อุปกรณ์รับส่งข้อมูลผ่านเครือข่าย

NETWORK I/O

สถาพร ตันตระกูล

สมสิน ทองไกรรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

อุปกรณ์รับส่งข้อมูลผ่านเครือข่าย

NETWORK I/O

สถาพร ตันตระกูล

สมสิน ทองไกรรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

ปริญญานิพนธ์ปีการศึกษา 2557

ภาควิชาวิศวกรรมศาสตร์คอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง อุปกรณ์รับส่งข้อมูลผ่านเครือข่าย

NETWORK I/O

ผู้จัดทำ

1. นาย สถาพร ตันตระกูล รหัสนักศึกษา 54011312
2. นายสมสิน ทองไกรรัตน์ รหัสนักศึกษา 54011321

..... อาจารย์ที่ปรึกษา

(อาจารย์ วัจนพงศ์ เกษมศิริ)

..... อาจารย์ที่ปรึกษาร่วม

(ผศ. เจริญ วงษ์ชุ่มเย็น)

..... อาจารย์ที่ปรึกษาร่วม

(ดร. ปกรณ์ วัฒนจตุรพร)

อุปกรณ์รับส่งข้อมูลผ่านเครือข่าย

นายสถาพร	ตันตระกูล	54011312
นายสมสิน	ทองไกรรัตน์	54011321
อาจารย์วัจนพงศ์	เกษมศิริ	อาจารย์ที่ปรึกษา
ผศ.เจริญ	วงษ์ชุ่มเย็น	อาจารย์ที่ปรึกษาร่วม
ดร.ปกรณ์	วัฒนจตุรพร	อาจารย์ที่ปรึกษาร่วม

ปีการศึกษา 2557

บทคัดย่อ

การจัดทำโครงงานอุปกรณ์รับส่งข้อมูลผ่านเครือข่ายนี้มีวัตถุประสงค์เพื่อพัฒนาระบบที่สามารถทำงานได้อย่างอัตโนมัติ คิดค้นรูปแบบในการติดตามการทำงานของระบบผ่านอินเตอร์เน็ตที่มีประสิทธิภาพ คิดค้นรูปแบบการติดต่อระหว่างอุปกรณ์ทั้งภายในเครือข่ายเดียวกันและต่างเครือข่ายกัน คิดค้นรูปแบบการควบคุมอุปกรณ์เครื่องใช้ไฟฟ้าผ่านไมโครคอนโทรลเลอร์ และเพื่อให้บุคคลทั่วไปสามารถพัฒนาระบบขึ้นมาใช้ติดตาม ควบคุมและสั่งงานเครื่องใช้ไฟฟ้าต่างๆ ภายในบ้านได้ด้วยตนเองและมีต้นทุนต่ำ โดยโครงงานอุปกรณ์รับส่งข้อมูลผ่านเครือข่ายเป็นการนำไมโครคอนโทรลเลอร์ตระกูล AVR มาสร้างเป็นบอร์ดที่สามารถเชื่อมต่อกับเครื่องใช้ไฟฟ้าหรือเซ็นเซอร์ต่างๆ และยังสามารถติดตามหรือสั่งการผ่านอินเทอร์เน็ตได้ โดยบอร์ด Network I/O แบ่งเป็น 2 ประเภทคือ บอร์ด Network I/O Gateway และบอร์ด Network I/O Node โดยบอร์ด Network I/O Gateway ใช้เชื่อมต่อกับอินเทอร์เน็ตผ่านสายแลนเพื่อให้เชื่อมต่อกับเซิร์ฟเวอร์ได้ และบอร์ด Network I/O Gateway ยังเชื่อมต่อกับบอร์ด Network I/O Node เพื่อรับค่าจากเซ็นเซอร์บนบอร์ด Network I/O Node หรือรับคำสั่งที่ผู้ใช้สั่งงานผ่านอินเทอร์เน็ตและสั่งให้เครื่องใช้ไฟฟ้าทำงานโดยติดต่อสื่อสารผ่านสัญญาณไร้สาย 2.4 GHz สำหรับการติดต่อผ่านอินเทอร์เน็ตของบอร์ด Network I/O Gateway นั้นจะใช้โพรโทคอล MQTT ในการติดต่อสื่อสาร ซึ่งจะภายในเซิร์ฟเวอร์จะมีซอฟท์แวร์ mosquitto ซึ่งเป็น MQTT Broker โดยมีหน้าที่จัดการกับแพ็กเกต MQTT ที่เข้ามา นอกจากนี้ภายในเซิร์ฟเวอร์ยังติดตั้งซอฟท์แวร์ openHAB ซึ่งมีหน้าที่ติดต่อกับซอฟท์แวร์ mosquito เพื่อรับค่าจากเซ็นเซอร์มาแสดงบนแอปพลิเคชั่น และรับคำสั่งที่ผู้ใช้ป้อนผ่านทางแอปพลิเคชั่นเพื่อสั่งการเครื่องใช้ไฟฟ้า

จากผลการดำเนินงานพบว่าโครงงานอุปกรณ์รับส่งข้อมูลผ่านเครือข่ายสามารถนำไปใช้จริงได้ ซึ่งรูปแบบการใช้งานก็ขึ้นอยู่กับผู้ใช้ว่าต้องการนำไปควบคุมเซ็นเซอร์ใดหรือ

เครื่องใช้ไฟฟ้าชนิดใดบ้าง โดยมีข้อจำกัดคือเครื่องใช้ไฟฟ้าหรือเซ็นเซอร์เหล่านั้นต้องเป็นสิ่งที่บอร์ด Network I/O รองรับ สำหรับโครงงานนี้ยังมีบางจุดที่ยังไม่สมบูรณ์และสามารถพัฒนาต่อได้อีกในอนาคต เช่น การเพิ่มระบบรักษาความปลอดภัย หรือการพัฒนารูปแบบของบอร์ดให้ดูทันสมัยน่าใช้งาน ซึ่งผู้ใช้สามารถนำไปประยุกต์ใช้ได้กับ ระบบบ้านอัจฉริยะ ระบบการเกษตรอัจฉริยะ หรือ ระบบรายงานสภาพอากาศสดจากพื้นที่จริง

Network I/O

Mr. Sathaporn	Tantrakoon	54011312
Mr. Somsin	Thongkrairat	54011321
Mr. Watjanapong .	Kasemsiri	Advisor
Asst.Prof. Charoen	Vongchumyen	Co- Advisor
Dr.Pakorn	Watanachaturaporn	Co- Advisor

Academic Year 2014

ABSTRACT

The aims of Network I/O project are to develop the automatic system, to monitor the system via the internet efficiently, to monitor and control the electrical appliances and sensors via the internet, and the last one is to make the low cost electronics board that people can implement it by themselves

The Network I/O project, the board is developed from AVR microcontroller. It can connect, control the electrical appliances and monitor the sensor. There are 2 kinds of Network I/O board i.e. Network I/O Gateway board and Network I/O Node.

The Network I/O Gateway can connect the internet via LAN to communicate with server that installed openHAB and mosquitto (MQTT Broker) for sending the sensor's value received from Network I/O board to the server and show the result on the application and receiving the command from user via application to turn on or turn off the electrical appliances. The Network I/O Gateway can also communicate with the Network I/O Node via wireless signal 2.4 GHz. The Network I/O Node is a board that connect directly with the electrical appliances or sensors.

The result of this project, Network I/O can be used to monitor sensors and control the electrical appliances. The limitation of this project are it cannot use in large area or the area that has many obstruction and it must only use with the compatible electrical appliances or sensors.

The Network I/O project can further development in the future by improving the security and its design to attract users to use it. The Network I/O project is not only used for smart home, it is also used for smart farm or real-time weather report system.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้อย่างดี ด้วยคำแนะนา จากอาจารย์ที่ปรึกษา อาจารย์วัจนพงศ์ เกษมศิริ และอาจารย์ที่ปรึกษาร่วม ผศ.เจริญ วงษ์ชุ่มเย็น และดร.ปกรณ์ วัฒนจตุรพร ที่คอยให้ความช่วยเหลือ กลุ่มของข้าพเจ้าขอขอบคุณอาจารย์ทั้งสามท่านเป็นอย่างสูง

ขอขอบคุณอาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกท่าน ตลอดจนครู อาจารย์ที่ได้ให้ความรู้กับข้าพเจ้าตั้งแต่อดีตจนถึงปัจจุบัน

ขอขอบคุณชุมนุมโรบอท คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้สนับสนุนเครื่องมือ ข้อมูล หนังสือต่างๆ ตลอดจนเพื่อนๆ พี่ๆ น้องๆ ที่ให้ความช่วยเหลือในการทำโครงงานจนสำเร็จ

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัว ผู้ซึ่งเป็นทั้งกาลังใจ แรงบันดาลใจ ความฝัน ตลอดจนการให้ความสนับสนุนในทุกๆ เรื่อง

สถาพร ตันตระกูล
สมสิน ทองไกรรัตน์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญตาราง (ต่อ)

ตาราง **หน้า**

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูป **หน้า**

สารบัญรูป (ต่อ)

รูป **หน้า**

บทที่ 1

บทนำ

1.1. ความสำคัญและที่มาของโครงงาน

ในปัจจุบันระบบ automation หรือระบบควบคุมการทำงานแบบอัตโนมัติถือเป็นระบบที่ช่วยอำนวยความสะดวกในชีวิตประจำวันให้กับมนุษย์ได้เป็นอย่างมาก เช่น ระบบเปิดปิดไฟฟ้าตามความสว่างที่วัดได้ การรดน้ำต้นไม้ตามความชื้นของดิน หรือการปรับเปลี่ยนสีของหลอดไฟตามปฏิทินที่ตั้งไว้ และจากการมาของเทคโนโลยี Internet of Things ที่ทำให้สิ่งต่างๆ สามารถเชื่อมต่อกับอินเทอร์เน็ตได้ จึงทำให้ผู้ใช้สามารถสั่งงานให้อุปกรณ์เหล่านั้นผ่านอินเทอร์เน็ตได้ แต่เมื่อพิจารณาแล้วเทคโนโลยีเหล่านี้มักพบในต่างประเทศ ยังไม่พบในประเทศไทยมากนัก แต่โชคดีที่ในปัจจุบันอุปกรณ์ระบบสมองกลฝังตัวนั้นและไมโครคอนโทรลเลอร์นั้นสามารถเข้าถึงได้ง่ายและมีราคาถูกลงมาโดยเฉพาะบอร์ด Arduino ซึ่งปัจจุบันได้รับความนิยมและสามารถนำมาทำประโยชน์ต่างๆ ทั้งในด้านการเรียนรู้ และการนำไปพัฒนาอุปกรณ์เพื่อใช้งานจริง

ด้วยเหตุนี้กลุ่มของข้าพเจ้าจึงมีความตั้งใจที่จะสร้างอุปกรณ์ที่สามารถนำมาทำระบบ automation โดยใช้ไมโครคอนโทรลเลอร์ ให้ใช้งานง่าย โดยบุคคลทั่วไปสามารถนำไปใช้งานและติดตั้งได้ด้วยตนเองโดยใช้เวลาไม่นาน และไม่จำเป็นต้นมีความรู้ในการเขียนโปรแกรมควบคุมไมโครคอนโทรเลอร์เลย ที่สำคัญคือการออกแบบให้อุปกรณ์นี้มีราคาถูก โดยตั้งชื่ออุปกรณ์ชิ้นนี้ว่า Network I/O หรืออุปกรณ์รับส่งข้อมูลผ่านเครือข่าย

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อพัฒนาระบบที่สามารถทำงานได้อย่างอัตโนมัติ
2) เพื่อคิดค้นรูปแบบในการติดตามการทำงานของระบบผ่านอินเตอร์เน็ตที่มีประสิทธิภาพ
3) เพื่อคิดค้นรูปแบบการติดต่อระหว่างอุปกรณ์ทั้งภายในเครือข่ายเดียวกันและต่างเครือข่ายกัน
4) เพื่อคิดค้นรูปแบบการควบคุมอุปกรณ์เครื่องใช้ไฟฟ้าผ่านไมโครคอนโทรลเลอร์
5) เพื่อให้บุคคลทั่วไปสามารถพัฒนาระบบขึ้นมาใช้ติดตาม ควบคุมและสั่งงานเครื่องใช้ไฟฟ้าต่างๆ ภายในบ้านได้ด้วยตนเอง

1.3 ขอบเขตของโครงงาน

1) บอร์ด Network I/O ต้องถูกติดตั้งภายใต้พื้นที่ที่สามารถใช้งานอินเทอร์เน็ตได้

2) การตั้งค่าต่างๆ ของ Network I/O มีพื้นที่จำกัดขึ้นอยู่กับขนาดของหน่วยความจำบน Network I/O

3) อุปกรณ์ต่างๆ ถูกพัฒนาโดยไมโครคอนโทรลเลอร์ อุปกรณ์ต่อพ่วงต่างๆ (peripheral) ที่จะนำมาใช้ต้องอยู่ภายใต้ข้อกำหนดของโมดูลที่นำมาใช้งาน

4) พื้นที่ที่บอร์ด Network I/O สามารถทำงานร่วมกันได้ต้องอยู่ในพื้นที่ที่จำกัด ต้องไม่ห่างกันมากเกินไป และขึ้นอยู่กับสิ่งกีดขวาง เนื่องจากจะทำให้ไม่สามารถรับส่งข้อมูลและคำสั่งถึงกันได้

1.4 วิธีการดำเนินงาน

1) สำรวจและประเมินความเป็นไปได้ของทฤษฎีต่างๆที่ใช้ในโครงงาน

2) ออกแบบระบบโดยรวมและสร้างสมมุติฐาน

3) สร้างชิ้นงานต้นแบบและทดสอบการทำงานของระบบ รวมทั้งแก้ไขข้อผิดพลาดที่เกิดขึ้น

4) สรุปผลการทำงานและสร้างชิ้นงานและระบบที่สมบูรณ์

5) สร้างผลงานขั้นสุดท้ายและจัดทำเล่มรายงาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้ชิ้นงาน Network I/O ที่เสร็จสมบูรณ์

2) ระบบที่ทำงานร่วมกับ Network I/O

3) ได้ระบบที่สามารถติดตั้งได้ง่าย ผู้ใช้ทั่วไปสามารถนำไปใช้งานได้

4) กระบวนการวิเคราะห์ และการแก้ไขปัญหาต่างๆ

5) ได้รับความรู้ต่างๆ ที่เกี่ยวข้องกับโครงงาน

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บทด้วยกันคือ

1) บทที่ 1 บทนำกล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

2) บทที่ 2 ทฤษฎีที่เกี่ยวข้อง กล่าวถึงทฤษฎีพื้นฐานที่ใช้ในโครงงานซึ่งเนื้อหาหลักๆ ที่เกี่ยวข้องจะเป็นโพรโทคอล MQTT การสื่อสารผ่านสัญญาณไร้สาย 2.4 GHz

3) บทที่ 3 การออกแบบระบบ กล่าวถึงการออกแบบระบบในภาพรวม การทางานของระบบและส่วนติดต่อกับผู้ใช้งาน (แอปพลิเคชั่น และเว็บแอปพลิเคชั่น)

4) บทที่ 4 การทดลอง และผลการทดลอง กล่าวถึงการทำการทดลองเพื่อทดสอบการทำงานของระบบ ทดสอบเพื่อเพิ่มประสิทธิภาพของระบบ และหาข้อผิดพลาดของระบบเพื่อทำการปรับปรุงแก้ไข

5) บทที่ 5 สรุปและแนวทางในการดำเนินงานต่อ กล่าวถึงบทสรุปของโครงงาน วิเคราะห์สิ่งที่ได้รับจากโครงงาน และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อไป

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

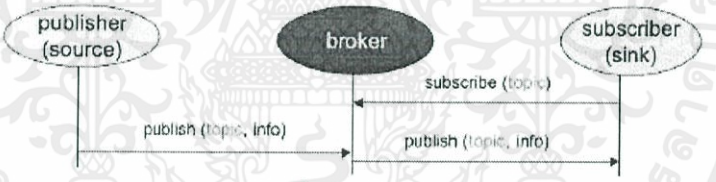
2.1 โพรโทคอล MQTT

โพรโทคอล MQTT หรือ Message Queue Telemetry Transport เป็นโพรโทคอลการส่งข้อความผ่านตัวแทน (broker-based) และใช้ทรัพยากรในการทำงานน้อย (lightweight) ซึ่งถูกออกแบบมาสำหรับงานที่มีข้อจำกัดต่างๆ เช่น ระบบที่มี bandwidth ในการติดต่อสื่อสารต่ำ และไม่มีความน่าเชื่อถือ (unreliable) หรือ ระบบสมองกลฝังตัว (embedded system) ซึ่งมีข้อจำกัดด้านการความเร็วในการประมวลผลและพื้นที่ที่ใช้เก็บข้อมูล

2.1.1 คุณสมบัติของโพรโตคอล MQTT

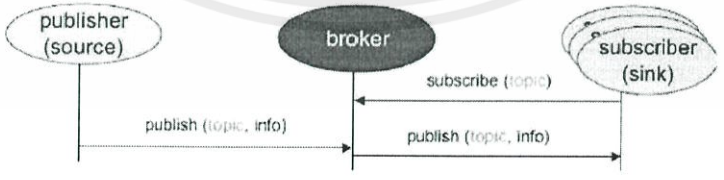
1) ใช้หลักการ publish และ subscribe (ประกาศและติดตาม) ข้อความไปยังต้นทางหรือปลายทางที่ต้องการ โดยมีรูปแบบการติดต่อสื่อสาร 3 รูปแบบ ได้แก่

หนึ่งต่อหนึ่ง (One to One, 1:1)



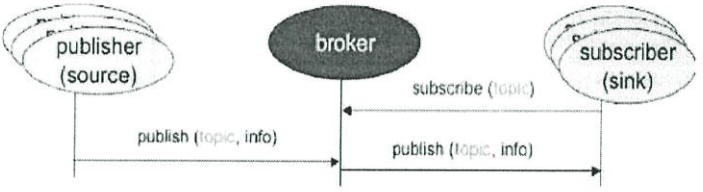
รูป 2.1 การติดต่อสื่อสารแบบหนึ่งต่อหนึ่ง

หนึ่งต่อหลาย (One to Many, 1:M)



รูป 2.2 การติดต่อสื่อสารแบบหนึ่งต่อหลาย

หลายต่อหลาย (Many to Many, M:N)

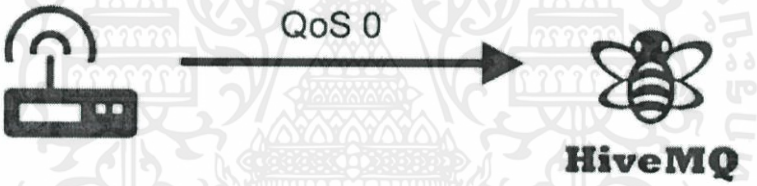


รูป 2.3 การติดต่อสื่อสารแบบหลายต่อหลาย

2) ใช้โพรโทคอล TCP/IP เป็นพื้นฐานในการสื่อสารผ่านระบบเครือข่าย

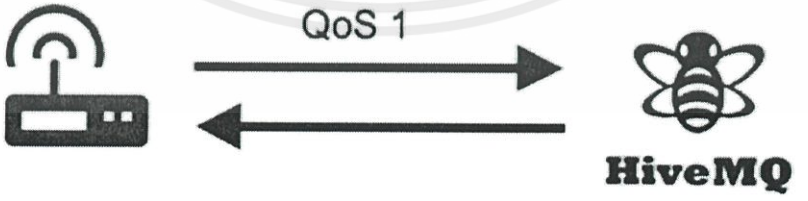
3) มีคุณภาพการบริการ หรือ QoS (Quality of Service) 3 ระดับ ได้แก่

QoS ระดับ 0 หรือ At most once คือ การส่งข้อความผ่านระบบเครือข่ายแบบ best efforts ซึ่งข้อความอาจสูญหายโดยที่ไม่มีการส่งข้อมูลซ้ำ



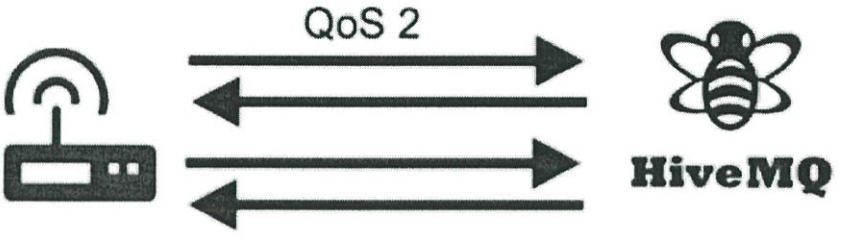
รูป 2.4 การส่งข้อมูลที่มี QoS ระดับ 0

QoS ระดับ 1 หรือ At least once คือ การส่งข้อความที่รับประกันว่าข้อความจะถูกส่งถึงปลายทาง แต่อาจจะมีการส่งข้อมูลซ้ำ



รูป 2.5 การส่งข้อมูลที่มี QoS ระดับ 1

QoS ระดับ 2 หรือ Exactly once คือ การส่งข้อความที่รับประกันว่าข้อความจะถูกส่งถึงปลายทางและมีการส่งเพียงครั้งเดียว



รูป 2.6 การส่งข้อมูลที่มี QoS ระดับ 2

4) มี Overhead ขนาดเล็ก (มีขนาด 2 ไบต์หากเป็นเฮดเดอร์ที่มีความยาวคงที่) ซึ่งจะช่วยลดความหนาแน่นของการจราจรในระบบเครือข่าย

2.1.2 รูปแบบของข้อความในโพรโทคอล MQTT

2.1.2.1 รูปแบบข้อความแบบ Fixed Header

เฮดเดอร์ของข้อความแต่ละข้อความที่ส่งผ่านโพรโทคอล MQTT จะประกอบไปด้วยข้อมูลต่างๆ ดังตาราง 2.1

ตาราง 2.1 รูปแบบของข้อความ

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

โดย ไบต์ที่ 1 ประกอบไปด้วย

1) ฟิลด์ Message Type อยู่ที่บิตที่ 7-4 ใช้สำหรับบอกชนิดของข้อความ โดยมีชนิดต่างๆ ดังตาราง 2.2

ตาราง 2.2 Message Type ชนิดต่างๆ

Mnemonic	Enumeration	Description
Reserved	0	Reserved
CONNECT	1	Client request to connect to Server
CONNACK	2	Connect Acknowledgment

ตาราง 2.2 Message Type ชนิดต่างๆ (ต่อ)

Mnemonic	Enumeration	Description
PUBLISH	3	Publish message
PUBACK	4	Publish Acknowledgment
PUBREC	5	Publish Received (assured delivery part 1)
PUBREL	6	Publish Release (assured delivery part 2)
PUBCOMP	7	Publish Complete (assured delivery part 3)
SUBSCRIBE	8	Client Subscribe request
SUBACK	9	Subscribe Acknowledgment
UNSUBSCRIBE	10	Client Unsubscribe request
UNSUBACK	11	Unsubscribe Acknowledgment
PINGREQ	12	PING Request
PINGRESP	13	PING Response
DISCONNECT	14	Client is Disconnecting
Reserved	15	Reserved

2) ฟิลด์ DUP flag ถ้าหาก flag นี้ถูกกำหนดค่าหมายความว่าไคลเอนต์หรือเซิร์ฟเวอร์พยายามที่จะส่งข้อความ PUBLISH, PUBREL, SUBSCRIBE หรือ UNSUBSCRIBE โดยจะใช้ได้กับข้อความที่มี QoS ระดับที่มากกว่า 0 เท่านั้น และจำเป็นต้องมี acknowledgment นอกจากนี้ฝั่งผู้รับที่รับข้อความที่ DUP flag ถูกกำหนดค่าจะต้องตรวจสอบได้ว่ามีการรับข้อความซ้ำหรือไม่

3) ฟิลด์ QoS Level ใช้ในการรับประกันการส่ง PUBLISH message ซึ่งระดับของ QoS เป็นดังตาราง 2.3

ตาราง 2.3 ระดับของคุณภาพในการให้บริการ

QoS value	bit 2	bit 1	Description		
0	0	0	At most once	Fire and Forget	<=1
1	0	1	At least once	Acknowledged delivery	>=1
2	1	0	Exactly once	Assured delivery	=1
3	1	1	Reserved		

4) ฟิลด์ RETAIN ถูกใช้ใน PUBLISH message เท่านั้น โดยเมื่อไคลเอนต์ต้องการส่ง PUBLISH message ไปยังเซิร์ฟเวอร์ ฟิลด์ RETAIN จะถูกกำหนดค่าเป็น '1' ซึ่งทำให้เซิร์ฟเวอร์ต้องทำการเก็บข้อความที่ไคลเอนต์ส่งไปเมื่อได้รับข้อความแล้ว และเมื่อมีไคลเอนต์ใหม่ต้องการ subscribe topic ดังกล่าว retained message ล่าสุดจะถูกส่งไปยัง subscriber นั้น โดยข้อความนั้นมีการกำหนดค่าretain flag ให้เป็น 1 ด้วย แต่หากไม่มีการกำหนดค่าRetain flag ให้เป็น 1 จะไม่มีการส่ง retained message ไปยังไคลเอนต์ที่มา subscribe topic นั้น สำหรับ retained message จะถูกเก็บไว้ในเซิร์ฟเวอร์จนกว่าเซิร์ฟเวอร์จะ restart นอกจากนี้เซิร์ฟเวอร์จะทำการลบ retained message เมื่อได้รับข้อความที่มีขนาดเท่ากับศูนย์

สำหรับ ไบต์ที่ 2 จะบอกถึงจำนวนไบต์ที่เหลือของข้อมูล ซึ่งประกอบไปด้วยข้อมูล 2 ประเภท ได้แก่ variable header และ payload โดยการเก็บข้อมูลเป็นแบบ Big-endian หรือเรียงจากไบต์ที่มีนัยสำคัญสูงไปหาไบต์ที่มีนัยสำคัญต่ำ ซึ่งข้อมูลในฟิลด์นี้จะใช้ข้อมูล 1 ไบต์สำหรับบอกความยาวของข้อความ 127 ไบต์ โดยแต่ละไบต์จะใช้ 7 บิตแรกเพื่อเก็บข้อมูล และใช้บิตสุดท้ายเพื่อบอกว่ามีข้อมูลต่อจากไบต์นี้หรือไม่ ซึ่งบิตที่ 8 นี้ถูกเรียกว่า "continuation bit" ซึ่งโพรโทคอล MQTT จะสามารถใช้จำนวนไบต์ในการแสดงความยาวของข้อความมากที่สุด 4 ไบต์ หรือแทนขนาดความยาวของข้อความได้ 268,435,456 ไบต์ หรือ 256 MB ดังตาราง 2.4

ตาราง 2.4 การแทนขนาดความยาวของข้อความ

Digits	From	To
1	0 (0x00)	127 (0x7F)
2	128 (0x80, 0x01)	16 383 (0xFF, 0x7F)
3	16 384 (0x80, 0x80, 0x01)	2 097 151 (0xFF, 0xFF, 0x7F)
4	2 097 152 (0x80, 0x80, 0x80, 0x01)	268 435 455 (0xFF, 0xFF, 0xFF, 0x7F)

2.1.2.2 รูปแบบข้อความ Remaining Header

สำหรับ remaining length นี้เป็นส่วนหนึ่งของ fixed header ซึ่งได้แก่

1) เฮดเดอร์ที่มีขนาดไม่คงที่ (Variable header) ข้อความคำสั่ง MQTT บางข้อความจะบรรจุ variable header ไว้ระหว่าง fixed header และ payload

2) ชื่อของโพรโทคอล (Protocol name) ปรากฏใน variable header ของ MQTT CONNECT message โดยเป็นข้อมูลที่ได้รับการเข้ารหัสข้อความแบบ UTF-8 ซึ่ง Protocol name จะถูกกำหนดให้เป็น "MQIsdp"

3) เวอร์ชันของโพรโทคอล (Protocol version) ปรากฏใน variable header ของ MQTT CONNECT message ปรากฏใน variable header ของ MQTT CONNECT message เป็นฟิลด์ที่มีชนิดข้อมูลแบบไม่มีเครื่องหมายขนาด 8-bit ใช้แสดงเวอร์ชั่นของโพรโทคอล MQTT ที่ไคลเอนต์ใช้งาน โดยในโครงงานนี้ใช้โพรโทคอล MQTT เวอร์ชั่น 3.1 ดังนั้นจึงต้องกำหนดให้ protocol version เป็น 3 ซึ่งจะได้ฟิลด์ลักษณะดังตาราง 2.5

ตาราง 2.5 ฟิลด์ที่เก็บเวอร์ชันของโพรโทคอล MQTT

bit	7	6	5	4	3	2	1	0
	Protocol Version							
	0	0	0	0	0	0	1	1

CONNECT flags Clean session, Will, Will QoS, Will Retain, User Name และ Password flag ปรากฏอยู่ใน variable header ของ CONNECT message ซึ่ง บิต 0 ของ CONNECT flags จะไม่ถูกใช้งาน ซึ่ง CONNECT flags มีลักษณะดังตาราง 2.6

ตาราง 2.6 ฟิลด์ต่างๆของ variable header

bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved

Variable header ประกอบด้วยฟิลด์ต่างๆ ดังนี้

1) Clean session flag อยู่ในตำแหน่งบิตที่ 1 ของไบต์ CONNECT flags ถ้าไม่กำหนดค่า (ค่าเป็น 0) เซิร์ฟเวอร์จะเก็บ subscriptions ของไคลเอนต์ไว้หลังจากที่ไคลเอนต์ยกเลิกการติดต่อไปแล้ว แต่ถ้าหากกำหนดค่าเป็น 1เซิร์ฟเวอร์จะทำการเคลียร์ข้อมูลที่เก็บไว้ก่อนหน้าทิ้งไป

2) Will flag อยู่ตำแหน่งบิตที่ 2 ของไบต์ CONNECT flags โดย Will flag จะถูกส่งจากเซิร์ฟเวอร์ในกรณีที่ เกิดข้อผิดพลาดที่เกี่ยวข้องกับ I/O ของเซิร์ฟเวอร์หรือการเชื่อมต่อของฝั่งไคลเอนต์ล้มเหลว โดยการส่ง will message ของเซิร์ฟเวอร์จะไม่ถูกมองเป็นการส่ง DISCONNECT message ของฝั่งไคลเอนต์ เมื่อ Will flag ถูกกำหนดค่าแล้ว ใน ไบต์ Connect flags จะต้องทำการกำหนด Will QoS และ Will Retain ด้วย นอกจากนี้ Will Topic และ Will Message จะต้องปรากฏใน payload ด้วย

3) Will QoS อยู่ในตำแหน่งบิตที่ 3 และ 4 ของ CONNECT Flags โดย Will QoS ใช้สำหรับกำหนดระดับของการรับประกันการส่ง Will message ในกรณีที่การเชื่อมต่อของไคลเอนต์ถูกยกเลิกแบบไม่คาดคิด โดย Will Message จะอยู่ใน payload ของ CONNECT message

4) Will Retain Flag อยู่ในตำแหน่งบิตที่ 5 ของไบต์ CONNECT Flags ซึ่งถ้าหาก Will Retain Flag ถูกกำหนดค่าหมายความว่าเซิร์ฟเวอร์จะต้องทำการเก็บ Will Message เอาไว้ โดยการที่จะกำหนดค่าWill Retain Flag ได้นั้น Will Flag จะต้องถูกกำหนดค่าก่อน

5) User name และ Password Flags อยู่ในตำแหน่งบิตที่ 6 และ 7 ของ CONNECT Flags ถ้าหากทั้งสองบิดนี้ถูกกำหนดค่าหมายถึงต้องการให้มีการระบุ username และ password ในการสร้างการเชื่อมต่อ ซึ่งค่าของ username และ password จะอยู่ใน payload ของ CONNECT message

6) Keep Alive Timer มีขนาด 16 บิต โดยปรากฏในส่วนของ Variable header ของ MQTT CONNECT Message ค่าของ Keep Alive Timer มีหน่วยเป็นวินาที สามารถกำหนดค่าได้มากที่สุดเป็นระยะเวลา 18 ชั่วโมง โดยเป็นค่าที่บอกถึงช่วงเวลาที่มากที่สุดที่จะมีการส่งข้อมูลจากไคลเอนต์ถ้าไคลเอนต์ไม่ส่งข้อความมาในระยะเวลาที่กำหนดไว้ แสดงว่าการเชื่อมต่ออาจมีปัญหา

ตาราง 2.7 ฟิลด์สำหรับกำหนด Keep Alive Timer

bit	7	6	5	4	3	2	1	0
	Keep Alive MSB							
	Keep Alive LSB							

Connect return code ปรากฏอยู่ในส่วน variable header ของ CONACK Message ใช้สำหรับรายงานผลการเชื่อมต่อ โดยมีผลต่างๆดังตาราง 2.8

ตาราง 2.8 Return code แสดงผลการเชื่อมต่อ

Enumeration	HEX	Meaning
0	0x00	Connection Accepted
1	0x01	Connection Refused: unacceptable protocol version
2	0x02	Connection Refused: identifier rejected
3	0x03	Connection Refused:เซิร์ฟเวอร์unavailable
4	0x04	Connection Refused: bad user name or password
5	0x05	Connection Refused: not authorized
6-255		Reserved for future use

ตาราง 2.9 รูปแบบฟิลด์ที่ใช้กำหนดค่า Connect return code

bit	7	6	5	4	3	2	1	0
	Return Code							

Topic name มีขนาด 8 บิต โดยปรากฏอยู่ใน variable header ของ MQTT PUBLISH message เพื่อใช้ระบุช่องทางของข้อมูลใน payload ที่ถูก publish โดยข้อความใน topic name จะถูกเข้ารหัสแบบ UTF-8 และสามารถตั้ง topic name ได้ขนาดสูงสุด 32,767 ตัวอักษร

2.1.2.3 รูปแบบข้อความใน Payload

ภายใน Payload ประกอบไปด้วย

1) CONNECT โดย payload จะต้องมีตัวอักษรที่เป็น UTF-8 อย่างน้อย 1 ตัวอักษร และจะต้องเป็นค่าเฉพาะไม่ซ้ำกัน เพราะต้องใช้ระบุถึง Client

2) SUBSCRIBE payload จะเก็บข้อมูล topic name ที่ไคลเอนต์สามารถ subscribe ได้ และระดับเอง QoS โดยเก็บเป็นสตริงแบบ UTF-8

3) SUBACK payload เก็บระดับของ QoS ที่เซิร์ฟเวอร์อนุญาตให้ไคลเอนต์เข้ามา subscribe ได้

2.1.2.4 ตัวระบุประเภทของข้อความ (Message Identifiers)

Message Identifiers ปรากฏอยู่ใน variable header ของ MQTT message ประเภท PUBLISH, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE และ UNSUBACK ซึ่งปรากฏในกรณีที่มีการระบุระดับของ QoS เป็นระดับ 1 หรือ 2 เท่านั้น ซึ่งจะไม่มี message ID ที่มีค่า 0

2.1.2.5 โพรโทคอล MQTT กับการเข้ารหัสแบบ UTF-8

การเข้ารหัสแบบ UTF-8 ไบต์ที่ 1 และ 2 ใช้สำหรับเก็บจำนวนตัวอักษรของข้อความ ไบต์ที่ 3 เป็นต้นไปใช้ระบุตัวอักษรแบบ ASCII ซึ่งมีรูปแบบดังตาราง 2.10

ตาราง 2.10 รูปแบบฟิลด์สำหรับการกำหนด Message Identifiers

bit	7	6	5	4	3	2	1	0
byte 1	String Length MSB							
byte 2	String Length LSB							
bytes 3 ...	Encoded Character Data							

ตาราง 2.11 ตัวอย่างการกำหนด Message Identifiers คำว่า OTWP

bit	7	6	5	4	3	2	1	0
byte 1	Message Length MSB (0x00)							
	0	0	0	0	0	0	0	0
byte 2	Message Length LSB (0x04)							
	0	0	0	0	0	1	0	0
byte 3	'O' (0x4F)							
	0	1	0	0	1	1	1	1
byte 4	'T' (0x54)							
	0	1	0	1	0	1	0	0
byte 5	'W' (0x57)							
	0	1	0	1	0	1	1	1
byte 6	'P' (0x50)							
	0	1	0	1	0	0	0	0

2.1.2.6 bit ที่ไม่มีการใช้งาน Unused bits

ในกรณีที่บิตใดไม่ถูกนำมาใช้งานจะต้องกำหนดค่าค่าให้เป็น 0

2.1.3 Command messages

2.1.3.1 CONNECT

CONNECT ใช้ในการร้องขอการเชื่อมต่อของไคลเอนต์ไปยัง Server

ตาราง 2.12 Fixed header ของ CONNECT message

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (1)				DUP flag	QoS level		RETAIN
	0	0	0	1	x	x	x	x
byte 2	Remaining Length							

ข้อสังเกต ฟิลด์ DUP, QoS, และ RETAIN flags ไม่ถูกนำมาใช้งานVariable header ของโพรโทคอล MQTT V 3.1 มีขนาด 12 byte

ตาราง 2.13 ตัวอย่างการกำหนด Variable header

	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Protocol Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (6)	0	0	0	0	0	1	1	0
byte 3	'M'	0	1	0	0	1	1	0	1
byte 4	'Q'	0	1	0	1	0	0	0	1
byte 5	'I'	0	1	0	0	1	0	0	1
byte 6	's'	0	1	1	1	0	0	1	1
byte 7	'd'	0	1	1	0	0	1	0	0
byte 8	'p'	0	1	1	1	0	0	0	0
Protocol Version Number									
byte 9	Version (3)	0	0	0	0	0	0	1	1
Connect Flags									
	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 10	User name flag (1) Password flag (1) Will RETAIN (0) Will QoS (01) Will flag (1) Clean Session (1)	1	1	0	0	1	1	1	x
Keep Alive timer									
byte 11	Keep Alive MSB (0)	0	0	0	0	0	0	0	0
byte 12	Keep Alive LSB (10)	0	0	0	0	1	0	1	0

สำหรับ payload ของ CONNECT Message จะประกอบไปด้วยสตริงที่เข้ารหัสแบบ UTF-8 ที่มีขนาด 1 ตัวอักษรหรือมากกว่านั้น และส่วนที่เหลือจะขึ้นอยู่กับการกำหนด Flag ต่างๆ ใน Variable Header ซึ่งถ้าหากมีการกำหนดค่าถูก Flag จะพบว่า payload ของ CONNECT Message จะประกอบไปด้วยข้อมูลดังนี้

1) Client Identifier เก็บข้อมูลสตริงแบบ UTF-8 โดยเป็นไคลเอนต์ID ซึ่งมีขนาด 1-23 ตัวอักษร ซึ่งถ้าหากกำหนดเกิน 23 ตัวอักษร จะทำให้เกิดข้อผิดพลาด Identifier Rejected

2) Will Topic ถ้า Flag นี้ถูกกำหนดค่าWill Message จะถูก Publish ไปยัง Will Topic โดยระดับของ QoS จะถูกกำหนดโดย Will QoS และ Retain Status จะถูกกำหนดโดย Will Retain flag ที่อยู่ใน Variable Header

3) Will message ถ้า Flag นี้ถูกกำหนดค่าWill Message จำกำหนดข้อความที่ถูก Publish ไปยัง Will Topic และถ้าไคลเอนต์ขาดการติดต่อไปแบบไม่คาดคิด Will Message อาจจะมีความยาวเป็น 0 ตัวอักษร

4) User Name ใช้ในการระบุตัวตนและใช้พิสูจน์สิทธิ์ในการเชื่อมต่อ โดย User Name Flag ต้องถูกกำหนดค่าซึ่ง User name จะต้องเป็นสตริงที่เข้ารหัสแบบ UTF-8 และมีความยาวไม่เกิน 12 ตัวอักษร

5) Password ใช้ในการพิสูจน์สิทธิ์ในการเชื่อมต่อ โดย Password Flag ต้องถูกกำหนดค่าซึ่ง Password จะต้องเป็นสตริงที่เข้ารหัสแบบ UTF-8 และมีความยาวไม่เกิน 12 ตัวอักษร

เซิร์ฟเวอร์จะส่ง CONACK message เพื่อตอบ CONNNECT message จากไคลเอนต์ถ้าหากเซิร์ฟเวอร์ไม่ได้รับ CONNECT message ภายในเวลาที่กำหนดเซิร์ฟเวอร์จะทำการปิดการเชื่อมต่อ ในขณะเดียวกันถ้าไคลเอนต์ไม่ได้รับ CONACK message จากเซิร์ฟเวอร์ภายในเวลาที่กำหนดไคลเอนต์ก็จะทำการปิด socket ของการเชื่อมต่อของโพรโทคอล TCP/IP และทำการเปิด socket ใหม่ เพื่อส่ง CONNECT message ไปยังเซิร์ฟเวอร์

2.1.3.2 CONACK

CONACK เซิร์ฟเวอร์จะส่ง CONACK message ไปยังไคลเอนต์เพื่อตอบรับการเชื่อมต่อด้วยCONNECT message

ตาราง 2.14 Fixed header ของ CONACK message

bit	7	6	5	4	3	2	1	0
byte 1	Message type (2)				DUP flag	QoS flags		RETAIN
	0	0	1	0	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

ตาราง 2.15 Variable header ของ CONACK message

	Description	7	6	5	4	3	2	1	0
Topic Name Compression Response									
byte 1	Reserved values. Not used.	x	x	x	x	x	x	x	x
Connect Return Code									
byte 2	Return Code								

ตาราง 2.16 Return code ของ CONACK message

Enumeration	HEX	Meaning
0	0x00	Connection Accepted
1	0x01	Connection Refused: unacceptable protocol version
2	0x02	Connection Refused: identifier rejected
3	0x03	Connection Refused:เซิร์ฟเวอร์unavailable
4	0x04	Connection Refused: bad user name or password
5	0x05	Connection Refused: not authorized
6-255		Reserved for future use

2.1.3.3 PUBLISH

PUBLISH ใช้ในการ publish ข้อความ โดยไคลเอนต์จะส่ง PUBLISH Message ไปยังเซิร์ฟเวอร์ โดย PUBLISH Message แต่ละอันจะต้องถูกส่งไปยัง Topic ที่เกี่ยวข้อง และถ้าหากไคลเอนต์ทำการ Subscribe Topic หนึ่ง Topic หรือมากกว่า ข้อความที่ถูก publish ไปยัง Topic เหล่านั้นจะถูกเซิร์ฟเวอร์ส่งมายังไคลเอนต์ แบบ PUBLISH Message

ตาราง 2.17 Fixed header ของ PUBLISH

bit	7	6	5	4	3	2	1	0
byte 1	Message type (3)				DUP flag	QoS level		RETAIN
	0	0	1	1	0	0	1	0
byte 2	Remaining Length							

Remaining Length จะเป็นขนาดของความยาวของ Variable Header + ความยาวของ payload โดยมีขนาดเป็นจำนวนเท่าของไบต์

Variable header ประกอบด้วยฟิลด์ดังต่อไปนี้

1) Topic name เป็นสตริงที่เข้ารหัสแบบ UTF-8 และจะต้องไม่มี Wildcard Character รวมอยู่ด้วย

2) Message ID จะปรากฏเมื่อใช้ QoS ระดับ 1 หรือ 2 เท่านั้น

ตาราง 2.18 ตัวอย่างของ Variable Header ของ PUBLISH Message

Field	Value
Topic Name:	"a/b"
QoS level	1
Message ID:	10

ตาราง 2.19 ตัวอย่างการเก็บข้อมูลของ PUBLISH message

	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Topic Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 3	'a' (0x61)	0	1	1	0	0	0	0	1
byte 4	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 5	'b' (0x62)	0	1	1	0	0	0	1	0
Message Identifier									
byte 6	Message ID MSB (0)	0	0	0	0	0	0	0	0
byte 7	Message ID LSB (10)	0	0	0	0	1	0	1	0

Payload เป็นข้อมูลที่ต้องการ PUBLISH ซึ่งขึ้นอยู่กับการนำข้อมูลที่ต้องการนำไปใช้งานResponse ใช้ตอบรับ PUBLISH Message ซึ่งขึ้นอยู่กับระดับ QoS ที่ใช้ ซึ่งเป็นไปตามตารางต่อไปนี้นี้

ตาราง 2.20 ระดับ QoS ของ PUBLISH message

QoS Level	Expected response
QoS 0	None
QoS 1	PUBACK
QoS 2	PUBREC

PUBLISH message อาจจะถูกส่งมาจาก ผู้ส่ง (Publisher) ไปยังเซิร์ฟเวอร์หรือ จากเซิร์ฟเวอร์ไปยังผู้ติดตาม (Subscriber) ซึ่งเมื่อผู้รับได้รับข้อความแล้วก็จะมีการกระทำที่แตกต่างกันออกไปตามระดับของ QoS

2.1.3.4 PUBACK

PUBACK เป็นข้อความที่ใช้ตอบ PUBLISH message ที่มี QoS ระดับ 1 โดย PUBACK Message จะถูกส่งโดยเซิร์ฟเวอร์เพื่อตอบรับ PUBLISH message ที่มาจากการ Publish ของไคลเอนต์และโดย Subscriber เพื่อตอบรับ PUBLISH message ที่มาจาก Server

ตาราง 2.21 Fixed Header ของ PUBACK

bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	Message Type (4)				DUP flag	QoS level		RETAIN
	0	1	0	0	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

ข้อสังเกต ฟิลด์ DUP, QoS, และ RETAIN flags ไม่ถูกนำมาใช้งาน และฟิลด์ Remaining Length มีขนาด 2 ไบต์ และสามารถมีขนาดเป็นจำนวนเท่าของไบต์ได้ Variable header ประกอบไปด้วย Message ID ขนาด 16 บิต สำหรับใช้ใน PUBLISH message

ตาราง 2.22 Variable header ของ PUBACK message

bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	Message ID MSB							
byte 2	Message ID LSB							

เมื่อไคลเอนต์ได้รับ PUBACK Message แล้วจะทำการทิ้งข้อความเก่าไป เนื่องจากข้อความนั้นถูกเก็บไว้ในเซิร์ฟเวอร์เรียบร้อยแล้ว

2.1.3.5 PUBREC

PUBREC ใช้สำหรับรับประกันการส่งข้อความ (ส่วนที่ 1) เป็นข้อความที่ใช้ในการตอบรับ PUBLISH Message ที่มีระดับ QoS เท่ากับ 2 โดย PUBACK Message ถูกส่งโดยเซิร์ฟเวอร์เพื่อตอบรับ PUBLISH Message จากไคลเอนต์หรือ Subscriber ตอบรับ PUBLISH Message จาก Server

ตาราง 2.23 Fixed Header ของ PUBREC message

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (5)				DUP flag	QoS level		RETAIN
	0	1	0	1	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

ข้อสังเกต ฟิลด์ DUP, QoS, และ RETAIN flags ไม่ถูกนำมาใช้งาน และความยาวของ Variable Header มีขนาด 2 ไบต์ หรือ เป็นจำนวนเท่าของไบต์ Variable header ประกอบไปด้วย Message ID สำหรับการ Acknowledge PUBLISH Message

ตาราง 2.24 Variable header ของ PUBREC message

bit	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

เมื่อได้รับ PUBREC แล้ว ผู้รับจะส่ง PUBREL Message ไปยังผู้ส่ง โดยระบุ Message ID เดิมลงใน PUBREC Message

2.1.3.6 PUBREL

PUBREL ใช้สำหรับรับประกันการส่งข้อความ (ส่วนที่ 2) เป็นข้อความที่ได้รับการตอบรับจาก Publisher ไปยัง PUBREC Message จากเซิร์ฟเวอร์หรือ จากเซิร์ฟเวอร์ไปยัง PUBREC Message จาก Subscriber

ตาราง 2.25 Fixed header ของ PUBREL message

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (6)				DUP flag	QoS level		RETAIN
	0	1	1	0	0	0	1	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

ข้อสังเกต RETAIN ไม่ถูกนำมาใช้งาน และ Variable Header มีขนาด 2 ไบต์ PUBREL Message ใช้ QoS ระดับ 1 และกำหนด DUP Flag เป็น เมื่อมีการส่งข้อความเป็นครั้งแรก

ตาราง 2.26 Variable header ของ PUBREL message

bit	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

เมื่อเซิร์ฟเวอร์ได้รับ PUBREL Message จาก Publisherเซิร์ฟเวอร์จะเก็บข้อความเดิมไว้สำหรับ Subscriber และทำการส่ง PUBCOMP โดยใช้ Message ID เดิมไปยัง Publisher เมื่อ Subscriber ได้รับ PUBREL Message จากเซิร์ฟเวอร์แล้ว Subscriber จะเก็บข้อความไว้ และส่ง PUBCOMP ไปยัง Server

2.1.3.7 PUBCOMP

PUBCOMP ใช้สำหรับรับประกันการส่งข้อความ (ส่วนที่ 3) เป็นการตอบรับจากเซิร์ฟเวอร์ไปยัง Publisher ที่ส่ง PUBREL Message หรือ เป็นการตอบรับจาก Subscriber ไปยัง PUBREL Message จาก Server

ตาราง 2.27 Fixed header ของ PUBCOMP message

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (7)				DUP flag	QoS level		RETAIN
	0	1	1	1	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

ข้อสังเกต ฟิลด์ DUP, QoS, และ RETAIN flags ไม่ถูกนำมาใช้งาน Variable Header มีขนาด 2 ไบต์ Variable header ใช้เก็บ Message ID ขนาด 16 บิต ที่ได้จากการ Acknowledge PUBREL Message

ตาราง 2.28 Variable header ของ PUBCOMP message

bit	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

เมื่อไคลเอนต์ได้รับ PUBCOMP Message ไคลเอนต์จะลบข้อความดั้งเดิมเนื่องจากมันถูกส่งเซิร์ฟเวอร์ไปแล้ว

2.1.3.8 SUBSCRIBE

SUBSCRIBE ใช้สำหรับลงทะเบียนเพื่อติดตาม topic ที่ไคลเอนต์สนใจกับเซิร์ฟเวอร์ โดยเมื่อลงทะเบียนและทำการกำหนดค่าระดับ QoS ที่ต้องการแล้วเซิร์ฟเวอร์จะส่ง PUBLISH message มาให้ไคลเอนต์ซึ่งไคลเอนต์ที่เป็น subscriber ก็จะได้รับ PUBLISH Message ของ topic ที่ต้องการ

ตาราง 2.29 Fixed header ของ SUBSCRIBE message

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (8)				DUP flag	QoS level		RETAIN
	1	0	0	0	0	0	1	x
byte 2	Remaining Length							

ข้อสังเกต ฟิลด์ RETAIN ไม่ถูกนำมาใช้งานVariable header ประกอบไปด้วย Message ID สำหรับใช้ใน SUBSCRIBE Message ที่มีระดับ QoS เป็น 1 จากตารางด้านล่างจะเป็นตัวอย่างของการเก็บ Message ID 10 ลงใน Variable header

ตาราง 2.30 Variable header ของ SUBSCRIBE message

	Description	7	6	5	4	3	2	1	0
Message Identifier									
byte 1	Message ID MSB (0)	0	0	0	0	0	0	0	0
byte 2	Message ID LSB (10)	0	0	0	0	1	0	1	0

Payload จะเป็น list ของ Topic ที่ไคลเอนต์ต้องการ Subscribe และระดับของ QoS ที่ไคลเอนต์ใช้ในการรับข้อมูล โดยที่ Topic จะต้องกำหนดเป็น String ที่เข้ารหัสแบบ UTF-8 และสามารถมี Wildcard ได้

ตาราง 2.31 ตัวอย่าง payload ของ SUBSCRIBE message

Topic name	"a/b"
Requested QoS	1
Topic name	"c/d"
Requested QoS	2

ตาราง 2.32 ตัวอย่างการเก็บ payload ของ SUBSCRIBE message ของ topic “a/b” และ “c/d”

	Description	7	6	5	4	3	2	1	0
Topic name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	'a' (0x61)	0	1	1	0	0	0	0	1
byte 4	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 5	'b' (0x62)	0	1	1	0	0	0	1	0
Requested QoS									
byte 6	Requested QoS (1)	x	x	x	x	x	x	0	1
Topic Name									
byte 7	Length MSB (0)	0	0	0	0	0	0	0	0
byte 8	Length LSB (3)	0	0	0	0	0	0	1	1
byte 9	'c' (0x63)	0	1	1	0	0	0	1	1

ตาราง 2.32 ตัวอย่างการเก็บ payload ของ SUBSCRIBE message ของ topic “a/b”และ “c/d”(ต่อ)

	Description	7	6	5	4	3	2	1	0
byte 10	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 11	'd' (0x64)	0	1	1	0	0	1	0	0
Requested QoS									
byte 12	Requested QoS (2)	x	x	x	x	x	x	1	0

การกำหนดระดับของ QoS จะสามารถกำหนดได้ที่บิตที่ 0 และ 1 ของไบต์ Requested QoS

ตาราง 2.33 การกำหนดระดับของ QoS ของ SUBSCRIBE message

bit	7	6	5	4	3	2	1	0
	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	QoS level	
	x	x	x	x	x	x		

ข้อสังเกต บิตที่ 7-2 ไม่ถูกนำมาใช้งานเมื่อเซิร์ฟเวอร์ได้รับ SUBSCRIBE Message จากไคลเอนต์Server จะทำการส่ง SUBACK กลับไปที่ Client

2.1.3.9 SUBACK

SUBACK ถูกส่งโดยเซิร์ฟเวอร์เพื่อยืนยันว่าเซิร์ฟเวอร์ได้รับ SUBSCRIBE Message จากไคลเอนต์แล้ว

ตาราง 2.34 Fixed header ของ SUBACK message

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (9)				DUP flag	QoS level		RETAIN
	1	0	0	1	x	x	x	x
byte 2	Remaining Length							

ข้อสังเกต ฟิลด์ DUP, QoS, และ RETAIN flags ไม่ถูกนำมาใช้งาน และ Remain Length จะเป็นนความยาวของ PayloadVariable header ประกอบไปด้วย Message ID สำหรับ SUBSCRIBE Message ที่ได้รับมา ซึ่งมีรูปแบบดังนี้

ตาราง 2.35 Variable header ของ SUBACK message

	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

Payload จะเก็บระดับของ QoS ที่เกี่ยวข้องกับ Topic ตามที่ได้รับมาจาก SUBSCRIBE Message และการกำหนดระดับของ QoS จะสามารถกำหนดได้ที่บิตที่ 0 และ ของไบต์ Requested QoS

ตาราง 2.36 ตัวอย่าง payload ของ SUBACK message

bit	7	6	5	4	3	2	1	0
	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	QoS level	
	x	x	x	x	x	x		

ข้อสังเกต บิตที่ 7-2 ไม่ถูกนำมาใช้งาน

ตาราง 2.37 รูปแบบ payload ของ SUBACK message

	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	Granted QoS (0)	x	x	x	x	x	x	0	0
byte 1	Granted QoS (2)	x	x	x	x	x	x	1	0

2.1.3.10 UNSUBSCRIBE

UNSUBSCRIBE เป็นข้อความที่ถูกส่งจากไคลเอนต์ไปยังเซิร์ฟเวอร์เพื่อยกเลิกการติดตาม Topic ที่ติดตามอยู่

ตาราง 2.38 Fixed header ของ UNSUBSCRIBE message

bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	Message Type (10)				DUP flag	QoS level		RETAIN
	1	0	1	0	0	0	1	x
byte 2	Remaining Length							

ข้อสังเกต QoS Level จะต้องกำหนดให้เป็นระดับ 1 RETAIN ไม่ถูกนำมาใช้งาน และ Remaining Length จะเป็นความยาวของ PayloadVariable header ประกอบด้วย Message ID ที่ถูกส่งมากับ UNSUBSCRIBE Message

ตาราง 2.39 Variable header ของ UNSUBSCRIBE Message

	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Message Identifier									
byte 1	Message ID MSB (0)	0	0	0	0	0	0	0	0
byte 2	Message ID LSB (10)	0	0	0	0	1	0	1	0

ใช้เก็บ List ของ Topic Name ที่ต้องการจะยกเลิกการติดตาม(Unsubscribe)ตัวอย่างของ payload ของ Unsubscribe Message จะเป็นดังนี้

ตาราง 2.40 ตัวอย่างการเก็บ payload ของ Unsubscribe message ของ topic "a/b" และ "c/d"

	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Topic Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	'a' (0x61)	0	1	1	0	0	0	0	1
byte 4	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 5	'b' (0x62)	0	1	1	0	0	0	1	0
Topic Name									
byte 6	Length MSB (0)	0	0	0	0	0	0	0	0
byte 7	Length LSB (3)	0	0	0	0	0	0	1	1
byte 8	'c' (0x63)	0	1	1	0	0	0	1	1
byte 9	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 10	'd' (0x64)	0	1	1	0	0	1	0	0

เมื่อเซิร์ฟเวอร์ได้รับ UNSUBSCRIBE Message แล้วเซิร์ฟเวอร์จะส่ง UNSUBACK Message กลับมาที่ Client

2.1.3.11 UNSUBACK

UNSUBACK ถูกส่งโดยเซิร์ฟเวอร์ไปยังไคลเอนต์เพื่อยืนยันว่าได้รับ UNSUBSCRIBE Message เรียบร้อยแล้ว

ตาราง 2.41 Fixed header ของ UNSUBACK message

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (11)				DUP flag	QoS level		RETAIN
	1	0	1	1	x	x	x	x
byte 2	Remaining length (2)							
	0	0	0	0	0	0	1	0

ข้อสังเกต ฟิลด์ DUP, QoS, และ RETAIN flags ไม่ถูกนำมาใช้งาน และ Remaining Length จะเป็นขนาดของ Variable Header ซึ่งมีขนาด 2 ไบต์

ตาราง 2.42 การเก็บ Message ID ที่ถูกส่งมากับ UNSUBSCRIBE Message

bit	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

2.1.3.12 PINGREQ

PINGREQ ใช้ในการตรวจสอบการเชื่อมต่อระหว่างไคลเอนต์กับเซิร์ฟเวอร์โดยไคลเอนต์ที่เชื่อมต่อกับเซิร์ฟเวอร์จะเป็นฝ่ายส่ง

ตาราง 2.43 Fixed header ของ PINGREQ message

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (12)				DUP flag	QoS level		RETAIN
	1	1	0	0	x	x	x	x
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

ข้อสังเกต ฟิลด์ DUP, QoS, และ RETAIN flags ไม่ถูกนำมาใช้งาน

2.1.3.13 PINGRESP

PINGRESP เป็นข้อความที่เซิร์ฟเวอร์ใช้ตอบไคลเอนต์ที่ส่ง PINGREQ Message มาสอบถามสถานการณ์เชื่อมต่อ

ตาราง 2.44 การเก็บ Message ID ที่ถูกส่งมากับ PINGRESP Message

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (13)				DUP flag	QoS level		RETAIN
	1	1	0	1	x	x	x	x
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

ข้อสังเกต ฟิลด์ DUP, QoS, และ RETAIN flags ไม่ถูกนำมาใช้งาน

2.1.3.14 DISCONNECT

DISCONNECT ถูกส่งจากไคลเอนต์ไปยังเซิร์ฟเวอร์ในกรณีที่ไคลเอนต์ต้องการยกเลิกการเชื่อมต่อโดยการปิด TCP/IP Connection

ตาราง 2.45 การเก็บ Message ID ที่ถูกส่งมากับ DISCONNECT Message

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (14)				DUP flag	QoS level		RETAIN
	1	1	1	0	x	x	x	x
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

2.1.4 ระดับของคุณภาพในการให้บริการ (QoS Level) และ Flow การทำงาน

2.1.4.1 QoS ระดับ 0

QoS ระดับ 0 ไม่รับประกันว่าข้อมูลส่งถึงปลายทางหรือไม่ และมีการส่งเพียงครั้งเดียว (At most once delivery)

ตาราง 2.46 ขั้นตอนการทำงานของ QoS ระดับ 0

Client	message and direction	Server
QoS = 0	PUBLISH ---------->	Action: ทำการ Publish message ไปยัง subscriber

2.1.4.2 QoS ระดับ 1

QoS ระดับ 1 มีการรับประกันว่าข้อมูลส่งถึงปลายทางแน่นอน แต่อาจมีการส่งหลายครั้ง และปลายทางอาจได้รับข้อมูลซ้ำ (At least once delivery)

ตาราง 2.47 ขั้นตอนการทำงานของ QoS ระดับ 1

Client	message and direction	Server
QoS = 1 DUP = 0 Message ID = x Action: เก็บ message ลงในหน่วยความจำ	PUBLISH ---------->	Actions: • เก็บ message ลงในหน่วยความจำ • ทำการ Publish message ไปยัง subscriber • ลบ message ออกจากหน่วยความจำ
Action: ลบ message ออกจากหน่วยความจำ	PUBACK <----------	

2.1.4.3 QoS ระดับ 2

QoS ระดับ 2 มีการรับประกันว่าข้อมูลส่งถึงปลายทางแน่นอน มีการส่งข้อมูลเพียงครั้งเดียว และปลายทางไม่ได้รับข้อมูลซ้ำ (Exactly once delivery)

ตาราง 2.48 ขั้นตอนการทำงานของ QoS ระดับ 2

Client	message and direction	Server
QoS = 2 DUP = 0 Message ID = x Action: เก็บ message ลงในหน่วยความจำ	PUBLISH ---------->	Action: เก็บ message ลงในหน่วยความจำ หรือ Actions: เก็บ message ID ลงในหน่วยความจำ • Publish message to subscribers •
	PUBREC <----------	Message ID = x
Message ID = x	PUBREL<br---------->	Actions: • ทำการ Publish message ไปยัง subscriber • ลบ message ออกจากหน่วยความจำ หรือ Action: ลบ message ID ออกจากหน่วยความจำ
Action: ลบ message ออกจากหน่วยความจำ	PUBCOMP <----------	Message ID = x

Topic Wildcards แบ่งออกเป็น 3 ประเภทดังนี้

1) Topic level separator ใช้แบ่งระดับของ Topic โดยใช้เครื่องหมาย "/" โดยรูปแบบจะเป็นดังนี้ /v0/lv1/lv2/…/lvn
2) Multi-level wildcard ใช้อ้างถึง Subtree ของ Tree ของ Topic เช่น ถ้าหาก topic a/b มี topic ย่อยๆ ดังนี้ a/b/c, a/b/d, a/b/e, a/b/e/x และ a/b/e/z เราสามารถใช้ a/b/# ในการอ้างถึง topic ย่อยๆเหล่านั้นได้
3) Single-level wildcard ใช้อ้างถึง topic ทั้งหมดใน 1 ระดับ เช่น ถ้าหาก topic a/b มี topic ย่อยๆ ดังนี้ a/b/c, a/b/d, a/b/e, a/b/e/x และ a/b/e/z เราสามารถใช้ a/b/+ ในการอ้างถึง a/b/c และ a/b/d และ a/b/e จะไม่ครอบคลุม topic a/b/e/x และ a/b/e/z

2.2 เทคโนโลยี Internet of Things

Internet of Things (IoT) คือ เทคโนโลยีอินเตอร์เน็ตที่เชื่อมอุปกรณ์และ เครื่องมือต่างๆ เช่น โทรศัพท์มือถือ รถยนต์ ตู้เย็น โทรทัศน์ และอื่นๆ เข้าไว้ด้วยกัน โดยเครื่องมือต่างๆ จะสามารถเชื่อมโยงและสื่อสารกันได้โดยผ่านระบบอินเตอร์เน็ต ในอนาคต ผู้บริโภคทั่วไปจะเริ่มคุ้นเคยกับเทคโนโลยีที่ทำให้พวกเขา สามารถควบคุมสิ่งของต่างๆ ทั้งจากในบ้าน และสำนักงานหรือจากที่ไหนก็ได้ เช่น การควบคุมอุณหภูมิภายในบ้าน การเปิดปิดไฟ ไปจนถึงการสั่งให้เครื่องทำกาแฟ เริ่มต้มกาแฟ แต่อย่างไรก็ตาม ยังมีเทคโนโลยีอื่นๆ ที่จำเป็นจะต้องถูกพัฒนาก่อนที่ IoT จะเป็นความจริงขึ้นมา เช่น ระบบตรวจจับต่างๆ (Sensors) รูปแบบการ เชื่อมต่อระหว่างอุปกรณ์ และระบบที่ฝังตัวอยู่ในคอมพิวเตอร์ แต่ขณะนี้ บริษัทใหญ่ๆ อย่าง Microsoft และ Cisco ก็หันมาให้ความสนใจกับเทคโนโลยีนี้ และในปี 2013 เทคโนโลยี "Internet of Things" จะถูกพูดถึงกันมากขึ้น และจะมีการทำวิจัยและ พัฒนาเพื่อทำให้ สามารถนำมาใช้ได้จริงมากขึ้น

2.3 ซอฟต์แวร์ openHAB

เป็นซอฟต์แวร์ Opensource สำหรับการควบคุม ติดตาม เฝ้าดู และสั่งการระบบ Automation ต่างๆ โดยเฉพาะ Home Automation หรือบ้านอัจฉริยะ

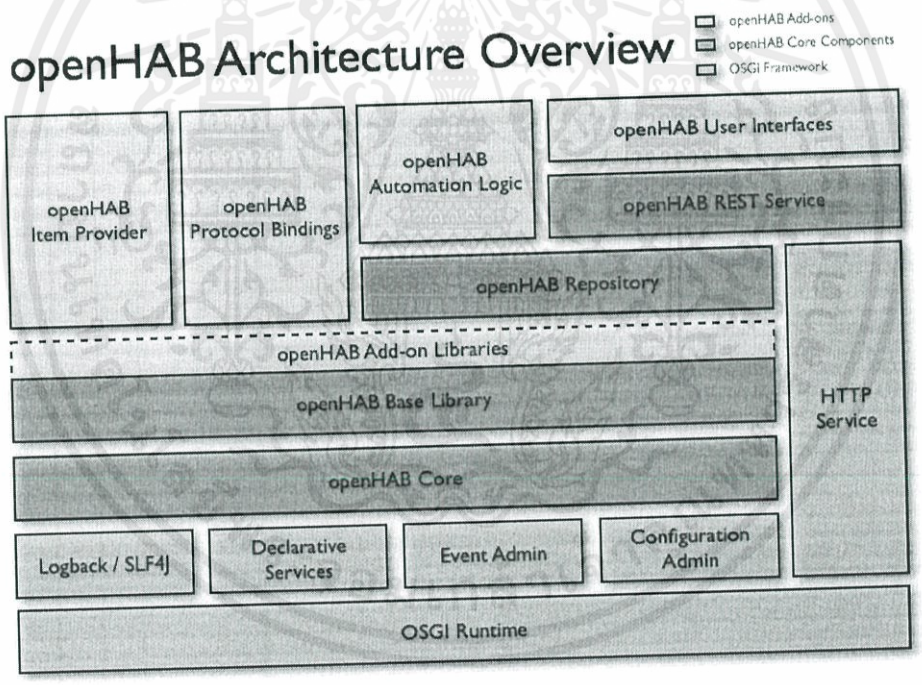
2.3.1 คุณสมบัติ

1) สามารถติดตั้งและใช้งานได้บนระบบปฏิบัติการที่รองรับภาษา Java
2) ใช้งานผ่านทางเว็บแอปพลิเคชั่นหรือแอปพลิเคชั่นบนสมาร์ทโฟนระบบปฏิบัติการ iOS และ Android
3) รองรับโพรโทคอลต่างๆ มากมาย เช่น TCP, UDP, HTTP, MQTT และ SNMP เป็นต้น

2.3.2 สถาปัตยกรรมของ openHAB

2.3.2.1 openHAB Runtime

ถูกพัฒนาด้วยภาษา Java ซึ่งเปรียบเสมือนเป็นเซิร์ฟเวอร์ที่มีหน้าที่จัดการระบบโดยผู้พัฒนาสามารถกำหนดรูปแบบการทำงานรวมทั้งหน้าตาของเว็บแอปพลิเคชั่นหรือการแสดงผลในแอปพลิเคชั่นได้



รูป 2.7 สถาปัตยกรรมของ openHAB

2.3.2.2 การติดต่อสื่อสารภายในของ openHAB

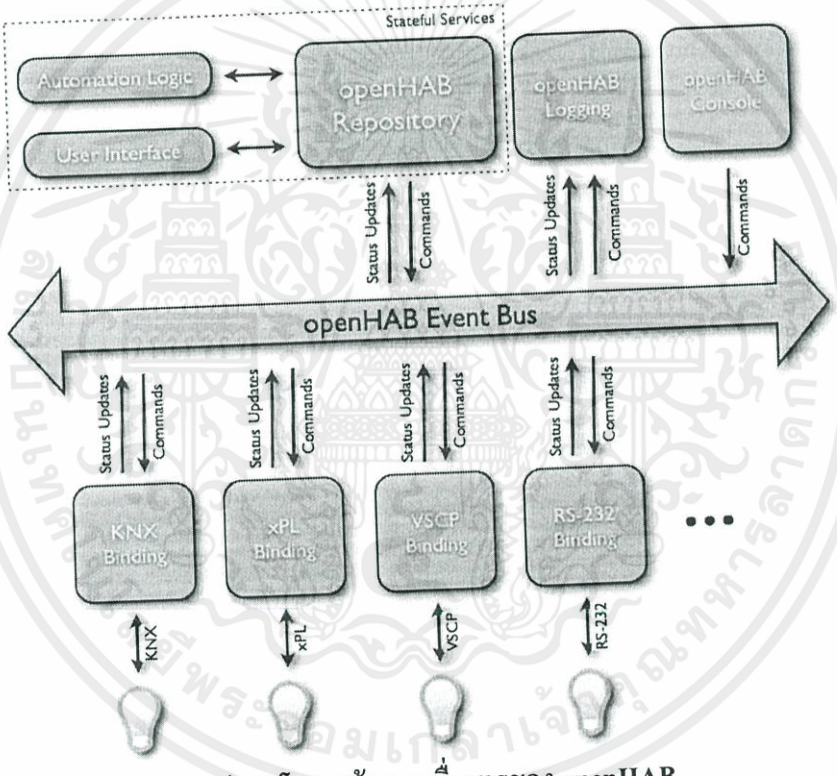
การติดต่อสื่อสารจะแบ่งออกเป็น 2 ช่องทาง ได้แก่ Asynchronous Event bus และ Stateful repository

2.3.2.3 The Event Bus

เป็นเส้นทางในการติดต่อสื่อสารระหว่าง openHAB กับอุปกรณ์ต่างๆ ผ่านช่องทางที่เรียกว่า Protocol binding ซึ่งค่าที่ส่งผ่าน Event bus จะเป็น Event ต่างๆที่เกิดขึ้นของอุปกรณ์ เช่น การเปลี่ยนสถานะของอุปกรณ์ต่างๆ เป็นต้น

2.3.2.4 Item Repository

เป็นการทำงานแบบ Stateful ซึ่ง ใช้ในการติดตามค่าที่ถูกส่งผ่าน Event bus สำหรับตัวอย่างของการใช้ Item repository คือ การติดตามค่าจากอุปกรณ์แล้วนำมาแสดงผลบนแอปพลิเคชั่นอย่างต่อเนื่อง ซึ่งจะมีการตรวจสอบและเก็บสถานการณ์เชื่อมต่อของอุปกรณ์



รูป 2.8 โครงสร้างการสื่อสารของ openHAB

2.3.2.5 Sitemap

เป็น Tree ที่เก็บโครงสร้างของและความสัมพันธ์ของ widget ต่างๆ ของ UI ภายในระบบ

2.3.2.6 Item UI Providers

เป็นตัวช่วยในการกำหนดค่าต่างๆที่เกี่ยวข้องกับการสร้าง UI โดยค่าที่กำหนดนั้นจะถูกเก็บลงใน Sitemap

2.3.2.7 openHAB Designer

เป็นซอฟต์แวร์ที่ใช้ออกแบบการทำงานของ openHAB Runtime รวมถึงการออกแบบและปรับเปลี่ยน GUI ของเว็บแอปพลิเคชั่นและแอปพลิเคชั่นบนสมาร์ทโฟน

2.4 MQTT Broker

เป็นซอฟต์แวร์ที่ทำหน้าที่ส่งต่อ message จาก MQTT Publisher หรือไคลเอนต์ไปยัง MQTT Receiver หรือเซิร์ฟเวอร์ผ่านระบบเครือข่ายโดยใช้โพรโทคอล TCP เป็นพื้นฐานในการส่งข้อมูล

2.4.1 Mosquitto MQTT Broker

เป็นซอฟต์แวร์ประเภท MQTT Broker ที่เป็น Opensource โดยมีส่วนประกอบ ดังนี้

2.4.1.1 mosquitto.conf

เป็นไฟล์ที่ใช้สำหรับกำหนดค่าเพื่อกำหนดการทำงานของ MQTT Broker

2.4.1.2 mosquitto_passwd

เป็นคำสั่งที่ใช้จัดการไฟล์ password ของ mosquitto

2.4.1.3 mosquitto_tls

เป็นคำสั่งที่ใช้ตั้งค่าการติดต่อสื่อสารผ่านโพรโทคอล SSL/TSL

2.4.1.4 mosquitto_pub

เป็นคำสั่งที่ใช้ในการ Publish ค่าไปยัง Topic ที่ต้องการ

2.4.1.5 mosquitto_sub

เป็นคำสั่งที่ใช้ในการ Subscribe ค่าจาก Topic ที่ต้องการ

2.4.1.6 libmosquitto

เป็นไลบรารีสำหรับเขียนโปรแกรมฝั่งไคลเอ็นต์

2.5 Wireless sensor network

Wireless Sensor Networks (WSN) คือ การใช้อุปกรณ์ sensor ขนาดเล็กจำนวนมากมาเชื่อมต่อกันแบบไร้สาย และส่งค่าต่างระหว่าง sensor แต่ละตัว เพื่อให้เกิดการทำงานร่วมกัน โดยมีจุดประสงค์เพื่อตรวจวัดสภาพแวดล้อมต่างๆ เช่น ตรวจวัดอุณหภูมิ ความชื้น แรงสั่นสะเทือน หรือความดันอากาศ เป็นต้น

2.6 Microcontroller

Microcontroller คือ อุปกรณ์อิเล็กทรอนิกส์ชนิดหนึ่งที่ภายในบรรจุ หน่วยประมวลผล, หน่วยความจำ, และช่องทางการติดต่อของข้อมูลหรือที่เรียกกันว่า I/O โดยส่วนใหญ่แล้วจะอยู่ในรูปของชิปขนาดเล็กตัวเดียว Microcontroller ถูกออกแบบให้สามารถเขียนชุดโปรแกรมคำสั่งเพื่อกำหนดการทำงานของ Microcontroller ซึ่งขึ้นอยู่กับบริษัทผู้ออกแบบและสถาปัตยกรรมของ Microcontroller รุ่นนั้นๆ

ปัจจุบันได้มีการนำ Microcontroller ไปประยุกต์ในงานด้านต่างๆมากมายเช่น รถยนต์, เครื่องมือการแพทย์, รีโมทคอนโทรล, ของเล่น และอื่นๆอีกมากมาย เนื่องจาก Microcontroller มีขนาดเล็กและราคาถูกเมื่อเทียบกับ processor แบบอื่น อีกทั้งยังมีให้เลือกหลากหลายรุ่นหลายยี่ห้อเพื่อให้ตรงตามความต้องการของผู้ใช้

2.7 AVR

AVR คือชื่อสถาปัตยกรรมของ Microcontroller ที่พัฒนาโดยบริษัทที่ชื่อว่า Atmel Microcontroller ตระกูล AVR นั้นพัฒนาขึ้นมาด้วยสถาปัตยกรรมแบบ RISC ขนาด 8 bit และ AVR นั้นยังเป็น Microcontroller ตระกูลแรกๆที่สามารถ flash memory (เขียนข้อมูลลงบนชิป) ลงไปบน ROM และ EEPROM ได้โดย Microcontroller ด้วยกันเองซึ่งทำให้ง่ายต่อการใช้งาน

Microcontroller ตระกูล AVR ได้ถูกออกแบบมาให้มี module ภายในหลากหลายเพื่อความหลากหลายในการใช้งานเช่น PWM, EEPROM, USART เป็นต้น

AVR นั้นได้จำแนกรุ่นของ Microcontroller ตามขนาดการใช้งานเป็น 3 ชนิดใหญ่ๆ ได้แก่

1) tinyAVR คือ Microcontroller ขนาดเล็กมากและมีจำนวนขาน้อยเช่นกัน(น้อยที่สุด 6 ขา)ใช้ในอุปกรณ์ขนาดเล็กเช่นนาฬิกา เป็นต้น
2) megaAVR เป็น Microcontroller ขนาดกลางเป็นรุ่นทั่วไปที่บรรจุ module ไว้มากมาย ปัจจุบันนิยมเป็นอย่างมากเนื่องจากสามารถใช้ใน platform ของ Arduino ได้
3) XMEGAVR เป็น Microcontroller 8 bit ขนาดใหญ่ ใช้ในงานที่ต้องการ I/O เป็นจำนวนมาก เนื่องจากมี module multiplex I/O สามารถเลือกขาที่ต้องการติดต่อกับ module ได้อย่างอิสระ

AVR 8-bit ทุกรุ่นจะมีคุณสมบัติเหมือนกันคือ มีหน่วยความจำ 2 ชุดคือ ROM สำหรับเก็บชุดคำสั่งในการทำงานและ EEPROM ในการเก็บข้อมูลแบบถาวร, GPIO ทุกขาสามารถเลือก PULL-UP เพื่อความง่ายในการใช้งาน และทุกรุ่นจะมีความเร็วสูงสุดที่ 20 MIPS เมื่อใช้ oscillator 20 MHz

2.8 EEPROM

EEPROM (Electrically Erasable Programmable Read-Only Memory) คือ หน่วยความจำที่สามารลบได้โดยวิธีการทางไฟฟ้า เป็นหน่วยความจำแบบ non-volatile memory คือสามารถคงสภาพอยู่ได้โดยปราศจากไฟฟ้า ปัจจุบันถูกใช้ในการเก็บข้อมูลขนาดเล็ก ซึ่งสามารถใช้กับอุปกรณ์อิเล็กทรอนิกส์หลายอย่างได้เช่น Microcontroller

EEPROM เป็นหน่วยความจำที่ถูกใช้อย่างแพร่หลายในวงการ Embedded system เนื่องจากราคาถูก และตรงตามความต้องการคือใช้เก็บข้อมูลขนาดเล็ก ในปัจจุบัน EEPROM ถูกทำให้อยู่ในรูปแบบของชิปที่มีขนาดเล็กและติดต่อผ่านทาง bus ต่างๆเช่น I2C SPI ซึ่งสามารถติดต่อผ่านอุปกรณ์ Embedded system ส่วนใหญ่ และ Microcontroller ส่วนใหญ่ในปัจจุบันจะมี module EEPROM บรรจุอยู่ภายใน

2.9 I/O

I/O หรือที่เรียกกันว่า input and output คือช่องทางในการส่งข้อมูลเข้าและออกระหว่างอุปกรณ์อิเล็กทรอนิกส์กับอุปกรณ์หรือระบบอื่นๆ รวบไปถึงการสื่อสารกับมนุษย์ input หมายถึงสัญญาณหรือข้อมูลที่รับเข้ามา output หมายถึงข้อมูลที่ส่งออกจากระบบไป

I/O นิยมจำแนกตามประเภทของระดับการใช้งาน I/O ระดับต่ำสุดจะพูดถึงการใช้ระดับแรงดันไฟฟ้าในการสื่อสารซึ่งเป็นพื้นฐานของระดับต่อๆไป I/O ระดับสูงจะพูดถึงการใช้งานแบบเป็น BUS ข้อมูลเช่น USB เป็นต้น ทั้งนี้ทั้งนั้นการเลือกใช้ I/O ขึ้นอยู่กับความเหมาะสมกับระบบที่พัฒนาขึ้นมาเป็นหลัก

2.10 I2C

I2C ย่อมาจาก Inter-Integrated Circuit เป็นชื่อของ protocol ที่ใช้งานแบบ master - slave โดยสามารถมี master หลายตัว หรือมี slave หลายตัวได้มีจุดเด่นที่ใช้สายเพียง 2 สายต่อ bus ซึ่งก็คือ Serial Data Line (SDA) คือขาข้อมูลและ Serial Clock Line (SCL) คือขาสัญญาณ โดย I2C ถูกพัฒนาขึ้นมาโดยบริษัท Philips Semiconductor เพื่อใช้ในการส่งข้อมูลใน embedded system ปัจจุบันนิยมใช้ในการอ่านค่าจาก sensor

2.11 SPI

SPI ย่อมาจาก Serial Peripheral Interface เป็นชื่อของ protocol แบบ synchronous ซึ่งสามารถใช้งานแบบ master – slave ได้ถูกคิดค้นขึ้นมาโดย Motorola มีสายสัญญาณที่เกี่ยวข้องด้วยกัน

ทั้งหมด 3 เส้นสำหรับส่งข้อมูล คือ 1. SCLK: Serial Clock (สายสัญญาณ) 2. MOSI: Master Output, Slave Input (สายข้อมูลขาออกจาก master) 3. MISO: Master Input, Slave Output (สายข้อมูลขาเข้าหา master) และ SS: Slave Select คือขาใช้เลือก slave ที่ master ต้องการติดต่อ สายสัญญาณ SS จะมีจำนวนตาม slave บน bus ข้อมูลเพื่อเลือก slave ที่ต้องการติดต่อ SPI มีจุดเด่นในด้านความเร็วของการส่งข้อมูลและเป็นการส่งข้อมูลแบบพร้อมกันทั้ง 2 ทางหรือ full duplex แต่มีจุดด้อยเรื่องจำนวนสายสัญญาณจำนวนมาก

2.12 โพรโทคอล SSH

โพรโทคอล SSH (Secure Shell) คือ โพรโทคอล (Protocol) ที่ใช้ในการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์บนระบบเครือข่ายผ่าน พอร์ท (Port) หมายเลข 22 ซึ่งโพรโทคอล SSH มีวัตถุประสงค์หลักเพื่อให้ผู้ใช้งานสามารถเข้าควบคุมหรือสั่งการเครื่อง คอมพิวเตอร์ที่ให้บริการ SSH ตามสิทธิของผู้ใช้งานซึ่งได้มาจากการพิสูจน์ตัวตนด้วยการล็อกอิน (Login) ด้วยการใช้ชื่อผู้ใช้และรหัสผ่าน โดยผ่านช่องทางการสื่อสารที่มีการรักษาความมั่นคงปลอดภัยด้วยการเข้ารหัสลับข้อมูล (Encryption) ซึ่งถูกออกแบบมาเพื่อใช้แทนที่การสื่อสารข้อมูลบนระบบเครือข่ายที่ส่งข้อมูลแบบไม่ได้เข้ารหัสลับ (Plaintext) เช่น Telnet, Rlogin หรือ FTP ปัจจุบันโพรโทคอล SSH มีสองเวอร์ชั่นคือ SSH-1 และ SSH-2

การทำงานของโพรโทคอล SSH จะทำงานในลักษณะไคลเอนต์และเซิร์ฟเวอร์ (Client-Server) โดยรูปแบบการใช้งานจะประกอบไปด้วยโปรแกรม 2 ส่วนคือ โปรแกรมส่วนที่ทำหน้าที่เป็นเครื่องที่ให้บริการ (Server) จะถูกติดตั้งลงที่เครื่องคอมพิวเตอร์ที่ต้องการให้บริการ SSH เช่น โปรแกรม OpenSSH-Server บนระบบปฏิบัติการ Linux โดยส่วนใหญ่แล้วเครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรมที่ให้บริการ SSH จะติดตั้งเพื่ออำนวยความสะดวกแก่ผู้ใช้งานร่วมกับบริการอื่นๆ ควบคู่ไป เช่น บริการเว็บเซิร์ฟเวอร์ หรือบริการอัพโหลดไฟล์ เป็นต้น และโปรแกรมอีกส่วนจะทำหน้าที่เป็นผู้เชื่อมต่อ (Client) ไปยังเครื่องคอมพิวเตอร์ที่ให้บริการ SSH เช่น โปรแกรม PuTTY บนระบบปฏิบัติการ Windows หรือ โปรแกรม OpenSSH-Client บนระบบปฏิบัติการ Linux

ถึงแม้โพรโทคอล SSH จะมีข้อดีในเรื่องของการรักษาความมั่นคงปลอดภัยโดยมีการเข้ารหัสลับข้อมูล และมีการล็อกอินก่อนการเข้าใช้งาน แต่ก็ยังพบว่ามีโอกาสสูงที่จะถูกโจมตีจากผู้ไม่หวังดี เนื่องจากผลลัพธ์และความสำเร็จในการเข้าโจมตีอาจหมายถึงการได้รับสิทธิในการ เข้าควบคุมและสั่งการเครื่องคอมพิวเตอร์ที่ให้บริการนั้นทันที โดยลักษณะการโจมตีที่เกิดขึ้นมักจะมาจากการใช้เทคนิคในการเข้าโจมตีที่ เครื่องคอมพิวเตอร์ที่ให้บริการโดยตรง เช่นการโจมตีด้วยวิธีการสุ่มรหัสผ่าน (Brute-force) เพื่อพยายามเข้าสู่ระบบเครื่องคอมพิวเตอร์ที่ให้บริการ SSH ซึ่งหากผู้ใช้งานหรือผู้ดูแลระบบตั้งค่ารหัสผ่านในการล็อกอินง่ายเกินไปก็จะ ทำให้โอกาสในการโจมตีสำเร็จง่าย

มากขึ้น โดยแนวทางในการป้องกันที่ได้ผลลัพธ์ดีที่สุดคือการรู้ทันการโจมตีและรู้วิธี ในการป้องกันการโจมตีดังกล่าว ซึ่งหนึ่งในวิธีการป้องกันที่ผู้ดูแลระบบส่วนใหญ่นิยมใช้ และจะกล่าวถึงในบทความนี้ คือ การเปลี่ยนวิธีการล็อกอินจากวิธีการปกติที่ใช้รหัสผ่าน เป็นการใช้เทคนิคการใช้คู่กุญแจ (Key Authentication) ซึ่งเป็นรูปแบบการเข้ารหัสแบบอสมมาตร (Asymmetric-key cryptography) โดยมีการสร้างคู่กุญแจ ซึ่งจะประกอบไปด้วยกุญแจสาธารณะ (Public Key) และกุญแจส่วนตัว (Private Key) มีหลักการทำงานคือ ถ้าใช้กุญแจ A ในการเข้ารหัสลับ จะต้องใช้กุญแจ B ในการถอดรหัสลับ โดยการเข้ารหัสและถอดรหัสดังกล่าวจะใช้ฟังก์ชันทางคณิตศาสตร์เข้ามาช่วย ซึ่งการใช้หลักการดังกล่าวในการพิสูจน์ตัวตนของผู้ใช้งานกับเครื่อง คอมพิวเตอร์ที่ให้บริการ SSH จะช่วยป้องกันการโจมตีด้วยวิธีการ Brute-force จากผู้โจมตีได้ และยังสามารถเพิ่มความมั่นคงปลอดภัยจากการใช้งาน Key Authentication ได้โดยการเข้ารหัสลับ Private key ด้วยรหัสผ่านอีกขั้นตอนหนึ่ง เพื่อป้องกันบุคคลอื่นนำกุญแจดังกล่าวไปใช้งาน

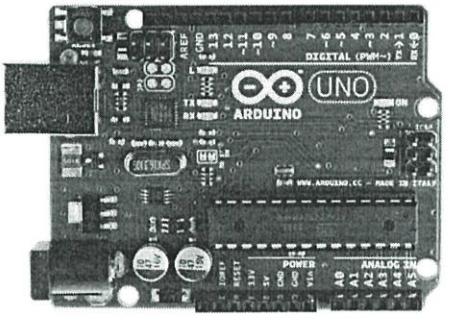
2.13 โพรโทคอล SFTP

SFTP หรือ Secure File Transfer Protocol เป็นโปรแกรมที่ใช้ในการส่งถ่ายข้อมูลไฟล์เอกสาร โดยลักษณะเด่นของ SFTP คือ การทำงานบน SSH ซึ่งมีการเข้ารหัสข้อมูล ทำให้การส่งย้ายไฟล์มีความปลอดภัยมากยิ่งขึ้นเมื่อเทียบกับ FTP ธรรมดา

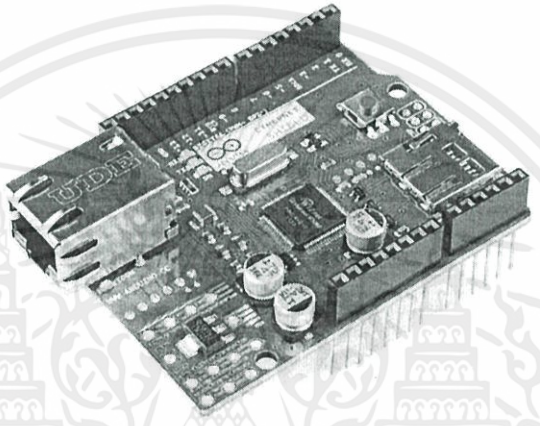
2.14 Arduino

Arduino (อ่านว่า อา-ดู-อิ-โน่ หรือ อาดุยโน่) เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software ตัว บอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ผู้ใช้งานยังสามารถดัดแปลง เพิ่มเติม พัฒนาต่อยอดทั้งตัวบอร์ด หรือโปรแกรมต่อได้อีกด้วย

ความง่ายของบอร์ด Arduino ในการต่ออุปกรณ์เสริมต่างๆ คือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิคส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ด หรือเพื่อความสะดวกสามารถเลือกต่อกับบอร์ดเสริม (Arduino Shield) ประเภท เช่น Arduino XBee Shield, Arduino Music Shield, Arduino Relay Shield, Arduino Wireless Shield, Arduino GPRS Shield เป็นต้น มาเสียบกับบอร์ดบนบอร์ด Arduino แล้วเขียนโปรแกรมพัฒนาต่อได้เลย



รูป 2.9 บอร์ด Arduino



รูป 2.10 บอร์ด Arduino Shield

2.15 Cloud Computing

Cloud computing คือวิธีการประมวลผลที่อิงกับความต้องการของผู้ใช้ โดยผู้ใช้ระบุความต้องการไปยังซอฟต์แวร์ของระบบ Cloud Computing จากนั้นซอฟต์แวร์จะร้องขอให้ระบบจัดสรรทรัพยากรและบริการให้ตรงกับความต้องการผู้ใช้ ทั้งนี้ระบบสามารถเพิ่มและลดจำนวนของทรัพยากร รวมถึงเสนอบริการให้พอเหมาะกับความต้องการของผู้ใช้ได้ตลอดเวลา โดยที่ผู้ใช้ไม่จำเป็นต้องทราบเลยว่าการทำงานหรือเหตุการณ์เบื้องหลังเป็นเช่นไร

2.16 TCP/IP Model

TCP/IP (Transmission Control Protocol/Internet Protocol) เป็นชุดของโปรโตคอลที่ถูกใช้ในการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต โดยมีวัตถุประสงค์เพื่อให้สามารถใช้สื่อสารจากต้นทางข้ามเครือข่ายไปยังปลายทางได้ และสามารถหาเส้นทางที่จะส่งข้อมูลไปได้เองโดยอัตโนมัติ ถึงแม้ว่าในระหว่างทางอาจจะผ่านเครือข่ายที่มีปัญหา โปรโตคอลก็ยังคงหาเส้นทางอื่นในการส่งผ่านข้อมูลไปให้ถึงปลายทางได้

ชุดโปรโตคอลนี้ได้รับการพัฒนามาตั้งแต่ปี 1960 ซึ่งถูกใช้เป็นครั้งแรกในเครือข่าย ARPANET ซึ่งต่อมาได้ขยายการเชื่อมต่อไปทั่วโลกเป็นเครือข่ายอินเตอร์เน็ต ทำให้ TCP/IP เป็นที่ยอมรับอย่างกว้างขวางจนถึงปัจจุบัน

TCP/IP มีจุดประสงค์ของการสื่อสารตามมาตรฐาน สามประการคือ 1. เพื่อใช้ติดต่อสื่อสารระหว่างระบบที่มีความแตกต่างกัน 2. ความสามารถในการแก้ไขปัญหาที่เกิดขึ้นในระบบเครือข่าย เช่นในกรณีที่ผู้ส่งและผู้รับยังคงมีการติดต่อกันอยู่ แต่โหนดกลางทีใช้เป็นผู้ช่วยรับ-ส่งเกิดเสียหายใช้การไม่ได้ หรือสายสื่อสารบางช่วงถูกตัดขาด กฎการสื่อสารนี้จะต้องสามารถจัดหาทางเลือกอื่นเพื่อทำให้การสื่อสารดำเนินต่อไปได้โดยอัตโนมัติ และ 3. มีความคล่องตัวต่อการสื่อสารข้อมูลได้หลายชนิดทั้งแบบที่ไม่มีความเร่งด่วน เช่น การจัดส่งแฟ้มข้อมูล และแบบที่ต้องการรับประกันความเร่งด่วนของข้อมูล เช่น การสื่อสารแบบ real-time และทั้งการสื่อสารแบบเสียง (Voice) และข้อมูล (data)

2.16.1 Encapsulation/Demultiplexing

การส่งข้อมูลผ่านในแต่ละเลเยอร์ แต่ละเลเยอร์จะทำการประกอบข้อมูลที่ได้รับมา กับข้อมูลส่วนควบคุมซึ่งถูกนำมาไว้ในส่วนหัวของข้อมูลเรียกว่า Header ภายใน Header จะบรรจุข้อมูลที่สำคัญของโปรโตคอลที่ทำการ Encapsulate เมื่อผู้รับได้รับข้อมูล ก็จะเกิดกระบวนการทำงานย้อนกลับคือ โปรโตคอลเดียวกัน ทางฝั่งผู้รับก็จะได้รับข้อมูลส่วนที่เป็น Header ก่อนและนำไปประมวลและทราบว่าข้อมูลที่ตามมามีลักษณะอย่างไร ซึ่งกระบวนการย้อนกลับนี้เรียกว่า Demultiplexing

2.16.2 โครงสร้างของ TCP/IP

2.16.2.1 ชั้นโฮสต์-เครือข่าย (Host-to-Network Layer)

ชั้นโฮสต์-เครือข่าย (Host-to-Network Layer)โพรโตคอลสำหรับการควบคุมการสื่อสารในชั้นนี้เป็นสิ่งที่ไม่มีการกำหนดรายละเอียดอย่างเป็นทางการ หน้าที่หลักคือการรับข้อมูลจากชั้นสื่อสาร IP มาแล้วส่งไปยังโหนดที่ระบุไว้ในเส้นทางเดินข้อมูลทางด้านผู้รับก็จะทำงานในทางกลับกัน คือรับข้อมูลจากสายสื่อสารแล้วนำส่งให้กับโปรแกรมในชั้นสื่อสาร

2.16.2.2 ชั้นสื่อสารอินเทอร์เน็ต (The Internet Layer)

ชั้นสื่อสารอินเทอร์เน็ต (The Internet Layer) ใช้ประเภทของระบบการสื่อสารที่เรียกว่า ระบบเครือข่ายแบบสลับช่องสื่อสารระดับแพ็กเก็ต (packet-switching network) ซึ่งเป็นการติดต่อแบบไม่ต่อเนื่อง (Connectionless) หลักการทำงานคือการปล่อยให้ข้อมูลขนาดเล็กที่เรียกว่า แพ็กเก็ต (Packet) สามารถไหลจากโหนดผู้ส่งไปตามโหนดต่างๆ ในระบบจนถึงจุดหมายปลายทางได้โดยอิสระ หากว่ามีการส่งแพ็กเก็ตออกมาเป็นชุดโดยมีจุดหมายปลายทางเดียวกันในระหว่างการเดินทางในเครือข่าย แพ็กเก็ตแต่ละตัวในชุดนี้ก็จะเป็นอิสระแก่กันและกัน ดังนั้น แพ็กเก็ตที่ส่งไปถึงปลายทางอาจจะไม่เป็นไปตามลำดับก็ได้

IP (Internet Protocol) IP เป็นโปรโตคอลในระดับเน็ตเวิร์คเลเยอร์ ทำหน้าที่จัดการเกี่ยวกับแอดเดรสและข้อมูล และควบคุมการส่งข้อมูลบางอย่างที่ใช้ในการหาเส้นทางของแพ็กเก็ต ซึ่งกลไกในการหาเส้นทางของ IP จะมีความสามารถในการหาเส้นทางที่ดีที่สุด และสามารถเปลี่ยนแปลงเส้นทางได้ในระหว่างการส่งข้อมูล และมีระบบการแยกและประกอบดาต้าแกรม (datagram) เพื่อรองรับการส่งข้อมูลระดับ data link ที่มีขนาด MTU (Maximum Transmission Unit) ที่แตกต่างกัน ทำให้สามารถนำ IP ไปใช้บนโปรโตคอลอื่นได้หลากหลาย เช่น Ethernet ,Token Ring หรือ Apple Talk

การเชื่อมต่อของ IP เพื่อทำการส่งข้อมูล จะเป็นแบบ connectionless หรือเกิดเส้นทางการเชื่อมต่อในทุกๆครั้งของการส่งข้อมูล 1 ดาต้าแกรม โดยจะไม่ทราบถึงข้อมูลดาต้าแกรมที่ส่งก่อนหน้าหรือส่งตามมา แต่การส่งข้อมูลใน 1 ดาต้าแกรม อาจจะเกิดการส่งได้หลายครั้งในกรณีที่มีการแบ่งข้อมูลออกเป็นส่วนย่อยๆ (fragmentation) และถูกนำไปรวมเป็นดาต้าแกรมเดิมเมื่อถึงปลายทาง

2.16.2.3 ชั้นสื่อสารนำส่งข้อมูล (Transport Layer)

ชั้นสื่อสารนำส่งข้อมูล (Transport Layer) แบ่งเป็นโพรโตคอล 2 ชนิดตามลักษณะ ลักษณะแรกเรียกว่า Transmission Control Protocol (TCP) เป็นแบบที่มีการกำหนดช่วงการสื่อสารตลอดระยะเวลาการสื่อสาร (connection-oriented) ซึ่งจะยอมให้มีการส่งข้อมูลเป็นแบบ Byte stream ที่ไว้ใจได้โดยไม่มีข้อผิดพลาด ข้อมูลที่มีปริมาณมากจะถูกแบ่งออกเป็นส่วนเล็กๆ เรียกว่า message ซึ่งจะถูกส่งไปยังผู้รับผ่านทางชั้นสื่อสารของอินเทอร์เน็ต ทางฝ่ายผู้รับจะนำ message มาเรียงต่อกันตามลำดับเป็นข้อมูลตัวเดิม TCP ยังมีความสามารถในการควบคุมการไหลของข้อมูลเพื่อป้องกันไม่ให้ผู้ส่ง ส่งข้อมูลเร็วเกินกว่าที่ผู้รับจะทำงานได้ทันอีกด้วย

โปรโตคอลการนำส่งข้อมูลแบบที่สองเรียกว่า UDP (User Datagram Protocol) เป็นการติดต่อแบบไม่ต่อเนื่อง (connectionless) มีการตรวจสอบความถูกต้องของข้อมูลแต่จะไม่มีการแจ้งกลับไปยังผู้ส่ง จึงถือได้ว่าไม่มีการตรวจสอบความถูกต้องของข้อมูล อย่างไรก็ตาม วิธีการนี้มีข้อดีในด้านความรวดเร็วในการส่งข้อมูล จึงนิยมใช้ในระบบผู้ให้และผู้ใช้บริการ (client/server

system) ซึ่งมีการสื่อสารแบบ ถาม/ตอบ (request/reply) นอกจากนั้นยังใช้ในการส่งข้อมูลประเภทภาพเคลื่อนไหวหรือการส่งเสียง (voice) ทางอินเทอร์เน็ต

2.16.2.4 ชั้นสื่อสารการประยุกต์ (Application Layer)

ชั้นสื่อสารการประยุกต์ (Application Layer)มีโพรโตคอลสำหรับสร้างจอเทอร์มินัลเสมือน เรียกว่า TELNET โพรโตคอลสำหรับการจัดการแฟ้มข้อมูล เรียกว่า FTP และโพรโตคอลสำหรับการให้บริการจดหมายอิเล็กทรอนิกส์ เรียกว่า SMTP โดยโพรโตคอลสำหรับสร้างจอเทอร์มินัลเสมือนช่วยให้ผู้ใช้สามารถติดต่อกับเครื่องโฮสต์ที่อยู่ไกลออกไปโดยผ่านอินเทอร์เน็ต และสามารถทำงานได้เสมือนกับว่ากำลังนั่งทำงานอยู่ที่เครื่องโฮสต์นั้น โพรโตคอลสำหรับการจัดการแฟ้มข้อมูลช่วยในการคัดลอกแฟ้มข้อมูลมาจากเครื่องอื่นที่อยู่ในระบบเครือข่ายหรือส่งสำเนาแฟ้มข้อมูลไปยังเครื่องใดๆก็ได้ โพรโตคอลสำหรับให้บริการจดหมายอิเล็กทรอนิกส์ช่วยในการจัดส่งข้อความไปยังผู้ใช้ในระบบ หรือรับข้อความที่มีผู้ส่งเข้ามา

2.17 ไดแอก (Diac)

ไดแอก(Diac) เป็นอุปกรณ์สารกึ่งตัวนำที่อยู่ในกลุ่มของของไทริสเตอร์ มี 2 ขั้วคือ ขั้วแอโนด 1 (A1) และขั้วแอโนด 2 (A2) เพราะไดแอกสามารถนำกระแสได้สองด้าน ไดแอกสามารถนำไปใช้กับแรงดันไฟฟ้าสลับและแรงดันไฟฟ้ากระแสตรงได้

2.17.1 โครงสร้างของไดแอก

เป็นอุปกรณ์สารกึ่งตัวนำมี 3 ตอนใหญ่ชนิดสาร PNP และยังประกอบด้วยสารกึ่งตัวนำ 2 ตอนย่อยชนิด N ต่อร่วมในสารกึ่งตัวนำชนิด P ทั้ง 2 ตอนด้านนอก มีขาต่อออกมาใช้งานเพียง 2 ขา แต่ละขาที่ต่อใช้งานจะต่อร่วมกับสารกึ่งตัวนำทั้งชนิด N และชนิด P จึงทำให้ไดแอกสามารถทำงานได้ทั้งแรงดันไฟบวกและลบ ขาแอโนด1 (A1) เรียกว่า ขาเทอร์มินอล 1 (Main Terminal 1) ใช้ตัวย่อ MT1 และขาแอโนด2 (A2) เรียกว่า ขาเทอร์มินอล 2 (Main Terminal 2) ใช้ตัวย่อ MT2 แต่ละขาสามารถต่อสลับกันได้

2.17.2 การทำงานของไดแอก

ไดแอกมี 2 ขา แต่มีคุณสมบัติสามารถทำงานได้กับแรงดันช่วงบวก และแรงดันช่วงลบ คือกระแสได้ 2 ทิศทาง ดังนั้นในการใช้งานจึงไม่จำเพาะเจาะจงในการต่อวงจร ใช้ขาด้านใดด้านหนึ่งต่อเข้าวงจรก็จะได้คุณสมบัติเหมือนกัน การทำงานของไดแอกเปรียบเหมือนกับชอคเลย์ไดโอด 2 ตัวต่อกลับหัวกัน มีวงจรสมมูลของไดแอกแทนด้วยทรานซิสเตอร์ 4 ตัว ดังรูปที่ 2 ก ทรานซิสเตอร์ Q1 และ Q2 แทนชอคเลย์ไดโอดตัวที่ 1 จะเป็นตัวนำกระแสแอโนดมาจากขั้ว A1 ไปสู่ขั้ว A2 ทรานซิสเตอร์ Q3 และ Q4 แทนแทนชอคเลย์ไดโอดตัวที่ 2 จะเป็นตัวนำกระแสแอโนดมาจากขั้ว วิชา อุปกรณ์อิเล็กทรอนิกส์และวงจร รหัส 2104-2112 A2 ไปสู่ขั้ว A1 ไดแอกเมื่อไบอัส

ตรงให้ขา A1 กับ A2 ไดแอกทำงานเมื่อแรงดันไบอัสตรงที่จ่ายให้ตัวไดแอกถึงค่าแรงดันเบรคโอเวอร์ของตัวไดแอก VBR(F) จะทำให้กระแสไหลผ่านขั้ว A1 ไปยังขั้ว A2 และเมื่อไดแอกทำงานเมื่อไบอัสกลับจ่ายให้ตัวไดแอกถึงค่าแรงดันเบรคโอเวอร์ของตัวไดแอก -VBR(R) จะทำให้กระแสไหลผ่านขั้ว A2 ไปยังขั้ว A1

2.18 ไตรแอก (Triac)

ไตรแอค (Triac) เป็นอุปกรณ์จำพวกสารกึ่งตัวนำในกลุ่มของไทริสเตอร์ มีลักษณะ โครงสร้างภายในคล้ายกับไดแอก แต่มีขาเกตเพิ่มขึ้นมาอีก 1 ขา ไตรแอคถูกสร้างขึ้นเพื่อแก้ไข ข้อบกพร่องของ SCR ซึ่งไม่สามารถนำกระแสในซีกลบของไฟฟ้าสลับได้การนำไตรแอคไปใช้งานส่วนใหญ่จะใช้ทำเป็นวงจรควบคุมการทำงานเป็นสวิตซ์ต่อแรงดันไฟสลับ ไตรแอคถูกสร้าง ขึ้นมาให้ใช้งานกับกระแสสูง ๆ ดังนั้นต้องระวังเรื่องของการระบายความร้อน

โครงสร้างของไตรแอคจะประกอบด้วยสารกึ่งตัวนำตอนใหญ่ 3 ตอน คือ PNP และในสารกึ่งตัวนำตอนใหญ่จะมีสารกึ่งตัวนำตอนย่อยชนิด N อีก 3 ตอน ต่อร่วมในสารกึ่งตัวนำ P ทั้ง 2 ตอน

2.18.1 คุณสมบัติพื้นฐานของไตรแอก

1) โดยปกติ ถ้าไม่มีสัญญาณทริกที่เกต ไตรแอกจะไม่ทำงานโดย จะมีลักษณะเหมือนกับสวิตช์ที่ถูกเปิดวงจร
2) ถ้าในกรณีที่ MT2 และ MT1 ถูกป้อนด้วยแรงดันบวกและลบตามลำดับไตรแอกจะถูกกระตุ้นให้ทำงานได้โดยการป้อนสัญญาณพัลส์เพียงสั้น ๆ ที่เกตของมัน โดยจะมีแรงดันตกคร่อมตัวมัน มีค่าประมาณ 1 หรือ 2 โวลต์ เท่านั้น และก็เช่นกันคือเมื่อไตรแอกเริ่มทำงานแล้ว ก็จะสามารถคงสภาพการทำงานอยู่เช่นนั้นต่อไปเรื่อย ๆ ตราบเท่าที่ยังมีกระแสไหลผ่านตัวมันอย่างต่อเนื่อง
3) หลังจากที่ไตรแอกคงสภาพการทำงานอยู่นั้น ทางเดียวที่จะหยุดการทำงานลงได้ ก็โดยการลดปริมาณกระแสที่ไหลผ่านตัวมันลง ให้มีค่าต่ำกว่ากระแสโฮลดิ้งของมัน ในกรณีที่ใช้ไตรแอกในการจ่ายกระแส AC การหยุดทำงานจะเกิดขึ้นอย่างอัตโนมัติ เมื่อแรงดันของไฟสลับเข้าใกล้จุดตัดศูนย์ที่เกิดขึ้นทุก ๆ ครึ่งคลื่น นั่นคือกระแสจะลดลงเป็นศูนย์
4) ไตรแอกถูกกระตุ้นให้ทำงานได้ ทั้งสัญญาณแบบบวกและลบที่ป้อนให้แก่ขาเกต โดยไม่คำนึงถึงขั้วที่ต่ออยู่ที่ MT1 และ MT2

5) ไตรแอกสามารถทนการกระชากของกระแสได้สูง เช่น โดยปกติสำหรับไตรแอกที่ทนกระแสปกติได้ 10 แอมแปร์ (rms) สามารถทนการกระชากของกระแสในช่วงหนึ่ง คาบเวลาของไฟ 60 เฮิรตซ์ได้สูงถึง 100 แอมแปร์ เป็นต้น

บทที่ 3

การออกแบบและพัฒนาระบบ

Network I/O เป็นบอร์ดไมโครคอนโทรลเลอร์ที่พัฒนามาจากไมโครโพรเซสเซอร์ตระกูล AVR มีความสามารถรับส่งข้อมูลผ่านเครือข่ายได้ซึ่งบอร์ด Network I/O แบ่งออกเป็น 2 ประเภท คือ Network I/O Gateway และ Network I/O Node ซึ่งบอร์ด Network I/O Gateway จะใช้ติดต่อกับ Network I/O Node ผ่านสัญญาณไร้สาย 2.4 GHz และติดต่อกับเซิร์ฟเวอร์ผ่านอินเทอร์เน็ตโดยใช้สาย LAN

3.1 องค์ประกอบของระบบ

1) บอร์ด Network I/O Gateway

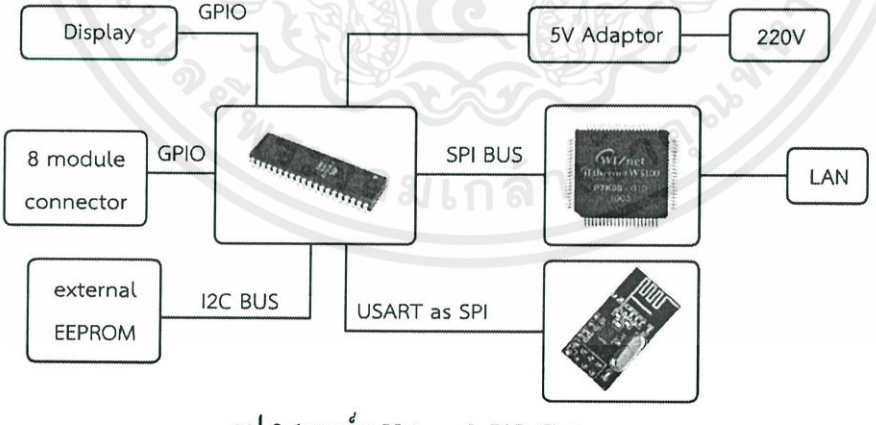
2) บอร์ด Network I/O Node

3) เซิร์ฟเวอร์ที่ติดตั้งซอฟท์แวร์ openHAB และ mosquitto

4) แอปพลิเคชั่นติดตามและควบคุมระบบ

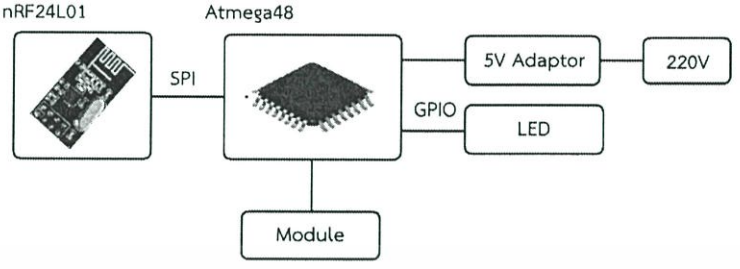
3.2 การออกแบบฮาร์ดแวร์

3.2.1 บอร์ด Network I/O Gateway



รูป 3.1 บอร์ด Network I/O Gateway

3.2.2 บอร์ด Network I/O Node



รูป 3.2 บอร์ด Network I/O Node

3.3 การออกแบบเซิร์ฟเวอร์

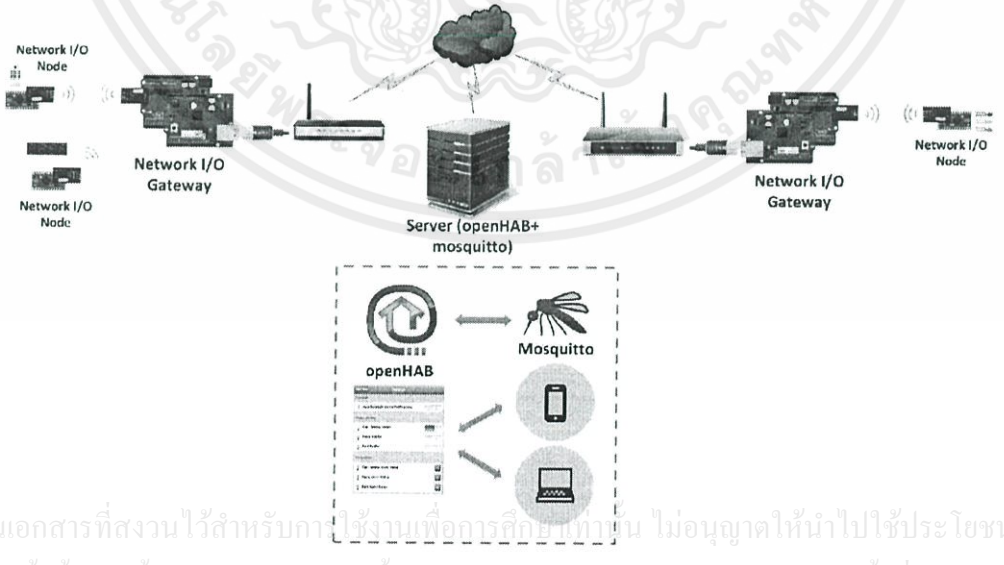
สำหรับเซิร์ฟเวอร์ที่ใช้งานร่วมกับ Network I/O นั้นต้องมีการติดตั้งซอฟท์แวร์ดังนี้

3.3.1 openHAB

openHAB เป็นซอฟท์แวร์สำเร็จรูปแต่สามารถปรับแต่งได้ตามต้องการ โดยเป็นซอฟท์แวร์ที่ใช้ควบคุมการทำงาน สั่งการ ติดตามเฝ้าดูบอร์ด Network I/O ผ่านทางเว็บแอปพลิเคชั่นและแอปพลิเคชั่นบนสมาร์ทโฟนระบบปฏิบัติการ iOS และ Android สำหรับการปรับแต่งนั้นจะทำการปรับแต่งดังนี้

3.3.2 mosquitto

mosquitto เป็นซอฟท์แวร์สำเร็จรูปที่ต้องทำการปรับแต่งค่าให้มีการทำงานสอดคล้องกับ MQTT Server และ openHAB runtime



รูป 3.3 โครงสร้างระบบของ Network I/O

3.4 รายละเอียดการทำงานของระบบ

3.4.1 การทำงานเมื่อระบบเริ่มทำงาน

เมื่อเซิร์ฟเวอร์เริ่มทำงาน item ของซอฟท์แวร์ openHAB ที่ประกาศไว้ จะทำการ Publish และ Subscribe Topic ตาม Topic ที่ผูกไว้กับ item ต่างๆ โดย item ใดที่มี MQTT Binding ในทิศทาง inbound item นั้นก็มีหน้าที่รับค่าเข้ามาหรือ Subscribe ส่วน item ที่มี MQTT Binding ในทิศทาง outbound item นั้นก็มีหน้าที่ส่งค่าออกไปหรือ Publish ทำให้ MQTT Broker รู้ตอนนี้ว่ามี Topic ใดบ้างที่อยู่ในระบบ จากนั้นเมื่อทำการเชื่อมต่อบอร์ด Network I/O Gateway และ Network I/O Node เข้าสู่ระบบแล้ว บอร์ด Network I/O Gateway เริ่มทำการ Publish และ Subscribe ค่าหรือคำสั่งต่างๆ ตาม Topic ที่เกี่ยวข้อง ไปยังเซิร์ฟเวอร์ ทำให้ขณะนี้ MQTT Broker มีข้อมูลของการ Publish และ Subscribe โดยสมบูรณ์ กล่าวคือ MQTT Broker รู้ว่าในระบบมี Topic ใดบ้าง และแต่ละ Topic มีอุปกรณ์ใด Publish หรือ Subscribe บ้าง

3.4.2 การส่งค่าจากเซ็นเซอร์ไปยังแอปพลิเคชั่น

เหตุการณ์สมมติ เซ็นเซอร์ส่งข้อมูลไปแสดงผลบนแอปพลิเคชั่น ซึ่งใช้ Topic “/home/sensor1” โดยอาศัยแผนผังการเชื่อมต่อดังรูป 3.3

1) บอร์ด Network I/O Node อ่านค่าจากเซ็นเซอร์จากนั้นส่งค่าไปยังบอร์ด Network I/O Gateway ผ่านสัญญาณไร้สาย 2.4 GHz
2) บอร์ด Network I/O Gateway รับข้อมูลซึ่งเป็นค่าจากเซ็นเซอร์มาสร้างเป็นแพ็กเกต MQTT แล้ว publish ค่าไปยัง Topic “/home/sensor1” ผ่านสายแลนไปยังอินเทอร์เน็ต
3) MQTT Broker ได้รับแพ็กเกต MQTT จากนั้นจะทำการแปลงแพ็กเกตให้อยู่ในรูปแบบที่ openHAB สามารถเข้าใจได้โดยใช้กระบวนการ MQTT Binding
4) ซอฟท์แวร์ openHAB ได้รับค่าจากเซ็นเซอร์จากนั้นค่าจะถูกส่งไปแสดงบนแอปพลิเคชั่น

3.4.3 การส่งคำสั่งจากแอปพลิเคชั่นเพื่อเปิดปิดเครื่องใช้ไฟฟ้า

เหตุการณ์สมมติ ผู้ใช้สั่งเปิดหลอดไฟ โดยหลอดไฟใช้ Topic “/home/light1” ในการควบคุมการเปิดปิดหลอดไฟ โดยอาศัยแผนผังการเชื่อมต่อดังรูป 3.3

1) ผู้ใช้สั่งเปิดหลอดไฟฟ้าจากแอปพลิเคชั่น openHAB ตัวแอปพลิเคชั่น openHAB ก็จะส่งคำสั่งของการเปิดหลอดไฟที่ผูกกับ item ที่มี topic “/home/light1” ไปยังเซิร์ฟเวอร์ที่มีซอฟท์แวร์ openHAB ทำงานอยู่
2) เมื่อซอฟท์แวร์ openHAB ทำการประมวลผลแล้วจะส่งข้อมูลไปยัง MQTT Broker โดยก่อนส่งจะมีการแปลงข้อมูลโดยใช้กระบวนการ MQTT Binding เพื่อให้สามารถสื่อสารกับ MQTT Broker ได้ และสามารถส่งคำสั่งถึงปลายทางได้ถูกต้อง

3) MQTT Broker รับข้อมูลจาก openHAB ตรวจสอบปลายทางและจะทำการ publish message ดังกล่าวไปที่บอร์ด Network I/O Gateway ที่ subscribe topic "/home/light1"

4) เมื่อข้อมูลถูกส่งมาถึงบอร์ด Network I/O Gateway ไมโครคอลโทรลเลอร์จะประมวลผลแล้วส่งคำสั่งเปิดหลอดไฟไปยังบอร์ด Network I/O Node ที่ควบคุมหลอดไฟ โดยส่งสัญญาณผ่านสัญญาณไร้สาย 2.4 GHz

5) บอร์ด Network I/O Node สั่งเปิดหลอดไฟ

3.4.4 การทำงานอัตโนมัติ

เหตุการณ์สมมติ เมื่อค่าที่ส่งมาจากเซ็นเซอร์ถึงค่าที่ตั้งค่าไว้ จะทำให้หลอดไฟเปิดอัตโนมัติ

1) เมื่อค่าจากเซ็นเซอร์ถูกส่งผ่าน Network I/O Node ผ่าน Network I/O Gateway และผ่าน MQTT Broker จนมาถึง openHAB

2) ซอฟท์แวร์ openHAB จะทำการประมวลผลอัตโนมัติตามสิ่งที่กำหนดไว้ในไฟล์ .rules ซึ่งถ้าหากค่าจากเซ็นเซอร์ถึงค่าที่กำหนดไว้ openHAB จะ publish คำสั่งเปิดหลอดไฟไปยัง Network I/O Gateway เพื่อส่งคำสั่งไปยังบอร์ด Network I/O Node เพื่อเปิดหลอดไฟ

3.5 ตัวอย่างส่วนติดต่อผู้ใช้งาน

3.5.1 ตัวอย่างการตั้งค่าไฟล์ที่ใช้สร้างส่วนติดต่อผู้ใช้

คำสั่ง 3.1 การตั้งค่าไฟล์ .items

```
Group     All
Group     o_sensors      (All)
Group     o_lights       (All)
Group     o_dimmers      (All)
Group o_allsensors "เซ็นเซอร์" <sun> (o_sensors)
Group o_alllights "หลอดไฟ" <lights> (o_lights)
Number o_SS1 "เซ็นเซอร์ตัวที่ 1 [%.2f หน่วย]" <sun> (o_allsensors)
{mqtt="<[iot:/homes/sensors/SS1:state:default]"}
Number o_SS2 "เซ็นเซอร์ตัวที่ 2 [%.2f หน่วย]" <sun> (o_allsensors)
{mqtt="<[iot:/homes/sensors/SS2:state:default]"}
Dimmer o_DimmedLight "Dimmer ห้องนั่งเล่น [%d %%]" <slider>
{mqtt=">[iot:/homes/switches/dimmer:state:*:default]"}
Switch o_led1 "หลอดไฟห้องรับแขก" <switch> (o_alllights)
{mqtt=">[iot:/homes/switches/led1:command:ON:1],>[iot:/homes/s
witches/led1:command:OFF:0]"}
Switch o_led2 "หลอดไฟสนาม" <switch> (o_alllights)
{mqtt=">[iot:/homes/switches/led2:command:ON:1],>[iot:/homes/s
witches/led2:command:OFF:0]"}
```

คำสั่ง 3.2 การตั้งค่าไฟล์ .sitemap

```
sitemap o label="Smart Home"
{
      Frame label="เซ็นเซอร์"{
            Group item=o_allsensors label="เซ็นเซอร์"
icon="temperature"
      }

      Frame label="หลอดไฟ"{
            Group item=o_alllights label="ชั้นล่าง" icon="switch"
      }
      Frame label="หลอดไฟแบบปรับความสว่างได้"{
            Slider item=o_DimmedLight switchSupport
      }
}
```

3.5.2 ตัวอย่างส่วนติดต่อกับผู้ใช้

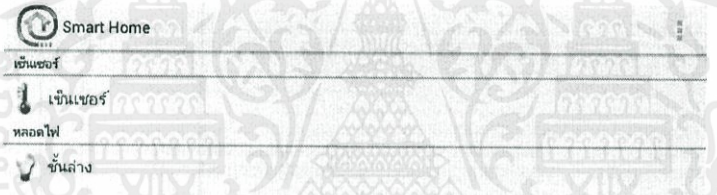
3.5.2.1 เว็บแอปพลิเคชั่น

รูป 3.4 การแสดงผลจากเว็บแอปพลิเคชั่น

3.5.2.2 แอปพลิเคชั่นบนระบบปฏิบัติการ iOS

รูป 3.5 การแสดงผลผ่านแอปพลิเคชั่นบนระบบปฏิบัติการ iOS

3.5.2.3 แอปพลิเคชั่นบนระบบปฏิบัติการ Android



รูป 3.6 การแสดงผลผ่านแอปพลิเคชั่นบนระบบปฏิบัติการ Android

บทที่ 4

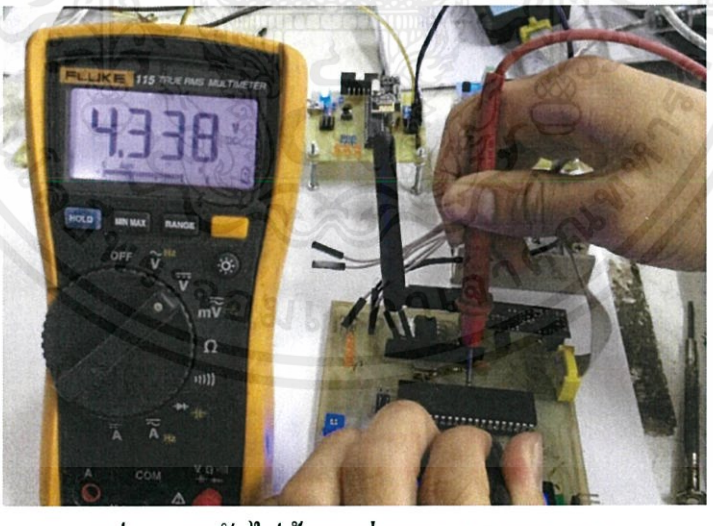
การทดลองและผลการทดลอง

4.1 ฮาร์ดแวร์

4.1.1 ทดลองความถูกต้องของ Microcontroller

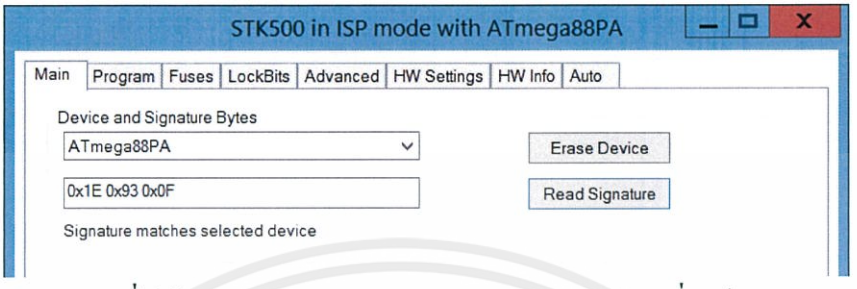
เมื่อทำการสร้างบอร์ด PCB และติดตั้งอุปกรณ์เสร็จต้องทำการตรวจสอบการทำงานของ microcontroller ว่าสามารถทำงานได้ตามปกติหรือไม่เนื่องจากหาก microcontroller ทำงานผิดพลาดจะทำให้การทดลองอื่นๆรวมไปถึงการทำงานของอุปกรณ์อื่นๆ ผิดไปจากความเป็นจริงการทดลองจะแบ่งออกเป็น 2 ส่วนคือ การตรวจสอบทางด้านกายภาพ (physical) และทางด้านตรรกะ (Logical)

การทดลองทางด้านกายภาพ (physical) คือการตรวจสอบว่า microcontroller นั้นอยู่ในสภาวะแวดล้อม (environment) ที่ถูกต้องหรือไม่และพร้อมที่จะรับ Firmware หรือ ROM ที่สร้างขึ้นหรือไม่ทดลองได้โดยการวัดแรงดันไฟฟ้าที่ขา VCC กับ GND ของ Microcontroller เพื่อตรวจสอบความถูกต้องของแรงดัน และต่อ Microcontroller เข้ากับ Programmer เพื่อตรวจสอบความถูกต้องของวงจร

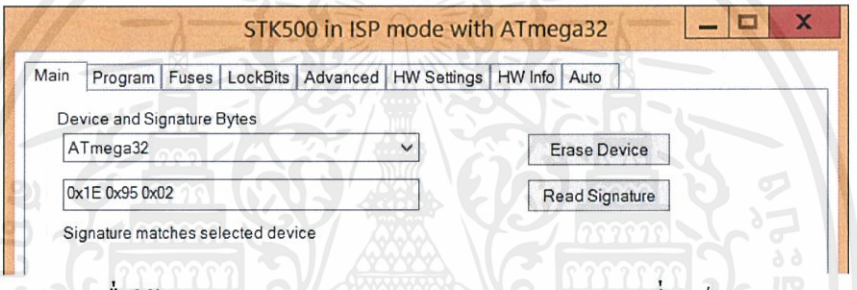


รูป 4.1 แรงดันไฟฟ้าระหว่างขา VCC และ GND

แรงดันไฟฟ้าที่ขา VCC ที่เป็นไฟเลี้ยงของวงจรอยู่ระหว่างระดับที่กำหนดไว้ (2.7 – 5.2) หมายความว่าแรงดันไฟฟ้าที่จ่ายให้กับอุปกรณ์นั้นเพียงพอ



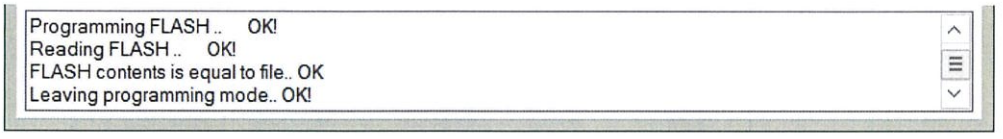
รูป 4.2 เมื่อใช้ Programming ตรวจสอบ Microcontroller ที่บอร์ด Node



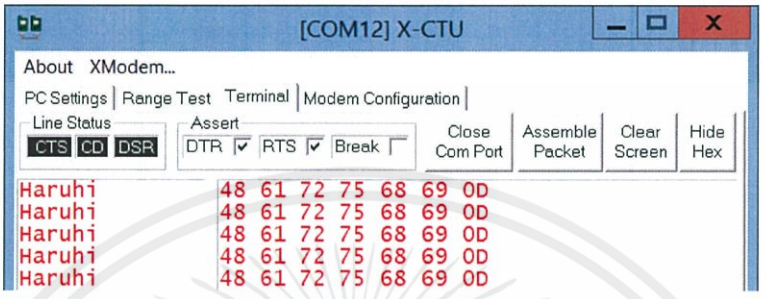
รูป 4.3 เมื่อใช้ Programming ตรวจสอบ Microcontroller ที่บอร์ด Gateway

เมื่อทำการ Read Signature ทั้งในบอร์ด Node และ Gateway ปรากฏรุ่นของ microcontroller ที่ถูกต้องและไม่แสดง Error ใดๆ หมายความว่าวงจรและอุปกรณ์ต่อพ่วงที่จำเป็นต่างๆ ทำงานได้ถูกต้อง

การทดลองด้านตรรกะ (Logical) คือการตรวจสอบว่า Microcontroller นั้นสามารถทำงานตาม Flash memory ที่ Download ลงไปได้อย่างถูกต้องโดยจะทำการเขียนโปรแกรมเพื่อให้ Microcontroller แสดงข้อความออกมาทาง Serial port หาก microcontroller สามารถทำงานได้ถูกต้องหมายความว่า configuration ต่างๆสัมพันธ์กับ Hardware ที่วางแผนไว้



รูป 4.4 สถานะของการ Download Flash memory ใส่บอร์ด



รูป 4.5 ข้อความที่ส่งออกมาจากบอร์ด

จากการทดลองแสดงให้เห็นว่า microcontroller สามารถทำงานได้ถูกต้องเนื่องจากสามารถทำงานตามที่เขียนโปรแกรมลงไปได้อย่างถูกต้องและไม่มี Error ใดๆ

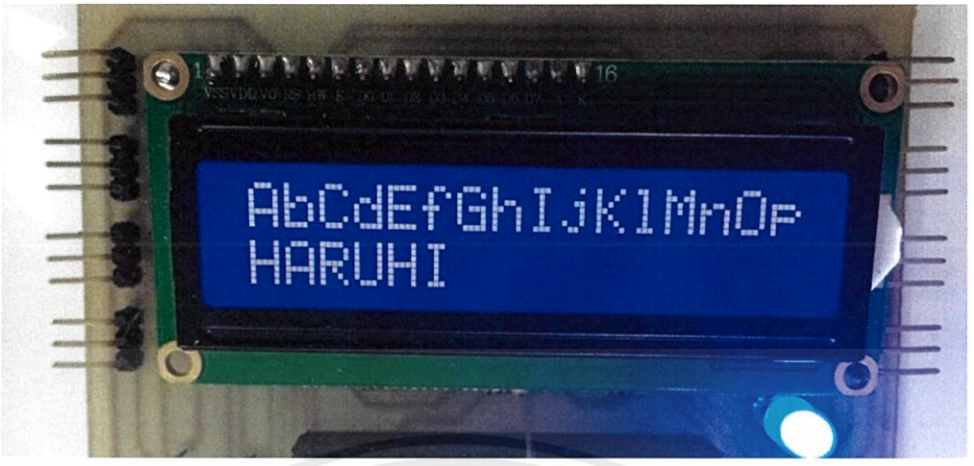
4.1.2 ทดลองการทำงานของจอ LCD

บนบอร์ด Gateway จะมีจอ LCD ชนิด Character ขนาด 16 x 2 ติดตั้งอยู่เพื่อแสดงข้อความโดยสามารถแสดงตัวอักษรตามที่ microcontroller ส่งได้ สามารถทดลองได้ด้วยการสั่งให้จอแสดงข้อความออกมา

```
LCDGotoXY(0,1);
LCDstring("HARUHI",6);

LCDGotoXY(0,0);
LCDstring("AbCdEfGhIjKlMnOp",16);
```

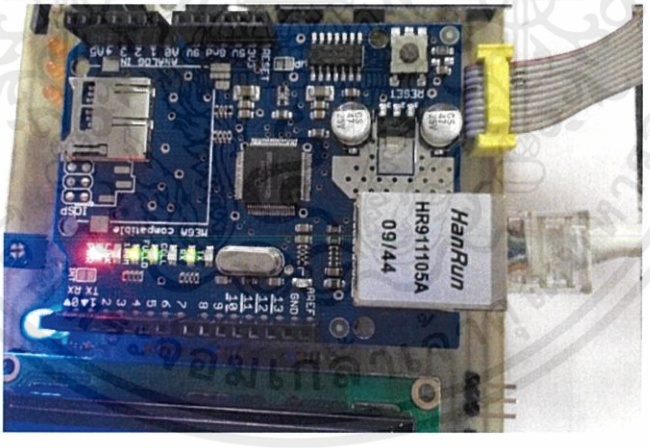
รูป 4.6 code ที่ใช้ในการส่งจอ LCD



รูป 4.7 ผลลัพธ์ของการสั่งจอ LCD

4.1.3 ทดลองการทำงานของ Ethernet shield

Ethernet shield คืออุปกรณ์ที่ใช้ในการเชื่อมต่อกับ Network โดย interface กับ microcontroller ผ่าน SPI BUS โดย Ethernet shield นี้สามารถเชื่อต่อกับระบบ LAN โดยผ่านโพรโตคอล TCP/IP ได้วิธีการทดลองคือติดตั้ง บอร์ดเข้ากับระบบ LAN และลอง ping หาบอร์ด



รูป 4.8 Ethernet shield ขณะทำงาน

```
Pinging 192.168.1.120 with 32 bytes of data:
Reply from 192.168.1.120: bytes=32 time<1ms TTL=128
Reply from 192.168.1.120: bytes=32 time<1ms TTL=128
Reply from 192.168.1.120: bytes=32 time=3ms TTL=128
Reply from 192.168.1.120: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.120:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 3ms, Average = 0ms
```

รูป 4.9 การ ping ไปยัง Ethernet shield

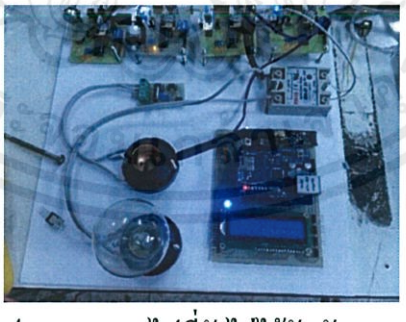
```
Connecting to 192.168.1.120 ...
Connected to 192.168.1.120
zHi. this is W5100
Connection closed
```

รูป 4.10 การ connect ไปยัง Network I/O Gateway

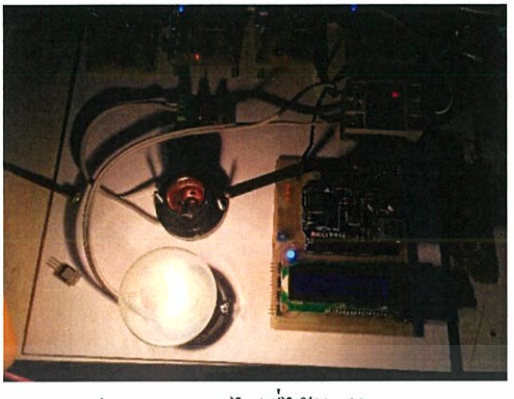
การทำงานของ Ethernet shield เป็นไปตามที่คาดหวังทุกอย่างเนื่องจากสามารถ ping หา และตัว Network I/O สามารถ Connect และ Listening บน Network ได้

4.1.4 ทดลองการทำงานของ Digital AC output

AC output ในที่นี้จะใช้ Solid State Relay เป็นอุปกรณ์ในการขับไฟกระแสสลับเพื่อใช้กับเครื่องใช้ไฟฟ้าภายในบ้านซึ่ง Solid State Relay จะรับสัญญาณ digital มาจาก microcontroller ในการทดลองจะใช้หลอดไฟ 40 W 220 VAC เป็นตัวทดลองและใช้บอร์ด Node ในการจ่ายสัญญาณ



รูป 4.11 หลอดไฟที่ยังไม่ได้รับสัญญาณ

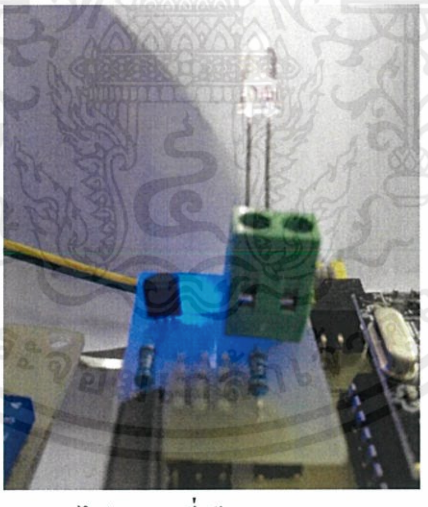


รูป 4.12 หลอดไฟที่ได้รับสัญญาณ

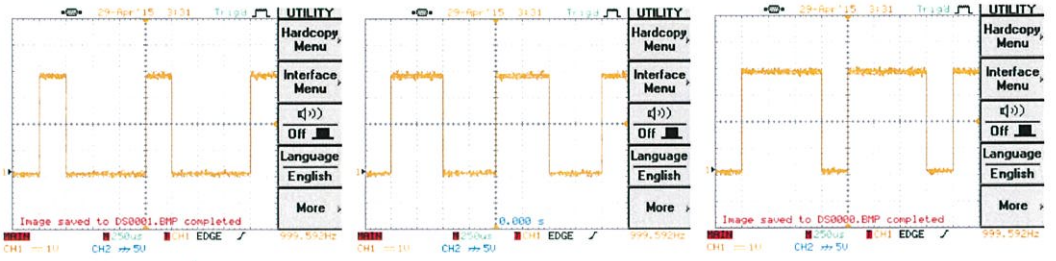
ผลการทดลองปรากฏว่า microcontroller สามารถควบคุมไฟฟ้ากระแสสลับ ได้โดยผ่าน Solid State Relay ซึ่งตามคุณสมบัติของ Relay จะสามารถควบคุมพลังงานไฟฟ้าได้ถึง 6000 วัตต์

4.1.5 ทดลองการทำงานของ Analog DC output (PWM)

Analog DC output คือการควบคุมไฟฟ้ากระแสตรงโดยสามารถกำหนดพลังงานโดย Duty Cycle ในที่นี้จะใช้หลอดไฟ LED ในการแสดงผลของ Duty Cycle ที่เปลี่ยนแปลง



รูป 4.13 ภาพหลอดไฟ LED ที่เป็น Analog DC output (PWM)

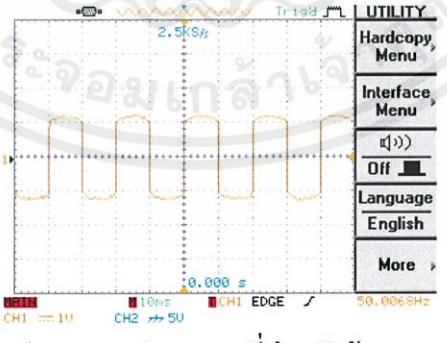


รูป 4.14 สัญญาณที่จ่ายให้กับ หลอด LED โดยผ่าน Transistor โดยมี Duty Cycle เป็น 25% 50% 75% ตามลำดับเรียงจากซ้ายไปขวา

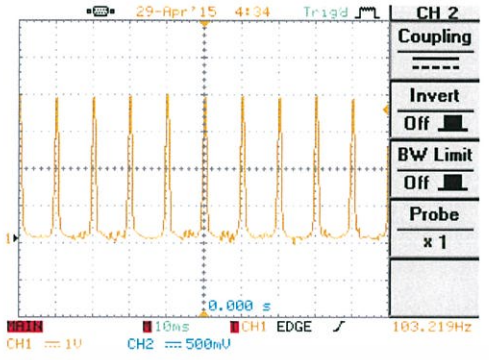
จากการทดลองปรากฏว่าสัญญาณที่สร้างขึ้นมาตรงตามที่คาดไว้โดยหลอด LED เพิ่มและลดความสว่างตาม Duty Cycle ที่เปลี่ยนแปลงหากตามคุณสมบัติของ transistor จะสามารถจ่ายไฟได้สูงสุด 5V 2A แต่เนื่องจากใช้ Adaptor 5V เป็นแหล่งพลังงานอาจทำให้แรงดันตกได้หากต้องจ่ายกระแสสูงมาก

4.1.6 ทดลองการทำงานของ Analog AC output (Dimmer)

Analog AC output คือการควบคุมไฟฟ้ากระแสสลับโดยสามารถกำหนดพลังงานได้โดยอาศัยหลักการของ Zero – crossing detection โดย opto – diac เป็น input ให้กับ microcontroller และขับไฟฟ้ากระแสสลับโดยใช้ triac ร่วมกับ diac โดยกำหนดเวลาในการติดดับเพื่อกำหนดพลังงานที่จะจ่ายให้กับอุปกรณ์ในที่นี้จะใช้หลอดไฟขนาด 40W เป็นตัวทดลอง การทดลองจะแบ่งเป็น 2 ส่วน คือส่วน input และ output ในส่วนของ input จะให้ microcontroller รับสัญญาณ Zero – crossing จาก diac input opto



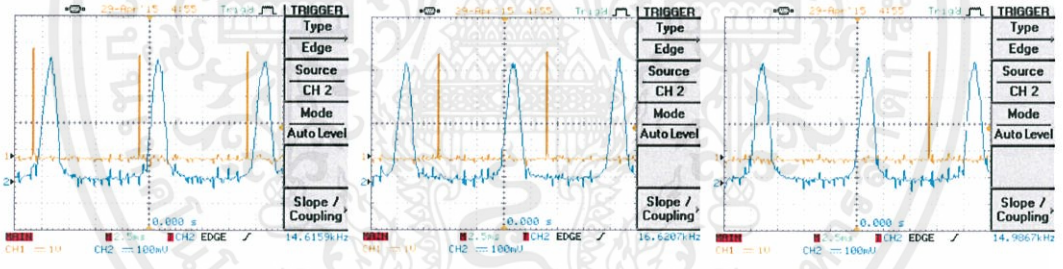
รูป 4.15 แรงดัน input ที่ผ่านตัวต้านทาน



รูป 4.16 แสดงให้เห็นถึงแรงดันเมื่อผ่าน diac input opto ออกมา

จากรูป 4.16 เป็นสัญญาณที่นำเข้ามาประมวลผลบน microcontroller โดยใช้หลักการของ external interrupt โดยจะจับสัญญาณที่เป็นขอบขาขึ้นซึ่งเป็นตำแหน่งของ Zero – crossing หรือตำแหน่งที่ Line และ neutral มีค่าแรงดันเท่ากัน

ในส่วนของ output จะทำการประมวลผลเมื่อได้รับสัญญาณ Zero – crossing จะทำการคำนวณเวลาที่ต้องทำการ delay ก่อนเปิดการทำงานของ triac โดยผ่าน opto – diac



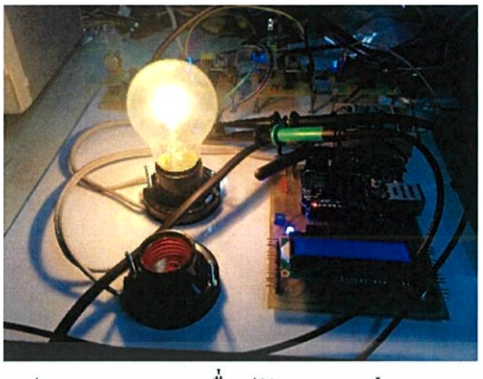
(a) (b) (c)

รูป 4.17 output ของ microcontroller (สีเหลือง) เมื่อเปรียบเทียบ กับ input (สีฟ้า) ในความสว่างต่างๆ เรียงจากมากไปน้อย (ซ้ายไปขวา)

(a) ที่ความสว่าง 90 %

(b) ที่ความสว่าง 10 %

(c) ที่ความสว่าง 50 %



รูป 4.18 dimmer เมื่อปรับความสว่าง 80 %

รูป 4.19 dimmer เมื่อปรับความสว่าง 10 %

Analog AC output สามารถทำงานได้ตามที่สั่งการแต่อาจมีบางจังหวะที่ไม่สามารถจับสัญญาณ zero – crossing ได้ทำให้เกิดการกระพริบเกิดขึ้นแต่เกิดขึ้นได้ไม่บ่อยนัก

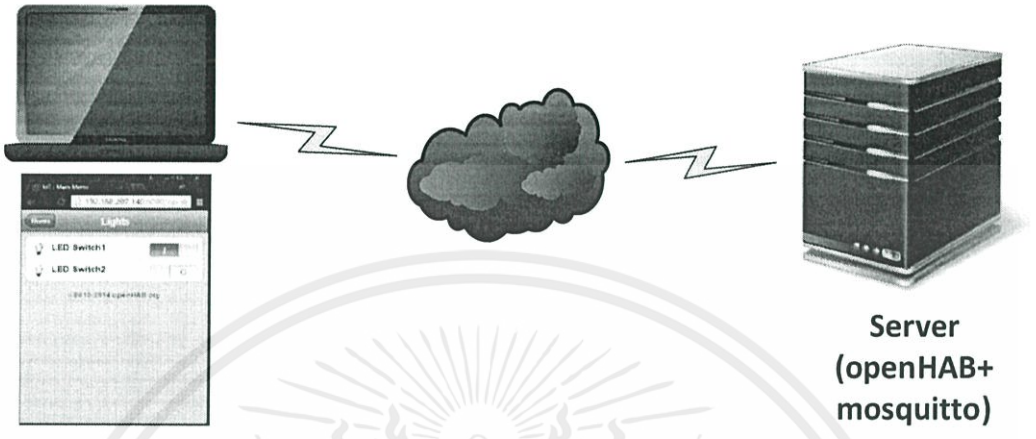
4.2 การทดลองระบบโดยรวม

การทดลองจะแบ่งออกเป็น 2 ช่วงคือ

1) ช่วงที่ 1 ช่วงที่บอร์ด Network I/O Gateway และ Network I/O Node ยังพัฒนาไม่เสร็จ จะใช้บอร์ด Arduino ทดลองระบบแทนไปก่อน

2) ช่วงที่ 2 ช่วงที่บอร์ด Network I/O Gateway และ Network I/O Node พัฒนาเสร็จแล้ว

4.2.1 การทดลองซอฟท์แวร์ openHAB



รูป 4.20 แผนผังการทดลองซอฟท์แวร์ openHAB

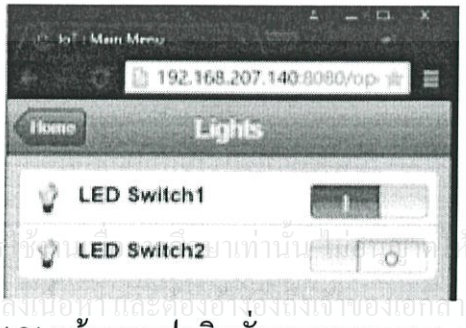
4.2.1.1 รูปแบบการทดลอง

หลังจากติดตั้งซอฟท์แวร์ openHAB Runtime และตั้งค่าต่างๆเรียบร้อยแล้ว ให้ดำเนินการดังนี้

1) เปิด terminal แล้วพิมพ์คำสั่ง sudo bash /etc/openhab/start.sh จากนั้นรอจน openHAB จัดเตรียมสภาพแวดล้อมให้เรียบร้อย
2) เปิดเว็บเบราว์เซอร์ แล้วเข้าไปที่

http://___Server_IP____:8080/openhab.app?sitemap=sitemap_name

4.2.1.2 ผลการทดลอง



รูป 4.21 หน้าจอแอปพลิเคชั่นของการทดสอง 4.2.1

4.2.2 การทดลองซอฟท์แวร์ mosquitto

รูป 4.22 แผนผังการทดลองซอฟท์แวร์ mosquitto

4.2.2.1 รูปแบบการทดลอง

ทดลองโดยใช้คอมพิวเตอร์ที่ติดตั้งซอฟท์แวร์ mosquitto ติดต่อสื่อสารกับเซิร์ฟเวอร์ที่ติดตั้งซอฟท์แวร์ mosquitto ดังนี้

1) ทดลองคำสั่ง mosquitto_pub ดังนี้

mosquitto_pub -h __server_ip__ -d -t /homes/switches/light1 -m 1

2) ทดลองคำสั่ง mosquitto_sub ดังนี้

mosquitto_sub -h __server_ip__ -d -t /homes/switches/light1

4.2.2.2 ผลการทดลอง

```
st047@st047-virtual-machine:~$ mosquitto_pub -h 192.168.207.140 -d -t /homes/swi
tches/light1 -m 1
Client mosqpub/35940-st047-vir sending CONNECT
Client mosqpub/35940-st047-vir received CONNACK
Client mosqpub/35940-st047-vir sending PUBLISH (d0, q0, r0, m1, '/homes/switches
/light1', ... (1 bytes))
Client mosqpub/35940-st047-vir sending DISCONNECT
st047@st047-virtual-machine:~$ mosquitto_pub -h 192.168.207.140 -d -t /homes/swi
tches/light1 -m 0
Client mosqpub/35942-st047-vir sending CONNECT
Client mosqpub/35942-st047-vir received CONNACK
Client mosqpub/35942-st047-vir sending PUBLISH (d0, q0, r0, m1, '/homes/switches
/light1', ... (1 bytes))
Client mosqpub/35942-st047-vir sending DISCONNECT
st047@st047-virtual-machine:~$
```


รูป 4.23 ผลจากการใช้คำสั่ง mosquitto_pub

```
st047@st047-virtual-machine:~$ mosquitto_sub -h 192.168.207.140 -d -t /homes/swi
tches/light1
Client mosqsub/35931-st047-vir sending CONNECT
Client mosqsub/35931-st047-vir received CONNACK
Client mosqsub/35931-st047-vir sending SUBSCRIBE (Mid: 1, Topic: /homes/switches
/light1, QoS: 0)
Client mosqsub/35931-st047-vir received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/35931-st047-vir received PUBLISH (d0, q0, r0, m0, '/homes/switche
s/light1', ... (1 bytes))
1
Client mosqsub/35931-st047-vir received PUBLISH (d0, q0, r0, m0, '/homes/switche
s/light1', ... (1 bytes))
0
Client mosqsub/35931-st047-vir sending PINGREQ
Client mosqsub/35931-st047-vir received PINGRESP
```

รูป 4.24 ผลจากการใช้คำสั่ง mosquitto_pub

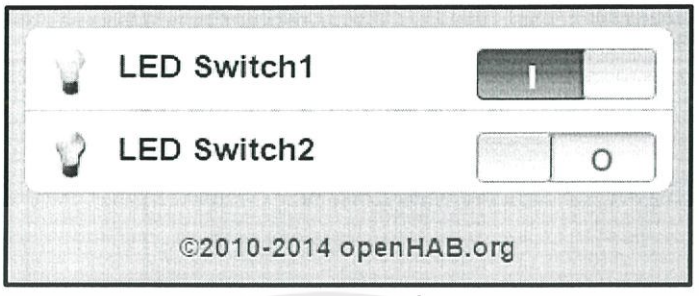
4.2.3 การทดลองการทำงานของซอฟท์แวร์ mosquitto ร่วมกับ openHAB

รูป 4.25 แผนผังการทดลองการทำงานของซอฟท์แวร์ mosquitto ร่วมกับ openHAB

4.2.3.1 รูปแบบการทดลอง

ทดลองสั่งการผ่านคำสั่ง mosquitto_pub แล้วตรวจสอบว่า openHAB ว่าได้รับคำสั่งหรือไม่

4.2.3.2 ผลการทดลอง



รูป 4.26 หน้าจอแอปพลิเคชั่นของการทดสอง 4.2.3

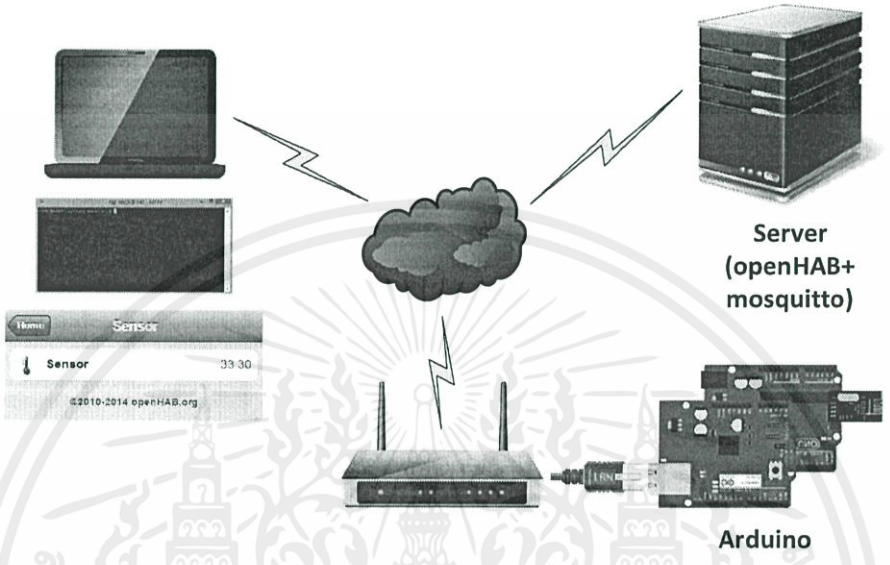
```
Client mosqsub/36212-st047-vir sending SUBSCRIBE (Mid: 1, Topic: /homes/switches
/light1, QoS: 0)
Client mosqsub/36212-st047-vir received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/36212-st047-vir received PUBLISH (d0, q0, r1, m0, '/homes/switche
s/light1', ... (1 bytes))
0
Client mosqsub/36212-st047-vir received PUBLISH (d0, q0, r0, m0, '/homes/switche
s/light1', ... (1 bytes))
1
Client mosqsub/36212-st047-vir received PUBLISH (d0, q0, r0, m0, '/homes/switche
s/light1', ... (1 bytes))
0
```

รูป 4.27 ทดลองการเปิดปิดหลอดไฟหลอดที่ 1

```
Client mosqsub/36215-st047-vir sending SUBSCRIBE (Mid: 1, Topic: /homes/switches
/light2, QoS: 0)
Client mosqsub/36215-st047-vir received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/36215-st047-vir received PUBLISH (d0, q0, r1, m0, '/homes/switche
s/light2', ... (1 bytes))
0
Client mosqsub/36215-st047-vir received PUBLISH (d0, q0, r0, m0, '/homes/switche
s/light2', ... (1 bytes))
1
Client mosqsub/36215-st047-vir received PUBLISH (d0, q0, r0, m0, '/homes/switche
s/light2', ... (1 bytes))
0
```

รูป 4.28 ทดลองการเปิดปิดหลอดไฟหลอดที่ 2

4.2.4 การทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และ บอร์ด Arduino 1 บอร์ด ผ่าน Ethernet



รูป 4.29 การทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และ บอร์ด Arduino 1 บอร์ด ผ่าน Ethernet

4.2.4.1 รูปแบบการทดลอง

ส่งค่าที่ได้จากเซ็นเซอร์ไปยังเซิร์ฟเวอร์และแสดงค่า

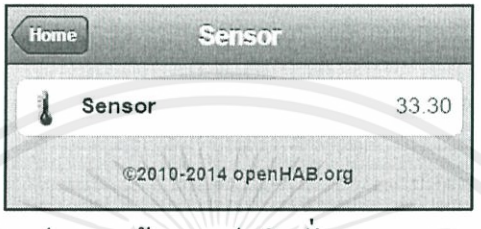
4.2.4.2 ผลการทดลอง

```
Published data : 33.5
Published data : 33.6
Published data : 33.3
Published data : 33.8
```

รูป 4.30 บอร์ด Arduino ทำการ Publish ค่าที่ได้จากเซ็นเซอร์

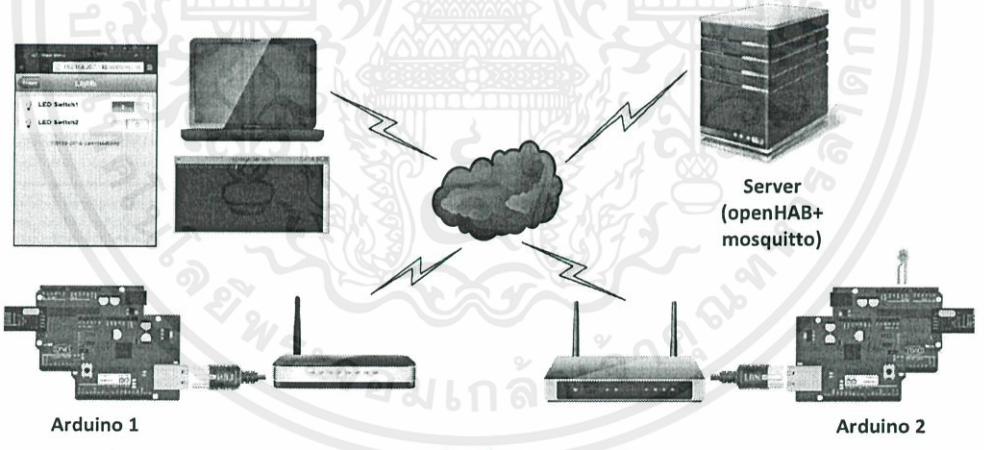
```
Client mosqsub/3402-st047-virt sending PINGREQ
Client mosqsub/3402-st047-virt received PINGRESP
Client mosqsub/3402-st047-virt received PUBLISH (d0, q0, r0, m0, '/test', ... (4 bytes))
33.8
Client mosqsub/3402-st047-virt received PUBLISH (d0, q0, r0, m0, '/test', ... (4 bytes))
33.5
Client mosqsub/3402-st047-virt received PUBLISH (d0, q0, r0, m0, '/test', ... (4 bytes))
33.6
Client mosqsub/3402-st047-virt received PUBLISH (d0, q0, r0, m0, '/test', ... (4 bytes))
33.3
```

รูป 4.31 ตรวจสอบค่าที่ MQTT Broker ได้รับจาก Topic "/test"



รูป 4.32 หน้าจอแอปพลิเคชั่นของการทดสอง 4.2.4

4.2.5 การทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และ Arduino 2 บอร์ด ผ่าน Ethernet



รูป 4.33 แผนผังการทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และ Arduino 2 บอร์ด ผ่าน Ethernet

4.2.5.1 รูปแบบการทดลอง

Arduino บอร์ดที่ 1 ส่งคำสั่งเปิด-ปิดหลอดไปยังเซิร์ฟเวอร์และสั่งให้เปิดหลอด LED บน Arduino บอร์ดที่ 2

4.2.5.2 ผลการทดลอง

```
Sent command : 1
To Topic : /light
```

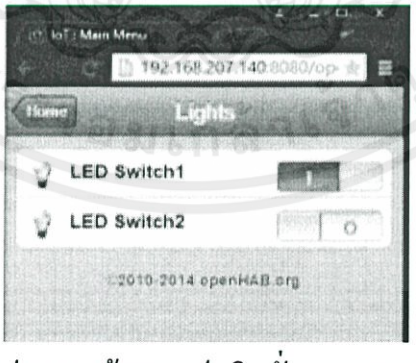
รูป 4.34 บอร์ด Arduino 1 ส่งคำสั่งไปยัง MQTT Broker เพื่อให้เปิดหลอด LED บนบอร์ด Arduino 2

```
Client mosqsub/3689-st047-virt received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/3689-st047-virt received PUBLISH (d0, q0, r0, m0, '/light', ... (1 bytes))
1
```

รูป 4.35 ตรวจสอบคำสั่งที่สั่งให้เปิดหลอด LED

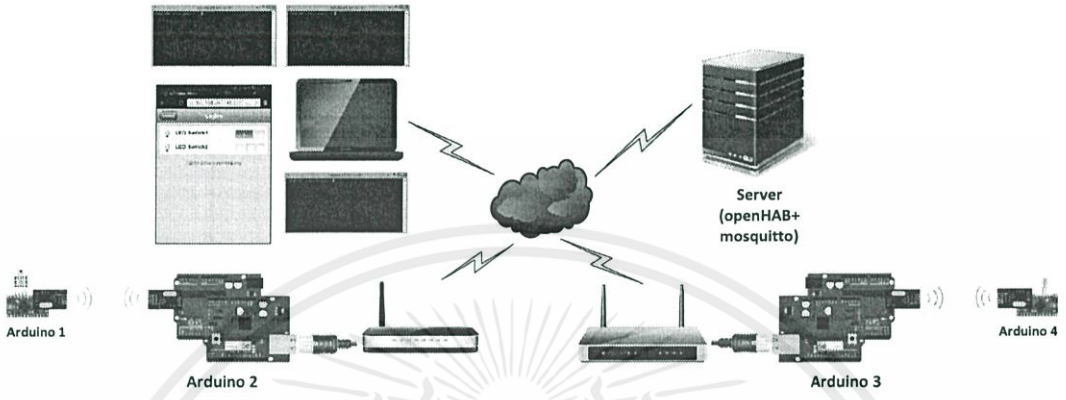
```
Received command : 1
From Topic : /light
```

รูป 4.36 บอร์ด Arduino 2 ได้รับคำสั่งที่ส่งมาจากบอร์ด Arduino 1



รูป 4.37 หน้าจอแอปพลิเคชั่นของการทดสอง 4.2.5

4.2.6 การทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และบอร์ด Arduino 4 บอร์ด ผ่าน Ethernet และ RF 2.4 Ghz



รูป 4.38 แผนผังการทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และบอร์ด Arduino 4 บอร์ด ผ่าน Ethernet และ RF 2.4 Ghz

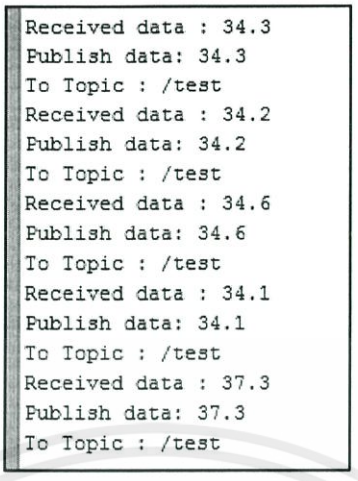
4.2.6.1 รูปแบบการทดลอง

Arduino บอร์ดที่ 1 รับค่าแสงผ่านเซ็นเซอร์ จากนั้นค่าจะส่งไปที่ Arduino บอร์ดที่ 2 ผ่าน RF 2.4 GHz จากนั้นค่าจะถูกส่งผ่านอินเทอร์เน็ตไปยังเซิร์ฟเวอร์ เซิร์ฟเวอร์จะประมวลผลว่าค่าที่ได้ถึงเกณฑ์ให้เปิดปิดหลอดไฟหรือไม่ เมื่อได้ผลแล้วแล้วคำสั่งดังกล่าวจะถูกส่งไปยัง Arduino บอร์ดที่ 3 ผ่านอินเทอร์เน็ต จากนั้นจึงส่งไปยัง Arduino บอร์ดที่ 4 เพื่อเปิดหรือปิดหรือ LED ผ่าน RF 2.4 GHz

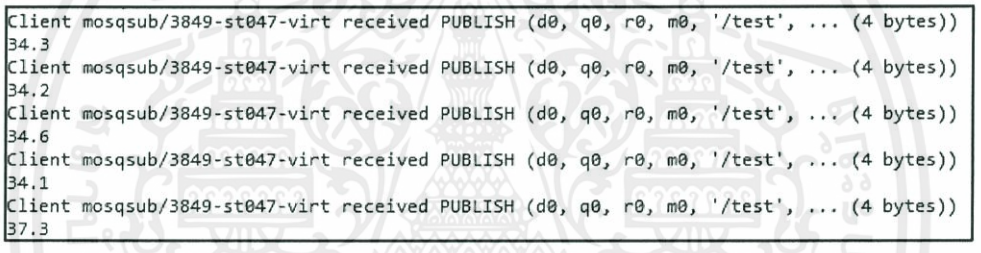
4.2.6.2 ผลการทดลอง

```
Sent data : 34.3
Sent data : 34.2
Sent data : 34.6
Sent data : 34.1
Sent data : 37.3
```

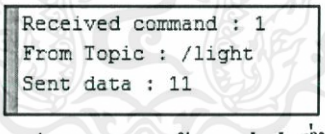
รูป 4.39 บอร์ด Arduino 1 ส่งข้อมูลจากเซ็นเซอร์ไปยังบอร์ด Arduino 2 ผ่าน RF 2.4 GHz



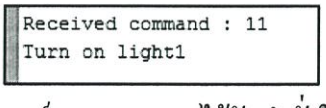
รูป 4.40 บอร์ด Arduino 2 รับค่าจากบอร์ด Arduino 1 แล้วทำการ Publish ข้อมูลไปยัง Topic "/test"



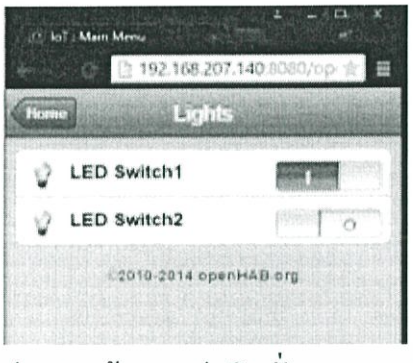
รูป 4.41 ตรวจสอบการส่งข้อมูลเข้ามาใน Topic "/test"



รูป 4.42 openHAB ทำการประมวลผล แล้วพบว่าค่าที่ได้ เกินกว่าที่กำหนดให้ จริงทำการ Publish คำสั่งเปิดหลอดไฟ (เปิด = 1) ไปยังบอร์ด Arduino 3 ซึ่ง Subscribe Topic "/light" และทำการส่งคำสั่ง 11 ไปยังบอร์ด Arduino 4 เพื่อเปิดหลอด LED

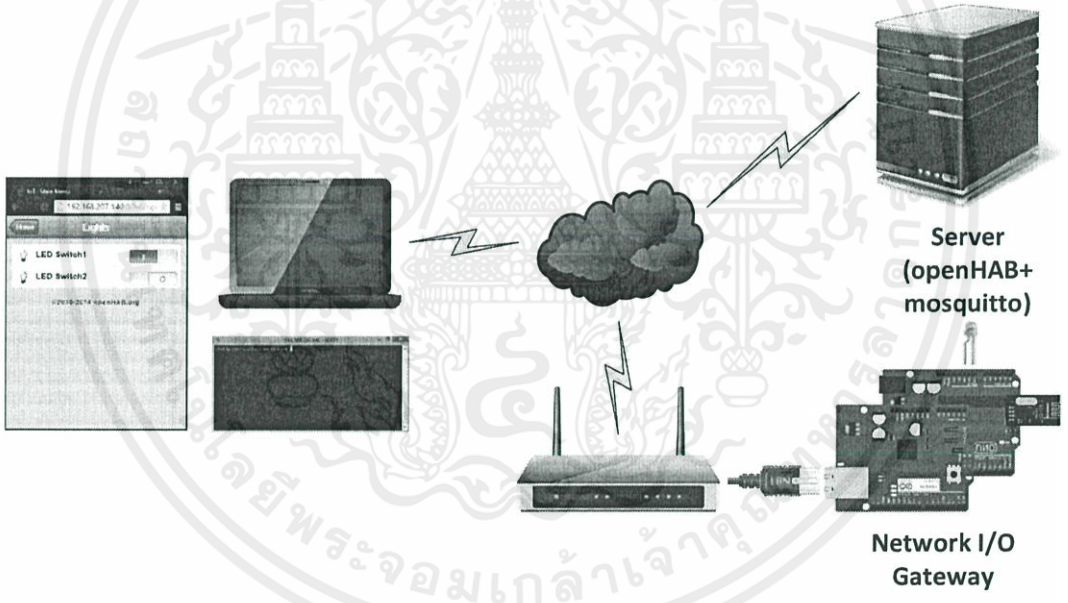


รูป 4.43 บอร์ด Arduino 4 ได้รับคำสั่งให้เปิดหลอด LED



รูป 4.44 หน้าจอแอปพลิเคชั่นของการทดสอง 4.2.6

4.2.7 การทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และ บอร์ด Network I/O Gateway 1 บอร์ด ผ่าน Ethernet



รูป 4.45 แผนผังการทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และ บอร์ด Network I/O Gateway 1 บอร์ด ผ่าน Ethernet

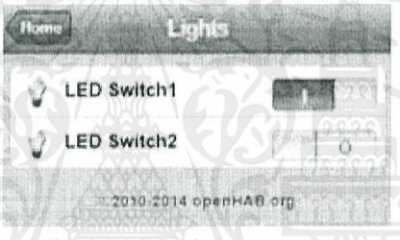
4.2.7.1 รูปแบบการทดลอง

บอร์ด Network I/O Gateway จะส่งค่าจากบอร์ดไปยังเซิร์ฟเวอร์ จากนั้นเซิร์ฟเวอร์จะพิจารณาว่าค่าที่ส่งมาตรงตามเกณฑ์ที่ตั้งไว้หรือไม่ จากนั้นเซิร์ฟเวอร์จะส่งคำสั่งกลับไปยังบอร์ด Network I/O Gateway เพื่อเปิดหรือปิดหลอด LED

4.2.7.2 ผลการทดลอง

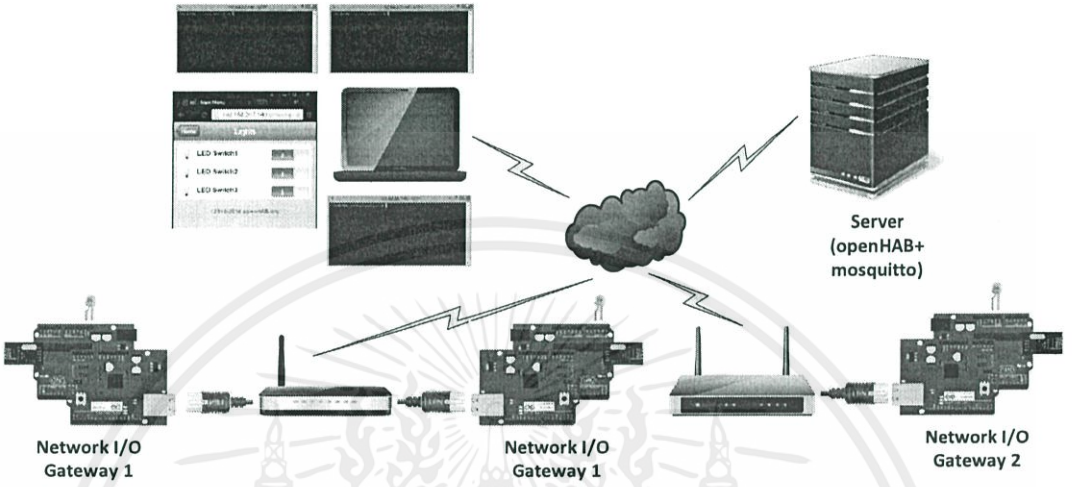
```
Client mosqsub/3205-st047-virt sending SUBSCRIBE (Mid: 1, Topic: /light1, QoS: 0
)
Client mosqsub/3205-st047-virt received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/3205-st047-virt received PUBLISH (d0, q0, r0, m0, '/light1', ...
(1 bytes))
1
```

รูป 4.46 บอร์ด Network I/O Gateway ได้รับคำสั่งเปิดหลอดไฟจากแอปพลิเคชั่น



รูป 4.47 หน้าจอแอปพลิเคชันของการทดสอง 4.2.7

4.2.8 การทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และ Network I/O Gateway 3 บอร์ด ผ่าน Ethernet

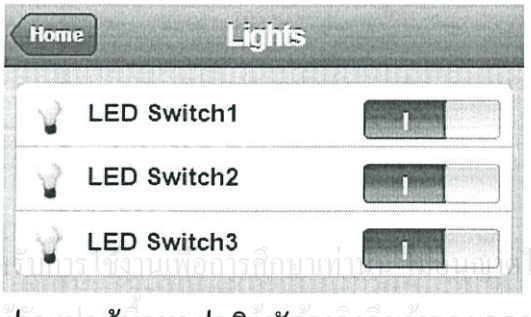


รูป 4.48 แผนผังการทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และ บอร์ด Network I/O Gateway 3 บอร์ด ผ่าน Ethernet

4.2.8.1 รูปแบบการทดลอง

บอร์ด Network I/O Gateway 1 และ 2 อยู่ในเครือข่ายเดียวกัน และบอร์ดที่ 3 อยู่อีกเครือข่ายหนึ่ง โดยให้บอร์ดที่ 1 ส่งคำสั่งไปยังบอร์ดที่ 2 เพื่อเปิดหลอด LED และ ให้บอร์ดที่ 2 ส่งคำสั่งไปยังบอร์ดที่ 3 เพื่อเปิดหลอด LED จากนั้นให้บอร์ดที่ 3 ส่งคำสั่งเพื่อเปิดหลอด LED บนบอร์ดที่ 1

4.2.8.2 ผลการทดลอง



รูป 4.49 หน้าจอแอปพลิเคชันของการทดสอง 4.2.7

```
Client mosqsub/3205-st047-virt sending SUBSCRIBE (Mid: 1, Topic: /light1, QoS: 0
)
Client mosqsub/3205-st047-virt received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/3205-st047-virt received PUBLISH (d0, q0, r0, m0, '/light1', ...
(1 bytes))
1
```

รูป 4.50 บอร์ด Network I/O Gateway 1 ได้รับคำสั่งเปิดหลอดไฟจากแอปพลิเคชั่น

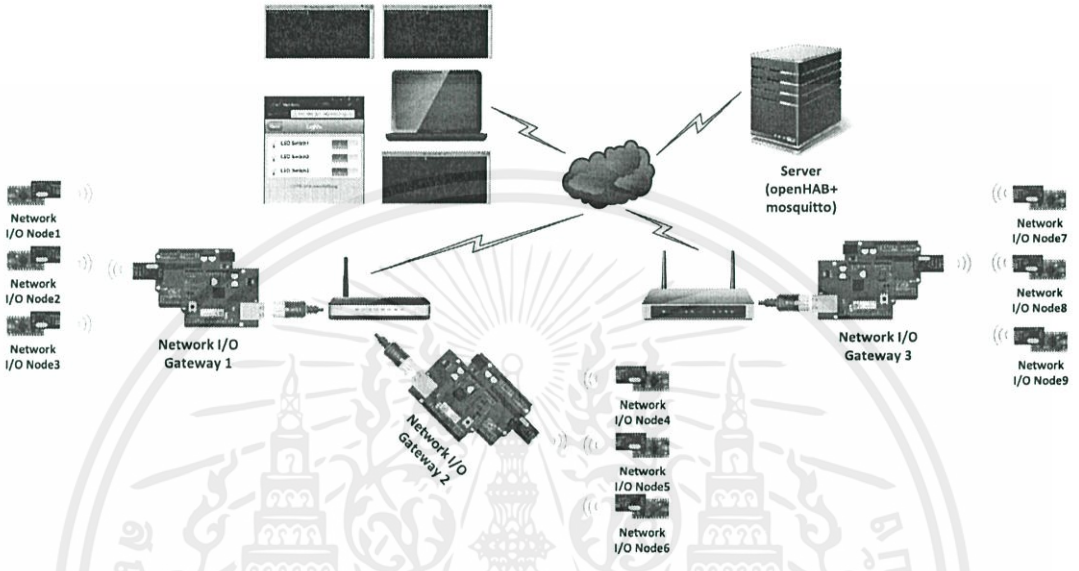
```
Client mosqsub/3207-st047-virt sending SUBSCRIBE (Mid: 1, Topic: /light2, QoS: 0
)
Client mosqsub/3207-st047-virt received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/3207-st047-virt received PUBLISH (d0, q0, r0, m0, '/light2', ...
(1 bytes))
1
```

รูป 4.51 บอร์ด Network I/O Gateway 2 ได้รับคำสั่งเปิดหลอดไฟจากแอปพลิเคชั่น

```
Client mosqsub/3208-st047-virt sending SUBSCRIBE (Mid: 1, Topic: /light3, QoS: 0
)
Client mosqsub/3208-st047-virt received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/3208-st047-virt received PUBLISH (d0, q0, r0, m0, '/light3', ...
(1 bytes))
1
```

รูป 4.52 บอร์ด Network I/O Gateway 3 ได้รับคำสั่งเปิดหลอดไฟจากแอปพลิเคชั่น

4.2.9 การทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และ Network I/O Gateway 3 บอร์ด โดยแต่ละ Network I/O Gateway เชื่อมต่อกับ Network I/O Node 3 บอร์ด ผ่าน RF 2.4 Ghz



รูป 4.53 แผนผังการทดลองซอฟท์แวร์ mosquitto ร่วมกับ openHAB และ Network I/O Gateway 3 บอร์ด โดยแต่ละ Network I/O Gateway เชื่อมต่อกับ Network I/O Node 3 บอร์ด ผ่าน RF 2.4 Ghz

4.2.9.1 รูปแบบการทดลอง

บอร์ด Network I/O Gateway 1 และ 2 อยู่ในเครือข่ายเดียวกัน และบอร์ดที่ 3 อยู่อีกเครือข่ายหนึ่ง โดยบอร์ด Network I/O Node บอร์ดแรกที่เชื่อมต่อกับบอร์ด Network I/O Gateway แต่ละบอร์ดจะเชื่อมต่อกับเซ็นเซอร์เพื่อรายงานค่าไปยังแอปพลิเคชั่น ส่วนบอร์ด Network I/O Node บอร์ดที่ 2 บอร์ดที่ 5 และบอร์ดที่ 8 จะเชื่อมต่อกับหลอด LED โดยรับคำสั่งเปิดปิดจากแอปพลิเคชั่นโดยตรง ส่วน Network I/O Node บอร์ดที่ 3 บอร์ดที่ 6 และบอร์ดที่ 9 จะเชื่อมต่อกับหลอด LED โดยการเปิดปิดหลอด LED ขึ้นอยู่กับค่าของเซ็นเซอร์ต่างๆ ดังนี้ คำสั่งเปิดปิดหลอด LED ของบอร์ด Network I/O Node 3 ขึ้นอยู่กับค่าเซ็นเซอร์ที่เชื่อมต่อกับบอร์ด Network I/O Node 7, คำสั่งเปิดปิดหลอด LED ของบอร์ด Network I/O Node 6 ขึ้นอยู่กับค่าเซ็นเซอร์ที่เชื่อมต่อกับ

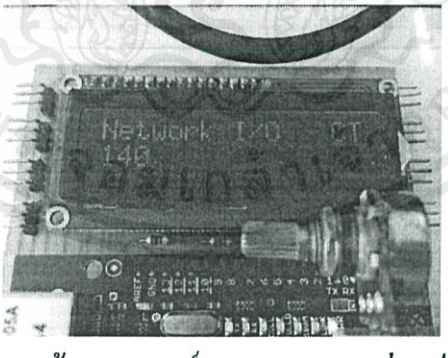
บอร์ด Network I/O Node 1 และคำสั่งเปิดปิดหลอด LED ของบอร์ด Network I/O Node 9 ขึ้นอยู่กับค่าเซ็นเซอร์ที่เชื่อมต่อกับบอร์ด Network I/O Node 4

4.2.9.2 ผลการทดลอง

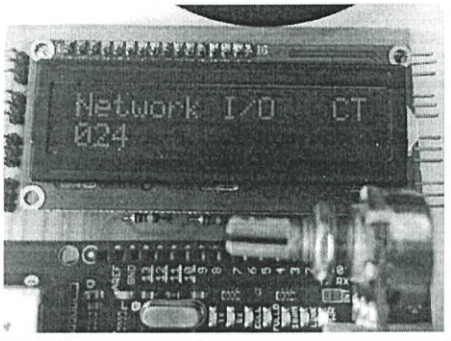
บอร์ด Network I/O Gateway ทั้ง 3 บอร์ด สามารถเชื่อมต่อกับอินเทอร์เน็ตและเชื่อมต่อกับบอร์ด Network I/O Node ต่างๆ ตามแผนผังได้ โดยสามารถสั่งงานผ่านอินเทอร์เน็ตผ่านแอปพลิเคชั่นได้ ค่าจากเซ็นเซอร์ที่ได้จาก Network I/O Node 1, 4 และ 7 สามารถส่งไปที่บอร์ด Network I/O Gateway 1, 2 และ 3 เพื่อส่งค่าไปแสดงยังแอปพลิเคชั่นได้ แต่ Network I/O Node ที่เหลือที่รับคำสั่งจากบอร์ด Network I/O Gateway ไม่สามารถสื่อสารกันได้ ซึ่งเกิดจากบอร์ด Network I/O Gateway ไม่สามารถส่งข้อมูลไปยังบอร์ด Network I/O Node ได้ โดยเป็นข้อจำกัดของโมดูลการสื่อสารไร้สาย nRF24L01+ ที่จำเป็นต้องฮาร์ดแวร์พิเศษเพิ่มเติมจึงจะสามารถทำงานได้สมบูรณ์ ดังนั้นหากมีการนำเอาบอร์ด Network I/O ไปใช้งานจึงจำเป็นต้องกำหนดให้บอร์ด Network I/O Gateway ทำงานอย่างใดอย่างหนึ่งระหว่าง การส่งข้อมูลไปยังบอร์ด Network I/O Node หรือการรับข้อมูลจากบอร์ด Network I/O Node

4.2.10 การทดลอง การทำงานของ module nRF24L01+

NRF24 เป็นอุปกรณ์เชื่อมต่อแบบไร้สายที่ใช้คลื่นความถี่ 2.4 GHz ซึ่งสามารถติดต่อกับ microcontroller ผ่านทาง SPI BUS การทดลองนี้เป็นการทดลองส่งข้อความจาก Node ไปยัง Gateway รายงานค่า sensor ที่เปลี่ยนแปลงไป



รูป 4.54 หน้าจอของบอร์ด Network I/O ก่อนปรับค่า



รูป 4.55 หน้าจอของบอร์ด Network I/O หลังปรับค่า

เมื่อหมุนตัวต้นทานแบบรับค่าได้ที่ใช้เป็นตัวแทนของ sensor ปรากฏว่า node สามารถรายงานค่าของ sensor ที่เปลี่ยนแปลงไปได้อย่างถูกต้อง

ต่อมาเพื่อเป็นการทดสอบการรับค่าจาก sensor หลายๆตัวในเวลาเดียวกันจึงทำการเพิ่ม node ที่เชื่อมต่อกับปุ่มกด เพื่อทำการทดลอง

รูป 4.56 หน้าจอของบอร์ด Network I/O Gateway แสดงค่าที่ได้รับจาก Network I/O Node

ปรากฏว่า Gateway สามารถรับค่าจาก node 2 node ได้ และสามารถแสดงค่าที่เปลี่ยนไปของ node ทั้ง 2 node ได้อย่างถูกต้อง

4.2.11 การทดลอง การทำงานของ Ethernet module ต่อ MQTT broker

MQTT broker เป็น Service ที่ติดตั้งอยู่บน server เพื่อทำงานเกี่ยวกับการเชื่อมต่อที่ใช้ MQTT protocol ดังนั้นหากต้องการให้ Gateway สามารถรับค่าจาก Server ได้ต้องทำให้ Gateway ใช้ MQTT protocol ได้

การทดลองจะทำโดยให้ Gateway จดจำ state ของการเชื่อต่อไว้จะมีทั้งหมด 3 state โดยในแต่ละ state จะแสดงข้อความบนจอ LCD แตกต่างกันไปได้แก่

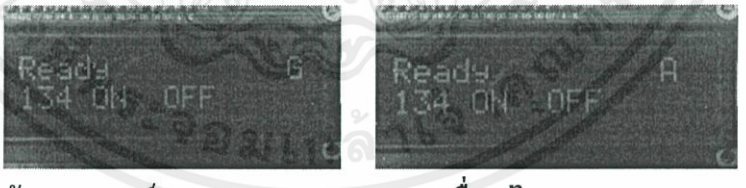
1) Connecting คือการส่ง packet เพื่อขอ connect และรอ Acknowledge

2) Subscribing คือการได้รับ Acknowledge ของ การ connect จากนั้นจะส่ง packet เพื่อขอ Subscribe และรอ Acknowledge

3) Ready คือได้รับ Acknowledge ของการ Subscribe จะทำการส่ง Ping packet และรอรับ Ping Acknowledge ทุกๆ 3 วินาที

รูป 4.57 หน้าจอของบอร์ด Network I/O Gateway เมื่ออยู่ใน Connecting state

รูป 4.58 หน้าจอของบอร์ด Network I/O Gateway เมื่ออยู่ใน Subscribing state

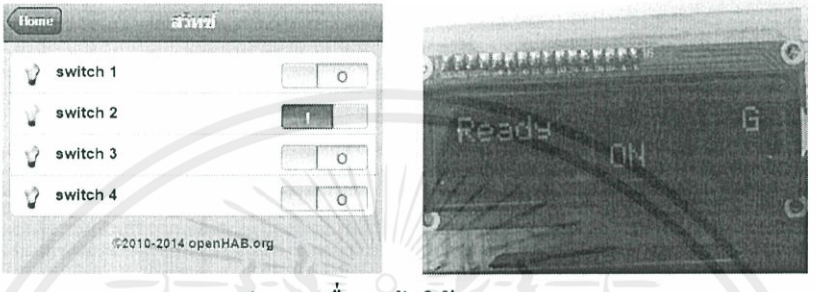


รูป 4.59 หน้าจอของบอร์ด Network I/O Gateway เมื่ออยู่ใน Ready state (A หมายถึงส่ง ping ออกไป แล้ว G หมายถึงได้รับ Ping Acknowledge แล้ว)

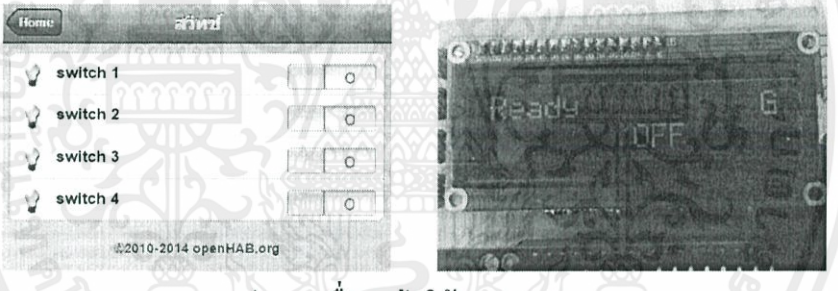
ผลปรากฏว่า Gateway สามารถทำงานได้อย่างถูกต้อง เนื่องจากสามารถเชื่อมต่อกับ server ได้อย่างถูกต้อง และสามารถ connect ใหม่ได้หากการเชื่อมในครั้งนั้นถูกตัดไป

4.2.12 การทดลอง การทำงานของ Ethernet module ต่อ Open HAB

Open Hab เป็น interface ที่เชื่อมต่อกับ MQTT Broker ซึ่งจะสามารถดูค่าและสามารถสั่งการ MQTT Broker ได้อีกด้วยการทดลองนี้เป็นการทดลองเพื่อดูความสัมพันธ์ระหว่าง Open Hab และ Gateway ว่าทำงานสัมพันธ์กันหรือไม่



รูป 4.60 เมื่อกดปุ่มให้ Switch ON



รูป 4.61 เมื่อกดปุ่มให้ Switch OFF

ผลปรากฏว่า Gateway ทำงาน สัมพันธ์กับ openHAB ได้โดยสามารถแสดงค่าได้ตรงกับที่เป็นบน openHAB

บทที่ 5
บทสรุป

5.1 บทสรุป

ปัจจุบันเทคโนโลยี Internet of Things เริ่มได้รับความนิยม และในต่างประเทศเริ่มมีการทำผลิตภัณฑ์ออกมาขายกันบ้างแล้ว และโพรโทคอลหนึ่งที่ได้รับความนิยมนำมาใช้ในการสื่อสารของ Internet of Things ก็คือ โพรโทคอล MQTT ซึ่งจุดเด่นของโพรโทคอลนี้คือ ข้อมูลที่ส่งมีขนาดเล็กและสามารถควบคุมคุณภาพในการส่งข้อมูลได้ ว่าต้องการความรวดเร็วหรือความน่าเชื่อถือมากกว่ากัน และซอฟท์แวร์ openHAB ทำให้ผู้ใช้สามารถควบคุมการทำงานของระบบผ่านอินเทอร์เน็ต รวมทั้งสามารถออกแบบหน้าตาและการทำงานของแอปพลิเคชั่นได้ตามที่ต้องการ

อุปกรณ์รับส่งข้อมูลผ่านเครือข่ายเป็นโครงงานที่นำเอาบอร์ดไมโครคอนโทรลเลอร์ที่พัฒนาขึ้นเองโดยใช้ไมโครโพรเซสเซอร์ตระกูล AVR โดยใช้การสื่อสารผ่านสายแลนและสัญญาณไร้สายความถี่ 2.4 Ghz. ในการติดต่อสื่อสารผ่านอินเทอร์เน็ตจะรับส่งข้อมูลโดยใช้โพรโทคอล MQTT ซึ่งส่งข้อมูลผ่านโพรโทคอล TCP อีกทีหนึ่ง และการติดต่อสื่อสารระหว่างบอร์ด Network I/O Gateway และ บอร์ด Network I/O Node จะสื่อสารผ่านสัญญาณไร้สาย 2.4 Ghz โดยผ่านโมดูล nRF24l01+

สำหรับการนำบอร์ด Network I/O Gateway และ Network I/O Node ไปใช้งานนั้นจะต้องมีระบบที่รองรับการทำงาน ซึ่งเป็นเซิร์ฟเวอร์ที่ติดตั้งซอฟท์แวร์ mosquitto และ openHAB เซ็นเซอร์และโมดูลต่างๆ และซอฟท์แวร์ที่ใช้ตั้งค่าการทำงานของบอร์ด Network I/O Gateway และที่สำคัญต้องมีอินเทอร์เน็ตเพื่อให้สามารถ ติดตาม ควบคุมและสั่งงานทางไกลได้

5.2 ปัญหาอุปสรรคและแนวทางการแก้ไข

1) การสื่อสารผ่านสัญญาณไร้สายผ่าน nRF24l01+ แบบธรรมดา ส่งและรับข้อมูลได้ไม่ไกลเท่าที่ควร จึงต้องเปลี่ยนมาใช้ nRF24l01+ แบบมีเสาอากาศแยกบนบอร์ด Network I/O Gateway
2) ข้อมูลที่เกี่ยวข้องกับการพัฒนาระบบโดยใช้โพรโทคอล MQTT openHAB และ mosquitto ยังมีน้อย จึงต้องอาศัยการลองผิดลองถูกเป็นหลัก
3) สภาพแวดล้อมมีผลต่อการใช้งาน เช่น หากติดตั้งในอาคารที่มีห้องเล็กๆ จำนวนมาก สัญญาณที่ใช้ในการรับส่งข้อมูลอาจจะไปไม่ถึงอุปกรณ์ที่ต้องการ ซึ่งแก้ปัญหาโดยใช้เสา

อากาศที่มีกำลังสูงขึ้น หรือ เพิ่มจำนวน Network I/O Node มารับสัญญาณและส่งสัญญาณไปยัง Network I/O Node ปลายทางที่แท้จริง

4) ปัญหาการสื่อสารระหว่างบอร์ด Network I/O Gateway และ MQTT Broker ในบางครั้งไม่สามารถสื่อสารกันได้ จำเป็นต้องรีเซ็ตการทำงานของเซิร์ฟเวอร์ให้เริ่มทำงานใหม่

5) ปัญหาการสื่อสารระหว่างบอร์ด Network I/O Gateway และ Network I/O Node ที่สามารถเชื่อต่อแบบสองทิศทางเมื่อทำการติดต่อสื่อสารแบบหนึ่งต่อหนึ่ง แต่เมื่อทำการติดต่อสื่อสารแบบหนึ่งต่อหลายโดยให้บอร์ด Network I/O Gateway เป็นศูนย์กลางของการสื่อสารไม่สามารถทำได้ เนื่องจากเป็นข้อจำกัดของโมดูล nRF24l01+ ที่จำเป็นต้องสร้างฮาร์ดแวร์เพิ่มเติม จึงจะทำให้การสื่อสารสมบูรณ์ ซึ่งเมื่อเกิดปัญหานี้ขึ้นทำให้บอร์ด Network I/O Gateway สามารถทำงานได้เป็นผู้ส่งข้อมูลอย่างเดียว หรือผู้รับสัญญาณอย่างเดียวเท่านั้น ไม่สามารถสื่อสารได้ทั้งสองแบบในเวลาใดเวลาหนึ่ง โดยจะเกิดขึ้นเฉพาะการสื่อสารผ่านสัญญาณไร้สาย 2.4 GHz การแก้ปัญหาเบื้องต้นคือ บอร์ด Network I/O Gateway ทำงานได้เพียงหนึ่งโหมดคือให้ทำหน้าที่รับข้อมูลอย่างเดียวหรือจะให้ส่งข้อมูลเพียงอย่างเดียว

5.3 แนวทางการพัฒนาต่อ

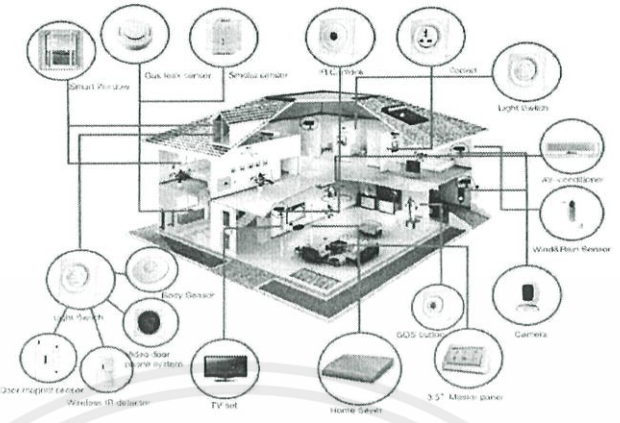
1) ปรับปรุงระบบให้ 1 Server รองรับผู้ใช้ได้หลายคน (ตั้ง Password ต่างกันได้) โดยฟีเจอร์นี้จะสามารถใช้งานได้ใน openHAB 2.0 ซึ่งในขณะจัดทำปริญญานิพนธ์ฉบับนี้ openHAB 2.0 ยังเป็น Version Alpha อยู่ โดยต้องรอให้ทีมงานพัฒนาเสร็จจึงจะนำมาใช้งานได้สมบูรณ์

2) ปรับปรุงระบบรักษาความปลอดภัย โดยการปรับปรุงให้ระบบส่งข้อมูลผ่านโพรโทคอล SSL/TLS ระหว่าง mosquitto และ openHAB และเปิดใช้งาน HTTPS เมื่อใช้งานผ่านเว็บเบราว์เซอร์

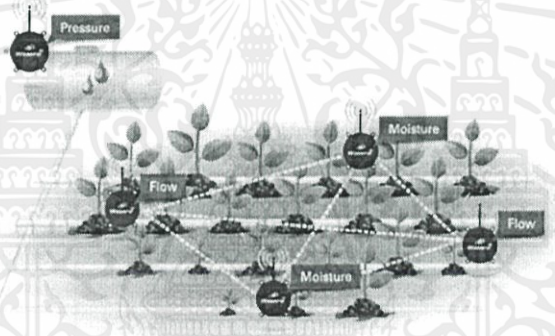
3) พัฒนาให้รองรับการนำไปใช้ในรูปแบบอื่นๆนอกจากการควบคุมและติดตามการทำงานของเครื่องใช้ไฟฟ้า เช่น นำไปใช้ในการเกษตรอัจฉริยะ หรือ Smart farm การนำไปใช้รายงานสภาพอากาศ หรือแจ้งเตือนภัยพิบัติต่างๆ

4) พัฒนาให้บอร์ด Network I/O Gateway และ Network I/O Node มีขนาดกะทัดรัดลง และเพิ่มจำนวนชนิดของโมดูลที่สามารถนำมาเชื่อมต่อให้มากขึ้น เพื่อการนำไปใช้งานได้หลากหลาย

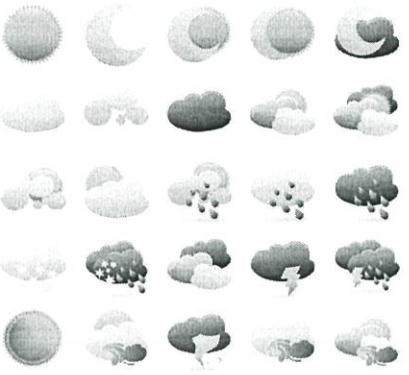
5) พัฒนาซอฟท์แวร์ที่ช่วยให้ผู้ใช้ทั่วไปสามารถออกแบบหน้าตาและรูปแบบการใช้งานสำหรับติดตาม ควบคุมและสั่งการระบบได้ง่ายขึ้น



รูป 5.1 ระบบบ้านอัจฉริยะ



รูป 5.2 ระบบเกษตรอัจฉริยะ



รูป 5.3 ระบบรายงานสภาพอากาศ

บรรณานุกรม

Atmel. 2011. **ATmega88.** [Online].Available : http://www.atmel.com/ devices/atmega88.aspx

Eclipse. 2014. **Configure SSL.** [Online].Available : http://wiki.eclipse.org/Jetty/ Howto/Configure_SSL

Electronichamsters. 2014. **Uber Home Automation w/ Arduino & Pi.** [Online].Available : http://www.instructables.com/id/Uber-Home-Automation-w-Arduino-Pi/

IBM. 2014. **MQTT V3.1 Protocol Specification**. [Online].Available : http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html

Jpmens. **Configure openHAB keystore to use our own TLS server certificates.** [Online].Available : https://gist.github.com/jpmens/8029383

Kalle Löfgren. 2013. **Tutorial - nRF24L01 and AVR**. [Online].Available : http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01

Light R. 2015. **Mosquitto Documentation.** [Online].Available : http://mosquitto.org/documentation.

openHAB. 2015. **Explanation of items.** [Online].Available : https://github.com/openhab/openhab/wiki/Explanation-of-items

openHAB. 2015. **Explanation of Sitemaps.** [Online].Available : https://github.com/openhab/openhab/wiki/Explanation-of-Sitemaps

openHAB. 2015. **MQTTBinding.** [Online].Available :
https://github.com/openhab/openhab/wiki/MQTT-Binding

openHAB. 2015. **Quick Setup Linux OS X.** [Online].Available :
https://github.com/openhab/openhab/wiki/Linux---OS-X

openHAB. 2015. **Samples Item Definitions.** [Online].Available :
https://github.com/openhab/openhab/wiki/Samples-Item-Definitions

openHAB. 2015. **Samples Rules.** [Online].Available :
https://github.com/openhab/openhab/wiki/Samples-Rules

openHAB. 2015. **Samples Sitemap Definitions.** [Online].Available :
https://github.com/openhab/openhab/wiki/Samples-Sitemap-Definitions

Uzrasai**. Internet of Things messaging MQTT 1: Installing mosquitto server.**
[Online].Available : http://lukse.lt/uzrasai/2015-02-internet-of-things-messaging-mqtt-1-installing-mosquitto-server/

Wiznet. 2014. **TCP/IP Chip W5100.** [Online].Available : http://www.wiznet.co.kr/Sub_Modules/en/product/Product_Detail.asp?cate1=5&cate2=7&cate3=26&pid=1011

ภาคผนวก ก
การติดตั้งและใช้งานระบบ

ก.1 การติดตั้งระบบ

ก.1.1 การติดตั้งฮาร์ดแวร์

ทำการเชื่อมต่อบอร์ด Network I/O Node เข้ากับเครื่องใช้ไฟฟ้า หรือเซ็นเซอร์ที่ต้องการติดตามค่า จากนั้นเชื่อมต่อบอร์ด Network I/O

ก.1.2 การติดตั้งระบบเซิร์ฟเวอร์

สำหรับคำสั่งที่ใช้ติดตั้งซอฟท์แวร์ต่างๆ อ้างอิงคำสั่งจากระบบปฏิบัติการ Ubuntu ซึ่งระบบปฏิบัติการที่อยู่คนละตระกูลกับระบบปฏิบัติการ Ubuntu อาจจะใช้คำสั่งเหล่านี้ไม่ได้

ก.1.2.1 การติดตั้งซอฟท์แวร์ openssh-server

ติดตั้งโดยใช้คำสั่ง ดังนี้

ตัวอย่าง ก.1 คำสั่งที่ใช้ในการติดตั้งซอฟท์แวร์ openssh-server

```
sudo apt-get install openssh-server
```

ก.1.2.2 การติดตั้งซอฟท์แวร์ mosquitto

ติดตั้งโดยใช้คำสั่ง ดังนี้

ตัวอย่าง ก.2 คำสั่งที่ใช้ในการติดตั้งซอฟท์แวร์ mosquitto

```
1. sudo apt-add-repository ppa:mosquitto-dev/mosquitto-
ppa
2. sudo apt-get update
3. sudo apt-get install mosquitto mosquitto-clients
python-mosquitto
```

ก.1.2.3 การติดตั้งซอฟท์แวร์ openjdk

ติดตั้งโดยใช้คำสั่ง ดังนี้

ตัวอย่าง ก.3 คำสั่งที่ใช้ในการติดตั้งซอฟท์แวร์ openjdk

```
1. sudo apt-add-repository ppa:mosquitto-dev/mosquitto-
ppa
2. sudo apt-get update
3. sudo apt-get install mosquitto mosquitto-clients
python-mosquitto
```

ก.1.2.4 การติดตั้งซอฟท์แวร์ openHAB

ติดตั้งโดยใช้คำสั่ง ดังนี้

ตัวอย่าง ก.4 คำสั่งที่ใช้ในการติดตั้งซอฟท์แวร์ openHAB

```
1. sudo apt-add-repository ppa:mosquitto-dev/mosquitto-
ppa
2. sudo apt-get update
3. sudo apt-get install mosquitto mosquitto-clients
python-mosquitto
4. sudo echo > "deb http://repository-
openhab.forge.cloudbees.com/release/1.6.2/apt-repo/ " >
/etc/apt/sources.list.d/openhab.list
5. sudo apt-get update
6. sudo apt-get install openhab-runtime
```

ในขั้นต่อมาจะเป็นการติดตั้ง openHAB Runtime ให้สมบูรณ์พร้อมใช้งาน

1) ทำการ download ไฟล์ในส่วน openHAB Runtime (ในขณะที่จัดทำโครงงานเป็น version 1.6.2) ซึ่งประกอบด้วย Runtime Core (distribution-1.6.2-runtime.zip), Addons (distribution-1.6.2-addons.zip) และ Demo Setup (distribution-1.6.2-demo-configuration.zip)

2). ทำการแตกไฟล์ทั้ง 3 ไฟล์

3). ใช้โปรแกรม WinSCP ทำการถ่ายโอนไฟล์ไปยัง server โดยใช้ protocol SFTP สำหรับ โดย

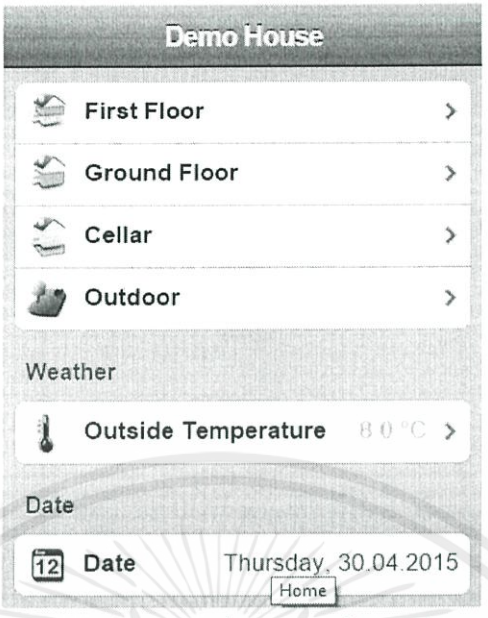
3.1. ไฟล์ที่ได้จากการแตกไฟล์ distribution-1.6.2-addons.zipให้ทำการถ่ายโอนไฟล์ไปไว้ที่ /etc/openhab/

3.2. ไฟล์ที่ได้จากการแตกไฟล์ distribution-1.6.2-addons.zip ให้ทำการถ่ายโอนเฉพาะไฟล์ org.openhab.binding.mqtt-1.6.2.jar ไปที่ /etc/openhab/addons

3.3. ไฟล์ที่ได้จากการแตกไฟล์ distribution-1.6.2-demo-configuration.zip ให้ทำการถ่ายโอนไฟล์ไปไว้ที่ /etc/openhab/

รูป ก.1 ตำแหน่งที่เก็บไฟล์สำคัญของ openHAB

4). จากนั้นทำการทดสอบระบบของ openHAB โดยการรันคำสั่ง bash /etc/openhab/start.sh ผ่าน PuTTY แล้วใช้ Web Browser โดยเข้าไปที่ http://__server_ip__:8080/openhab.app?sitemap=demo ถ้าหากซอฟท์แวร์ทำงานปกติจะได้ผลลัพธ์ดังนี้



รูป ก.2 ตัวอย่างของแอปพลิเคชัน

ก.1.3 การปรับแต่งค่าของระบบ และการสร้างส่วนติดต่อผู้ใช้สำหรับควบคุมระบบ

ก.1.3.1 ซอฟท์แวร์ openHAB

ซอฟท์แวร์ openHAB มีไฟล์ที่เกี่ยวข้องหลากหลายประเภท ซึ่งมีเพียงบางไฟล์ที่ผู้ใช้ต้องทำการสร้างขึ้นเองหรือปรับแก้ไขค่าเพิ่มเติม ซึ่งได้แก่

1) ไฟล์ .items ใช้กำหนด object สำหรับรับค่าหรือส่งค่าระหว่าง openHAB และ MQTT Broker
2) ไฟล์ .sitemap ใช้กำหนดโครงสร้างและหน้าตาของแอปพลิเคชั่น
3) ไฟล์ .rules ใช้กำหนดกฎต่างๆของ openHAB ซึ่งใช้ในการกำหนดให้ระบบทำงานแบบอัตโนมัติ เช่น สั่งให้เปิดไฟเมื่อถึงเวลา 18:00 น. เป็นต้น
4) ไฟล์ openhab.cfg ใช้กำหนดการทำงานของ openHAB
5) ไฟล์ start.sh ใช้สำหรับเริ่มการทำงานของ openHAB Runtime
6) ไฟล์ user.cfg ใช้สำหรับกำหนด username และ password ในการเข้าใช้ระบบของ openHAB
7) ไฟล์ login.cfg ใช้สำหรับการกำหนดชื่อผู้ใช้และรหัสผ่านเพื่อป้องกันผู้ที่ไม่ได้รับอนุญาตเข้ามาใช้งานระบบ

ก.1.3.1.1 การกำหนดค่าในไฟล์ openhab.cfg

ในเบื้องต้นจะทำการกำหนดเพียงส่วน Transport Configuration ซึ่งกำหนดเพื่อให้ติดต่อกับ mosquitto (MQTT Broker)

ซึ่งรูปแบบคำสั่งก่อนที่ผู้ใช้จะปรับค่าจะมีรูปแบบดังนี้

ตัวอย่าง ก.5 ค่าเริ่มต้นภายใน Transport Configuration ของไฟล์ openhab.cfg

```
mqtt:<broker>.url=<url>
mqtt:<broker>.clientId=<clientId>
mqtt:<broker>.qos=<qos>
mqtt:<broker>.retain=<retain>
mqtt:<broker>.async=<async>
```

ตัวอย่าง ก.6 คำสั่งภายใน Transport Configuration ของไฟล์ openhab.cfg หลังจากตั้งค่าแล้ว

```
mqtt:iot.url=tcp://localhost:1883
mqtt:iot.clientId=arduinoGW
mqtt:iot.qos=0
mqtt:iot.retain=true
mqtt:iot.async=true
```

ก.1.3.1.2 การกำหนดไฟล์ .items

Syntax ของไฟล์ .items มีรูปแบบดังนี้

itemtype itemname ["labeltext"] [<iconname>] [(group1, group2, ...)] [{bindingconfig}]

ข้อสังเกต สิ่งที่อยู่ใน [ ] หมายถึง จะใส่ค่าหรือไม่ใส่ค่าก็ได้ เป็นเพียงตัวเลือก ซึ่งถ้าหากไม่ใส่ค่าเลย การประกาศ item ที่สั้นที่สุด จะเป็นลักษณะ itemtype itemname

1) itemtype คือ ชนิดของ item ของ openHAB ซึ่งชนิดของ item ที่จำเป็นและใช้บ่อย มีดังนี้

ตาราง ก.1 ชนิดของ itemtype ที่จำเป็นและใช้บ่อย

ชื่อของ item	คำอธิบาย	ชนิดของคำสั่ง
Dimmer	Item carrying a percentage value for dimmers	OnOff, IncreaseDecrease, Percent
Group	Item to nest other items / collect them in groups	-
Number	Stores values in number format	Decimal
Switch	Typically used for lights (on/off)	OnOff

2) itemname คือ ชื่อของ item ซึ่งสามารถตั้งเป็นชื่ออะไรก็ได้ แต่ต้องตั้งตามกฎการตั้งชื่อ คือ ต้องไม่มีช่องว่าง เป็นตัวอักษรภาษาอังกฤษโดยไม่มีอักขระพิเศษ ขึ้นต้นด้วยตัวอักษรภาษาอังกฤษ และต้องไม่ซ้ำกับชื่อของ item อื่นๆที่ประกาศไว้

3) labeltext คือ ข้อความที่ใช้แสดง item นั้นๆ เมื่อเรียกใช้ผ่าน Application สามารถตั้งเป็นข้อความใดก็ได้ตามที่ต้องการ (รองรับภาษาไทย)

4) iconname คือ ชื่อของ icon ที่ต้องการนำมาใช้ (ต้องตั้งตามชื่อของ icon ที่ระบบมีใช้เลือกใช้เท่านั้น)

5) group คือ ชื่อของ group ที่ item นั้นต้องการเข้าไปอยู่

6) bindingconfig คือ คำสั่งที่ใช้กำหนดว่า item นั้นจะรับค่าหรือส่งค่าไปยัง MQTT Broker สำหรับ bindingconfig ของ MQTT นั้น จะมี 2 รูปแบบคือ Inbound กับ Outbound โดย Inbound ใช้ในกรณีที่ item นี้ต้องการรับค่าจาก MQTT Broker และ Outbound ใช้ในกรณีที่ item นี้ต้องการส่งค่าไปยัง MQTT Broker

ตัวอย่างการตั้งค่าในไฟล์ .items

```
Group   All
Group   o_sensors       (All)
Group   o_lights        (All)
Group   o_dimmers       (All)

Group o_allsensors "เซ็นเซอร์" <sun> (o_sensors)
Group o_alllights "หลอดไฟ" <lights> (o_lights)

Number o_SS1 "เซ็นเซอร์ตัวที่ 1 [%.2f หน่วย]" <sun> (o_allsensors)
{mqtt="<[iot:/homes/sensors/SS1:state:default]"}
Number o_SS2 "เซ็นเซอร์ตัวที่ 2 [%.2f หน่วย]" <sun> (o_allsensors)
{mqtt="<[iot:/homes/sensors/SS2:state:default]"}

Dimmer o_DimmedLight "Dimmer ห้องนั่งเล่น [%d %%]" <slider>
{mqtt=">[iot:/homes/switches/dimmer:state:*:default]"}

Switch o_led1 "หลอดไฟห้องรับแขก" <switch> (o_alllights)
{mqtt=">[iot:/homes/switches/led1:command:ON:1],>[iot:/homes/switches/led1:command:OFF:0]"}
Switch o_led2 "หลอดไฟสนาม" <switch> (o_alllights)
{mqtt=">[iot:/homes/switches/led2:command:ON:1],>[iot:/homes/switches/led2:command:OFF:0]"}
```

รูป ก.3 ตัวอย่างการตั้งค่าในไฟล์ .items

อธิบายตัวอย่าง

จากตัวอย่าง จะมี item อยู่ 5 items แบ่งเป็น item ที่รับค่าจากเซ็นเซอร์มาแสดงผลเป็นตัวเลข 2 items มี item ชนิด Dimmer ซึ่งใช้ปรับค่าความสว่างของหลอดไฟฟ้า 1 item และ item ที่เป็นสวิทช์เปิดปิดหลอดไฟฟ้า 2 items

ก.1.3.1.3 การกำหนดไฟล์ .sitemap

Syntax ที่บรรทัดบนสุดของไฟล์ .sitemap จะต้องประกาศชื่อของ sitemap โดยใช้ element ชื่อ sitemap ซึ่งทุก sitemap จะต้องมี element นี้เป็นส่วนประกอบ ไม่เช่นนั้นจะไม่สามารถเข้าใช้งาน application ได้ ซึ่งสามารถกำหนดได้โดยใช้คำสั่ง

sitemap [sitemapname] [label="<title of the main screen>"]

ตาราง ก.2 element ที่ใช้งานบ่อยๆ

ชื่อ Element	อธิบาย
Chart	Adds a time-series chart object for displaying logged data.
Frame	Area with either various other sitemap elements or further nested frames
Group	Renders all elements of a given group defined in an items definiton file
Slider	Renders a slider
Switch	Renders a switch item
Text	Renders a text element

การใช้งาน element Frame

Frame [label="<labelname>"] [icon="<icon>"] [item=<item>]

{

[additional sitemap elements]

}

ตัวอย่าง ก.7 ตัวอย่างการใช้คำสั่งของ element Frame

```
Frame label="เซ็นเซอร์"{
          Group item=sensors label="เซ็นเซอร์"
icon="temperature"
}
```

การใช้งาน element Slider

Slider item=<itemname> [label="<labelname>"] [icon="<iconname>"]

[sendFrequency="frequency"] [switchSupport]

ตัวอย่าง ก.8 ตัวอย่างการใช้คำสั่งของ element Slider

```
Slider item=dimmers label="ดิมเมอร์"
icon="slider"
```

การใช้งาน element Text

Text item=<itemname> [label="<labelname>"] [icon="<iconname>"]

ตัวอย่าง ก.9 ตัวอย่างการใช้คำสั่งของ element Text

```
Text item=status label="ประตู" icon="door"
```

การใช้งาน element Group

Group [item=<itemname>] [label="<labelname>"] [icon="<iconname>"]

ตัวอย่าง ก.10 ตัวอย่างการใช้คำสั่งของ element Group

```
Group item=switched label="สวิทช์" icon="switch"
```

การใช้งาน element Switch

Switch item=<itemname> [label="<labelname>"] [icon="<iconname>"]

[mappings="<mapping definition>"]

ตัวอย่าง ก.11 ตัวอย่างการใช้คำสั่งของ element Switch

```
Switch item=switch label="สวิทช์" icon="switch"
```

ตัวอย่างการตั้งค่าในไฟล์ .sitemap

```
sitemap o label="Smart Home"
{
    Frame label="เซ็นเซอร์"{
        Group item=o_allsensors label="เซ็นเซอร์" icon="temperature"
    }

    Frame label="หลอดไฟ"{
        Group item=o_alllights label="ชั้นล่าง" icon="switch"
    }
    Frame label="หลอดไฟแบบปรับความสว่างได้"{
        Slider item=o_DimmedLight switchSupport
    }
}
```

รูป ก.4 ตัวอย่างการกำหนดไฟล์ .sitemap

อธิบายตัวอย่าง

เป็น sitemap ชื่อ o และมีข้อความ Smart Home อยู่หน้าแรก โดยที่หน้าแรกจะมีหมวดหมู่ของ items ใหญ่ๆ อยู่ 3 หมวด ซึ่งถูกแยกตาม frame ที่กำหนด ซึ่งได้แก่ กลุ่มเซ็นเซอร์ กลุ่มของสวิทช์หลอดไฟ และกลุ่มของ Dimmer ปรับความสว่างของหลอดไฟฟ้า ซึ่งเมื่อผู้ใช้ทำการเลือกหมวดของ item แต่ละหมวดแล้วก็จะเจอกับ item ย่อยๆด้านใน

ก.1.3.1.4 การกำหนดไฟล์ .rules

สำหรับโครงสร้างของไฟล์ .rules นั้น จะมีส่วนประกอบด้วย 3 ส่วน คือ

1) ส่วน Imports

2) ส่วน Variable Declarations

3) ส่วน Rules

โดยในส่วนของ Imports และ Variable Declarations นั้นจะใช้ในบางกรณี แต่ในส่วนของ Rules นั้นจะต้องมีทุกครั้ง โดยการกำหนดกฎต่างๆจะมีรูปแบบคำสั่งดังนี้

```
rule "rule name"
  when
    <TRIGGER CONDITION1> or
    <TRIGGER_CONDITION2> or
    <TRIGGER_CONDITION3>
    ...
  then
    <EXECUTION_BLOCK>
  end
```

ตัวอย่างการตั้งค่าในไฟล์ .rules

```
import org.openhab.core.library.types.*
import org.openhab.core.persistence.*
import org.openhab.model.script.actions.*

var Number lastCheck = 0

rule "autosw"

when
    Item tempSS received update
then
    if(tempSS.state>30){
        if(lastCheck == 1){}
        else{
            sendCommand(autosw,ON)
            lastCheck = 1;
        }
    }
    else {
        sendCommand(autosw,OFF)
        lastCheck = 0
    }
end
```

รูป ก.5 ตัวอย่างการตั้งค่าในไฟล์ .rules

อธิบายตัวอย่าง

เมื่อ openHAB รับค่ามาใส่ items tempSS ก็จะนำค่าของ tempSS ไปตรวจสอบ โดยถ้ามีค่ามากกว่า 30 ก็จะสั่งให้เปิดสวิชท์โดยอัตโนมัติ (กำหนดให้ item autosw เป็น ON)

ก.1.3.2 ซอฟท์แวร์ mosquitto

เป็นซอฟท์แวร์ประเภท MQTT Broker สำหรับการเรียกใช้งานทำได้โดย

mosquitto_pub ใช้สำหรับ publish ค่า ไปยัง topic ตามที่ต้องการ

mosquittto_sub ใช้สำหรับ subscribe topic ตามที่ต้องการ

ตัวอย่าง ก.12 การใช้งาน mosquitto_pub

```
mosquitto_pub -d -t /homes/sensors/lightSS -m 15
```

ตัวอย่าง ก.13 การใช้งาน mosquitto_sub

```
mosquitto_sub -d -t /homes/sensors/lightSS
```

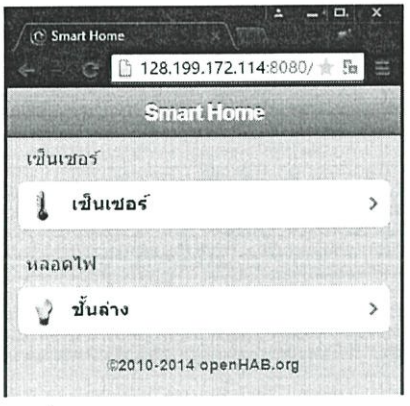
ก.2 การใช้งานระบบ

ก.2.1 การใช้งานเว็บแอปพลิเคชั่นผ่านเว็บเบราว์เซอร์

การใช้งานผ่านเว็บเบราว์เซอร์ทำให้ระบบปฏิบัติการอื่นๆที่ไม่ใช่ iOS และ Android สามารถใช้งานระบบได้ ซึ่งการแสดงผลในเบราวเซอร์แต่ละตัวอาจจะแตกต่างกันเล็กน้อย สำหรับการใช้งานมีขั้นตอนดังนี้

1. เปิดเว็บเบราว์เซอร์แล้วเข้าไปที่ http://__server_ip__:8080/openhab.app?sitemap=stitemap_name

2. ทำการติดตาม ควบคุมหรือสั่งการระบบตามที่ต้องการ



รูป ก.6 ผลการทดสอบ openHAB

ก.2.2 การใช้งานผ่านแอปพลิเคชั่นบนระบบปฏิบัติการ iOS

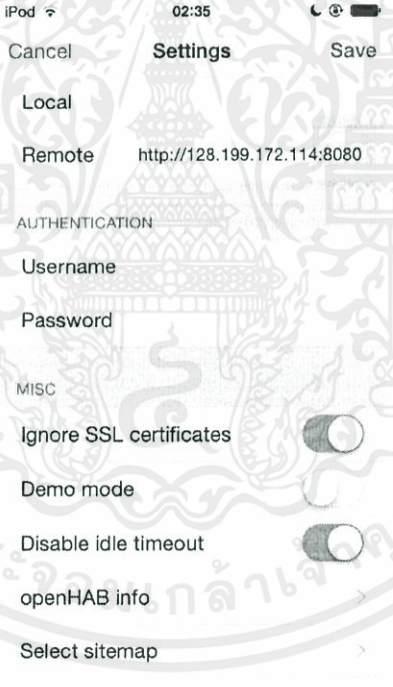
1) เข้า App store แล้วค้าหาแอปพลิเคชั่น openHAB จากนั้นดาวน์โหลดมาติดตั้ง

รูป ก.7 ค้นหาแอปพลิเคชัน openHAB ใน App store

2) เปิดแอปพลิเคชั่นแล้วทำจะเจอกับโหมด Demo ของแอปพลิเคชัน ให้เลือก Settings เพื่อปรับแต่งแอปพลิเคชันให้สามารถใช้งานกับระบบได้

รูป ก.8 หน้าจอแอปพลิเคชัน openHAB เมื่อเปิดใช้งานครั้งแรก

3) ทำการเปลี่ยนให้ Remote ตรงกับเซิร์ฟเวอร์ของผู้ใช้ ที่เหลือให้ปรับแต่งแอปพลิเคชันให้เหมือนรูปด้านล่าง จากนั้นให้เลือก sitemap ที่ผู้ใช้ต้องการนำมาใช้งาน



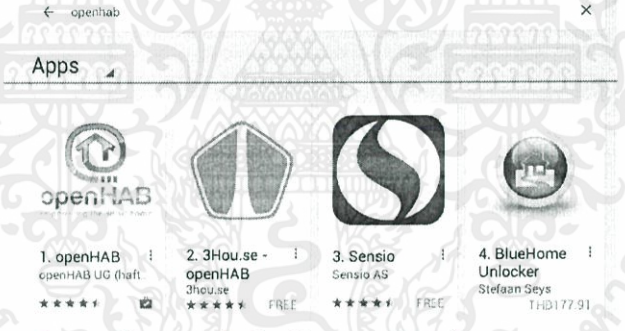
รูป ก.9 ทำการตั้งค่าแอปพลิเคชัน

4) ทำการติดตาม ควบคุมหรือสั่งการระบบตามที่ต้องการ

รูป ก.10 หน้าจอของแอปพลิเคชัน

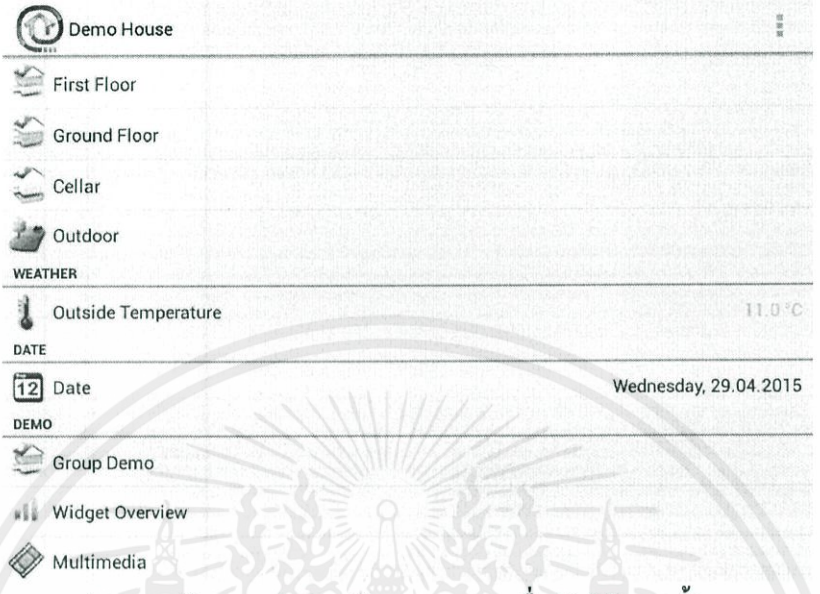
ก.2.3 การใช้งานผ่านแอปพลิเคชั่นบนระบบปฏิบัติการ Android

1) เข้า Play store แล้วค้าหาแอปพลิเคชั่น openHAB จากนั้นดาวน์โหลดมาติดตั้ง



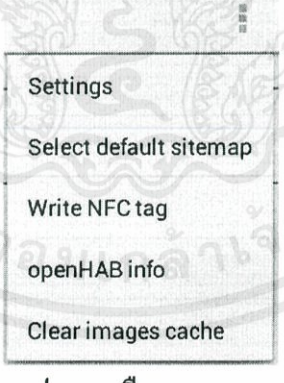
รูป ก.11 ค้นหาแอปพลิเคชัน openHAB ใน App store

2) เปิดแอปพลิเคชั่นจะพบกับโหมด Demo ของแอปพลิเคชัน



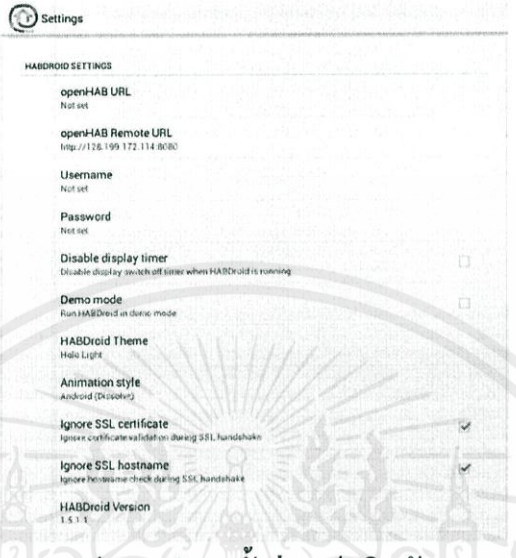
รูป ก.12 หน้าจอแอปพลิเคชัน openHAB เมื่อเปิดใช้งานครั้งแรก

3) แล้วทำการปรับแต่งค่าก่อนใช้งาน โดยเลือก จุดสี่เหลี่ยมสามจุดด้านบนขวาของแอป-พลิเคชัน แล้วเลือก Settings



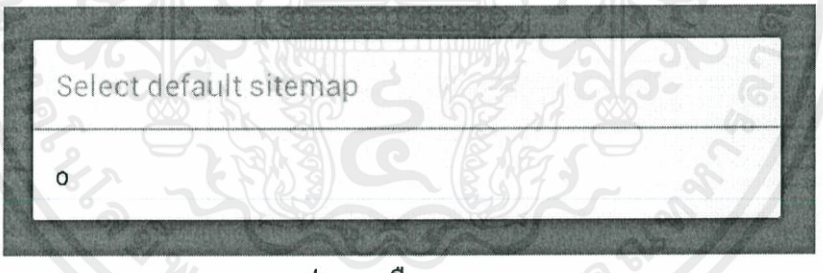
รูป ก.13 เลือก Settings

4) ทำการเปลี่ยนให้ Remote ตรงกับเซิร์ฟเวอร์ของผู้ใช้ ที่เหลือให้ปรับแต่งแอปพลิเคชันให้เหมือนรูปด้านล่าง จากนั้นให้เลือก sitemap ที่ผู้ใช้ต้องการนำมาใช้งาน



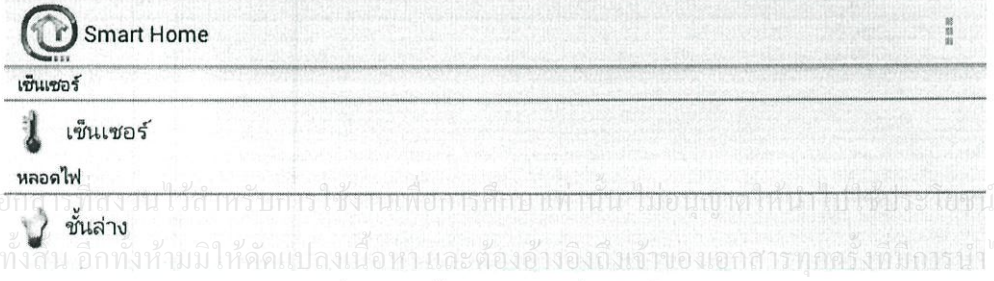
รูป ก.14 ทำการตั้งค่าแอปพลิเคชัน

5) เลือก sitemap ที่ต้องการ



รูป ก.15 เลือก sitemap

6) ทำการติดตาม ควบคุมหรือสั่งการระบบตามที่ต้องการ



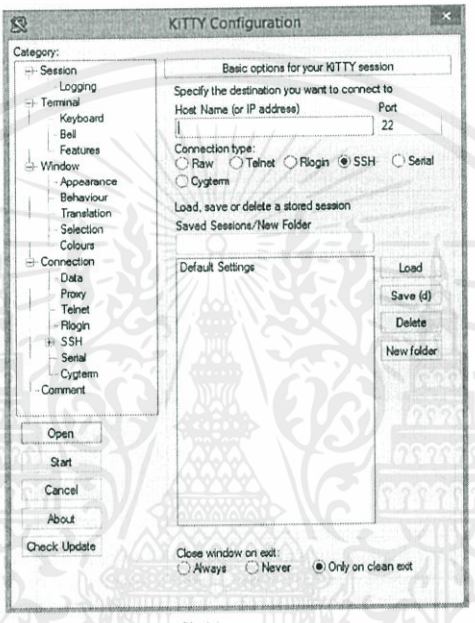
รูป ก.16 หน้าจอของแอปพลิเคชัน

ภาคผนวก ข

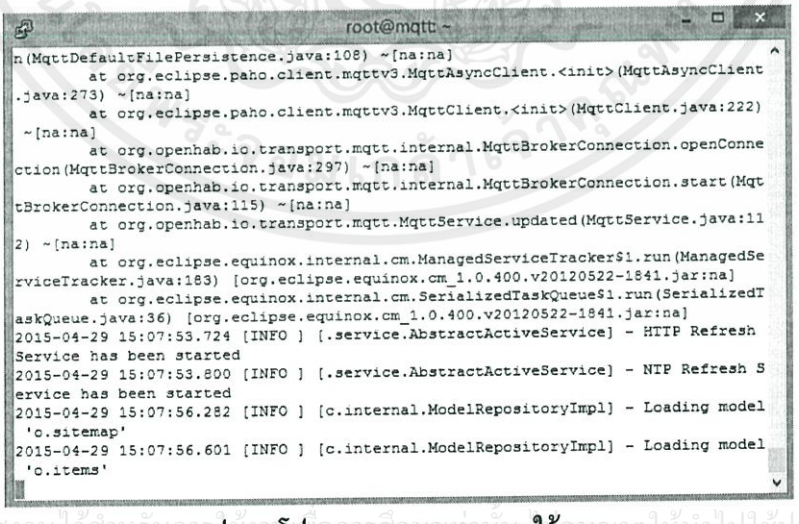
โปรแกรมคอมพิวเตอร์ที่จำเป็นต้องใช้ในการใช้งานระบบ

ข.1 KiTTY หรือ PuTTY

ใช้สำหรับเชื่อมต่อกับเซิร์ฟเวอร์เพื่อรันคำสั่งต่างๆ ที่เกี่ยวข้อง เช่น ทดสอบการ publish และการ subscribe ของ mosquitto



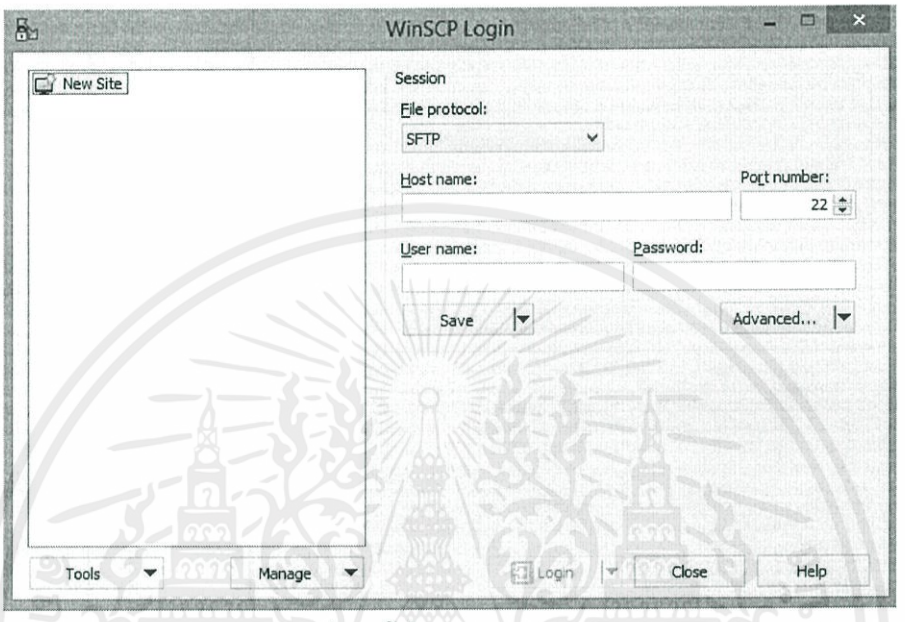
รูป ข.1 โปรแกรม KiTTY



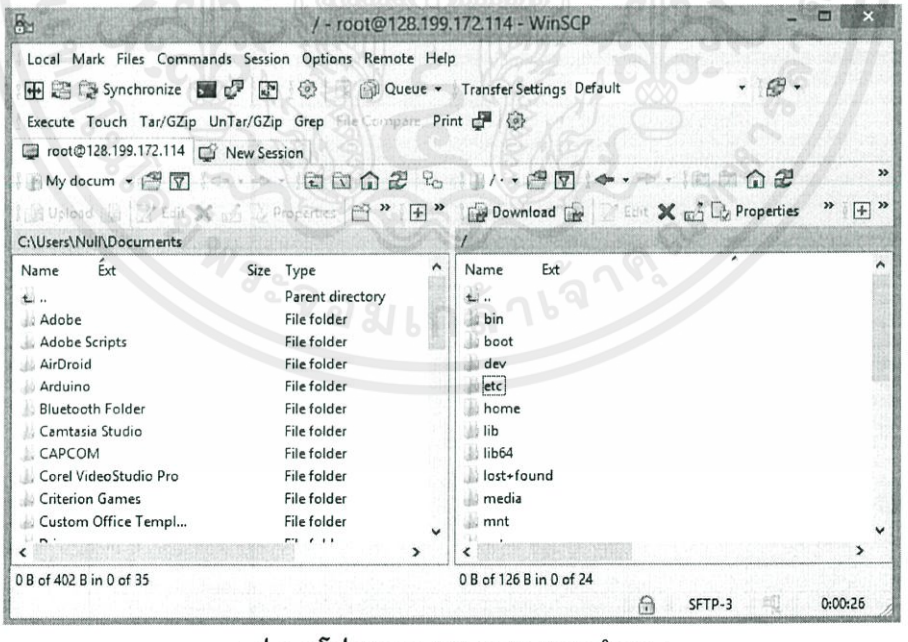
รูป ข.2 โปรแกรม KiTTY ขณะใช้งาน

ข.2 WinSCP

ใช้สำหรับเชื่อมต่อกับเซิร์ฟเวอร์เพื่อบริหารจัดการไฟล์ต่างๆ ของ openHAB เช่น การแก้ไขไฟล์ configuration



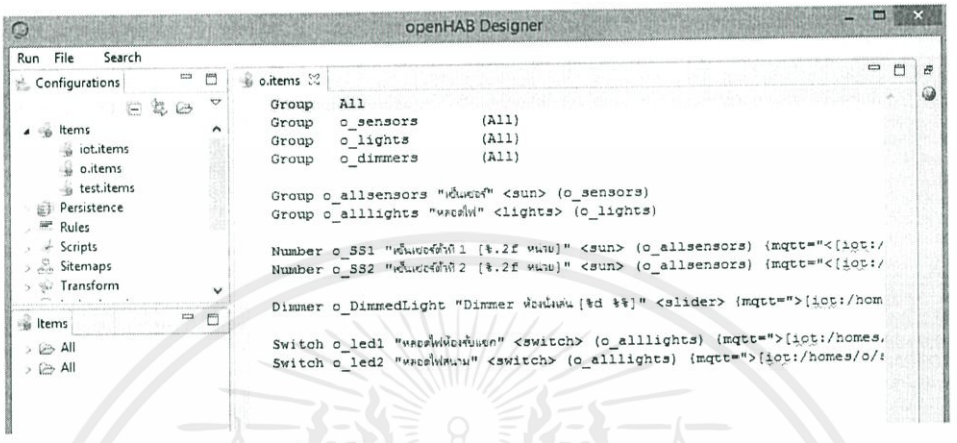
รูป ข.3 โปรแกรม WinSCP



รูป ข.4 โปรแกรม WinSCP ขณะทำงาน

ข.3 openHAB Designer

ใช้สำหรับแก้ไขไฟล์ที่เกี่ยวข้องกับ openHAB เช่น ไฟล์ .items, ไฟล์ .sitemaps, ไฟล์ .rules และไฟล์ configuration ต่างๆ

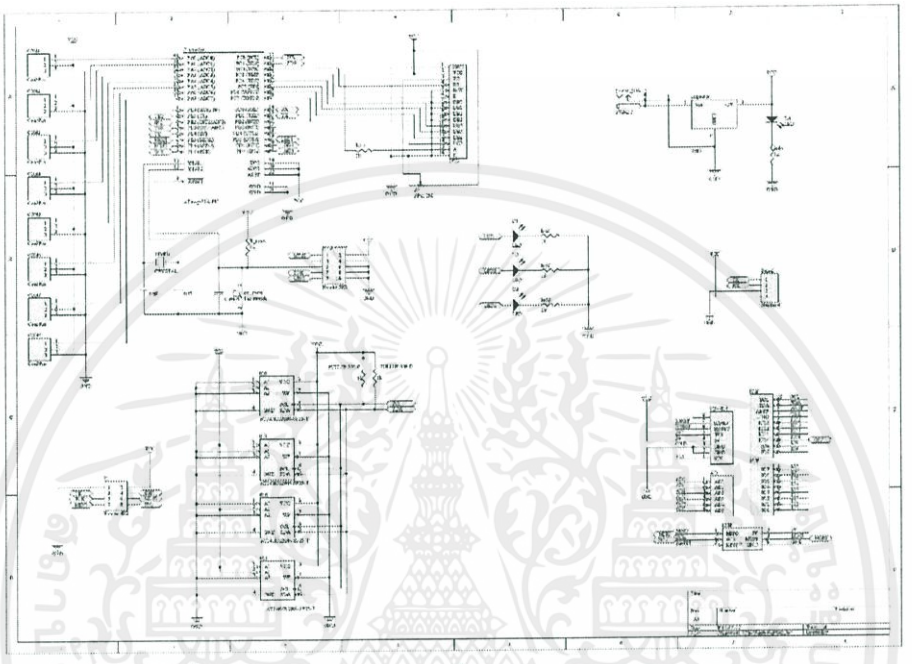


รูป ข.5 โปรแกรม openHAB Designer

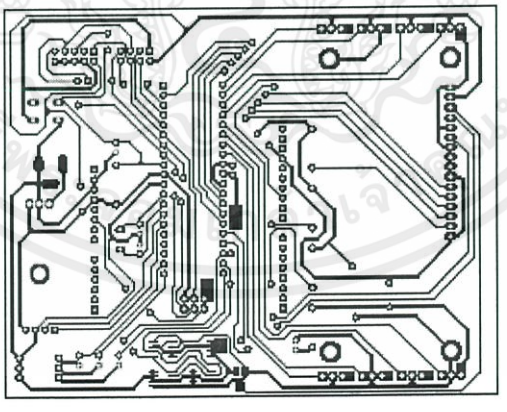
ภาคผนวก ค

Schematic และลายวงจรของบอร์ด Network I/O

ค.1 Network I/O Gateway

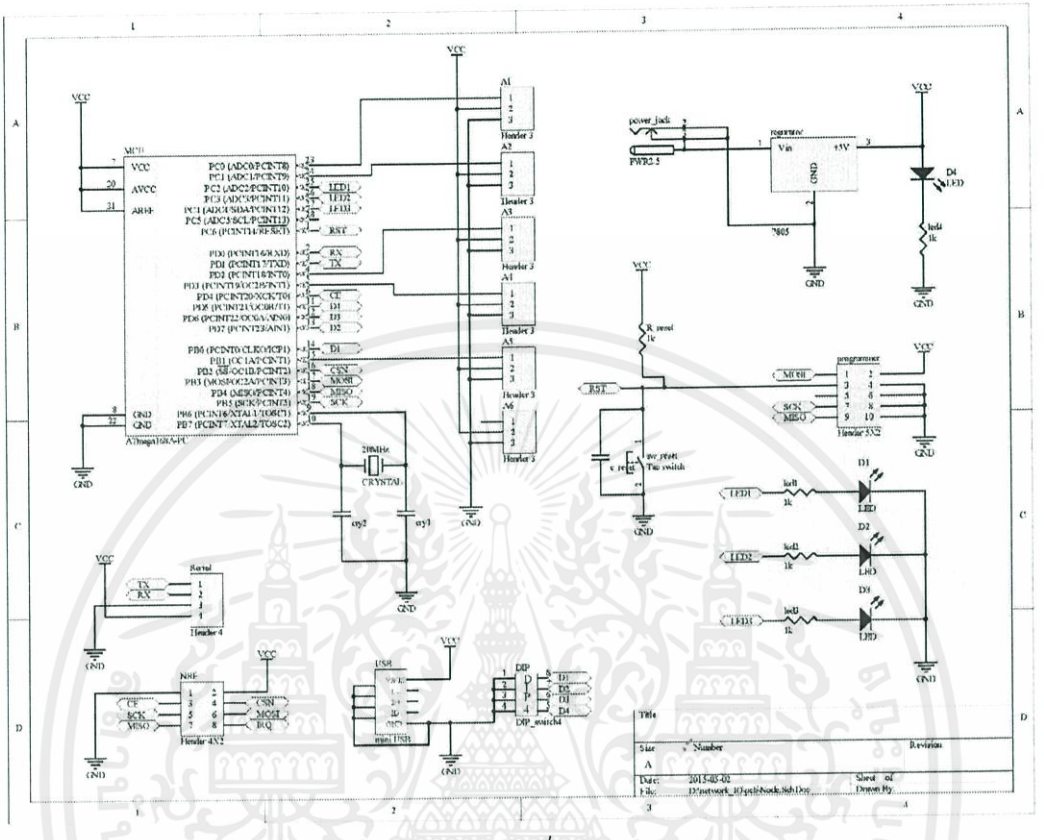


รูป ค.1 Schematic ของบอร์ด Network I/O Gateway

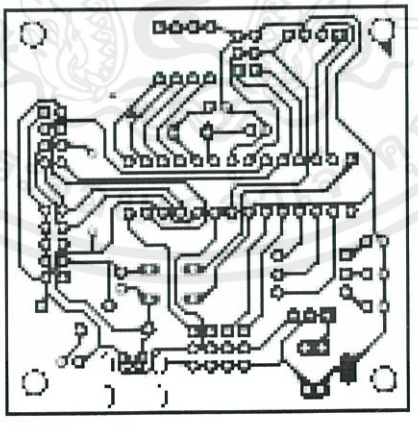


รูป ค.2 ลายวงจรของ Network I/O Gateway

ค.2 Network I/O Node



รูป ค.3 Schematic ของบอร์ด Network I/O Node



รูป ค.4 ลายวงจรของ Network I/O Node