ระบบควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบไร้สาย

WIRELESS HOME APPLIANCE CONTROL SYSTEM

กัญญารัตน์ ก้านทอง
ณัฏฐ์ชยาน์ ศุภศิลป์
ศรัยสุดา กระแสสินธุ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมระบบควบคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2557

ระบบควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบไร้สาย

WIRELESS HOME APPLIANCE CONTROL SYSTEM

กัญญารัตน์ ก้านทอง

ณัฏฐ์ธยาน์ ศุภศิลป์

สร้อยสุดา กระแสสินธุ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

WIRELESS HOME APPLIANCE CONTROL SYSTEM

KANYARAT KARNTHONG
NATTHAYA SUPASIL
SROYSUDA KRASAESIN

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN CONTROL ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2014

ปริญญานิพนธ์ปีการศึกษา 2557

ภาควิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบไร้สาย

WIRELESS HOME APPLIANCE CONTROL SYSTEM

ผู้จัดทำ นางสาวกัญญารัตน์ ก้านทอง 54010083

นางสาวณัฏฐ์ธยาน์ ศุภศิลป์ 54011291

นางสาวสร้อยสุดา กระแสสินธุ์ 54011337

.....อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร.วรพงศ์ ตั้งศรีรัตน์)

ระบบควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบไร้สาย

โดย

นางสาวกัญญารัตน์ ก้านทอง 54010083

นางสาวณัฏฐ์ธยาน์ ศุภศิลป์ 54011291

นางสาวสร้อยสุดา กระแสสินธุ์ 54011337

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร.วรพงศ์ ตั้งศรีรัตน์

ปีการศึกษา 2557

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอระบบควบคุมอุปกรณ์ไฟฟ้าภายในบ้านแบบไร้สายนี้ ได้นำเสนอการใช้เทคโนโลยีที่มีอยู่ในปัจจุบันมาประยุกต์ใช้กับอุปกรณ์ไฟฟ้าภายในบ้าน โดยมีวัตถุประสงค์เพื่อให้ผู้ใช้งานได้รับความสะดวกสบายมากขึ้น ด้วยการใช้แอพพลิเคชั่นแอนดรอยด์ (Android) และระบบอินเตอร์เน็ต (Internet) ที่ควบคุมด้วยไมโครคอนโทรลเลอร์ AVR ATmega2560 ที่ป้อนข้อมูลจากสมาร์ทโฟน (Smart Phone), แท็ปเล็ต (Tablet) หรือคอมพิวเตอร์ เพื่อส่งเข้าสู่ไมโครคอนโทรลเลอร์ที่ได้โปรแกรมไว้ ให้ทำการควบคุมอุปกรณ์ไฟฟ้า โดยมีการแสดงสถานะการทำงานของอุปกรณ์ไฟฟ้า และจะส่งข้อมูลชุดคำสั่ง (Code) ผ่านตัวส่งไปยังตัวรับโดยใช้อุปกรณ์ XBee เข้าสู่ไมโครคอนโทรลเลอร์ AVR ATmega2560 อีกตัว เพื่อสั่งการให้อุปกรณ์ไฟฟ้าภายในบ้านเปิด หรือปิดการทำงาน

WIRELESS HOME APPLIANCE CONTROL SYSTEM

By

Ms. Kanyarat Karnthong 54010084

Ms. Natthaya Supasil 54011291

Ms. Sroysuda Krasaesin 54011337

Advisor

Assoc. Prof. Dr. Worapong Tangsrirat

Academic Year 2014

ABSTRACT

The concept of the home appliance control system via wireless communication is presented. The purpose of this project is to provide the controlling of home appliances by using android application and internet to control the microcontroller AVR ATmega2560 that put data from smart phone, tablet or computer and send to the microcontroller which is programmed for controlling the electrical equipments. The command code will be sent from one transmitter to one receiver by using XBee and the microcontroller AVR ATmega2560 will also order the devices in the residence on/off.

กิตติกรรมประกาศ

การที่ปริญญานิพนธ์ และโครงงานสามารถสำเร็จลุล่วงไปได้ด้วยดีนั้น ทางผู้จัดทำขอกราบขอบพระคุณ รองศาสตราจารย์ ดร.วรพงศ์ ตั้งศรีรัตน์ อาจารย์ที่ปรึกษา ที่ให้คำแนะนำที่ดีมาโดยตลอด ขอขอบพระคุณคณาจารย์ในภาควิชาวิศวกรรมการวัดและควบคุม และภาควิชาวิศวกรรมเทคโนโลยีสารสนเทศทุกท่าน ที่ประสิทธิประสาทความรู้ให้กับคณะผู้จัดทำโครงงานนี้ เพื่อที่จะให้โครงงานนี้สามารถบรรลุ และสำเร็จไปได้ด้วยดี

ขอขอบคุณ คุณจิระวัฒน์ สุขสวัสดิ์ พี่บัณฑิตจากคณะวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมการวัดคุม มหาวิทยาลัยพระจอมเกล้าพระนครเหนือ สำหรับคำแนะนำ และคำปรึกษาที่ดีมาโดยตลอด

ขอขอบคุณพี่ปริญญาโททุกคนที่คอยให้คำปรึกษา และต้องขอบคุณเพื่อนๆ ที่ร่วมช่วยเหลือกัน และให้คำแนะนำในการปฏิบัติโครงงานในด้านต่างๆ

สุดท้ายนี้ ผู้จัดทำโครงงานต้องขอกราบขอบพระคุณบิดา มารดา ของพวกเรา ผู้ที่มีพระคุณสูงสุด ที่เป็นผู้ให้โอกาสในการศึกษา ตลอดจนเป็นแรงบันดาลใจที่ดีที่สุดที่ทำให้โครงงานนี้สำเร็จสมบูรณ์ลงได้

ผู้จัดทำ
นางสาวกัญญารัตน์ ก้านทอง
นางสาวณัฏฐ์ธยาน์ ศุภศิลป์
นางสาวสร้อยสุดา กระแสสินธุ์

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

ในโลกของการติดต่อสื่อสารในปัจจุบันมีการพัฒนาที่ก้าวหน้าเป็นอย่างมาก โดยเฉพาะการสื่อสารแบบไร้สายได้เข้ามามีบทบาทในชีวิตประจำวันมากยิ่งขึ้น สร้างความสะดวกสบายในการดำรงชีวิตของมนุษย์ ซึ่งเครื่องมืออุปกรณ์ต่างๆ ก็นิยมนำมาทำเป็นแบบไร้สายเพื่อความสะดวกสบายและไม่ต้องเสียเวลาเดินสายจากระยะไกล แม้กระทั่งการวางระบบบางระบบภายในบ้าน และภายในโรงงานก็ทำเป็นระบบไร้สาย ด้วยเหตุนี้จึงได้พัฒนาระบบควบคุมอุปกรณ์ไฟฟ้าแบบไร้สายขึ้นมา โดยใช้โมดูล XBee ในการควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้า

ปัจจุบันสมาร์ทโฟน เป็นอุปกรณ์พกพาที่ได้รับความนิยมจากผู้ใช้งานเป็นจำนวนมาก โดยเฉพาะแอนดรอยด์ (Android) ซึ่งเป็นระบบปฏิบัติการที่เปิดโอกาสให้บุคคลอื่นสามารถนำเอาระบบไปพัฒนาต่อได้ (Open Source) เนื่องจากอุปกรณ์ที่เป็นระบบปฏิบัติการแอนดรอยด์นี้มีจำนวนมาก และได้รับความนิยมเป็นอย่างสูง ส่งผลให้แนวทางในการพัฒนาแอพพลิเคชั่น (Application) เพื่อนำไปสั่งงานมีความหลากหลายมากยิ่งขึ้น

1.1 ความเป็นมาของโครงงาน

ปัจจุบันเทคโนโลยีได้เข้ามามีบทบาทในชีวิตประจำวันมากยิ่งขึ้น ความก้าวหน้าของเทคโนโลยีใหม่ๆ มากมายที่เป็นส่วนทำให้เกิดอุปกรณ์เครื่องมือต่างๆ จึงก่อให้เกิดความสะดวกสบายและสนองความต้องการของมนุษย์ได้เป็นอย่างดี ซึ่งเทคโนโลยีแบบไร้สาย (Wireless Telecommunication) นั้น มีมากมายหลายแบบ และเป็นเทคโนโลยีที่สำคัญในปัจจุบันนี้เลยก็ว่าได้ จึงเป็นอีกหนึ่งเหตุผลที่อยากจะศึกษา และจัดทำโครงงานเกี่ยวกับเรื่องนี้ขึ้นมา โดยได้ทำการศึกษาเกี่ยวกับการนำสัญญาณวิทยุ (Radio Frequency) มาใช้ในการสื่อสารเพื่อที่จะควบคุมอุปกรณ์ไฟฟ้าในระยะไกลโดยไม่ต้องใช้สาย ผ่านโมดูลXBee มาเป็นตัวรับ และตัวส่งข้อมูล และใช้ไมโครคอนโทรลเลอร์ในการควบคุม และสั่งงานจากระยะไกล อีกทั้งตัวโมดูล XBee มีการใช้พลังงานน้อย ใช้ได้นาน และไม่ซับซ้อนมาก ถ้าเทียบกับเทคโนโลยีแบบอื่นๆ ซึ่งถือว่าเป็นจุดเด่น และข้อได้เปรียบของ XBee

ระบบปฏิบัติการแอนดรอยด์ เป็นที่ได้รับความนิยมเป็นอย่างสูงในปัจจุบัน ทั้งในสมาร์ทโฟน, แท็ปเล็ต, นาฬิกา, แว่นตา และอุปกรณ์สื่อสารอื่นๆ อีกมากมาย จากเหตุผลดังกล่าว ทำให้มีความสนใจในการพัฒนาแอพพลิเคชั่น เพื่อนำไปสั่งงานบนอุปกรณ์ไฟฟ้าภายในบ้าน ซึ่งจะทำให้ง่าย และ

เกิดความสะดวกต่อการใช้งาน รวมไปถึงการป้องกันความปลอดภัย ในกรณีที่ไม่ทราบสถานะการทำงานของเครื่องใช้ไฟฟ้า

1.2 วัตถุประสงค์ของโครงงาน

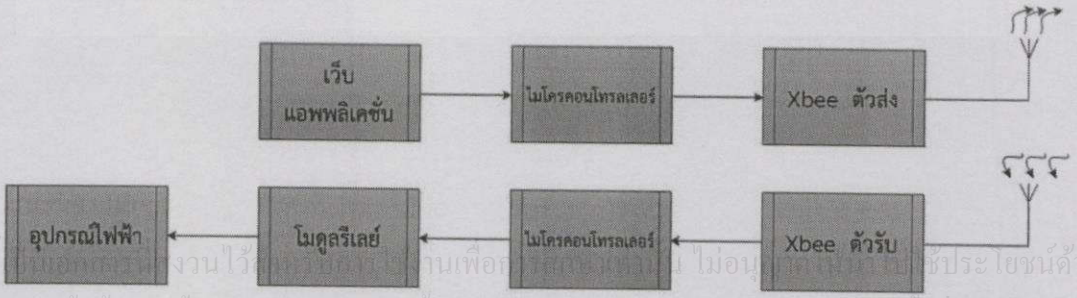
1. เพื่อศึกษาการรับส่งข้อมูลผ่านแอพพลิเคชั่น และความถี่วิทยุ
2. เพื่อศึกษาการควบคุมอุปกรณ์ไฟฟ้าแบบไร้สาย ให้สามารถเปิด-ปิดได้
3. เพื่อเรียนรู้การทำงานที่เป็นระบบ และรู้จักวิธีการแก้ไขปัญหา
4. เพื่อเรียนรู้หลักในการออกแบบระบบไร้สายให้สามารถใช้งานได้ง่าย และใช้งานได้จริง
5. เพื่อนำความรู้ที่ได้จากการศึกษาการทำงานของระบบปฏิบัติการแอนดรอยด์ นำมาใช้ในการออกแบบแอพพลิเคชั่นให้เหมาะสมกับการใช้งาน

1.3 ขอบเขตการศึกษา

1. ศึกษาเทคโนโลยีการรับ-ส่งสัญญาณวิทยุของโมดูล XBee
2. ศึกษาการควบคุมการทำงานของอุปกรณ์ไฟฟ้า โดยการใช้ไมโครคอนโทรลเลอร์ตระกูล AVR เป็นอุปกรณ์ควบคุม
3. ศึกษาการใช้เครื่องควบคุมระยะไกลผ่านแอพพลิเคชั่น และการออกแบบโครงสร้าง
4. ศึกษาการทำงานของระบบปฏิบัติการแอนดรอยด์ เพื่อทำการออกแบบแอพพลิเคชั่น

1.4 ผลที่คาดว่าจะได้รับ

1. รู้จักการประยุกต์ใช้งานระบบไมโครคอนโทรลเลอร์เพื่อให้เกิดประโยชน์สูงสุด
2. มีความเข้าใจในการเขียนโปรแกรมการสั่งงานไมโครคอนโทรลเลอร์เพื่อใช้ในงานควบคุม
3. ช่วยเพิ่มขีดความสามารถในการส่งสัญญาณ เพื่อทำการควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้าแบบไร้สาย เช่น คนพิการ ผู้สูงอายุ หรือผู้ที่ไม่สามารถช่วยเหลือตัวเองได้ ตลอดจนในกรณีที่ไม่อยู่บ้านก็สามารถสั่งงานการเปิด-ปิดอุปกรณ์ไฟฟ้าผ่านแอพพลิเคชั่นได้ โดยได้มีการออกแบบให้แสดงสถานะของอุปกรณ์ไฟฟ้า ดังรูปที่ 1.1



รูปที่ 1.1 บล็อกไดอะแกรมแสดงการทำงาน

บทที่ 2

ทฤษฎี และหลักการ

2.1 XBee

อุปกรณ์ XBee เป็นเทคโนโลยีไร้สายที่ร่วมกันสื่อสารข้อมูลผ่านเซนเซอร์ขนาดเล็ก จำนวนเป็นพันๆ หมื่นๆ ชิ้น ที่ฝังอยู่ตามส่วนต่างๆ ในอาคาร สำนักงาน โรงงาน หรือแม้แต่ภายในบ้าน การทำงานจะเป็นการรับ-ส่งคลื่นสัญญาณข้อมูล ผ่านชิพขนาดเล็กๆ จุดต่อจุดไปเรื่อยๆ จนถึงปลายทางที่ต้องการดาวน์โหลดข้อมูลลงในเครื่องคอมพิวเตอร์เพื่อใช้ในการวิเคราะห์ข้อมูล ข้อมูลที่ได้อาจจะเป็นการวัดอุณหภูมิ การเคลื่อนไหวของสิ่งมีชีวิต จับปริมาณมลพิษในอากาศ ปริมาณน้ำ ท่อแก๊สโดยใช้พลังงานแสงอาทิตย์ หรือแบตเตอรี่ขนาดเล็กที่กินไฟน้อยมาก จึงสามารถฝังทิ้งไว้ในที่ห่างไกลได้เป็น 10 ปี เทคโนโลยี XBee นี้จะช่วยทำให้บริษัทที่เกี่ยวข้องกับการส่งพลังงาน เช่น น้ำมัน ประปา น้ำในเขื่อน ท่อแก๊ส สามารถประหยัดการสูญเสียได้อย่างน้อย 10-15% และในอนาคตอันใกล้นี้เมื่อเทคโนโลยีนาโนก้าวหน้ามากขึ้น XBee จะมีขนาดเล็กเท่าหัวเข็มหมุด สามารถฝังได้แม้กับในร่างกายของสิ่งมีชีวิต

ชื่อ XBee ได้มาจากพฤติกรรมการสื่อสารกันของผึ้ง โดยผึ้งจะมีการบินแบบซิกแซ็ก และจะให้ข้อมูลข่าวสารระหว่างผึ้งด้วยกัน ที่เกี่ยวกับตำแหน่ง ระยะทาง และทิศทางของอาหารที่มันหาอยู่

XBee ถูกสร้างขึ้นในการทำระบบเครือข่ายไร้สายส่วนบุคคล (WPAN) อยู่ภายใต้มาตรฐาน IEEE 802.15.4 โดยมาตรฐานนี้ใช้งานสำหรับการสื่อสารความเร็วต่ำ ใช้กำลังไฟฟ้าน้อย และมีคุณสมบัติการจัดการตัวเองได้

ลักษณะของ XBee คือมีทางเข้าช่องสัญญาณโดยการใช้ CSMA –CA[1] เพื่อหลีกเลี่ยงการชนกัน ระยะทางโดยทั่วไปประมาณ 50 เมตร มีโทโปโลยี (Topology)[1] แบบสตาร์ (Star), จุดต่อจุด (Peer-to-peer), เมช (Mesh) ทั้งนี้แต่ละอุปกรณ์จะมีแอดเดรส ที่มีความยาว 64 บิต หรือความยาว 16 บิต (รองรับได้ 64,000 อุปกรณ์)

คุณสมบัติทั่วไปของ XBee

1. ทำงานในย่านความถี่ 2.4 GHz.
2. มีสายอากาศแบบวิป (Whip)[2]
3. มีกำลังส่ง 60 mW (18 dBm)
4. ระยะทำการในร่ม ประมาณ 100 เมตร และมีระยะทำการกลางแจ้ง (Line of Sight) ประมาณ 1,500 เมตร

---

[1] โทโปโลยี (Topology) การเชื่อมโยง สายสื่อสารเข้ากับอุปกรณ์อิเล็คทรอนิกต่างๆ ภายในเครือข่ายเข้าด้วยกัน

[2] วิป (Whip) หมายถึง ส่วนแผ่คลื่นของสายอากาศรถยนต์ที่เป็นเส้นตรงเรียว โค้งงอได้คล้ายแส้ม้า

5. ความไวในการรับส่งสัญญาณ -100 dBm (1% Packet Error Rate)

6. การทำงานของขาพอร์ต สามารถกำหนดผ่านทางซอฟต์แวร์เพื่อให้ทำงานเป็นอินพุตอนาล็อก ใช้ในการแปลงสัญญาณอนาล็อกเป็นดิจิตอล และอินพุต/เอาต์พุตดิจิตอล

7. มีขนาด 0.96" *1.297" หรือ 2.438 cm. *3.249 cm.

8. ไฟเลี้ยง 2.8 - 3.4 V

9. โหมดการลดพลังงานไฟเลี้ยง 3.3 V

10. ใช้งานที่อุณหภูมิ -40 ถึง 80 องศาเซลเซียส

11. การจัดขา และฟังก์ชันการทำงานแสดง

2.1.1 การส่งข้อมูลของ XBee

การส่งข้อมูลแบบสัญญาณความถี่วิทยุ ของแต่ละแพ็กเกจในส่วนบน (Header) นั้น จะประกอบไปด้วย แอดเดรสต้นทาง (Source Address) และแอดเดรสปลายทาง (Destination Address) โดยที่ IEEE 802.15.4 จะมีโครงสร้าง 2 แบบ คือ แบบสั้น แอดเดรส 16-บิต และแบบยาว แอดเดรส 64-บิต ซึ่ง 64-บิต จะสามารถอ่านคำสั่ง SL (Serial Number Low) และ SH (Serial Number High) และการส่งข้อมูลแบบสัญญาณวิทยุ จะส่งได้ 2 โหมด คือ โหมด Unicast และโหมด Broadcast [3]

- การส่งแพ็กเกจโดยใช้โครงสร้างแอดเดรส 16-บิต ให้ตั้งค่าแอดเดรสปลายทางทางต่ำ (DL) ให้เท่ากับตัวแปร MY และตั้งค่าตัวแปรแอดเดรสปลายทางทางสูง (DH) เป็น '0'

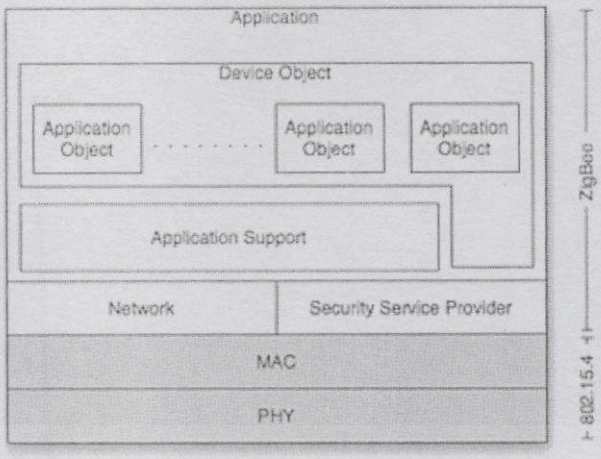
- การส่งแพ็กเกจโดยใช้โครงสร้างแอดเดรส 64-บิต ให้ตั้งค่าแอดเดรสปลายทาง (DL+DH) ให้เข้ากับแอดเดรสต้นทาง (SL+SH) ของปลายทางที่จะส่งแพ็คเกจไป

2.1.2 มาตรฐาน IEEE 802.15.4

มาตรฐานของระบบ IEEE 804.15.4 เป็นมาตรฐานที่ใช้งานสำหรับการสื่อสารไร้สายแบบ WPAN (Wireless Personal Area Network) กำหนดขึ้นสำหรับการรับ-ส่งข้อมูลเบื้องต้นในวงจรสัญญาณวิทยุ และมาตรฐานที่กำหนดขึ้นใช้ในชั้นเชื่อมโยงข้อมูล (Data Link Layer) มีจุดประสงค์เพื่อสื่อสารกันด้วยอัตราการส่งข้อมูลต่ำ และมีการใช้งานในระยะทางใกล้ๆ ระหว่างอุปกรณ์สองอุปกรณ์ ใช้กำลังไฟฟ้าต่ำ โดยมีสถาบันวิศวกรไฟฟ้า และอิเล็กทรอนิกส์ (IEEE) เป็นหน่วยงานหลักในการกำหนดมาตรฐานการใช้งาน เพื่อให้การพัฒนาเทคโนโลยีเป็นไปในทิศทางเดียวกัน โดยในส่วนของ XBee นี้ จะมาทำงานในระดับที่ต่ำกว่า (2 ระดับล่างสุด) คือ ระดับทางกายภาพ (Physical Layer) และระดับเครือข่ายคอมพิวเตอร์ และกลไกการควบคุม (Media Access Control Layer) ดังรูปที่ 2.1

[3] โหมด Unicast การรับส่งข้อมูลจากโหนดหนึ่งไปยังโหนดหนึ่งในลักษณะ 1 ต่อ 1

โหมด Broadcast การส่งข้อมูลจากโหนดต้นทางหนึ่งโหนดไปยังโหนดปลายทางทุกโหนดที่ติดต่ออยู่ในลักษณะของการแพร่กระจายข้อมูล แบบ 1 ต่อ ทั้งหมด



รูปที่ 2.1 ระดับของ PHY และ MAC

จากที่กล่าวมานั้น XBee จะสามารถสร้างเป็นเครือข่ายได้ เพราะอิงตามมาตรฐาน IEEE 802.15.4 และมีการจัดการในแบบของ XBee ในระดับถัดไป ทั้งนี้ IEEE 802.15.4 แบ่งชนิดอุปกรณ์ในเครือข่ายออกเป็น 2 ประเภท คือ FFD (Full Function Device) หมายถึง อุปกรณ์ที่สามารถทำงานได้ทุกอย่างในเครือข่าย และ RFD (Reduce Function Device) หมายถึง อุปกรณ์ที่ถูกลดความสามารถการทำงานในเครือข่าย

2.1.3 ชนิดของอุปกรณ์ XBee

อุปกรณ์ของ XBee มีอยู่ 2 ชนิด ประกอบไปด้วย อุปกรณ์ทางกายภาพ (Physical Device) และอุปกรณ์ทางตรรกะ (Logical Device)

อุปกรณ์ทางกายภาพ (Physical Device) มี 2 ประเภท คือ

1. Full Function Device (FFD) : ลูกข่าย (Routers) ที่เป็นสื่อกลางในการส่งข้อมูลจากอุปกรณ์อื่นๆ ใช้พลังงานจากสายไฟ (Power Line) ทำงานได้ในทุกรูปแบบ และสามารถทำเป็นจุดเชื่อมต่อกันได้

2. Reduced-Function Device (RFD) : เหมาะกับการเชื่อมต่อกันภายในเครือข่ายโดยใช้พลังงานจากแบตเตอรี่ ไม่สามารถถ่ายทอดข้อมูลจากอุปกรณ์อื่นๆ ได้ ทำให้ง่ายในเครือข่ายแบบสตาร์

อุปกรณ์ทางตรรกะ (Logical Device) มี 3 ประเภท คือ

1. แม่ข่าย (Coordinators) จะมีหน้าที่เป็นเหมือนกับแกนหลักของเครือข่าย เพื่อสร้างการสื่อสาร เชื่อมโยงเครือข่าย ระหว่างอุปกรณ์ปลายทางกับลูกข่าย, แม่ข่ายกับแม่ข่าย, แม่ข่ายกับลูกข่าย โดยกำหนดตำแหน่งแอดเดรส (Address) ให้อุปกรณ์ในวงเครือข่ายไม่ให้ซ้ำกัน จัดการเรื่องการหาเส้นทาง (Routing) ซึ่งสามารถเทียบได้กับ FFD ที่เป็นลูกข่ายสื่อกลางในการส่งข้อมูล

2. ลูกข่าย มีหน้าที่รับ-ส่งข้อมูลในเส้นทางต่างๆ ของเครือข่าย ซึ่งเทียบได้กับ FFD

3. อุปกรณ์ปลายทาง (End-device) เป็นโหนดสุดท้ายของเครือข่าย และเป็นอุปกรณ์ที่ใช้รับสัญญาณจากเซนเซอร์ปลายทาง โดยที่ใช้พลังงานต่ำในการทำงาน

2.1.4 เครือข่ายโปรโตคอล XBee

เครือข่ายไร้สายโดยใช้โปรโตคอล XBee สามารถตั้งค่าได้หลายแบบ ซึ่งสามารถแบ่งได้เป็นหมวดใหญ่ๆ 2 แบบ คือ จุดเชื่อมต่อ และอุปกรณ์ปลายทาง อุปกรณ์เชื่อมต่อของโปรโตคอล XBee เป็นอุปกรณ์ประเภท FFD ที่ได้รวมการทำงานของโปรโตคอล XBee ไว้เป็นจำนวนมาก ส่วนอุปกรณ์ปลายทางนั้น สามารถจะเป็นได้ทั้ง FFD และ RFD ซึ่ง RFD จัดเป็นอุปกรณ์ที่เล็ก และง่ายที่สุดของโปรโตคอล XBee ที่มีการทำงานของโปรโตคอล XBee น้อยมาก

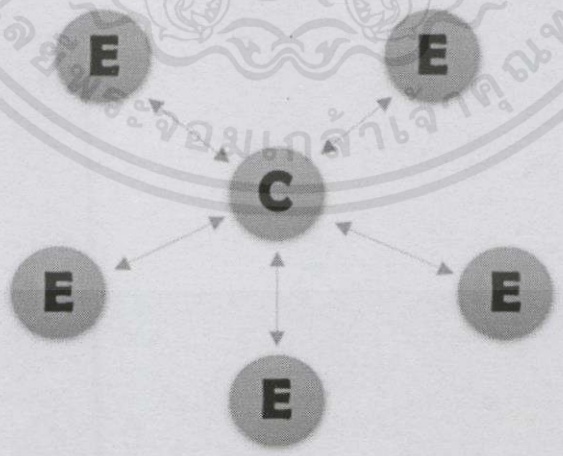
2.1.4.1 การเชื่อมต่อแบบระดับเดียว (Peer-to-Peer)

การเชื่อมต่อแบบระดับเดียว โดยกำหนดให้ตัวแรกเป็นแม่ข่าย ส่วนอีกตัวกำหนดเป็นลูกข่าย ดังรูปที่ 2.2

รูปที่ 2.2 การเชื่อมต่อแบบระดับเดียว

2.1.4.2 การเชื่อมต่อแบบสตาร์ (Star Network)

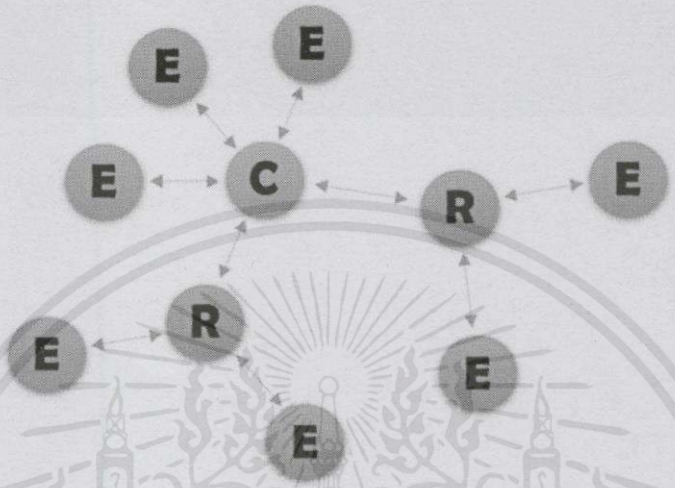
การเชื่อมต่อแบบสตาร์ เป็นการรับส่งข้อมูลแบบไม่เฉพาะเจาะจงจุดหมายปลายทาง โดย XBee ทุกตัวที่อยู่ในระบบเครือข่ายเดียวกันสามารถรับข้อมูลได้ทุกตัว ดังรูปที่ 2.3



รูปที่ 2.3 การเชื่อมต่อแบบสตาร์

2.1.4.3 การเชื่อมต่อแบบคัสเตอร์ (Cluster Tree)

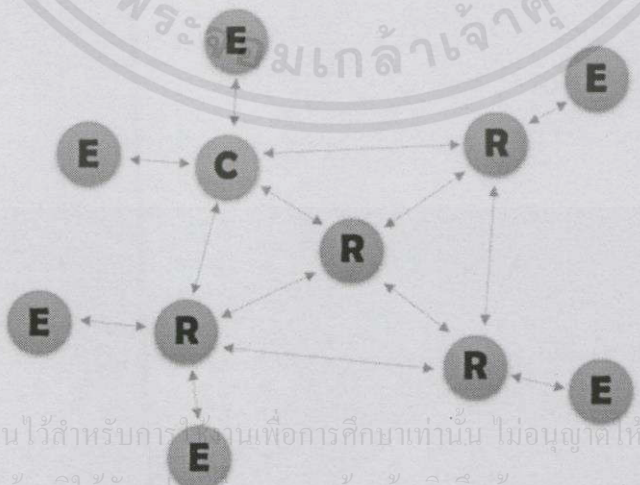
การรับส่งข้อมูลแบบส่งผ่าน หรือส่งต่อ เช่น A ต้องการติดต่อกับ C แต่ C อยู่ไกลจาก A จน A ไม่สามารถติดต่อกับ C ได้โดยตรง แต่เนื่องจากมี B อยู่ระหว่าง A กับ C ดังนั้นการเชื่อมต่อแบบคัสเตอร์จะใช้ B เป็นเหมือนตัวกลางเชื่อมการติดต่อ (Repeater) ระหว่าง A กับ C ดังรูปที่ 2.4



รูปที่ 2.4 การเชื่อมต่อแบบคัสเตอร์

2.1.4.4 การเชื่อมต่อแบบเมช (Mesh Network)

การเชื่อมต่อเครือข่ายแบบเมช เป็นโครงข่ายที่มีประสิทธิภาพสูงเนื่องจากข้อมูลสามารถส่งไปถึงเป้าหมายได้หลายเส้นทาง ทำให้ระบบสามารถรับ-ส่งข้อมูลไปยังจุดหมายปลายทางได้แน่นอน แม้จะเกิดความเสียหายของระบบในบางส่วนก็ตาม ระบบนี้จึงเป็นระบบที่ได้รับความนิยมเป็นอย่างมาก ดังรูปที่ 2.5



รูปที่ 2.5 การเชื่อมต่อแบบเมช

2.1.5 ขั้นตอนการทำงานของเครือข่ายโปรโตคอล IEEE 802.15.4 ผ่าน XBee

2.1.5.1 ขั้นตอนการทำงานของแม่ข่าย XBee (XBee Coordinator)

การสร้างแม่ข่ายของ XBee จะเริ่มต้นเครือข่าย โดยการตรวจสอบการใช้ช่องสัญญาณวิทยุภายในบริเวณรอบๆ ถ้ามีช่องสัญญาณที่ไม่ถูกใช้โดยแม่ข่ายตัวอื่นก็สามารถเริ่มต้นเครือข่ายได้ หลังจากนั้นแม่ข่ายก็จะทำหน้าที่เป็นศูนย์กลางของเครือข่าย รองรับการเข้าร่วมเครือข่ายอุปกรณ์ปลายทาง XBee และรองรับการร้องขออื่นๆ ตามมาตรฐานด้วยเช่นกัน

2.1.5.2 ขั้นตอนการทำงานอุปกรณ์ปลายทางของเครือข่าย XBee

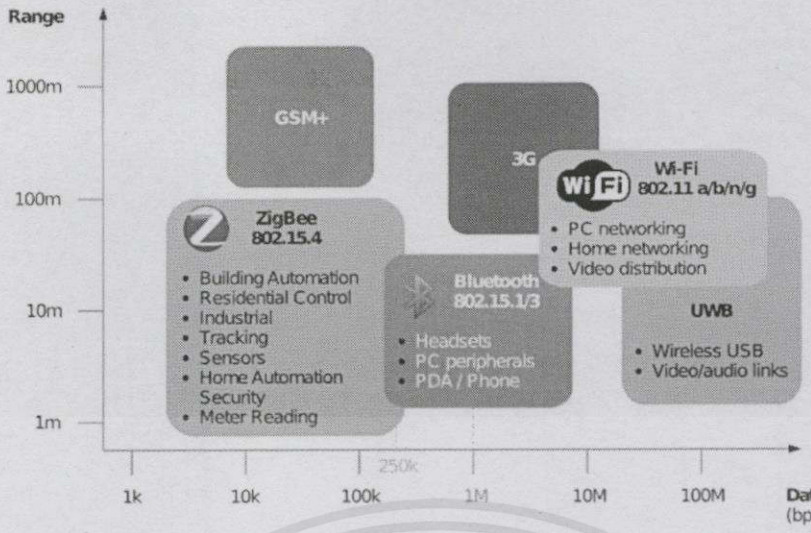
เริ่มต้นการทำงาน โดยการร้องขอการเข้าร่วมเครือข่ายไปยังแม่ข่ายประจำเครือข่ายนั้นๆ โดยการตรวจสอบผ่านช่องสัญญาณต่างๆ ว่า แม่ข่ายใช้ช่องสัญญาณใดอยู่ เมื่อเข้าร่วมเครือข่ายแล้ว อุปกรณ์ปลายทางจึงสามารถทำการร้องขอคำสั่งอื่นๆ ผ่านทางแม่ข่ายได้

2.1.6 XBee กับมาตรฐานการสื่อสารไร้สายแบบอื่นๆ

XBee เมื่อเปรียบเทียบกับมาตรฐานระบบสื่อสารต่างๆ แสดงดังตารางที่ 2.1 จะเห็นได้ว่าเทคโนโลยีของบลูทูธ (Bluetooth) และ Wifi นั้น กินพลังงานสูง จึงไม่เหมาะสมที่จะนำมาใช้งานกับเครือข่ายเซนเซอร์ ซึ่งอัตราการส่งข้อมูลไม่เยอะ และไม่จำเป็นต้องเร็วมาก เนื่องจากมีการเรียกข้อมูลเป็นคาบเวลา โดยมาตรฐานระบบสื่อสารต่างๆ สามารถแสดงได้ดังรูปที่ 2.6

ตารางที่ 2.1 การเปรียบเทียบมาตรฐานการสื่อสารแบบไร้สาย

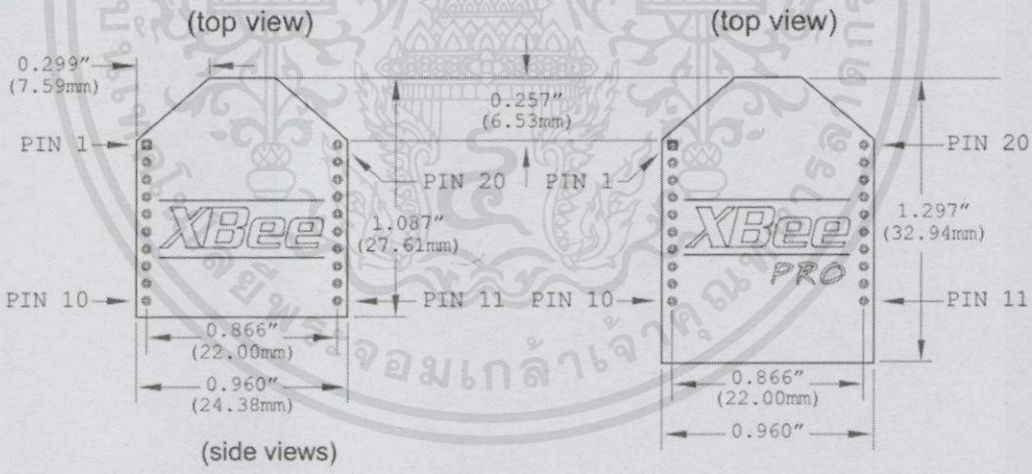
มาตรฐาน	XBee 802.15.4	WIFI 802.11b	บลูทูธ 802.15.1
ช่วงการส่ง (m.)	1-100	1-100	1-10
อายุการใช้งาน (วัน)	100-1000	0.5-5	1-7
ขนาดเครือข่าย (ต่อโหนด)	มากกว่า 64,000	32	7
แอพพลิเคชั่น	การติดตาม และควบคุม	เว็บ, อีเมล์, วีดีโอ	สายส่งแทนที่
ขนาด	4-32	1000	250
การประมวลผล (Kbps)	20-250	11,000	720



รูปที่ 2.6 ข้อแตกต่างระหว่างมาตรฐานไร้สายแบบต่างๆ

2.1.7 ขนาด และขาต่างๆ ของโมดูล XBee

ขนาดของโมดูล XBee มาตรฐานกว้าง 24.38 mm ยาว 27.61 mm มีขาเชื่อมต่อ 20 ขา แต่ละขาห่างกัน 2.00 mm แสดงดังตารางที่ 2.2 แต่ถ้าใน XBee รุ่น Pro ความยาวของโมดูลจะยาวขึ้นเป็น 32.94 mm แต่ตำแหน่ง และจำนวนของขาเท่าเดิม ดังรูปที่ 2.7



รูปที่ 2.7 ขนาด และขาของโมดูล XBee

ตารางที่ 2.2 รายละเอียดขาต่างๆ ของ XBee

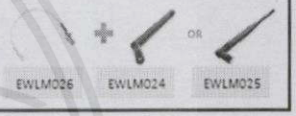
ขาที่	ชื่อ	หน้าที่
1	VCC	ขาไฟเลี้ยง 3.3 V
2	DOUT	ขา Tx ของพอร์ตอนุกรมโมดูล XBee
3	DIN	ขา Rx ของพอร์ตอนุกรมโมดูล XBee
4	D08	ขา 8 พอร์ต ดิจิตอล-เอาต์พุต
5	RESET	ขารีเซ็ต (Reset)
6	PWMO (RSSI)	ขา 0 พอร์ต สัญญาณ PWM หรือ ขา RSSI (Rx Signal Strength Indicator) ขาแสดงความแรงของสัญญาณที่รับได้
7	PWM1	ขา 1 พอร์ต สัญญาณ PWM
8	Reserved	ไม่ใช้
9	DTR/SLEEP/DI8	ขาสั่งให้ XBee เข้าโหมด SLEEP หรือ ขา 8 พอร์ต ดิจิตอล-อินพุต
10	GND	ขากราวด์
11	AD4/DIO4	ขา 4 ของพอร์ต อ่านสัญญาณอนาล็อก หรือ ขา 4 พอร์ต ดิจิตอล อินพุต/เอาต์พุต
12	CTS/DIO7	ขา CTS (Clear-To-Send Flow Control) หรือ ขา 7 พอร์ต ดิจิตอล อินพุต/เอาต์พุต
13	ON/SLEEP	ขาแสดงสถานะ XBee อยู่ในโหมด ON (ทำงาน) หรือโหมด SLEEP (หลับ)
14	VREF	ขาแรงดันอ้างอิงสำหรับอ่านสัญญาณอนาล็อก
15	ASSOC/AD5/DIO5	ขา Associated หรือ ขา 4 ของพอร์ตสัญญาณอนาล็อก หรือ ขา 5 พอร์ต ดิจิตอล อินพุต/เอาต์พุต
16	RTS/AD6/DIO6	ขา RTS (Request to send) หรือ ขา 6 ของพอร์ตสัญญาณอนาล็อก หรือ ขา 6 พอร์ต ดิจิตอล อินพุต/เอาต์พุต
17	AD3/DIO3	ขา 3 ของพอร์ตสัญญาณอนาล็อก หรือ ขา 3 พอร์ต ดิจิตอล อินพุต/เอาต์พุต
18	AD2/DIO2	ขา 2 ของพอร์ตสัญญาณอนาล็อก หรือ ขา 2 พอร์ต ดิจิตอล อินพุต/เอาต์พุต
19	AD1/DIO1	ขา 1 ของพอร์ตสัญญาณอนาล็อก หรือ ขา 1 พอร์ต ดิจิตอล อินพุต/เอาต์พุต
20	AD0/DIO0	ขา 0 ของพอร์ตสัญญาณอนาล็อก หรือ ขา 0 พอร์ต ดิจิตอล อินพุต/เอาต์พุต

2.1.8 ประเภทของโมดูล XBee

XBee แบ่งออกเป็นรุ่นที่มีคุณสมบัติแตกต่างกันออกไป เช่น XBee DigiMesh 2.4, XBee ZB, XBee-PRO XSC เป็นต้น การเชื่อมต่อ XBee ในเครือข่ายเดียวกันต้องใช้ XBee รุ่นเดียวกัน จากนั้นในแต่ละรุ่นจะมีรุ่นย่อยที่มีกำลังส่ง หรือการเชื่อมต่อสายอากาศที่แตกต่างกัน เช่น XBee-PRO DigiMesh 2.4, XBee-PRO ZB รุ่นย่อยเหล่านี้สามารถสื่อสารกับรุ่นย่อยอื่นที่มีกำลังส่งต่ำกว่าหรือสูงกว่าได้ เช่น XBee ZB สามารถสื่อสารกับ XBee-PRO ZB หรือ XBee-PRO ZB Wire Antenna ได้ แต่ไม่สามารถสื่อสารข้ามรุ่นกับ XBee-PRO XSC ได้ มีรายละเอียดดังนี้

2.1.8.1 XBee 802.15.4 (XBee Series 1)

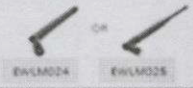
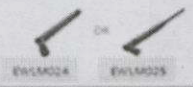
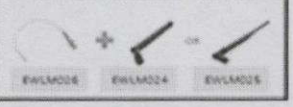
เป็นโมดูล XBee ที่ใช้ความถี่ 2.4 GHz บนมาตรฐาน IEEE 802.15.4 ดังรูปที่ 2.8 การเชื่อมต่อแบบเมช จึงใช้เฟิร์มแวร์ (Firmware) ของทาง Digi ชื่อว่าดิจิเมช "DigiMesh" โดยใน XBee Series 1 ก็จะแบ่งรุ่นตามกำลังส่งดังนี้[4]

ภาพสินค้า	ชื่อสินค้า	เสาอากาศ	ระยะ *(Line-of-Sight)	กำลังส่ง	กระแส	อุปกรณ์ต่อเพิ่ม
	XBee 1mW PCB Antenna	PCB Antenna	100m	1mW	3.3V @50mA	-
	XBee 1mW Wire Antenna	Wire Antenna	100m	1mW	3.3V @50mA	-
	XBee Pro 60mW Wire Antenna	Wire Antenna	1,500m	60mW	3.3V @215mA	-
	XBee Pro 60mW U.FL Connection	U.FL Connector	1,500m	60mW	3.3V @215mA	 EWLM026 EWLM024 OR EWLM025

รูปที่ 2.8 อุปกรณ์ XBee Series 1

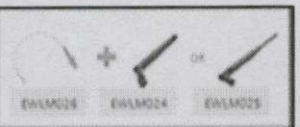
2.1.8.2 XBee Series 2

เป็นโมดูล XBee ที่ใช้ความถี่ 2.4 GHz บนมาตรฐาน IEEE 802.15.4 และมีระดับ (Stack Layer) ของ XBee โดยใน XBee Series 2 แบ่งรุ่นตามกำลังส่ง ดังรูปที่ 2.9

ภาพสินค้า	ชื่อสินค้า	เสาอากาศ	ระยะ *(Line-of-Sight)	กำลังส่ง	กระแส	อุปกรณ์ต่อเพิ่ม
	XBee 2mW PCB Antenna	PCB Antenna	120m	2mW	3.3V @40mA	
	XBee Pro 50mW RPSMA	RPSMA	1,600m	50mW	3.3V @295mA	 EWLM024 OR EWLM025
	XBee 2mW RPSMA	RPSMA	120m	2mW	3.3V @40mA	 EWLM024 OR EWLM025
	XBee Pro 60mW U.FL Connection	U.FL Connector	1,500m	60mW	3.3V @215mA	 EWLM026 EWLM024 OR EWLM025

รูปที่ 2.9 อุปกรณ์ XBee Series 2

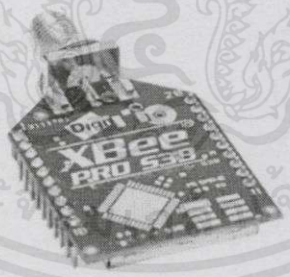
[4] *ระยะ (Line-of-Sight) โดยระยะที่ทำได้ขึ้นอยู่กับสภาพแวดล้อมของระบบและสายอากาศที่ใช้ เนื่องจากความถี่ 2.4 GHz เป็นย่านความถี่สูง ทำให้อัตราการลดทอนสัญญาณสูงและสิ่งกีดขวางมีผลอย่างมากกับระยะทางที่ใช้งานได้

	XBee 2mW U.FL Connection	U.FL Connector	120m	2mW	3.3V @40mA	
	XBee 2mW Wire Antenna	Wire Antenna	120m	2mW	3.3V @40mA	
	XBee Pro 50mW PCB Antenna	PCB Antenna	1,600m	50mW	3.3V @295mA	
	XBee Pro 50mW Wire Antenna	Wire Antenna	1,600m	50mW	3.3V @295mA	
	XBee Pro 50mW	U.FL Connector	1,600m	50mW	3.3V @295mA	

รูปที่ 2.9 (ต่อ) อุปกรณ์ XBee Series 2

2.1.8.3 XBee 900 MHz

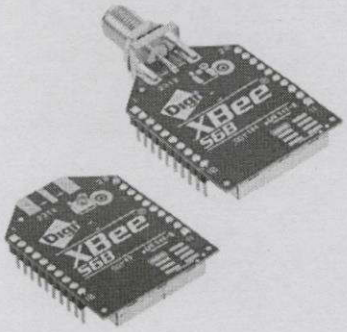
เป็นโมดูล XBee ความถี่ 900 MHz ทำให้ระยะการรับ-ส่งได้ไกลกว่าทั้ง รุ่น 1 และ 2 ดังรูปที่ 2.10 โดยจะมีรุ่นย่อย 2 รุ่นที่เป็นเฟิร์มแวร์ที่มีการเชื่อมต่อแบบจุดต่อหลายจุด (Point to Multipoint) และดิจิเมช อัตราการรับส่งข้อมูลสูงสุดที่ 200 Kbps กำลังส่ง 3.3 V ที่ 250 mW โดยยังมีรุ่นย่อยออกเป็นประเภทของสายอากาศต่างๆ



รูปที่ 2.10 อุปกรณ์ XBee 900 MHz

2.1.8.4 XBee Wifi

เป็นโมดูล XBee ความถี่ 2.4 GHz บนมาตรฐาน IEEE 802 นอกจากความสามารถในการสื่อสารกับโมดูล XBee Wifi ด้วยกันเองได้แล้ว ยังสามารถสื่อสารกับอุปกรณ์ Wifi อื่นได้ หรือนำโมดูลเชื่อมต่อกับอินเตอร์เน็ต สามารถเชื่อมต่อกับของเซิร์ฟเวอร์จำนวนมาก (Cloud) โดยมีอัตราการรับส่งข้อมูลสูงสุดที่ 72 Mbps กำลังส่ง 3.3 V ที่ 309 mA โดยโมดูล XBee Wifi แสดงดังรูปที่ 2.11



รูปที่ 2.11 อุปกรณ์ XBee Wifi

2.1.9 โหมดการสื่อสารของ XBee

2.1.9.1 โหมด AT

ตัว XBee มีการติดต่อกัน และรับ-ส่งข้อมูลกันตลอดเวลา (Transparent) ข้อมูลที่ส่งจาก XBee ตัวหนึ่งจะส่งไปยัง XBee อีกตัวหนึ่งตามแอดเดรสปลายทางที่กำหนดไว้ในรีจิสเตอร์ของโมดูลนั้นๆ โดยสามารถเปลี่ยนเป็นโหมดคำสั่งเพื่อทำการปรับเปลี่ยนค่าพารามิเตอร์บนโมดูล XBee ได้ เรียกว่าการใช้คำสั่ง AT

2.1.9.2 โหมด API

การส่งข้อมูลที่มีความยืดหยุ่นมากกว่า แต่ก็ซับซ้อนกว่าด้วยเล็กน้อย โดยต้องรับ-ส่งข้อมูลตามโปรโตคอลที่กำหนดไว้ โดยมีชุดคำสั่งแสดงดังรูปที่ 2.12

รูปที่ 2.12 ชุดคำสั่งโหมด API

API ใน 1 ชุดคำสั่งจะแบ่งออกเป็น 4 กลุ่มด้วยกันคือ

1. ส่วนเริ่มต้น (Start Delimiter) เป็นการเริ่มต้นของ API จะใช้ 0X7E เป็นตัวคั่นเพื่อบอกให้รู้ว่านี่คือจุดเริ่มต้นของ API มีขนาด 1 ไบต์

2. ระยะ (Length) คือ จำนวนไบต์ของ Frame Data มีขนาด 2 ไบต์

3. โครงสร้างข้อมูล (Frame Data) คือ คำสั่ง และข้อมูลที่ต้องการส่ง หรือรับมา ซึ่งจะมีรายละเอียดแตกต่างกันไป ตามรูปแบบโครงสร้างที่ใช้

4. การตรวจสอบความถูกต้อง (Checksum) เป็นตัวที่เอาไว้ตรวจเช็คความถูกต้องของข้อมูลที่ได้รับมาว่าถูกต้องหรือไม่ มีขนาด 1 ไบต์

2.1.10 การประยุกต์การใช้งาน XBee

การประยุกต์การใช้งาน XBee แบ่งตามประเภทของข้อมูลข่าวสารที่มี 3 แบบ คือ

1. ข้อมูลแบบแบ่งออกเป็นช่วงเวลา (Periodic) โปรแกรมสามารถควบคุมอัตราการส่ง และตรวจจับสัญญาณกระตุ้น เช็คข้อมูล และทำให้ข้อมูลไม่เคลื่อนไหว ใช้สำหรับเซ็นเซอร์ และมิเตอร์ต่างๆ

2. ข้อมูลแบบเป็นช่วงๆ (Intermittent) เป็นลักษณะที่มีการส่งผ่านข้อมูล เมื่อมีการใช้งาน เช่น สวิตซ์ไฟ

3. ข้อมูลแบบใช้งานซ้ำ (Repetitive Low Battery) ใช้ในการที่ต้องการศักยภาพน้อยๆ โดยการสื่อสารจะใช้วิธีจัดสรรช่องเวลา และสามารถใช้กลไกการรักษาความปลอดภัยของ XBee แบบ GTS (Guarantee Time Slot) เพื่อรับประกันคุณภาพของการบริการนำไปใช้งาน

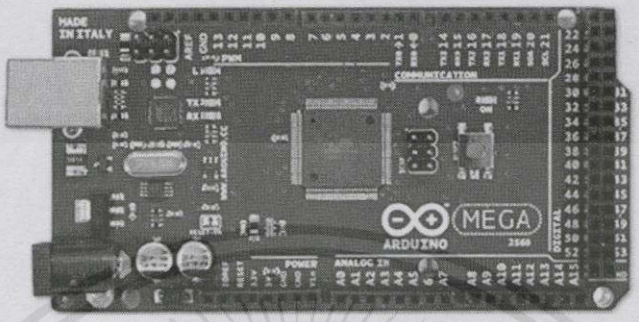
2.2 ไมโครคอนโทรลเลอร์ AVR

ไมโครคอนโทรลเลอร์ AVR เป็นหนึ่งในไมโครคอนโทรลเลอร์ที่ผลิตขึ้นโดยบริษัท ATMEL AVR ซึ่งจัดเป็นไมโครคอนโทรลเลอร์ตระกูลใหม่ของบริษัท ATMEL ที่มีสถาปัตยกรรมแบบ RISC (Advanced RISC Architecture) คือ หนึ่งคำสั่งทำงานโดยใช้สัญญาณนาฬิกาเพียง 1 ลูก (Instructions in a Single Clock Cycle) ซึ่งเร็วกว่าไมโครคอนโทรลเลอร์ MCS-51 ที่ใช้ 1 คำสั่ง/12 วงรอบการทำงานของคำสั่ง (Machine Cycle) เป็นไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพสูง แบ่งออกเป็นหลายอนุกรม ในแต่ละอนุกรมยังแบ่งออกเป็นหลายเบอร์ เพื่อรองรับความต้องการที่แตกต่างของผู้ใช้งาน ในขณะที่ยังคงความมีประสิทธิภาพที่เท่ากัน โดยในโครงงานนี้ได้ใช้ AVR แบบ Arduino ATmega2560 ซึ่งแสดงโครงสร้าง และรายละเอียดต่างๆ ดังรูปที่ 2.13, รูปที่ 2.14 และรูปที่ 2.15 ตามลำดับ

2.2.1 คุณสมบัติของบอร์ด Arduino ATmega2560

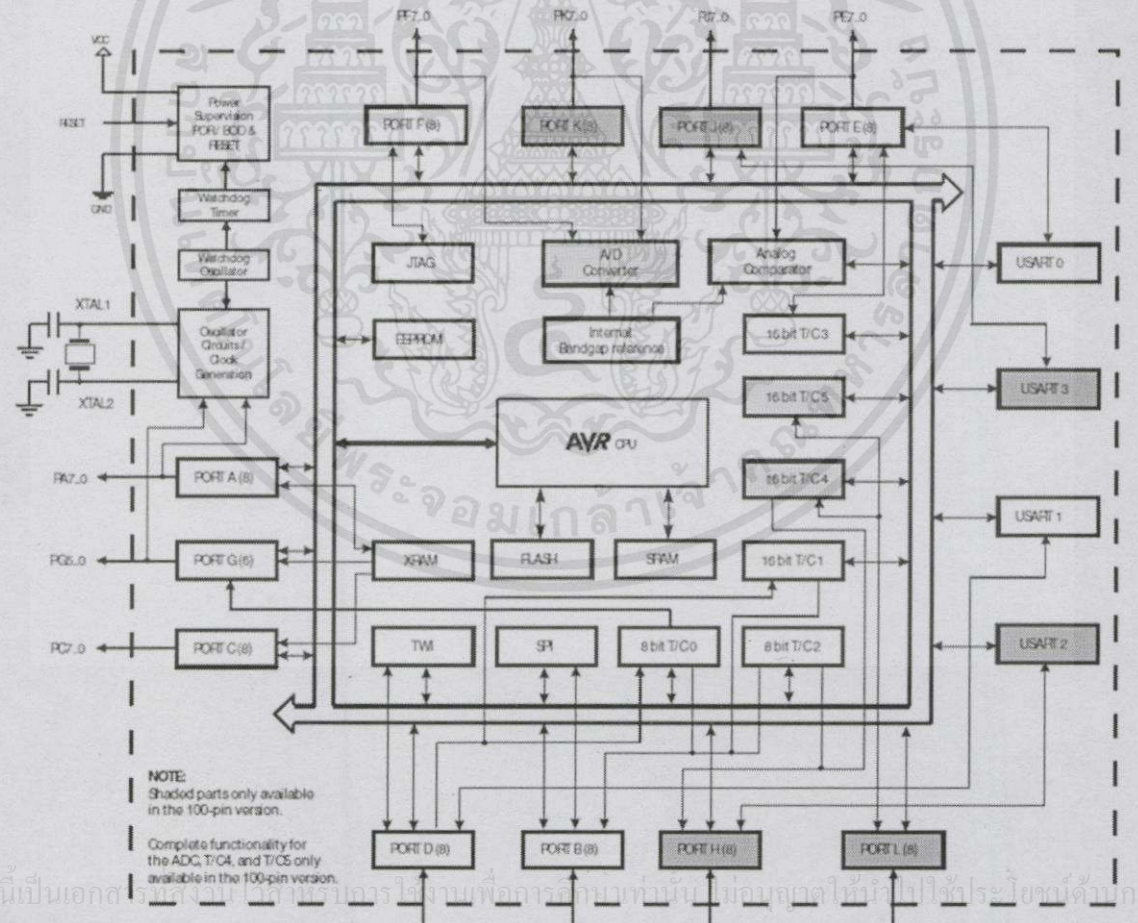
1. แรงดันอินพุต (Recommended) 7-12 V แรงดันไฟฟ้าในการทำงาน 5 V
2. แรงดันอินพุต (limits) 6-20 V
3. หน่วยความจำแฟลช 256 KB (8 KB for Bootloader)
4. หน่วยความจำแบบ SRAM 8 KB
5. หน่วยความจำแบบ EEPROM 4 KB
6. บอร์ดใช้ไฟเลี้ยง 7 ถึง 12 V (Input Voltage)
7. ระดับแรงดันไฟฟ้าในการทำงาน และขาสัญญาณอยู่ที่ 5 V (TTL)
8. มีขาดิจิตอลอินพุต/เอาต์พุต 54 ขา (เป็น PWM Output 15 ขา)

9. มีขาอนาล็อกอินพุต 16 ขา
10. ความสามารถในการใช้งานความถี่สัญญาณนาฬิกาอยู่ที่ 16 MHz (Clock Speed)
11. กระแสไฟตรง ต่อขา I/O 40 mA
12. กระแสไฟตรง ของขา 3.3 V 50 mA

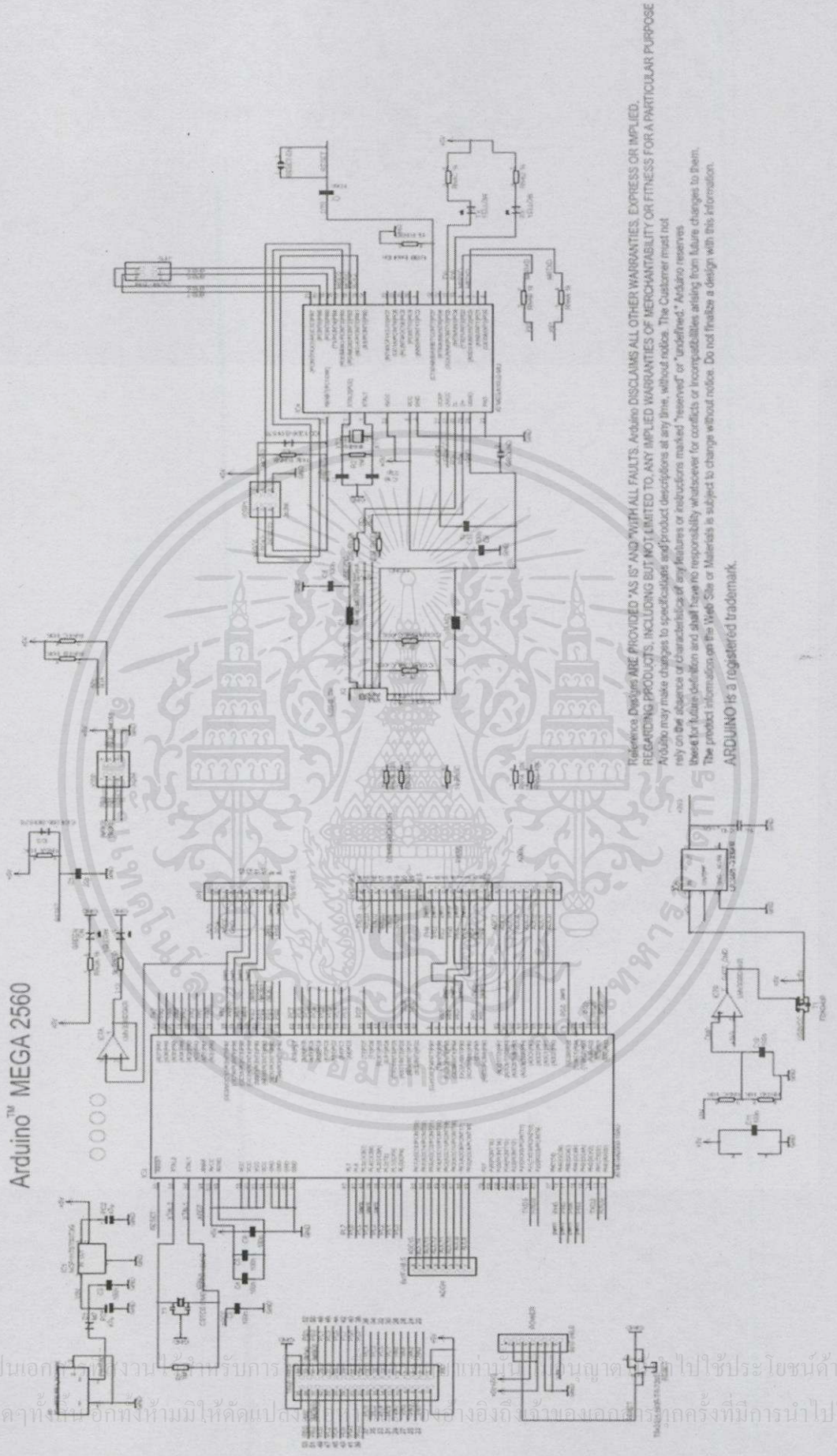


รูปที่ 2.13 ลักษณะบอร์ด AVR ATmega2560

2.2.2 รายละเอียดภายในไมโครคอนโทรลเลอร์ AVR ATmega2560



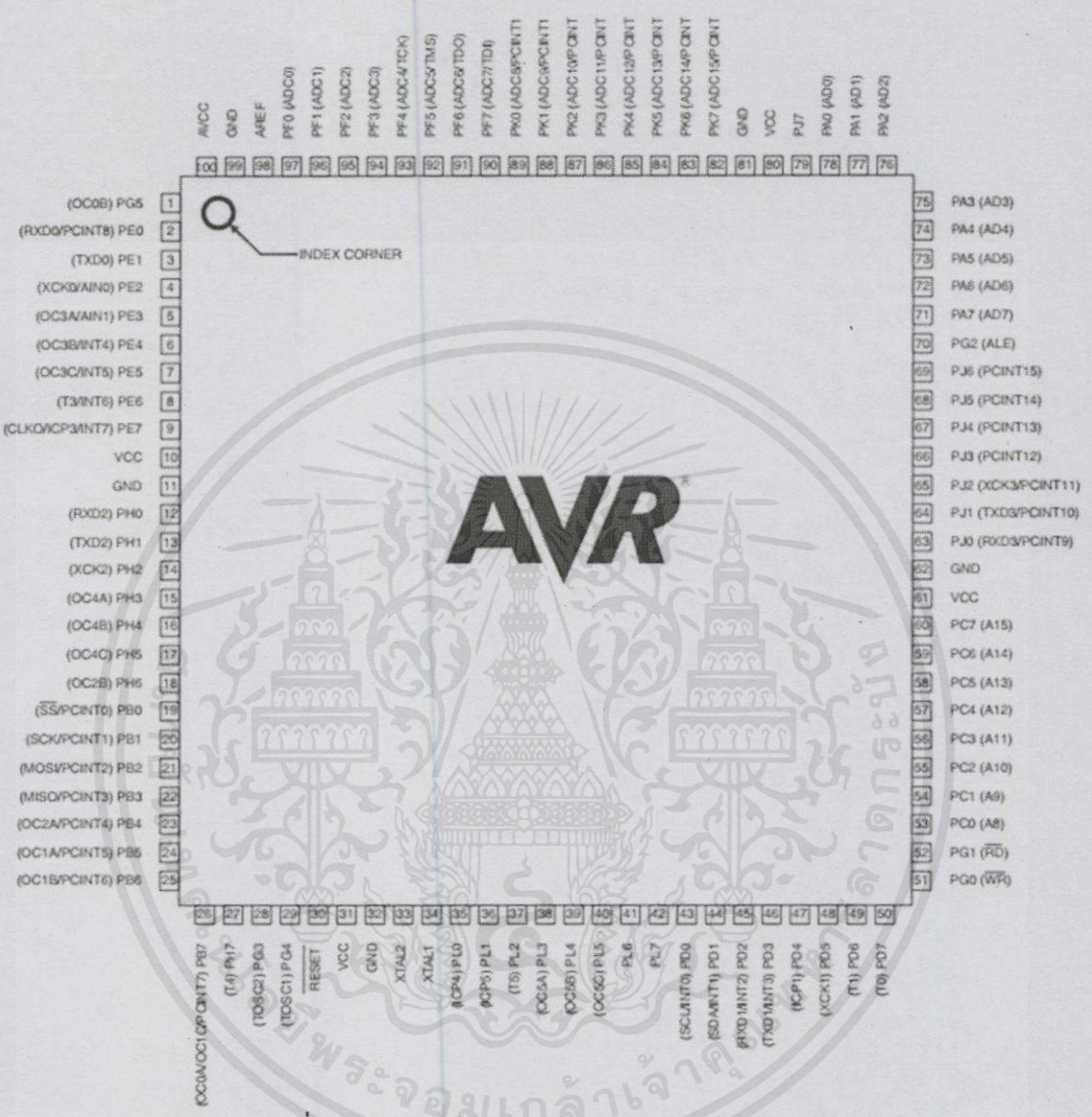
รูปที่ 2.14 บล็อกไดอะแกรม AVR ATmega2560



รูปที่ 2.15 วงจรไฟฟ้า AVR ATmega2560

2.2.3 ขาพอร์ตอินพุต เอาต์พุตไมโครคอนโทรลเลอร์ AVR

ATmega2560 มีขาพอร์ตอินพุต-เอาต์พุต แสดงดังรูปที่ 2.16



รูปที่ 2.16 ขาพอร์ตของ AVR ATmega2560

- VCC ขาสำหรับจ่ายไฟ

- GND ขากราวด์

- พอร์ต B (PB7...PB0) ขาพอร์ตอินพุต/เอาต์พุตดิจิตอล สามารถกำหนดการพูลอัพภายในขาพอร์ตได้ (Internal Pull-up Register)

- พอร์ต C (PB7...PB0) ขาพอร์ตอินพุต/เอาต์พุตดิจิตอล สามารถกำหนดการพูลอัพภายในขาพอร์ตได้

- พอร์ต D (PB7...PB0) ขาพอร์ตอินพุต/เอาต์พุตดิจิตอล สามารถกำหนดการพูลอัพภายในขาพอร์ตได้

- RESET ขารีเซ็ตวงจร

- XTAL1 ขาต่อคริสตอลออสซิเลเตอร์เพื่อสร้างสัญญาณนาฬิกา ในการกำหนดจังหวะการทำงานของไมโครคอนโทรเลอร์ ช่องที่ 1 ด้านอินพุต

- XTAL2 ขาต่อคริสตอลออสซิเลเตอร์เพื่อสร้างสัญญาณนาฬิกา ในการกำหนดจังหวะการทำงานของไมโครคอนโทรเลอร์ ช่องที่ 2 ด้านเอาต์พุต

- AREF ขาแรงดันอะนาล็อก อ้างอิงสำหรับโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอล

2.2.4 หลักการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์

การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ เช่น โมดูลพอร์ต ทำหน้าที่เกี่ยวกับพอร์ตอินพุต/เอาต์พุต โมดูลไทม์เมอร์ เกี่ยวกับการนับเวลา หรือการจับเวลา เมื่อเข้าใจการทำงานของโมดูลที่ต้องการใช้งานแล้ว ให้ศึกษาและทำความเข้าใจกับรีจิสเตอร์ที่เกี่ยวข้องกับโมดูลนั้นๆ เนื่องจากรีจิสเตอร์เปรียบเสมือนกับสวิตช์เปิด-ปิดการใช้งานโมดูลนั้นๆ เมื่อกำหนดสวิตช์เปิด-ปิดเรียบร้อยแล้ว ไมโครคอนโทรลเลอร์ก็จะเริ่มทำงานตามที่กำหนดไว้ในรีจิสเตอร์ที่เกี่ยวข้องทันที

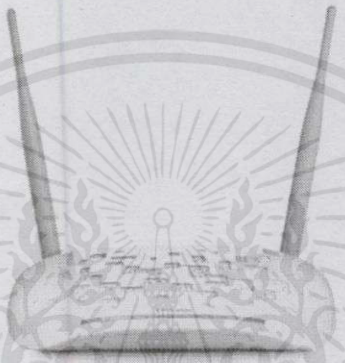
รีจิสเตอร์บางโมดูลจะมีบิตเฉพาะสำหรับใช้ในการเปิด-ปิดการใช้งาน หรืออาจเรียกได้ว่าเป็นสวิตช์หลัก แต่บางโมดูลจะไม่มี เพียงกำหนดรีจิสเตอร์ที่จะใช้งานก็เริ่มต้นทำงานได้ทันที บางโมดูลที่เกี่ยวข้องกับการอินเตอร์รัปต้องมีการกำหนดฟังก์ชันที่เกี่ยวข้องกับอินเตอร์รัปของโมดูลที่ใช้งานด้วย หลังจากที่กำหนดค่าบิตในรีจิสเตอร์ที่ใช้งานโมดูลแล้ว การเขียนโปรแกรมจะขึ้นอยู่กับพื้นฐานการเขียนโปรแกรม และพื้นฐานทางด้านอิเล็กทรอนิกส์ของแต่ละบุคคล หากมีพื้นฐานทางการเขียนโปรแกรมเพียงอย่างเดียว โดยไม่มีพื้นฐานทางด้านอิเล็กทรอนิกส์เลย ผลลัพธ์การทำงานของโปรแกรมที่ได้อาจไม่ถูกต้อง เนื่องจากวงจรการใช้งานผิดพลาด ดังนั้นการเขียนโปรแกรมเพื่อควบคุมไมโครคอนโทรลเลอร์จึงต้องมีพื้นฐานทางด้านอิเล็กทรอนิกส์บ้าง ซึ่งจะช่วยให้การเขียนโปรแกรมและการใช้งานของไมโครคอนโทรลเลอร์เป็นไปตามความต้องการมากยิ่งขึ้น

2.3 เราเตอร์ตัวส่ง และตัวรับ

เราเตอร์ (Router) คือ อุปกรณ์ที่ทำหน้าที่เชื่อมต่อกับระบบเครือข่ายอย่างหนึ่ง โดยหน้าที่หลัก คือ การหาเส้นทางในการส่งผ่านข้อมูลที่ดีที่สุด และเป็นตัวกลางสำคัญในการส่งต่อข้อมูลไปยังเครือข่ายอื่นๆ ทั้งนี้เราเตอร์ยังสามารถเชื่อมโยงเครือข่ายที่ใช้สื่อสัญญาณหลายแบบแตกต่างกันได้ โดยในการทดลองครั้งนี้จะใช้การเชื่อมโยงเครือข่ายแบบระบบเอทเทอร์เน็ต (Ethernet) ซึ่งเป็นระบบที่นิยมใช้กันอย่างกว้างขวางในปัจจุบัน โดยเราเตอร์ที่ใช้ในการทดลอง มีดังนี้

2.3.1 ตัวส่งสัญญาณ Wifi (TP-Link)

เป็นเราเตอร์ตัวส่งที่ใช้ในการส่งสัญญาณ Wifi แสดงดังรูปที่ 2.17 เพื่อให้สามารถเชื่อมต่อเข้ากับระบบที่ได้ทำการตั้งค่าไว้ ผ่านเราเตอร์ตัวรับสัญญาณ Wifi โดยเราเตอร์ตัวส่ง เป็นเพียงเราเตอร์ที่ใช้ในการทดลอง หากในอนาคตมีผู้นำการทดลองครั้งนี้ไปประยุกต์ใช้งาน ก็สามารถที่จะใช้เราเตอร์ตัวอื่นๆ ที่มีอยู่ตามบ้านพักอาศัย หรือตามสถานที่ทำงานต่างๆ ได้ ซึ่งจะมีความแตกกต่างที่การตั้งค่า เราเตอร์ให้สามารถเชื่อมต่อกันได้ โดยเหตุที่เลือกใช้เราเตอร์รุ่นนี้ เนื่องจากสามารถตั้งค่าให้เชื่อมต่อกับเราเตอร์ตัวรับได้สะดวก และรวดเร็วมากกว่าเราเตอร์ยี่ห้ออื่นๆ



รูปที่ 2.17 เราเตอร์ตัวส่งสัญญาณ Wifi (TP-Link TD-W8961ND 300 Mbps Wireless N ADSL2)

2.3.2 ตัวรับสัญญาณ Wifi (Tenda W150M)

เราเตอร์ตัวรับสัญญาณ Wifi คือ ตัวทำหน้าที่รับสัญญาณ เพื่อทำการส่งต่อไปยังอุปกรณ์ที่เชื่อมต่อ โดยในการทดลองได้เลือกใช้เราเตอร์ตัวรับสัญญาณ Wifi Tenda W150M โดยแสดงดังรูปที่ 2.18 ซึ่งเป็นเราเตอร์ที่นิยมใช้กันอย่างแพร่หลาย และสะดวกในการตั้งค่าเพื่อให้สามารถใช้งานได้

คุณสมบัติของ Tenda

1. Tenda จัดเป็นเราเตอร์ขนาดเล็กที่สามารถพกพาได้ รองรับการทำงานได้ 5 โหมด คือ AP, Client+AP (WiFi Bridge), WDS+AP, WISP และเราเตอร์แลนไร้สาย (Wireless Router) นิยมใช้ 2 โหมด คือ

   - AP = แอคเซสพอยต์ต่อ โมเด็ม/สวิตช์ ADSL แล้วกระจายเป็น Wifi
   - Client+AP = รับสัญญาณผ่าน Wifi แล้วกระจายต่อผ่าน Wifi หรือแลน

2. รองรับ WiFi b/g/n ความเร็วสูงสุด 150 Mbps
3. รองรับความปลอดภัย (Security) แบบ 64/128-บิต WEP, WPA และ WPA2
4. รองรับ WPS (PBC and PIN)
5. มี LAN/WAN 1 ช่อง แบบ 10/100Mbps
6. รองรับการตั้งค่าผ่านเว็บ (Web Management)

7. ต่อกำลังไฟฟ้า (Power) เป็นแบบกระแสสลับ (AC) และ USB
8. ขนาด 103 x 66 x 18 mm

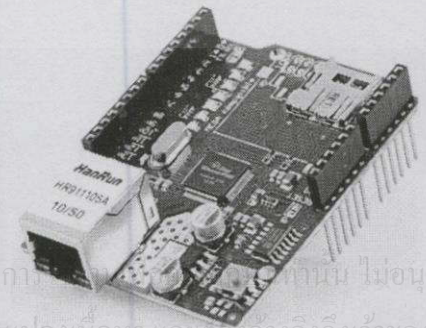
รูปที่ 2.18 เราเตอร์ตัวรับสัญญาณ Wifi (Tenda W150M)

2.3.3 Ethernet Shield

ใช้สำหรับการเชื่อมต่อบอร์ด Arduino เข้ากับระบบอินเตอร์เน็ต แสดงดังรูปที่ 2.19 ซึ่งสามารถใช้งานได้ง่ายเพียงเสียบคอนเน็กเตอร์เข้ากับบอร์ด Arduino แล้วต่อสายแลน และที่สำคัญข้อมูลไลบรารี่ รวมไปถึงโค้ดต่างๆ ยังไม่จำเป็นจะต้องเสียค่าใช้จ่ายเพิ่มเติม เนื่องจาก Arduino ถือเป็นระบบปฏิบัติการที่เปิดโอกาสให้บุคคลอื่นสามารถนำเอาระบบไปพัฒนาต่อได้

คุณสมบัติของ Ethernet Shield

1. ใช้ร่วมกับบอร์ด Arduino
2. ระดับแรงดันไฟฟ้าในการทำงาน 5 V (บอร์ด Arduino. เป็นแหล่งจากไฟ)
3. ความเร็วในการเชื่อมต่อ 10/100 Mb
4. เชื่อมต่อกับ Arduino บนพอร์ต SPI
5. แรงดันอินพุต 36 ถึง 57 V
6. ป้องกันเกิดภาวะโหลดเกิน และการช็อตของวงจร (Overload and Short-circuit)
7. แรงดันเอาต์พุต 9 V



รูปที่ 2.19 Ethernet Shield

2.4 รีเลย์ (Relay)

จากการทดลองครั้งนี้ ใช้บอร์ดรีเลย์ขนาด 8 ช่อง แสดงดังรูปที่ 2.20 ซึ่งมีเอาต์พุตหน้าสัมผัสที่รีเลย์เป็น NO/COM/NC สามารถใช้กับโหลดได้ทั้งแรงดันไฟฟ้ากระแสตรง (DC) และกระแสสลับ (AC) โดยใช้สัญญาณในการควบคุมการทำงานด้วยสัญญาณโลจิก TTL[5]

คุณสมบัติของรีเลย์

1. รีเลย์เอาต์พุตแบบ SPDT จำนวน 8 ช่อง
2. สั่งงานด้วยระดับแรงดัน TTL
3. เอาต์พุตหน้าสัมผัส (CONTACT OUTPUT) ของรีเลย์ สามารถรับแรงดันได้สูงสุดที่ 250 VAC 10 A, 30 VDC 10 A
4. มีไฟ LED แสดงสถานะการทำงานของรีเลย์ และแสดงสถานะของบอร์ด
5. มีจัมพ์เปอร์สำหรับเลือกว่าจะใช้กราวด์ร่วม หรือแยก
6. มีออปโต้คัปเปลอร์ (OPTO-ISOLATED) เพื่อแยกกราวด์ส่วนสัญญาณควบคุมกับไฟที่ขับรีเลย์ออกจากกัน

ข้อควรระวังในการใช้งาน

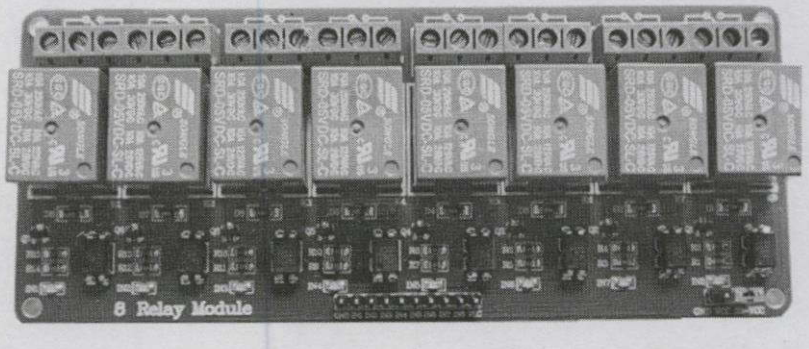
1. สัญญาณที่ควบคุมรีเลย์ต้องไม่เกินระดับ TTL
2. รีเลย์เอาต์พุตสามารถขับโหลดได้ไม่เกิน 10 A, 250 VAC
3. ควรหลีกเหลี่ยงการต่อวงจรให้เกิดการลัดวงจร ซึ่งจะทำให้รีเลย์เสียหายได้
4. ควรอ่านเอกสารก่อนการต่อวงจรจริง
5. ถ้าต้องการแยกกราวด์ระหว่างไฟสัญญาณ และไฟที่ขับรีเลย์ ให้ทำการถอดจัมเปอร์ออก โดยนำไฟที่ใช้ขับรีเลย์ต่อเข้าขา VCC กับขา GND และสัญญาณต่อเข้าขา IN1-IN8 กับกราวด์ของสัญญาณต่อเข้าขา COM ฝั่งอินพุต

คุณลักษณะ

1. ควบคุมไฟกระแสตรงสูงสุด 30VDC 10A และไฟกระแสสลับสูงสุด 250VAC 10A
2. ระดับสัญญาณอินพุตควบคุมแบบ TTL ทำงานด้วยสัญญาณแบบ Active High[7]
3. ขนาดรูยึดบอร์ด 3 mm
4. ขนาด (L x W x H) : 135 x 55 x 20 mm

[5] TTL (Transistor-transistor logic) :วงจรลอจิกเกตมีการขยายด้วยอุปกรณ์ คือทรานซิสเตอร์-ทรานซิสเตอร์ลอจิก

[7] Active High คือวงจรที่ทำงานเมื่อสัญญาณเป็นสถานะสูง (High) หรือลอจิก 1



รูปที่ 2.20 บอร์ดรีเลย์ขนาด 8 ช่อง

2.5 ระบบปฏิบัติการแอนดรอยด์ (Android)

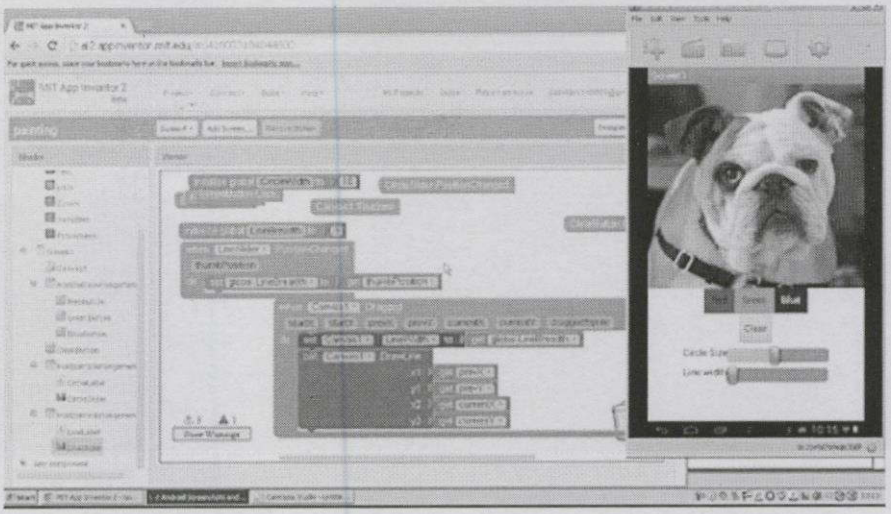
แอนดรอยด์คือ ระบบปฏิบัติการแบบเปิดเผยซอร์ฟแวร์ ที่ถือได้ว่าได้รับความนิยมเป็นอย่างสูง เนื่องจากอุปกรณ์ที่ใช้ในระบบปฏิบัติการแอนดรอยด์มีจำนวนมาก ซึ่งระบบปฏิบัติการแอนดรอยด์หากมองในทิศทางสำหรับนักพัฒนาโปรแกรม (Programmer) แล้วนั้น การพัฒนาโปรแกรมเพื่อใช้งานบนระบบปฏิบัติการแอนดรอยด์ก็ไม่ใช่เรื่องที่ยาก เนื่องจากมีข้อมูลในการพัฒนาระบบรวมทั้ง Android SDK (Software Development Kit) เตรียมไว้ให้กับนักพัฒนาได้เรียนรู้

2.5.1 การพัฒนาแอพพลิเคชั่นแอนดรอยด์ (Android Application Development)

แอนดรอยด์ จะใช้โครงสร้างของภาษาจาวา ในการพัฒนาเป็นหลัก และในการเขียนโปรแกรมจะมี API Library ที่ถูกพัฒนาให้เลือกใช้มากมาย เช่น API Library ที่ช่วยจัดการเกี่ยวกับพวกกราฟิก (Graphic) การออกแบบมัลติมีเดีย (Multimedia) หรือ API Library ที่เกี่ยวข้องระบบกำหนดตำแหน่งบนพื้นโลกผ่านดาวเทียม (GPS), บลูทูธ, 3G หรือ Wifi ที่จะเข้ามาจัดการเกี่ยวกับฐานข้อมูล

2.5.1.1 เครื่องมือพัฒนาโปรแกรมแอพพลิเคชั่นในระบบแอนดรอยด์

1. MIT App Inventor เป็นเครื่องมือตัวหนึ่งที่ใช้ในการสร้างแอพพลิเคชั่นบนอุปกรณ์ที่ใช้ระบบปฏิบัติการแอนดรอยด์ ที่มีความง่ายต่อการทำความเข้าใจ ใช้งานง่าย โดยความร่วมมือระหว่างบริษัทกูเกิล (Google) และสถาบันเทคโนโลยีแมสซาชูเซตส์ (MIT) ทำการผลิต App Inventor ออกมา โดยมีวัตถุประสงค์เพื่อให้ผู้ที่สนใจนั้น สามารถทำความเข้าใจในหลักการพัฒนาแอพพลิเคชั่นบนอุปกรณ์ที่ใช้ในระบบปฏิบัติการแอนดรอยด์ซึ่งเป็นของกูเกิล โดยจุดเด่นที่ทำให้ App Inventor ถูกเลือกใช้สำหรับเป็นเครื่องมือแรกในการเรียนการสอน หรือการเริ่มต้นพัฒนาแอพพลิเคชั่นบนระบบปฏิบัติการแอนดรอยด์ เนื่องจากขั้นตอนการพัฒนาแอพพลิเคชั่นเป็นแบบวิชวลไลเซชัน (Visualization) คือการใช้บล็อก (Block) แทนรหัสคำสั่ง ดังรูปที่ 2.21 โดยในการพัฒนาแอพพลิเคชั่น ผู้พัฒนาสามารถเข้าเว็บไซต์ ได้ที่ http://beta.appinventor.mit.edu



รูปที่ 2.21 บล็อกแทนรหัสคำสั่งในโปรแกรม App Inventor

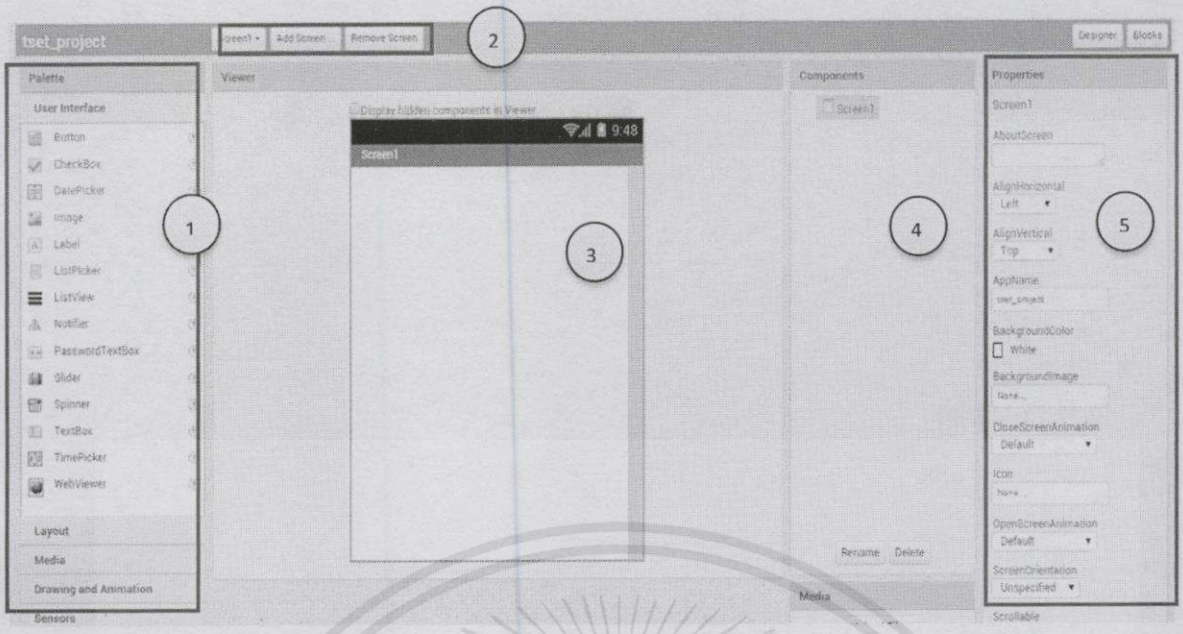
ฉะนั้นจึงกล่าวได้ว่า MIT App Inventor เป็นเครื่องมือที่ใช้สำหรับการสร้างแอพพลิเคชั่นบนระบบปฏิบัติการแอนดรอยด์ ที่ใช้หลักการของการพัฒนาซอฟต์แวร์เชิงคอมโพเน็นท์ (Component Based Software Development) ในรูปแบบของการเขียนโปรแกรมให้ผู้ที่ใช้โปรแกรมสำเร็จ สามารถกำหนดรายการคำสั่ง (Menu) ต่างๆ ได้เอง (Visual Programming) นอกจากนั้นยังใช้หลักการการประมวลผลแบบผู้ให้บริการ และผู้ใช้งาน (Client/Server) ทำให้เครื่องที่ใช้ในการสร้างแอพพลิเคชั่นไม่ต้องติดตั้งโปรแกรมเพิ่มเติม ทำให้สะดวกในการใช้งาน โดยจะมีข้อจำกัดอยู่บ้างตรงที่แพลตฟอร์มสำหรับพัฒนาซอฟต์แวร์ (Framework) ที่มี ไม่ครอบคลุมขนาดจอภาพของอุปกรณ์ที่หลากหลาย และตลอดเวลาที่ใช้งานเครื่องมือนี้จะต้องเชื่อมต่ออินเตอร์เน็ตอยู่ตลอดเวลา

2.6 โครงสร้างพื้นฐานการออกแบบแอพพลิเคชั่น

การออกแบบแอพพลิเคชั่นด้วยโปรแกรม App Inventor (Create Project on App Inventor) ช่วยให้สามารถพัฒนาแอพพลิเคชั่นสำหรับโทรศัพท์ระบบปฏิบัติการแอนดรอยด์ ซึ่งทำผ่านการใช้เว็บบราวเซอร์ และทดสอบบนโทรศัพท์ที่เชื่อมต่ออยู่กับคอมพิวเตอร์ หรือบนโทรศัพท์จำลองในเครื่องคอมพิวเตอร์ โครงงานที่สร้างทั้งหมดจะถูกจัดเก็บไว้บนเซิร์ฟเวอร์ App Inventor ซึ่งช่วยให้สามารถพัฒนางานต่อที่เครื่องคอมพิวเตอร์เครื่องใดก็ได้ เพียงแค่ได้มีการเชื่อมต่อกับระบบอินเตอร์เน็ตไว้เท่านั้น

การสร้างแอพพลิเคชั่นจะแบ่งการทำงานออกเป็น 2 ส่วน คือ

1. ส่วนการออกแบบ (App Inventor Designer) ที่จะให้เราเลือกคอมโพเนนท์ที่ต้องการสำหรับที่จะให้สร้างแอพพลิเคชั่น ดังรูปที่ 2.22

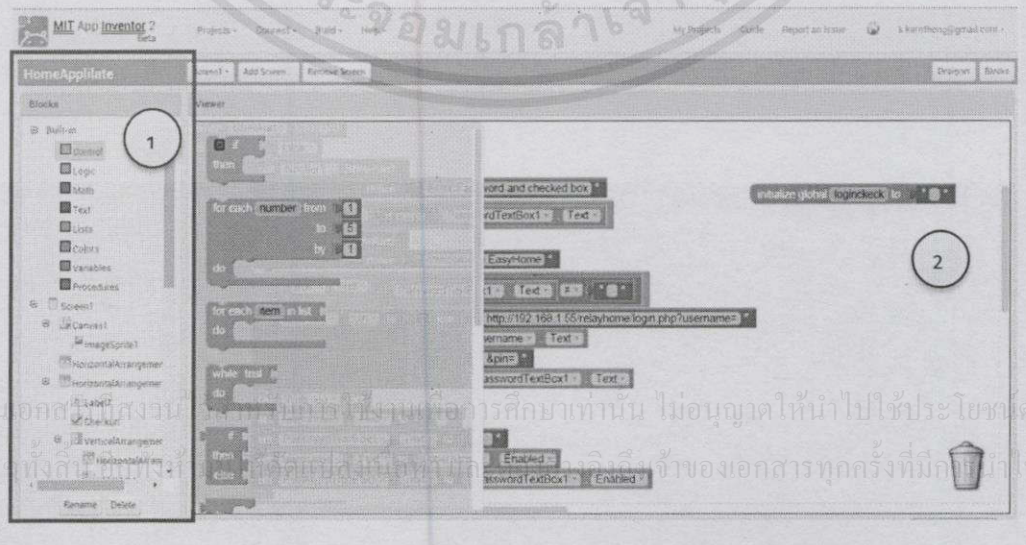


รูปที่ 2.22 โครงสร้างส่วนการออกแบบ

คำอธิบายหน้าจอ

1. กลุ่มเครื่องมือสำหรับออกแบบหน้าจอแอพพลิเคชั่น
2. กลุ่มแถบเมนู ที่ใช้ในจัดการ และเชื่อมต่อกับแอพพลิเคชั่น
3. หน้าจอการออกแบบ (Viewer)
4. หน้าจอส่วนคอมโพเนนท์ (Components) ที่เลือกนำมาใช้ในโปรเจค
5. หน้าจอส่วนคุณสมบัติของคอมโพเนนท์ (Properties)

2. ส่วนการเขียนโค้ด (App Inventor Blocks Editor) เป็นการเขียนโค้ดด้วยการต่อบล็อกต่างๆ เข้าด้วยกันเป็นคำสั่ง ซึ่งจะเป็นการกำหนดพฤติกรรม หรือเหตุการณ์ที่เกิดขึ้นกับคอมโพเนนท์ ดังรูปที่ 2.23



รูปที่ 2.23 โครงสร้างส่วนการเขียนโค้ด

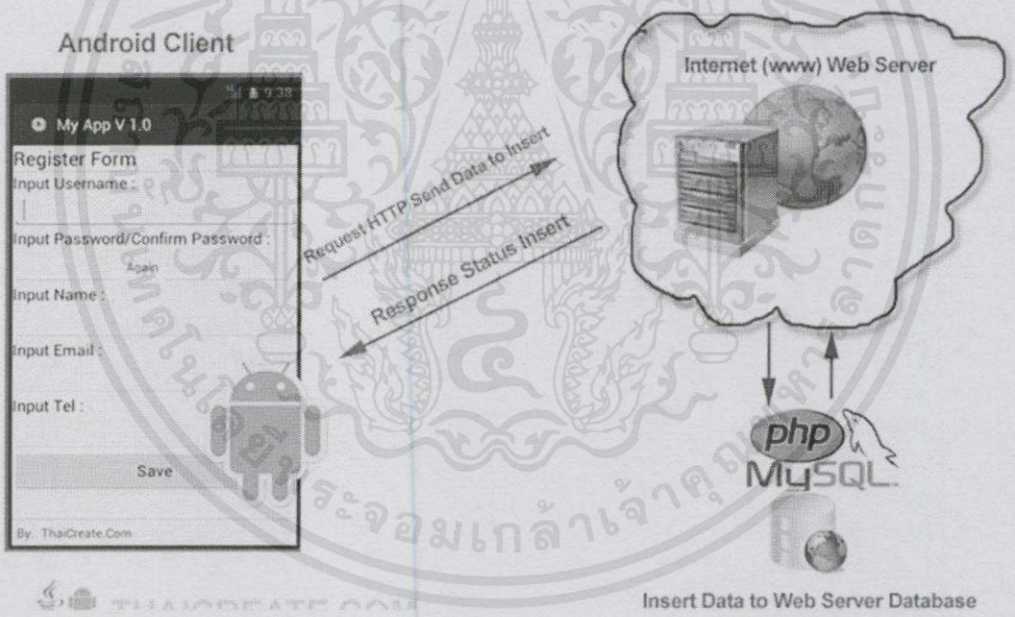
คำอธิบายหน้าจอ

1. กลุ่มสัญลักษณ์บล็อกที่ใช้ในการควบคุมแอพพลิเคชั่น
2. พื้นที่ทำงานใช้สำหรับวางปุ่มควบคุม

ในการพัฒนาด้วยโปรแกรม App Inventor นั้น สนับสนุนระบบปฏิบัติการที่หลากหลาย ไม่ว่าจะเป็นระบบปฏิบัติการบนคอมพิวเตอร์ (Windows), ระบบปฏิบัติการการลบข้อมูลอย่างปลอดภัย (Mac OS X, GNU/Linux) และแอพพลิเคชั่นที่สร้างขึ้นนั้น สามารถติดตั้ง และทำงานได้บนโทรศัพท์ระบบปฏิบัติการแอนดรอยด์ หลากหลายรุ่นที่เป็นที่นิยมในปัจจุบัน

2.7 แอนดรอยด์กับระบบฐานข้อมูล_(Android to Server Database)

การเขียนแอนดรอยด์ เพื่อส่งข้อมูลบนฟอร์มไปบันทึกที่ระบบฐานข้อมูล โดยในเว็บเซิร์ฟเวอร์ จะใช้ PHP กับ MySQL ทำหน้าที่รอรับข้อมูลที่ส่งด้วยแอนดรอยด์ และเมื่อรับข้อมูลที่แอนดรอยด์ส่งมาแล้ว ก็จะทำการเพิ่ม (Insert) ลงในฐานข้อมูล MySQL พร้อมๆ กับส่งสถานะแจ้งมายังผู้ใช้ (Client) ให้ทราบว่าการเพิ่มข้อมูลสมบูรณ์ หรือว่าผิดพลาด แสดงดังรูปที่ 2.24



รูปที่ 2.24 โครงสร้างการทำงานระหว่างแอนดรอยด์กับระบบฐานข้อมูล

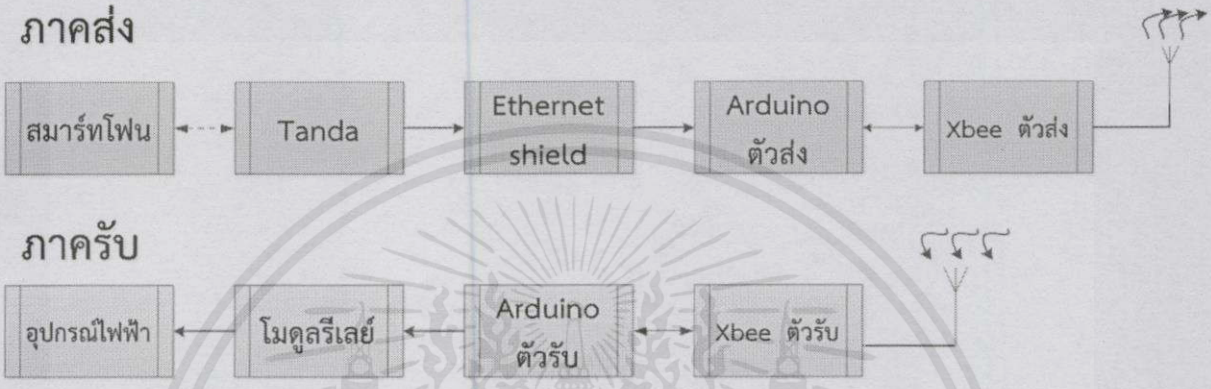
จากภาพอธิบายจะเห็นว่าแอนดรอยด์จะส่งข้อมูลผ่าน (HTTP) เพื่อจะไปเพิ่ม หรือบันทึกที่ฝั่งของเว็บเซิร์ฟเวอร์โดยในเว็บเซิร์ฟเวอร์ ใช้แอพพลิเคชั่นของ PHP ซึ่งในการเขียนแอนดรอยด์ เพื่อติดต่อกับอินเตอร์เน็ต จะต้องกำหนดตัวแปร (Permission) ในส่วนนี้ด้วยทุกครั้ง

บทที่ 3

การออกแบบ และโครงสร้าง

3.1 หลักการทำงานของระบบ

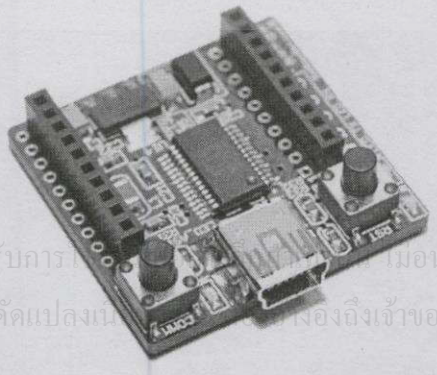
สามารถแสดงโครงสร้างการทำงานทั้งหมดของระบบได้ ดังรูปที่ 3.1



รูปที่ 3.1 บล็อกไดอะแกรมแสดงโดยรวมของระบบ

3.2 การตั้งค่าให้กับโมดูล XBee

ในการตั้งค่า XBee นั้น จำเป็นต้องมีโปรแกรม X-CTU และ Mini XBee USB Dongle V2 ไว้เชื่อมต่อโมดูล XBee กับพอร์ต USB ของคอมพิวเตอร์ เพื่อใช้ในการอัพเดทเฟิร์มแวร์, การกำหนดค่าพารามิเตอร์ (Config Parameter) หรือการอัพเดทเฟิร์มแวร์ผ่านคอมพิวเตอร์ ร่วมกับโปรแกรม X-CTU ซึ่งคอมพิวเตอร์จะมองเป็นพอร์ตคอมพิวเตอร์ (Serial UART) เนื่องจากใช้ IC FT232RL ซึ่งเป็นพอร์ต USB และพอร์ตคอมพิวเตอร์ (USB to Serial) ในบอร์ดเดียว ไม่ต้องต่ออุปกรณ์ใดๆ เพิ่มเติม สามารถใช้ร่วมกับ XBee ได้ทุกรุ่น ซึ่งในการทดลองใช้ Mini XBee USB Dongle V2 แสดงดังรูปที่ 3.2

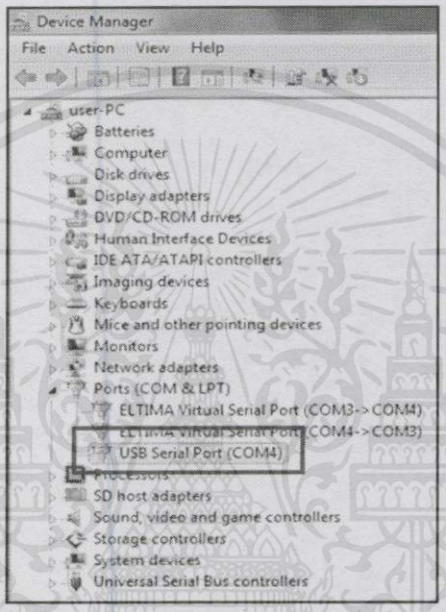


รูปที่ 3.2 Mini XBee USB Dongle V2

3.2.1 การตั้งค่าให้กับ XBee ฝั่งส่ง (Coordinator)

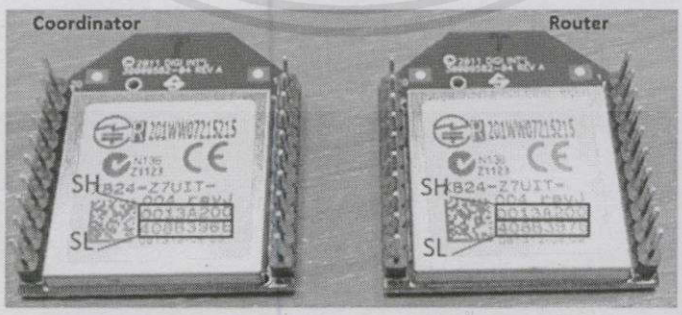
1. หลังจากติดตั้งโปรแกรม X-CTU เรียบร้อยแล้ว จะทำการติดตั้งไดร์เวอร์ Mini XBee USB Dongle โดยเสียบสาย USB เข้ากับบอร์ด และคอมพิวเตอร์ เมื่อเข้าไปที่การจัดการอุปกรณ์ (Device Manager) → Other Device คลิกขวา ที่ USB Serial Port → Update Driver Software จากนั้นเลือกที่อยู่ไฟล์ไดร์เวอร์ (Driver) ในข้อ 2 แสดงดังรูปที่ 3.3

หลังจากติดตั้งไดร์เวอร์เสร็จเรียบร้อยแล้ว คอมพิวเตอร์จะเห็นบอร์ดเป็น “USB Serial Port” หนึ่งพอร์ตในระบบ



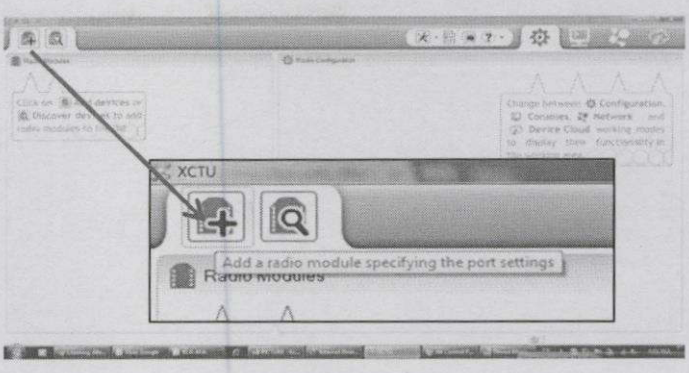
รูปที่ 3.3 หน้าจอคอมพิวเตอร์แสดงผลบอร์ดเป็น “USB Serial Port”

2. ตรวจสอบหมายเลขประจำอุปกรณ์ (Serial Number) โดยใน XBee ทุกตัวจะมีหมายเลขประจำอุปกรณ์อยู่ แสดงดังรูปที่ 3.4 มีตัวเลขอยู่ 2 ชุด คือ SH (Serial Number High) และ SL (Serial Number Low) ใช้ค่านี้เพื่อกำหนดให้ XBee ทั้งสองติดต่อกัน



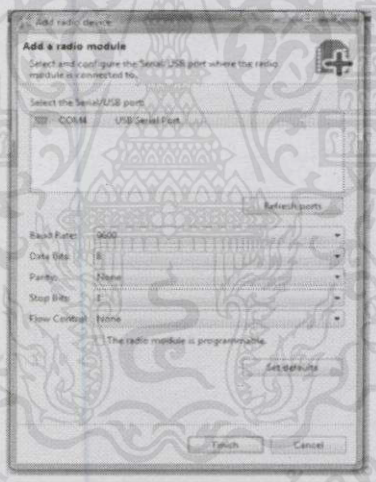
รูปที่ 3.4 หมายเลขประจำอุปกรณ์ XBee

3. เปิดโปรแกรม XCTU กดปุ่ม “Add a Radio Module” ดังรูปที่ 3.5



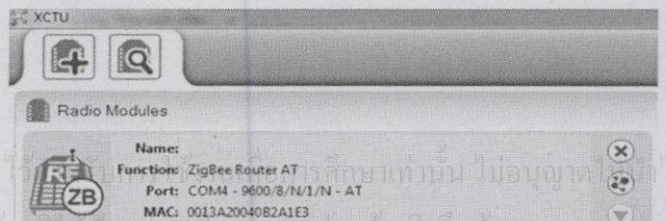
รูปที่ 3.5 การเพิ่มโมดูลในโปรแกรม XCTU

4. เลือกหมายเลขพอร์ตอนุกรมของ Mini XBee USB Dongle กำหนดอัตราความเร็วในการส่งสัญญาณ (Baud Rate) เป็น 9600 bps Data Bits เท่ากับ 8 Parity เป็น “None Stop Bits” เท่ากับ 1 และการควบคุมการส่งข้อมูล (Flow Control) เป็น “None” จากนั้นกดปุ่ม “Finish” แสดงดังรูปที่ 3.6



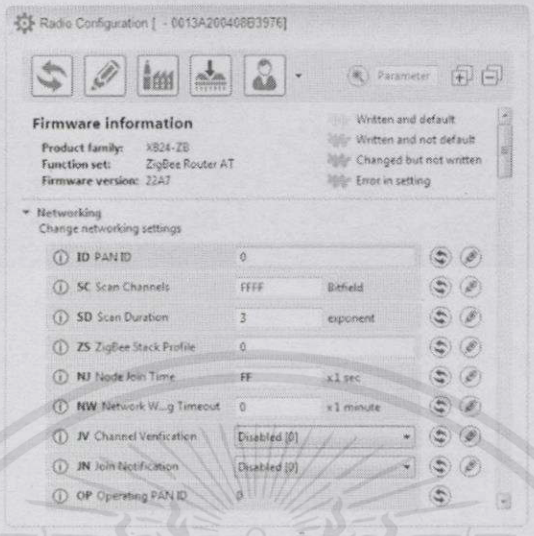
รูปที่ 3.6 หน้าต่างแสดงให้ใส่ค่าตามที่กำหนด

5. จากนั้นโปรแกรมจะแสดงชื่อรุ่นของ XBee ขึ้นมา ดังรูปที่ 3.7 ให้กดที่โมดูลเพื่อดูรายละเอียดพารามิเตอร์ต่างๆ ของโมดูล



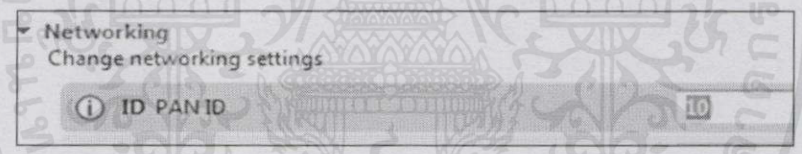
รูปที่ 3.7 หน้าต่างแสดงชื่อรุ่น XBee

6. จากนั้นที่ด้านขวาของหน้าต่างโปรแกรมในหัวข้อ “Radio Configuration” จะแสดงค่าพารามิเตอร์ของ XBee ดังรูปที่ 3.8



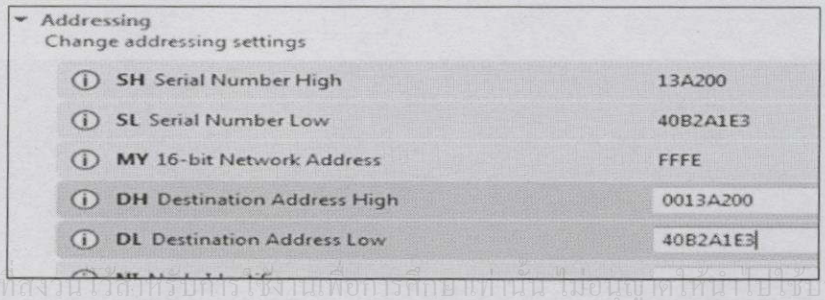
รูปที่ 3.8 หน้าต่างแสดงค่าพารามิเตอร์ของ XBee

7. ไปที่หัวข้อ “Networking” → “PAN ID” (Personal Area Network Identifier) แสดงหมายเลขไอดีของเน็ตเวิร์กที่ XBee ใช้ ในตัวอย่างนี้กำหนดเป็น 10 ดังรูปที่ 3.9



รูปที่ 3.9 กำหนดค่าไอดีของเน็ตเวิร์กที่ XBee ใช้ตามที่กำหนด

8. ไปที่หัวข้อ “Addressing” → DH Destination Address High และ DL Destination Address Low กำหนด เลขประจำอุปกรณ์ของ XBee ที่ต้องการติดต่อในตัวอย่างนี้ กำหนดเป็น DH เป็น 13A200 และ DL เป็น 408B396E ดังรูปที่ 3.10

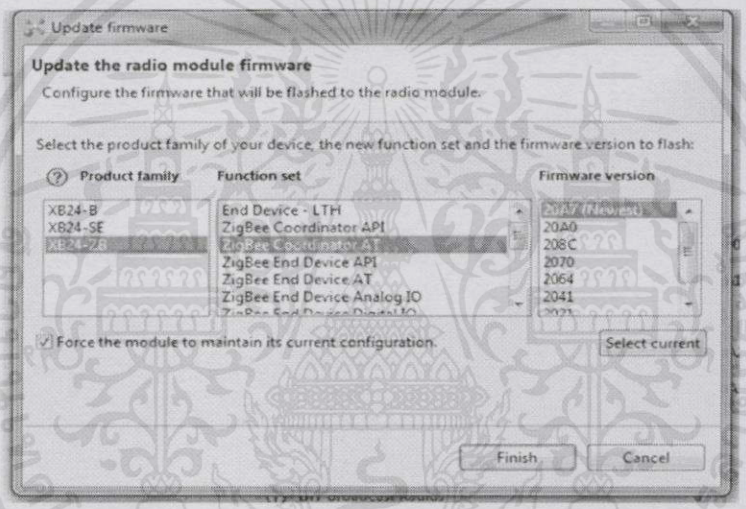


รูปที่ 3.10 กำหนดค่า DH และ DL ของ XBee

9. กดปุ่ม “Update Firmware” ดังรูปที่ 3.11

รูปที่ 3.11 กดปุ่มเพื่ออัพเดทเฟิร์มแวร์

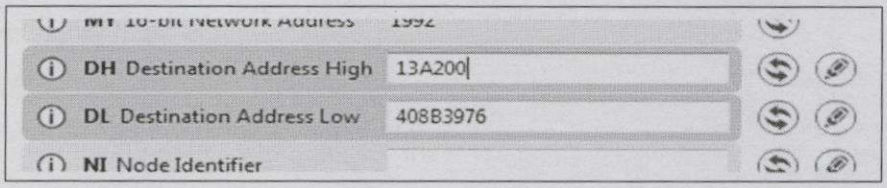
10. เลือกรุ่น (Product Family) ของ XBee เป็น XB24-ZB ที่ช่อง “Function Set” กำหนดให้ XBee เป็นแม่ข่ายโหมดการสื่อสารแบบ AT และเลือกเวอร์ชั่นเป็นรุ่นล่าสุด (20A7 Newest) ดังรูปที่ 3.12 แล้วกดปุ่ม “Finish”



รูปที่ 3.12 เลือกรุ่น โหมดการสื่อสาร และเวอร์ชั่นของ XBee

3.2.2 การตั้งค่าให้กับ XBee ฝั่งรับ (เราเตอร์ XBee/อุปรณ์ปลายทาง)

1. ในหัวข้อ “Addressing” ส่วนของ DH และ DL กำหนดเลขประจำอุปกรณ์ของแม่ข่ายลงไป กำหนดเป็น DH เป็น 13A200 และ DL เป็น 408B3976 ดังรูปที่ 3.13

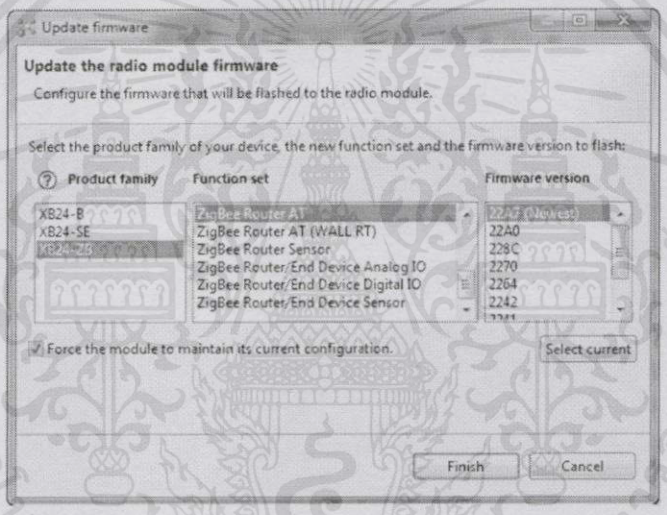


รูปที่ 3.13 กำหนดค่า DH และ DL ของ XBee

2. กดปุ่ม “Update Firmware” ดังรูปที่ 3.14

รูปที่ 3.14 กดปุ่มเพื่ออัพเดทเฟิร์มแวร์

3. เลือกรุ่น (Product Family) ของ XBee เป็น XB24-ZB ที่ช่อง “Function Set” กำหนดให้ XBee เป็นเราเตอร์โหมดการสื่อสารแบบ AT และเลือกเฟิร์มแวร์เวอร์ชั่นเป็นรุ่นล่าสุด (22A7-Newest) ดังรูปที่3.15

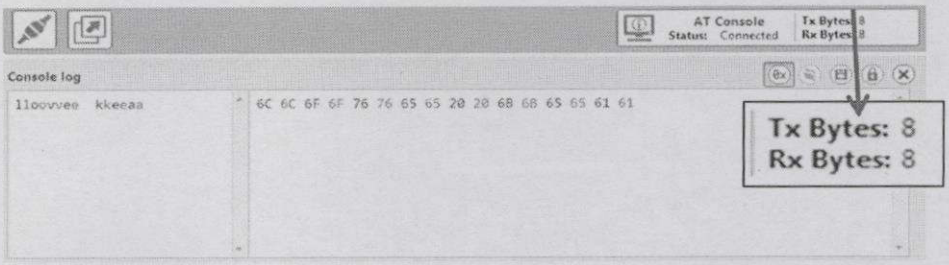


รูปที่ 3.15 รุ่น, โหมดการสื่อสาร และเวอร์ชั่นของ XBee

3.2.3 ทดสอบการเชื่อมต่อเบื้องต้นระหว่าง XBee ตัวส่ง และตัวรับ

ถ้าการเซต XBee ไม่มีข้อผิดพลาด แม่ข่ายก็จะสามารถส่งข้อมูลไปยังเราเตอร์ได้ และในทางกลับกันเราเตอร์สามารถส่งข้อมูลไปยังแม่ข่ายได้

การทดสอบเชื่อมต่อแม่ข่ายเข้ากับคอมพิวเตอร์ โดยส่งข้อมูลผ่านโปรแกรม XCTU ที่ฝั่งเราเตอร์ จั้มขา Tx กับ Rx ของโมดูลเข้าด้วยกัน หมายความว่าข้อมูลทั้งหมดที่ส่งจากแม่ข่าย จะถูกส่งกลับไปยังแม่ข่าย เป็นการใช้โปรแกรม XCTU ส่งข้อมูลเข้าไปที่แม่ข่าย จากรูปที่ 3.16 จะเห็นได้ว่าข้อมูลทั้งรับ และส่งเป็นข้อมูลเดียวกัน

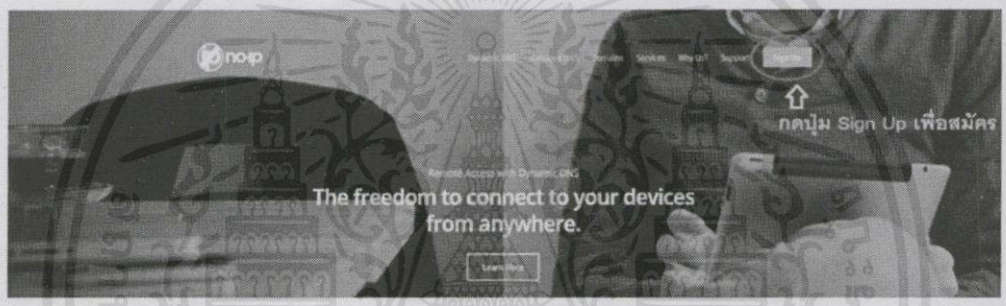


รูปที่ 3.16 หน้าจอแสดงผลข้อมูลทางด้านส่ง และด้านรับ

3.3 วิธีการสมัครใช้งาน NO-IP (DynDNS) และโฮสเนม (Add Hostname)

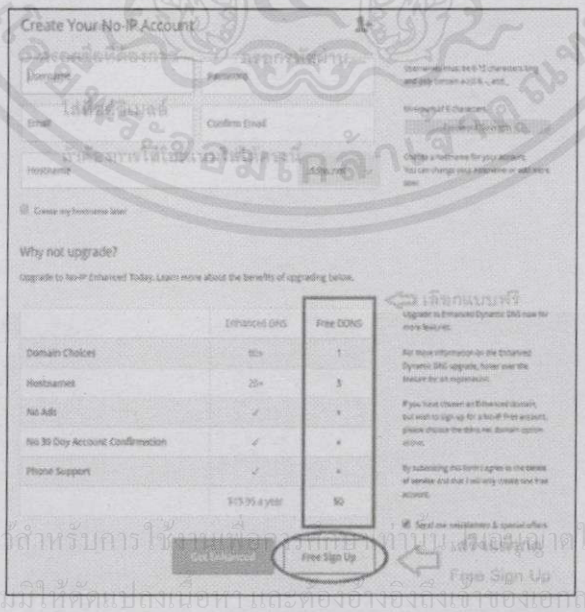
1. เริ่มต้นด้วยการเข้าเว็บ แล้วพิมพ์ www.no-ip.com ดังรูปที่ 3.17

2. เมื่อเข้ามาที่หน้า index ของ www.no-ip.com แล้วให้กดปุ่มสีเขียว “Sign Up” เพื่อทำการสมัครไอดี



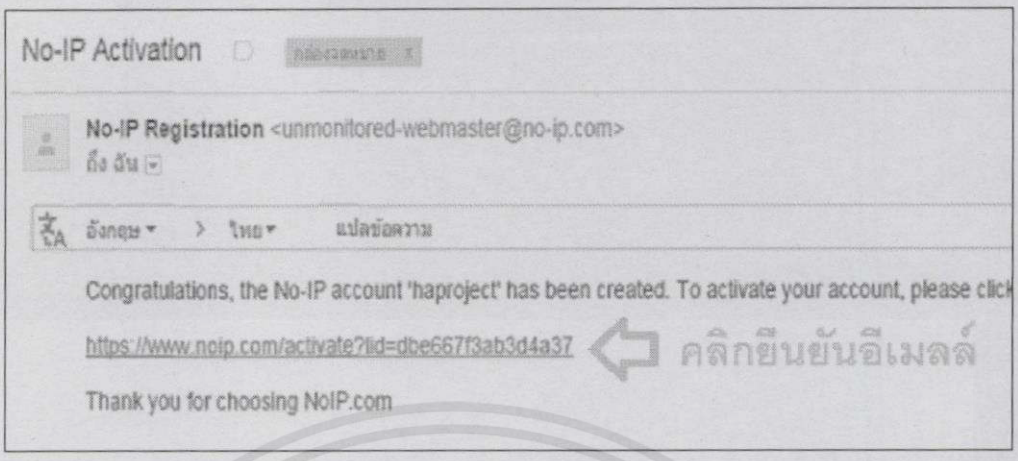
รูปที่ 3.17 หน้าจอเว็บเริ่มแรก

3. กรอกรายละเอียดต่างๆ ดังรูปที่ 3.18



รูปที่ 3.18 กรอกรายละเอียด

4. ทำการเข้าอีเมล์ที่ได้ กรอกรายละเอียดสมัครไว้ โดยการกดลิงค์เพื่อยืนยัน ดังรูปที่ 3.19

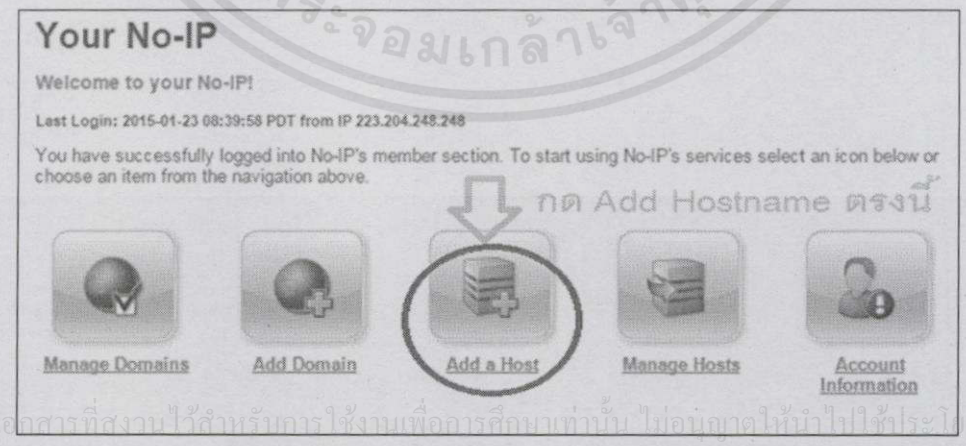


รูปที่ 3.19 ลิงค์เพื่อยืนยันอีเมล์

5. หลังจากกดลิงค์ที่ทาง no-ip ส่งไปที่อีเมล์ จะเข้าสู่หน้าล็อกอิน เพื่อกรอกชื่อผู้ใช้งาน และรหัสผ่าน ดังรูปที่ 3.20

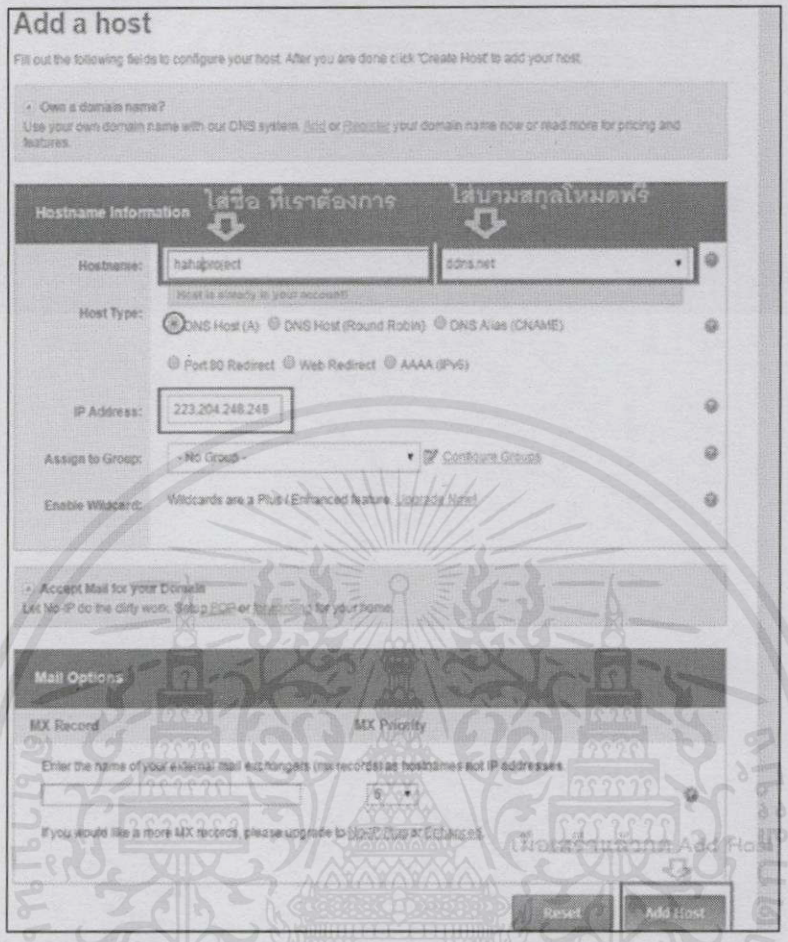
รูปที่ 3.20 หน้าล็อกอินของระบบ

6. ขั้นตอนเพิ่มชื่อโฮสเนม (Hostname) ไว้ใช้ดูออนไลน์ ดังรูปที่ 3.21



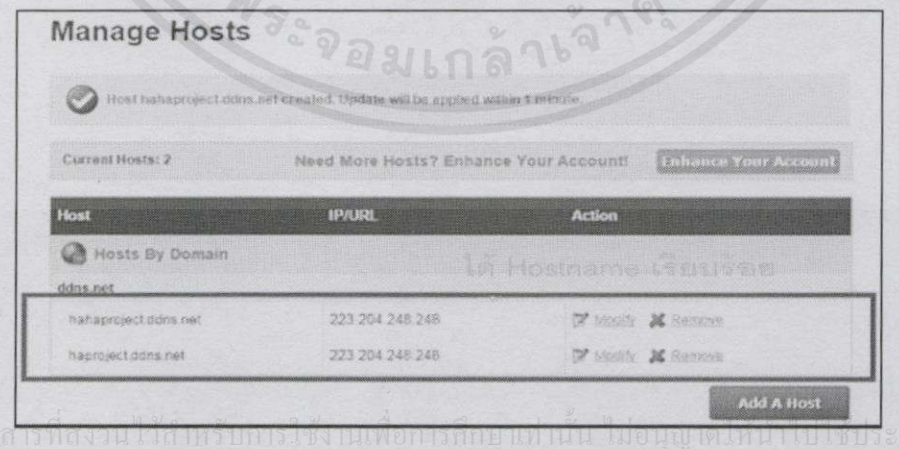
รูปที่ 3.21 กด “Add Hostname” เพื่อใช้ดูออนไลน์

7. หลังจากกดปุ่ม “Add a Host” แล้ว จะเข้าสู่หน้า “Add Hostname” ดังรูปที่ 3.22



รูปที่ 3.22 กรอกข้อมูลเมื่อเข้าสู่หน้าโฮสเนม

8. เสร็จกระบวนการสมัคร ดังรูปที่ 3.23 ได้ “โฮสเนม” เรียบร้อยแล้ว



รูปที่ 3.23 เสร็จกระบวนการการสมัครโฮสเนม

3.4 การตั้งค่าเราเตอร์ตัวหลัก (TP-Link TD-W8961ND)

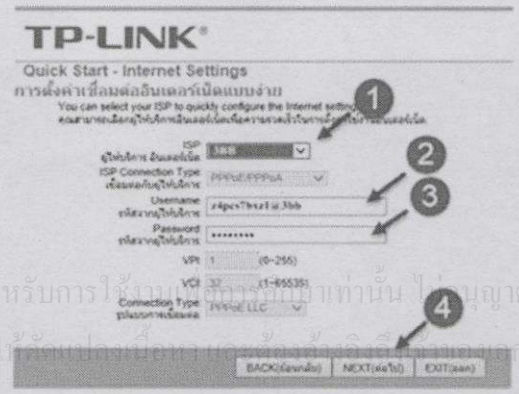
1. การตั้งค่าสำหรับเชื่อมต่ออินเตอร์เน็ตของ 3BB (Tipple Broad Band) ดังรูปที่ 3.24

- เชื่อมต่อ TD-W8961ND ด้วยสายแลน
- เปิดบราวเซอร์ พิมพ์ 192.168.1.1 กด “Enter”
- พิมพ์ ชื่อผู้ใช้งาน = admin
- พิมพ์ รหัสผ่าน = admin
- กดปุ่ม “Login”

รูปที่ 3.24 หน้าจอการเชื่อมต่ออินเตอร์เน็ต

2. ตั้งค่าการเชื่อมต่อ (เตรียม ชื่อผู้ใช้งาน/รหัสผ่านให้พร้อม) โดยปกติจะได้มาตอนขอบริการติดตั้งอินเตอร์เน็ต) แสดงดังรูปที่ 3.25

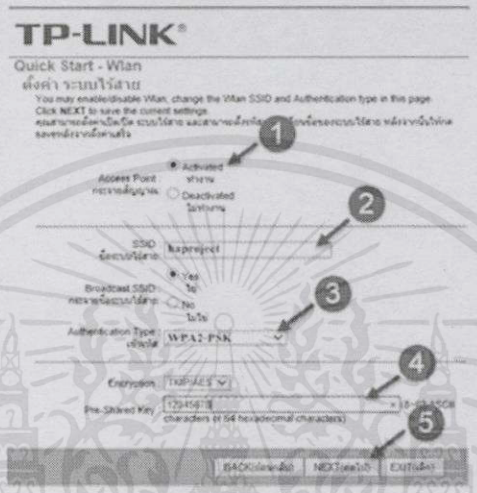
- เลือกว่าใช้อินเตอร์เน็ตของอะไร TOT, TRUE หรือ 3BB
- พิมพ์ ชื่อผู้ใช้งาน
- พิมพ์ รหัสผ่าน
- กดปุ่ม “NEXT”



รูปที่ 3.25 ตั้งค่าเชื่อมต่ออินเตอร์เน็ต

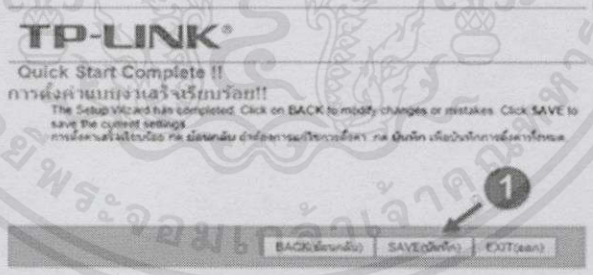
3. ตั้งค่า Wifi แสดงดังรูปที่ 3.26

- กด “Activated” เพื่อเปิดสัญญาณ Wifi
- พิมพ์ชื่อ Wifi ในช่อง SSID
- เลือก “Authentication Type = WPA2-PSK”
- พิมพ์รหัสผ่านในช่อง Pre-Shared Key (ตัวเลข 8 หลัก)
- กดปุ่ม “NEXT”



รูปที่ 3.26 ตั้งค่า Wifi

4. กดปุ่ม “SAVE” ดังรูปที่ 3.27 แล้วรอสักครู่ จะสามารถใช้งานอินเตอร์เน็ตได้



รูปที่ 3.27 กดบันทึกเพื่อใช้งานอินเตอร์เน็ต

3.5 การตั้งค่าเราเตอร์ตัวรับ หรือตัวทวนสัญญาณ (Tenda)

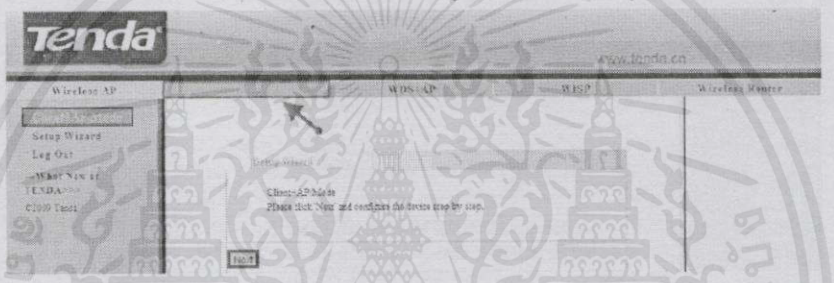
ในการตั้งค่าเราเตอร์ตัวรับนี้ จะใช้โหมดในการตั้งค่าเป็นแบบ “Client + AP Mode” เพื่อทำหน้าที่ทวนสัญญาณจากเราเตอร์ตัวหลัก

3.5.1 ข้อแนะนำก่อนทำการตั้งค่า "Client + AP Mode" (Repeater)

ก่อนที่จะทำการตั้งโหมดเป็น Client + AP หรือทวนสัญญาณนั้น แนะนำให้ทำการตรวจสอบไอพีแอดเดรส (IP Address) ของเน็ตเวิร์กที่จะทำการทวนสัญญาณด้วยว่าเป็นพารามิเตอร์ที่ต้องระบุคู่กับไอพีแอดเดรส (Subnet)[6] 192.168.2.0/24 หรือใช้ไอพีแอดเดรสช่วง 192.168.2.1 ถึงช่วง 192.168.2.254 หรือไม่ ถ้าใช่ต้องทำการเปลี่ยนไอพีแอดเดรสของ W150M ให้เป็นพารามิเตอร์อื่น เช่น 192.168.0.0/24 หรือ 192.168.1.0/24

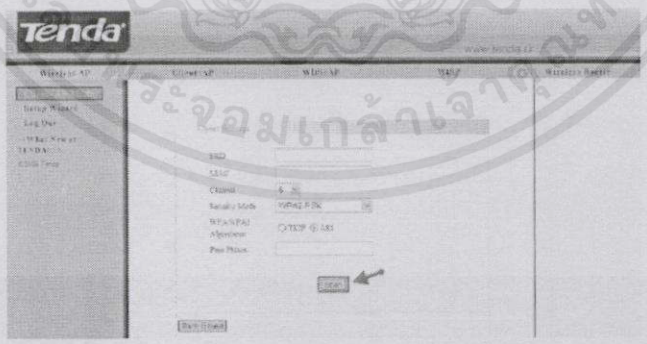
3.5.2 กรณีที่เน็ตเวิร์กวงที่จะทำการทวนสัญญาณไม่ได้ ใช้พารามิเตอร์ที่ต้องระบุคู่กับไอพีแอดเดรส 192.168.2.0/24

1. กดที่เมนูหลัก "Client + AP Mode" ในครั้งแรกจะเข้าสู่หน้า "Setup Wizard" สำหรับช่วยตั้งค่า ให้กดปุ่ม "Next" เพื่อเข้าสู่การตั้งค่าต่อไป ดังรูปที่ 3.28



รูปที่ 3.28 หน้าหลักเพื่อเข้าสู่การตั้งค่า Tenda

2. จะเข้าสู่หน้า "Client Settings" ซึ่งเป็นหน้าสำหรับตั้งค่าเพื่อทำการจับกับแอคเซสพอยต์ (Access Point)[7] ตัวหลัก โดยหลังจากเข้ามาครั้งแรกจะยังเป็นค่าเปล่าๆ ให้กดปุ่ม "Scan" เพื่อทำการค้นหาแลนไร้สาย (Wireless) ที่ต้องการจับ ดังรูปที่ 3.29

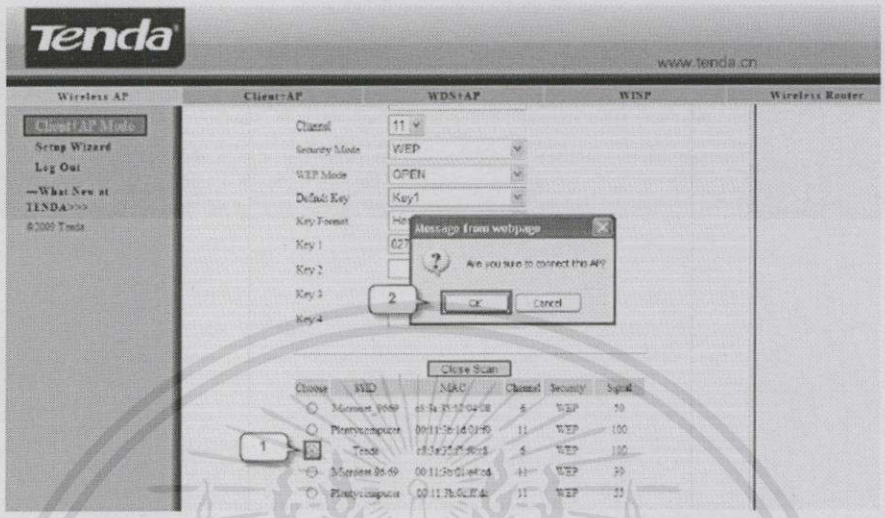


รูปที่ 3.29 กดปุ่ม "Scan" เพื่อค้นหาแลนไร้สาย

[6] Subnet คือตัวที่จะช่วยทำให้เราสามารถระบุได้ว่า ไอพีแอดเดรสเบอร์นั้นๆ ว่ามีบิตไหนบ้างที่เป็นไอดีเน็ตเวิร์ก และมีบิตไหนบ้างที่เป็นไอดีโฮส (IP address AND Subnet Mask = Network ID)

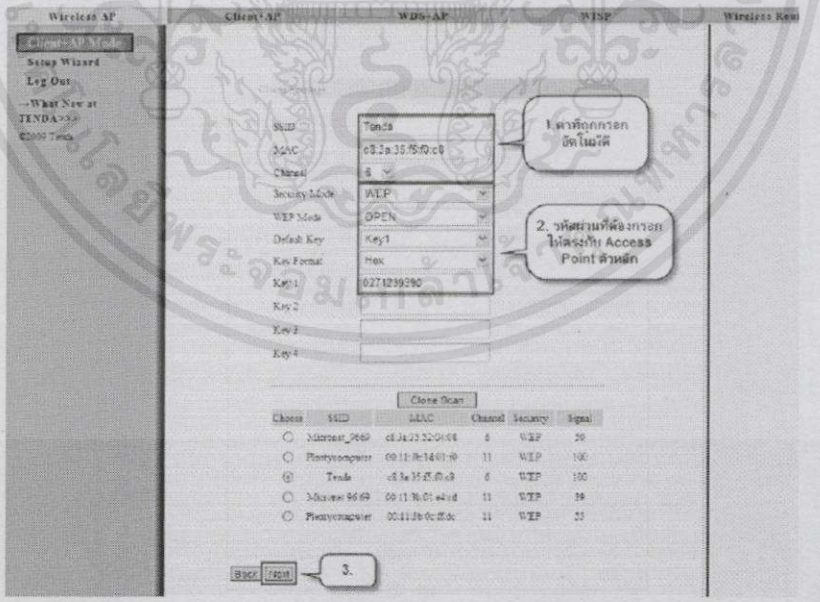
[7] Access Point คือ อุปกรณ์ในเครือข่ายคอมพิวเตอร์ ที่ช่วยให้อุปกรณ์ไร้สายสามารถเชื่อมต่อกับเครือข่ายแบบมีสายได้โดยการใช้เทคโนโลยีของแลนไร้สาย

3. ต่อไปให้กดตรงช่อง “Choose” ของแอคเซสพอยต์ที่ต้องการเชื่อมต่อ จะมีกรอบขึ้นมาให้ยืนยันการเชื่อมต่อกับแอคเซสพอยต์ดังกล่าว ว่าต้องการหรือไม่ ให้กดตกลง (OK) เพื่อยืนยัน ดังรูปที่ 3.30 หรือกดยกเลิก (Cancel) เพื่อเลือกแอคเซสพอยต์อื่น



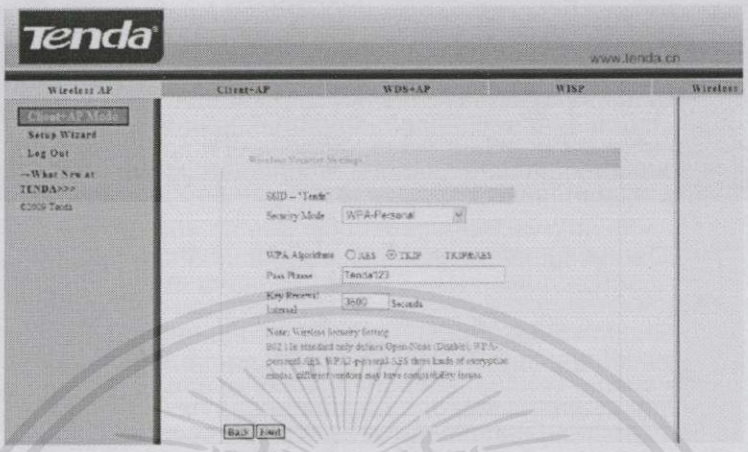
รูปที่ 3.30 เลือกแอคเซสพอยต์ที่ต้องการ

4. ค่า SSID, MAC และช่องของแอคเซสพอยต์ที่เลือกไว้ จะขึ้นไปแสดงตรงช่อง (ส่วนที่ 1) โดยอัตโนมัติ (ส่วนที่ 2) เป็นส่วนที่ผู้ใช้ต้องกรอกรหัสผ่าน, เลือก “Key” และเลือกโหมดให้ถูกต้องตรงกับแอคเซสพอยต์ตัวหลักที่ต้องการจับ เสร็จแล้วกดปุ่ม “Next” (ส่วนที่ 3) ดังรูปที่ 3.31



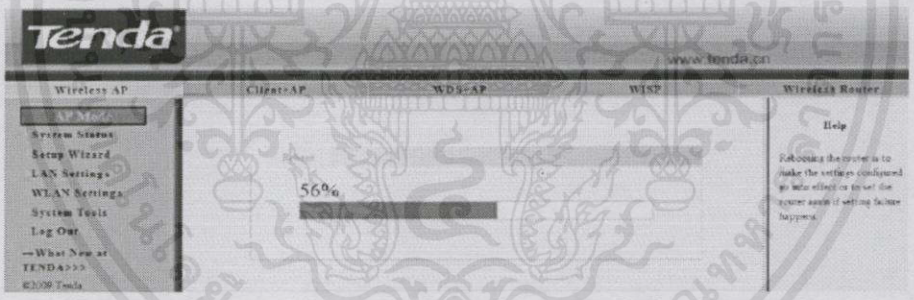
รูปที่ 3.31 การกำหนดค่าให้ตรงกับแอคเซสพอยต์ที่ต้องการ

5. การตั้งค่าความปลอดภัย (Security Wireless) สำหรับโหมด AP โดยที่ค่าความปลอดภัยดังกล่าวไม่จำเป็นต้องตั้งค่าเหมือนกับแอคเซสพอยต์ตัวหลักก็ได้ สามารถตั้งค่าความปลอดภัยแตกต่างกันได้ เสร็จแล้วให้กดปุ่ม “NEXT” ดังรูปที่ 3.32



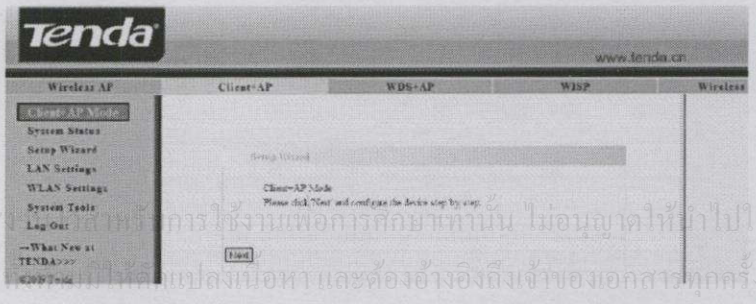
รูปที่ 3.32 การตั้งค่าความปลอดภัย

6. รอจนแถบน้ำเงิน ดังรูปที่ 3.33 แล้วเปอร์เซนต์วิ่งจนถึง 100% เพื่อแสดงว่าอุปกรณ์ได้ทำการคืนค่าเดิม (Reboot) เสร็จเรียบร้อยแล้ว



รูปที่ 3.33 หน้าต่างแสดงการอัพโหลด เพื่อแสดงสถานะการคืนค่าเดิม

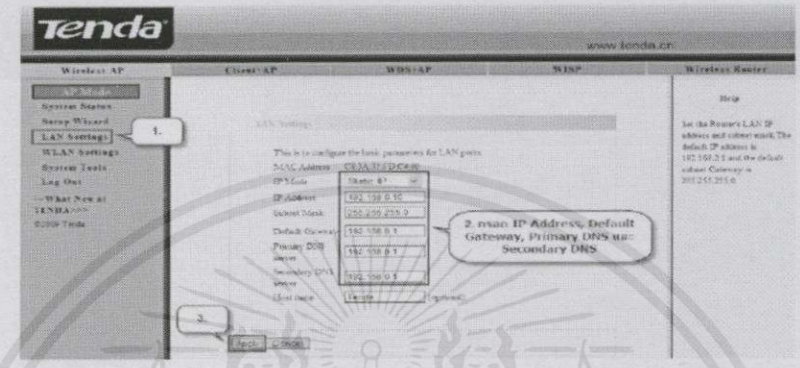
7. หลังจากทำการคืนค่ากลับเข้ามา ดังรูปที่ 3.34 ก็จะเห็นว่าสถานะที่ไฟ LED เปลี่ยนจาก “AP” เป็น “Client + AP” ให้แบบอัตโนมัติ



รูปที่ 3.34 หน้าจอแสดงสถานะไฟ LED

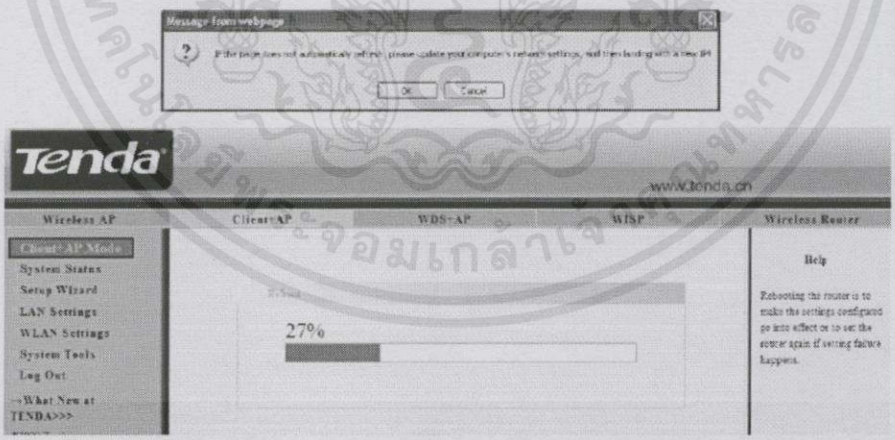
3.5.3 กรณีที่เน็ตเวิร์กวงที่จะทำการทวนสัญญาณ ใช้พารามิเตอร์ที่ต้องระบุคู่กับไอพีแอดเดรส 192.168.2.0/24 เดียวกันกับ W150M

1. ทำการล็อกอินเข้าหน้า W150M ด้วยโหมด AP เพื่อเข้าไปตั้งค่าไอพีแอดเดรสจากเมนูด้านซ้ายมือตรง “LAN Settings” แล้วทำการเปลี่ยนค่า “IP Address[8], Default Gateway, Primary DNS และ Security DNS” ตามรายละเอียด ดังรูปที่ 3.35



รูปที่ 3.35 การตั้งค่าเปลี่ยนไอพีแอดเดรส

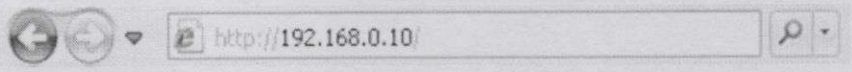
2. หลังจากนั้นจะขึ้นหน้าต่างแจ้ง ให้ทำการเปลี่ยนไอพีของเครื่องคอมพิวเตอร์ให้ตรงกับพารามิเตอร์ใหม่ (โดยอาจจะตั้งค่ากำหนดไอพีเครื่องคอมพิวเตอร์เป็น 192.168.0.2 เป็นต้น) เสร็จแล้วเข้าหน้า W150M ด้วยไอพีใหม่อีกครั้ง ให้กดปุ่ม “OK” เพื่อยืนยัน และรอจนครบ 100% ดังรูปที่ 3.36



รูปที่ 3.36 หน้าต่างแสดงการอัพโหลดไอพีแอดเดรส

[8] IP Address คือเลขรหัสประจำคอมพิวเตอร์ที่ต่ออยู่บนเครือข่าย ซึ่งประกอบด้วยตัวเลข 4 ชุด และมีเครื่องหมายจุดขั้นระหว่างชุด เช่น 192.168.1.1 เป็นต้น

3. จากนั้นเข้าหน้าโครงสร้าง (Config) ของอุปกรณ์ด้วยไอพีแอดเดรสใหม่ โดยพิมพ์ตรงแอดเดรสบาร์ (Address Bar) ดังรูปที่ 3.37 แล้วย้อนกลับไปทำตามข้อ 3.6.2.



รูปที่ 3.37 ไอพีแอดเดรสใหม่ของอุปกรณ์

3.6 หลักการออกแบบแอพพลิเคชั่น

ในส่วนของการออกแบบแอพพลิเคชั่นประกอบไปด้วย 3 ส่วนหลักๆ คือ

1. ส่วนของหน้าการทำงานที่เกี่ยวข้องกับการจัดทำระบบฐานของมูลของผู้ใช้งาน

1.1 หน้าล็อกอิน (Login) เพื่อเข้าใช้งาน เป็นหน้าเริ่มแรกของแอพพลิเคชั่น ที่ผู้ใช้งานจำเป็นต้องมีรหัสผ่าน (Password) เพื่อเข้าถึงแอพพลิเคชั่นนี้ เพราะเนื่องจากเป็นแอพพลิเคชั่นที่เกี่ยวข้องกับการควบคุมเครื่องใช้ไฟฟ้าภายในบ้าน ซึ่งมีความเป็นส่วนตัวในการใช้งานของแต่ละบุคคล จึงจำเป็นต้องมีการเข้ารหัสผ่านเพื่อความปลอดภัยของผู้ใช้งาน จากรูปที่ 3.38 แสดงองค์ประกอบของแอพพลิเคชั่นภายในหน้าล็อกอินมีดังนี้

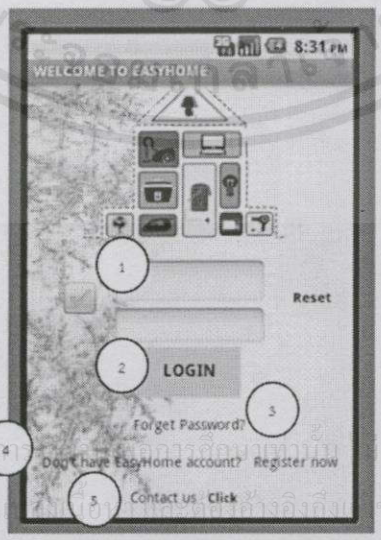
1. ชื่อผู้ใช้งาน (Username) และรหัสผ่าน ประกอบไปด้วยตัวเลข 4-8 ตัว โดยในส่วนของขั้นตอนการล็อกอิน มีผังการทำงาน (Flow Chart) ดังรูปที่ 3.38

2. ปุ่ม “Login” สำหรับเข้าใช้งานในหน้าถัดไป (ควบคุมเครื่องใช้ไฟฟ้า)

3. กรณีที่ไม่สามารถจำรหัสผ่านเดิมได้ จำเป็นต้องทำการร้องขอ เพื่อขอดูรหัสผ่านของผู้ใช้งาน

4. ในกรณีที่ไม่เคยเข้ารับการใช้งาน จำเป็นต้องสมัครบัญชี เพื่อขอเข้ารับการใช้งาน

5. เมื่อเกิดปัญหาขึ้นจากการใช้งาน หรือต้องการที่จะติดต่อสอบถามข้อมูลเพิ่มเติม สามารถดูช่องทางการติดต่อได้จากหน้านี้



รูปที่ 3.38 องค์ประกอบแอพพลิเคชั่นหน้าล็อกอิน

รูปที่ 3.39 ผังการทำงานส่วนของระบบล็อกอิน

1.2 หน้าลงทะเบียนผู้ใช้งาน (Register) เพื่อสมัครการใช้งานแอพพลิเคชั่นของผู้ใช้งาน ซึ่งต้องกรอกรายละเอียด ดังรูปที่ 3.40 เพื่อเข้าถึงแอพพลิเคชั่น แล้วนำรหัสผ่านมาใช้งานหน้าล็อกอิน

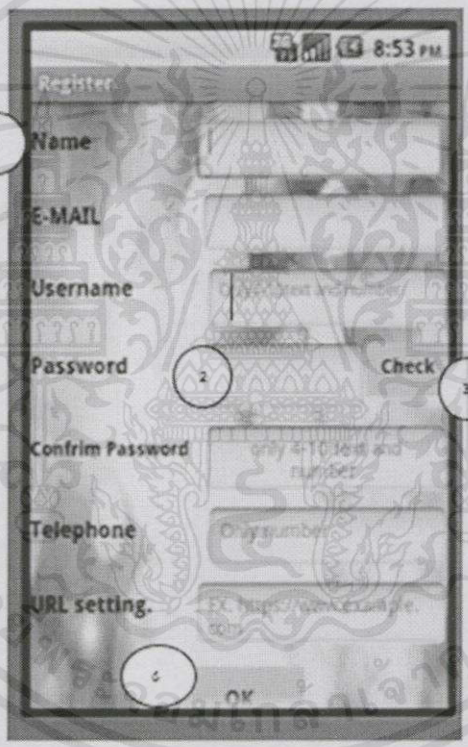
1. รายละเอียดของการสมัคร เพื่อเข้าใช้งานแอพพลิเคชั่น

2. รหัสผ่านที่ใช้ในการเข้าถึงตัวแอพพลิเคชั่นนั้น จะระบุให้ใส่ตัวเลขเพียง 4-8 ตัว ของผู้เข้าใช้งาน

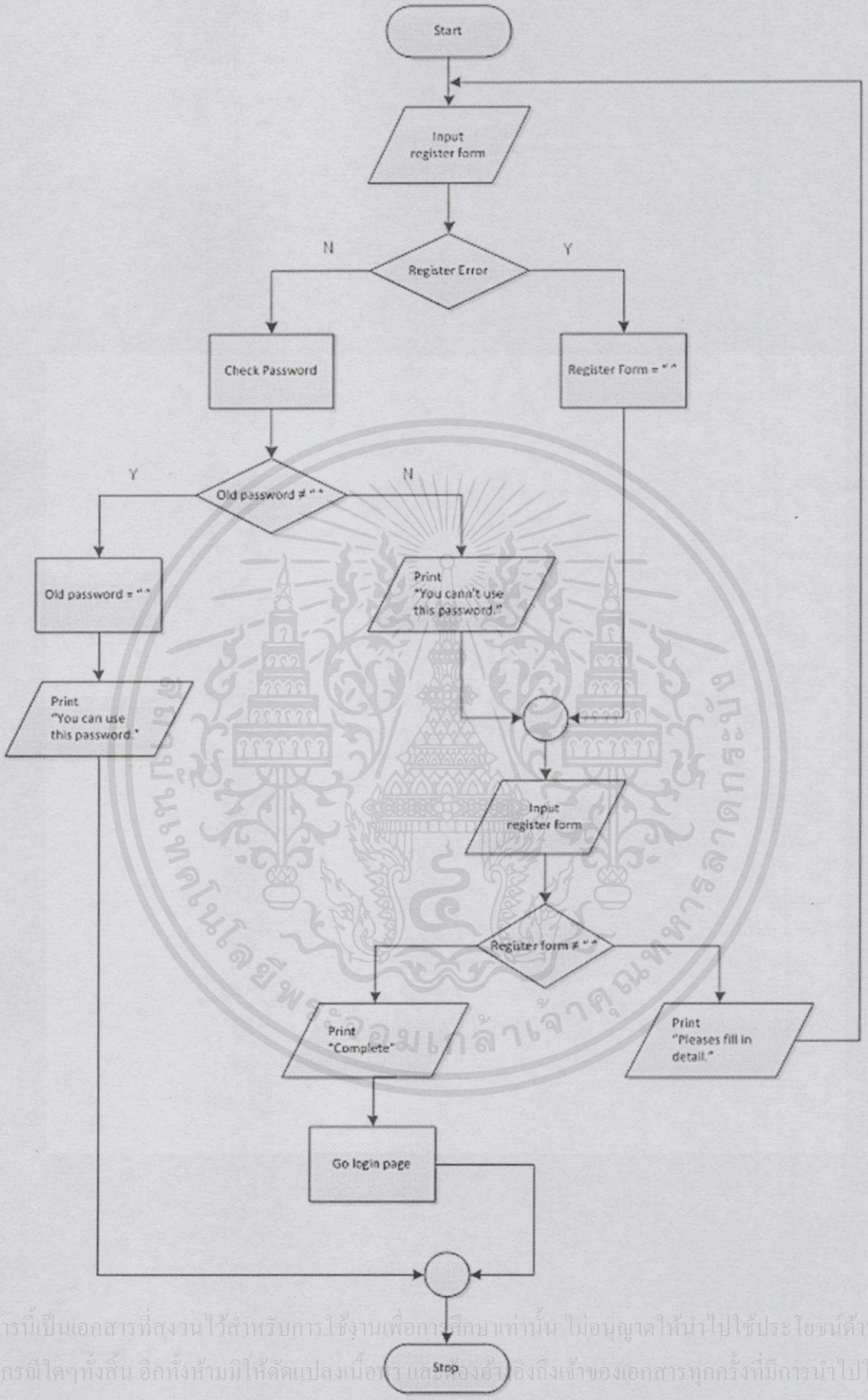
3. ปุ่ม “Check” เพื่อทำการตรวจสอบว่าจะสามารถใช้รหัสผ่าน ในการสมัครเข้าใช้งานได้หรือไม่

4. ปุ่ม “OK” สำหรับการยืนยันการสมัครขอเข้าใช้งาน

โดยในส่วนของการสมัครสมาชิก เพื่อเข้าใช้งานมีผังการทำงาน (Flow Chart) ดังรูปที่ 3.41



รูปที่ 3.40 องค์ประกอบแอพพลิเคชั่นหน้าลงทะเบียนผู้ใช้งาน



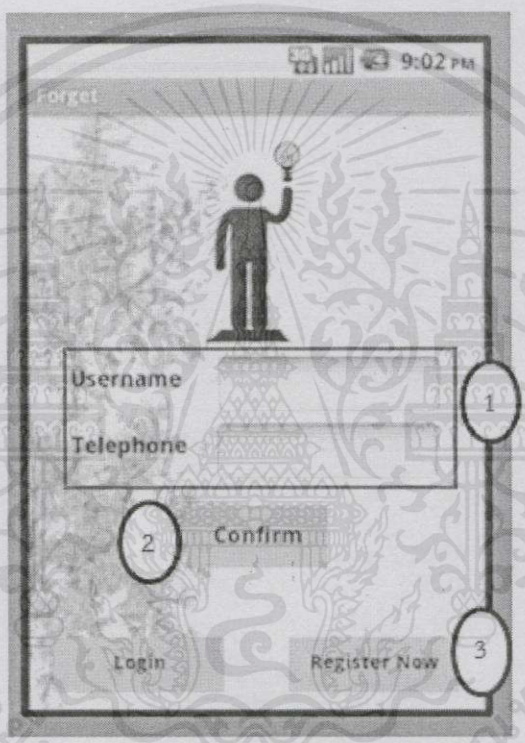
รูปที่ 3.41 ผังการทำงานส่วนของระบบลงทะเบียนผู้ใช้งาน

1.3 หน้าลืมรหัสผ่าน (Forget) เพื่อทำการขอรหัสผ่านเดิม ซึ่งจำเป็นต้องกรอกรายละเอียด ดังรูปที่ 3.42 เพื่อเข้าถึงแอพพลิเคชั่น แล้วนำรหัสผ่านมาใช้งานหน้าล็อกอินต่อไป โดยมีผังการทำงาน ดังรูปที่ 3.43

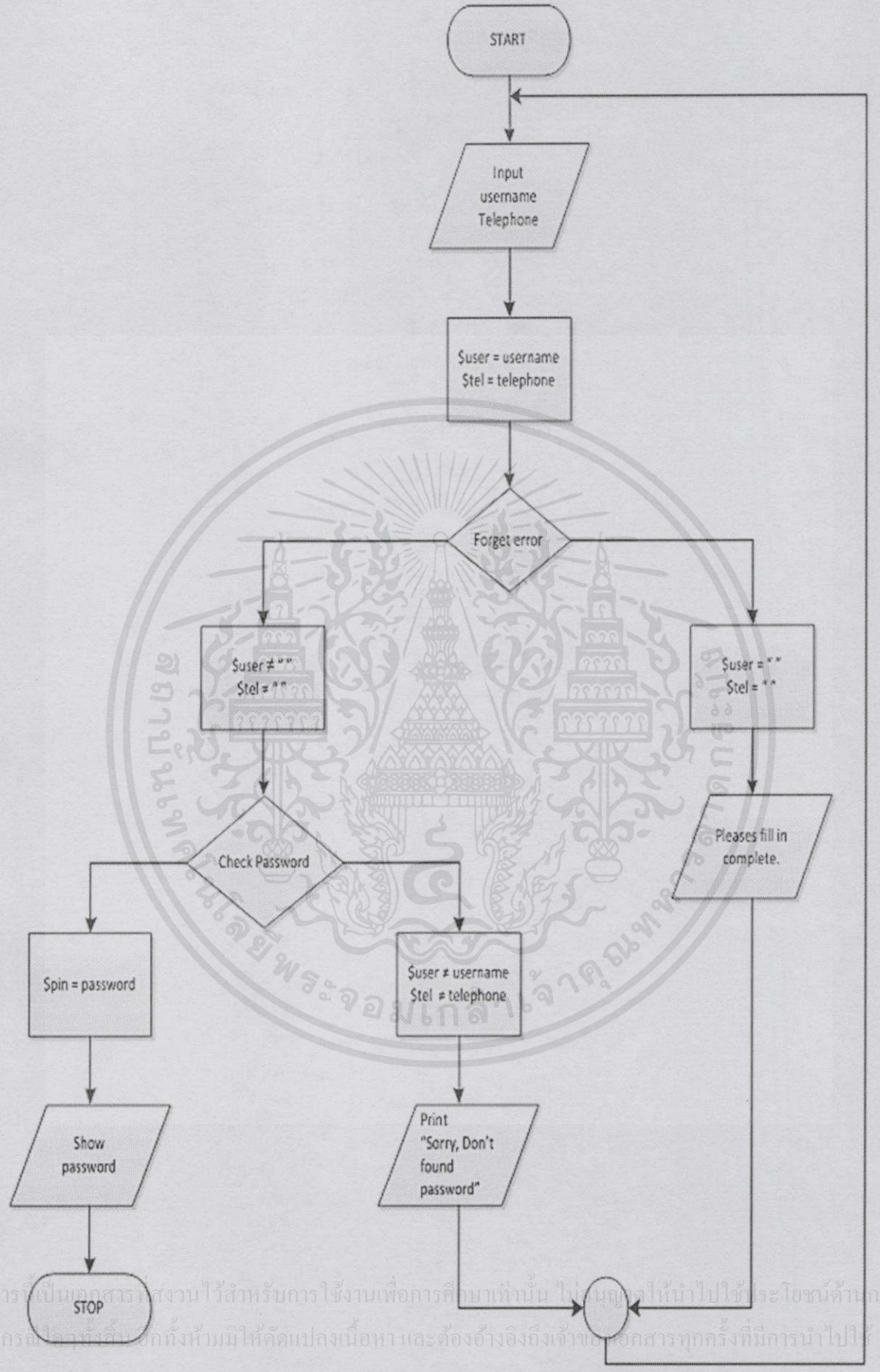
1. ข้อมูลผู้ใช้งานที่จำเป็นต้องกรอก คือชื่อผู้ใช้งาน และเบอร์โทรศัพท์

2. ปุ่ม “Confirm” เพื่อทำการตรวจสอบว่าข้อมูลที่กรอกถูกต้องหรือไม่ เพื่อใช้ในการขอรหัสผ่าน

3. ปุ่ม “Login” และ “Register Now” เพื่อย้ายไปยังหน้าเข้าใช้งาน หรือทำการสมัครใหม่ ตามลำดับ

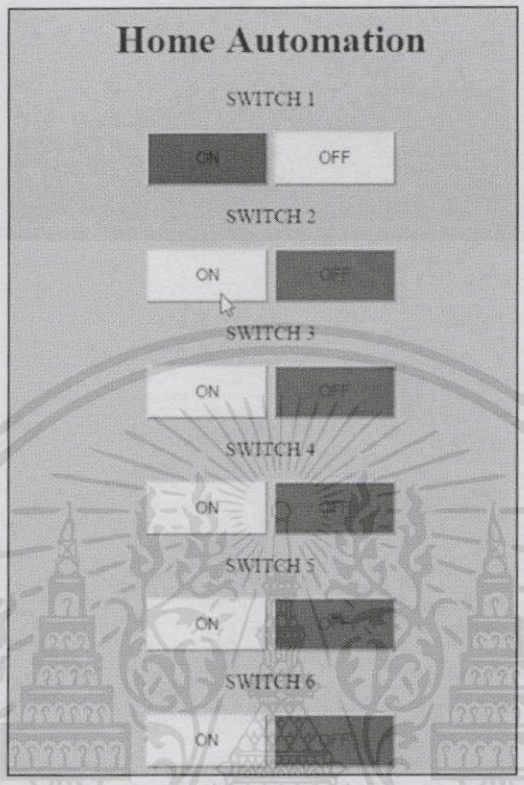


รูปที่ 3.42 องค์ประกอบแอพพลิเคชั่นหน้าลงทะเบียนผู้ใช้งาน



รูปที่ 3.43 ผังการทำงานกรณีผู้ใช้ลืมรหัสผ่าน

2. ส่วนของหน้าการควบคุมเครื่องใช้ไฟฟ้า ซึ่งในหน้านี้ถือว่าเป็นส่วนสำคัญที่สุดของแอพพลิเคชั่นเนื่องจากเป็นส่วนที่ติดต่อกับอุปกรณ์ทั้งหมด ดังรูปที่ 3.44



รูปที่ 3.44 องค์ประกอบแอพพลิเคชั่นหน้าส่วนควบคุมอุปกรณ์ไฟฟ้า

ซึ่งประกอบด้วย

2.1 ชื่อโครงงาน (Project) ที่ทำการสร้าง ชื่อว่า “Home Automation”

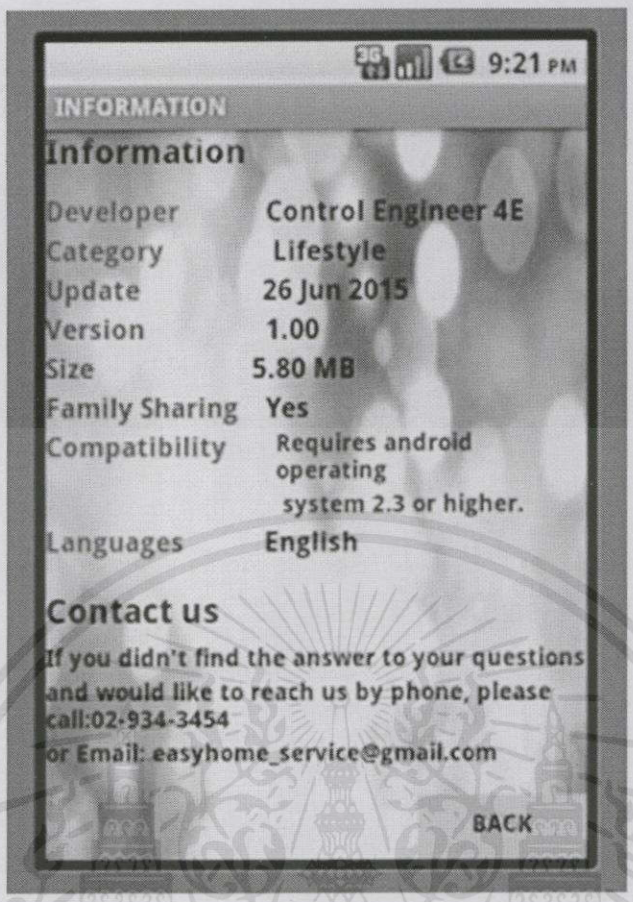
2.2 รูปแสดงสถานะการทำงานของสวิตช์ เพื่อให้ผู้ใช้งานมองเห็นภาพของสถานะการทำงานของสวิตช์ได้ดียิ่งขึ้น

2.3 สวิตช์ 6 ตัว ที่ใช้ควบคุมอุปกรณ์ไฟฟ้าโดยที่ผู้ใช้งาน สามารถเปลี่ยนอุปกรณ์ไฟฟ้าได้อย่างอิสระ ตามความต้องการของผู้ใช้งาน ซึ่งจะมีการแสดงสถานะการทำงานของตัวอุปกรณ์ไฟฟ้า ให้เห็นผ่านหน้าจอสมาร์ทโฟน ดังรูปที่ 3.44 โดยกำหนดให้

สีแดง = สวิตช์ไม่ทำงาน (Off) ไม่มีกระแสไฟฟ้าไหลผ่าน อุปกรณ์ไฟฟ้าไม่สามารถทำงานได้

สีเขียว = สวิตช์ทำงาน (On) กระแสไฟฟ้าไหลผ่าน อุปกรณ์ไฟฟ้าสามารถทำงานได้

3. ข้อมูลของแอพพลิเคชั่น ในหน้านี้จะเป็นส่วนที่บอกรายละเอียด ข้อมูลเบื้องต้นของการพัฒนาแอพพลิเคชั่น รวมไปถึงเวอร์ชั่นของระบบปฏิบัติการแอนดรอยด์ ที่แอพพลิเคชั่นสามารถรองรับได้

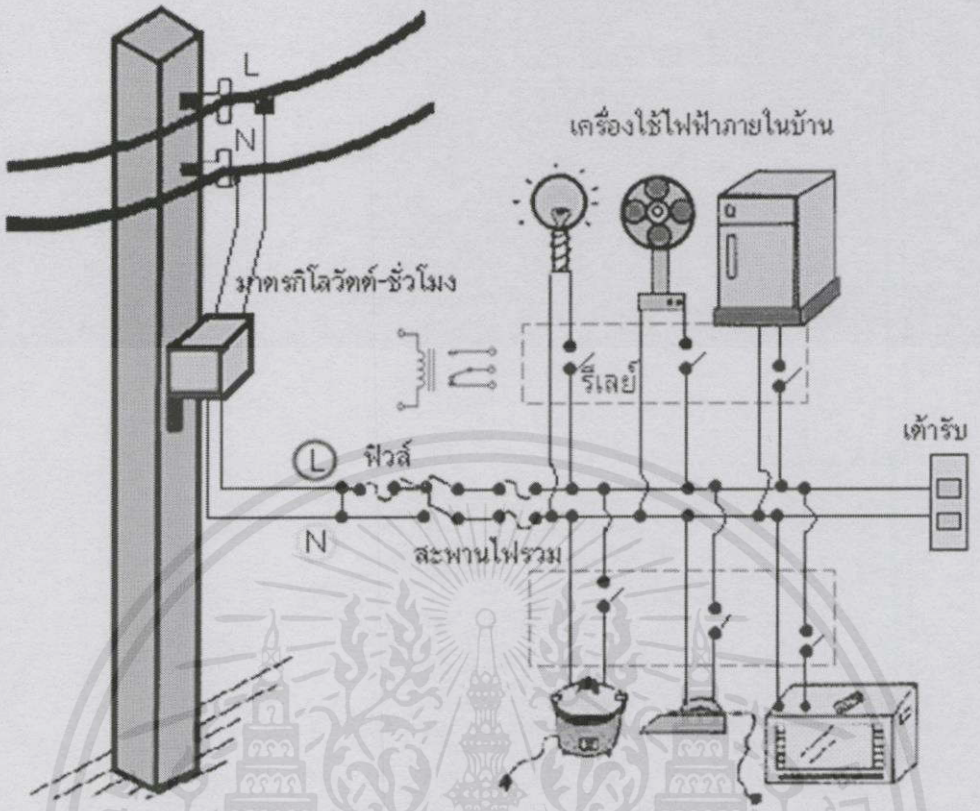


รูปที่ 3.45 ข้อมูลเบื้องต้นของแอพพลิเคชั่น

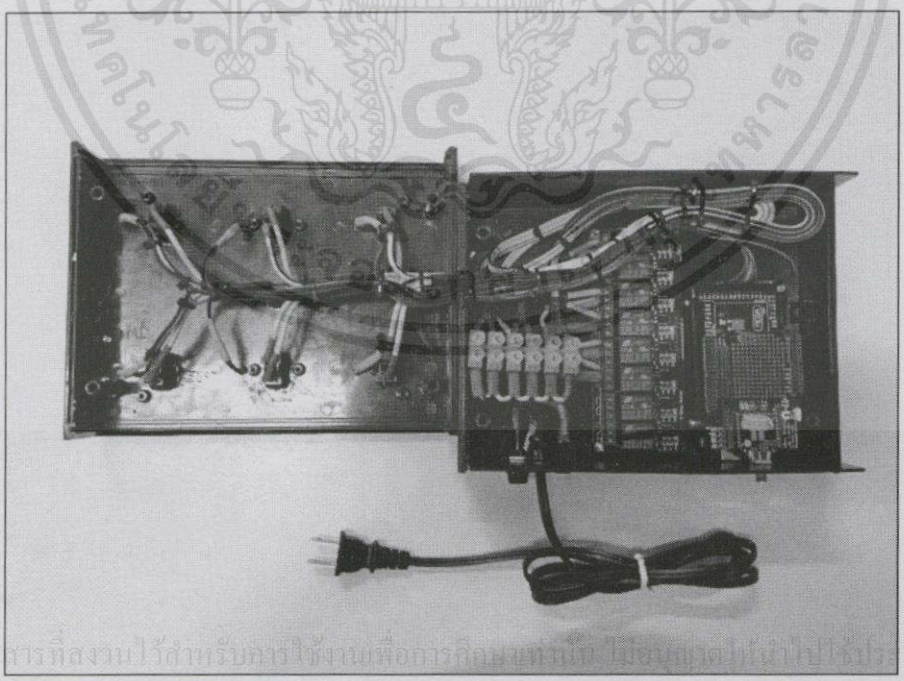
3.7 ภาพรวมของวงจรไฟฟ้าในการทดลอง

วงจรไฟฟ้าภายในบ้านจะใช้ไฟฟ้ากระแสสลับ มีค่าความต่างศักย์ไฟฟ้าเฉลี่ยอยู่ที่ 220 V สายไฟที่ต่อเข้ามาในบ้านจะมี 2 สาย โดยต่อจากสายหลักที่เสาไฟฟ้า ผ่านมาตรกิโลวัตต์-ชั่วโมง เข้าไปในบ้าน ซึ่งสาย 2 สายนี้ จะมีสายหนึ่งเป็นสายกลาง (N) และอีกสายหนึ่งเป็นสายมีศักย์ (L) โดยสายมีศักย์จะต่อผ่านฟิวส์ ซึ่งเป็นตัวป้องกันอันตรายที่จะเกิดขึ้นจากไฟฟ้าลัดวงจร หรือจากการใช้กระแสไฟฟ้าเกินขนาดที่ฟิวส์จะรับได้

เครื่องใช้ไฟฟ้าภายในบ้านจะต่อกันแบบขนานหลังจากผ่านสะพานไฟรวม ซึ่งการเปิด-ปิดเครื่องใช้ไฟฟ้าปกติแล้วจะใช้สวิตช์ในการควบคุม แต่สำหรับการทดลองนี้จะใช้รีเลย์เป็นตัวควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้าแทน โดยรับคำสั่งจากไมโครคอนโทรลเลอร์ ดังรูปที่ 3.46 และรูปที่ 3.47



รูปที่ 3.46 แผนภาพวงจรไฟฟ้าที่ใช้ในการทดลอง



รูปที่ 3.47 การต่อวงจรไฟบ้านร่วมกับอุปกรณ์ควบคุม

บทที่ 4

การทดลองและผลการทดลอง

4.1 อุปกรณ์การทดลอง

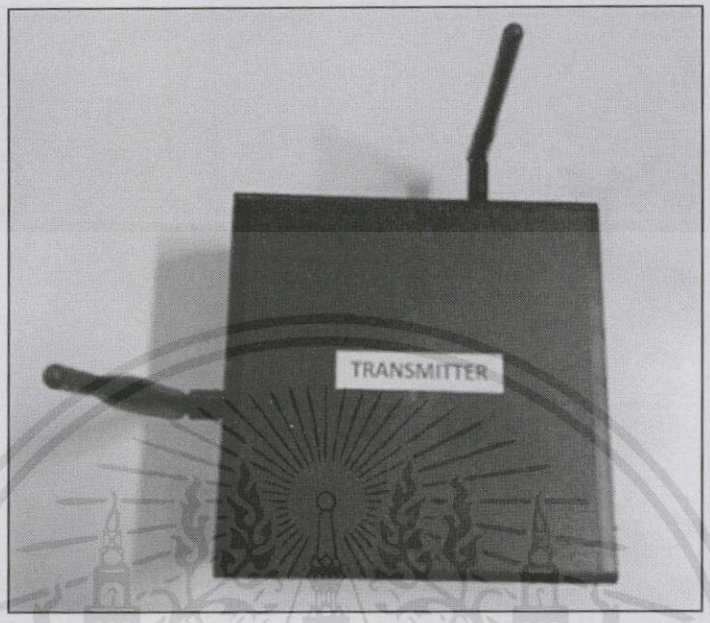
1. อุปกรณ์ผู้ใช้งาน (User interface) ได้แก่ สมาร์ทโฟน, แท็ปเลต, คอมพิวเตอร์ เป็นต้น ทำหน้าที่สั่งงานควบคุมอุปกรณ์ไฟฟ้าในบ้านบนเครือข่ายอินเตอร์เน็ต ดังรูปที่ 4.1

รูปที่ 4.1 อุปกรณ์ที่ใช้ในการสั่งงาน

2. เราเตอร์ตัวส่งสัญญาณ Wifi จะส่งสัญญาณอินเตอร์เน็ตไปที่เราเตอร์ตัวรับสัญญาณ Wifi Tenda ในภาคตัวส่ง (Transmitter) ดังรูปที่ 4.2

รูปที่ 4.2 เราเตอร์ตัวส่งสัญญาณ Wifi TP-Link TD-W8961ND 300Mbps Wireless N ADSL2

3. ภาคตัวส่ง รับสัญญาณ Wifi จากเราเตอร์ตัวส่ง และส่งข้อมูลชุดคำสั่งผ่านอุปกรณ์ XBee ไปยังภาคตัวรับ (Receiver) ดังรูปที่ 4.3

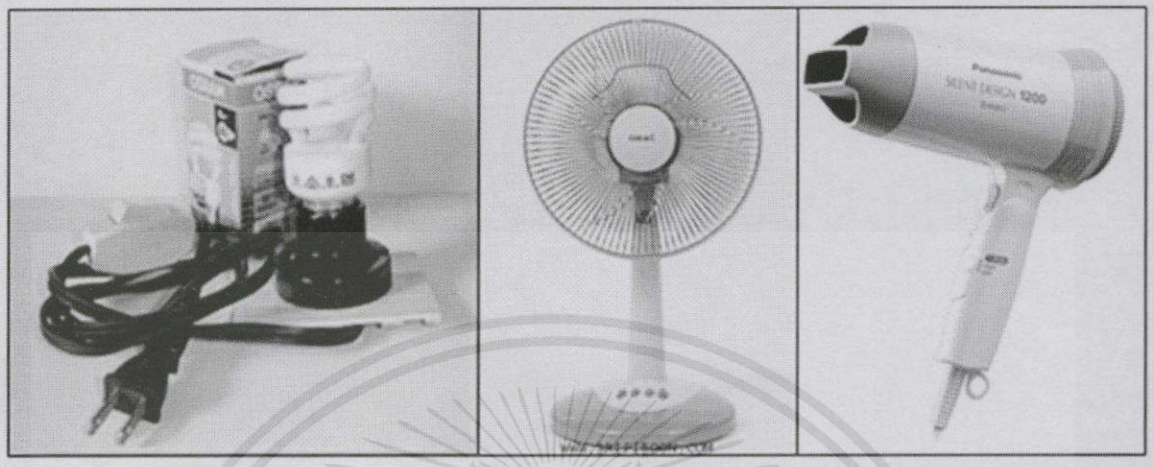


รูปที่ 4.3 กล่องภาคตัวส่ง

4. ภาคตัวรับ จะรับข้อมูลชุดคำสั่งผ่านอุปกรณ์ XBee และทำการควบคุมอุปกรณ์ไฟฟ้าตามคำสั่งที่ได้รับ รวมไปถึงแสดงสถานะการทำงานของอุปกรณ์ไฟฟ้าด้วยไฟ LED ดังรูปที่ 4.4

รูปที่ 4.4 กล่องภาคตัวรับ

5. อุปกรณ์ไฟฟ้า (Electrical Equipment) ได้แก่ หลอดไฟ พัดลม ไดร์เป่าผม เป็นต้น ดังรูปที่ 4.5



รูปที่ 4.5 อุปกรณ์ไฟฟ้า

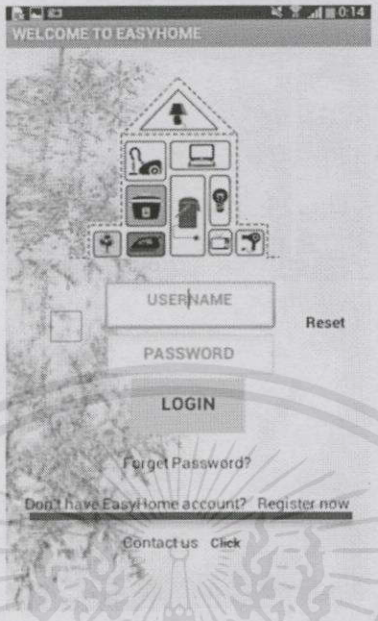
4.2 การทดลองควบคุมอุปกรณ์ไฟฟ้าแบบออนไลน์

4.2.1 การทดลองการควบคุมอุปกรณ์ไฟฟ้าด้วยสมาร์ทโฟนระบบปฏิบัติการแอนดรอยด์ผ่านแอพพลิเคชั่น

1. แอพพลิเคชั่น Easy-Home เพื่อเข้าสู่การควบคุมอุปกรณ์ไฟฟ้า ดังรูปที่ 4.6

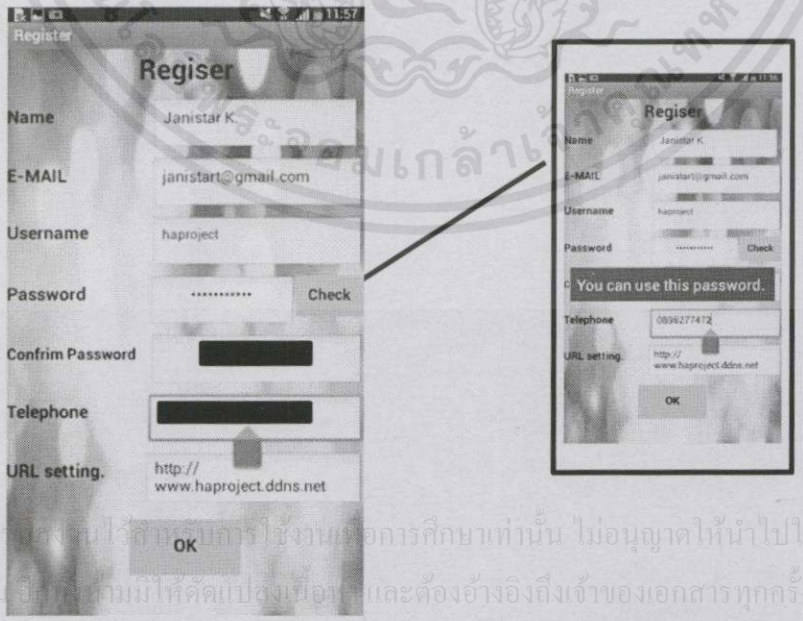
รูปที่ 4.6 ไอคอนแอพพลิเคชั่น Easy-Home

2. เมื่อเข้าสู่หน้าแอพพลิเคชั่น Easy-Home ให้ทำการสมัครสมาชิก เพื่อขอเข้าใช้งานแอพพลิเคชั่น ดังรูปที่ 4.7



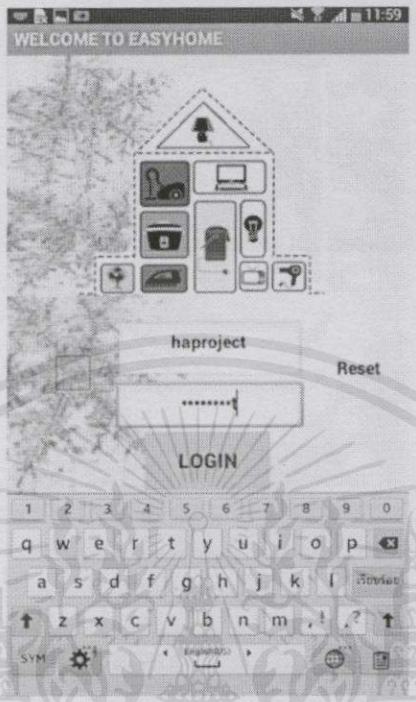
รูปที่ 4.7 หน้าแรกของแอพพลิเคชั่น

3. ในหน้า “Register” ให้ทำการกรอกรายละเอียดข้อมูลให้ครบถ้วน โดยช่อง “URL Setting” คือช่องที่ผู้ใช้จำเป็นต้องใส่หน้าเว็บไซต์ของการควบคุมอุปกรณ์ไฟฟ้าที่ได้รับ ซึ่งในที่นี้คือ http://www.haproject.ddns.net:1234 แล้วทำการตรวจสอบรหัสผ่านว่าสามารถใช้ในการสมัครได้หรือไม่ ดังรูปที่ 4.8 จากนั้นกด “OK” เพื่อเสร็จสิ้นกระบวนการลงทะเบียน



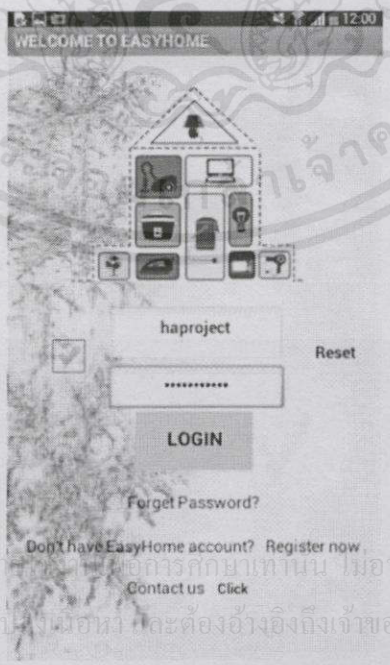
รูปที่ 4.8 ปุ่ม “Check” เพื่อตรวจสอบรหัสผ่าน

4. หลังจากทำการลงทะเบียนเพื่อขอเข้าใช้งานเรียบร้อยแล้ว แอพพลิเคชั่นจะกลับเข้าสู่หน้าจอหลักให้อัตโนมัติ จากนั้นใส่ชื่อผู้ใช้งาน และรหัสผ่านให้ถูกต้อง ดังรูปที่ 4.9



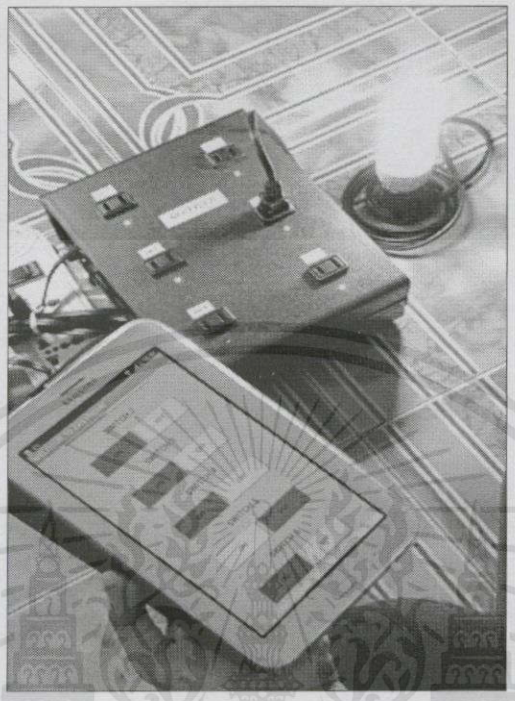
รูปที่ 4.9 ขั้นตอนการขอเข้าใช้งาน (Login)

5. กดปุ่มกล่องข้อความ (Check Box) เพื่อตรวจสอบความถูกต้องของชื่อผู้ใช้งาน และรหัสผ่าน ดังรูปที่ 4.10 จากนั้นกดปุ่ม “Login” เพื่อยืนยัน และเข้าสู่หน้าควบคุมอุปกรณ์ไฟฟ้า



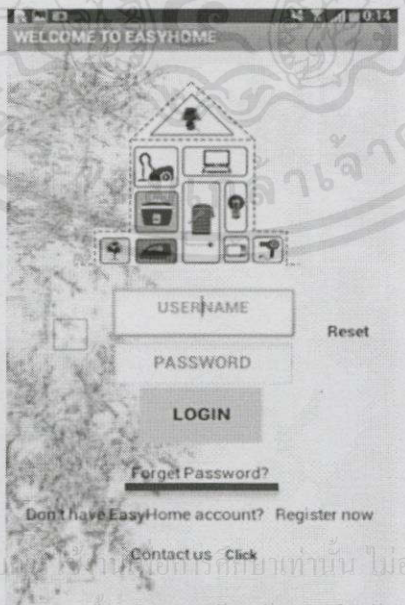
รูปที่ 4.10 ขั้นตอนการขอเข้าใช้งานต่อ

6. เมื่อเข้าสู่หน้าสั่งการ ผู้ใช้ก็จะสามารถควบคุมอุปกรณ์ไฟฟ้าผ่านหน้าแอพพลิเคชั่น Easy-Home บนระบบปฏิบัติการแอนดรอยด์ได้ ดังรูปที่ 4.11



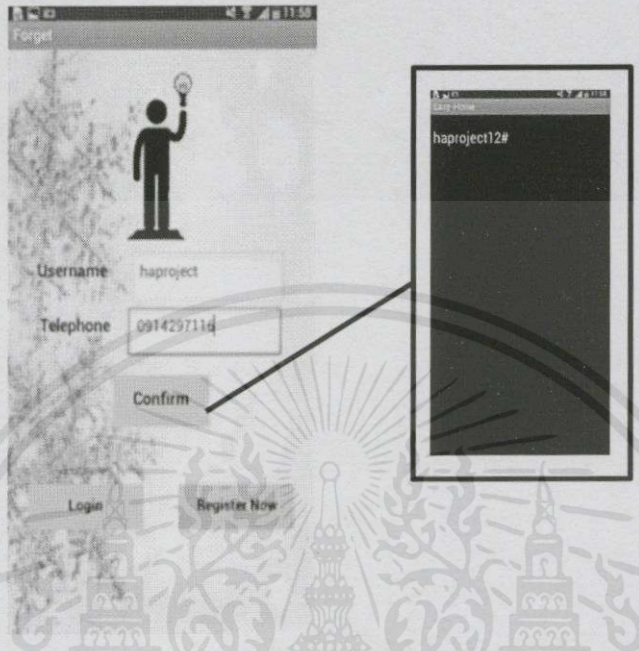
รูปที่ 4.11 หน้าจอสั่งงานควบคุมอุปกรณ์ไฟฟ้าผ่านแอพพลิเคชั่น “Easy-Home”

7. กรณีที่ผู้ใช้งานไม่สามารถจำรหัสผ่านเข้าใช้งานได้ สามารถร้องขอดูรหัสผ่านได้ที่หน้า“Forget” ดังรูปที่ 4.12



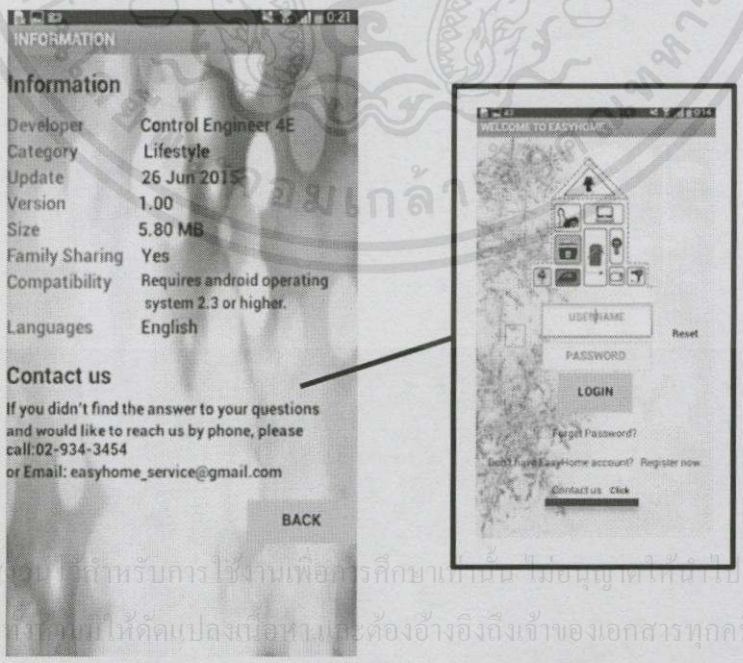
รูปที่ 4.12 ข้อความเพื่อคลิกเข้าสู่ลืมรหัสผ่าน

8. เมื่อเข้าสู่หน้า “Forget” ให้ทำการกรอกไอดีผู้ใช้งาน และเบอร์โทรศัพท์ ที่ได้ทำการลงทะเบียนไว้ เพื่อทำการขอเรียกดูรหัสผ่าน จากฐานข้อมูลของระบบ กด“Confirm” เพื่อทำการยืนยันขอเรียกดูรหัสผ่าน ถ้าข้อมูลถูกต้อง ระบบจะทำการแสดงรหัสผ่านผู้ใช้งาน ดังรูปที่ 4.13



รูปที่ 4.13 ขั้นตอนการขอเรียกดูรหัสผ่าน

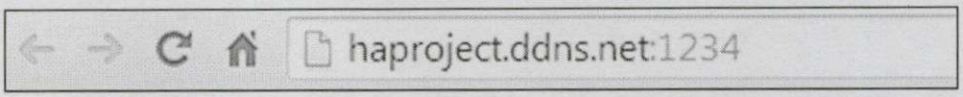
9. หากผู้ใช้ต้องการดูข้อมูล หรือรายละเอียดของแอพพลิเคชั่น ดังรูปที่ 4.14 สามารถคลิกดูได้จาก“Contact Us” ในหน้าแรกของแอพพลิเคชั่น



รูปที่ 4.14 รายละเอียดของแอพพลิเคชั่น

4.2.2 การทดลองควบคุมอุปกรณ์ไฟฟ้าด้วยคอมพิวเตอร์

1. เริ่มต้นด้วยการเข้าเว็บแล้วพิมพ์ haproject.ddns.net:1234 ดังรูปที่ 4.15



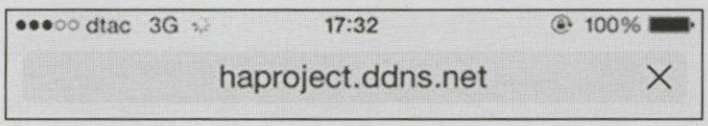
รูปที่ 4.15 พิมพ์ haproject.ddns.net:1234 เพื่อเข้าหน้าเว็บ

2. สามารถกดสั่งงานควบคุมอุปกรณ์ไฟฟ้าได้ ดังรูปที่ 4.16

รูปที่ 4.16 หน้าเว็บที่สามารถกดสั่งงานควบคุมอุปกรณ์ไฟฟ้าผ่านคอมพิวเตอร์

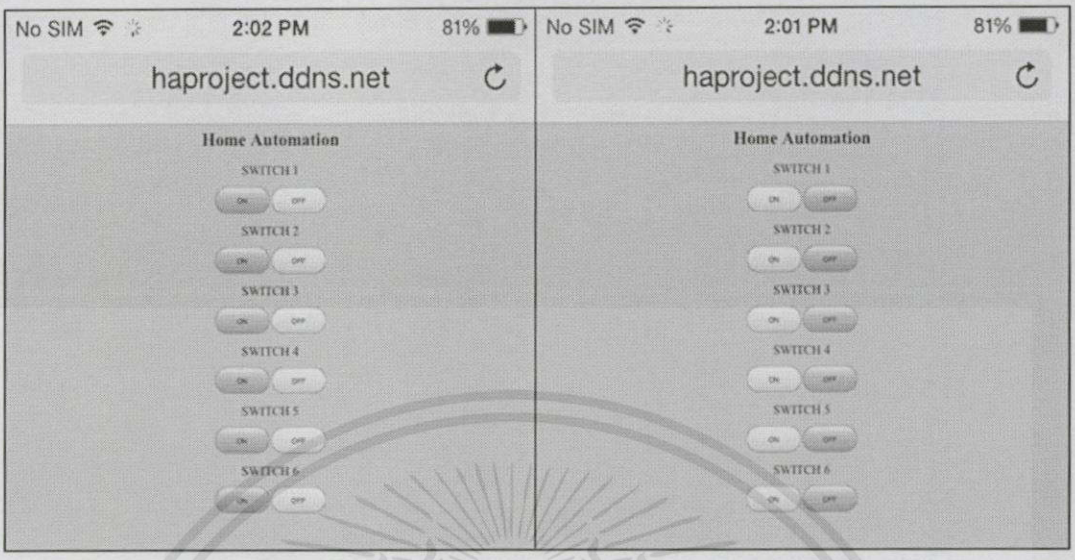
4.2.3 การทดลองการควบคุมอุปกรณ์ไฟฟ้าโดยเว็บบราวเซอร์บนโทรศัพท์มือถือ (Iphone)

1. เริ่มต้นด้วยการเข้าเว็บแล้วพิมพ์ haproject.ddns.net:1234 ดังรูปที่ 4.17



รูปที่ 4.17 พิมพ์ haproject.ddns.net:1234 เพื่อเข้าหน้าเว็บ

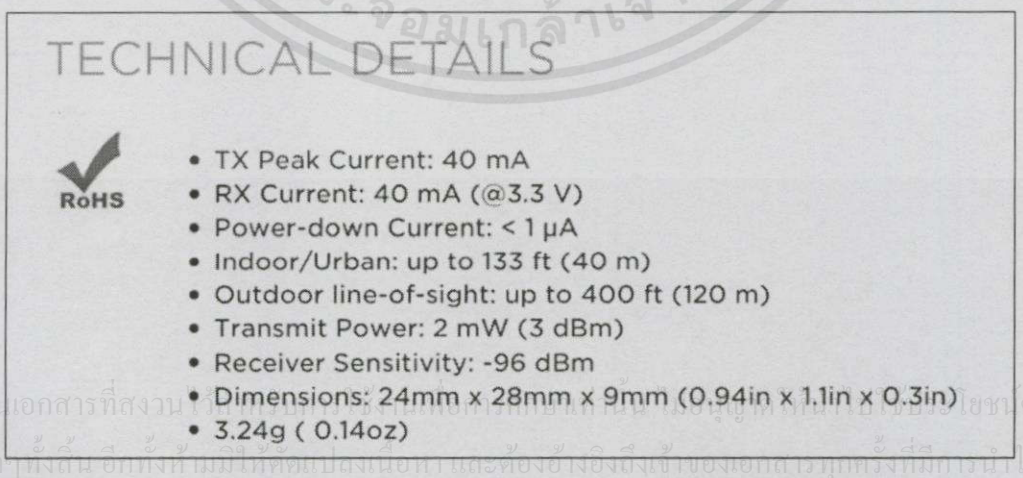
2. สามารถกดสั่งงานควบคุมอุปกรณ์ไฟฟ้าได้ ดังรูปที่ 4.18



รูปที่ 4.18 หน้าเว็บที่สามารถกดสั่งงานควบคุมอุปกรณ์ไฟฟ้าผ่านโทรศัพท์มือถือ

4.3 การทดลองระยะการรับ-ส่งข้อมูลของอุปกรณ์ XBee แบบออฟไลน์

ในการควบคุมอุปกรณ์ไฟฟ้าแบบไร้สาย จำเป็นต้องคำนึงถึงระยะการับ-ส่งข้อมูลของอุปกรณ์ โดยในที่นี้คืออุปกรณ์ XBee ฉะนั้นจึงจำเป็นต้องทดสอบประสิทธิภาพของระยะการรับ-ส่งข้อมูลของ XBee เพื่อให้ผู้ใช้งาน สามารถมั่นใจได้ว่าอุปกรณ์ที่ใช้งานนั้นมีประสิทธิภาพหรือข้อจำกัดการใช้งานมากน้อยเพียงใด ซึ่งจากข้อมูลคุณสมบัติของ XBee Series 2 Wire Antenna ที่ใช้ในการทดลอง ดังรูปที่ 4.19 จะเห็นได้ว่าอุปกรณ์ XBee มีระยะการรับ-ส่งข้อมูลภายในอาคารอยู่ที่ 40 เมตร และภายนอกอาคาร (กลางแจ้ง) อยู่ที่ 120 เมตร



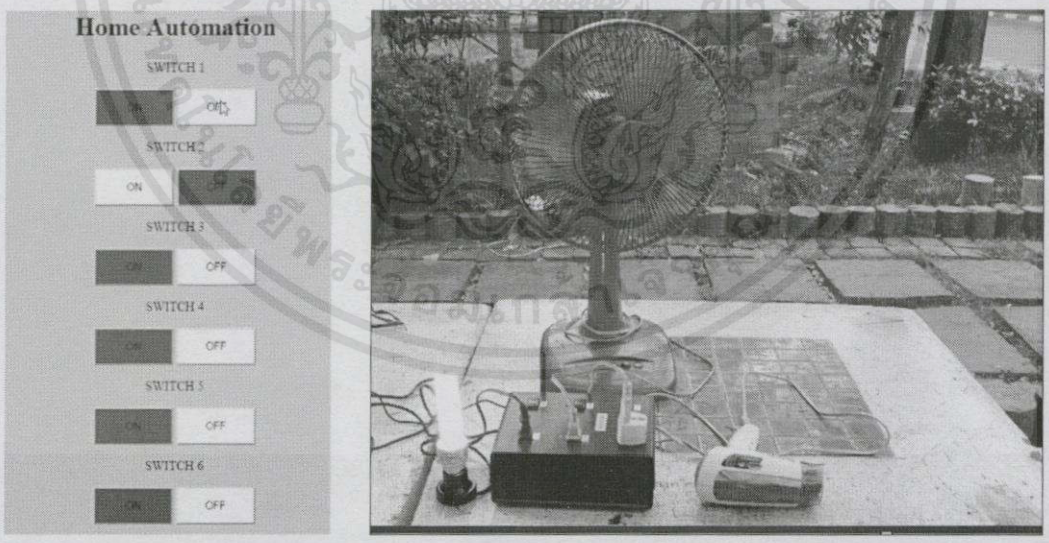
รูปที่ 4.19 คุณสมบัติของ XBee Series 2 Wire Antenna

จากการทดลอง ได้ทำการทดสอบประสิทธิภาพการรับ-ส่งข้อมูลของ XBee เพื่อนำมาใช้ควบคุมอุปกรณ์ไฟฟ้า ดังตารางที่ 4.1 โดยแบ่งการทดลองออกเป็น 2 กรณี คือ

1. อุปกรณ์ที่ใช้ในการรับส่ง-ข้อมูล ภายในอาคาร (มีสิ่งกีดขวาง) โดยผลการทดลองแสดงตัวอย่างการทดสอบระยะ ดังรูปที่ 4.20

2. อุปกรณ์ที่ใช้ในการรับส่ง-ข้อมูล บริเวณกลางแจ้ง โดยแสดงตัวอย่างการทดสอบระยะ ดังรูปที่ 4.21

รูปที่ 4.20 ระยะการรับส่ง-ข้อมูล ภายในอาคารที่ 30 เมตร



รูปที่ 4.21 ระยะการรับส่ง-ข้อมูล ภายนอกอาคารที่ 70 เมตร

ตารางที่ 4.1 การเปรียบเทียบความสามารถในการรับ-ส่งข้อมูลของอุปกรณ์ XBee

ระยะการรับ-ส่งข้อมูล (เมตร)	ผลการทดลอง	
	ภายในอาคาร	ภายนอกอาคาร
10	สั่งงานได้	สั่งงานได้
20	สั่งงานได้	สั่งงานได้
30	สั่งงานได้	สั่งงานได้
40	สั่งงานได้	สั่งงานได้
50	สั่งงานไม่ได้	สั่งงานได้
60	สั่งงานไม่ได้	สั่งงานได้
70	สั่งงานไม่ได้	สั่งงานได้
80	สั่งงานไม่ได้	สั่งงานได้
90	สั่งงานไม่ได้	สั่งงานได้
100	สั่งงานไม่ได้	สั่งงานได้
110	สั่งงานไม่ได้	สั่งงานไม่ได้
120	สั่งงานไม่ได้	สั่งงานไม่ได้

ดังนั้น สามารถสรุปได้ว่าสิ่งกีดขวางมีผลต่อการรับ-ส่งข้อมูลของอุปกรณ์ XBee ซึ่งจะทำให้ความสามารถในการทำงานของอุปกรณ์ลดลง

นอกเหนือจากประสิทธิภาพของระยะการรับ-ส่งข้อมูลของอุปกรณ์ XBee แล้ว ช่องทางการติดต่อสื่อสารก็มีความสำคัญเช่นกัน เพื่อให้สามารถสั่งการควบคุมอุปกรณ์ไฟฟ้าได้อย่างรวดเร็ว ง่าย สะดวก และมีความปลอดภัยในการใช้งาน จึงได้ทำการทดสอบประสิทธิภาพการสั่งการควบคุมอุปกรณ์ไฟฟ้าแบบออนไลน์ โดยสั่งการผ่านระบบอินเตอร์เน็ต หรือเลือกใช้งานผ่านแอพพลิเคชั่น

จากการทดสอบระบบสั่งการควบคุมการเปิด-ปิดอุปกรณ์ไฟฟ้าแบบออนไลน์ ผลที่ได้คือสามารถสั่งการได้อย่างมีประสิทธิภาพ และต่อเนื่องในการใช้งาน ทั้งนี้ขึ้นอยู่กับความเร็วของสัญญาณอินเตอร์เน็ตที่ใช้ ไปจนถึงระยะการวางอุปกรณ์รับ-ส่งข้อมูล ที่จะต้องอยู่ในระยะห่างที่กำหนดไว้ ดังการทดลองที่ 4.3

บทที่ 5

บทวิจารณ์ และสรุปผล

5.1 สรุปผลการทดลอง

โครงงานฉบับนี้ ได้นำเสนอการออกแบบระบบควบคุมการทำงานของอุปกรณ์ไฟฟ้าภายในบ้านแบบไร้สายผ่านอุปกรณ์ XBee และสั่งงานในรูปแบบของแอพพลิเคชั่นแอนดรอยด์บนสมาร์ทโฟน อีกทั้งยังทำการสั่งงานบนหน้าจอคอมพิวเตอร์ได้อีกด้วย โดยที่ระบบสามารถสั่งการทางไกลแบบออนไลน์บนเครือข่ายอินเตอร์เน็ตได้

จากการศึกษา ออกแบบ ทดลอง และแก้ไขข้อมูลที่ผิดพลาดของการทำงานทั้งหมดของระบบนี้ สามารถสรุปได้ ดังนี้

1. คำสั่งจากผู้ใช้งาน (User) สามารถสั่งงานผ่านแอพพลิเคชั่นแอนดรอยด์บนสมาร์ทโฟน หรือคอมพิวเตอร์ เพื่อควบคุมอุปกรณ์ไฟฟ้าในบ้าน ทั้งนี้ผู้ใช้งานสามารถสั่งงานทางไกลแบบออนไลน์ผ่านเครือข่ายอินเตอร์เน็ตได้ทุกที่

2. ภาคส่ง จะส่งคำสั่งควบคุมไปยังตัวรับ แบบไร้สายผ่านอุปกรณ์ XBee

3. ภาครับ จะนำคำสั่งที่ได้รับไปควบคุมรีเลย์เพื่อทำการเปิด-ปิดอุปกรณ์ไฟฟ้า โดยออกแบบวงจรไฟบ้านต่อร่วมกับอุปกรณ์ควบคุม และมีไฟ LED แสดงผลสถานะการทำงานของอุปกรณ์ไฟฟ้า รวมทั้งยังสามารถต่อเข้ากับเครื่องใช้ไฟฟ้าภายในบ้านได้จริง

4. ส่วนอุปกรณ์ไฟฟ้า สามารถใช้งานได้ตามเป้าหมาย โดยขึ้นอยู่กับการทำงานของรีเลย์จากภาคตัวรับ

5. ประสิทธิภาพในการรับ-ส่งข้อมูลของอุปกรณ์ XBee นั้น ขึ้นอยู่กับปัจจัยภายนอก ซึ่งหมายถึงสถานที่ที่จัดวางกล่องภาคตัวส่ง และกล่องภาคตัวรับ โดยการวางไว้ในที่โล่งกว้างจะมีประสิทธิภาพในการรับ-ส่งข้อมูลที่ดีกว่าการวางไว้ในที่ที่มีสิ่งกีดขวาง หรือเป็นอุปสรรคต่อการสื่อสารกันของอุปกรณ์

5.2 ปัญหาที่พบ และแนวทางการแก้ไข

จากการศึกษา ออกแบบ ทดลอง ตลอดจนการแก้ไขข้อผิดพลาดของโครงงานนี้ จะพบปัญหาที่เกิดขึ้นในหลายส่วน ทั้งในส่วนฮาร์ดแวร์ และส่วนของซอฟแวร์ แต่ก็สามารถแก้ไขปัญหา

ต่างๆ เพื่อให้ชิ้นงานสามารถทำงานได้ตามเป้าหมายที่วางไว้ได้อย่างสมบูรณ์ โดยที่ปัญหาต่างๆ และแนวทางการแก้ไขปัญหาที่เกิดขึ้นมีดังนี้

1. การเลือกใช้โปรแกรมในการเขียนแอพพลิเคชั่น

ในตอนแรกเลือกเขียนแอพพลิเคชั่นโดยใช้โปรแกรม Eclipse แต่เนื่องจากความไม่ชำนาญและไม่เข้าใจคำสั่งในการเขียนภาษาจาวามากพอ ทำให้ต้องหาโปรแกรมอื่นที่เหมาะสำหรับผู้ใช้เพื่อให้สามารถดำเนินงานต่อไปได้

แนวทางการแก้ไข

เปลี่ยนมาเขียนแอพพลิเคชั่นโดยใช้โปรแกรม App Invertor 2 ซึ่งสถาบัน MIT ร่วมกันพัฒนากับ GOOGLE เป็นโปรแกรมที่เหมาะกับผู้เริ่มต้นที่อยากจะพัฒนาแอพพลิเคชั่นบนระบบปฏิบัติการแอนดรอยด์

2. ระยะการนำส่งข้อมูลของอุปกรณ์ XBee

ระยะการรับส่งข้อมูลของอุปกรณ์ XBee ขึ้นอยู่กับเสาอากาศ เนื่องจากอุปกรณ์ XBee รุ่นเดียวกัน หากเสาอากาศสั้นจะไม่สามารถต่อออกมานอกกล่องได้ ซึ่งกล่องถือเป็นสิ่งกีดขวางของสัญญาณการรับ-ส่งข้อมูล

แนวทางการแก้ไข

เลือกใช้อุปกรณ์ XBee ที่มีเสาอากาศยาวเพียงพอที่จะสามารถต่อออกมาภายนอกกล่องได้ ทำให้ไม่เป็นอุปสรรคต่อการรับ-ส่งสัญญาณของอุปกรณ์

3. ข้อมูลที่ใช้ในการทำโครงงาน

เกิดปัญหาด้านซอฟแวร์ เนื่องจากเป็นการศึกษาความรู้ใหม่

แนวทางการแก้ไข

สอบถามผู้รู้ และค้นคว้าจากอินเตอร์เน็ตเพิ่มเติม

4. จำนวนข้อมูลคำสั่งกับการเลือกใช้งานไมโครคอนโทรลเลอร์

คำสั่งสามารถใช้งานได้เพียงบางส่วน เนื่องจากจำนวนบรรทัดของโค้ดคำสั่งมากเกินไป เกินกว่าที่หน่วยความจำของไมโครคอนโทรลเลอร์จะรับได้

แนวทางการแก้ไข

ทำการเปลี่ยนไมโครคอนโทรลเลอร์จาก Arduino UNO เป็น Arduino ATmega2560 เนื่องจากมีหน่วยความจำมากกว่า ทำให้เพียงพอต่อจำนวนบรรทัดของโค้ดคำสั่งที่ต้องการ

5. การเลือกใช้ไฟในการแสดงสถานะ

ไฟที่ใช้ในการแสดงสถานะการทำงานของอุปกรณ์ไฟฟ้า ไม่ควรมีไฟเลี้ยงสูงถึง 220 V เพราะจะส่งผลให้อุปกรณ์ควบคุมเกิดความเสียหายได้

แนวทางการแก้ไข

เลือกใช้ไฟ LED เป็นตัวแสดงสถานะ เนื่องจากต้องการไฟเลี้ยงต่ำ เพื่อลดโอกาสในการทำให้อุปกรณ์ควบคุมเกิดความเสียหาย

6. การออกแบบกล่องตัวรับกับวงจรภายในกล่อง

การวางตำแหน่งของอุปกรณ์ภายในกล่อง โดยภายในกล่องจะมีฝั่งที่เชื่อมต่อเข้ากับไฟ 220 V จากไฟบ้าน และไฟ 5 V จากอุปกรณ์ควบคุม หากวางในตำแหน่งที่ไม่เหมาะสมจะทำให้เกิดความเสียหายกับอุปกรณ์ควบคุมได้

แนวทางการแก้ไข

แยกฝั่งระหว่างไฟ 220 V จากไฟบ้านกับไฟ 5 V จากอุปกรณ์ควบคุมอย่างชัดเจน โดยการใช้น็อตยึดกับฐานกล่อง เพื่อไม่ให้เกิดการเคลื่อนที่เข้าหากัน และใช้ท่อหดในการคลุมลวดส่วนเกินที่อาจจะทำให้เกิดการลัดวงจรอย่างแน่นหนา และมิดชิด เพื่อป้องกันความเสียหายของอุปกรณ์ควบคุมที่อาจจะเกิดขึ้นได้

5.3 ข้อเสนอแนะ และแนวทางในการค้นคว้าพัฒนา

จากที่กล่าวมาทั้งหมด จะเห็นได้ว่าโครงงานนี้สามารถนำไปประยุกต์ใช้งานได้หลากหลายรูปแบบ เนื่องจากอุปกรณ์ XBee มีหลายฟังก์ชันให้เลือกใช้งาน รวมทั้งตัวไมโครคอนโทรลเลอร์เองก็มีความสามารถที่ไม่ได้กล่าวถึงไว้ ขึ้นอยู่กับการเลือกใช้ และวัตถุประสงค์ในการใช้งานของผู้พัฒนาต่อไป ซึ่งแนวทางในการพัฒนาอีกประการหนึ่งคือ สามารถนำโครงงานนี้ไปใช้ร่วมกับการแจ้งเตือนด้านความปลอดภัย เช่น ระบบแจ้งเตือนไฟไหม้ ระบบประตูอัตโนมัติ ระบบป้องกันขโมย โดยทำงานและแสดงผลผ่านเครือข่ายอินเตอร์เน็ต นอกจากนี้ยังสามารถปรับปรุงด้านการแสดงผลให้มีความสวยงามน่าใช้มากยิ่งขึ้น โดยอาจทำเป็นภาพกราฟิก ซึ่งสามารถเขียนได้โดยโปรแกรม Visual Studio C++ ได้

ในด้านของการพัฒนาแอพพลิเคชั่นบนโทรศัพท์มือถือในอนาคต อาจจะมีการขยายความสามารถครอบคลุมไปถึงระบบปฏิบัติการอื่นๆ ทั้งระบบปฏิบัติการการจัดสรรทรัพยากรในเครื่องคอมพิวเตอร์ (operating system, OS) และระบบปฏิบัติการโทรศัพท์มือถือ (Window Phone) และในด้านของฟังก์ชั่นการทำงานของระบบความปลอดภัย จะมีความเสถียร และมี

ประสิทธิภาพมากยิ่งขึ้น โดยอาจทำให้ระบบมีความสามารถโต้ตอบกับผู้ใช้งาน มีการแจ้งเตือนแบบระบบเวลาจริง (Real Time) เพื่อให้ผู้ใช้งานสามารถรู้สถานะการใช้งานของตนได้

การเพิ่มเติมอุปกรณ์ฮาร์ดแวร์ เช่น กล้อง ก็เป็นอีกแนวทางหนึ่งที่จะทำให้ผู้ใช้งานสามารถมองเห็นสภาพแวดล้อมโดยรวมภายในบ้านของตน ซึ่งสามารถสร้างความเชื่อมั่นของระบบความปลอดภัยให้กับผู้ใช้งานได้มากยิ่งขึ้น

เอกสารอ้างอิง

[1] David Wolber, Hal Abelson, Ellen Spertusl, et al.**AI2 Tutorials.** App Inventor 2 Create Your Own Andriod App Second Edition. Sebastopol:O'Reilly; 2014. P.1-32

[2] จักรี กฤษณะเศรณี, ชนาทร ธรรมกิจ และชวิศ ภมรานนท์. (2554). **ระบบควบคุมไฟฟ้าบ้านแบบไร้สาย**. ปริญญานิพนธ์สาขาวิศวกรรมแมคคาทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[3] **XBee คืออะไร.**เข้าถึงได้จาก: http://thaieasyelec.com/article-wiki/basic-electronics/what-is-zigbee.html (วันที่สืบค้นข้อมูล: 27 ตุลาคม 2557).

[4] **การเลือกซื้อ XBee** เข้าถึงได้จาก: http://thaieasyelec.com/article-wiki/review-product-article/how-to-choose-xbee-with-suitable-your-project.html (วันที่สืบค้นข้อมูล: 27 ตุลาคม 2557).

[5] **XBee Learning Practice with XBee Series 2** เข้าถึงได้จาก: http://thaieasyelec.com/article-wiki/embedded-electronics-application/learning-xbee-with-xbee-series-2-starter -kit.html (วันที่สืบค้นข้อมูล: 2 พฤศจิกายน 2557).

[6] **Arduino web server LED control** เข้าถึงได้จาก:http://startingelectronics.com/tutorials/arduino/ethernet-shield-web-server-tutorial/web-server-LED-control/ (วันที่สืบค้นข้อมูล: 2 พฤศจิกายน 2557).

[7] **Android Tutorial.** เข้าถึงได้จาก: http://www.thaicreate.com/mobile/android.html (วันที่สืบค้นข้อมูล: 4 ธันวาคม 2557).

[8] **Connect App Inventor to MySQL** เข้าถึงได้จาก:https://www.youtube.com/watch?v =fvH865tKOLE&feature=youtu.be (วันที่สืบค้นข้อมูล: 7 มกราคม 2558).

[9] **Connect App Inventor to MySQL Andriod Development Tutorial for Beginners** เข้าถึงได้จาก: https://www.youtube.com/watch?v=w-Jq-MTLstk&feature=youtu.be (วันที่สืบค้นข้อมูล: 7 มกราคม 2558).

[10] **พื้นฐานการใช้งาน Ethernet Shield** เข้าถึงได้จาก: http://www.arduitronics.com /article/พื้นฐานการใช้งาน-ethernet-shield-กับ-arduino-ตอนที่-1 (วันที่สืบค้นข้อมูล: 7 มกราคม 2558).

[11] **การตั้งค่าเราเตอร์ TP-link** เข้าถึงได้จาก http://www.freewarelands.com/wp4/วิธีเซ็ต-router-tp-link-td-w8951nd-เร้าเตอร์-tot-/ (วันที่สืบค้นข้อมูล: 11 มกราคม 2558).

[12] **วิธีการตั้งค่า Tenda** เข้าถึงได้จาก: http://www.tenda.co.th/download/manual/QIG_ W308R _W309R.pdf (วันที่สืบค้นข้อมูล: 12 มกราคม 2558).

ภาคผนวก

ภาคผนวก ก.

[1] โปรแกรมภาคส่ง

```
#include <SPI.h>

#include <Ethernet.h>

#include <SoftwareSerial.h>

#define RxD 8

#define TxD 9

////////////////////////////////////////////////////////////////////////////////// Ethernet

SoftwareSerial mySerial(RxD,TxD); // RX, TX

byte mac[] = { 0x94, 0xDE, 0x80, 0xC6, 0x70, 0xE5 };

IPAddress ip(192, 168, 1, 40); // IP address, may need to change depending on
network

EthernetServer server(1234);  // create a server at port 80

String HTTP_req;          // stores the HTTP request

boolean LED_status = 0;   // state of LED, off by default

String readString;

////////////////////////////////////////////////////////////////////////////////// TESTING PIN

int ry1 = 30;

   int ry100 = 43;  ////////////////////// FOR TEST

int ry2 = 32;

   int ry200 = 45;  ////////////////////// FOR TEST
```

```
int ry3 = 34;

int ry4 = 36;

int ry5 = 38;

int ry6 = 40;

int ry1_rcv = 0;

int ry2_rcv = 0;

int ry3_rcv = 0;

int ry4_rcv = 0;

int ry5_rcv = 0;

int ry6_rcv = 0;

int ry1_tx = 31;

int ry2_tx = 33;

int ry3_tx = 35;

int ry4_tx = 37;

int ry5_tx = 39;

int ry6_tx = 41;

void
setup()//////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////  VOID SETUP
{
pinMode(ry1,INPUT);
```

```
    pinMode(ry100,OUTPUT); ////////////////////// FOR TEST
pinMode(ry2,INPUT);
    pinMode(ry200,OUTPUT); ////////////////////// FOR TEST
pinMode(ry3,INPUT);
pinMode(ry4,INPUT);
pinMode(ry5,INPUT);
pinMode(ry6,INPUT);
pinMode(ry1_tx,OUTPUT);
pinMode(ry2_tx,OUTPUT);
pinMode(ry3_tx,OUTPUT);
pinMode(ry4_tx,OUTPUT);
pinMode(ry5_tx,OUTPUT);
pinMode(ry6_tx,OUTPUT);
digitalWrite(ry1,LOW);
    digitalWrite(ry100,HIGH); ////////////////////// FOR TEST
digitalWrite(ry2,LOW);
    digitalWrite(ry200,HIGH); ////////////////////// FOR TEST
digitalWrite(ry3,LOW);
digitalWrite(ry4,LOW);
digitalWrite(ry5,LOW);
digitalWrite(ry6,LOW);
```

```
////////////////////////////////////////////////////////////////////////////// Ethernet

Ethernet.begin(mac, ip);  // initialize Ethernet device

server.begin();           // start to listen for clients

Serial.begin(9600);       // for diagnostics

    pinMode(RxD, INPUT);

    pinMode(TxD, OUTPUT);

    mySerial.begin(9600);

}

void

loop()//////////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////  VOID LOOP

{

ry1_rcv =digitalRead(ry1);

ry2_rcv =digitalRead(ry2);

ry3_rcv =digitalRead(ry3);

ry4_rcv =digitalRead(ry4);

ry5_rcv =digitalRead(ry5);

ry6_rcv =digitalRead(ry6);

/*if(ry1_rcv==HIGH)

  {

    digitalWrite(ry1_tx,HIGH);
```

```
  }
if(ry1_rcv==LOW)
  {
     digitalWrite(ry1_tx,LOW);
  }
if(ry2_rcv==HIGH)
  {
     digitalWrite(ry2_tx,HIGH);
  }
if(ry2_rcv==LOW)
  {
     digitalWrite(ry2_tx,LOW);
  }
*/
////////////////////////////////////////////////////////////////////////////////
<Ethernet_HTML>
  EthernetClient client = server.available();
  if (client) {
     while (client.connected()) {
        if (client.available()) {
           char c = client.read();
```

```
//read char by char HTTP request
if (readString.length() < 100) {  /*Original 100 */
  //store characters to string
  readString += c;
  //Serial.print(c);
}
//if HTTP request has ended
if (c == '\n') {
  ///////////////
  Serial.println(readString); //print to serial monitor for debuging
  client.println("HTTP/1.1 200 OK"); //send new page
  client.println("Content-Type: text/html");
  client.println();
  client.println("<HTML>");
  client.println("<HEAD>");
  client.println("<TITLE>Arduino GET test page</TITLE>");
  client.println("</HEAD>");
  ////////////////////////////////////////////////////////////////////////////// <SUB
FUNCTION>
  client.println("<script>")
/*      //1111111111111111111111111111111111111111111111111111//
```

```
client.println("function doOnclick1() {");
client.println("document.getElementById('A').style.background ='#009900';");
client.println("document.getElementById('B').style.background ='#FFFFFF';");
//client.println("document.getElementById('M').style.background='#FFFFFF';");
//client.println("location.href='/?on2;' ;");
client.println("document.getElementById('A').style.background
='#009900';");
client.println("document.getElementById('B').style.background
='#FFFFFF';");
client.println("return flase;");
client.println("}");
client.println("function dropOnclick1() {");
client.println("document.getElementById('A').style.background ='#FFFFFF';");
client.println("document.getElementById('B').style.background='#CC0000';");
//client.println("document.getElementById('M').style.background='#FFFFFF';");
//client.println("location.href ='/?off3;';");
client.println("document.getElementById('A').style.background
='#FFFFFF';");

client.println("document.getElementById('B').style.background='#CC0000';");
client.println("return flase;");
client.println("}");
```

```
//22222222222222222222222222222222222222222222222222222//

client.println("function doOnclick2() {");

client.println("document.getElementById('C').style.background = '#009900';");

client.println("document.getElementById('D').style.background ='#FFFFFF';");

//client.println("document.getElementById('M').style.background='#FFFFFF';");

//client.println("location.href='/?on4;';");

client.println("return flase;");

client.println("}");

client.println("function dropOnclick2() {");

client.println("document.getElementById('C').style.background ='#FFFFFF';");

client.println("document.getElementById('D').style.background='#CC0000';");

//client.println("document.getElementById('M').style.background='#FFFFFF';");

//client.println("location.href='/?off5;';");

client.println("return flase;");

client.println("}");

//33333333333333333333333333333333333333333333333333333//

client.println("function doOnclick3() {");

client.println("document.getElementById('E').style.background = '#009900';");

client.println("document.getElementById('F').style.background ='#FFFFFF';");

//client.println("document.getElementById('M').style.background='#FFFFFF';");

client.println("document.location.href='/?on6;';");
```

```
client.println("return flase;");

client.println("}");

client.println("function dropOnclick3() {");

client.println("document.getElementById('E').style.background ='#FFFFFF';");

client.println("document.getElementById('F').style.background='#CC0000';");

//client.println("document.getElementById('M').style.background='#FFFFFF';");

client.println("document.location.href='/?off7;';");

client.println("return flase;");

client.println("}");

//4444444444444444444444444444444444444444444444444444444444//

client.println("function doOnclick4() {");

client.println("document.getElementById('G').style.background = '#009900';");

client.println("document.getElementById('H').style.background ='#FFFFFF';");

//client.println("document.getElementById('M').style.background='#FFFFFF';");

client.println("document.location.href='/?on8;';");

client.println("return flase;");

client.println("}");

client.println("function dropOnclick4() {");

client.println("document.getElementById('G').style.background ='#FFFFFF';");

client.println("document.getElementById('H').style.background='#CC0000';");

//client.println("document.getElementById('M').style.background='#FFFFFF';");
```

```
client.println("document.location.href='/?off9;';");

client.println("return flase;");

client.println("}");

//555555555555555555555555555555555555555555555555555555//

client.println("function doOnclick5() {");

client.println("document.getElementById('I').style.background = '#009900';");

client.println("document.getElementById('J').style.background ='#FFFFFF';");

//client.println("document.getElementById('M').style.background='#FFFFFF';");

client.println("document.location.href='/?onA;';");

client.println("return flase;");

client.println("}");

client.println("function dropOnclick5() {");

client.println("document.getElementById('I').style.background ='#FFFFFF';");

client.println("document.getElementById('J').style.background='#CC0000';");

//client.println("document.getElementById('M').style.background='#FFFFFF';");

client.println("document.location.href='/?offB;';");

client.println("return flase;");

client.println("}");

client.println("function doOnclick6() {");

client.println("document.getElementById('K').style.background = '#009900';");

client.println("document.getElementById('L').style.background ='#FFFFFF';");
```

```
        //client.println("document.getElementById('M').style.background='#FFFFFF';");

        client.println("document.location.href='/?onC;';");

        client.println("return flase;");

        client.println("}");

        client.println("function dropOnclick6() {");

        client.println("document.getElementById('K').style.background ='#FFFFFF';");

        client.println("document.getElementById('L').style.background='#CC0000';");

        //client.println("document.getElementById('M').style.background='#FFFFFF';");

        client.println("document.location.href='/?offD;';");

        client.println("return flase;");

        client.println("}");

*/

        client.println("</script>");

        ////////////////////////////////////////////////////////////////////////////// </SUB
FUNCTION>

        client.println("<style>");

        client.println("body {background-color:#FFAEB9;}");

        client.println("</style>");

        client.println("<BODY>");

        client.println("<CENTER><H1><style=font-size:40px;color:sienna>Home
Automation</H1></CENTER>");
```

```
////////////////////////////////////////////////////////////////////////////////

// For simple testing, pin 5, 6, 7, and 8 are used in buttons

// custom buttons 1
//1111111111111111111111111111111111111111111111111111//

client.print("<center><P>SWITCH 1</P></center>");

if (ry1_rcv==HIGH){

client.println("<CENTER><input type=submit value=ON id=\"A\"
style=background-color:green;font-size:13px;width:100px;height:45px
onClick=location.href='/?on2;'>");

client.println("<input type=submit value=OFF id=\"B\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?off3;'></center>");

//doOnclick1();

//client.println("<CENTER><input type=submit value=\"\"
style=background-color:green; width:60px;height:45px></CENTER>");

//client.println ("<input type=submit value=ON  style=background-
color:green;font-size:110%;width:100px;height:45px onClick=location.href='/?on2;'>");

//client.println ("<input type=submit value=OFF style=background-
color:White;font-size:110%;width:100px;height:45px</DIV>");

} if (ry1_rcv==LOW){

client.println("<center><input type=submit value=ON id=\"A\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?on2;'>");
```

```
client.println("<input type=submit value=OFF id=\"B\"
style=background-color:red;font-size:13px;width:100px;height:45px
onClick=location.href='/?off3;'></center>");

//dropOnclick1();

//client.println ("<CENTER><input type=submit value=\"\"
style=background-color:red; width:60px;height:45px></CENTER>");

//client.println ("<input type=submit value=ON  style=background-
color:white;font-size:110%;width:100px;height:45px >");

//client.println ("<input type=submit value=OFF style=background-
color:red;font-size:110%;width:100px;height:45px
onClick=location.href='/?off3;'></DIV>");

}

//client.println ("<CENTER><input type=submit value=ON id=\"A\" style=font-
size:110%;width:100px;height:45px onClick=location.href='/?on2;'>");

//client.println ("<input type=submit value=OFF id=\"B\" style=font-
size:110%;width:100px;height:45px onClick=location.href='/?off3;'></CENTER></P>");

//client.print(" <input type=submit value='ALL OFF'
style=width:100px;height:45px onClick=location.href='/?off3579;'><br><br>");

// custom buttons 2
//2222222222222222222222222222222222222222222222222222//

client.print("<center><P>SWITCH 2</P></center>");

if (ry2_rcv==HIGH){
```

```
client.println("<CENTER><input type=submit value=ON id=\"C\"
style=background-color:green;font-size:13px;width:100px;height:45px
onClick=location.href='/?on4;'>");

client.println("<input type=submit value=OFF id=\"D\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?off5;'></center>");

} if (ry2_rcv==LOW){

client.println("<center><input type=submit value=ON id=\"C\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?on4;'>");

client.println("<input type=submit value=OFF id=\"D\"
style=background-color:red;font-size:13px;width:100px;height:45px
onClick=location.href='/?off5;'></center>");

}

//client.println ("<CENTER><input type=submit value=ON id=\"C\" style=font-
size:110%;width:100px;height:45px onClick=location.href='/?on4;'>");

//client.println ("<input type=submit value=OFF id=\"D\" style=font-
size:110%;width:100px;height:45px onClick=location.href='/?off5;'></CENTER></P>");

//client.print(" <input type=submit value='ALL OFF'
style=width:100px;height:45px onClick=location.href='/?off3579;'><br><br>");

// custom buttons 3
//3333333333333333333333333333333333333333333333333333333333//
```



```
client.print("<center><P>SWITCH 3</P></center>");

if (ry3_rcv==HIGH){
```

```
client.println("<CENTER><input type=submit value=ON id=\"E\"
style=background-color:green;font-size:13px;width:100px;height:45px
onClick=location.href='/?on6;'>");

client.println("<input type=submit value=OFF id=\"F\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?off7;'></center>");

} if (ry3_rcv==LOW){

client.println("<center><input type=submit value=ON id=\"E\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?on6;'>");

client.println("<input type=submit value=OFF id=\"F\"
style=background-color:red;font-size:13px;width:100px;height:45px
onClick=location.href='/?off7;'></center>");

}

//client.println ("<CENTER><input type=submit value=ON id=\"E\" style=font-
size:110%;width:100px;height:45px onClick=\"doOnclick3()\">");

//client.println ("<input type=submit value=OFF id=\"F\" style=font-
size:110%;width:100px;height:45px onClick=\"dropOnclick3()\"></CENTER></P>");

//client.print(" <input type=submit value='ALL OFF'
style=width:100px;height:45px onClick=location.href='/?off3579;'><br><br>");

// custom buttons 4
//4444444444444444444444444444444444444444444444444444444//
client.print("<center><P>SWITCH 4</P></center>");
if (ry4_rcv==HIGH){
```

```
client.println("<CENTER><input type=submit value=ON id=\"G\"
style=background-color:green;font-size:13px;width:100px;height:45px
onClick=location.href='/?on8;'>");

client.println("<input type=submit value=OFF id=\"H\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?off9;'></center>");

} if (ry4_rcv==LOW){

client.println("<center><input type=submit value=ON id=\"G\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?on8;'>");

client.println("<input type=submit value=OFF id=\"H\"
style=background-color:red;font-size:13px;width:100px;height:45px
onClick=location.href='/?off9;'></center>");

}

//client.println ("<CENTER><input type=submit value=ON id=\"G\" style=font-
size:110%;width:100px;height:45px onClick=\"doOnclick4()\">");

//client.println ("<input type=submit value=OFF id=\"H\" style=font-
size:110%;width:100px;height:45px onClick=\"dropOnclick4()\"></CENTER></P>");

//client.print(" <input type=submit value='ALL OFF'
style=width:100px;height:45px onClick=location.href='/?off3579;'><br><br>");

// custom buttons 5
//555555555555555555555555555555555555555555555555555555//
client.print("<center><P>SWITCH 5</P></center>");

if (ry5_rcv==HIGH){
```

```
client.println("<CENTER><input type=submit value=ON id=\"I\"
style=background-color:green;font-size:13px;width:100px;height:45px
onClick=location.href='/?onA;'>");

client.println("<input type=submit value=OFF id=\"J\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?offB;'></center>");

} if (ry5_rcv==LOW){

client.println("<center><input type=submit value=ON id=\"I\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?onA;'>");

client.println("<input type=submit value=OFF id=\"J\"
style=background-color:red;font-size:13px;width:100px;height:45px
onClick=location.href='/?offB;'></center>");

}

//client.println ("<CENTER><input type=submit value=ON id=\"I\" style=font-
size:110%;width:100px;height:45px onClick=\"doOnclick5()\">");

//client.println ("<input type=submit value=OFF id=\"J\" style=font-
size:110%;width:100px;height:45px onClick=\"dropOnclick5()\"></CENTER></P>");

//client.print(" <input type=submit value='ALL OFF'
style=width:100px;height:45px onClick=location.href='/?off3579;'><br><br>");

// custom buttons 6
//6666666666666666666666666666666666666666666666666666666666//
client.print("<center><P>SWITCH 6</P></center>");

if (ry6_rcv==HIGH){
```

```
client.println("<CENTER><input type=submit value=ON id=\"K\"
style=background-color:green;font-size:13px;width:100px;height:45px
onClick=location.href='/?onC;'>");

client.println("<input type=submit value=OFF id=\"L\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?offD;'></center>");

} if (ry6_rcv==LOW){

client.println("<center><input type=submit value=ON id=\"K\"
style=background-color:white;font-size:13px;width:100px;height:45px
onClick=location.href='/?onC;'>");

client.println("<input type=submit value=OFF id=\"L\"
style=background-color:red;font-size:13px;width:100px;height:45px
onClick=location.href='/?offD;'></center>");

}

//client.println ("<CENTER><input type=submit value=ON id=\"K\" style=font-
size:110%;width:100px;height:45px onClick=\"doOnclick6()\">");

//client.println ("<input type=submit value=OFF id=\"L\" style=font-
size:110%;width:100px;height:45px onClick=\"dropOnclick6()\"></CENTER></P>");

//client.print("<CENTER> <input type=submit value='ALL OFF'
style=width:200px;height:45px
onClick=location.href='/?off3579;'></CENTER><br><br>");

client.println("</BODY>");

client.println("</HTML>");

delay(1);
```

```
//stopping client

client.stop();

////////////////////// control arduino pin

//1111111111111111111111111111111111111111111111111111//

if(readString.indexOf('2') >0)//checks for 2

{

  digitalWrite(5, HIGH);    // set pin 5 high

  Serial.println("Led 5 On");

  char incoming  = 15;

  mySerial.write(incoming);

        digitalWrite(ry1_tx,HIGH);

}

if(readString.indexOf('3') >0)//checks for 3

{

  digitalWrite(5, LOW);    // set pin 5 low

  Serial.println("Led 5 Off");

  char incoming  = 25;

  mySerial.write(incoming);

        digitalWrite(ry1_tx,LOW);

}

//2222222222222222222222222222222222222222222222222222//
```

```
if(readString.indexOf('4') >0)//checks for 4
{
  digitalWrite(6, HIGH);    // set pin 6 high
  Serial.println("Led 6 On");
  char incoming  = 3;
  mySerial.write(incoming);
        digitalWrite(ry2_tx,HIGH);
}
if(readString.indexOf('5') >0)//checks for 5
{
  digitalWrite(6, LOW);    // set pin 6 low
  Serial.println("Led 6 Off");
  char incoming  = 4;
  mySerial.write(incoming);
        digitalWrite(ry2_tx,LOW);
}
//333333333333333333333333333333333333333333333333333333333//
 if(readString.indexOf('6') >0)//checks for 6
{
  digitalWrite(7, HIGH);    // set pin 7 high
  Serial.println("Led 7 On");
```

```
  char incoming  = 5;
  mySerial.write(incoming);
        digitalWrite(ry3_tx,HIGH);
}
if(readString.indexOf('7') >0)//checks for 7
{
  digitalWrite(7, LOW);    // set pin 7 low
  Serial.println("Led 7 Off");
  char incoming  = 6;
  mySerial.write(incoming);
        digitalWrite(ry3_tx,LOW);
}
//444444444444444444444444444444444444444444444444444444444//
  if(readString.indexOf('8') >0)//checks for 8
{
  digitalWrite(7, HIGH);    // set pin 8 high
  Serial.println("Led 8 On");
  char incoming  = 7;
  mySerial.write(incoming);
        digitalWrite(ry4_tx,HIGH);
}
```

```
if(readString.indexOf('9') >0)//checks for 9

{

  digitalWrite(7, LOW);    // set pin 8 low

  Serial.println("Led 8 Off");

  char incoming  = 8;

  mySerial.write(incoming);

        digitalWrite(ry4_tx,LOW);

}

 //55555555555555555555555555555555555555555555555555555555//

     if(readString.indexOf('A') >0)//checks for 8

{

  digitalWrite(7, HIGH);    // set pin 8 high

  Serial.println("Led 8 On");

  char incoming  = 11;

  mySerial.write(incoming);

        digitalWrite(ry5_tx,HIGH);

}

if(readString.indexOf('B') >0)//checks for 9

{

  digitalWrite(7, LOW);    // set pin 8 low

  Serial.println("Led 8 Off");
```

```
  char incoming  = 12;
  mySerial.write(incoming);
        digitalWrite(ry5_tx,LOW);
}
//6666666666666666666666666666666666666666666666666666666//
    if(readString.indexOf('C') >0)//checks for 8
{
  digitalWrite(7, HIGH);    // set pin 8 high
  Serial.println("Led 8 On");
  char incoming  = 13;
  mySerial.write(incoming);
        digitalWrite(ry6_tx,HIGH);
}
if(readString.indexOf('D') >0)//checks for 9
{
  digitalWrite(7, LOW);    // set pin 8 low
  Serial.println("Led 8 Off");
  char incoming  = 14;
  mySerial.write(incoming);
        digitalWrite(ry6_tx,LOW);
}
```

```
            //clearing string for next read
            readString="";
          }
        }
      }
    }
////////////////////////////////////////////////////////////////////////////////
</Ethernet_HTML>
}
```

[2] โปรแกรมภาครับ

```
/*#include <SoftwareSerial.h>

#define RxD 5

#define TxD 6*/

#define Relay_1 30

    #define Relay_10 31

#define Relay_2 32

    #define Relay_20 33

#define Relay_3 34

    #define Relay_30 35

#define Relay_4 36

    #define Relay_40 37

#define Relay_5 38

    #define Relay_50 39

#define Relay_6 40

    #define Relay_60 41

/*SoftwareSerial mySerial(RxD,TxD); // RX, TX  */

void setup() {

    Serial.begin(9600);

    Serial1.begin(9600);

    pinMode(Relay_1, OUTPUT);
```

```
    pinMode(Relay_10, OUTPUT);
pinMode(Relay_2, OUTPUT);
    pinMode(Relay_20, OUTPUT);
pinMode(Relay_3, OUTPUT);
    pinMode(Relay_30, OUTPUT);
pinMode(Relay_4, OUTPUT);
    pinMode(Relay_40, OUTPUT);
pinMode(Relay_5, OUTPUT);
    pinMode(Relay_50, OUTPUT);
pinMode(Relay_6, OUTPUT);
    pinMode(Relay_60, OUTPUT);
digitalWrite(Relay_1,HIGH);
    digitalWrite(Relay_10,LOW);
digitalWrite(Relay_2,HIGH);
    digitalWrite(Relay_20,LOW);
digitalWrite(Relay_3,HIGH);
    digitalWrite(Relay_30,LOW);
digitalWrite(Relay_4,HIGH);
    digitalWrite(Relay_40,LOW);
digitalWrite(Relay_5,HIGH);
    digitalWrite(Relay_50,LOW);
```

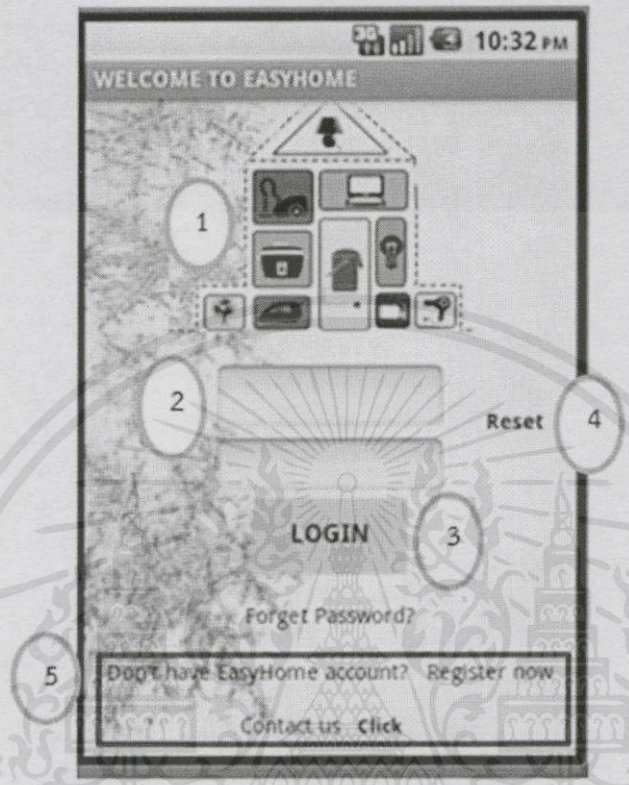
```
    digitalWrite(Relay_6,HIGH);
        digitalWrite(Relay_60,LOW);
}
void loop() {
    if (Serial1.available()) {
        char incoming  = Serial1.read();
        Serial1.write(incoming);
        Serial1.println(" Command : ");
        switch(incoming) {
            case 15 :  digitalWrite(Relay_1,LOW);
                       Serial1.println("Relay 1 Energize");
                       digitalWrite(Relay_10,HIGH);
                       break;
            case 25 :  digitalWrite(Relay_1,HIGH);
                       Serial1.println("Relay 1 De-energize");
                       digitalWrite(Relay_10,LOW);
                       break;
            case 3 :  digitalWrite(Relay_2,LOW);
                       Serial1.println("Relay 2 Energize");
                       digitalWrite(Relay_20,HIGH);
                       break;
```

```
case 4 :  digitalWrite(Relay_2,HIGH);
          Serial1.println("Relay 2 De-energize");
          digitalWrite(Relay_20,LOW);
          break;
case 5 :  digitalWrite(Relay_3,LOW);
          Serial1.println("Relay 3 Energize");
          digitalWrite(Relay_30,HIGH);
          break;
case 6 :  digitalWrite(Relay_3,HIGH);
          Serial1.println("Relay 3 De-energize");
          digitalWrite(Relay_30,LOW);
          break;
case 7 :  digitalWrite(Relay_4,LOW);
          Serial1.println("Relay 4 Energize");
          digitalWrite(Relay_40,HIGH);
          break;
case 8 :  digitalWrite(Relay_4,HIGH);
          Serial1.println("Relay 4 De-energize");
          digitalWrite(Relay_40,LOW);
          break;
case 11 :  digitalWrite(Relay_5,LOW);
```

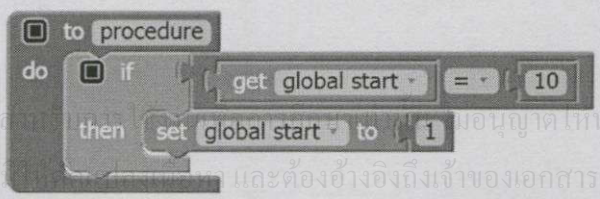
```
                Serial1.println("Relay 5 Energize");
                digitalWrite(Relay_50,HIGH);
                break;
        case 12 :  digitalWrite(Relay_5,HIGH);
                Serial1.println("Relay 5 De-energize");
                digitalWrite(Relay_50,LOW);
                break;
        case 13 :  digitalWrite(Relay_6,LOW);
                Serial.println("Relay 6 Energize");
                digitalWrite(Relay_60,HIGH);
                break;
        case 14 :  digitalWrite(Relay_6,HIGH);
                Serial.println("Relay 6 De-energize");
                digitalWrite(Relay_60,LOW);
                break;
}
}
if (Serial.available()) {
    char incoming_2 = Serial.read();
    Serial1.write(incoming_2);
}
```

[3] โปรแกรมสร้างแอพพลิเคชั่น

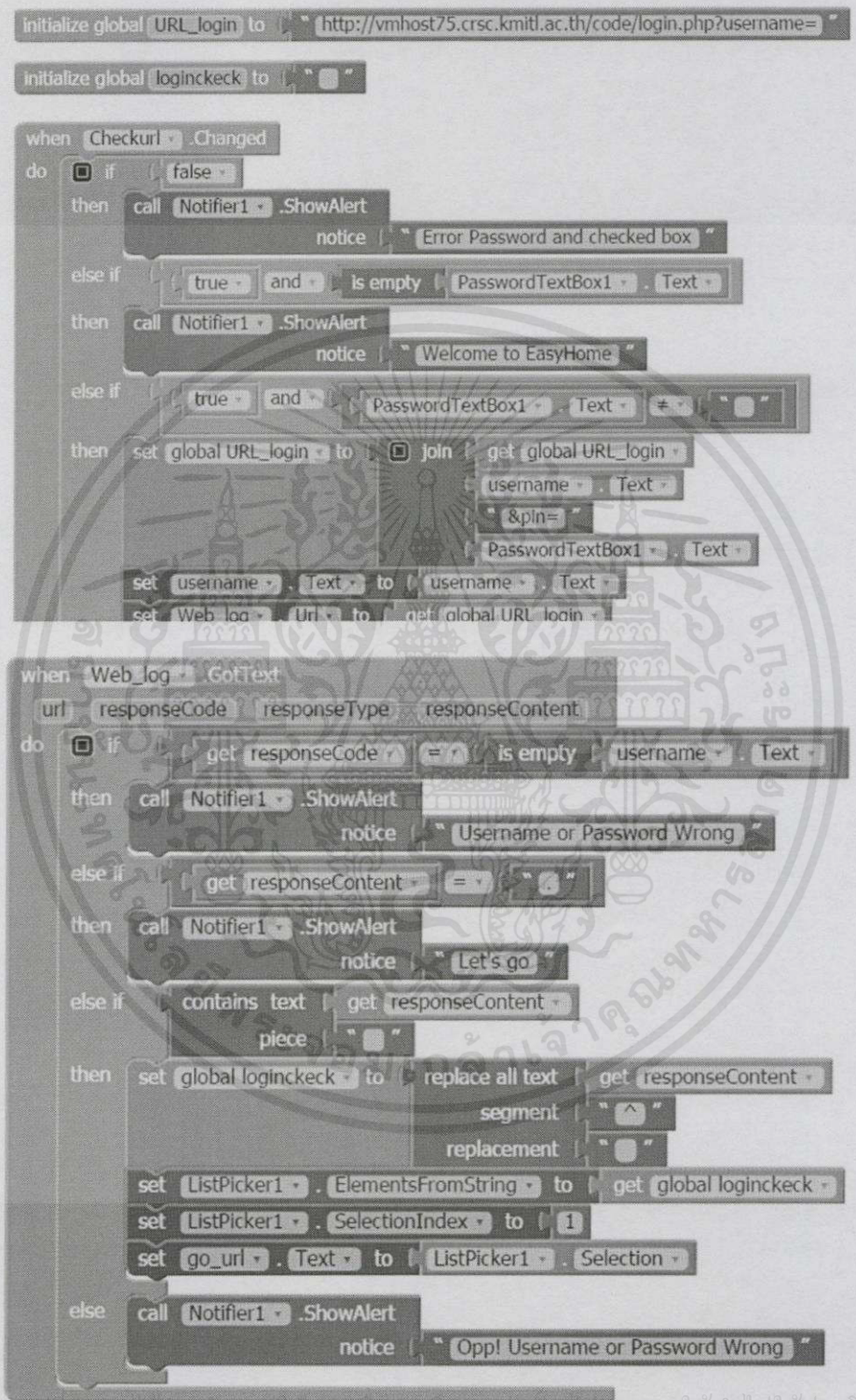
หน้าล็อคอิน (Login)



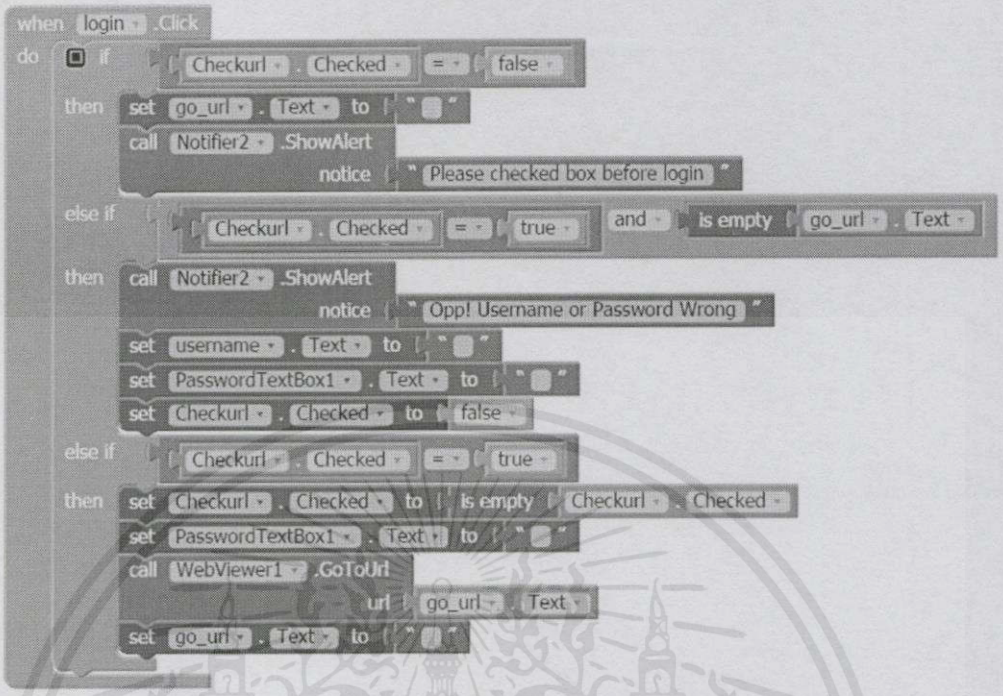
1. Animation หน้าจอ (ทำให้ภาพเคลื่อนไหว)



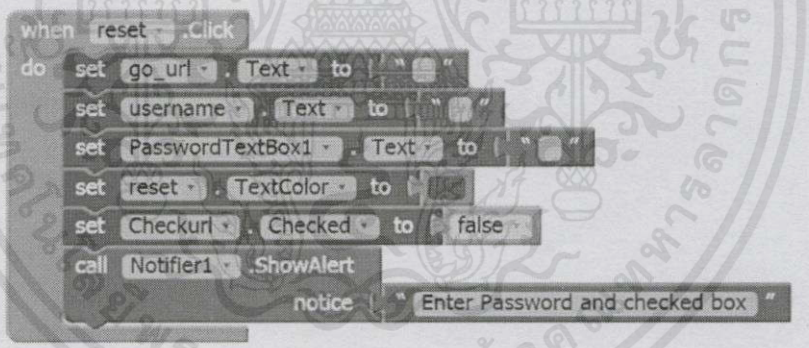
2. กด BOX เพื่อเช็คไอดีผู้ใช้งาน (Username) และ รหัสผ่าน (Password) ถ้าถูกต้องจะดึง URL จากฐานข้อมูลของผู้ใช้งานออกมา



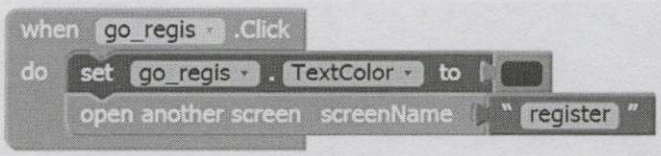
3. ปุ่ม LOGIN เพื่อแสดง action ใน URL ของผู้ใช้ ซึ่งจะใช้ในการสั่งการเปิด-ปิดอุปกรณ์ไฟฟ้า

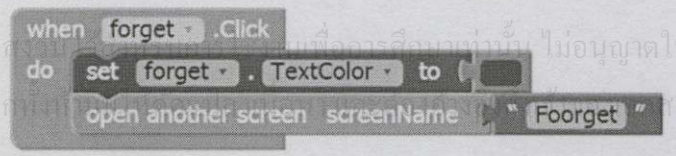


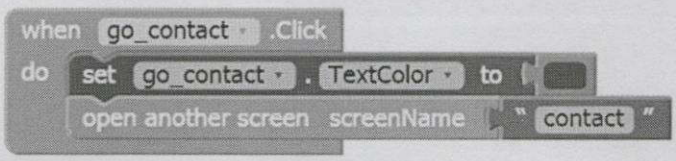
4. ปุ่ม RESET เป็นการคืนค่าทั้งหมดให้กับหน้าเพจแอพพลิเคชั่น



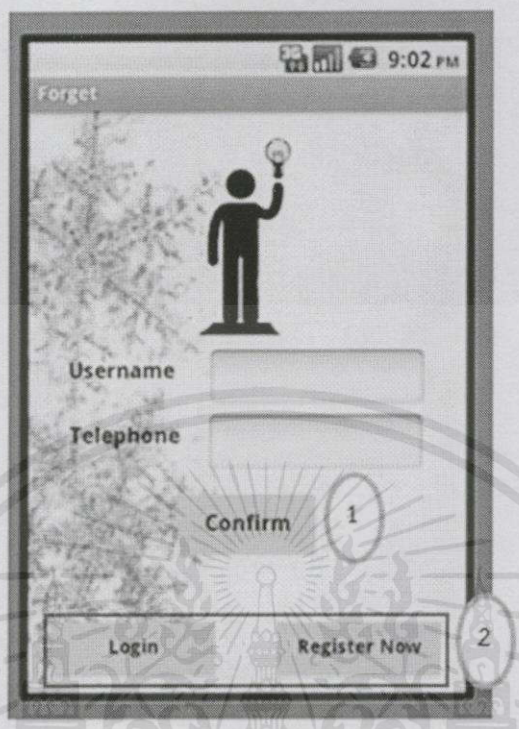
5. ปุ่ม click เพื่อเข้าสู่หน้า REGISTER, FORGET และ CONTACT ตามลำดับ



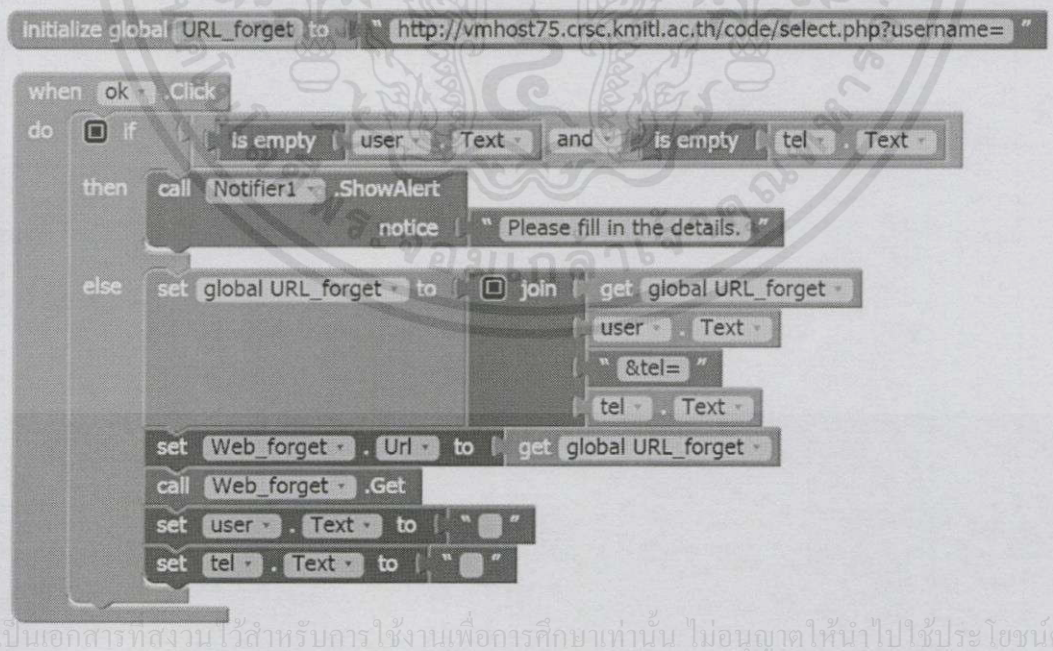


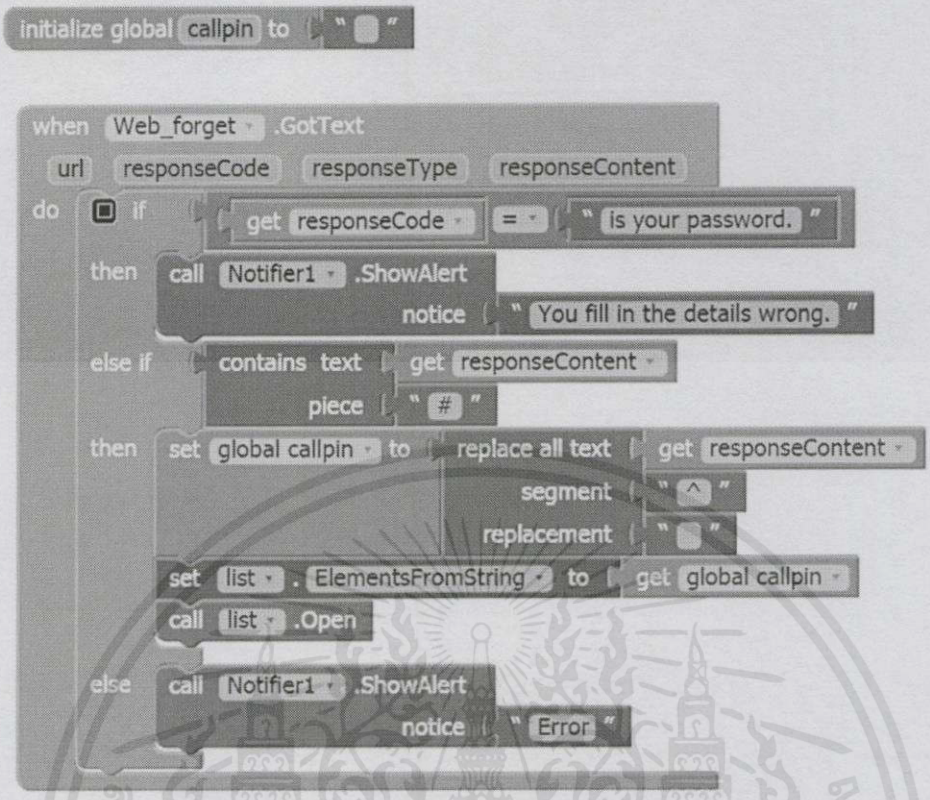


หน้าลืมรหัสผู้ใช้ (FORGET)



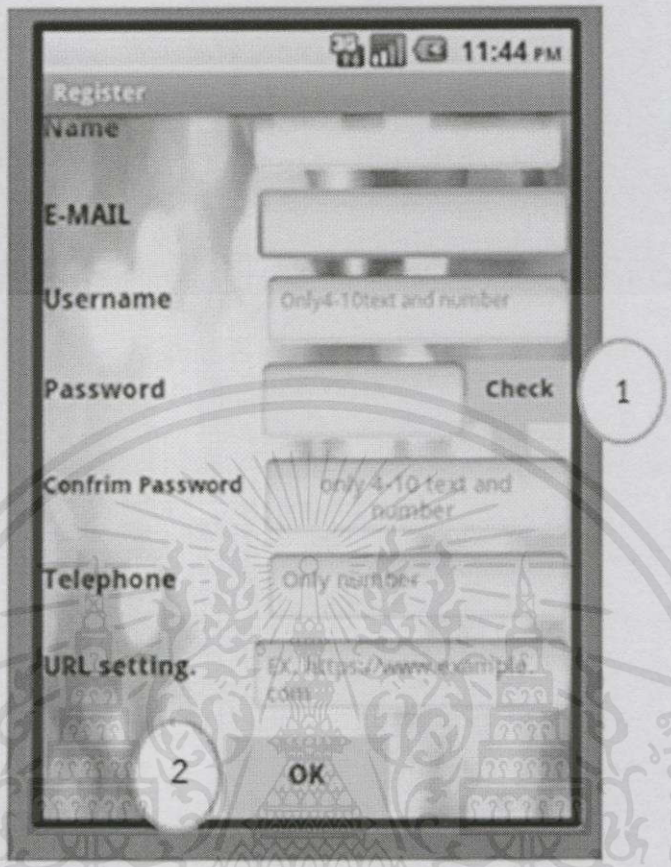
1. ปุ่ม CONFIRM เพื่อเช็คไอดีผู้ใช้งาน (Username) และเบอร์โทรศัพท์ที่ผู้ใช้ได้ลงทะเบียน ถ้าถูกต้องจะดึงรหัสผ่านจากฐานข้อมูลของผู้ใช้งานออกมา และแสดงในหน้าถัดไป



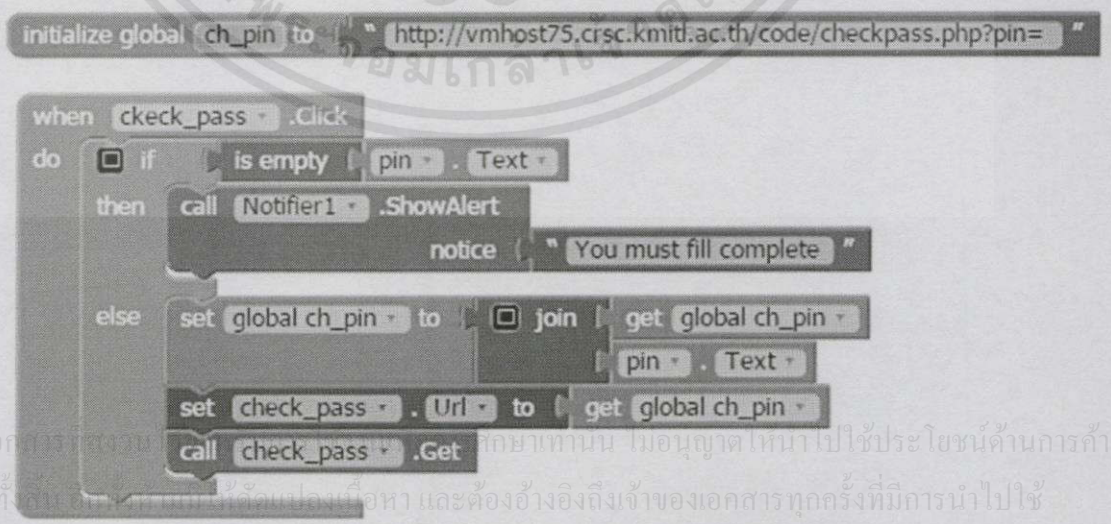


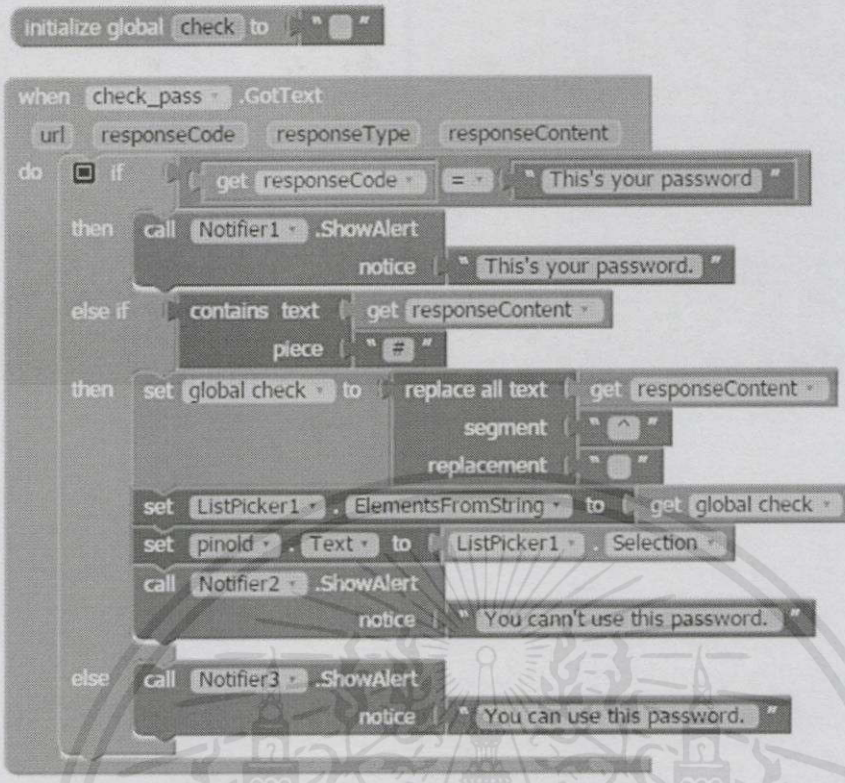
2. ปุ่มคลิกเพื่อเข้าสู่หน้า LOGIN และ REGISTER

หน้าลงทะเบียนผู้ใช้งาน (Register)

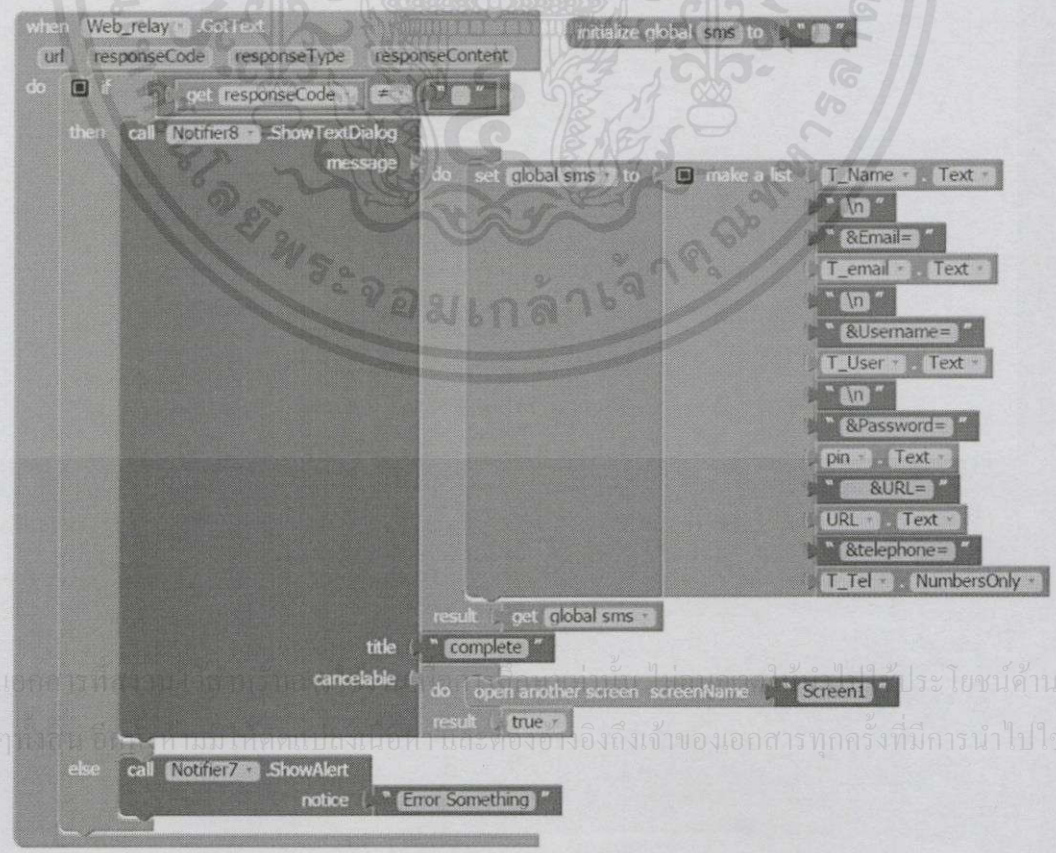


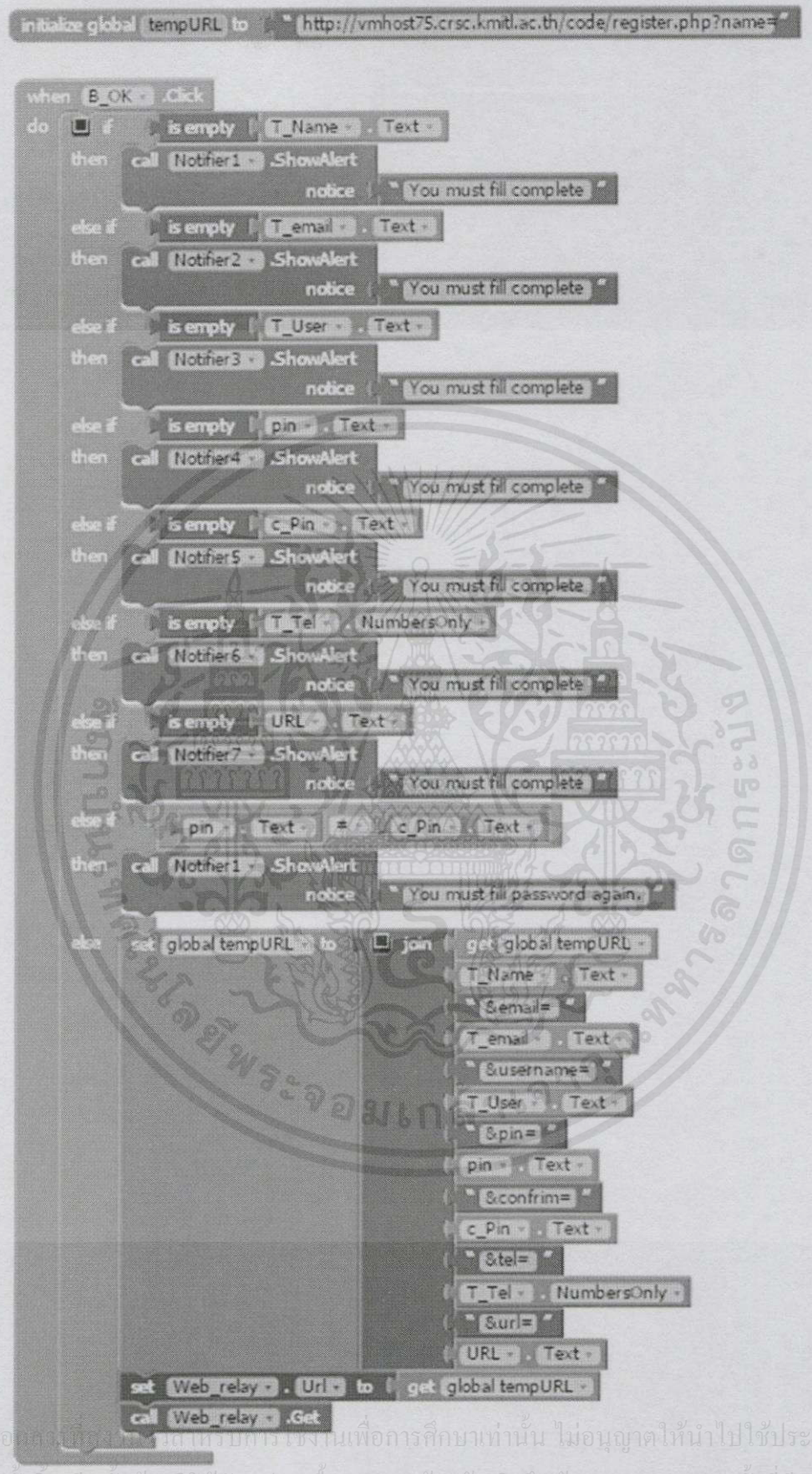
1. ปุ่มCHECK เพื่อเช็ครหัสผู้ใช้งาน ว่าซ้ำกับในฐานข้อมูลหรือไม่ ถ้าไม่ซ้ำผู้ใช้จะสามารถใช้รหัสผ่านนี้ได้





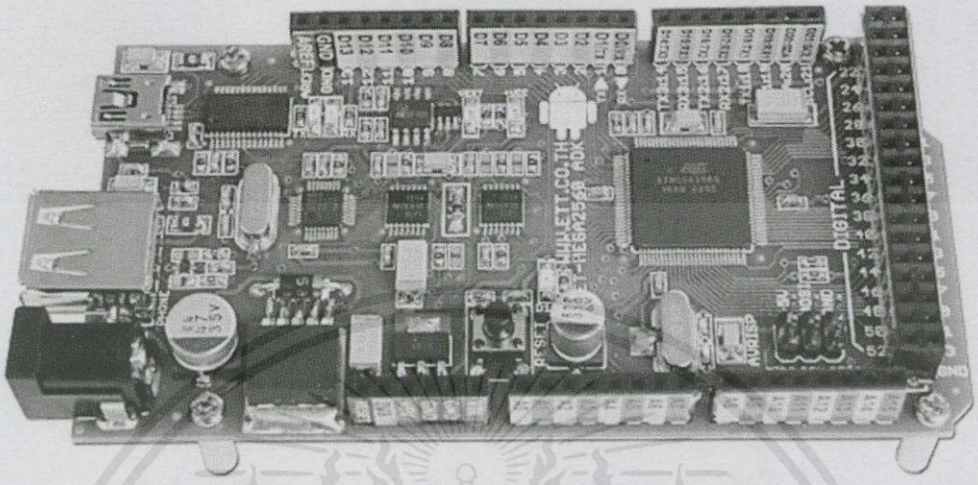
2. ปุ่ม OK เพื่อยืนยันการลงทะเบียน ถ้ากรอกรายละเอียดครบถ้วน และถูกต้อง ก็จะสามารถนำข้อมูลข้างต้นมาขอรับการใช้งานในหน้า LOGIN ต่อไปได้





ภาคผนวก ข.

ET-MEGA2560-ADK



จากการที่ Arduino ที่เป็นโครงการพัฒนาระบบ MCU ของ AVR แบบ Open Source ได้รับการแนะนำเผยแพร่ออกมาสู่สาธารณะ ซึ่งได้รับความนิยมกันอย่างแพร่หลายจากผู้คนทั่วโลก ภายในระยะเวลาอันรวดเร็ว ทางด้านของ Software ก็มีการพัฒนากันอย่างต่อเนื่อง ซึ่งในขณะนี้ (กันยายน 2554) โปรแกรมของ Arduino ได้รับการปรับปรุงเป็น Version "arduino-0022" แล้ว โดยทางด้าน Hardware เองก็ได้มีการพัฒนาปรับปรุงอย่างต่อเนื่องควบคู่กันไปด้วยเช่นเดียวกัน ซึ่งจากเดิมที่มีการพัฒนาโปรแกรมให้รองรับกับการใช้งานได้กับชิพ MCU รุ่นเล็ก 28 ขา อย่าง ATMEGA8,ATMEGA88/ATMEGA168/ATMEGA328 และพัฒนาต่อมาจนเป็นรุ่นใหญ่แบบ 100Pin อย่าง ATMEGA1280/ATMEGA2560 ตามลำดับ

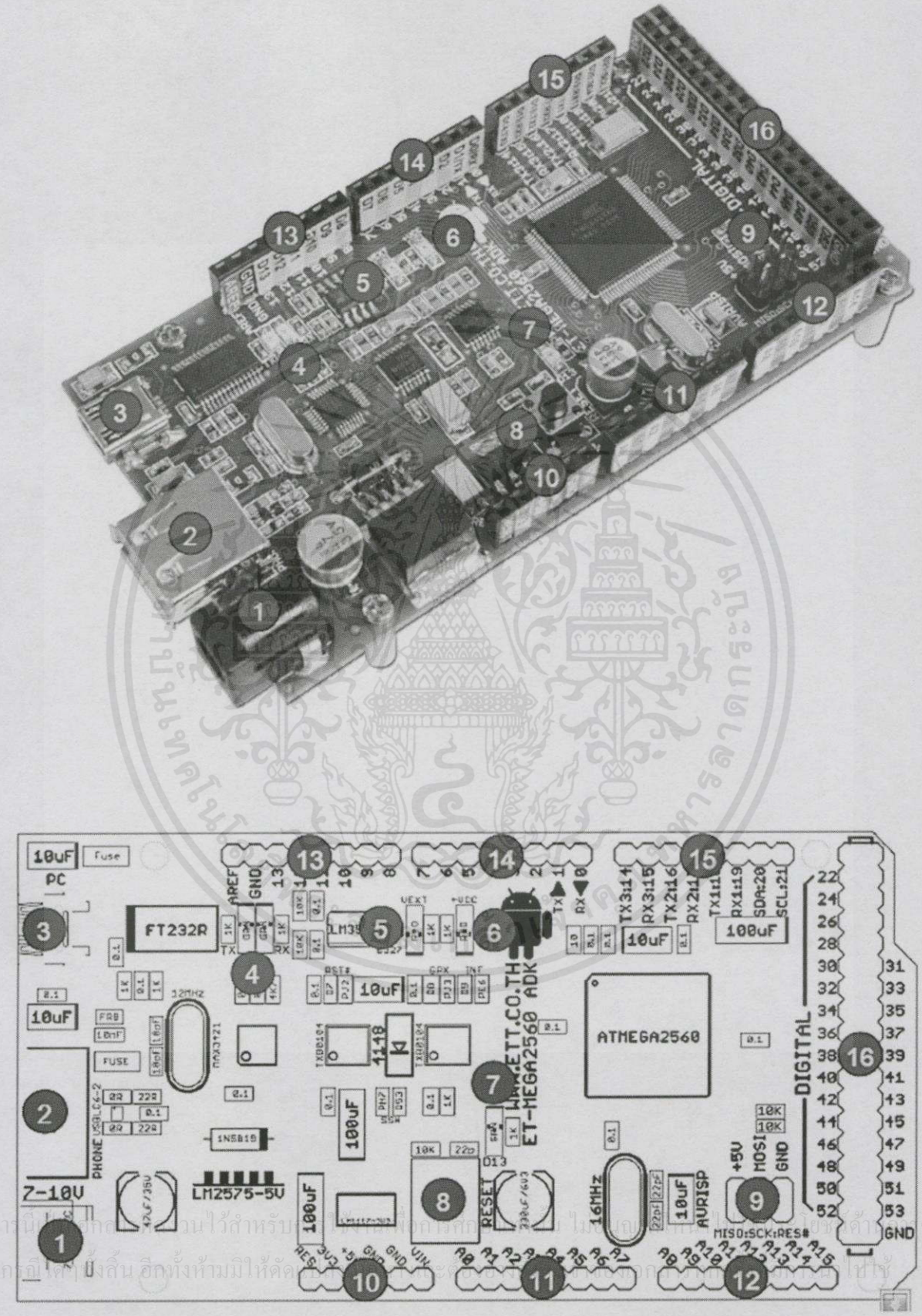
และล่าสุดได้มีการพัฒนาขีดความสามารถของ Arduino บน AVR ให้สามารถเชื่อมต่อกับอุปกรณ์ USB Host ได้ ทำให้สามารถนำ Arduino ไปดัดแปลงประยุกต์เชื่อมต่อกับอุปกรณ์ USB Device แบบต่างๆ เช่น USB HID Keyboard หรือ USB HID Mouse เป็นต้น และที่น่าตื่นเต้นและน่ายินดีเป็นอย่างยิ่งก็คือการนำเอา Arduino ไปประยุกต์เชื่อมต่อกับสมาร์ทโฟนที่ใช้ระบบปฏิบัติการแอนดรอยด์ (Android) ซึ่งเป็นระบบปฏิบัติการแบบ โอเพ่นซอร์ส จากค่าย Google ยักษ์ใหญ่ด้านเวปบราวเซอร์ของโลก ซึ่งเป็นที่รู้จักกันในแวดวงผู้ใช้ ในนามของ แอนดรอยด์โฟน ซึ่งทำให้เราสามารถเชื่อมต่อสื่อสารสั่งงานบอร์ด Arduino ผ่านอุปกรณ์ แอนดรอยด์โฟน ได้ ซึ่งนับเป็นพัฒนาการอีกขั้นของ Arduino บน AVR ที่ได้รับการพัฒนาขึ้นมา ทำให้ Arduino มีความโดดเด่นและน่าสนใจมากยิ่งขึ้นไปอีก

และในวันนี้ทาง อีทีที จึงได้นำชิพ MCU ตระกูล AVR เบอร์ ATMEGA2560 และ MAX3421 มาพัฒนาเป็นบอร์ด Arduino แบบมี USB Host เพื่อรองรับการเชื่อมต่ออุปกรณ์ USB Device และ อุปกรณ์แอนดรอยด์โฟน โดยใช้ชื่อว่า "ET-MEGA2560-ADK" โดยได้ออกแบบให้มีการจัดสรร Pin I/O ต่างๆ รวมทั้งขนาดให้ตรงตามมาตรฐานของบอร์ด "Arduino Mega" เพียงแต่ได้มีการเพิ่ม USB Host และปรับปรุงข้อจำกัดบางอย่างให้ดียิ่งขึ้น เพื่อเพิ่มความสะดวกกับผู้ใช้งานมากยิ่งขึ้น

คุณสมบัติของบอร์ด

- ใช้ ATMEGA2560 เป็น MCU ประจำบอร์ด Run ความถี่ 16MHz จาก Crystal Oscillator
- 256KByte Flash(สงวนไว้ 4KByte สำหรับ Bootloader) / 8KByte SRAM / 4KByte EEPROM
- รองรับการพัฒนาโปรแกรมด้วยภาษา C++ ของ Arduino ตามแบบ Arduino Mega ได้ 100%
- ใช้ USB Bridge ของ FTDI เบอร์ FT232RL พร้อม Over Current Protection สำหรับติดต่อสื่อสารและ Download Code จากคอมพิวเตอร์ให้บอร์ด โดยไม่ต้องใช้เครื่องโปรแกรมจากภายนอก
- On Board USB Host(MAX3421) สำหรับเชื่อมต่อ USB Device หรืออุปกรณ์ Android ADK
    - รองรับการพัฒนาโปรแกรมด้วย ADK (Android Open Accessories development Kit) โดยใช้ Google Open Accessories API เมื่อใช้กับอุปกรณ์แอนดรอยด์ที่ได้รับการติดตั้งระบบปฏิบัติการแอนดรอนด์ V2.3.4 หรือสูงกว่า
    - รองรับการพัฒนาโปรแกรมด้วย Android Debug Bridge (ADB) โดยใช้ Library ของ Microbridge เมื่อใช้กับอุปกรณ์แอนดรอยด์ที่ได้รับการติดตั้งระบบปฏิบัติการแอนดรอนด์ V1.5 หรือสูงกว่า
- 54 Pin Digital I/O โดยมี
    - 16 Pin Analog Input (ADC ขนาด 10 บิต 16 ช่อง)
    - 14 PWM outputs
    - 4 UART(Hardware Serial Port) แบบ TTL Logic
    - 1 Hardware TWI (I2C)
    - 1 Hardware SPI (up to 8Mbps)
- ขนาดของ PCB บอร์ด และ ตำแหน่ง Pin Connector ต่างๆ ตรงกันกับ Arduino Mega ทั้งหมด ทำให้สามารถนำไปติดตั้งใช้งานร่วมกับบอร์ด Shield แบบต่างๆที่มีการผลิตขึ้นมาใช้งานร่วมกันกับบอร์ด Arduino Mega ได้ทั้งหมด โดยบอร์ดมีขนาด PCB Size 5.3cm x 10.2cm
- รองรับการใช้งานกับ External Supply ทั้งแบบ AC และ DC ขนาด 7-12V โดยเลือกใช้ Regulate แบบ Switching ขนาด 1A (LM2575-5V) ลดปัญหาเรื่องความร้อนเมื่อมีการใช้กระแสสูงๆ สามารถใช้แหล่งจ่ายจากพอร์ต USB ได้ในกรณีใช้กระแสไม่เกิน 500mA โดยมีวงจรเลือกแหล่งจ่ายอัตโนมัติ โดยจะตัดการใช้ไฟเลี้ยงจาก USB โดยอัตโนมัติ เมื่อมีการต่อแหล่งจ่ายจากภายนอกให้บอร์ด

โครงสร้างบอร์ด ET-MEGA2560-ADK



รูปแสดง โครงสร้างของบอร์ด ET-MEGA2560-ADK

- **หมายเลข 1** คือ ขั้วต่อแหล่งจ่ายไฟเลี้ยงจากภายนอก สามารถใช้ได้กับแหล่งจ่ายทั้งแบบ AC และ DC พร้อมวงจร Bridge Rectifier และ Regulate แบบ Switching ช่วยลดความร้อนของ IC Regulate เมื่อมีการดึงกระแสมากๆได้เป็นอย่างดี สามารถใช้กับแรงดันInput 7-12V
- **หมายเลข 2** เป็นขั้วต่อ USB Host สำหรับเชื่อมต่อกับอุปกรณ์USB Device ต่างๆ
- **หมายเลข 3** เป็นขั้วต่อ USB Device สำหรับติดต่อสื่อสารกับคอมพิวเตอร์ PC โดยใช้ FT232RL เป็น USB Bridge ในการเชื่อมต่อระหว่างคอมพิวเตอร์ PC และ MCU ในบอร์ต และยังสามารถใช้ไฟจาก พอร์ต USB เป็นแหล่งจ่ายให้กับบอร์ดได้ด้วย โดยจะมี Poly Fuse ขนาด 500mA สำหรับป้องกันการดึงกระแสเกินจากพอร์ต USB ด้วย และที่พิเศษคือมีวงจรสำหรับตรวจสอบแหล่งจ่ายเพื่อสลับการใช้งานแหล่งจ่ายจาก USB ไปเป็น External Supply ได้เอง โดยอัตโนมัติ โดยเมื่อไม่ได้ต่อ External Supply บอร์ดจะใช้ไฟจากพอร์ต USB เป็นแหล่งจ่ายในการทำงาน แต่เมื่อมีการต่อ External Supply วงจรจะสลับไปใช้แหล่งจ่ายจาก External Supply เองโดยอัตโนมัติ
    - LED +VCC ใช้แสดงสถานะเมื่อมีการจ่ายไฟให้กับบอร์ด
    - LED VEXT ใช้แสดงสถานะเมื่อมีการจ่ายไฟจาก External Supply
- **หมายเลข 4** เป็น LED VEXT ใช้แสดงสถานะเมื่อมีการจ่ายไฟเลี้ยงจาก External Supply
- **หมายเลข 5** เป็น LED +VCC ใช้แสดงสถานะของแหล่งจ่ายไฟเลี้ยง (+VCC) ของบอร์ด โดยเมื่อบอร์ดใช้แหล่งจ่ายจาก External Supply จะแสดงสถานะโดยการให้ LED VEXT และ LED +VCC ติดสว่างพร้อมกันทั้งคู่ แต่ถ้าบอร์ดใช้แหล่งจ่ายจากพอร์ต USB จะแสดงสถานะโดยการให้ LED +VCC ติดสว่างเพียงดวงเดียว
- **หมายเลข 6** เป็น LED แสดงสถานะของ RX และ TX ใช้สำหรับแสดงการรับส่งข้อมูลระหว่างบอร์ด ET-MEGA2560-ADK กับคอมพิวเตอร์ PC ผ่านทางพอร์ต USB
- **หมายเลข 7** เป็น LED D13 ใช้สำหรับทดแสดงการทำงานของ Bootloader และ ใช้ทดสอบการทำงานของบอร์ดจากการควบคุมของ Pin Digital-13 ทำงานด้วย Logic "1" และ หยุดทำงานด้วย Logic "0"
- **หมายเลข 8** เป็นสวิตช์ Reset ใช้สำหรับสั่ง Reset การทำงานของบอร์ด
- **หมายเลข 9** เป็นขั้วต่อ AVRISP ใช้สำหรับโปรแกรม Bootloader ให้กับ MCU
- **หมายเลข 10** เป็นขั้วต่อ Power
- **หมายเลข 11,12** เป็นขั้วต่อสัญญาณ Analog A[0..7] และ Analog A[8..15] ตามลำดับ
- **หมายเลข 13,14,15** เป็นขั้วต่อสัญญาณ Digital D[0..7],D[8..13] และ D[14..21]
- **หมายเลข 16** เป็นขั้วต่อสัญญาณ Digital D[22..53]

คุณสมบัติของสัญญาณต่าง ๆของบอร์ด ET-MEGA2560-ADK

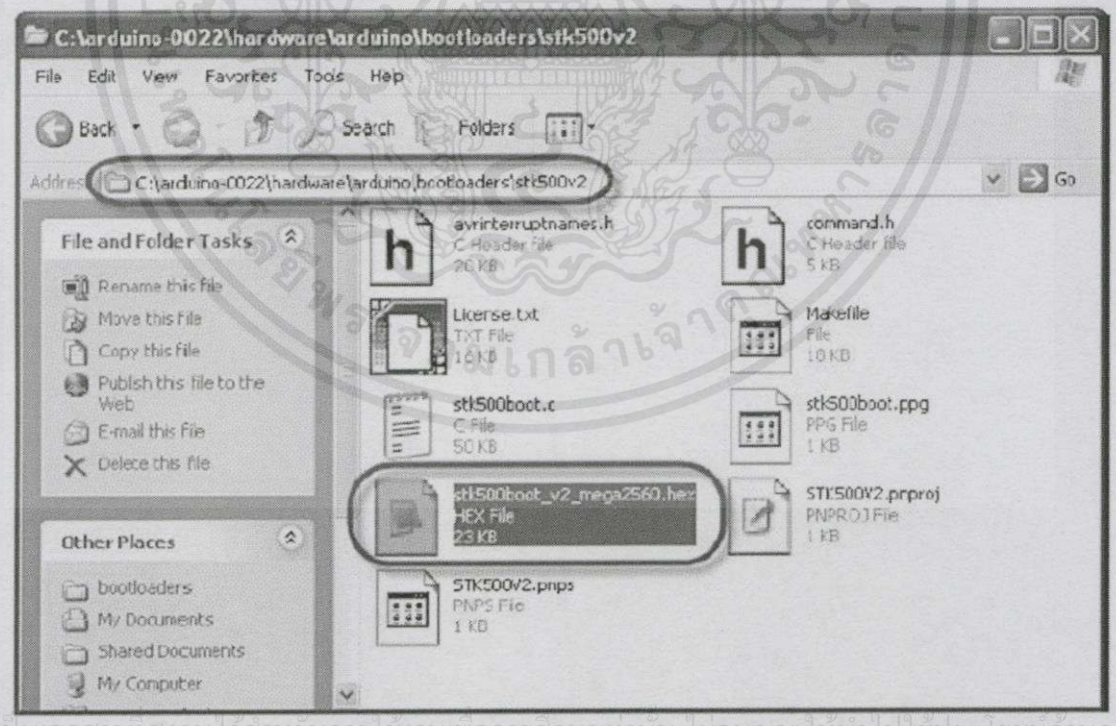
- RESET# เป็นสัญญาณ Input Reset ของ MCU ทำงานเมื่อเป็น Logic Low โดยสัญญาณ RESET# นี้จะถูกควบคุมจาก 2 แหล่ง คือ จาก สวิตช์ RESET ภายในบอร์ด และ จากสัญญาณ DTR ของ FT232RL
- +3V3 เป็นแหล่งจ่ายไฟขนาด +3.3V ที่ได้จากวงจร Regulate ของ LM1117-3V3 สามารถจ่ายกระแสได้สูงสุด 500mA
- +5V เป็นจุดต่อแหล่งจ่ายไฟของบอร์ดออกไปใช้งาน ซึ่งมาจากแหล่งกำเนิด 2 แหล่ง คือ จากพอร์ต USB และจาก External Supply ซึ่งถ้าต่อแหล่งจ่ายให้บอร์ดจาก External Supply ผ่านทาง Jack VIN แหล่งจ่าย +5V นี้จะมาจาก Switching Regulate (LM2575-5V) สามารถจ่ายกระแสได้สูงสุดถึง 1A แต่ถ้าใช้แหล่งจ่ายจากพอร์ต USB แหล่งจ่าย +5V นี้จะมาจากพอร์ต USB โดยตรงโดยจะมีฟิวส์ แบบ Poly ขนาด 500mA ต่อป้องกันการดึงกระแสเกินเพื่อป้องกันความเสียหายของพอร์ต USB โดยจะจ่ายกระแสได้สูงสุดไม่เกิน500mA ขึ้นอยู่กับการ Configure ค่าให้กับ FT232RL ด้วย
- +VIN เป็นไฟ DC ที่รับมาจาก Jack VIN(External Supply) แต่ผ่านการ Rectifier และ Filter เป็น DC แล้ว มีขนาดแรงดันเฉลี่ยตามขนาดแรงดันที่ป้อนให้กับบอร์ดทาง Jack VIN
- A0-A15 เป็นขาสัญญาณ Analog Input แบบ ADC มีขนาดความละเอียด 10บิต มี 16 Pin สามารถรับแรงดัน Analog Input ได้ 0-5VDC
- D0-D53 เป็นขาสัญญาณ Digital Input/Output แบบ TTL มีทั้งหมด 54 Pin สามารถใช้ทำหน้าที่เป็น Input หรือ Output ตามการกำหนดจากโปรแกรม โดยมีบาง Pin สามารถกำหนดหน้าที่ใช้งานเป็นฟังก์ชั่นพิเศษต่างๆเพิ่มเติมได้อีก
  - D0-D1 ถูกสงวนไว้ใช้ทำหน้าที่เป็นพอร์ตสื่อสารอนุกรม RS232 (UART0) โดยได้ทำการเชื่อมต่อกับ USB Bridge ของ FT232RL เพื่อใช้ Upload Code ให้กับบอร์ด และยังสามารถใช้ทดลองติดต่อสื่อสารรับส่งข้อมูลระหว่างบอร์ดกับคอมพิวเตอร์ PC ได้ด้วย
  - D2-D13 สามารถ โปรแกรมหน้าที่เป็น PWM ขนาด 8 บิต มี 14 Pin ได้
  - D14 สามารถ โปรแกรมหน้าที่เป็น TX3 สำหรับ ส่งข้อมูลของ UART3 ได้ด้วย
  - D15 สามารถ โปรแกรมหน้าที่เป็น RX3 สำหรับ รับข้อมูลให้กับ UART3 ได้ด้วย
  - D16 สามารถ โปรแกรมหน้าที่เป็น TX2 สำหรับ ส่งข้อมูลของ UART2 ได้ด้วย
  - D17 สามารถ โปรแกรมหน้าที่เป็น RX2 สำหรับ รับข้อมูลให้กับ UART2 ได้ด้วย
  - D18 สามารถ โปรแกรมหน้าที่เป็น TX1 สำหรับ ส่งข้อมูลของ UART1 ได้ด้วย
  - D19 สามารถ โปรแกรมหน้าที่เป็น RX1 สำหรับ รับข้อมูลให้กับ UART1 ได้ด้วย
  - D20,D21 สามารถ โปรแกรมหน้าที่เป็น SDA,SCL ของ I2C Bus ของ I2C ได้ด้วย

- AREF เป็นสัญญาณ Analog Reference จากภายนอกที่ต้องการป้อนให้กับ MCU ซึ่งตามปรกติแล้ว ATMEGA2560 สามารถโปรแกรมให้เลือกใช้แรงดันอ้างอิงจากภายในได้อยู่แล้วโดยสามารถเลือกเป็น 1.1V หรือ 2.56V หรือ AVCC(+5V) โดยไม่จำเป็นต้องป้อนแรงดันอ้างอิงจากภายนอกให้กับบอร์ดอีก แต่ถ้าต้องการแรงดันอ้างอิงที่มีความแตกต่างจากที่กล่าวมาแล้วก็สามารถป้อนเป็นแรงดันอ้างอิงจากภายนอกผ่านทางขา AREF นี้เข้าไปเองได้ระหว่าง 0-5V
- USB Host ใช้เชื่อมต่อกับอุปกรณ์ USB Device หรือ แอนดรอยด์โฟน โดยใช้ชิพ USB Host เบอร์ MAX3421 เป็นตัวกลางในการเชื่อมต่อระหว่างอุปกรณ์ USB กับ MCU ATMEGA2560 ซึ่งในปัจจุบันมีการ สร้าง Library ขึ้นมาสนับสนุนการเชื่อมต่อให้นำไปประยุกต์ดัดแปลงใช้งานกันได้ฟรีๆ ทั้งแบบ USB Host และแบบเชื่อมต่อกับอุปกรณ์ แอนดรอยน์โฟน ซึ่งถ้าใช้แอนดรอยน์โฟนที่ติดตั้งระบบปฏิบัติการรุ่น V2.3.4 หรือสูงกว่าสามารถใช้การพัฒนาด้วย Google ADK ได้ แต่ถ้าแอนดรอยน์โฟนไม่รองรับ ADK ก็สามารถใช้ ADB ของ Microbridge แทนได้เช่นกัน
    - **การพัฒนาโปรแกรมแบบ USB Host** โดยใช้รูปแบบการพัฒนาโปรแกรมเช่นเดียวกันกับบอร์ด Arduino ปรกติทั่วๆไป โดยในกรณีนี้จะประยุกต์ใช้ MAX3421 ทำหน้าที่เป็นอุปกรณ์ USB Host เพื่อเชื่อมต่อกับอุปกรณ์ USB Device ทั่วๆไป เช่น USB HID Keyboard, USB HID Mouse ฯลฯ
    - **การพัฒนาโปรแกรมแบบ Android** สามารถทำได้ 2 แนวทาง
        - พัฒนาโปรแกรมผ่าน Google Open Accessories API ด้วยชุดพัฒนาของ ADK (Android Open Accessories development Kit) มุ่งเน้นไปที่การนำความสามารถของอุปกรณ์ แอดดรอยน์โฟน เช่น หน้าจอแสดงผล ระบบ Touch Screen และอุปกรณ์เซ็นเซอร์ต่างๆที่มีบรรจุไว้ใน แอนดรอยน์โฟน มาพัฒนาต่อยอดใช้งาน ซึ่งความสามารถนี้จะใช้ได้กับอุปกรณ์แอนดรอยน์โฟนรุ่นที่สามารถติดตั้งระบบปฏิบัติการของแอนดรอยน์ ตั้งแต่เวอร์ชั่น 2.3.4 หรือสูงกว่า
        - พัฒนาโปรแกรมผ่าน Library ของ Microbridge ด้วย ADB (Android Debug Bridge) มุ่งเน้นไปที่การเชื่อมต่อสื่อสาร สั่งงานอุปกรณ์ I/O ภายนอกกับแอนดรอยน์ ซึ่งในกรณีของการเชื่อมต่อกับ Arduino ก็จะทำให้เราสามารถนำ แอนดรอยน์โฟน ส่งคำสั่งออกไป หรือ รับข้อมูลจาก Arduino ได้ตามต้องการ ไม่ว่าจะเป็น Digital I/O,PWM,I2C Bus หรือ Analog Input(ADC) ซึ่งความสามารถนี้จะใช้ได้กับอุปกรณ์แอนดรอยน์โฟนทุกรุ่นที่สามารถติดตั้งระบบปฏิบัติการของแอนดรอยน์ ตั้งแต่เวอร์ชั่น V1.5 หรือสูงกว่า

การพัฒนาโปรแกรมของ ET-MEGA2560-ADK ด้วย Arduino

ตามปรกติแล้วบอร์ด ET-MEGA2560-ADK จะทำการ ติดตั้งโปรแกรม Bootloader ไว้ให้กับ MCU เป็นที่เรียบร้อยแล้ว โดยใช้ Bootloader ชื่อ "stk500boot_v2_mega2560.hex" ซึ่งเป็น Bootloader มาตรฐานจาก Arduino โดยโปรแกรม Bootloader นี้จะใช้สำหรับติดต่อสื่อสารเพื่อสั่ง Upload Code จากคอมพิวเตอร์ PC ให้กับ MCU ในบอร์ดทำงาน โดยไม่ต้องใช้เครื่องโปรแกรมภายนอกให้ยุ่งยาก ซึ่งคุณสมบัติของ Bootloader รุ่น Arduino-0022 มีคุณสมบัติการทำงานเป็นดังนี้

- สื่อสารกับโปรแกรมภายนอกด้วย Protocol แบบ stk500v2
- ใช้ความเร็ว Baudrate 115200 โดยใช้ความถี่ XTAL 16 MHz
- โปรแกรม Bootloader มีขนาด 8KByte ทำงานที่ตำแหน่ง 0x3E000-0x3FFFF
- ใช้ LED ที่ต่อกับขา Digital-13 เป็นตัวแสดงสถานะในขณะที่ Bootloader ทำงาน
- โปรแกรมใน Bootloader จะทำงานโดยอัตโนมัติทุกครั้งหลังการรีเซ็ต โดย MCU จะเริ่มต้นทำงานใน Bootloader นี้ก่อนเสมอ เพื่อรอการติดต่อสื่อสารจากโปรแกรมสำหรับสั่งให้ทำการ Upload Code ให้กับ MCU แต่ถ้าไม่มีการติดต่อสื่อสารเข้ามาภายในเวลาที่กำหนดไว้ ก็จะกระโดดไปทำงานตามโปรแกรมที่ผู้ใช้โหลดไว้ให้ทันที

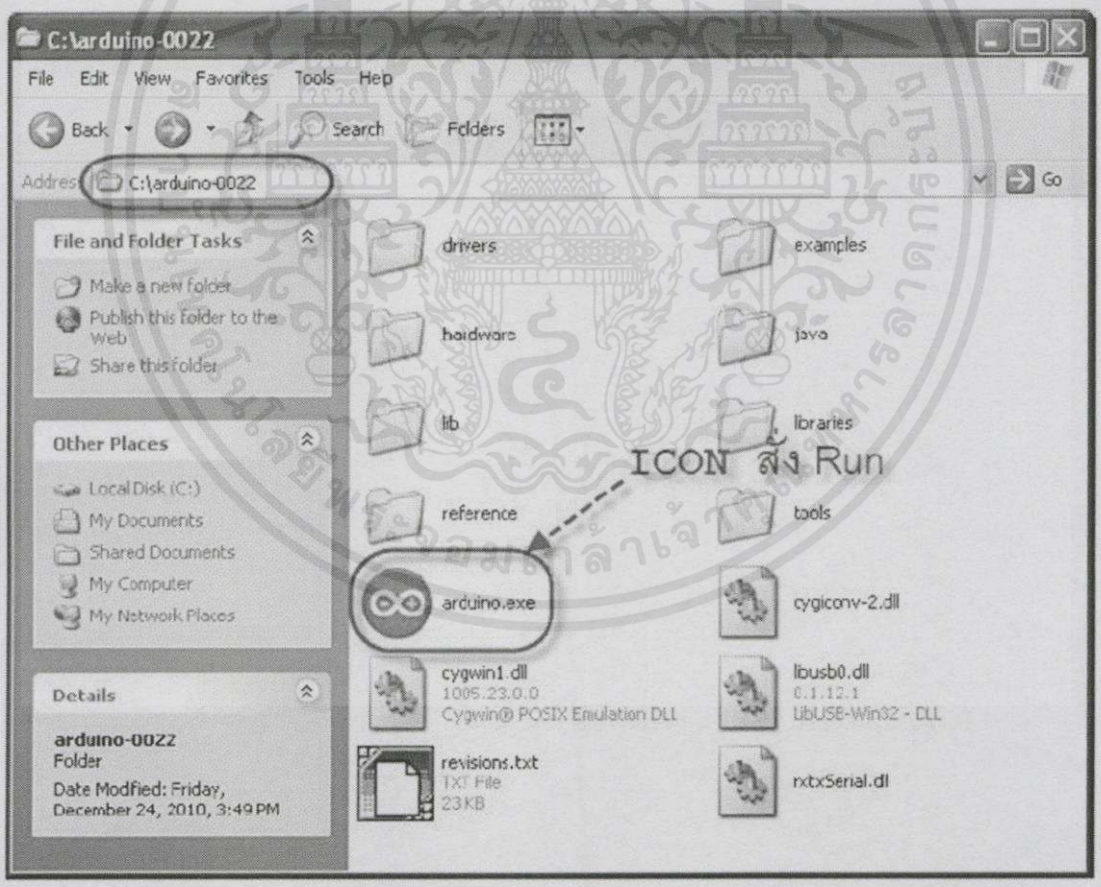


สำหรับบอร์ด ET-MEGA2560-ADK นั้น จะรองรับการ Reset MCU แบบอัตโนมัติจาก USB Bridge (FT232RL) โดยใช้ขา DTR จาก FT232RL เป็นขาควบคุมการรีเซ็ต MCU

การติดตั้งโปรแกรม Arduino

สำหรับโปรแกรม Arduino นั้น ได้รับการพัฒนาขึ้นมาให้สามารถใช้งานกับระบบปฏิบัติการแบบต่างๆ ได้หลาย Platform ซึ่งปัจจุบัน (เดือน กันยายน พศ.2554) โปรแกรมของ Arduino ได้รับการปรับปรุงเป็นรุ่น เวอร์ชั่น "Arduino-0022" แล้ว โดยมีโปรแกรมให้เลือกใช้งาน 4 Platform ทั้ง Windows,Mac OSx และ Linux โดยผู้อ่านสามารถเข้าไป ตรวจสอบ หรือ Download โปรแกรมรุ่นใหม่ๆของ Arduino มาใช้งานได้ฟรีโดยไม่เสียค่าใช้จ่ายใดๆจาก "http://arduino.cc/" หรือ "http://arduino.cc/en/Main/Software" ซึ่งเป็นเวปไซต์ที่ได้รวบรวมรายละเอียดและข่าวคราวความเคลื่อนไหวต่างๆ เกี่ยวกับ Arduino มากมาย ซึ่งข้อมูลต่างๆจะได้รับการปรับปรุงอย่างต่อเนื่องเป็นประจำ

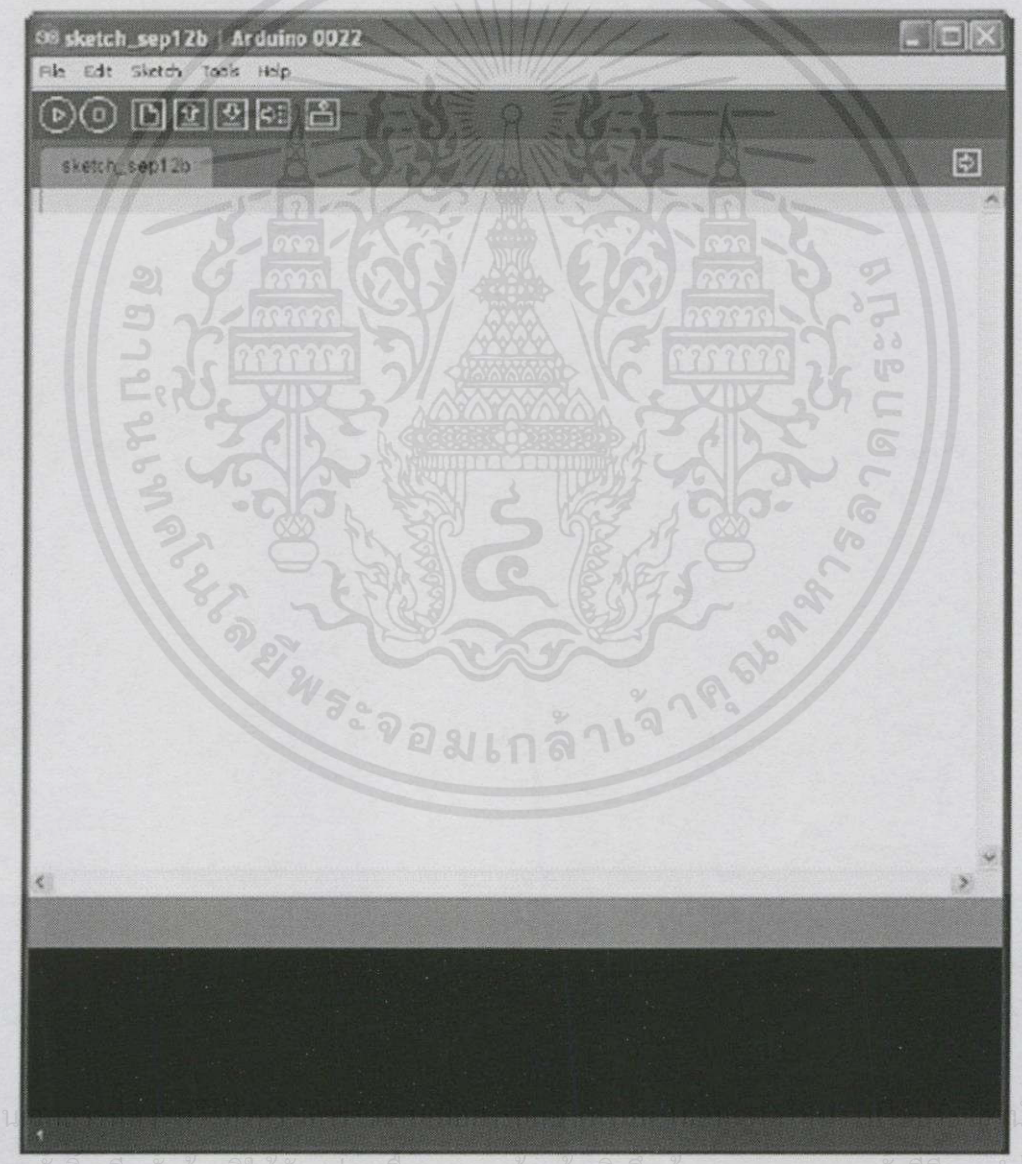
โดยในการติดตั้งโปรแกรมของ Arduino นั้นให้ทำการ Unzip แล้ว Copy ไปติดตั้งไว้ในตำแหน่งโฟลเดอร์ "c:\arduino-0022" ดังตัวอย่าง



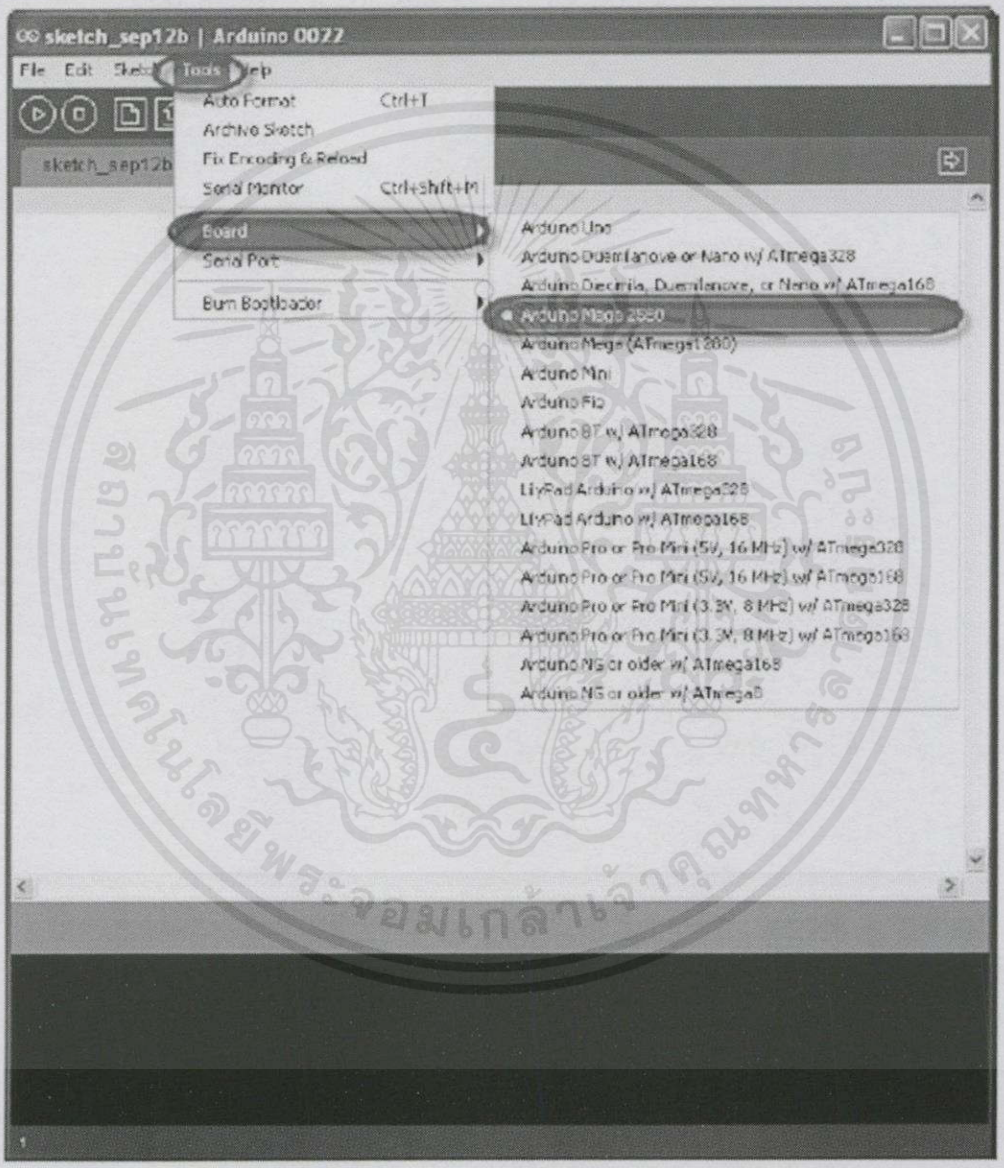
ทดสอบเขียนโปรแกรมใช้งานด้วย Arduino

หลังจากที่เราได้ทำการติดตั้งโปรแกรม Arduino เป็นที่เรียบร้อยแล้ว ก็เป็นอันเสร็จสิ้นขั้นตอนของการเตรียมการแล้ว ลำดับขั้นตอนต่อจากนี้เป็นต้นไป ก็เป็นเรื่องของการใช้งาน การเขียนโปรแกรม และการศึกษาเรียนรู้ต่างๆตามความต้องการแล้ว แต่ก่อนอื่นเราจะต้องทำการติดตั้งโปรแกรมของ Arduino เพื่อใช้เป็นโปรแกรมสำหรับศึกษาเรียนรู้ ซึ่งมีลำดับขั้นตอนดังต่อไปนี้

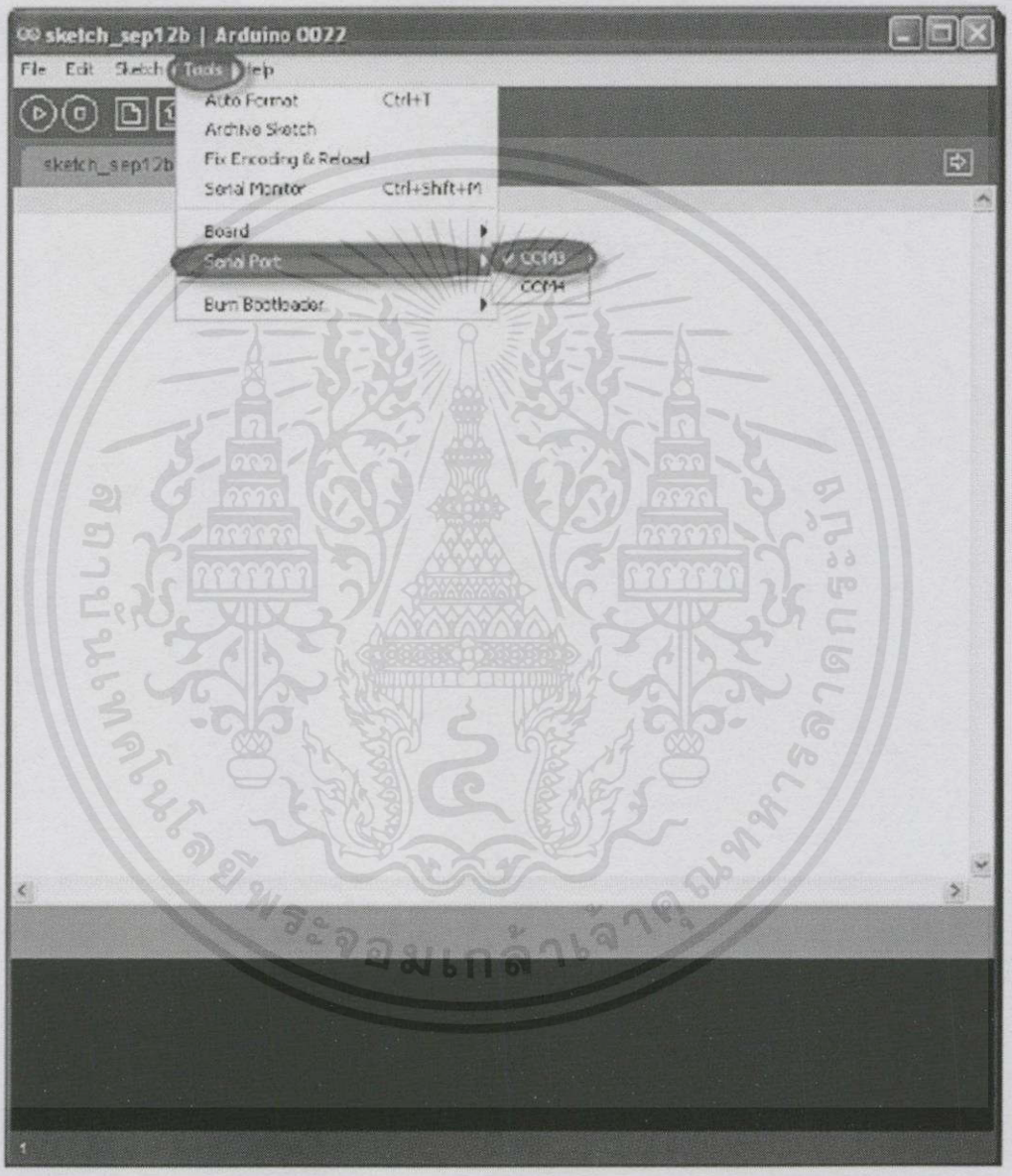
1. ทำการสั่ง Run โปแกรม “arduino.exe” จะได้ผลดังรูป



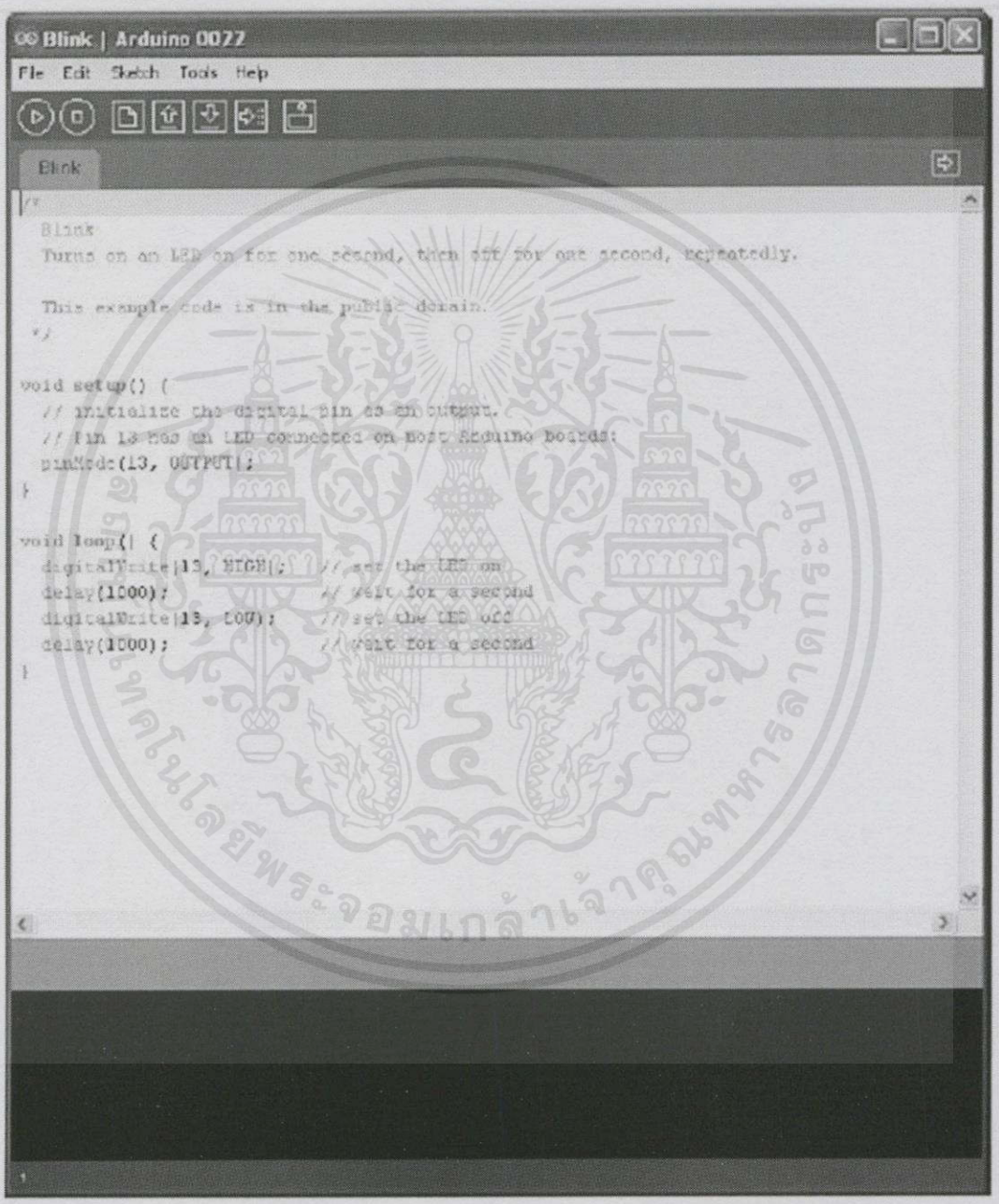
2. ในครั้งแรกของการเรียกใช้งานโปรแกรม ให้ทำการกำหนดระบบฮาร์ดแวร์ที่จะใช้งานกับโปรแกรมของ Arduino ให้เรียบร้อยเสียก่อน เนื่องจากในปัจจุบันนี้ มีการออกแบบวงจรและสร้างฮาร์ดแวร์บอร์ดแบบต่างๆสำหรับนำมาใช้งานร่วมกับโปรแกรมพัฒนาของ Arduino ไว้มากมายหลายรุ่น โดยในกรณีของบอร์ด ET-MEGA2560-ADK ให้ทำการเลือกกำหนดชื่อบอร์ดเป็น "Arduino Mega" โดยคลิกเมาส์ที่ "Tools → Board → "Arduino Mega" ดังรูป



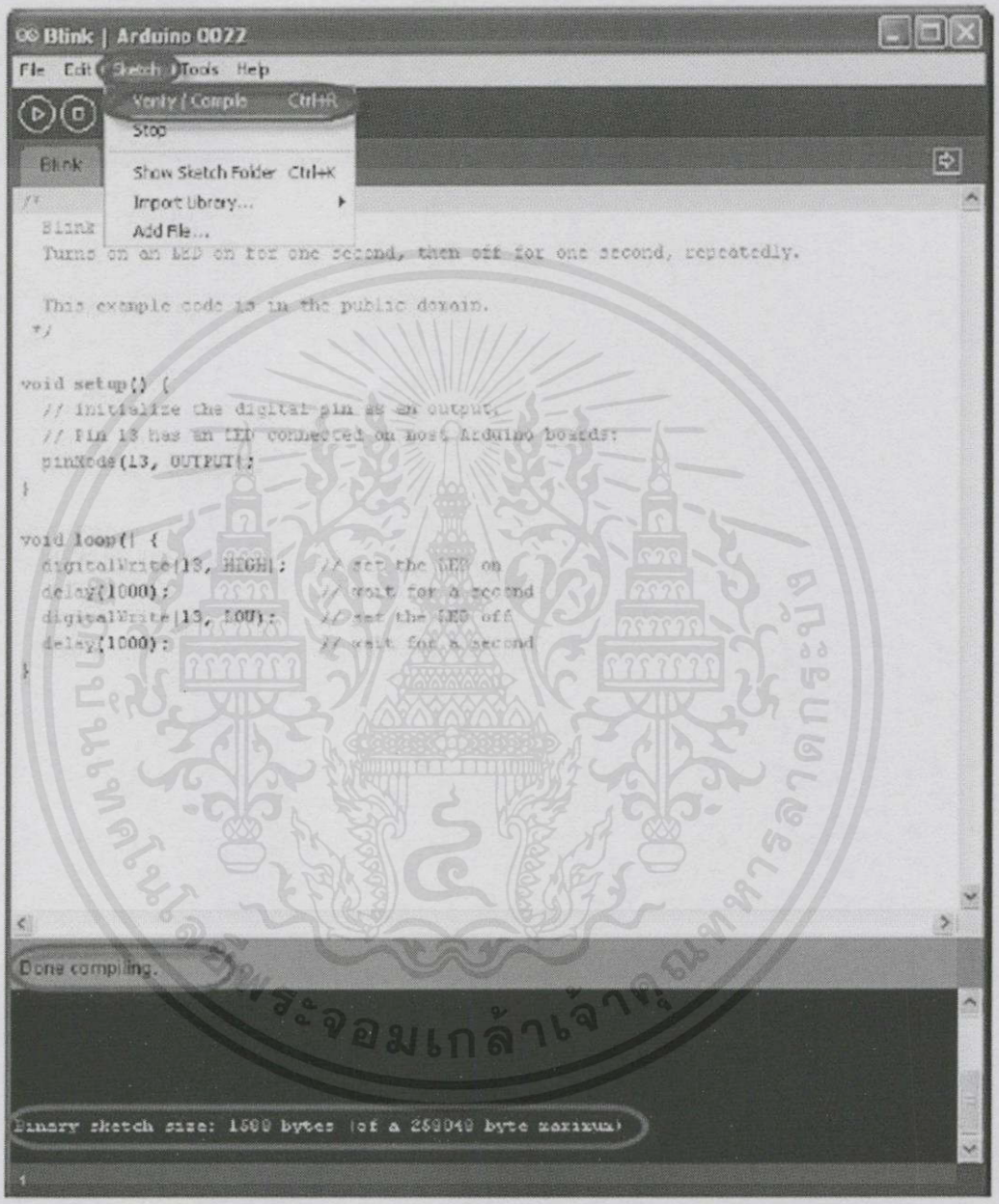
3. เลือกกำหนดหมายเลขพอร์ต สำหรับติดต่อสื่อสารกับบอร์ด ให้ตรงกับหมายเลข Comport ที่ต่อใช้งานไว้จริงในเครื่องคอมพิวเตอร์ PC เช่น ถ้าหมายเลข Comport ของเครื่องคอมพิวเตอร์ PC เป็น COM3 ให้คลิกเมาส์ที่ Tools → Serial Port → COM3 ดังรูป



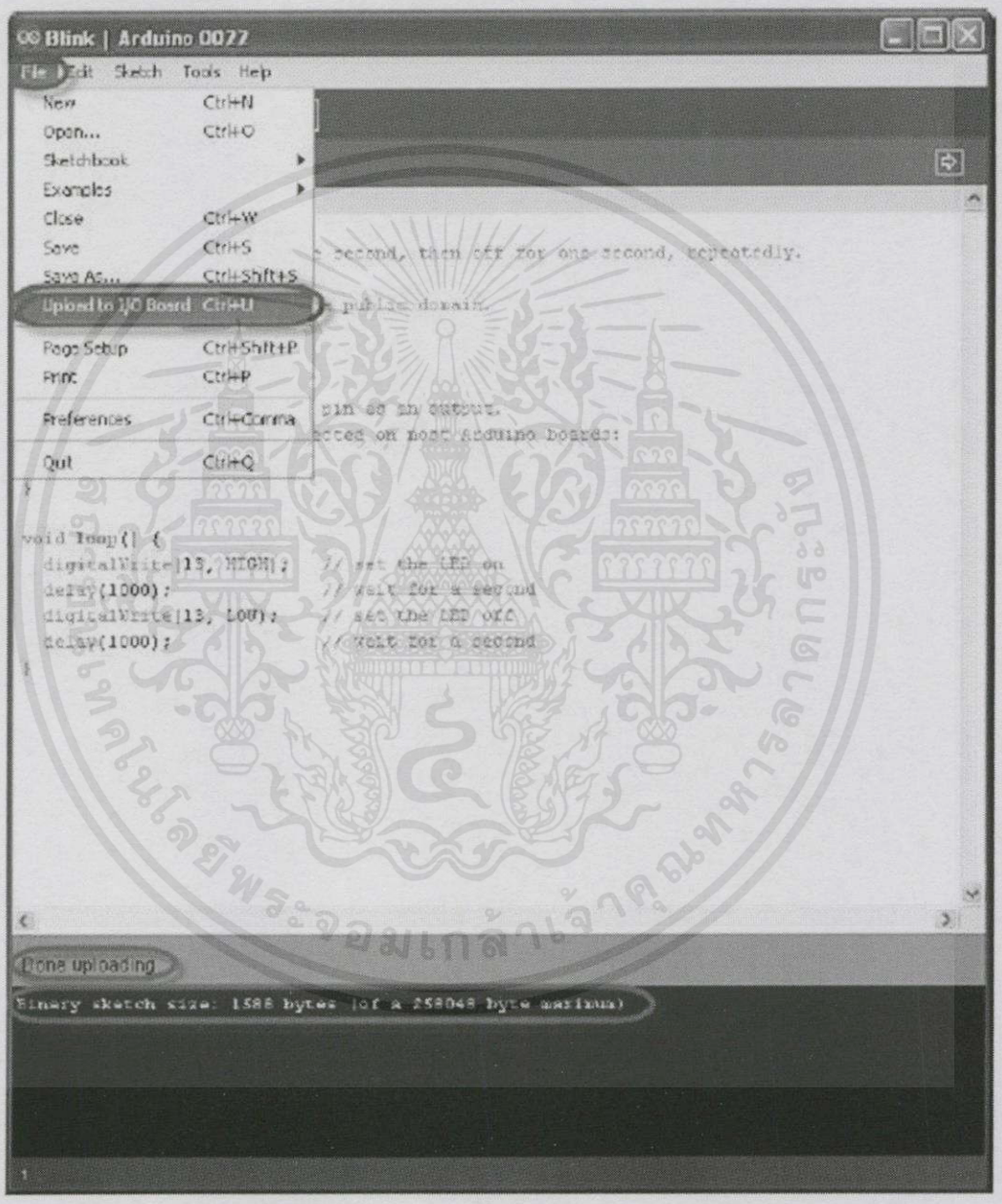
4. ทดสอบเขียนโปรแกรม โดยคลิกเมาส์ที่ File → New แล้วพิมพ์โปรแกรมทดสอบ หรืออาจใช้การสั่งเปิดไฟล์ตัวอย่างที่สร้างไว้แล้วขึ้นมาแทนก็ได้ โดยในที่นี้ขอแนะนำให้ทดสอบด้วยโปรแกรมไฟกระพริบ โดยให้เลือก "File → sketchbook → Examples → Digital → Blink" ซึ่งจะได้ดังรูป



5. สั่งแปลโปรแกรมโดยคลิกเมาส์ที่ "Sketch → Verify/Compile" เพื่อตรวจสอบคำสั่งต่างๆใน โปรแกรมว่าถูกต้องหรือไม่ ดังตัวอย่าง



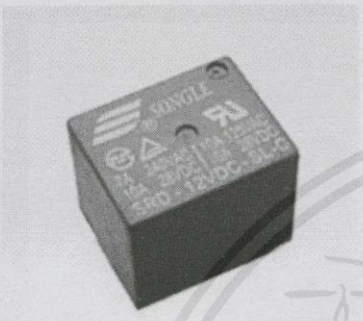
6. สั่ง Download Code ให้กับบอร์ด โดยคลิกเมาส์เลือกที่ "File → Upload to I/O Board" แล้วรอสักครู่จนโปรแกรมทำงานเสร็จ หลังจากที่ทำการ Upload Code ให้กับบอร์ดเป็นที่เรียบร้อยแล้ว บอร์ดก็จะเริ่มต้นทำงานตามคำสั่งที่เขียนไว้ในโปรแกรมทันที โดยจะสังเกตเห็น LED กระพริบ ติด และ ดับ สลับกันไปมา ด้วยความเร็วประมาณ 1 วินาที ตลอดเวลา ซึ่งควรได้ผลดังรูป



ภาคผนวก ค.

SONGLE RELAY

松乐继电器 ® SONGLE RELAY	RELAY ISO9002	SRD



1. MAIN FEATURES

- Switching capacity available by 10A in spite of small size design for highdensity P.C. board mounting technique.
- UL,CUL,TUV recognized.
- Selection of plastic material for high temperature and better chemical solution performance.
- Sealed types available.
- Simple relay magnetic circuit to meet low cost of mass production.

2. APPLICATIONS

- Domestic appliance, office machine, audio, equipment, automobile, etc.
  (Remote control TV receiver, monitor display, audio equipment high rushing current use application.)

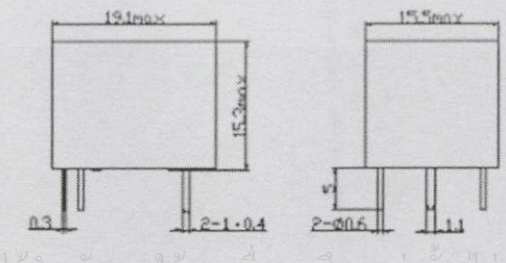
3. ORDERING INFORMATION

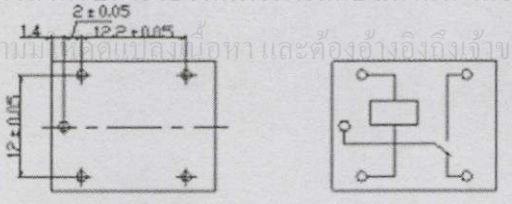
SRD	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil sensitivity	Contact form
SRD	03、05、06、09、12、24、48VDC	S:Sealed type	L:0.36W	A:1 form A
				B:1 form B
		F:Flux free type	D:0.45W	C:1 form C

4. RATING

CCC	FILE NUMBER:	CQC03001003731	10A/250VDC
UL /CUL	FILE NUMBER:	E167996	10A/125VAC 28VDC
TUV	FILE NUMBER:	R50056114	10A/250VAC 30VDC

5. DIMENSION(unit:mm) DRILLING(unit:mm) WIRING DIAGRAM





6. COIL DATA CHART (AT20°C)

Coil Sensitivity	Coil Voltage Code	Nominal Voltage (VDC)	Nominal Current (mA)	Coil Resistance (Ω) ±10%	Power Consumption (W)	Pull-In Voltage (VDC)	Drop-Out Voltage (VDC)	Max-Allowable Voltage (VDC)
SRD (High Sensitivity)	03	03	120	25	abt. 0.36W	75%Max.	10% Min.	120%
	05	05	71.4	70				
	06	06	60	100				
	09	09	40	225				
	12	12	30	400				
	24	24	15	1600				
	48	48	7.5	6400				
SRD (Standard)	03	03	150	20	abt. 0.45W	75% Max.	10% Min.	110%
	05	05	89.3	55				
	06	06	75	80				
	09	09	50	180				
	12	12	37.5	320				
	24	24	18.7	1280				
	48	48	10	4500	abt. 0.51W			

7. CONTACT RATING

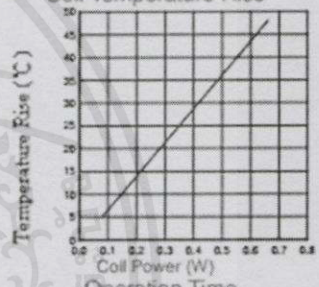
Item \ Type	SRD FORM C	SRD FORM A
Contact Capacity Resistive Load (cosΦ=1)	10A 125VAC	10A 30VDC 10A 250VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	3A 120VAC 3A 28VDC	5A 120VAC 5A 28VDC
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	800VAC/240W	1200VA/300W
Contact Material	AgCdO	AgCdO

8. PERFORMANCE (at initial value)

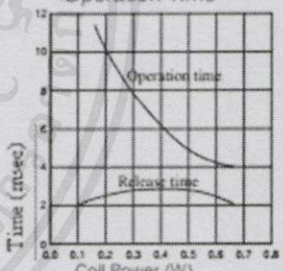
Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength Between coil & contact Between contacts	 1500VAC 50/60HZ (1 minute) 1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching Mechanically Electrically	 300 operation/min 30 operation/min
Ambient Temperature	-40°C to +85°C
Operating Humidity	45 to 85% RH
Vibration Endurance Error Operation	 10 to 55Hz Double Amplitude 1.5mm 10 to 55Hz Double Amplitude 1.5mm
Shock Endurance Error Operation	 100G Min. 10G Min.
Life Expectancy Mechanically Electrically	 10^7 operations. Min. (no load) 10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

9.REFERENCE DATA

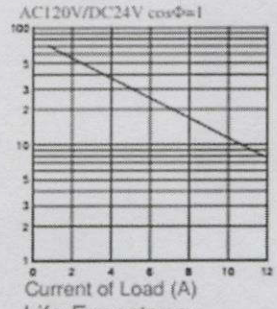
Coil Temperature Rise



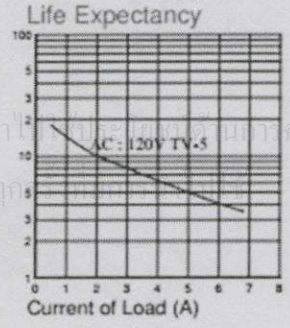
Operation Time



Life Expectancy

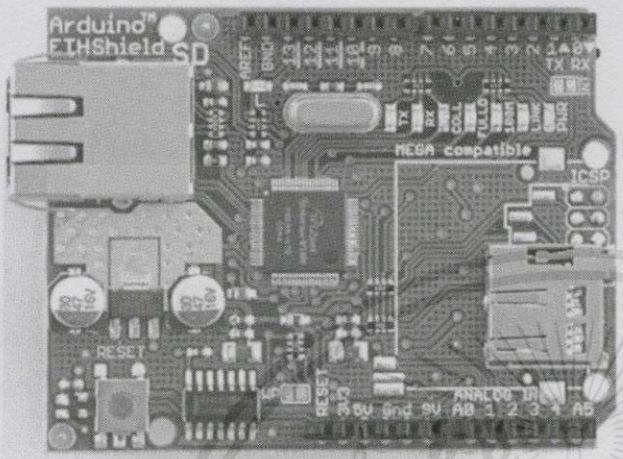


Life Expectancy



ภาคผนวก ง.

Arduino Ethernet Shield



Download: arduino-ethernet-shield-05-schematic.pdf, arduino-ethernet-shield-05-reference-design.zip

Download: arduino-ethernet-shield-schematic.pdf, arduino-ethernet-shield-reference-design.zip

The Arduino Ethernet Shield allows an Arduino board to connect to the internet. It is based on the Wiznet W5100 ethernet chip (datasheet). The Wiznet W5100 provides a network (IP) stack capable of both TCP and UDP. It supports up to four simultaneous socket connections. Use the Ethernet library to write sketches which connect to the internet using the shield. The ethernet shield connects to an Arduino board using long wire-wrap headers which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top.

The latest revision of the shield adds a micro-SD card slot, which can be used to store files for serving over the network. It is compatible with the Arduino Duemilanove and Mega (using the Ethernet library coming in Arduino 0019). An SD card library is not yet included in the standard Arduino distribution, but the sdfatlib by Bill Greiman works well. See this tutorial from Adafruit Industries for instructions (thanks Limor!).

The latest revision of the shield also includes a reset controller, to ensure that the W5100 Ethernet module is properly reset on power-up. Previous revisions of the shield were not compatible with the Mega and need to be manually reset after power-up. The original revision of the shield contained a full-size SD card slot; this is not supported.

Arduino communicates with both the W5100 and SD card using the SPI bus (through the ICSP header). This is on digital pins 11, 12, and 13 on the Duemilanove and pins 50, 51, and 52 on the Mega. On both boards, pin 10 is used to select the W5100 and pin 4 for the SD card. These pins cannot be used for general i/o. On the Mega, the hardware SS pin, 53, is not used to select either the W5100 or the SD card, but it must be kept as an output or the SPI interface won't work.

Note that because the W5100 and SD card share the SPI bus, only one can be active at a time. If you are using both peripherals in your program, this should be taken care of by the corresponding libraries. If you're not using one of the peripherals in your program, however, you'll need to explicitly deselect it. To do this with the SD card, set pin 4 as an output and write a high to it. For the W5100, set digital pin 10 as a high output.

The shield provides a standard RJ45 ethernet jack.

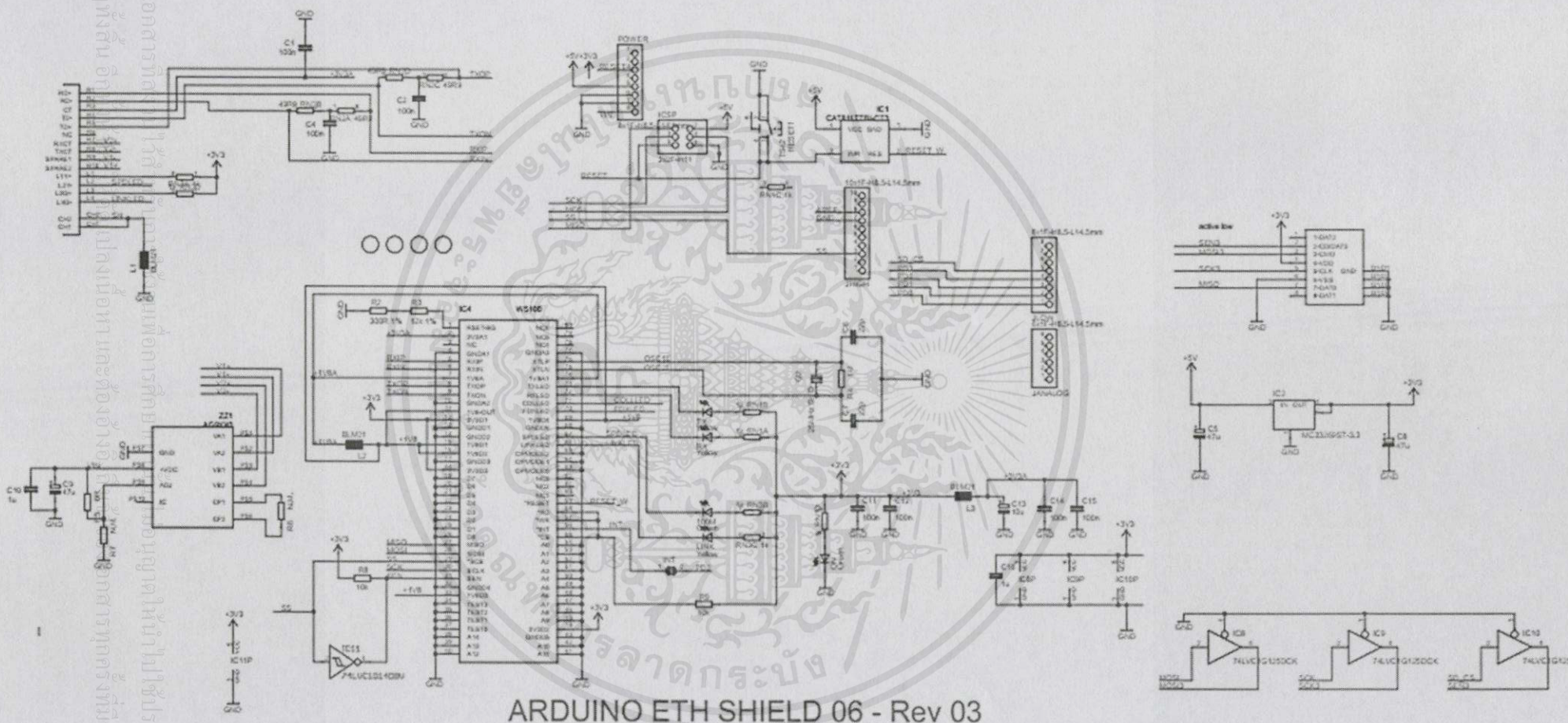
The reset button on the shield resets both the W5100 and the Arduino board.

The shield contains a number of informational LEDs:

- PWR: indicates that the board and shield are powered
- LINK: indicates the presence of a network link and flashes when the shield transmits or receives data
- FULLD: indicates that the network connection is full duplex
- 100M: indicates the presence of a 100 Mb/s network connection (as opposed to 10 Mb/s)
- RX: flashes when the shield receives data
- TX: flashes when the shield sends data
- COLL: flashes when network collisions are detected

The solder jumper marked "INT" can be connected to allow the Arduino board to receive interrupt-driven notification of events from the W5100, but this is not supported by the Ethernet library. The jumper connects the INT pin of the W5100 to digital pin 2 of the Arduino.

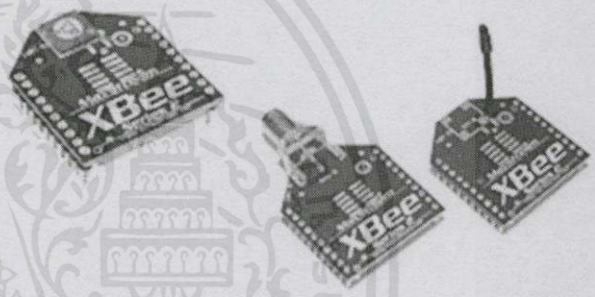
See also: getting started with the ethernet shield and Ethernet library reference



ARDUINO ETH SHIELD 06 - Rev 03

ภาคผนวก จ.

XBee™ Series 2 OEM RF Modules



Product Manual v1.x.1x - *ZigBee* Protocol
For OEM RF Module Part Numbers: XB24-BxIT-00x

ZigBee OEM RF Modules by MaxStream, Inc. - a Digi International brand

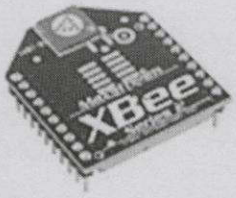
Firmware Versions:
- 1.0xx - Coordinator, Transparent Operation
- 1.1xx - Coordinator, API Operation
- 1.2xx - Router, End Device, Transparent Operation
- 1.3xx - Router, End Device, API Operation

Contents

1. XBee Series 2 OEM RF Modules

The XBee Series 2 OEM RF Modules were engineered to operate within the ZigBee protocol and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between remote devices.

The modules operate within the ISM 2.4 GHz frequency band.



1.1. Key Features

High Performance, Low Cost

- Indoor/Urban: up to 133' (40 m)
- Outdoor line-of-sight: up to 400' (120 m)
- Transmit Power: 2 mW (+3 dBm)
- Receiver Sensitivity: -95 dBm

RF Data Rate: 250,000 bps

Advanced Networking & Security

Retries and Acknowledgements

DSSS (Direct Sequence Spread Spectrum)

Each direct sequence channel has over 65,000 unique network addresses available

Point-to-point, point-to-multipoint and peer-to-peer topologies supported

Self-routing, self-healing and fault-tolerant mesh networking

Low Power

XBee Series 2

- TX Current: 40 mA (@3.3 V)
- RX Current: 40 mA (@3.3 V)
- Power-down Current: < 1 µA @ 25°C

Easy-to-Use

No configuration necessary for out-of box RF communications

AT and API Command Modes for configuring module parameters

Small form factor

Extensive command set

Free X-CTU Software (Testing and configuration software)

Free & Unlimited Technical Support

1.1.1. Worldwide Acceptance

FCC Approval (USA) Refer to Appendix A [p50] for FCC Requirements. Systems that contain XBee Series 2 RF Modules inherit MaxStream Certifications.

ISM (Industrial, Scientific & Medical) **2.4 GHz frequency band**

Manufactured under **ISO 9001:2000** registered standards

XBee Series 2 RF Modules are optimized for use in **US, Canada, Australia, Israel and Europe** (contact MaxStream for complete list of agency approvals).

1.2. Specifications

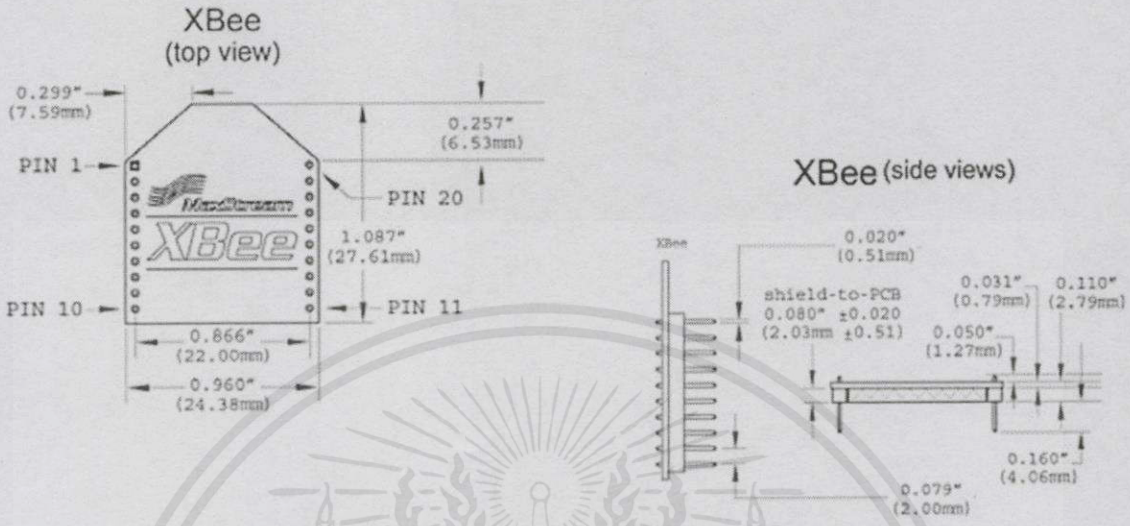
Table 1-01. Specifications of the XBee Series 2 OEM RF Module (PRELIMINARY)

Specification	XBee Series 2
Performance	
Indoor/Urban Range	up to 133 ft. (40 m)
Outdoor RF line-of-sight Range	up to 400 ft. (120 m)
Transmit Power Output (software selectable)	2mW (+3dBm)
RF Data Rate	250,000 bps
Serial Interface Data Rate (software selectable)	1200 - 230400 bps (non-standard baud rates also supported)
Receiver Sensitivity	-95 dBm (1% packet error rate)
Power Requirements	
Supply Voltage	2.8 - 3.4 V
Operating Current (Transmit)	40mA (@ 3.3 V)
Operating Current (Receive)	40mA (@ 3.3 V)
Power-down Current	< 1 uA @ 25°C
General	
Operating Frequency Band	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)
Operating Temperature	-40 to 85° C (industrial)
Antenna Options	Integrated Whip, Chip, RPSMA, or U.FL Connector
Networking & Security	
Supported Network Topologies	Point-to-point, Point-to-multipoint, Peer-to-peer & Mesh
Number of Channels (software selectable)	16 Direct Sequence Channels
Addressing Options	PAN ID and Addresses, Cluster IDs and Endpoints (optional)
Agency Approvals	
United States (FCC Part 15.247)	Pending
Industry Canada (IC)	Pending
Europe (CE)	Pending

Antenna Options: The ranges specified are typical when using the integrated Whip (1.5 dBi) and Dipole (2.1 dBi) antennas. The Chip antenna option provides advantages in its form factor; however, it typically yields shorter range than the Whip and Dipole antenna options when transmitting outdoors. For more information, refer to the "XBee Series 2 Antenna" application note located on MaxStream's web site http://www.maxstream.net/support/knowledgebase/article.php?kb=153

1.3. Mechanical Drawings

Figure 1-01. Mechanical drawings of the XBee Series 2 OEM RF Modules (antenna options not shown)



1.4. Mounting Considerations

The XBee Series 2 RF Module (through-hole) was designed to mount into a receptacle (socket) and therefore does not require any soldering when mounting it to a board. The XBee Series 2 Development Kits contain RS-232 and USB interface boards which use two 20-pin receptacles to receive modules.

Figure 1-02. XBee Series 2 Module Mounting to an RS-232 Interface Board.

The receptacles used on MaxStream development boards are manufactured by Century Interconnect. Several other manufacturers provide comparable mounting solutions; however, MaxStream currently uses the following receptacles:

- Through-hole single-row receptacles -
  Samtec P/N: MMS-110-01-L-SV (or equivalent)
- Surface-mount double-row receptacles -
  Century Interconnect P/N: CPRMSL20-D-0-1 (or equivalent)
- Surface-mount single-row receptacles -
  Samtec P/N: SMM-110-02-SM-S

MaxStream also recommends printing an outline of the module on the board to indicate the orientation the module should be mounted.

1.5. Pin Signals

Figure 1-03. XBee Series 2 RF Module Pin Number
(top sides shown • shields on bottom)

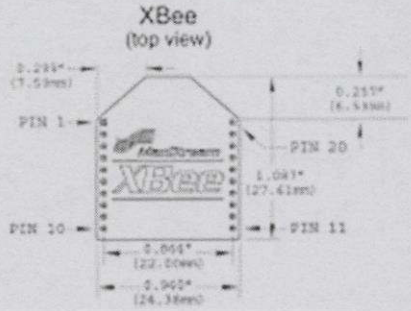


Table 1-02. Pin Assignments for the XBee Series 2 Modules
(Low-asserted signals are distinguished with a horizontal line above signal name.)

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / CONFIG	Input	UART Data In
4	DIO8	Either	Digital I/O 8
5	RESET	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI / DIO10	Output	PWM Output 0 / RX Signal Strength Indicator / Digital I/O
7	PWM / DIO11	Either	Digital I/O 11
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ/ DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	DIO4	Either	Digital I/O 4
12	CTS / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / SLEEP	Output	Module Status Indicator
14	[reserved]	-	Do not connect
15	Associate / DIO5	Either	Associated Indicator, Digital I/O 5
16	RTS / DIO6	Either	Request-to-Send Flow Control, Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

Design Notes:

- Minimum connections: VCC, GND, DOUT & DIN
- Minimum connections to support firmware upgrades: VCC, GND, DIN, DOUT, RTS & DTR
- Signal Direction is specified with respect to the module
- Module includes a 30k Ohm resistor attached to RESET
- Several of the input pull-ups can be configured using the PR command
- Unused pins should be left disconnected

1.6. Electrical Characteristics

Table 1-03. DC Characteristics of the XBee Series 2 (VCC = 2.8 - 3.4 VDC)

Symbol	Parameter	Condition	Min	Typical	Max	Units
VIL	Input Low Voltage	All Digital Inputs	-	-	0.2 * VCC	V
V_{IH}	Input High Voltage	All Digital Inputs	0.8 * VCC	-	0.18* VCC	V
V_{OL}	Output Low Voltage	I_{OL} = 2 mA, VCC >= 2.7 V	-	-	0.18*VCC	V
V_{OH}	Output High Voltage	I_{OH} = -2 mA, VCC >= 2.7 V	0.82*VCC	-	-	V
I_{IN}	Input Leakage Current	V_{IN} = VCC or GND, all inputs, per pin	-	-	0.5uA	uA
TX	Transmit Current	VCC = 3.3 V	-	45	-	mA
RX	Receive Current	VCC = 3.3 V	-	50	-	mA
PWR-DWN	Power-down Current	SM parameter = 1	-	< 10	-	uA

2. RF Module Operation

2.1. Serial Communications

The XBee Series 2 OEM RF Modules interface to a host device through a logic-level asynchronous serial port. Through its serial port, the module can communicate with any logic and voltage compatible UART; or through a level translator to any serial device (For example: Through a MaxStream proprietary RS-232 or USB interface board).

2.1.1. UART Data Flow

Devices that have a UART interface can connect directly to the pins of the RF module as shown in the figure below.

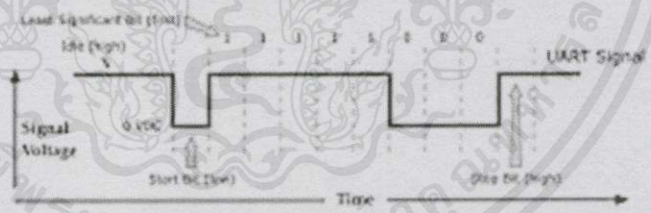
Figure 2-01. System Data Flow Diagram in a UART-interfaced environment
(Low-asserted signals distinguished with horizontal line over signal name.)

Serial Data

Data enters the module UART through the DIN (pin 3) as an asynchronous serial signal. The signal should idle high when no data is being transmitted.

Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). The following figure illustrates the serial bit pattern of data passing through the module.

Figure 2-02. UART data packet 0x1F (decimal number "31") as transmitted through the RF module
Example Data Format is 8-N-1 (bits - parity - # of stop bits)



The module UART performs tasks, such as timing and parity checking, that are needed for data communications. Serial communications depend on the two UARTs to be configured with compatible settings (baud rate, parity, start bits, stop bits, data bits).

2.1.2. Serial Buffers

The XBee Series 2 modules maintain small buffers to collect received serial and RF data. The serial receive buffer collects incoming serial characters and holds them until they can be processed. The serial transmit buffer collects data that is received via the RF link that will be transmitted out the UART.

Serial Receive Buffer

When serial data enters the RF module through the DIN Pin (3 pin), the data is stored in the serial receive buffer until it can be processed.

Hardware Flow Control ($\overline{CTS}$). When the serial receive buffer is 17 bytes away from being full, by default, the module de-asserts $\overline{CTS}$ (high) to signal to the host device to stop sending data [refer to D7 (DIO7 Configuration) parameter]. $\overline{CTS}$ is re-asserted after the serial receive buffer has 34 bytes of memory available.

Cases in which the serial receive buffer may become full and possibly overflow:

1. If the module is receiving a continuous stream of RF data, any serial data that arrives on the DIN pin is placed in the serial receive buffer. The data in the serial receive buffer will be transmitted over-the-air when the module is no longer receiving RF data in the network.
2. When data is ready to be transmitted, the module may need to discover a Network Address and/or a Route in order to reach the destination node. Discovery overhead may delay packet transmission.

   Refer to the ZigBee Networks --> Mesh Routing sections for more information.

Serial Transmit Buffer

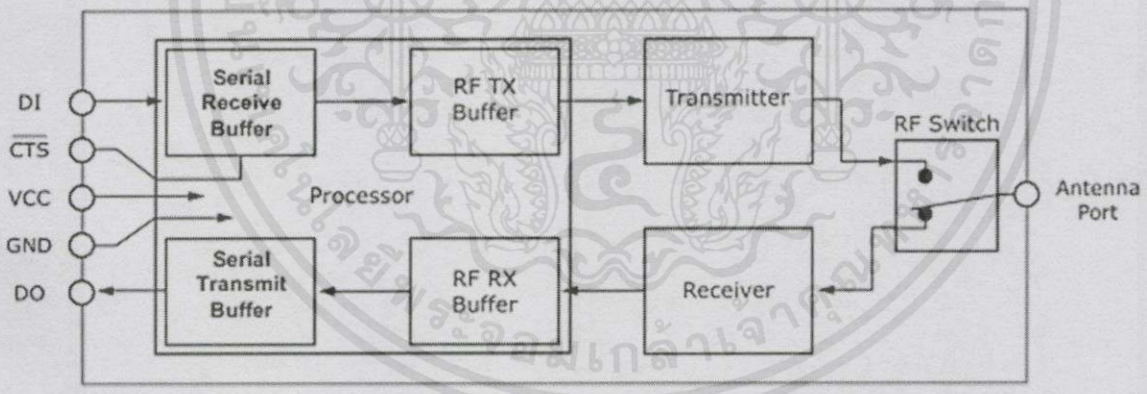
When RF data is received, the data is moved into the serial transmit buffer and is sent out the serial port. If the serial transmit buffer becomes full enough such that all data in a received RF packet won't fit in the serial transmit buffer, the entire RF data packet is dropped.

Hardware Flow Control ($\overline{RTS}$). If $\overline{RTS}$ is enabled for flow control (D6 (DIO6 Configuration) Parameter = 1), data will not be sent out the serial transmit buffer as long as $\overline{RTS}$ (pin 16) is de-asserted.

Cases in which the serial transmit buffer may become full resulting in dropped RF packets

1. If the RF data rate is set higher than the interface data rate of the module, the module could receive data faster than it can send the data to the host.
2. If the host does not allow the module to transmit data out from the serial transmit buffer because of being held off by hardware flow control.

Figure 2-03. Internal Data Flow Diagram



2.1.3. Transparent Operation

RF modules that contain the following firmware versions will support Transparent Mode: 1.0xx (coordinator) and 1.2xx (router/end device).

When operating in Transparent Mode, modules are configured using AT Commands and API operation is not supported. The modules act as a serial line replacement - all UART data received through the DIN pin is queued up for RF transmission. Data is sent to a module as defined by the DH (Destination Address High) and DL (Destination Address Low) parameters.

When RF data is received by a module, the data is sent out the DOUT pin.

Serial-to-RF Packetization

Data is buffered in the serial receive buffer until one of the following causes the data to be packetized and transmitted:

1. No serial characters are received for the amount of time determined by the RO (Packetization Timeout) parameter. If RO = 0, packetization begins when a character is received.
2. Maximum number of characters that will fit (72) in an RF packet is received.
3. The Command Mode Sequence (GT + CC + GT) is received. Any character buffered in the serial receive buffer before the sequence is transmitted.

2.1.4. API Operation

API (Application Programming Interface) Operation is an alternative to the default Transparent Operation. The frame-based API extends the level to which a host application can interact with the networking capabilities of the module. RF modules that contain the following firmware versions will support API operation: 1.1xx (coordinator) and 1.3xx (router/end device).

When in API mode, all data entering and leaving the module is contained in frames that define operations or events within the module.

Transmit Data Frames (received through the DIN pin (pin 3)) include:

- RF Transmit Data Frame
- Command Frame (equivalent to AT commands)

Receive Data Frames (sent out the DOUT pin (pin 2)) include:

- RF-received data frame
- Command response
- Event notifications such as reset, associate, disassociate, etc.

The API provides alternative means of configuring modules and routing data at the host application layer. A host application can send data frames to the module that contain address and payload information instead of using command mode to modify addresses. The module will send data frames to the application containing status packets; as well as source, and payload information from received data packets.

The API operation option facilitates many operations such as the examples cited below:

-> Transmitting data to multiple destinations without entering Command Mode
-> Receive success/failure status of each transmitted RF packet
-> Identify the source address of each received packet

To implement API operations, refer to the API Operation chapter 6.

2.2. Modes of Operation

2.2.1. Idle Mode

When not receiving or transmitting data, the RF module is in Idle Mode. During Idle Mode, the RF module is also checking for valid RF data. The module shifts into the other modes of operation under the following conditions:

- Transmit Mode (Serial data in the serial receive buffer is ready to be packetized)
- Receive Mode (Valid RF data is received through the antenna)
- Sleep Mode (End Devices only)
- Command Mode (Command Mode Sequence is issued)

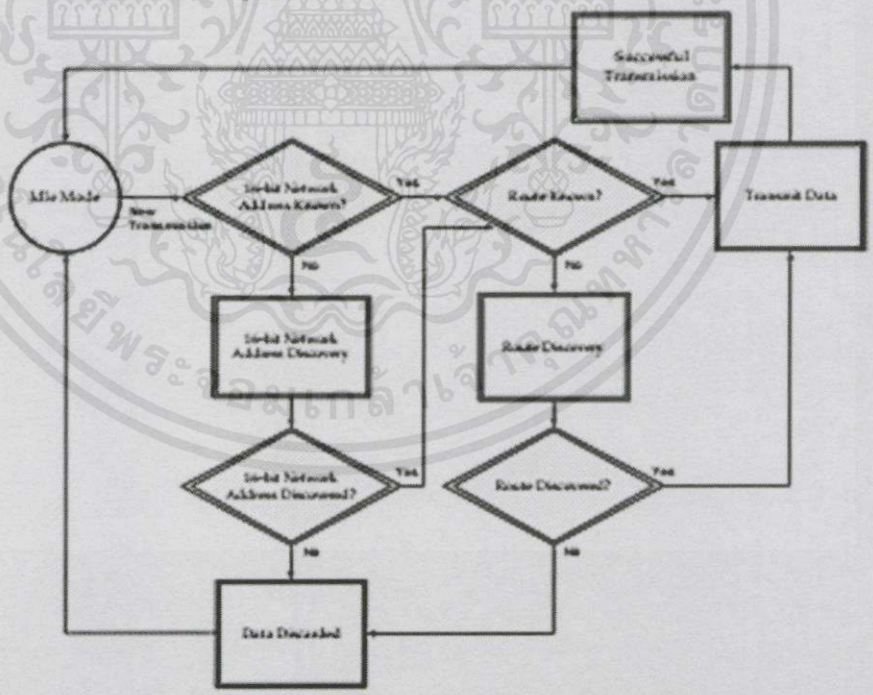
2.2.2. Transmit Mode

When serial data is received and is ready for packetization, the RF module will exit Idle Mode and attempt to transmit the data. The destination address determines which node(s) will receive the data.

Prior to transmitting the data, the module ensures that a 16-bit Network Address and route to the destination node have been established.

If the 16-bit Network Address is not known, Network Address Discovery will take place. If a route is not known, route discovery will take place for the purpose of establishing a route to the destination node. If a module with a matching Network Address is not discovered, the packet is discarded. The data will be transmitted once a route is established. If route discovery fails to establish a route, the packet will be discarded.

Figure 2-04. Transmit Mode Sequence



When data is transmitted from one node to another, a network-level acknowledgement is transmitted back across the established route to the source node. This acknowledgement packet indicates to the source node that the data packet was received by the destination node. If a network acknowledgement is not received, the source node will re-transmit the data. See Data Transmission and Routing in chapter 3 for more information.

2.2.3. Receive Mode

If a valid RF packet is received and its address matches the RF module's MY (16-bit Source Address) parameter, the data is transferred to the serial transmit buffer.

2.2.4. Command Mode

To modify or read RF Module parameters, the module must first enter into Command Mode - a state in which incoming serial characters are interpreted as commands. Refer to the API Mode section for an alternate means of configuring modules.

AT Command Mode

To Enter AT Command Mode:

Send the 3-character command sequence "+++" and observe guard times before and after the command characters. [Refer to the "Default AT Command Mode Sequence" below.]

Default AT Command Mode Sequence (for transition to Command Mode):

- No characters sent for one second [GT (Guard Times) parameter = 0x3E8]
- Input three plus characters ("+++") within one second [CC (Command Sequence Character) parameter = 0x2B.]
- No characters sent for one second [GT (Guard Times) parameter = 0x3E8]

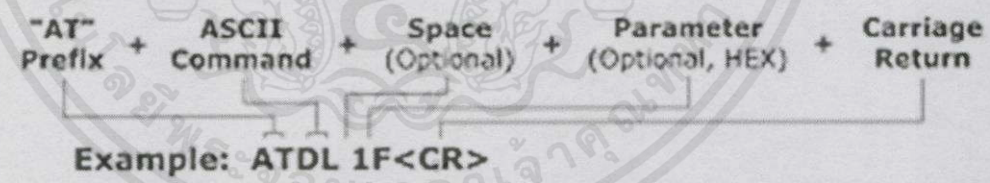
All of the parameter values in the sequence can be modified to reflect user preferences.

NOTE: Failure to enter AT Command Mode is most commonly due to baud rate mismatch. Ensure the 'Baud' setting on the "PC Settings" tab matches the interface data rate of the RF module. By default, the BD parameter = 3 (9600 bps).

To Send AT Commands:

Send AT commands and parameters using the syntax shown below.

Figure 2-05. Syntax for sending AT Commands



To read a parameter value stored in the RF module's register, omit the parameter field.

The preceding example would change the RF module Destination Address (Low) to "0x1F". To store the new value to non-volatile (long term) memory, subsequently send the WR (Write) command.

For modified parameter values to persist in the module's registry after a reset, changes must be saved to non-volatile memory using the WR (Write) Command. Otherwise, parameters are restored to previously saved values after the module is reset.

System Response. When a command is sent to the module, the module will parse and execute the command. Upon successful execution of a command, the module returns an "OK" message. If execution of a command results in an error, the module returns an "ERROR" message.

To Exit AT Command Mode:

1. Send the ATCN (Exit Command Mode) command (followed by a carriage return).
   [OR]
2. If no valid AT Commands are received within the time specified by CT (Command Mode Timeout) Command, the RF module automatically returns to Idle Mode.

For an example of programming the RF module using AT Commands and descriptions of each configurable parameter, refer to the "RF Module Configuration" chapter.

2.2.5. Sleep Mode

Sleep modes are supported on end devices only. Router and coordinator devices participate in routing data packets and are intended to be mains powered. End devices must be joined to a parent (router or coordinator) before they can participate on a ZigBee network. The parent device does not track when an end device is awake or asleep. Instead, the end device must inform the parent when it is able to receive data. The parent must be able to buffer incoming data packets destined for the end device until the end device can awake and receive the data. When an end device is able to receive data, it sends a poll command to the parent. When the parent router or coordinator receives the poll command, it will transmit any buffered data packets for the end device. Routers and coordinators are capable of buffering one broadcast transmission for sleeping end device children.

The SM, ST, SP, and SN commands are used to configure sleep mode operation.

Data Management

The SP command on the parent determines how long the parent will buffer a packet. It should be set to match the maximum SP value on any end device that may join to it. SP can be set up to 28 seconds (0xAF0).

End Device Sleep Modes

Pin Sleep

Setting SM=1 or SM=2 configures a device as a pin-sleep enabled end device. When operating in this mode, an end device monitors the Sleep_Request pin for a high state. When Sleep_Request goes high, the module enters sleep mode once any pending transmissions have finished. The module remains in a low power state until the Sleep_Request pin goes low.

When the module wakes from pin sleep, it sends a poll request to the parent to see if any data is pending for the end device. Since routers and coordinators can only buffer data up to 30 seconds, end devices should not remain in pin sleep longer than about 28 seconds if incoming data packets must be received. Using pin sleep for more than 28 seconds is recommended only if incoming data packets are not expected.

When the module wakes from a pin sleep mode, the CTS line goes low, and On/Sleep goes high.

Cyclic Sleep

Cyclic sleep allows the end device to sleep for a specified period of time. The period of time is specified by SP. Since routers and coordinators can only buffer data packets for up to 30 seconds, SP on end devices can be set up to 28 seconds (0xAF0). The module will wake after SP time and send a poll request to the parent to check for data. If any serial or RF data is received, the ST time is restarted. Once ST time has expired with no serial or RF activity, the end device will resume cyclic sleep operation.

When the module wakes, CTS goes low allowing the application to send serial data to the module if necessary. The On/Sleep indicator will be set high to alert the application that the end device has awakened. If serial or RF data is received, the ST timer will be reset, otherwise, the end device will resume low power operation.

Off board peripherals may wish to sleep longer than the maximum SP time of the end device. The SN command can be used to not wake off board peripherals for longer than SP time.

For example, if SP=28 seconds, and SN=5, the end device will wake up every 28 seconds and poll the parent for data. If no data is pending, the end device will return to sleep. In this example, if the parent has no data for the end device, On/Sleep will go high after 140 seconds, assuming the parent never has data for the end device. If the parent has data for the end device, On/Sleep will go high and the SN counter will reset.

3. ZigBee Networks

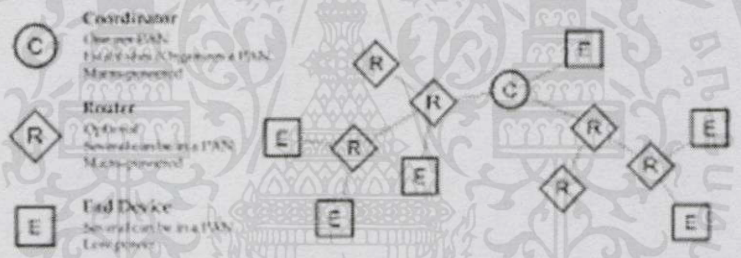
3.1. ZigBee Network Formation

A ZigBee Personal Area Network (PAN) consists of one coordinator and one or more routers and/or end devices. A ZigBee Personal Area Network (PAN) is created when a coordinator selects a channel and PAN ID to start on. Once the coordinator has started a PAN, it can allow router and end device nodes to join the PAN.

When a router or end device joins a PAN, it receives a 16-bit network address and can transmit data to or receive data from other devices in the PAN. Routers and the coordinator can allow other devices to join the PAN, and can assist in sending data through the network to ensure data is routed correctly to the intended recipient device. When a router or coordinator allows an end device to join the PAN, the end device that joined becomes a child of the router or coordinator that allowed the join.

End devices, however can transmit or receive data but cannot route data from one node to another, nor can they allow devices to join the PAN. End devices must always communicate directly to the parent they joined to. The parent router or coordinator can route data on behalf of an end device child to ensure it reaches the correct destination. End devices are intended to be battery powered and can support low power modes.

Figure 3-01. Node Types / Sample of a Basic ZigBee Network Topology



The network address of the PAN coordinator is always 0. When a router joins a PAN, it can also allow other routers and end devices to join to it. Joining establishes a parent/child relationship between two nodes. The node that allowed the join is the parent, and the node that joined is the child. The parent/child relationship is not necessary for routing data.

3.1.1. Starting a ZigBee Coordinator

When a coordinator first comes up, it performs an energy scan on mulitple channels (frequencies) to select an unused channel to start the PAN. After removing channels with high detected energy levels, the coordinator issues an 802.15.4 beacon request command on the remaining, low energy level channels. Any routers or coordinators respond to the beacon request frame with a small beacon transmission that indicates the PAN identifier (PAN ID) that they are operating on, and whether or not they are allowing joining.The coordinator will attempt to start on an unused PAN ID and channel. After starting, the coordinator may allow other devices to join its PAN.

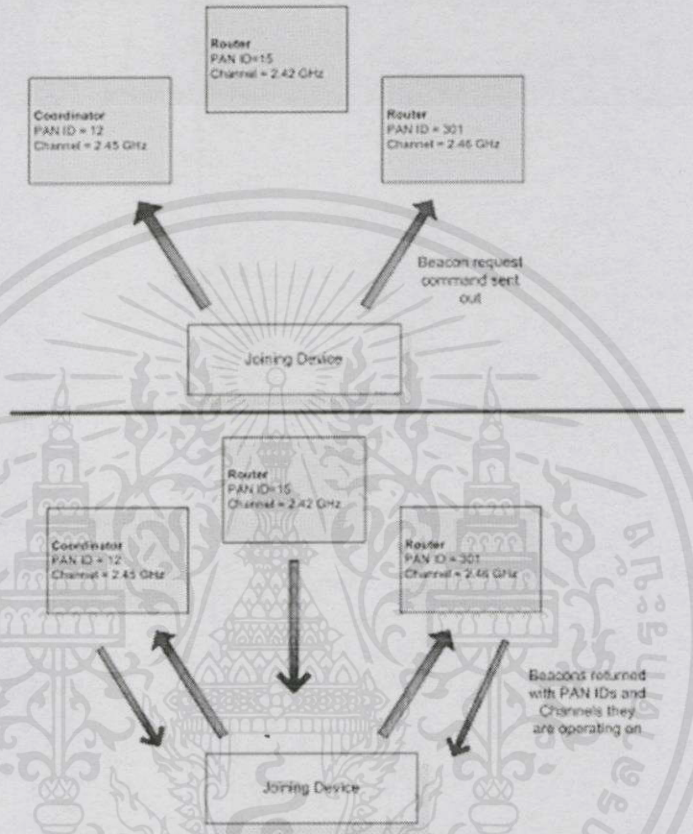
3.1.2. Joining a Router

When a router first comes up, it must locate and join a ZigBee PAN. To do this, it issues an 802.15.4 beacon request command on multiple channels to locate nearby PANs. Nearby routers and coordinators respond to the beacon request frame with a small beacon transmission, indicating which channel and PAN ID they are operating on. The router listens on each channel for these beacon frames, and determines which device it should join. If a valid PAN is found from one of the received beacons, the router issues a join request to the device that sent the beacon. If joining succeeds, the router will then receive a join confirmation from the device, indicating the join was successful. Once the router joins the PAN, it can communicate with other devices on the PAN and allow new devices to join to it.

3.1.3. Joining an End Device

When an end device first comes up, it must also locate and join a PAN. End devices follow the same process as a router to join a PAN. Once the end device has successfully joined a PAN, it can communicate with other devices on the PAN. However, since end devices cannot route data, it must always communicate directly with its parent and allow the parent to route data in its behalf.

Figure 3-02. Demonstration of Beacon Request and Beacon transmissions that take place during joining.



3.2. ZigBee Network Communications

Zigbee supports device addressing and application layer addressing. Device addressing specifies the destination address of the device a packet is destined to. Application layer addressing indicates a particular application recipient, known as a Zigbee endpoint, along with message type fields called cluster IDs.

3.2.1. ZigBee Device Addressing

The 802.15.4 protocol upon which the ZigBee protocol is built specifies two address types:

- 16-bit Network Addresses
- 64-bit Addresses

16-bit Network Addresses

A 16-bit Network Address is assigned to a node when the node joins a network. The Network Address is unique to each node in the network. However, Network Addresses are not static - it can change.

The following two conditions will cause a node to receive a new Network Address:

1. If an end device cannot communicate with its parent it may need to leave the network and rejoin to find a new parent.
2. If the device type changes from router to end device, or vice-versa, the device will leave the network and rejoin as the new device type.

ZigBee requires that data be sent to the 16-bit network address of the destination device. This requires that the 16-bit address be discovered before transmitting data. See 3.2.3 Network Address Discovery for more information.

64-bit Addresses

Each node contains a unique 64-bit address. The 64-bit address uniquely identifies a node and is permanent.

3.2.2. ZigBee Application-layer Addressing

The ZigBee application layers define endpoints and cluster identifiers (cluster IDs) that are used to address individual services or applications on a device. An endpoint is a distinct task or application that runs on a ZigBee device, similar to a TCP port. Each ZigBee device may support one or more endpoints. Cluster IDs define a particular function or action on a device. Cluster IDs in the ZigBee home controls lighting profile, for example, would include actions such as "TurnLightOn", "TurnLightOff", "DimLight", etc.

Suppose a single radio controls a light dimmer and one or more light switches. The dimmer and switches could be assigned to different endpoint values. To send a message to the dimmer, a remote radio would transmit a message to the dimmer endpoint on the radio. In this example, the radio might support cluster IDs to "TurnLightOn", "TurnLightOff", or "DimLight". Thus, for radio A to turn off a light on radio B, radio A would send a transmission to the light switch endpoint on radio B, using cluster ID "TurnLightOff". This is shown in the figure below.

Figure 3-03. ZigBee Data Transmission Higher Layer Addressing Fields

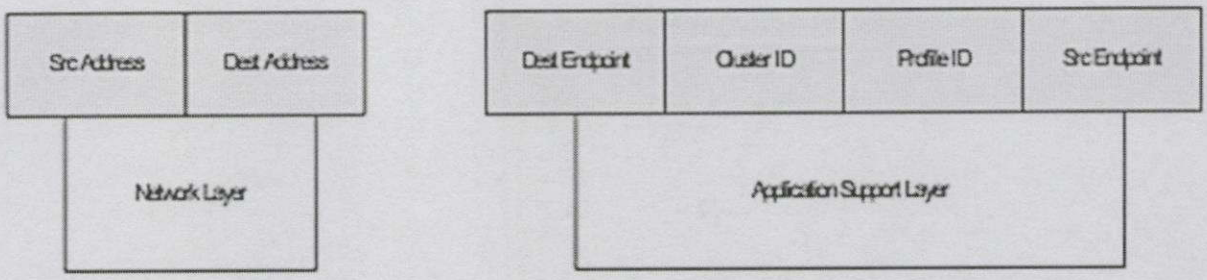
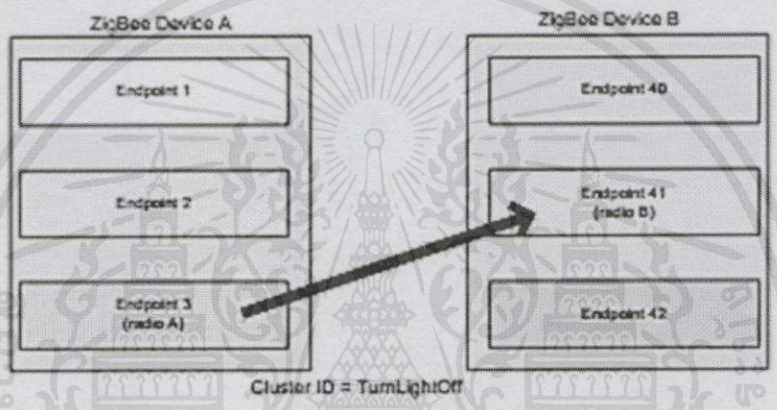


Figure 3-04. ZigBee Layer-Addressing Example



3.2.3. Data Transmission and Routing

All data packets are addressed using both device and application layer addressing fields. Data can be sent as a broadcast, multicast, or unicast transmission.

Broadcast Transmissions

Broadcast transmissions within the ZigBee protocol are intended to be propagated throughout the entire network such that all nodes receive the transmission. To accomplish this, all devices that receive a broadcast transmission will retransmit the packet 3 times. Each node that transmits the broadcast will also create an entry in a local broadcast transmission table. This entry is used to keep track of each received broadcast packet to ensure the packets are not endlessly transmitted. Each entry persists for 8 seconds. The broadcast transmission table holds 8 entries.

Since broadcast transmissions are retransmitted by each device in the network, broadcast messages should be used sparingly.

Multicast Transmissions

Multicast transmissions operate similar to broadcast transmissions. Data packets are broadcast throughout the network in a similar fashion. However, only devices that are part of the multicast group will receive the data packets.

Unicast Transmissions

Unicast ZigBee transmissions are always addressed to the 16-bit address of the destination device. However, only the 64-bit address of a device is permanent; the 16-bit address can change. Therefore, ZigBee devices may employ network address discovery to identify the current 16-bit

address that corresponds to a known 64-bit address. Once the 16-bit address is known, a route to the destination device must be discovered. ZigBee employs mesh routing using the Ad-hoc On-demand Distance Vector routing (AODV) protocol to establish a route between the source device and the destination.

Network Address Discovery

Data transmissions are always sent to the 16-bit network address of the destination device. However, since the 64-bit address is unique to each device and is generally known, ZigBee devices must discover the network address that was assigned to a particular device when it joined the PAN before they can transmit data.

To do this, the device initiating a transmission sends a broadcast network address discovery transmission throughout the network. This packet contains the 64-bit address of the device the initiator needs to send data to. Devices that receive this broadcast transmission check to see if their 64-bit address matches the 64-bit address contained in the broadcast transmission. If the addresses match, the device sends a response packet back to the initiator, providing the network address of the device with the matching 64-bit address. When this response is received, the initiator can then transmit data.

Mesh Routing

Mesh routing allows data packets to traverse multiple nodes (hops) in a network to route data from a source to a destination. The route a packet can take in a mesh network is independent of the parent/child relationships established during joining. Before transmitting a data packet from source to destination nodes, a route must be established. Route discovery is based on the AODV (Ad-hoc On-demand Distance Vector routing) protocol.

AODV (Ad-hoc On-demand Distance Vector) Routing Algorithm

Routing under the AODV protocol is accomplished using tables in each node that store in the next hop (intermediary node between source and destination nodes) for a destination node. If a next hop is not known, route discovery must take place in order to find a path. Since only a limited number of routes can be stored on a Router, route discovery will take place more often on a large network with communication between many different nodes.

When a source node must discover a route to a destination node, it sends a broadcast route request command. The route request command contains the source Network Address, the destination Network Address and a Path Cost field (a metric for measuring route quality). As the route request command is propagated through the network (refer to the Broadcast Transmission), each node that re-broadcasts the message updates the Path Cost field and creates a temporary entry in its route discovery table.

When the destination node receives a route request, it compares the 'path cost' field against previously received route request commands. If the path cost stored in the route request is better than any previously received, the destination node will transmit a route reply packet to the node that originated the route request. Intermediate nodes receive and forward the route reply packet to the Source Node (the node that originated route request).

Retries and Acknowledgments

ZigBee includes acknowledgment packets at both the Mac and Application Support (APS) layers. When data is transmitted to remote device, it may traverse multiple hops to reach the destination. As data is transmitted from one node to its neighbor, an acknowledgment packet (Ack) is transmitted in the opposite direction to indicate that the transmission was successfully received. If the Ack is not received, the transmitting device will retransmit the data, up to 4 times. This Ack is called the Mac layer acknowledgment.

In addition, the device that originated the transmission expects to receive an acknowledgment packet (Ack) from the destination device. This Ack will traverse the same path that the data traversed, but in the opposite direction. If the originator fails to receive this Ack, it will retransmit the data, up to 2 times until an Ack is received. This Ack is called the ZigBee APS layer acknowledgment.

Refer to the ZigBee specification for more details.

---

4. XBee Series 2 Network Formation

4.1. XBee Series 2 Network Formation

To create a ZigBee network, a coordinator must be started on a channel and PAN ID. Once the coordinator has started, routers and end device can join the network. Routers and coordinator devices can support up to 8 end device children each. Network formation is governed by the SC (Scan Channels), ID (PAN ID), SD (Scan Duration), and NJ (Node Join Time) commands. The SC and ID settings must be written using the WR command to affect network formation and joining.

4.1.1. Starting an XBee Series 2 Coordinator

In order to form a network, a coordinator must select an unused operating channel and PAN ID on behalf of its network. To do this, the coordinator first performs an energy scan on all channels specified by its SC (Scan Channels) parameter. The scan time on each channel is determined by the SD (Scan Duration) parameter. Once the energy scan is completed, the coordinator sends a beacon request on each of the SC channels and listens for any beacons. The information from the energy scan and the beacon scan (active scan) is used to select an unused channel and PAN ID. If the ID (PAN ID) parameter is set to 0xFFFF, the coordinator will select a random PAN ID. Otherwise, the coordinator will start on the PAN ID specified by its ID parameter.

After the coordinator has started, it will allow nodes to join to it for a time based on its NJ (Node Join Time) parameter. If the Associated LED function is enabled (D5 (DIO5 Configuration) command), the Associate pin (pin 15) will toggle its output state 1x per second after the coordinator stated. At this point, the operating channel and PAN ID can be read using the CH (Operating Channel) and ID (PAN ID) commands. The 16-bit address of the coordinator is always 0. If API is enable (AP parameter > 0): The API modem status "coordinator Started" frame is sent out the UART. The AI (Association Indication) command can be used at any point during the coordinator startup routine to determine the status of the startup operation.

4.1.2. Joining an XBee Series 2 Router to an existing PAN

Before a router can participate in a ZigBee network, the router must locate a coordinator or another router that has already joined a PAN, and attempt to join to it. To do this, it sends a beacon request frame on each of the SC channels and listens for beacon frames. The scan duration on each channel is determined by the SD parameter. The joining router will evaluate the received beacons to find a coordinator or router that is allowing joins on a valid PAN ID, and attempt to join to that device. If ID = 0xFFFF, the router will attempt to join to a device on any PAN ID. Otherwise, the router will only attempt joining with a device that operates on the PAN ID specified by the ID parameter. If a valid router/ coordinator is found, the router will attempt to join to that node. If the join succeeds, the Router has successfully started.

After the Router has started, it will allow nodes to join to it for a time based on the NJ (Node Join Time) parameter. If the Associated LED function is enabled(D5 (DIO5 Configuration) command) the Associate pin (pin 15) will toggle its output state 2x per second after the router has joined. At this point, the operating channel and PAN ID can be read using the CH (Operating Channel) and ID (PAN ID) commands. The 16-bit Network Address of the router can be read using the MY (16-bit Source Address) command. If API is enabled (AP parameter > 0): The API modem status "Joined" is sent out the UART. The AI (Association Indication) command can be used at any point during the router join routine to know the status of the startup operation.

4.1.3. Joining an XBee Series 2 End Device to an Existing PAN

Joining an end device to a PAN is similar to joining a router. Once the end device joins a PAN, however, the end device cannot allow other devices to join to it. If the Associate LED function is enabled (D5 (DIO5 Configuration) command), the Associate pin (pin 15) will toggle its output state 2x per second after the end device has joined. At this point, the operating channel and PAN ID can be read using the CH (Operating Channel) and ID (PAN ID) commands. The 16-bit network

address of the end device can be read using the MY (16-bit Source Address) command. If API is enabled (AP parameter > 0), the API modem status "Joined" is sent out the UART. The AI (Association Indication) command can be used at any point during the end device join routine to know the status of the startup operation.

4.1.4. Network Reset

Once a coordinator has started, or a router or end device has joined the network, the device will continue operating on that channel and PAN ID unless one of the following occurs:

1. The ID parameter changes, and is saved using the WR command
2. The SC parameter changes and is saved using the WR command, such that the current operating channel is not included in the new SC parameter
3. The NR command is issued with either 0 or 1 as a parameter

If any of the above occurs on a coordinator, the coordinator will attempt to restart on a channel and PAN ID based on the new saved ID and SC commands. On a router or end device, the above conditions will cause the device to leave the network (if previously joined) and attempt to join a new PAN using the saved ID and SC parameters.

4.2. XBee Series 2 Addressing

XBee modules support both ZigBee device addressing and application-layer addressing.

4.2.1. Device Addressing

All XBee/XBee-Pro modules can be identified by their unique 64-bit adresses or a user-configurable ASCII string identifier The 64-bit address of a module can read using the SH and SL commands. The ASCII string identifier is configured using the NI command. To transmit using device addressing, only the destination address must be configured. The destination address can be specified using either the destination device's 64-bit address or its NI-string. The XBee modules also support coordinator and broadcast addressing modes. Device addressing in the AT firmware is configured using the DL, DH, or DN commands. In the API firmware, the ZigBee Transmit Request API frame (0x10) can be used to specify destination addresses.

64-Bit Addressing

To address a node by its 64-bit address, the destination 64-bit address must be set to match the 64-bit address of the remote. In the AT firmware, the DH and DL commands set the destination 64-bit address. In the API firmware, the destination 64-bit address is set in the ZigBee Transmit Request frame. The coordinator can be addressed by either setting the destination address to 0 or by setting it to match the coordinator's 64-bit address. Broadcast transmissions can be sent by setting the 64-bit address to 0x000000000000FFFF.

To send a packet to an RF module using its 64-bit Address (Transparent Mode)

> Set the DH (Destination Address High) and DL (Destination Address Low) parameters of the source node to match the 64-bit Address (SH (Serial Number High) and SL (Serial Number Low) parameters) of the destination node.

To send a packet to an RF module using its 64-bit Address (API Mode)

> Use the ZigBee Transmit Request API frame to set the DH (Destination Address High) and DL (Destination Address Low) parameters of the source node to match the 64-bit Address (SH (Serial Number High) and SL (Serial Number Low) parameters) of the destination node.
>
> If the 16-bit address of the destination node is not known, set 16-bit Destination Network Address to 0xFFFE (refer to the 'API Addressing section below).

Since the ZigBee protocol relies on the 16-bit Network Address for routing, the 64-bit address must be converted into a 16-bit Network Address prior to transmitting data. If a module does not know the 16-bit Network Address for a given 64-bit address, it will transmit a broadcast Network Address Discovery command. The module with a matching 64-bit address will transmit its 16-bit network address back. Once the network address is discovered, the data will be transmitted.

The modules maintain a table that can store up to seven 64-bit addresses and their corresponding 16-bit Network Addresses.

API Addressing

API Mode provides the ability to store and maintain 16-bit Network Address tables on an external processor. The 16-bit Network Address information is provided to the application through the following:

- The ZigBee Transmit Status Frame
  (contains the current 16-bit Network Address of the remote)
- The ND and DN commands
  (return 64-bit and 16-bit Network Addresses of remote nodes)

With this information, a table can be built in an application that maps a 64-bit Address to the corresponding 16-bit Network Address.

The ZigBee Transmit Request API frame specifies the 64-bit Address and the Network Address (if known) that the packet should be sent to. By supplying both addresses, the module will forego Network Address Discovery and immediately attempt to route the data packet to the remote. If the Network Address of a particular remote changes, Network Address and route discovery will

take place to establish a new route to the correct node. Upon successful packet delivery, the TX Status Frame will indicate the correct Network Address of the remote.

Table 4-01. Sample table mapping 64-bit Addresses to 16-bit Network Addresses

Index	64-bit Address	16-bit Network Address
0	0013 4000 4000 0001	1234
1	0013 4000 4000 0002	5678
2	0013 4000 4000 01A0	A479
3	0013 4000 4000 0220	1F70

NI-String Addressing

The NI string can alternatively be used to address a remote module.

To send a packet to an RF module using its NI-string (Transparent Mode)

Issue the DN (Destination Node) command using the NI (Node Identifier)-string of the destination node as the parameter.

To send a packet to an RF module using its NI-string (API Mode)

Issue the DN command as stated above using the AT Command API frame.

When the DN command is issued, a broadcast transmission is sent across the network to discover the module that has a matching NI (Node Identifier) parameter. If a module is discovered with a matching NI-string, the DH and DL parameters will be configured to address the destination node and the command will return both the 64-bit Address and the 16-bit Network Address of the discovered node. Data can be transmitted after the DN (Destination Node) command finishes.

the AO command. See "API Frames" section for details.

Coordinator Addressing

A Coordinator can be addressed using its 64-bit address or NI string as described in the "NI-String Addressing" section. Alternatively, since the ZigBee Coordinator has a Network Address of "0", it can be addressed by its 16-bit Network Address.

To send a transmission to a Coordinator using its 16-bit Network Address:

Set the Destination Addresses of the transmitting module as shown below:
DL (Destination Low Address) = 0
DH (Destination High Address) = 0

Broadcast Addressing

Broadcast transmissions are sent using a 64-bit address of 0x0000FFFF. Any RF module in the PAN will accept a packet that contains a broadcast address. When configured to operate in Broadcast Mode, receiving modules do not send ACKs (Acknowledgements).

To send a broadcast packet to all modules

Set the Destination Addresses of the transmitting module as shown below:
DL (Destination Low Address) = 0x0000FFFF
DH (Destination High Address) = 0x00000000

NOTE: When programming the module, parameters are entered in hexadecimal notation (without the "0x" prefix). Leading zeros may be omitted.

Refer to the "Broadcast Transmissions" for more information.

4.2.2. Application-layer Addressing

Application-layer addressing allows the application to specify endpoint and cluster ID values for each transmission. Addressing multiple endpoints and cluster IDs can be accomplished by explicitly setting these values as needed.

In AT firmware, application-layer addressing must be enabled using the ZA command. When application-layer addressing is enabled, the DE and SE commands specify the source and destination endpoints, and the CI command sets the cluster ID that will be used in the transmission.

In API firmware, the Explicit Addressing ZigBee Command frame (0x11) can be used to configure the endpoint and cluster ID addressing parameters as needed. The destination device can indicate application-layer addressing information if the Explicit Receive API frame is addressing information using either the explicit receive indicator or the binding receive API frames. The receive RF data frame is set using Binding Table Addressing

The XBee Series 2 modules maintain several entries in a binding table. The binding table contains a destination 64-bit address, a type field, and endpoints for each transmission. Non-broadcast transmissions make use of the binding table to specify the addressing values for the transmission. Some entries in the binding table are reserved by MaxStream for special purposes. Binding table entries can be accessed by setting the BI command to a valid index in AT firmware, or by using the Binding Table API Command frame in the API firmware. The binding table entries are organized as follows.

Table 4-02.

Binding Table Index	Name	Access
0	Coordinator Binding	Read-Write
1	Tx-Aggregation Binding	Read-Only
2	Tx-Explicit Binding	Read-Write
3-4	Command Binding	Read-Only
5-8	Received Data Bindings	Read-Only
9	User Bindings.	Read-Write

Coordinator Binding

The coordinator binding contains the 64-bit address of the coordinator. This table entry is populated when the device joins the network.

Tx-Aggregation Binding

This binding table entry contains the 64-bit address of the aggregate (sink) node if one exists. Data can be sent to the aggregate node by addressing this index in the binding table.

Tx-Explicit Binding

The Tx-Explicit binding table entry contains the destination address and endpoint information from the last explicit transmission that was issued. This entry is modified whenever explicit addressing is used in either the AT or API firmware as described in the "XBee Series 2 Addressing" section.

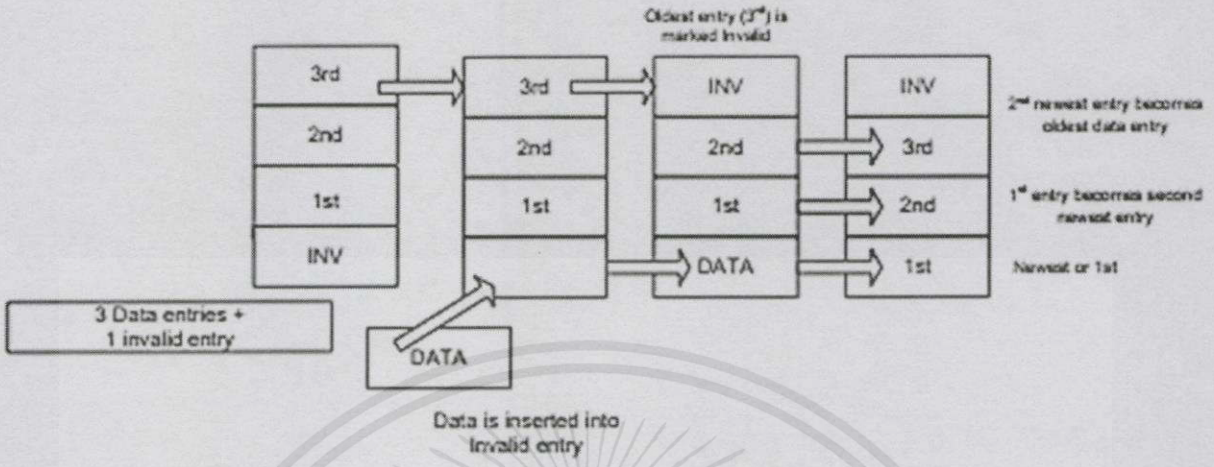
Command Binding

If a remote command request is received, the command binding entry stores information from the device that initiated the command. For example, if the ND or DN command is issued, this binding table entry would contain the source address of the device that sent the ND command.

Received Data Bindings

The received data bindings contain addressing information for the last three received data packets. The fourth entry is marked invalid. When a data packet is received, the address and endpoint information is stored into the invalid entry. Then, the oldest entry is made invalid. Thus, once an entry is created in the Received Data binding indexes, it will remain valid until three more RF data packets are received.

Figure 4-05. Demonstration of how entries in the received data bindings are replaced when an RF data packet is received.



User Bindings

These entries can be created and maintained by the application if needed. The following commands can be used to modify the user bindings. See the command descriptions for formatting details.

Table 4-03.

Command	Name	Description
B+	Add Binding	Creates a binding table entry at a specified User Binding index.
B-	Remove Binding	Removes a binding from a specified User Binding index.
BV	View Binding	Views one or more bindings in the binding table.
WB	Write Binding	Writes the binding table to non-volatile memory.

Multicast Addressing

Multicast addressing sends a broadcast message that will only be received by devices who subscribe to a multicast group. The binding table is used to subscribe to a multicast group. To send a multicast transmission, a binding table entry must exist where the type field is set to the multicast type value. The 64-bit address in this entry becomes a multicast group address. Only remote devices with a matching 64-bit multicast group will receive multicast transmissions. Once the binding table is configured with a multicast binding entry, the binding table index can be specified for a transmission using the BI command (AT firmware), or the Binding Table API Command Frame (API firmware). See the XBee Binding Table section for details.

Endpoint Addressing

The ZigBee specification, Ember stack, and MaxStream application have reserved some endpoints for different uses. Some of these endpoints are not accessible. Applications that will support custom endpoints should select endpoints not already used by ZigBee, Ember, or MaxStream.

The cluster ID used by MaxStream on the serial data endpoint for serial data transmissions is 0x11.

4.2.3. XBee Series 2 Endpoint Table

The XBee Series 2 modules maintain a table of supported endpoints. If an endpoint will be used as the source endpoint in a data transmission, the endpoint must first be defined in the endpoint table.

The XBee Series 2 endpoint table operates similar to the binding table. Entries may be added, removed, or viewed using the E+, E-, and EV commands respectively. Some table entries are reserved for special purposes

Table 4-04.

Endpoint Table Index	Name	Access
0	Command Endpoint	Read-Only
1	Data Endpoint	Read-Only
2	Tx-Explicit Endpoint	Read-Write
3- 4	User Endpoints	Read-Write

Command Endpoint

The command endpoint is used to send or reply to various commands. This endpoint must exist in the application.

Data Endpoint

This endpoint is used to send serial data to other XBee Series 2 modules. It must always exist in the application.

Tx-Explicit Endpoint

This entry is used as needed to define the source endpoint that must be defined for a data transmission. If a transmit request is made, and the specified source endpoint does not exist, it will be created temporarily at this endpoint table index.

User Endpoints

User endpoints are controlled entirely by the application. These endpoints may be added, removed, or viewed in the API firmware using the following commandsSee the command descriptions for command formatting details. At present, changes to the endpoint table are saved to non-volatile memory when WR is issued..

Table 4-05. ZigBee Data Transmissions Addressing Fields

Command	Name	Description
E+	Add Endpoint	Creates an endpoint entry at a specified user endpoint index.
E-	Remove Endpoint	Removes an endpoint entry from a specified user endpoint index.
EV	View Endpoint	Views one or more endpoints in the endpoint table.

4.3. Advanced Network Features

Network Mapping

Network mapping has provisions to identify all devices on a PAN.There are currently two ways to do this either through the Node Discover (ND) Command or the API Child Joined Indicator. Both are explained below.

Node Discover (ND) Command

Issuing the ND command on a device sends a broadcast node discovery command throughout the PAN. All devices that receive the command will send a response that includes the device's 64-bit and 16-bit addresses, along with the NI-string and other information.

API Child Joined Indicator

Routers and end devices can be configured to send a transmission after joining to alert the coordinator, or the entire network, that the device has joined the network. When this message is transmitted, the receiving device(s), if running API firmware, will send an advanced modem status indicator out the UART to indicate the 64-bit and 16-bit addresses of the joining device.

4.4. I.O. Line Configuration

The XBee Series 2 modules support both analog input and digital IO line modes on several configurable pins.

Configuring A/D and Digital Lines

The following table lists the pin functions supported on the modules

.

Table 4-06.

Module Pin Names	Module Pin Numbers	Configuration Command
CD/DIO12	4	P2
PWM0/RSSI/DIO10	6	P0
PWM/DIO11	7	P1
SLEEP_RQ/DIO8	9	IO Configuration not supported
DIO4	11	D4
CTS/DIO7	12	D7
ON_SLEEP/DIO9	13	IO Configuration not supported
ASSOC/DIO6	15	D5
RTS/DIO6	16	D6
AD3/DIO3	17	D3
AD2/DIO2	18	D2
AD1/DIO1	19	D1
AD0/DIO0	20	D0

Setting the configuration command that corresponds to a particular pin will configure the pin. Parameters for the pin configuration commands typically include the following:

.

Table 4-07.

Pin Command Parameter	Description
0	Unmonitored digital input
1	Reserved for pin-specific alternate functionalities
2	Analog input, single ended (A/D pins only)
3	Digital input, monitored
4	Digital output, default low
5	Digital output, default high
6-9	Alternate functionalities, where applicable
0	Unmonitored digital input
1	Reserved for pin-specific alternate functionalities
2	Analog input, single ended (A/D pins only)
3	Digital input, monitored
4	Digital output, default low
5	Digital output, default high

See the command table for more information. Pullup resistors for each digital input can be enabled using the PR command.

Sampling A/D and Digital Input Lines

The IS command can be used to read the current value of all enabled A/D and digital input lines. The format for the IS response is shown below. At the time, only one sample set is supported in this frame.

Bytes	Name	Description
1	Sample sets in packet	Number of sample sets in the packet
2	Digital Channel Mask	Each bit in the digital channel mask corresponds to one digital IO line. The bits, from LSB to MSB, correspond to DIO0-DIO5 on the module. For example a digital channel mask of 0x002F means DIO0,1,2,3, and 5 are enabled as digital input lines.
1	Analog Channel Mask	Each bit in the analog channel mask corresponds to one analog channel. The bits from LSB to MSB correspond to AIN0-AIN7 on the module. For example, if the analog channel mask is 0x06, AIN1 and AIN3 are enabled as analog input lines.
Var	Sampled Data Set	A sample set consisting of 1 sample for each enabled ADC and/or DIO channel. If any digital input lines are enabled, the first two bytes indicate the state of all enabled digital input lines. Each bit in these two bytes corresponds to one digital IO line, similar to the way each bit in the digiossia channel mask corresponds. Note: only the digital input line that are enabled in the detail channel mask have valid readings. Channels that are not enabled as digital input lines will return a 0 in the sampled data set. If no pins are configured as digital inputs, these 2 bytes will be omitted. Following the digital input data, if any, each enabled analog channel will return 2 bytes (10bits). The analog data is scaled such that 0 represents 0V, and 0x3FF=1.2V. The analog input lines cannot measure more than 1.2V. Information for each enabled analog channel is returned in order, starting with AIN0 and finishing with AIN4. Only enabled analog input channels will return data.

The AT firmware returns a carriage return delimited list containing the above-listed fields. The API firmware returns an AT command response API frame with the IO data included in the command data portion of the packet.

Example	Sample AT Response
0x01\r	[1 sample set]
0x0C0C\r	[Digital Inputs: DIO 2, 3, 10, 11 low]
0x03\r	[Analog Inputs: ADC0 0, 1]
0x0408\r	[Digital input states: DIO 3, 10 high, DIO 2, 11 low]
0x03D0\r	[Analog input ADIO 0= 0x3D0]
0x0124\r	[Analog input ADIO 1=0x120]

To convert the A/D reading to mV, do the following:

AD(mV)= (ADIO reading/0x3FF)*1200mV

The reading in the sample frame represent voltage inputs of 1144.9 and 342.5mV for ADIO0 and ADIO1 respectively.