้อมูลของ IEEE1888
- บรรทัดที่ 34: การประกาศตัวแปร buffer ที่เป็นอักขระว่างสำหรับการบันทึกอักขระที่ได้รับการตอบกลับข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888
- บรรทัดที่ 35: การกำหนดเมื่อส่วนย่อย res มีการรับข้อมูลจะทำฟังก์ชันสำหรับบันทึกข้อมูลที่มีอินพุตอาร์กิวเมนต์เป็นอ็อบเจกต์ของเหตุการณ์ data สำหรับการบันทึกอักขระใน buffer
- บรรทัดที่ 36: การกำหนดเมื่อส่วนย่อย res สิ้นสุดการรับข้อมูลจะทำฟังก์ชันในการแสดงผลของข้อมูลใน buffer
- บรรทัดที่ 38-40: การกำหนดเมื่อส่วนย่อย req มีข้อผิดพลาดในการส่งข้อมูลจะทำฟังก์ชันสำหรับแสดงผลของข้อผิดพลาดที่มีอินพุตอาร์กิวเมนต์เป็นอ็อบเจกต์ของเหตุการณ์ e
- บรรทัดที่ 42: การส่งข้อมูลเพย์โหลด body_trap ของส่วนย่อย req ไปยังหน่วยเก็บข้อมูลของ IEEE1888
- บรรทัดที่ 43: การสิ้นสุดการทำงานของส่วนย่อย req
- บรรทัดที่ 45-58: การใช้ฟังก์ชัน setInterval สำหรับการร้องขอการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลแบบคาบเวลาประกอบด้วยอินพุตอาร์กิวเมนต์ 2 ตัว คือ ฟังก์ชันสำหรับส่ง

และรับข้อมูลทำนองเดียวกับบรรทัดที่ 33-43 และคาบเวลา (มิลลิวินาที) โดยคาบเวลากำหนดโดยตัวแปร the_interval แสดงดังบรรทัดที่ 45

```
1  var http = require('http');
2  var uuid = require('node-uuid');
3  /* trap query body message ---------------------------------*/
4  var body_trap =
5  '<?xml version="1.0" encoding="utf-8"?><soapenv:Envelope
6  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
7  <soapenv:Body><ns2:queryRQ xmlns:ns2="http://soap.fiap.org/">
8  <transport xmlns="http://gutp.jp/fiap/2009/11/">
9  <header><query id="'+uuid.v4()+'" type="stream" ttl="600"
10 callbackData="http://161.200.90.70"
11 callbackControl="http://161.200.90.70">
12 <key id="http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/
13 elevetorfront/z1/sensor1/monitor/pir"
14 attrName="time" trap="changed" />
15 <key id="http://bems.ee.eng.chula.ac.th/eng4/fl13/corridor/
16 elevetorfront/z1/sensor2/monitor/pir"
17 attrName="time" trap="changed" />
18 </query></header>
19 </transport></ns2:queryRQ></soapenv:Body></soapenv:Envelope>';
20 /* options -------------------------------------------------*/
21 var postRequest = {
22       host: "161.200.90.122",
23       path: "/axis2/services/FIAPStorage",
24       port: 80,
25       method: "POST",
26       headers: {
27           'Content-Type': 'text/xml charset=UTF-8',
28           'SOAPAction': 'http://soap.fiap.org/query',
29           'Content-Length': Buffer.byteLength(body_trap)
30       }
31 };
32 /* data reading --------------------------------------------*/
33 var req = http.request( postRequest, function ( res ) {
34   var buffer = "";
35   res.on( "data", function( data ) { buffer = buffer + data; } );
36   res.on( "end", function() { console.log( buffer ); } );
37 });
38 req.on('error', function(e) {
39   console.log('problem with request: ' + e.message);
40 });
41 /* trap query message writing ------------------------------*/
42 req.write( body_trap );
43 req.end();
```

```
44 // loop for sending trap query
45 var minutes = 10, the_interval = minutes * 60 * 1000;
46 setInterval(function() {
47   var 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     req.on('error', function(e) {
54       console.log('problem with request: ' + e.message);
55     });
56   req.write( body_trap );
57   req.end();
58 }, the_interval);
```

ข โปรแกรมการรับการตอบกลับข้อมูลตัวรับรู้จากหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล TRAP

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 1-2: การประกาศตัวแปรอ้างอิงถึงส่วนจำเพาะของ http และ xml2js
- บรรทัดที่ 3: การประกาศตัวแปร trap อ้างอิงถึงส่วนจำเพาะของ trap ซึ่งเป็นโปรแกรมการร้องขอการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล TRAP ดังภาคผนวกที่ ก
- บรรทัดที่ 4: การประกาศตัวแปร export_create อ้างอิงถึงส่วนจำเพาะของ export_create ซึ่งเป็นโปรแกรมการส่งข้อมูลไปยังอาร์เออาร์ด้วยวิธี CREATE ดังภาคผนวก ค สำหรับดำเนินโปรแกรมที่เอ็นไอพี
- บรรทัดที่ 5: การประกาศตัวแปร export_xbee อ้างอิงถึงส่วนจำเพาะของ export_xbee ซึ่งเป็นโปรแกรมการส่งข้อมูลไปยังตัวกระตุ้นด้วย ZigBee ดังภาคผนวก ฉ สำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888
- บรรทัดที่ 7-14: การประกาศตัวแปร body_ok สำหรับตอบกลับ ok ตามมาตรฐาน IEEE1888 ที่เป็นเพย์โหลดสายอักขระโดยใช้โพรโทคอล SOAP มีรูปแบบภาษาเป็น XML
- บรรทัดที่ 17: การประกาศตัวแปร server เป็นส่วนย่อยของฟังก์ชัน createServer ในอ็อบเจกต์ http สำหรับทำหน้าที่เป็นตัวบริการที่มีอินพุตอาร์กิวเมนต์เป็นฟังก์ชันสำหรับรับส่งข้อมูล ภายในฟังก์ชันมีอินพุตอาร์กิวเมนต์ประกอบด้วย ส่วนย่อย req และ res โดยการทำงานของฟังก์ชันแสดงบรรทัดที่ 17-60
- บรรทัดที่ 18: การประกาศตัวแปร buffer ที่เป็นอักขระว่างสำหรับการบันทึกอักขระที่ได้รับการตอบกลับข้อมูลจากหน่วยเก็บข้อมูลของ IEEE1888

- บรรทัดที่ 20: การบันทึกข้อมูลทำนองเดียวกับโปรแกรมการร้องขอการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลภาคผนวกที่ ก บรรทัดที่ 35
- บรรทัดที่ 21: การกำหนดเมื่อส่วนย่อย req สิ้นสุดการรับข้อมูลจะทำฟังก์ชันสำหรับการแจงข้อมูล การตอบกลับ ok และการปรับข้อมูลตามมาตรฐาน ETSI M2M โดยการทำงานของฟังก์ชันแสดงบรรทัดที่ 22-76
- บรรทัดที่ 23-26: การนำข้อมูล buffer มาแปลงข้อมูลจากภาษา XML เป็นภาษา JSON และบันทึกในตัวแปรชื่อ jsonbody
- บรรทัดที่ 28-30: การประกาศตัวแปรที่เป็นแถวลำดับ (array) ประกอบด้วย idc, timec, และ valuec สำหรับบันทึกข้อมูลของ point id ตราเวลา และ ค่า ที่ได้จากการแจงข้อมูล
- บรรทัดที่ 33-41: การแจงข้อมูลของตัวแปร jsonobject เป็น key และ value หลังจากนั้นทำการตรวจสอบเพื่อบันทึกข้อมูลเพียง point id ตราเวลา และค่า ในแถวลำดับของ idc, timec และ valuec ตามลำดับ
- สุดท้ายบรรทัดที่ 43-46 การนำข้อมูลของแถวลำดับ idc เฉพาะข้อมูลที่ทำการแจ้งเตือนบันทึกในแถวลำดับ id
- บรรทัดที่ 52-54: การกำหนดฟังก์ชัน writeHead ของส่วนย่อย res สำหรับเป็นส่วนหัวของ http ที่มีอินพุตอาร์กิวเมนต์ประกอบด้วย รหัสสถานะ 200 และออบเจกต์สำหรับเป็นส่วนหัวมีการระบุเนื้อหาเป็นภาษา XML ที่เป็นอักขระแบบ UTF-8 มีการระบุการกระทำของ SOAP ที่เป็นแบบข้อมูล และขนาดของเพย์โหลด
- บรรทัดที่ 55: การสิ้นสุดการทำงานของส่วนย่อย res โดยการส่งข้อมูลเพย์โหลด body_ok สำหรับการตอบกลับ ok ไปยังหน่วยเก็บข้อมูลของ IEEE1888
- บรรทัดที่ 56-71: การปรับข้อมูลตามมาตรฐาน ETSI M2M ประกอบด้วย ชื่อ cotainerId และค่าของอุณหภูมิ ความชื้นสัมพัทธ์ แสง การเคลื่อนไหวของคน และเวลาที่เก็บในรูปแบบ JSON ในตัวแปรชื่อ json_object_data
- บรรทัดที่ 73: การทำฟังก์ชัน create ในออบเจกต์ export_create ของโปรแกรมภาคผนวก ค ในการส่งข้อมูลไปยังอาร์เออาร์ด้วยวิธี CREATE มีอินพุตอาร์กิวเมนต์ประกอบด้วย cotainer-Id และ json_object_data สำหรับโดยดำเนินโปรแกรมที่เอ็นไอพี
- บรรทัดที่ 76: การทำฟังก์ชัน send ในออบเจกต์ export_xbee ของโปรแกรมภาคผนวก ฉ ในการส่งข้อมูลไปยังตัวกระตุ้นผ่านมอดูลไร้สาย ZigBee มีอินพุตอาร์กิวเมนต์ประกอบด้วย id[0] และ json_object_data.pir.value สำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888
- บรรทัดที่ 80: การกำหนดฟังก์ชัน listen ที่ช่องทาง 80 สำหรับช่องทางการสื่อสารของตัวบริการ

```
1 var http = require('http');
2 var parseString = require('xml2js').parseString;
3 var trap = require('./trap');
```

```
4  var export_create = require('./export_create');//@ETSI M2M network
5  var export_xbee = require('./export_xbee');//if @IEEE1888 gateway
6  /* OK body message ----------------------------------------*/
7  var body_ok =
8  '<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope
9  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
10 <soapenv:Body><ns2:dataRS xmlns:ns2="http://soap.fiap.org/">
11 <transport xmlns="http://gutp.jp/fiap/2009/11/">
12 <header><OK/></header>
13 </transport></ns2:dataRS></soapenv:Body>
14 </soapenv:Envelope>';
15
16 /* server creation ----------------------------------------*/
17 var server = http.createServer(function (req, res) {
18   var buffer = "";
19     /** data reading */
20   req.on('data', function (data) { buffer = buffer + data; });
21   req.on('end', function () {
22     /** convert XML to JSON */
23     parseString(buffer, function (err, result) {
24     jsonbody = JSON.stringify(result);  //convert an object in to a
          JSON text
25     return jsonbody;
26   })
27     try {
28   var idc=[];
29   var timec=[];
30   var valuec=[];
31             /** parsing json */
32   JSON.parse(jsonbody, function (key, value) {
33     if (key==='id'&& value[0]==='h'){
34       idc.push(value);
35     }
36     else if (key==='time'){
37       timec.push(value);
38     }
39     else if (key==='_'){
40       valuec.push(value);
41     }
42     });
43     var id=[];
44     for (var i=idc.length - valuec.length; i < idc.length; i++){
45       id.push(idc[i]);
46     }
47     } catch (er) {
48       console.log(er);
```

```
49      }
50
51        /** ok message sending*/
52      res.writeHead(200, {  'Content-Type': 'text/xml charset=UTF-8',
53        'SOAPAction': 'http://soap.fiap.org/data',
54        'Content-Length': Buffer.byteLength(body_ok) });
55      res.end(body_ok);
56      var time=moment().format();
57      if (id.length == 4){
58      var parseId = id[0].split('/');
59      var containerId = parseId.slice(parseId.length-6,parseId.length
            -1).join('_');
60      var motion=valuec[0];
61      var data_illuminance=valuec[1];
62      var data_temperature=valuec[2];
63      var data_humidity=valuec[3];
64
65      var json_object_data = { "illuminance": {"value":
            data_illuminance}, "temperature": {"value": data_temperature
            }, "humidity": {"value": data_humidity}, "pir": {"value":
            motion}, "timestamp": time };
66      }else{
67      var parseId = id[0].split('/');
68      var containerId = parseId.slice(parseId.length-6,parseId.length
            -1).join('_');
69      var motion=valuec[0];
70      var json_object_data = {"pir": {"value": motion}, "timestamp":
            time };
71      }
72        /*sending data to NRAR by CREATE method*/
73       export_create.create(containerId, json_object_data);//if @ETSI
            M2M network
74
75        /*sending data to actuator by xbee*/
76      //export_xbee.send(id[0], json_object_data.pir.value);//if
            @IEEE1888 gateway
77    });
78  });
79  /* server running-----------------------------------------*/
80  server.listen(80);
```

ค โปรแกรมการส่งข้อมูลไปยังอาร์เออาร์ตามวิธี CREATE

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 2-21: การประกาศตัวแปรสำหรับอ้างอิงมอดูล ฟังก์ชัน ส่วนย่อย และอ็อบเจกต์
- บรรทัดที่ 23-29: การประกาศตัวแปร config ที่เป็นอ็อบเจกต์สำหรับการกำหนดองค์ประกอบของโปรแกรม
- บรรทัดที่ 35-55: ฟังก์ชัน createAPP สำหรับการสร้างทรัพยากร application โดยกำหนดฟังก์ชัน requestIndication ของส่วนย่อย dIaclient สำหรับส่งข้อมูลด้วยวิธีสื่อสารที่กำหนด มีอินพุตอาร์กิวเมนต์ประกอบด้วย วิธีการสื่อสาร รหัสสถานะ ยูอาร์ไอปลายทาง และลักษณะจำเพาะ ตามลำดับ เมื่อส่วนย่อย dIaclient สร้างทรัพยากร application เรียบร้อย โปรแกรมจะส่งออกฟังก์ชัน create ที่มีอินพุตอาร์กิวเมนต์ประกอบด้วย containerId และอ็อบเจกต์ json_object สำหรับสร้างทรัพยากร container ด้วยฟังก์ชัน pushData
- บรรทัดที่ 57-74: ฟังก์ชัน pushData สำหรับการสร้างทรัพยากร container ด้วยวิธี CREATE เมื่อมีการสร้างทรัพยากร container เรียบร้อย โปรแกรมจะสร้างทรัพยากร contentInstance ด้วยฟังก์ชัน addContentInstance สำหรับบันทึกข้อมูล
- บรรทัดที่ 75-89: ฟังก์ชัน addContentInstance สำหรับการสร้างทรัพยากร contentInstance ด้วยวิธี CREATE มีอินพุตอาร์กิวเมนต์ประกอบด้วย containerID และ data โดยข้อมูลตัวรับรู้ถูกเข้ารหัสเป็นแบบ base64
- บรรทัดที่ 92: การเริ่มฟังก์ชัน createAPP สำหรับสร้าง application เป็นการเริ่มต้นส่งข้อมูล

```
1  /* external modules ------------------------------------- */
2  var os = require('os');
3  /* internal modules ------------------------------------- */
4  var NIP = require('../lib/NIP'); // network IP library @network ETSI
       M2M
5  //var GIP = require('../lib/NGIP'); // gateway IP library if @
      gateway ETSI M2M (*change nscl to gscl)
6  var openmtc = require('openmtc');
7
8  /* configuration ---------------------------------------- */
9  /** configuration file for NSCL */
10 var nscl = openmtc.config_nscl.scl;
11 /** dIa interface */
12 var XIA = openmtc.interfaces.xIa;
13
14 /* create client for dIa interface ----------------------- */
15 /** generic communication module */
16 var httpHandler = new HttpClient();
17 /** dIa interface */
18 var dIaClient = new XIA(nscl.dia.uri, httpHandler);
19
20 /** logging */
21 var logger = openmtc.config_nscl.log4js.getLogger('[nip]');
22 /** configuration of our app */
```

```
23 var config = {
24   host: os.hostname(),
25   port: '9810',
26   appId: 'NIPA_IEEE1888', // appID for this application
27   maxNrOfInstances: undefined,
28   scanInterval: 50
29 };
30 /* variables ------------------------------------------------ */
31 /** the application */
32 var appID = config.appId;
33 var containers = [];
34
35 function createAPP() {
36   /* register at GSCL */
37 // create APPLICATION
38   dIaClient.requestIndication("CREATE", null, nscl.dia.uri + '/
       applications', {
39     application: {appId: appID}}).on('STATUS_CREATED', function () {
40     'use strict';
41     logger.info("APP CREATED");
42     exports.create = function(containerId, json_object){
43         pushData(containerId,json_object);
44     };
45     }).on("ERROR", function(data) {
46     if (data == 409) {
47         logger.info("App already exists. Assuming that's fine.'");
48         exports.create = function (containerId, json_object){
49             pushData(containerId, json_object);
50         };
51     }
52     else {
53         logger.error("Error creating app: " + data);
54     });
55 };
56
57 function pushData(containerID, data) {
58   'use strict';
59   logger.debug("NIP_IEEE1888 with ID:" + containerID + " pushing
       data: ");
60   logger.debug(data);
61
62   if (containers.indexOf(containerID) === -1) {
63     /** create containers and contentInstance*/
64     dIaClient.requestIndication('CREATE', null, nscl.dia.uri +'/
         applications/' + appID + '/containers', {
```

```
65      container: {id: containerID}}).on('STATUS_CREATED', function ()
            {
66          logger.info("CONTAINER CREATED");
67          addContentInstance(containerID, data);
68          containers.push(containerID);
69        });
70    } else {
71          /** create contentInstance only*/
72          addContentInstance(containerID, data);
73    }
74  }
75  function addContentInstance(containerID, data){
76      'use strict';
77      dIaClient.requestIndication('CREATE', null,
78      nscl.dia.uri + '/applications/' + appID + '/containers/' +
79        containerID + '/contentInstances', {
80        contentInstance: {
81         content: {
82          $t: new Buffer(JSON.stringify(data)).toString('base64'), //
                convert to base64
83          contentType: 'application/json'
84         }
85        }
86      }).on('STATUS_CREATED', function (data) {
87        logger.info("CONTENTINSTANCE CREATED");
88        });
89  }
90
91  /* create APP ------------------------------------------- */
92  createAPP();
```

ง โปรแกรมการส่งติดตามข้อมูล และรับการตอบกลับข้อมูลตัวรับรู้จากอาร์เออาร์ตามวิธี CREATE และ NOTIFY

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 2-21: การประกาศตัวแปรสำหรับอ้างอิงมอดูล ฟังก์ชัน ส่วนย่อย และอ็อบเจกต์
- บรรทัดที่ 22-27: การประกาศตัวแปร config ที่เป็นอ็อบเจกต์สำหรับการกำหนดองค์ประกอบของโปรแกรม
- บรรทัดที่ 29: การประกาศตัวแปร targetApplicationID เป็นชื่อทรัพยากร application ที่ต้องการติดตามข้อมูล
- บรรทัดที่ 31: การประกาศตัวแปร contactURI เป็นยูอาร์ไอสำหรับรับการรับแจ้งเตือนข้อมูล

- บรรทัดที่ 36-44: การกำหนดองค์ประกอบของ app สำหรับโปรแกรมประยุกต์เว็บ (web application)
- บรรทัดที่ 48-60: ฟังก์ชัน getRepresentation และ getNotificationData สำหรับการถอดรหัสข้อมูลจาก base64 เป็นสายอักขระ UTF8 ที่ได้จากการรับแจ้งเตือนข้อมูล
- บรรทัดที่ 67-90: ฟังก์ชัน handleContent มีอินพุตอาร์กิวเมนต์ประกอบด้วย gscl, container และ content สำหรับแจงข้อมูลพร้อมกับปรับข้อมูลตามมาตรฐาน IEEE1888 ประกอบด้วย ชื่อ point id และข้อมูลของค่าอุณหภูมิ ค่าความชื้นสัมพัทธ์ ค่าแสง ค่าการเคลื่อนไหวของคน และเวลา โดยบันทึกในตัวแปรแถวลำดับ fiap_element และ pointset ซึ่งเป็นอินพุตอาร์กิวเมนต์ของฟังก์ชัน write ในออบเจกต์ export_write ของโปรแกรมภาคผนวก จ ในการส่งข้อมูลไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล WRITE สำหรับดำเนินโปรแกรมที่เอ็นไอพี ในส่วนของส่งข้อมูลไปยังตัวกระตุ้นด้วย ZigBee จะทำฟังก์ชัน send ในออบเจกต์ export_xbee ของโปรแกรมภาคผนวก ฉ ที่มีอินพุตอาร์กิวเมนต์ประกอบด้วย container และ content.pir.value สำหรับดำเนินโปรแกรมที่เกตเวย์ของ ETSI M2M
- บรรทัดที่ 107-111 148-152 และ 173-180: การกำหนดส่วนย่อย app โดยใช้ฟังก์ชัน post สำหรับรับการแจ้งเตือนข้อมูลที่มีอินพุตอาร์กิวเมนต์ประกอบด้วย notifyPath และฟังก์ชันการรับข้อมูลแจ้งเตือน ภายในฟังก์ชันมีการนำข้อมูลที่ได้รับการแจ้งเตือนเข้าสู่ฟังก์ชัน getNotificationData เพื่อแปลงข้อมูลเป็นสายอักขระ UTF8 และนำตัวแปร representation เป็นอินพุตอาร์กิวเมนต์ของฟังก์ชัน handleContent, handleContainers, handleApplications ตามลำดับ สำหรับการนำข้อมูลไปใช้ต่อในแต่ละฟังก์ชัน และมีการตอบกลับ 200 ด้วยฟังก์ชัน send ของส่วนย่อย res
- บรรทัดที่ 113-118, 153-158 และ 181-186: การส่งข้อความ subscription ติดตามข้อมูลของทรัพยากร contentInstance, container และ application ตามลำดับ ด้วยวิธี CREATE
- บรรทัดที่ 188: การดำเนินการฟังก์ชัน handlegscl มีอินพุตอาร์กิวเมนต์เป็นอ็อบเจกต์ที่ประกอบด้วย id และยูอาร์ไอของ GSCL สำหรับเริ่มการติดตาม และรับการแจ้งเตือนข้อมูล

```
1  /* external modules */
2  var http = require('http');
3  var os = require('os');
4  var express = require('express');
5  var moment = require('moment');
6
7  /* internal modules */
8  // write protcol*/
9  var export_write = require('./export_write');//@ETSI M2M network
10 // send data by zigbee*/
11 //var export_xbee = require('./export_xbee');//if@ETSI M2M gateway
12
13 // provides dIa primitives
14 var openmtc = require('openmtc');
15 var XIA = openmtc.interfaces.xIa;
```

```
16 // maps primitives to http
17 var HttpClient = openmtc.transport.http.client;
18 // configuration file for gscl
19 var gscl = openmtc.config_gscl.scl; //@ETSI M2M network subscribe
       data at GSCL
20 //var nscl = openmtc.config_nscl.scl; //if@ETSI M2M gateway
       subscribe data at NSCL (*change gscl to nscl)
21 var logger = openmtc.config_gscl.log4js.getLogger('[nip]'); //
       logging
22 var config = {
23   host: '161.200.90.70',
24   port: '1000',
25   notificationResource: '/notify',
26   appID: 'nip'
27 };
28 /* we want to subscribe for data from container in application below
        */
29 var targetApplicationID = 'GIPA_XB';
30 /* contactURI used in subscription */
31 var contactURI = 'http://' + config.host + ':' + config.port;
32 /* express */
33 var app = express();
34 var gscls = {};
35 /* express configuration */
36 app.configure(function () {
37   'use strict';
38   app.use(express.bodyParser());
39   app.use(express.cookieParser());
40   app.use(express.session({
41     secret: 'verysecret'
42   }));
43   app.use(express.methodOverride());
44 });
45 /* web server (dIa) */
46 var dIaServer = http.createServer(app);
47 dIaServer.listen(config.port);
48 function getRepresentation(o) {
49   console.log("GetRepresentation");
50   console.log(o);
51   if (o.representation.contentType !== 'application/json') {
52     logger.error('no json.');
53     throw new Exception("no json");
54   }
55   return JSON.parse(new Buffer(o.representation.$t, 'base64').
        toString('utf8'));
56 }
```

```
57 function getNotificationData(req) {
58     console.log("getNotificationData: ");
59   return getRepresentation(req.body.notify);
60 }
61 //create client for dIa interface
62 //generic communication module
63 var httpClient = new HttpClient(config);
64 //dIa interface
65 var dIaClient = new XIA(gscl.uri, httpClient, 'dIa');
66
67 function handleContent(gscl, container, content) {
68 /** data exporting to write protocol in IEEE1888 standard */
69   var parseId = container.id.split('_');
70   var fiap_id_prefix  = 'http://bems.ee.eng.chula.ac.th/eng4/fl13/';
71   var pointset = fiap_id_prefix+parseId[0]+'/'+parseId.slice(1,
        parseId.length-3).join('_')
72                     +'/'+parseId.slice(parseId.length-3,parseId.
                           length-1).join('/')
73   var pir_point_id = pointset + '/' + 'monitor/pir';
74   var ill_point_id = pointset + '/' + 'monitor/illuminance';
75   var temp_point_id = pointset + '/' + 'monitor/temperature';
76   var hum_point_id = pointset + '/' + 'monitor/humidity';
77   var time=moment().format();
78   if(content.temperature!==undefined){
79     var fiap_element = [
80      [pir_point_id, content.pir.value, time],
81      [ill_point_id, content.illuminance.value, time],
82      [temp_point_id, content.temperature.value, time],
83      [hum_point_id, content.humidity.value, time]
84     ];
85   }else{
86     var fiap_element = [[pir_point_id, content.pir.value, time]];
87   }
88     /*sending data to IEEE1888 storage by WRITE protocol*/
89   export_write.write(pointset, fiap_element_pir);//@ETSI M2M network
90     /*sending data to actator by xbee*/
91   //export_xbee.send(container, content.pir.value);//if @ETSI M2M
        gateway
92   console.log("Handled content");
93 }
94
95 function handleContainer(gscl, container) {
96   if (gscl.containers[container.id] !== undefined) {
97     console.log("Container " + container.id + " of " + gscl.sclId +
          " is known.");
98     return;
```

```
99    }
100   console.log("Handling new container " + container.id + " of " +
          gscl.sclId);
101   container.data = [];
102   container.containerId = container.id;
103   container.sclId = gscl.sclId;
104   gscl.containers[container.id] = container;
105   var notifyPath = config.notificationResource + "/" + gscl.sclId +
          "/" + container.id;
106     var notifyUri = contactURI + notifyPath;
107   app.post(notifyPath, function(req, res) {
108     var representation = getNotificationData(req);
109     handleContent(gscl, container, representation);
110     res.send(200);
111   });
112   console.log("subscribe");
113   dIaClient.requestIndication('CREATE', null, gscl.link + '/
          applications/' + targetApplicationID + '/containers/' +
          container.id + '/contentInstances/subscriptions', {
114     subscription: {contact: notifyUri, filterCriteria: {
            attributeAccessor: 'latest/content'}}
115   }).on('STATUS_CREATED', function (data) {
116     'use strict';
117     logger.info('subscribed to content of ' + gscl.sclId + '/' +
            container.id + ' (' + notifyUri + ')');
118   });
119   console.log("retri");
120 }
121
122 function handleContainers(gscl, containers) {
123   console.log("Handling containers: ");
124   console.log(containers.containerCollection.namedReference);
125   containers.containerCollection.namedReference.forEach(function(
          container) {
126     handleContainer(gscl, container);
127   });
128   console.log("Handled containers.");
129 }
130
131 function handleApplications(gscl, applications) {
132   if (gscl.initialized)
133     return;
134     console.log("handling applications: ");
135     console.log(applications);
136   applications.applicationCollection.namedReference.forEach(function
          (application) {
```

```
137     if (application.id == targetApplicationID) {
138       gscl.initialized = true;
139       initgscl(gscl);
140       return false;
141     }
142   });
143 }
144
145 function initgscl(gscl) {
146   var notifyPath = config.notificationResource + "/" + gscl.sclId;
147     var notifyUri = contactURI + notifyPath;
148   app.post(notifyPath, function(req, res) {
149     var representation = getNotificationData(req);
150     handleContainers(gscl, representation.containers);
151     res.send(200);
152   });
153   dIaClient.requestIndication('CREATE', null, gscl.link + '/
        applications/' + targetApplicationID + '/containers/
        subscriptions', {
154     subscription: { contact: notifyUri }
155   }).on('STATUS_CREATED', function (data) {
156     'use strict';
157     logger.info('subscribed to containers of ' + gscl.sclId + ' (' +
          notifyUri + ')');
158   });
159   console.log("get containers for " + gscl.sclId);
160 }
161
162 function handlegscl(gscl) {
163   if (gscls[gscl.sclId] !== undefined) {
164     console.debug("gsclS " + gscl.sclId + " is already known.");
165     return;
166   }
167   console.log("Handling new gscl " + gscl.sclId + " (" + gscl.link +
        ")");
168   gscl.containers = {};
169   gscl.initialized = false;
170   gscls[gscl.sclId] = gscl;
171     var notifyPath = config.notificationResource +  "/" + gscl.sclId
          + "/apps";
172   var notifyUri = contactURI + notifyPath;
173   app.post(notifyPath, function(req, res) {
174         console.log("new app.");
175     var representation = getNotificationData(req);
176         console.log("Have applications representation: ");
177         console.log(representation);
```

```
178      handleApplications(gscl, representation.applications);
179      res.send(200);
180    });
181    dIaClient.requestIndication('CREATE', null, gscl.link + '/
           applications/subscriptions', {
182      subscription: { contact: notifyUri }
183    }).on('STATUS_CREATED', function (data) {
184      'use strict';
185      logger.info('subscribed to applications of ' + gscl.sclId + ' (
             notifyPath=' + notifyPath + " notifyUri=" + notifyUri + ')');
186    });
187  }
188  handlegscl({"sclId": gscl.id, "link": gscl.mid.uri});
```

จ โปรแกรมการส่งข้อมูลตัวรับรู้ไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล WRITE

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 1-2: การประกาศตัวแปรสำหรับการอ้างอิงมอดูล http และ moment
- บรรทัดที่ 3: การประกาศส่งออกฟังก์ชัน write สำหรับส่งข้อมูลตัวรับรู้ไปยังหน่วยเก็บข้อมูลของ IEEE1888 ตามโพรโทคอล WRITE ที่มีอินพุตอาร์กิวเมนต์ 2 ตัว คือ pointset และ fiap โดยการทำงานของฟังก์ชันแสดงบรรทัดที่ 4-40
- บรรทัดที่ 4-7: การประกาศตัวแปร point สำหรับบันทึกข้อมูลที่ได้จาก fiap และจัดให้อยู่ในรูปแบบ XML ตามมาตรฐาน IEEE1888
- บรรทัดที่ 8-15: การประกาศตัวแปร body สำหรับส่งข้อมูลตัวรับรู้ตามโพรโทคอล WRITE ของมาตรฐาน IEEE1888 ที่เป็นเพย์โหลดสายอักขระโดยใช้โพรโทคอล SOAP มีรูปแบบภาษาเป็น XML มีการระบุ poinset และ point
- บรรทัดที่ 17-27: การประกาศตัวแปร postRequest ทำนองเดียวกับบรรทัดที่ 21-31 ของโปรแกรมที่ ก โดยการระบุการกระทำของ SOAP เป็นแบบข้อมูล
- บรรทัดที่ 30-40: การประกาศตัวแปร req เป็นส่วนย่อยของฟังก์ชัน request ในอ็อบเจกต์ http สำหรับการรับการตอบกลับข้อมูล และการส่งข้อมูลเพย์โหลด body ทำนองเดียวกับโปรแกรมการร้องขอการจดทะเบียนเหตุการณ์ในการแจ้งเตือนข้อมูลภาคผนวกที่ ก บรรทัดที่ 33-43

```
1  var http = require('http');
2  var moment = require('moment');
3  exports.write = function (pointset, fiap){
4  var point= [];
5  for (var i = 0; i < fiap.length; i++) {
```

```
6    point = point + '<point id='+'"'+fiap[i][0]+'"'+'><value time="'+
         fiap[i][2]+'">'+fiap[i][1]+'</value></point>'
7  }
8  var body =
9  '<?xml version="1.0" encoding="UTF-8"?>
10 <soapenv:Envelope
11 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
12 <soapenv:Body><ns2:dataRQ xmlns:ns2="http://soap.fiap.org/">
13 <transport xmlns="http://gutp.jp/fiap/2009/11/"><body>
14 <pointSet id="'+pointset+'">'+point+'</pointSet></body>
15 </transport></ns2:dataRQ></soapenv:Body></soapenv:Envelope>';
16     /* options ----------------------------------------------------*/
17 var postRequest = {
18     host: "161.200.90.122",
19     path: "/axis2/services/FIAPStorage",
20     port: 80,
21     method: "POST",
22     headers: {
23       'Content-Type': 'text/xml charset=UTF-8',
24       'SOAPAction': 'http://soap.fiap.org/data',
25       'Content-Length': Buffer.byteLength(body),
26       }
27     };
28
29     /* data reading -----------------------------------------------*/
30 var req = http.request( postRequest, function ( res ) {
31   var buffer = "";
32   res.on( "data", function( data ) { buffer = buffer + data;});
33   res.on( "end", function( data ) { console.log( buffer );});
34 });
35 req.on('error', function(e) {
36   console.log('problem with request: ' + e.message);
37 });
38 /** message sending*/
39 req.write( body_write );
40 req.end();
41 }
```

ฉ โปรแกรมการรับส่งข้อมูลกับตัวรับรู้ และตัวกระตุ้น ZigBee

รายละเอียดการทำงานของโปรแกรมมีดังนี้

- บรรทัดที่ 2-5: การประกาศตัวแปรสำหรับการอ้างอิงมอดูล util, serialport, xbee-api และ moment

- บรรทัดที่ 7: การประกาศตัวแปรอ้างอิงมอดูล export_write สำหรับเรียกใช้ฟังก์ชันการส่งข้อมูลตามโพรโทคอล WRITE ของมาตรฐาน IEEE1888 สำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888
- บรรทัดที่ 8: การประกาศตัวแปรอ้างอิงมอดูล export_create สำหรับเรียกใช้ฟังก์ชันการส่งข้อมูลตามวิธี CREATE ของมาตรฐาน ETSI M2M สำหรับดำเนินโปรแกรมที่เกตเวย์ของ ETSI M2M
- บรรทัดที่ 12: การประกาศตัวแปร C อ้างอิงถึงอ็อบเจกต์ constants ในมอดูล xbee-api
- บรรทัดที่ 13-15: การประกาศตัวแปร xbeeAPI โดยกำหนดโมด API เป็นโมด 1
- บรรทัดที่ 17-25: การประกาศส่วนย่อย serialport อ้างอิงถึงมอดูล serialport สำหรับการเปิดพอร์ตอนุกรม /dev/ttyUSB ที่มีอินพุตอ็อบเจกต์ ประกอบด้วย baud rate เท่ากับ 9600 และ parser ใช้ฟังก์ชัน rawparser ในมอดูล xbeeAPI เมื่อเกิดข้อผิดพลาดจะแสดงข้อความขึ้น
- บรรทัดที่ 27-60: เมื่อส่วนย่อย serialport มีการเปิดพอร์ตอนุกรมจะทำฟังก์ชันในการส่งออกฟังก์ชัน send สำหรับส่งข้อมูลไปยังตัวกระตุ้นผ่านมอดูลไร้สาย ZigBee
- บรรทัดที่ 29-43: การส่งออกฟังก์ชัน send สำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888 ซึ่งมีอินพุตอาร์กิวเมนต์ ประกอบด้วย pointId และ data ภายในฟังก์ชันมีการดำเนินการฟังก์ชัน command_actuator สำหรับการแปลงข้อมูลตัวรับรู้เป็นข้อมูลควบคุม มีการกำหนดตัวแปร frame_obj1 ที่เป็นอ็อบเจกต์สำหรับกำหนดองค์ประกอบของการส่งข้อมูล ข้อมูลกำหนดที่ data และทำการส่งโดยการใช้ฟังก์ชัน write ในส่วนย่อย serialport
- บรรทัดที่ 45-59: การส่งออกฟังก์ชัน send สำหรับดำเนินโปรแกรมที่เกตเวย์ของ ETSI M2M
- บรรทัดที่ 62-100: เมื่อส่วนย่อย xbeeAPI มีการรับข้อมูลจะดำเนินการฟังก์ชันที่มีอินพุตอาร์กิวเมนต์คือ frame
- บรรทัดที่ 65: ดำเนินการฟังก์ชัน addressSelectSensorZone ที่มีอินพุตอาร์กิวเมนต์เป็นตัวแปร remote64 ในอ็อบเจกต์ frame สำหรับแปลงเลขที่อยู่ MAC เป็นชื่อ point id หรือ container
- บรรทัดที่ 67-70: การกำหนดตัวแปรสำหรับยูอาร์ไอที่เป็น point id ของตัวรับรู้แต่ละชนิดสำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888
- บรรทัดที่ 73-76: การดำเนินฟังก์ชัน pir, illuminant, temperature และ humidty ที่มีอินพุตอาร์กิวเมนต์ frame สำหรับการคำนวณเป็นค่าการเคลื่อนไหวของคน ความเข้มแสง อุณหภูมิและความชื้น
- บรรทัดที่ 78: การประกาศตัวแปร time สำหรับสร้างตราเวลาจากฟังก์ชัน format ในมอดูล moment
- บรรทัดที่ 80: การบันทึก point id และค่าตัวรับรู้ของตัวรับรู้แต่ละชนิด และเวลาในแถวลำดับที่ชื่อว่า fiap_element สำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888

- บรรทัดที่ 82-87: การบันทึกค่าของตัวรับรู้แต่ละชนิด และเวลา ในอ็อบเจกต์ที่ชื่อว่า json_object_data ซึ่งเป็นข้อมูลรูปแบบ JSON สำหรับดำเนินโปรแกรมที่เกตเวย์ของ ETSI M2M
- บรรทัดที่ 89: เมื่อมีการรับเพียงข้อมูลของการเคลื่อนไหวของคนจะนำมาคำนวณหาค่าการเคลื่อนไหวของคนจากฟังก์ชัน pir
- บรรทัดที่ 92: การบันทึก point id และค่าตัวรับรู้ของตัวรับรู้การเคลื่อนไหวของคน และเวลาในตัวแปรแถวลำดับที่ชื่อว่า fiap_element สำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888
- บรรทัดที่ 94: การบันทึกค่าของตัวรับรู้การเคลื่อนไหวของคน และเวลา ในอ็อบเจกต์ที่ชื่อว่า json_object_data สำหรับดำเนินโปรแกรมที่เกตเวย์ของ ETSI M2M
- บรรทัดที่ 97: การเรียกใช้ฟังก์ชัน write ในออบเจกต์ export_write ของโปรแกรมภาคผนวก จ มีอินพุตอาร์กิวเมนต์ประกอบด้วย point_id และ fiap_element ในการส่งข้อมูลด้วยโพรโทคอล WRITE ของมาตรฐาน IEEE1888 สำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888
- บรรทัดที่ 98: การเรียกใช้ฟังก์ชัน create ในออบเจกต์ export_create ของโปรแกรมภาคผนวก ค มีอินพุตอาร์กิวเมนต์ประกอบด้วย container_id และ json_object_data ในการส่งข้อมูลด้วยวิธี CREATE ของมาตรฐาน ETSI M2M สำหรับดำเนินโปรแกรมที่เกตเวย์ของ ETSI M2M
- บรรทัดที่ 103-124: ฟังก์ชัน addressSelectSensorZone ที่มีอินพุตอาร์กิวเมนต์เป็นเลขที่อยู่ MAC ของ ZigBee
- บรรทัดที่ 106-109: การแปลงเลขที่อยู่ MAC ของ ZigBee เป็นข้อความสตริงสำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888
- บรรทัดที่ 111-116: การแปลงเลขที่อยู่ MAC ของ ZigBee เป็นข้อความสตริงสำหรับดำเนินโปรแกรมที่เกตเวย์ของ ETSI M2M
- บรรทัดที่ 119-121: การรวมข้อความสตริงในตัวแปร point_id เป็นรูปแบบมาตรฐาน IEEE1888 สำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888
- บรรทัดที่ 123: การรวมข้อความสตริงในตัวแปร container_id เป็นรูปแบบมาตรฐาน ETSI M2M สำหรับดำเนินโปรแกรมที่เกตเวย์ของ ETSI M2M
- บรรทัดที่ 127-142: ฟังก์ชั้น illuminant สำหรับคำนวณความเข้มแสงจากค่าของ analog to digital 1 ในอ็อบเจกต์ frame
- บรรทัดที่ 145-162: ฟังก์ชั้น temperature สำหรับคำนวณอุณหภูมิจากค่าของ analog to digital 2 ในอ็อบเจกต์ frame
- บรรทัดที่ 165-180: ฟังก์ชั้น humidity สำหรับคำนวณความชื้นจากค่าของ analog to digital 3 ในอ็อบเจกต์ frame
- บรรทัดที่ 183-188: ฟังก์ชั้น pir สำหรับกำหนดสถานะการเคลื่อนไหวของคนจากค่าของ digital I/O 4 ในอ็อบเจกต์ frame

- บรรทัดที่ 190-202: ฟังก์ชั้น command_actuator สำหรับปรับข้อมูลตัวรับรู้เป็นข้อมูลควบคุมสำหรับดำเนินโปรแกรมที่เกตเวย์ของ IEEE1888
- บรรทัดที่ 204-216: ฟังก์ชั้น command_actuator สำหรับปรับข้อมูลตัวรับรู้เป็นข้อมูลควบคุมสำหรับดำเนินโปรแกรมที่เกตเวย์ของ ETSI M2M

```
1  /* external modules ---------------------------------------------*/
2  var util = require('util');
3  var SerialPort = require('serialport').SerialPort;
4  var xbee_api = require('xbee-api');
5  var moment = require('moment');
6  /* internal modules ---------------------------------------------*/
7  var export_write = require('./export_write'); //@IEEE1888 gateway
8  //var export_create = require('./export_create'); //@ETSI M2M
       gateway
9
10 var msb, lsb, str_area, str_room, zone, sensor, fiap_id_prefix,
       str_point_id ;
11
12 var C = xbee_api.constants;
13 var xbeeAPI = new xbee_api.XBeeAPI({
14   api_mode: 1 //[1, 2]; 1 is without escaping, 2 is with escaping (
         set ATAP=2)
15 });
16
17 var serialport = new SerialPort("/dev/ttyUSB0", {
18   baudrate: 9600,
19   parser: xbeeAPI.rawParser()
20 },  function(error) {
21      if(error){
22       console.log("INIT ERROR: " + error.message + "\n");
23     return;
24      }
25 });
26
27 serialport.on("open", function() {
28     /** @IEEE1888 gateway */
29   exports.send = function(pointId, data){
30     command_actuator(pointId, data)
31     if (command !== "undefined"){
32       var frame_obj1 = {
33         type: C.FRAME_TYPE.ZIGBEE_TRANSMIT_REQUEST,//xbee_api.
              constants.FRAME_TYPE.ZIGBEE_TRANSMIT_REQUEST
34         id: 0x01, // optional, nextFrameId() is called per default
35         destination64: "0013A200408B8F11",
36         destination16: "2497", // optional, "fffe" is default
```

```
37          broadcastRadius: 0x00, // optional, 0x00 is default
38          options: 0x00, // optional, 0x00 is default
39          data: command // Can either be string or byte array.
40          }
41          serialport.write(xbeeAPI.buildFrame(frame_obj1));  //massage
                writing
42      }
43    }
44      /** @ETSI M2M gateway */
45    /*exports.send = function(container, data){
46      command_actuator(container, data)
47      if (command !== "undefined"){
48           var frame_obj1 = {
49              type: C.FRAME_TYPE.ZIGBEE_TRANSMIT_REQUEST,//xbee_api.
                    constants.FRAME_TYPE.ZIGBEE_TRANSMIT_REQUEST
50              id: 0x01, // optional, nextFrameId() is called per
                    default
51              destination64: "0013A200408B8EFE",
52              destination16: "7F29", // optional, "fffe" is default
53              broadcastRadius: 0x00, // optional, 0x00 is default
54              options: 0x00, // optional, 0x00 is default
55              data: command // Can either be string or byte array.
56          }
57        serialport.write(xbeeAPI.buildFrame(frame_obj1)); //massage
              writing
58      }
59    }*/
60  });
61  // parsing ZigBee frames
62  xbeeAPI.on("frame_object", function(frame) {
63
64    if (frame.receiveOptions == '1' && !frame.data){
65      addressSelectSensorZone (frame.remote64);
66      /** @IEEE1888 gateway */
67        var pir_point_id = point_id+'monitor/pir';
68        var ill_point_id = point_id+'monitor/illuminance';
69        var temp_point_id = point_id+'monitor/temperature';
70        var hum_point_id = point_id+'monitor/humidity';
71  /** data exporting to write protocol in IEEE1888 standard */
72      if(Object.keys(frame.analogSamples).length == 3){
73        pir(frame);
74        illuminant(frame);
75        temperature(frame);
76        humidity(frame);
77
78        var time=moment().format();
```

```
79          /** @IEEE1888 gateway */
80        var fiap_element = [[pir_point_id, motion, time],[ill_point_id
              , data_illuminance, time],[temp_point_id, data_temperature,
               time],[hum_point_id, data_humidity, time]];
81            /** ETSI M2M gateway */
82              /*var json_object_data = {
83                  "illuminance": {"value": data_illuminance},
84                  "temperature": {"value": data_temperature},
85                  "humidity": {"value": data_humidity},
86                  "pir": {"value": motion}, "timestamp": time
87                  };*/
88      }else{
89        pir(frame, point_id);
90        var time=moment().format();
91          /** @IEEE1888 gateway */
92        var fiap_element = [[pir_point_id, motion, time]];
93          /** @ETSI M2M gateway */
94        /*var json_object_data = {"pir": {"value": motion},
             "timestamp": time };*/
95        }
96      }
97      export_write.write(point_id, fiap_element); //@IEEE1888 gateway
98      //export_create.create(container_id, json_object_data); //@ETSI
           M2M gateway
99    };
100 });
101
102 /** mac address to point id  */
103 function addressSelectSensorZone (mac_address){
104   switch(mac_address){
105         /** @IEEE1888 gateway */
106         case "0013a200408b6213": str_area = "north/";str_room = "
              lab_tsrl_dsprl_emrl/";  zone="z1/";  sensor = "sensor1/";
               break;  //node 1
107         case "0013a200408b6205": str_area = "north/";str_room = "
              lab_tsrl_dsprl_emrl/";  zone="z1/";  sensor = "sensor2/";
               break;  //node 2
108         case "0013a200408b61f3": str_area = "north/";str_room = "
              lab_tsrl_dsprl_emrl/";  zone="z1/";  sensor = "sensor3/";
               break;  //node 3
109         case "0013a200408b6202": str_area = "north/";str_room = "
              lab_tsrl_dsprl_emrl/";  zone="z1/";  sensor = "sensor4/";
               break;  //node 4
110         /** @ETSI M2M gateway */
111         /*case "0013a200408b6213": str_area = "north_";str_room = "
              lab_tsrl_dsprl_emrl_";  zone="z1_";  sensor ="sensor1_";
```

```
112         break;  //node 1
113         case "0013a200408b6205": str_area = "north_";str_room = "
               lab_tsrl_dsprl_emrl_";  zone="z1_";  sensor = "sensor2_";
114         break;  //node 2
115         case "0013a200408b61f3": str_area = "north_";str_room = "
               lab_tsrl_dsprl_emrl_";  zone="z1_";  sensor = "sensor3_";
                break;  //node 3
116         case "0013a200408b6202": str_area = "north_";str_room = "
               lab_tsrl_dsprl_emrl_";  zone="z1_";  sensor ="sensor4_";
               break;  //node 4*/
117     }
118     /** @ETSI M2M gateway */
119     fiap_id_prefix  = 'http://bems.ee.eng.chula.ac.th/eng4/fl13/';
120     str_point_id    = str_area + str_room +  zone + sensor;
121     point_id    = fiap_id_prefix + str_point_id
122      /** @ETSI M2M gateway */
123     container_id    = str_area + str_room +  zone + sensor;
124 }
125
126 /** illuminant calculation */
127 function illuminant(frame){
128   id1='illuminant';
129   var value1=frame.analogSamples.AD1;
130   var binvalue1=value1.toString(2);
131   if (binvalue1.length<9){
132     lsb = binvalue1.substring(0,binvalue1.length);
133     msb = 0
134   }else{
135     lsb = binvalue1.substring(binvalue1.length-8,binvalue1.length);
136     msb = binvalue1.substring(0, binvalue1.length-8);
137   };
138   var LightReading = parseInt(lsb,2) + (parseInt(msb,2) * 256);
139   var illuminance = (LightReading/1023.0)*1200.0;
140
141   data_illuminance =parseInt(illuminance)+'.'+ parseInt(illuminance
         * 10) % 10;
142 }
143
144 /** temperature calculation */
145 function temperature(frame){
146   id2='temperature';
147   var value2=frame.analogSamples.AD2
148   var binvalue2=value2.toString(2);
149   lsb = binvalue2.substring(binvalue2.length-8,binvalue2.length);
150   msb = binvalue2.substring(0, binvalue2.length-8);
151   if (binvalue2.length<9){
```

```
152     lsb = binvalue2.substring(0,binvalue2.length);
153     msb = 0
154   }else{
155     lsb = binvalue2.substring(binvalue2.length-8,binvalue2.length);
156     msb = binvalue2.substring(0, binvalue2.length-8);
157   };
158   var TempReading = parseInt(lsb,2) + (parseInt(msb,2) * 256);
159   var temperature = (TempReading/1023.0*1200.0)/10.0 -10.0;
160   data_temperature = parseInt(temperature)+'.'
161 +parseInt(temperature * 10) % 10 +parseInt(temperature*100)%10;
162 }
163
164 /** humidity calculation */
165 function humidity(frame){
166   id3='humidity';
167   var value3=frame.analogSamples.AD3;
168   var binvalue3=value3.toString(2);
169   if (binvalue3.length<9){
170     lsb = binvalue3.substring(0,binvalue3.length);
171     msb = 0
172   }else{
173     lsb = binvalue3.substring(binvalue3.length-8,binvalue3.length);
174     msb = binvalue3.substring(0, binvalue3.length-8);
175   };
176   var HumidityReading = parseInt(lsb,2) + (parseInt(msb,2) * 256);
177   var humidity = (HumidityReading/1023.0)*1200.0;
178   humidity  = ((humidity * 108.2 /33.2)/5000.0-0.16)/0.0062;
179   data_humidity = parseInt(humidity)+'.'+parseInt(humidity * 10) %
        10
180 }
181
182 /** motion calculation */
183 function pir(frame){
184   id4='pir';
185   motion=frame.digitalSamples.DIO4
186   if (motion == 1){ motion = 'ON';}
187   else if (motion == 0){ motion = 'OFF'};
188 }
189 /** actuator command: @IEEE1888 gateway */
190 function command_actuator(pointId, data){
191   switch(pointId+"_"+data){
192     case "http://bems.ee.eng.chula.ac.th/eng4/fl13/north/
          lab_tsrl_dsprl_emrl/z1/sensor1/monitor/pir_OFF":   command =
          "10"; break;  //channel 1
193     case "http://bems.ee.eng.chula.ac.th/eng4/fl13/north/
          lab_tsrl_dsprl_emrl/z1/sensor1/monitor/pir_ON":    command =
```

```
            "11"; break;  //channel 1
194     case "http://bems.ee.eng.chula.ac.th/eng4/fl13/north/
            lab_tsrl_dsprl_emrl/z1/sensor2/monitor/pir_OFF":   command =
            "20"; break;  //channel 2
195     case "http://bems.ee.eng.chula.ac.th/eng4/fl13/north/
            lab_tsrl_dsprl_emrl/z1/sensor2/monitor/pir_ON":    command =
            "21"; break;  //channel 2
196     case "http://bems.ee.eng.chula.ac.th/eng4/fl13/north/
            lab_tsrl_dsprl_emrl/z1/sensor3/monitor/pir_OFF":   command =
            "30"; break;  //channel 3
197     case "http://bems.ee.eng.chula.ac.th/eng4/fl13/north/
            lab_tsrl_dsprl_emrl/z1/sensor3/monitor/pir_ON":    command =
            "31"; break;  //channel 3
198     case "http://bems.ee.eng.chula.ac.th/eng4/fl13/north/
            lab_tsrl_dsprl_emrl/z1/sensor4/monitor/pir_OFF":   command =
            "40"; break;  //channel 4
199     case "http://bems.ee.eng.chula.ac.th/eng4/fl13/north/
            lab_tsrl_dsprl_emrl/z1/sensor4/monitor/pir_ON":    command =
            "41"; break;  //no command 4
200     default : command = "undefined"
201   }
202 }
203 /** actuator command: @ETSI M2M gateway */
204 /*function command_actuator(container, data){
205     switch(container+"_"+data){
206         case "north_lab_tsrl_dsprl_emrl_z1_sensor1_monitor_OFF":
               command = "10";break;  //command 1
207         case "north_lab_tsrl_dsprl_emrl_z1_sensor1_monitor_ON":
               command = "11";break;  //command 1
208         case "north_lab_tsrl_dsprl_emrl_z1_sensor2_monitor_OFF":
               command = "20";break;  //command 2
209         case "north_lab_tsrl_dsprl_emrl_z1_sensor2_monitor_ON":
               command = "21";break;  //command 2
210         case "north_lab_tsrl_dsprl_emrl_z1_sensor3_monitor_OFF":
               command = "30";break;  //command 3
211         case "north_lab_tsrl_dsprl_emrl_z1_sensor3_monitor_ON":
               command = "31";break;  //command 3
212         case "north_lab_tsrl_dsprl_emrl_z1_sensor4_monitor_OFF":
               command = "40";break;  //command 4
213         case "north_lab_tsrl_dsprl_emrl_z1_sensor4_monitor_ON":
               command = "41";break;  //ncommand 4
214         default : command = "undefined"
215     }
216 }*/
```

ประวัติผู้เขียนวิทยานิพนธ์

ณภัทร โกศลวรวัฒนกุล เกิดเมื่อวันจันทร์ที่ 26 กุมภาพันธ์ พ.ศ. 2533 จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิตจากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง คณะวิศวกรรมศาสตร์ สาขาอิเล็กทรอนิกส์ ปีการศึกษา 2554 และได้ศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ที่จุฬาลงกรณ์มหาวิทยาลัย คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมไฟฟ้า กลุ่มวิจัยโครงข่ายโทรคมนาคมและสารสนเทศ ปีการศึกษา 2556

บทความทางวิชาการจากวิทยานิพนธ์

[1] Kosolworrawattanakul, N., Elmangoush, A., Magedanz, T., and Aswakul, C. Development of real-time data synchronization for IEEE1888 and ETSI M2M standards. Workshop on Internet Architecture and Applications (IA 2014), pp.79-84. 2014.