การพัฒนาระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สาย

AN IMPLEMENTATION OF WIRELESS SENSOR NETWORK FOR MONITORING SYSTEM

ศุภาพงษ์ วุฒิพราหมณ์

อาจารย์ที่ปรึกษา

รศ.ดร. โชติพัชร์ ภรณวลัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2551

AN IMPLEMENTATION OF WIRELESS SENSOR NETWORK FOR MONITORING SYSTEM

SUPAPONG WUTTHIPRAM

A PROJECT SUBMITTED IN PARTIAL FULLFILLMENT

BACHELOR OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2/2008

ใบรับรองปริญญานิพนธ์ ประจำปีการศึกษา 2551

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สาย

AN IMPLEMENTATION OF WIRELESS SENSOR NETWORK FOR MONITORING SYSTEM

ผู้จัดทำ

1. นาย ศุภาพงษ์ วุฒิพราหมณ์ รหัสนักศึกษา 48070167

.....อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. โชติพัชร์ ภรณวลัย)

หัวข้อปริญญานิพนธ์	การพัฒนาระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สาย
นักศึกษา	นายศุภาพงษ์ วุฒิพราหมณ์
รหัสประจำตัว	48070167
ปริญญา	วิทยาศาสตรบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
ปีการศึกษา	2551
อาจารย์ที่ปรึกษา	รศ.ดร. โชติพัชร์ ภวณวลัย

บทคัดย่อ

ในปัจจุบันการศึกษาสภาพแวดล้อมเป็นสิ่งที่คนหันกลับมาใส่ใจ เนื่องจากการเปลี่ยนแปลงสภาพแวดล้อมมีผลทำให้เกิดผลกระทบที่ก่อให้เกิดความเสียหายต่อทรัพย์สิน ซึ่งทางคณะผู้จัดทำได้ตระหนักถึงปัญหาข้างต้นจึงได้พัฒนาระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สาย ซึ่งช่วยเพิ่มความสามารถในการเฝ้าระวังสภาพแวดล้อมที่มีปัจจัยทำให้เกิดความเสียหายต่อทรัพย์สิน แล้วนำข้อมูลที่ได้จากตัวตรวจจับมาแสดงผลผ่านเว็บบราวเซอร์ ในรูปแบบของกราฟและตารางเพื่อให้ผู้ใช้สามารถเข้าใจการเปลี่ยนแปลงของสภาพแวดล้อมในช่วงเวลาหนึ่ง โครงงานนี้ใช้ตัวตรวจจับที่สามารถตรวจจับความสว่างของแสง และอุณหภูมิ และสามารถนำไปประยุกต์เพื่อใช้ติดตามเฝ้าระวังไม่ให้เกิดความเสียหายต่อทรัพย์สินภายในอาคาร

Project Title	An Implementation of Wireless Sensor Network for Monitoring System
Student	Mr. Supapong Wutthipram
Student ID	48070167
Degree	Bachelor of Science
Major	Information Technology
Academic Year	2008
Project Advisor	Assoc.Prof. Dr.Chotipat Pornavalai

ABSTRACT

In the present time, environment is the most important thing to concern. In some case, the environment problem affects directly to property, for example, some machine create a problem when operate at high temperature. Therefore, by knowing the problem, the creator invents the wireless watching system in order to protect the property from the mistake of the machine. The data from the system will be sent through web browser in the form of graph and table in order to inform the people who control the system about the change in environment in the period of time. In this project, light and temperature indicator are used to protect the building property

กิตติกรรมประกาศ

โครงงานพัฒนาระบบงานนี้เกิดขึ้น และสำเร็จลุล่วงไปได้ด้วยดี ผู้จัดทำโครงงานขอกราบขอบพระคุณ รศ.ดร. โชติพัชร์ ภรณวลัย ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน ที่ได้กรุณาเสียสละเวลาอันมีค่าในการให้คำแนะนำแนวคิดในการจัดทำโครงงาน ให้คำปรึกษาด้านวิชาการที่เป็นประโยชน์ในการทำโครงงาน และให้ความช่วยเหลือด้านอื่นๆ ทั้งด้านสถานที่ทำโครงงาน เครื่องมือและอุปกรณ์ในการจัดทำโครงงาน การแก้ไขเอกสาร เรียบเรียงเอกสาร รวมทั้งได้รับการดูแลเอาใจใส่ให้ความเมตตา และให้กำลังใจแก่ผู้จัดทำด้วยดีเสมอมา ผู้จัดทำมีความซาบซึ้งในความกรุณาเป็นอย่างยิ่ง

ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ที่ให้กำเนิด ให้การศึกษา ให้กำลังใจและเป็นแรงผลักดันให้ผู้จัดทำมีกำลังใจที่จะมุ่งมั่นในการศึกษาครั้งนี้จนเป็นผลสำเร็จลุล่วงด้วยดี

ขอกราบขอบพระคุณคณาจารย์คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้แก่ผู้จัดทำ คอยให้คำปรึกษาและชี้แนะแนวทางในการแก้ไขปัญหาต่างๆ ให้สำเร็จลุล่วงไปได้

สุดท้ายขอขอบคุณพี่ น้อง และเพื่อนคณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้ความช่วยเหลือ คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

คุณค่าและประโยชน์อันพึงมาจากโครงงานฉบับนี้ผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

ศุภาพงษ์ วุฒิพราหมณ์

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญตาราง(ต่อ)

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบัน มีการใช้เทคโนโลยีต่างๆเข้ามาสนับสนุนผู้ใช้งานให้ทำงานได้ง่ายขึ้น และมีความถูกต้องแม่นยำเพิ่มขึ้น โดยการนำเทคโนโลยีมาประยุกต์ใช้กับการเฝ้าตรวจตราในพื้นที่หนึ่งๆ เช่น การตรวจวัดอุณหภูมิภายในโรงงานเก็บสารเคมี การตรวจวัดแสงภายในพื้นที่ที่ควบคุมแสง ตัวตรวจจับการเคลื่อนไหวภายในพื้นที่สนามรบ เป็นต้น แต่เนื่องจากด้วยเทคโนโลยีในอดีตไม่สามารถตอบสนองต่อความต้องการได้ เช่น ตัวตรวจจับไม่สามารถส่งข้อมูลไปยังระบบได้เอง จะต้องมีเจ้าหน้าที่คอยตรวจดูผลตลอดเวลา หรือความสะดวกที่จะย้ายจุดตรวจไปยังตำแหน่งใหม่ เนื่องจากใช้การเดินสายจ่ายพลังงานและเส้นทางรับส่งข้อมูล ทำให้การตรวจตราพื้นที่หนึ่งๆ ทำได้ไม่สะดวก ดังนั้นเครือข่ายตรวจจับแบบไร้สายจึงได้รับการนำมาประยุกต์ใช้ เพื่อเพิ่มความสามารถในตรวจตรา ควบคุมพื้นที่นั้นให้มีความมั่นคงและปลอดภัย แน่นอนมากยิ่งขึ้น

เครือข่ายตรวจจับแบบไร้สาย (Wireless Sensor Network) เป็นเครือข่ายไร้สายที่ประกอบด้วยอุปกรณ์ที่ทำงานอย่างเป็นอิสระ โดยใช้ตัวตรวจจับทำงานร่วมกันในการเฝ้าสังเกตการณ์สภาพแวดล้อม เช่น อุณหภูมิ เสียง การสั่นสะเทือน ความกดดัน การเคลื่อนไหว หรือของเสีย ในสถานที่แตกต่างกัน แล้วส่งข้อมูลไปยังสถานีฐานซึ่งทำหน้าที่เป็นประตูทางออกไปยังผู้ใช้งาน ผ่านทางโหนดต่างๆ (หรือที่เรียกว่าโมท (Mote)) ในเครือข่าย โดยโหนดทำงานภายใต้สภาพแวดล้อมที่จำกัด อาทิ พลังงาน หน่วยความจำ แบนด์วิดธ์ (bandwidth) และความสามารถในการติดต่อสื่อสาร

เนื่องจากจำนวนโหนดสามารถมีได้มาก การสังเกตการณ์ข้อมูลดิบจำนวนมากทำให้เห็นสภาพการเปลี่ยนแปลงสิ่งแวดล้อมในระยะเวลาหนึ่งๆ ทำได้ยากการพัฒนาระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สายเพื่อใช้ในการเฝ้าติดตามเหตุการณ์ที่เกิดให้เกิดผลเสียต่อทรัพย์สิน และนำข้อมูลดิบมาสรุปในรูปแบบของกราฟและตารางเพื่อให้ผู้ใช้เข้าใจการเปลี่ยนแปลงสภาพแวดล้อมในช่วงเวลาหนึ่ง แล้วนำมาแสดงผลผ่านเว็บบราวเซอร์

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

ปริญญานิพนธ์ฉบับนี้ มุ่งหวังเพื่อการศึกษาการพัฒนาการทำงานของระบบเครือข่ายตรวจจับแบบไร้สาย (Wireless Sensor Network System) เพื่อให้ได้ระบบที่สามารถรับส่งข้อมูลระหว่างโหนดกับสถานีฐานและเครื่องแม่ข่ายได้ และพัฒนาระบบสังเกตการณ์เครือข่ายตรวจจับ

แบบไร้สาย เพื่อให้ได้ระบบเฝ้าติดตามการเปลี่ยนแปลงสภาพแวดล้อมภายในอาคาร และนำข้อมูลที่ได้มาแปลงให้อยู่ในรูปแบบของกราฟและตารางเพื่อให้ผู้ใช้สามารถใช้ข้อมูลได้ง่ายดายมากขึ้น

1.3 สมมติฐานของการศึกษา

1. การรับส่งข้อมูลระหว่างโหนดไปจนถึงสถานีฐาน สามารถเปลี่ยนแปลงเส้นทางได้ หากเส้นทางเดิมไม่สามารถรับส่งได้
2. เครื่องแม่ข่ายสามารถดึงข้อมูลไปแสดงผลได้
3. นำข้อมูลที่ได้ไปแสดงผลในรูปแบบของตารางและกราฟผ่านเว็บบราวเซอร์
4. การแจ้งเตือนเมื่อเกิดเหตุการณ์ที่ได้กำหนดไว้ผ่านเว็บบราวเซอร์

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในโครงงาน

การรับส่งข้อมูลภายในระบบเครือข่ายตรวจจับแบบไร้สาย โดยสามารถแยกออกเป็นสองส่วนใหญ่ด้วยกัน คือการรับส่งข้อมูลระหว่างโหนด (โมท) และการรับส่งข้อมูลระหว่างต้นทางกับปลายทาง (โหนดกับสถานีฐาน) โดยการรับส่งข้อมูลระหว่างโหนดนั้นอยู่ภายใต้มาตรฐาน IEEE 802.15.4 ซึ่งเป็นมาตรฐานสำหรับ Media Access Control (MAC) และ Physical Layer (PHY) ส่วนการรับส่งข้อมูลระหว่างต้นทางกับปลายทางจะอยู่ภายใต้ XMesh Routing Protocol โดยใช้อุปกรณ์ตรวจจับชุดพัฒนา Wireless Sensor Network Classroom Kit ข้อดีของชุดพัฒนานี้คือมีอุปกรณ์ตรวจจับสิ่งแวดล้อมมีให้เลือกใช้ได้หลากหลายชนิด ที่เลือกใช้อุปกรณ์ตรวจจับชนิด MDA100CB เนื่องจากอุปกรณ์นี้สามารถตรวจจับความสว่างของแสง และอุณหภูมิ ซึ่งเหมาะกับการเฝ้าติดตามสภาพแวดล้อมภายในอาคาร

1.5 ขอบเขตของโครงงาน

1. สามารถเก็บข้อมูลจากตัวตรวจจับแล้วสามารถเก็บข้อมูล และแสดงผลในเชิงสถิติในรูปแบบของกราฟ และตารางได้
2. มีระบบแจ้งเตือนในการเฝ้าติดตามเหตุการณ์ที่ก่อให้เกิดความเสียหายต่อระบบ

1.6 ขั้นตอนของการศึกษา

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 8 บท ด้วยกัน ดังนี้

บทที่ 1 บทนำ จะกล่าวถึงความเป็นมาของโครงงาน ความมุ่งหมายและวัตถุประสงค์ สมมติฐาน ทฤษฎีที่ใช้ ขอบเขตของโครงงาน และขั้นตอนการศึกษา

บทที่ 2 เครือข่ายตรวจจับแบบไร้สาย จะกล่าวถึงทฤษฎีพื้นฐาน องค์ประกอบ มาตรฐาน และการรับส่งข้อมูลของเครือข่ายตรวจจับไร้สาย

บทที่ 3 ระบบปฏิบัติการ TinyOS จะกล่าวถึงทฤษฎีพื้นฐาน องค์ประกอบ และกลไกการทำงานของระบบปฏิบัติการ TinyOS ซึ่งเป็นระบบปฏิบัติการสำหรับระบบเครือข่ายแบบฝังตัวที่ใช้ในชุดพัฒนา Wireless Sensor Network Classroom Kit

บทที่ 4 การพัฒนาโปรแกรมสำหรับตัวตรวจจับแบบไร้สาย จะกล่าวถึงการวิเคราะห์ออกแบบโปรแกรมฝังตัวสำหรับตัวตรวจจับ รวมถึงการออกแบบฐานข้อมูล

บทที่ 5 การศึกษาการทำงานของเครือข่ายตรวจจับแบบไร้สาย จะกล่าวถึงการศึกษาการทำงานของการรับส่งข้อมูลตั้งแต่โหนดต้นทางไปถึงอุปกรณ์เกตเวย์ โดยการดูผลลัพธ์ผ่านโปรแกรมที่ชุดพัฒนา Wireless Sensor Network Classroom Kit จัดมาให้ โดยการศึกษานั้นจะจำลองสถานการณ์ต่างๆ เพื่อดูว่าการรับส่งข้อมูลนั้นเป็นไปตามทฤษฎีและหลักการหรือไม่

บทที่ 6 การพัฒนาระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สาย จะกล่าวถึงการวิเคราะห์ ออกแบบโปรแกรมสำหรับระบบเฝ้าติดตามสภาพแวดล้อมภายในอาคาร และการออกแบบฐานข้อมูล

บทที่ 7 การทดลองการทำงานของระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สาย จะกล่าวถึงยูสเซอร์อินเตอร์เฟส และผลของการใช้งานแอพพลิเคชั่น

บทที่ 8 สรุปผลการวิจัย และข้อเสนอแนะ จะกล่าวถึงบทสรุปผลของการทดลองว่าเป็นไปตามทฤษฎีและหลักการ และข้อเสนอแนะสำหรับการพัฒนาในขั้นต่อไป

บทที่ 2
เครือข่ายตรวจจับแบบไร้สาย

เครือข่ายตรวจจับแบบไร้สาย (Wireless Sensor Network) เป็นเครือข่ายไร้สายที่ประกอบด้วย อุปกรณ์ที่ทำงานอย่างเป็นอิสระ โดยใช้ตัวตรวจจับ (Sensor) ทำงานร่วมกันในการเฝ้าสังเกตการณ์สภาพแวดล้อม เช่น อุณหภูมิ เสียง การสั่นสะเทือน ความกดดัน การเคลื่อนไหว หรือของเสีย ในสถานที่แตกต่างกัน แล้วส่งข้อมูลไปยังสถานีฐานซึ่งทำหน้าที่เป็นประตูทางออกไปยังผู้ใช้งาน ผ่านทางโหนดต่างๆ (หรือที่เรียกว่าโมท (Mote)) ในเครือข่าย โดยโหนดทำงานภายใต้สภาพแวดล้อมที่จำกัด อาทิ พลังงาน หน่วยความจำ แบนด์วิดธ์ (bandwidth) และความสามารถในการติดต่อสื่อสาร

สำหรับเนื้อหาในบทนี้ เริ่มต้นจะกล่าวถึงที่มาและภาพรวมของเครือข่ายตรวจจับแบบไร้สาย จากนั้นจะกล่าวต่อในส่วนของการนำรูปแบบเครือข่ายดังกล่าวไปใช้งานในปัจจุบัน จากนั้นจะกล่าวถึงรายละเอียดของเครือข่ายตรวจจับแบบไร้สาย โดยจะกล่าวถึงสถาปัตยกรรมและรายละเอียดในระดับต่างๆ ซึ่งจะแบ่งออกเป็นสองส่วนด้วยกัน คือ มาตรฐาน IEEE 802.15.4 ซึ่งเป็นมาตรฐานสำหรับเครือข่ายไร้สายส่วนบุคคล (Wireless Personal Area Networks (WPANs)) สำหรับการสื่อสารความเร็วต่ำ (Low-Rate) และ โปรโตคอล ZigBee ซึ่งเป็นการพัฒนาเพื่อสนับสนุนการรับส่งข้อมูลข้ามเครือข่าย โดยจะกล่าวรวมถึงกระบวนการค้นหาเส้นทาง (Routing) ชนิด Ad-hoc On Demand Distance Vector (AODV) ซึ่งเป็นกระบวนการค้นหาเส้นทางใน Network Layer ภายใน โปรโตคอล ZigBee และ โปรโตคอล XMesh ซึ่งเป็นโปรโตคอลที่ใช้รับส่งข้อมูลในการทำโครงงานนี้ซึ่งเน้นในเรื่องระบบประหยัดพลังงานสามารถใช้งานได้นาน และสามารถเปลี่ยนเส้นทางเมื่อเกิดความผิดปกติแบบอัตโนมัติ

2.1 ภาพรวมของเครือข่ายตรวจจับแบบไร้สาย

เครือข่ายตรวจจับแบบไร้สาย (Wireless Sensor Network) คือการใช้ตัวตรวจจับ (sensor) ขนาดเล็ก ในการตรวจวัดคุณสมบัติภายในสิ่งแวดล้อมที่เราสนใจต้องการเก็บข้อมูล และประมวลผลข้อมูลเหล่านั้น เพื่อสร้างองค์ความรู้ใหม่เกี่ยวกับสิ่งแวดล้อมรอบตัวเรา หรือตอบสนองกับการเปลี่ยนแปลงของสภาพแวดล้อมได้โดยอัตโนมัติ รวมถึงสามารถรับส่งข้อมูลได้โดยผ่านระบบเครือข่ายแบบไร้สายได้

2.2 ส่วนประกอบของเครือข่ายตรวจจับแบบไร้สาย

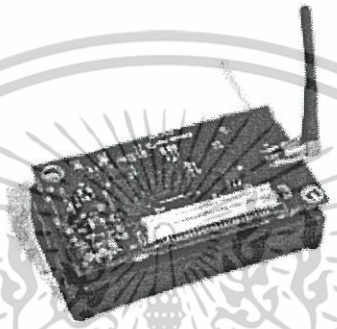
สำหรับเครือข่ายตรวจจับแบบไร้สาย โดยทั่วไปจะประกอบด้วยส่วนประกอบดังต่อไปนี้

2.2.1 ตัวตรวจจับแบบไร้สาย (Wireless Sensor Node)

สำหรับตัวตรวจจับแบบไร้สายจะมีส่วนประกอบด้วยทั่วไปดังนี้

2.2.1.1 หน่วยประมวลผลและแผงควบคุมการทำงานของการรับส่งคลื่นวิทยุ (Processor / Radio Boards)

ทำหน้าที่ประมวลผลข้อมูล และรับส่งข้อมูลภายในเครือข่าย โดยมีลักษณะดังแสดงในรูปที่ 2.1



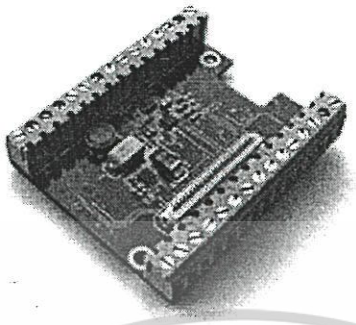
รูปที่ 2.1 หน่วยประมวลผลและแผงควบคุมการทำงานของการรับส่งคลื่นวิทยุ

2.2.1.2 แผงควบคุมการทำงานของตัวตรวจจับ (Sensor Boards)

มีหลากหลายรูปแบบ ขึ้นกับสิ่งที่จะวัด เช่น

- Accelerometer เป็นตัววัดความเร่ง
- Barometric Press / Temperature เป็นตัววัดความกดอากาศ
- GPS เป็นตัวระบุตำแหน่ง
- Magnetic Field เป็นตัววัดสนามแม่เหล็ก
- Microphone เป็นตัววัดระดับเสียง
- Photoresistor เป็นตัววัดปริมาณความเข้มแสง
- Relative Humidity / Temperature เป็นตัววัดความชื้น
- Thermistor เป็นตัววัดอุณหภูมิลมหายใจ
- Analog Inputs เป็นตัววัดสัญญาณอนาลอกขาเข้า

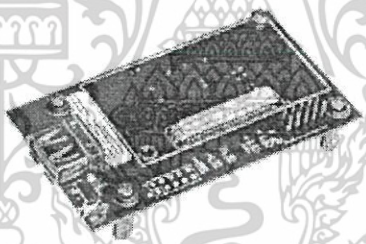
สำหรับลักษณะของตัวตรวจจับนั้น ดังแสดงในรูปที่ 2.2



รูปที่ 2.2 แผงควบคุมการทำงานของตัวตรวจจับ

2.2.1.3 แผงควบคุมการทำงานของเกตเวย์ (Gateway Boards)

ทำหน้าที่เป็นตัวเชื่อมต่อระหว่างอุปกรณ์เกตเวย์กับคอมพิวเตอร์หรือระบบเครือข่ายอื่น ดังแสดงในรูปที่ 2.3



รูปที่ 2.3 แผงควบคุมการทำงานของเกตเวย์

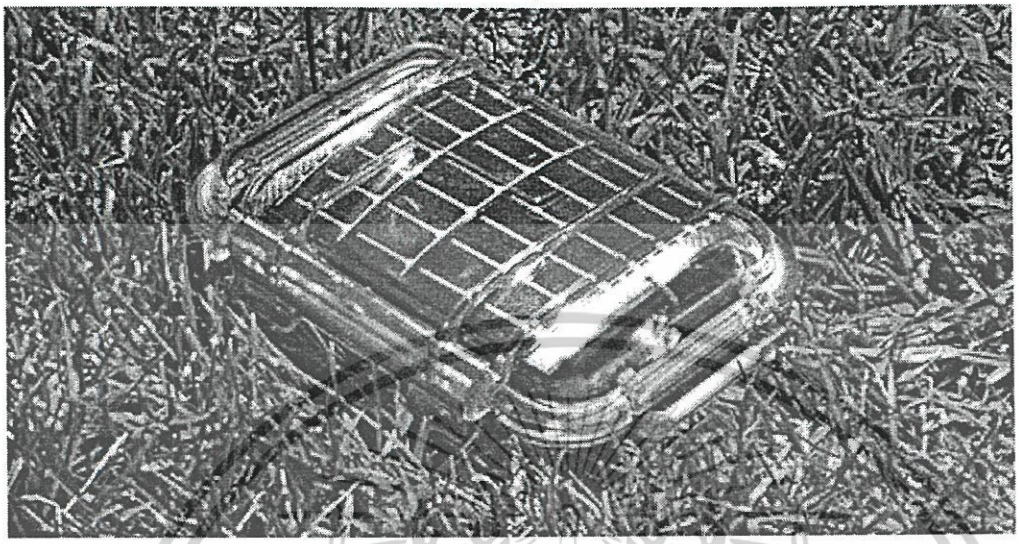
สำหรับประเภทของการเชื่อมต่อนั้นแบ่งออกเป็น 3 รูปแบบ ดังนี้

- พอร์ตอนุกรม (Serial Ports)
- พอร์ตยูเอสบี (Universal Serial Bus Ports)
- พอร์ต RJ-45 (Ethernet Ports)

2.2.1.4 แหล่งพลังงาน (Power Sources)

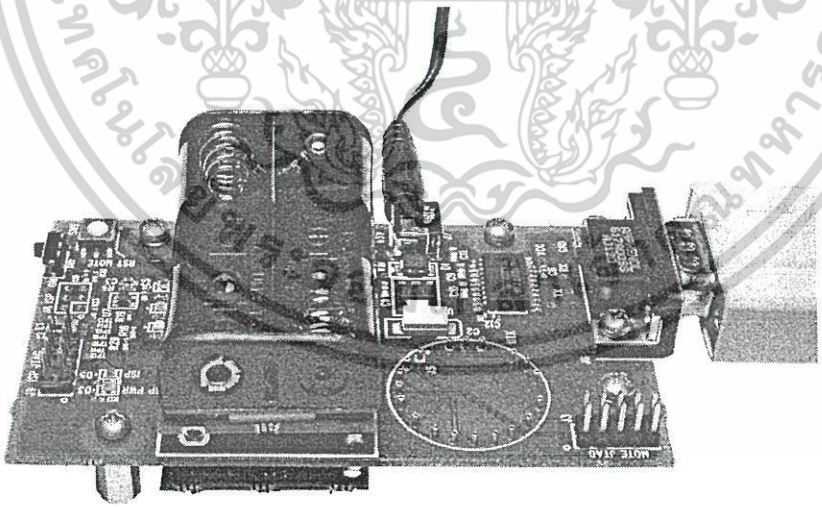
แหล่งพลังงานสำหรับตัวตรวจจับแบบไร้สาย จะถูกติดตั้งเข้ากับหน่วยประมวลผลและแผงควบคุมการทำงานของการรับส่งคลื่นวิทยุ ดังแสดงในรูปที่ 2.1

เมื่อทำการประกอบส่วนประกอบที่ 1), 2) และ 4) เข้าด้วยกันจะได้ตัวตรวจจับแบบไร้สาย ดังตัวอย่างในรูปที่ 2.4



รูปที่ 2.4 ตัวอย่างตัวตรวจจับแบบไร้สาย

เมื่อทำการประกอบส่วนประกอบที่ 1), 3) และ 4) เข้าด้วยกันจะได้อุปกรณ์เกตเวย์ ดังแสดงในรูปที่ 2.5



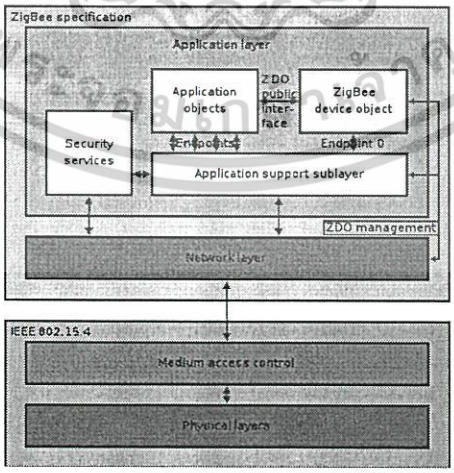
รูปที่ 2.5 ตัวอย่างอุปกรณ์เกตเวย์

2.3 การประยุกต์ใช้

สำหรับการนำรูปแบบเครือข่ายดังกล่าวไปใช้ในปัจจุบัน อาทิ การฝังตัวตรวจจับไว้ในรังนกหายากบางชนิด เพื่อตรวจจับการเปลี่ยนแปลงของอุณหภูมิที่มีผลต่อการย้ายถิ่นฐานของนกเหล่านั้น การติดตั้งตัวตรวจจับไว้ภายในอุปกรณ์ผสมสารเคมีขนาดใหญ่หรือท่อส่งสารเคมีในโรงงานอุตสาหกรรมเพื่อตรวจจับการรั่วซึมของสารเคมี การใช้ตัวตรวจจับตรวจวัดการสั่นไหวของอุปกรณ์หลายๆ อย่างที่ใช้สำหรับการสร้างชิปสำหรับคอมพิวเตอร์ เพื่อตรวจจับความผิดปกติของเครื่องมือเหล่านั้นเพื่อให้สามารถเข้าไปดูแลได้ก่อนที่จะเสียหาย เป็นต้น

2.4 สถาปัตยกรรมของเครือข่ายตรวจจับแบบไร้สาย

สำหรับเครือข่ายตรวจจับแบบไร้สาย จะประกอบไปด้วยมาตรฐานสองส่วนด้วยกัน ส่วนแรกคือ มาตรฐาน IEEE 802.15.4 ซึ่งเป็นมาตรฐานสำหรับเครือข่ายไร้สายส่วนบุคคล (Wireless Personal Area Networks (WPANs)) สำหรับการสื่อสารความเร็วต่ำ (Low-Rate) หากเปรียบเทียบกับ TCP/IP โปรโตคอลแล้วมาตรฐานดังกล่าวจะกล่าวถึงใน Physical Layer และ Data-Link Layer อีกส่วนหนึ่งจะเป็นโปรโตคอลที่สนับสนุนการทำงานใน Network Layer ขึ้นไปถึง Application Layer ซึ่งมาตรฐาน IEEE 802.15.4 ไม่ได้กล่าวถึงในส่วนนี้ จึงมีผู้พัฒนาของบริษัทเอกชนหรือองค์กรที่เป็นพันธมิตรกัน (Alliance) ในการพัฒนาขึ้นมา เช่น ZigBee โปรโตคอล ได้รับการพัฒนาขึ้นจาก ZigBee Alliance หรือ WirelessHART โปรโตคอล ได้รับการพัฒนาขึ้นจาก HART Communication Foundation เป็นต้น โปรโตคอลหนึ่งที่เป็นที่นิยมและได้รับการบรรจุในชุดพัฒนา Wireless Sensor Network Classroom Kit ที่ใช้ในโครงการนี้คือ ZigBee โปรโตคอล โดยโปรโตคอลดังกล่าว ได้อ้างอิงรูปแบบสถาปัตยกรรมของเครือข่ายตรวจจับแบบไร้สายไว้ ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 สถาปัตยกรรมโดยสรุปของเครือข่ายตรวจจับแบบไร้สาย

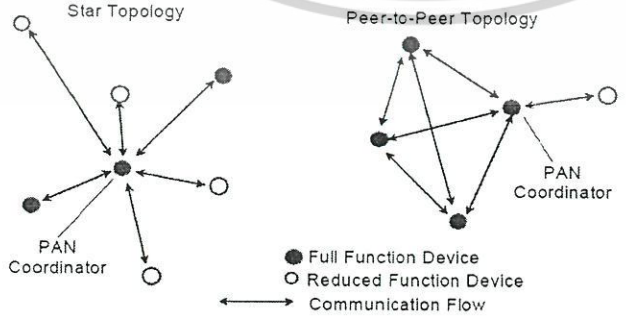
2.4.1 มาตรฐาน IEEE 802.15.4

LR-WPAN เป็นเครือข่ายไร้สายแบบส่วนบุคคลชนิดหนึ่งที่ความเร็วในการสื่อสารต่ำ พัฒนาขึ้นเพื่อให้ง่ายต่อการติดตั้ง มีการรับส่งมูลที่มีเสถียรภาพ ทำงานในระยะใกล้ ใช้พลังงานได้อย่างคุ้มค่า ดูแลจัดการได้ง่าย สำหรับลักษณะที่สำคัญของระบบเครือข่ายตรวจจับแบบไร้สายสามารถสรุปได้ดังนี้

- มีขนาดเล็ก เคลื่อนย้ายได้สะดวกและอิสระต่อโหนด (Node) อื่นในเครือข่าย
- มีพลังงานที่จ่ายให้อุปกรณ์จำกัด เนื่องจากส่วนใหญ่แหล่งจ่ายพลังงานจะเป็นแบตเตอรี่เพื่อให้เซ็นเซอร์สะดวกต่อการเคลื่อนย้าย
- สามารถทนต่อสภาพแวดล้อมต่างๆ รอบอุปกรณ์ได้
- สามารถจัดการต่อการหยุดการทำงานของโหนดต่างๆ ในเครือข่ายได้
- รูปแบบเครือข่าย (Topology) สามารถปรับเปลี่ยนได้อย่างอิสระ
- มีความเป็นลำดับชั้นของโหนด

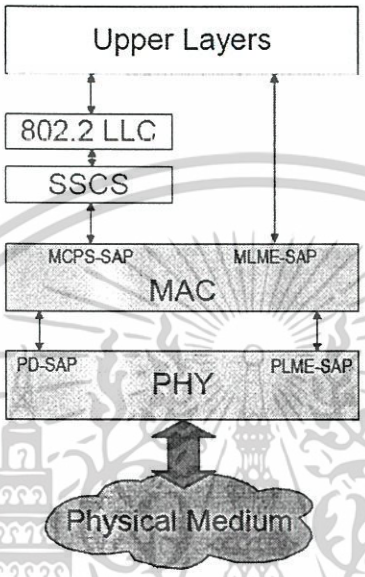
สำหรับชนิดของอุปกรณ์นั้นสามารถแบ่งออกได้เป็น 2 ชนิด คือ Full Function Device (FFD) เป็น อุปกรณ์ที่สามารถรับส่งข้อมูลจากเป็นได้ทั้งอุปกรณ์ในเครือข่ายธรรมดา หรือตัวประสานงาน (coordinator) หรือตัวประสานงานของเครือข่ายส่วนบุคคล (PAN coordinator) ซึ่งในสองกรณีหลังนั้นจะสามารถรับส่งข้อมูลระหว่างอุปกรณ์ประเภท FFD ด้วยกัน หรือกับ RFD ได้ และ Reduced Function Device (RFD) เหมาะแก่การเชื่อมต่อภายในเครือข่าย สามารถรับส่งได้กับอุปกรณ์ประเภท FFD เท่านั้น

สำหรับ รูปแบบเครือข่ายตามมาตรฐาน IEEE 802.15.4 แบ่งออกเป็น 2 รูปแบบ ได้แก่ แบบดาว (Star topology) และแบบโหนดต่อโหนด (Peer-to-peer topology) โดยมีตัวอย่างการเชื่อมต่อดังแสดงในรูปที่ 2.7 โดยในแต่ละเครือข่าย จะต้องมี FFD จำนวนหนึ่งตัวทำหน้าที่เป็นศูนย์กลางของเครือข่าย เรียกว่า ตัวประสานงานของเครือข่ายส่วนบุคคล (PAN coordinator) และ RFD จะเข้าร่วมเครือข่ายกับตัวประสานงานของเครือข่ายส่วนบุคคล ประจำเครือข่ายนั้นๆ



รูปที่ 2.7 ตัวอย่างรูปแบบเครือข่ายแบบดาวและแบบโหนดต่อโหนด

ดังที่ได้กล่าวไปแล้วตั้งแต่ต้นว่า หากเปรียบเทียบ TCP/IP โปรโตคอลแล้ว กับมาตรฐาน IEEE 802.15.4 นั้นได้กล่าวถึง ใน Physical Layer (PHY) และ Data-Link Layer (MAC) ดังแสดงในรูปที่ 2.8 เท่านั้น สำหรับระดับชั้นที่สูงกว่า (ในรูปที่ 2.3 จะหมายถึงตั้งแต่ SSCS เป็นต้นไป) จะอยู่นอกเหนือขอบเขตของมาตรฐาน IEEE 802.15.4



รูปที่ 2.8 สถาปัตยกรรมของ LR-WPAN

สำหรับรายละเอียดมาตรฐาน IEEE 802.15.4 จะอธิบายในรายละเอียดในหัวข้อต่อไปนี้

- การแบ่งส่วนของมาตรฐาน IEEE 802.15.4
- รูปแบบการโอนถ่ายข้อมูล
- โครงสร้างของเฟรม
- กระบวนการหลีกเลี่ยงการชนกันของข้อมูล

2.4.1.1 การแบ่งส่วนของมาตรฐาน IEEE 802.15.4

สำหรับรายละเอียดของ PHY และ MAC มีดังนี้

1. ระดับชั้น PHY

การทำงานของระดับชั้น PHY จะให้บริการสองรูปแบบ คือ การให้บริการข้อมูล (Data service) และการให้บริการในการดูแลจัดการ (Management service) โดยความรับผิดชอบในระดับชั้นนี้ ประกอบด้วย การเปิด/ปิดตัวรับส่งคลื่นวิทยุ การตรวจจับคลื่น (Energy Detection (ED)) การระบุคุณภาพของช่องสัญญาณ (Link Quality Indication (LQI)) การเลือกช่องสัญญาณ

(Channel selection) การประเมินช่องสัญญาณที่ดี (Clear Channel Assessment (CCA)) และ การรับส่งข้อมูลผ่านทางสื่อกลาง ซึ่งแบ่งออกได้เป็น 3 ช่วง ตามแต่ภูมิภาคดังนี้

- 868 - 868.6 MHz (อาทิ ทวีปยุโรป)
- 902 - 928 MHz (อาทิ ทวีปอเมริกาเหนือ)
- 2400 - 2483.5 MHz (ทั่วโลก)

2. ระดับชั้น MAC

เช่นเดียวกับการทำงานของระดับชั้น PHY ในระดับชั้น MAC จะให้บริการสองรูปแบบ คือ การให้บริการข้อมูล (MAC data service) และการให้บริการในการดูแลจัดการ (MAC management service) โดยเชื่อมต่อกับบริการ MAC sub-layer management entity (MLME) service access point (SAP) (MLME-SAP) โดยความรับผิดชอบในระดับชั้นนี้ ประกอบด้วย การจัดการเบคอนเฟรม การเข้าถึงช่องสัญญาณ การจัดการช่องเวลา (Guaranteed time slots (GTS) management) การตรวจสอบความถูกต้องของเฟรม (Frame validation) การนำส่งเฟรมตอบรับ (Acknowledged frame delivery) และอื่นๆ นอกจากนั้นระดับชั้น (sub-layer) ได้จัดเตรียมการพัฒนาแอพพลิเคชั่นที่มีกลไกด้านความปลอดภัยไว้ด้วย

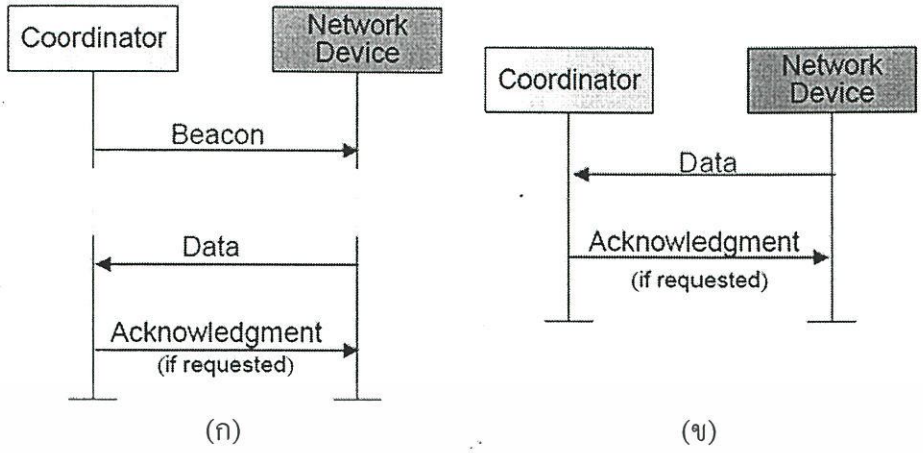
2.4.1.2 รูปแบบการโอนถ่ายข้อมูล

สำหรับรูปแบบการโอนถ่ายข้อมูลในเครือข่ายตรวจจับแบบไร้สาย มีอยู่ 3 รูปแบบดังนี้

- การส่งข้อมูลไปยังตัวประสานงาน
- การส่งข้อมูลจากตัวประสานงาน
- การโอนข้อมูลระหว่างโหนดต่อโหนด

2.4.1.2.1 การส่งข้อมูลไปยังตัวประสานงาน (Data transfer to a coordinator)

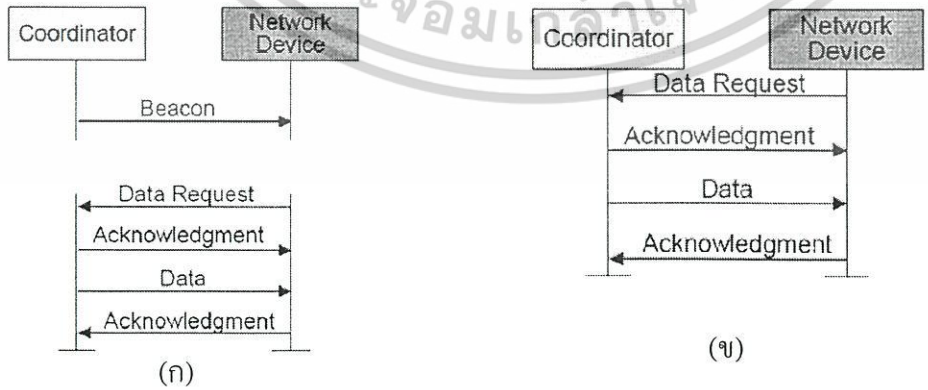
สำหรับเครือข่ายที่มีเบคอนเฟรม ผู้ที่ต้องการส่งข้อมูลจะต้องรับฟังก่อนว่ามีเบคอนเฟรมหรือไม่ หากมีก็จะทำการซิงโครนัสแล้วจึงส่งข้อมูล ภายใต้การดูแลของอัลกอริทึม Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) หากปลายทางได้รับจะตอบกลับมาทางเฟรมตอบรับ แต่ในกรณีไม่มีเบคอนเฟรม ผู้ที่ต้องการส่งข้อมูลจะส่งข้อมูลออกไปทันที โดยไม่ได้อยู่ภายใต้การดูแลของ CSMA/CA หากปลายทางได้รับจะตอบกลับมาทางเฟรมตอบรับ ดังแสดงในรูปที่ 2.9 (CSMA/CA จะกล่าวถึงในช่วงถัดไป)



รูปที่ 2.9 แสดงกระบวนการส่งข้อมูลไปยังตัวประสานงาน (ก) มีเบคอนเฟรม (ข) ไม่มี

2.4.1.2.2 การส่งข้อมูลจากตัวประสานงาน (Data transfer from a coordinator)

สำหรับเครือข่ายที่มีเบคอนเฟรม ตัวประสานงานจะส่งเบคอนเฟรมออกไป เมื่อผู้รับได้รับเบคอนเฟรมก็จะตอบกลับด้วยเฟรมร้องขอข้อมูล โดยอยู่ภายใต้การดูแลของอัลกอริทึม CSMA/CA หากตัวประสานงานได้รับจะส่งเฟรมตอบรับ จากนั้นผู้ประสานงานถึงส่งข้อมูลออกไป ภายใต้การดูแลของ CSMA/CA หากปลายทางได้รับจะตอบกลับมาทางเฟรมตอบรับ แต่ในกรณีไม่มีเบคอนเฟรม อุปกรณ์จะส่งเฟรมร้องขอข้อมูลออกไป โดยไม่ได้ดูภายใต้การดูแลของ CSMA/CA หากปลายทางได้รับจะตอบกลับมาทางเฟรมตอบรับ จากนั้นผู้ประสานงานถึงส่งข้อมูลออกไป โดยไม่ได้อยู่ภายใต้การดูแลของ CSMA/CA หากปลายทางได้รับจะตอบกลับมาทางเฟรมตอบรับ ดังแสดงในรูปที่ 2.10



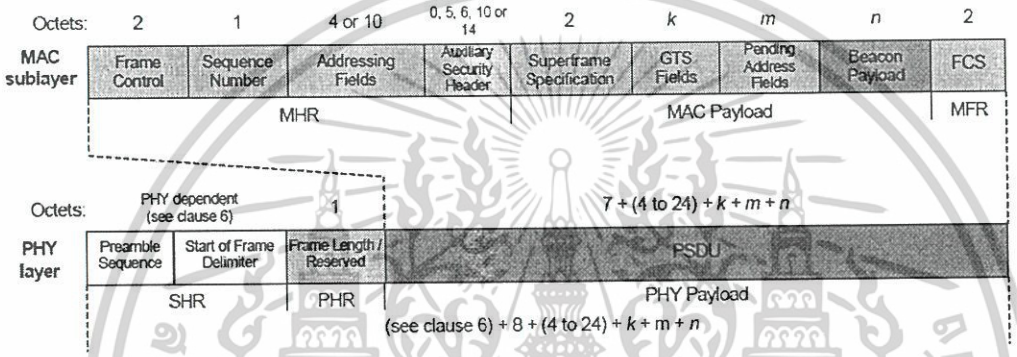
รูปที่ 2.10 แสดงกระบวนการส่งข้อมูลจากตัวประสานงาน (ก) มีเบคอนเฟรม (ข) ไม่มี

2.4.1.2.3. การโอนข้อมูลระหว่างโหนดต่อโหนด (Peer-to-peer data transfers)

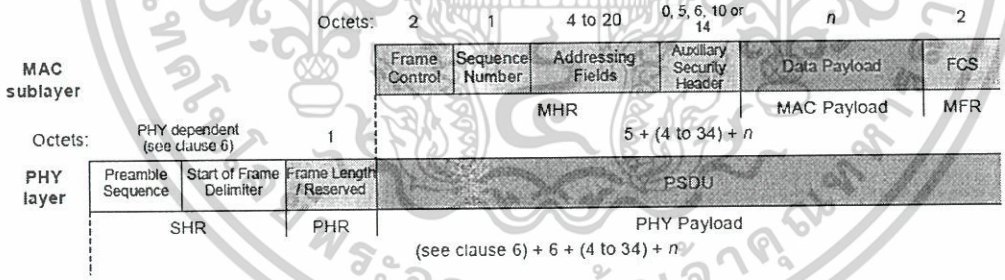
มีสองรูปแบบ คือ มีและไม่มี CSMA/CA สำหรับกรณีมี CSMA/CA จะอยู่นอกเหนือมาตรฐาน IEEE 802.15.4

2.4.1.3 โครงสร้างของเฟรม

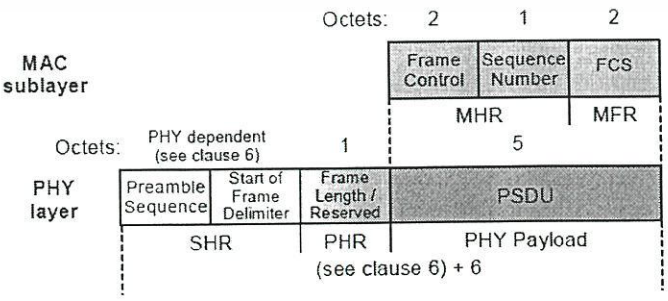
สำหรับโครงสร้างของเฟรมแต่ละรูปแบบแสดงดังรูปที่ 2.11 ถึง 2.14



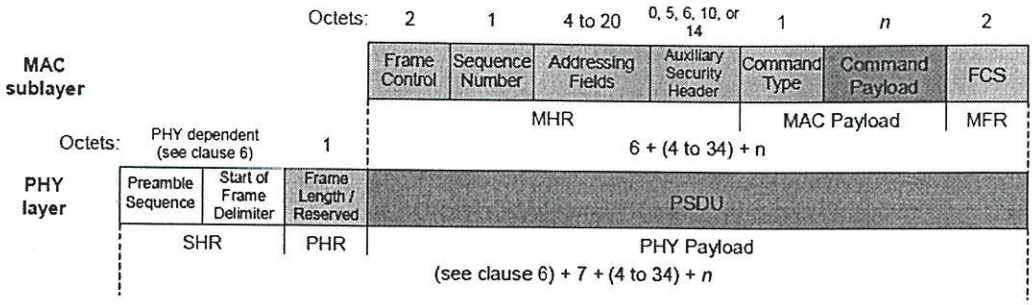
รูปที่ 2.11 โครงสร้างเบคอนเฟรมและ PHY แพ็คเก็ต



รูปที่ 2.12 โครงสร้างเฟรมข้อมูลและ PHY แพ็คเก็ต



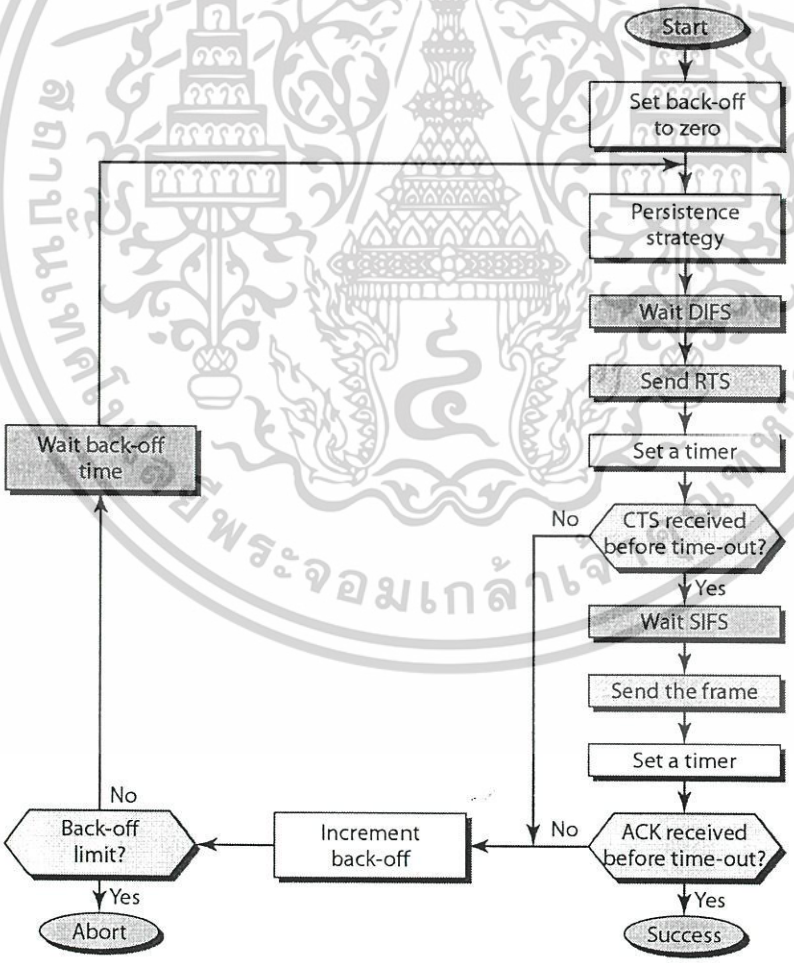
รูปที่ 2.13 โครงสร้างเฟรมตอบรับและ PHY แพ็คเก็ต



รูปที่ 2.14 โครงสร้างเฟรมคำสั่งและ PHY แพ็คเก็ต

2.4.1.4 กระบวนการหลีกเลี่ยงการชนกันของข้อมูล

เครือข่ายตรวจจับแบบไร้สาย จะใช้กระบวนการหลีกเลี่ยงการชนกันของข้อมูล CSMA/CA เช่นเดียวกับเครือข่ายไร้สายประเภทอื่น อาทิ เครือข่ายไร้สายท้องถิ่น (Wireless Local Area Networks (WLANs)) ดังแสดงในรูปที่ 2.15



รูปที่ 2.15 กระบวนการการทำงานของ CSMA/CA

สำหรับหลักการทำงานของกลไก Carrier Sense Multiple Access (CSMA) นั้น เมื่อสถานีหนึ่งต้องการเข้าใช้ช่องสัญญาณ ต้องตรวจสอบก่อนว่ามีสถานีอื่นทำการรับส่งสัญญาณข้อมูลอยู่หรือไม่และรอจนกว่าช่องสัญญาณจะว่าง เมื่อช่องสัญญาณว่าง แล้วสถานีที่ต้องการเข้าใช้ช่องสัญญาณจะต้องรอต่อไปอีกระยะหนึ่ง (Random Back-Off) ซึ่งแต่ละสถานีได้กำหนดระยะเวลาในการรอดังกล่าวไว้แล้วด้วยการสุ่มค่าหลังจากเสร็จการใช้ช่องสัญญาณครั้งก่อน สถานีที่สุ่มได้ค่าระยะเวลาในการรอน้อยกว่าก็จะมีสิทธิในการเข้าใช้ช่องสัญญาณก่อน แต่อย่างไรก็ตามในบางกรณีกลไกดังกล่าวอาจจะกำหนดให้สถานีมากกว่าหนึ่งสถานีส่งข้อมูลในเวลาพร้อมๆ กันซึ่งจะทำให้เกิดการชนกันของสัญญาณได้ ซึ่งหากเกิดการชนกันของสัญญาณขึ้นจะต้องมีการส่งสัญญาณข้อมูลเดิมซ้ำอีกครั้งด้วยกลไกที่กล่าวมาแล้วข้างต้น

การทำงานของกลไก CSMA/CA โดยหลักแล้วเป็นเช่นเดียวกับที่กล่าวไว้ในส่วนของ CSMA แต่จะมีรายละเอียดเพิ่มเติม เกี่ยวกับการหลีกเลี่ยงไม่ให้เกิดการชนกันของสัญญาณ และเทคนิคสำหรับการตรวจสอบว่าเกิดการชนของสัญญาณหรือไม่แบบเป็นนัย โดยสถานีผู้ส่งสัญญาณข้อมูลจะต้องรองรับ Acknowledgement จากสถานีที่ส่งข้อมูลไปให้ หากไม่ได้รับ Acknowledgement กลับมาภายในเวลาที่กำหนดจะถือว่าเกิดการชนของสัญญาณขึ้น และต้องทำการส่งข้อมูลเดิมซ้ำอีกต่อไป

2.4.2 ZigBee โปรโตคอล

ZigBee โปรโตคอล ได้รับการพัฒนาขึ้นเพื่อระบบเครือข่ายไร้สายส่วนบุคคล (WPAN) ซึ่งอยู่ภายใต้มาตรฐาน IEEE 802.15.4 โดยมาตรฐานนี้ใช้งานสำหรับการสื่อสารความเร็วต่ำ ใช้กำลังไฟฟ้าน้อย อุปกรณ์ราคาถูก และมีคุณสมบัติการจัดการตัวเองได้ เป็นเทคโนโลยีไร้สายที่ร่วมกันสื่อสารข้อมูลผ่านเซ็นเซอร์ขนาดเล็กจำนวนมาก ลักษณะของ ZigBee คือมีทางเข้าช่องสัญญาณโดยการใช้ CSMA/CA หลายช่องสัญญาณเพื่อหลีกเลี่ยงการชนกันของสัญญาณ มีรัศมีการทำงานครอบคลุมประมาณ 50 เมตร มีรูปแบบเครือข่ายแบบดาว (Star topology) แบบโหนดต่อโหนด และแบบโครงข่ายร่างแห (Mesh topology) ทั้งนี้แต่ละอุปกรณ์จะมีตำแหน่งที่อยู่ (Address) ที่มีความยาว 64 ไบต์ หรือ 16 บิต (รองรับได้ 64,000 อุปกรณ์)

ในหัวข้อมาตรฐาน IEEE 802.15.4 ที่ผ่านมา ได้ทำการแยกอุปกรณ์ออกเป็นสองประเภท คือ FFD และ RFD สำหรับ ZigBee โปรโตคอล

2.4.2.1 ประเภทของอุปกรณ์

2.4.2.1.1 ตัวประสานงาน (ZigBee Coordinators)

เป็นจุดที่ประสานเชื่อมต่อกัน ทำหน้าที่จัดเก็บข้อมูลในเครือข่าย

2.4.2.1.2 เราเตอร์ (ZigBee Routers)

ทำหน้าที่จัดการเส้นทางของข้อความที่ส่งผ่านภายในโครงข่าย ระหว่างคู่ของโหนดใดๆ

2.4.2.1.3 อุปกรณ์ปลายทาง (ZigBee End Devices)

เป็นโหนดที่อยู่ในส่วนของผู้ใช้งาน โดยสามารถเป็นได้ทั้งแบบ FFD และ RFD

2.4.2.2 องค์ประกอบของ ZigBee

2.4.2.2.1 Application layer

เป็นชั้นที่มีส่วนของปลายทาง (Endpoint) อยู่ เรียกว่า Application framework โดยมี ZigBee Device Object (ZDO) ทำหน้าที่จัดการในการเข้าถึงและใช้งาน Application layer

2.4.2.2.2 Application support sub-layer

ทำหน้าที่สร้างเฟรมของ Application layer และทำหน้าที่ในการรับส่งข้อมูลรวมถึงการจัดการด้านต่างๆ ที่เกี่ยวข้องกับ Application layer

2.4.2.2.3 Network layer

ทำหน้าที่ใช้ในการหาเส้นทางรับส่งข้อมูลจากต้นทางไปยังปลายทาง ที่อาจอยู่ภายในเครือข่ายเดียวกันหรือต่างเครือข่ายกัน

2.4.2.3 ขั้นตอนการทำงานของโปรโตคอล ZigBee

2.4.2.3.1 ขั้นตอนการทำงานของตัวประสานงาน

ตัวประสานงาน จะเริ่มต้นเครือข่าย โดยการตรวจสอบการใช้ช่องสัญญาณวิทยุภายในบริเวณรอบๆ ถ้ามีช่องสัญญาณที่ไม่ถูกใช้โดยตัวประสานงานตัวอื่น ก็สามารถเริ่มต้นเครือข่ายได้ หลังจากนั้นตัวประสานงานก็จะทำหน้าที่เป็นศูนย์กลางของเครือข่าย รองรับการเข้าร่วมเครือข่ายของอุปกรณ์ปลายทาง และรองรับการร้องขออื่นๆ ตามมาตรฐานด้วยเช่นกัน

2.4.2.3.2 ขั้นตอนการทำงานของอุปกรณ์ปลายทาง

อุปกรณ์ปลายทาง จะเริ่มต้นการทำงานโดยการร้องขอการเข้าร่วมเครือข่ายไปยังตัวประสานงานประจำเครือข่ายนั้นๆ โดยการตรวจสอบผ่านช่องสัญญาณต่างๆ ว่าตัวประสานงานใช้ช่องสัญญาณใดอยู่เมื่อเข้าร่วม

เครือข่ายแล้ว อุปกรณีปลายทาง จึงสามารถทำการร้องขอคำสั่งอื่นๆ ผ่านทางตัวประสานงานได้

2.4.2.4 การประยุกต์ใช้งาน ZigBee

2.4.2.4.1 ข้อมูลตามช่วงเวลา (Periodic)

ข้อมูลเป็นช่วงเวลา โดยสามารถควบคุมอัตราการส่ง และการตรวจจับได้ ใช้สำหรับ ตัวตรวจจับ และมิเตอร์

2.4.2.4.2 ข้อมูลตามการทำงาน (Intermittent)

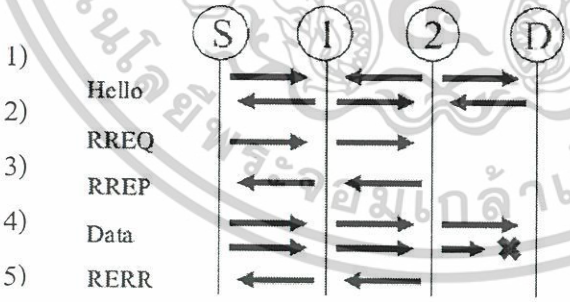
เป็นลักษณะที่มีการส่งผ่านข้อมูลเมื่อมีการใช้งาน เช่น สวิตซ์ไฟ

2.4.2.4.3 ข้อมูลซ้ำที่มีความล่าช้าต่ำ (Repetitive low latency)

ใช้ในงานที่ต้องการความล่าช้าต่ำ โดยการสื่อสารจะใช้วิธีจัดสรรช่องเวลา และสามารถใช้กลไกแบบ GTS เพื่อรับประกันคุณภาพของการบริการ นำไปใช้ในงาน เช่น เมาส์ไร้สาย

2.5 Ad-hoc On Demand Distance Vector (AODV)

สำหรับกระบวนการค้นหาเส้นทางใน ZigBee โปรโตคอลนั้นได้ใช้อัลกอริทึม AODV โดยกระบวนการดังกล่าวมีลักษณะตามความต้องการที่โต้ตอบได้ (On-demand driven reactive) สำหรับจุดมุ่งหมายที่สำคัญของอัลกอริทึมนี้ต้องการส่งแพ็คเกตที่ใช้ค้นหาเส้นทาง (discovery packets) เท่าที่จำเป็น



รูปที่ 2.16 การรับส่งแพ็คเกตในโปรโตคอล AODV

เมื่อโหนดต้นทางต้องการสื่อสารกับอีกโหนดหนึ่ง ซึ่งไม่มีตารางเราติ้งที่เก็บผลการค้นหาเส้นทาง (Routing information tables) อยู่ กระบวนการค้นหาเส้นทางจะเริ่มขึ้น (ทุกโหนดจะดูแลตัวนับอยู่สองตัวแปร คือ เลขลำดับ (Sequence number) และหมายเลขการกระจายข้อมูล (Broadcast ID)) โดยโหนดต้นทางจะเริ่มต้นค้นหาเส้นทางโดยใช้กระจายการร้องขอ (Route request (RREQ))

ไปให้เพื่อนบ้าน (ดังแพ็คเกตลำดับที่ 2 ที่แสดงในรูปที่ 2.16) โดยข้อมูลที่อยู่ในแพ็คเกตนั้นประกอบด้วย หมายเลขที่อยู่ต้นทาง (Source address) เลขลำดับของแพ็คเกตต้นทาง (Source sequence number) หมายเลขการกระจายข้อมูล (Broadcast ID) หมายเลขที่อยู่ปลายทาง (Destination address) เลขลำดับของแพ็คเกตปลายทาง (Destination sequence number) และสุดท้ายคือจำนวนโหนดที่ผ่านไป (Hop count) โดยทั้งเลขลำดับของแพ็คเกตต้นทางและหมายเลขการกระจายข้อมูลของทั้งสองโหนดใดๆ จะมีค่าเป็นหนึ่งเดียว

ถ้าโหนดระหว่างทางได้รับแพ็คเกต RREQ อื่นที่มีเลขลำดับของแพ็คเกตต้นทางและหมายเลขการกระจายข้อมูลซ้ำกันแล้ว แพ็คเกตนั้นจะถูกทิ้งไป กรณีอื่นนอกจากนั้น โหนดนั้นจะทำการเพิ่มค่าจำนวนโหนดที่ผ่านไปและส่งแบบกระจายไปยังโหนดอื่นต่อ และแต่จะโหนดก็จะบันทึกหมายเลขที่อยู่ต้นทาง หมายเลขการกระจายข้อมูล หมายเลขที่อยู่ปลายทาง และเวลาหมดอายุเพื่อใช้ในการย้อนเส้นทาง และเลขลำดับของแพ็คเกตต้นทางลงในตารางเราติ้ง

ขณะที่ RREQ ได้เดินทางจากต้นทางไปปลายทาง จะทำการบันทึกเส้นทางที่ผ่านมาอัตโนมัติสำหรับทุกๆ โหนด เพื่อใช้เป็นเส้นทางย้อนกลับไปยังโหนดต้นทางได้ เมื่อบันทึกเส้นทางย้อนกลับแล้ว โหนดจะทำการบันทึกค่าตำแหน่งที่อยู่ของโหนดเพื่อนบ้านจากที่มันได้รับ RREQ ครั้งแรก โดยเวลาหมดอายุนั้นจะมากพอที่ RREQ เดินทางไปทั่วทั้งเครือข่ายและสร้าง Route reply packet (RREP) กลับไปยังโหนดต้นทาง

เมื่อ RREQ มาถึงแต่ละโหนด โหนดดังกล่าวจะตรวจสอบก่อนว่าโหนดดังกล่าวเป็นโหนดปลายทางของแพ็คเกตนี้หรือไม่ หากไม่ใช่แต่มีเส้นทางที่บันทึกไว้ว่าไปถึงโหนดปลายทางนั้นได้ มันจะทำการเปรียบเทียบเลขลำดับของแพ็คเกตปลายทางที่ตนมีกับใน RREQ ถ้าเลขลำดับของแพ็คเกตปลายทางใน RREQ มากกว่า โหนดดังกล่าวจะทำการส่งแบบกระจายต่อโดยไม่ได้สนใจเส้นทางที่บันทึกไว้

แต่หากเลขลำดับของแพ็คเกตปลายทางที่โหนดเก็บไว้มากกว่าหรือเท่ากับใน RREQ แต่น้อยกว่าค่าจำนวนโหนดที่ผ่านไป จะให้สร้าง RREP ส่งกลับไป (ดังแพ็คเกตลำดับที่ 3 ที่แสดงในรูปที่ 2.16) โดย RREP นั้นมีข้อมูล หมายเลขที่อยู่ต้นทาง เลขลำดับของแพ็คเกตปลายทาง หมายเลขที่อยู่ปลายทาง จำนวนโหนดที่ผ่านไป และช่วงอายุ ขณะที่ RREP เดินทางกลับไปยังโหนดต้นทางจะปรับปรุง (update) ข้อมูลเวลาสำหรับเส้นทางที่บันทึกอยู่ในโหนดนั้น และบันทึกเลขลำดับของแพ็คเกตปลายทางสำหรับปลายทางของการร้องขอนั้น

โหนดต้นทางสามารถเริ่มการส่งข้อมูลได้หลังจากที่ได้รับ RREP (ดังแพ็คเกตลำดับที่ 4 ที่แสดงในรูปที่ 2.16) และสามารถปรับปรุงในภายหลังได้หากเรียนรู้เส้นทางได้ดีกว่า

ท้ายที่สุดในแต่ละตารางเราติ้ง จะประกอบด้วย หมายเลขที่อยู่ปลายทาง โหนดถัดไป จำนวนโหนด (Metric) เลขลำดับของแพ็คเกตปลายทาง โหนดเพื่อนบ้านที่ทำงานอยู่ เวลาที่จะหมดอายุสำหรับข้อมูลชุดนี้

2.6 โปรโตคอล XMesh

2.6.1 องค์ประกอบด้านการใช้พลังงานโปรโตคอล XMesh

แบ่งออกเป็น 3 ชนิด

2.6.1.1 XMesh-HP

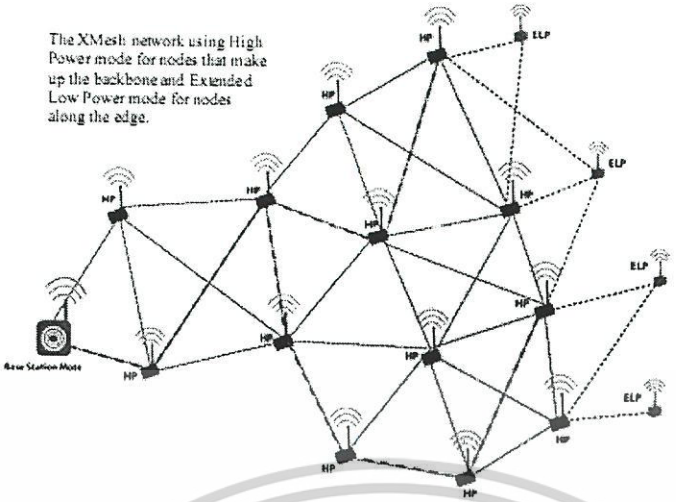
ในโหมดนี้จะใช้พลังงานประมาณ 15-30 mA (มิลลิแอมแปร์) หากกำหนดค่าเป็น XMesh-HP แล้วโหนดจะสามารถรับส่งข้อมูลได้ตอดลเวลา โดยเมสเซส Route Update และ Health จะส่งข้อมูลด้วยความเร็วที่สูง เพื่อลดเวลาในการสร้างรูปเครือข่ายแบบเมสหรือการเข้าร่วมเครือข่ายที่มีอยู่

2.6.1.2 XMesh-LP

ในโหมดนี้จะใช้พลังงานน้อยกว่า XMesh-HP แต่ยังคงคุณสมบัติของการซิงโครนัสเวลาในแต่ละโหนดภายในเครือข่ายไว้ โดยที่แต่ละโหนดจะซิงโครนัสภายในเวลา ±1 msec (มิลลิวินาที) โหนดจะทำงานซิงโครนัสเวลาเป็นจำนวน 8 ครั้ง ใน 1 วินาทีเพื่อตรวจดูสัญญาณวิทยุในบริเวณของโหนดดังกล่าว ในการทำงานดังกล่าวจะใช้พลังงาน 80 μA และค่าเฉลี่ยของการส่งข้อมูลใน 3 นาทีในการส่งข้อมูล 50 โหนด จะใช้พลังงาน 220 μA

2.6.1.3 XMesh-ELP

เป็นโหมดที่ทำหน้าที่ในการส่งข้อมูลเพียงอย่างเดียว จึงไม่ทำการซิงโคไนส์เวลา โหมดนี้เป็นโหมดที่ใช้พลังงานต่ำเพราะสามารถหลับได้เป็นเวลานาน และจะตื่นขึ้นมาเพื่อส่งข้อความ โดยที่โหนดนี้ต้องจำเป็นต้องมีโหนดผู้ดูแลเพื่อใช้ในการส่งข้อมูลและไม่ต้องการข้อความตอบกลับ และเมื่อต้องการหาผู้ดูแลใหม่ก็ดูจากค่าความแรงของสัญญาณความถี่



รูปที่ 2.17 แสดง XMesh-ELP

ตารางที่ 2.1 สรุปประสิทธิภาพการทำงานของโปรโตคอล XMesh

Parameter	XMesh-HP	XMesh-LP	XMesh-ELP
Route Update Interval	36 sec.	360 sec.	36 sec if built using HP 360 sec if built using LP
Data Message Rate	10 sec. typ.	180 sec.. typ.	N/A
Mesh formation time	2-3 times Route Update Interval for mesh with average of 2.5 hops		
Average current usage	20-30 mA	<250 μA [1] < 400 μA [2]	50 μA

[1] MICA2-LP using time synchronized mesh.

[2] MICAz-LP using asynchronous mesh

2.6.2 การสร้างเครือข่าย Multi-hop Mesh Network

มี 2 รูปแบบดังนี้

2.6.2.1 การประเมินค่าลิงค์

หากในตารางเราติ้งมีข้อมูลของแต่ละโหนดเพื่อนบ้านรวมกันมากกว่า 16 โหนดขึ้นไป XMesh จะทำการตัดลิงค์ที่มีคุณภาพต่ำที่สุดออก โดยใช้อัลกอริทึมแบบ EWMA (Exponentially Weight Moving Average) (ในกรณีของสถานีฐานจะสามารถเก็บข้อมูลของโหนดเพื่อนบ้านลงในตารางเราติ้งได้สูงสุด 40 โหนด)

2.6.2.1.1 การเลือกโหนดผู้ดูแล

จะเลือกโหนดเพียงโหนดเดียวจากเพื่อนบ้านที่มีค่าในการใช้พลังงานต่ำที่สุด โดยดูจากปัจจัยดังนี้

- โหนดเพื่อนบ้านนั้นจะต้องเข้าเป็นสมาชิกใน XMesh
- โหนดนั้นต้องมีข้อมูลในการอัตเดตเราติ้งของตัวเอง (Route Update Intervals) ไม่น้อยกว่า 3 ครั้ง
- ต้องไม่อยู่ในโหมด ELP (Extended Low Power)

ข้อมูลในแมสเซส Route Update ประกอบด้วย

- Parent ID ถ้าโหนดยังไม่เข้าร่วมเป็นสมาชิกใน mesh ฟิลด์นี้จะมีค่าเป็น 0xFFFF
- Cost เป็นค่าที่บอกให้เพื่อนบ้านทราบว่าจะต้องใช้ค่าเท่าไหร่ในการส่งข้อมูลไปหาสถานีฐาน
- Hop Count จำนวนโหนดที่ต้องใช้ในการส่งข้อมูลไปถึงสถานีฐาน
- รายชื่อของลิงค์ที่ผ่านการประเมิน

แมสเซส Route Update จะถูกบรอดแคสทุกๆ RUI โดยค่าเริ่มต้นของโหมด High power จะอยู่ที่ 36 วินาที และ Low power จะอยู่ที่ 360 วินาที (ตาราง 2.1)

RE = Receive Estimate หมายถึงคุณภาพการรับการสื่อสารในแต่ละเพื่อนบ้าน เป็นค่าที่เกิดจากการคำนวณโดยใช้อัลกอริทึม EWMA เพื่อหาร้อยละของการรับแพ็คเกจ โดยเป็นไปตามสมการดังนี้

$$New\_Estimate = 255 * received / (received + missed) \quad (2.1)$$

$$RE = (1 - alpha) * RE + alpha * New\_Estimate \quad (2.2)$$

โดยที่ alpha คือองค์ประกอบหนึ่งใน EWMA โดยค่าจะอยู่ในช่วง 0-1

SE = Send Estimate หมายถึงคุณภาพการส่งการสื่อสารในแต่ละเพื่อนบ้านได้รับมากจากการส่งบรอดแคสในแมสเซส Route Update และทำการส่งแพ็คเก็ตไปพร้อมกับ RE

SE และ RE จะมีค่าในช่วง 0-255 โดยที่ค่า 255 เป็นค่าที่มีคุณภาพที่ดีที่สุด

LC = Link Cost เป็นค่าที่เกิดจากการคำนวณค่า SE และ RE ในแต่ละเพื่อนบ้าน โดยมีสมการดังนี้

$$LC = (1<<18) / (SE * RE) \quad (2.3)$$

Link Cost จะดีที่สุดก็ต่อเมื่อ SE และ RE เท่ากับ 255 จะได้ LC เท่ากับ 4 ในแต่ละ hop

NC: Neighbor's cost เป็นค่า cost ที่ได้มาจากแมสเซส Route Update จากโหนดเพื่อนบ้าน

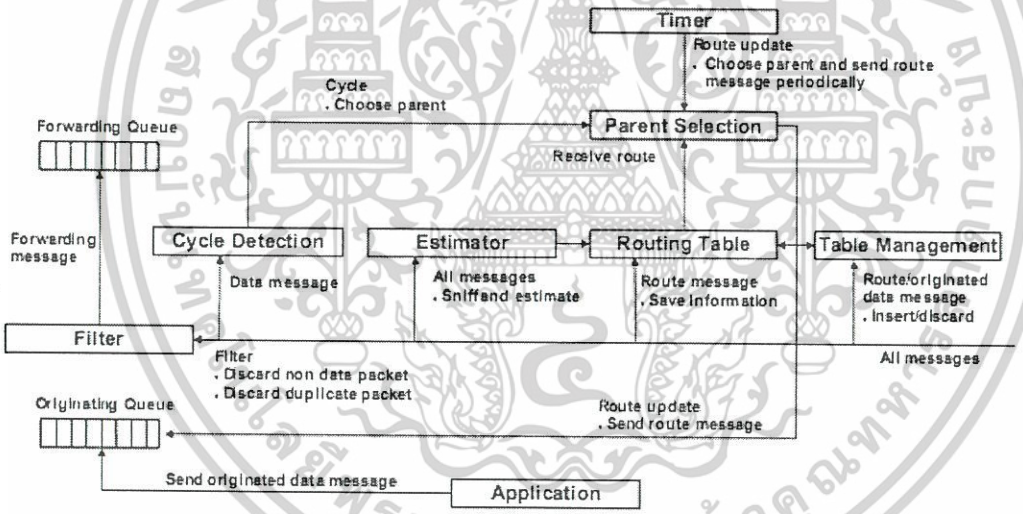
OC = ค่า Cost รวมที่จะส่งข้อความไปหาเกตเวย์มีการคำนวณตามสมการดังนี้

$$OC = LC + NC \tag{2.4}$$

โหนดจะเลือกเพื่อนบ้านที่มีค่า OC น้อยที่สุดมาเป็นผู้ปกครอง

เกตเวย์มีค่า cost เท่ากับ 0 ในแมสเซส Route Update

XMesh จะทำการเลือกผู้ปกครองทุกๆ 8 RUI เพราะโหนดต้องการข้อมูลในการตัดสินใจ



รูปที่ 2.18 แสดงอัลกอริทึมของ XMesh Routing

2.6.3 กระบวนการเร้าติ้งเส้นทางโดยใช้ XMesh Routing Protocol

สำหรับกระบวนการเราติ้งนั้นแบ่งออกเป็น 2 รูปแบบ แบบแรกคือการอัพเดตข้อมูลเส้นทาง เริ่มต้นจะกำหนด **Timer** ไว้ เมื่อถึงเวลาจะทำการเลือกโหนดผู้ดูแลที่ **Parent Selection** เมื่อเลือกได้ก็จะส่งข้อความไปยัง **Originate Queue** เพื่อรอส่งออกไป เช่นเดียวกับ **Application** ที่เมื่อจะส่งข้อมูลไปยังคิวนี้เช่นกันเพื่อรอส่งออกไป รูปแบบที่สองคือการประมวลผลข้อความที่รับเข้ามา เริ่มต้นข้อความจะเข้ามายัง **Table**

Management เพื่อดูว่าเป็นข้อความประเภทอัพเดตเส้นทางหรือเป็นข้อความข้อมูลที่ส่งมาจากโหนดอื่น หากเป็นประเภทอัพเดตเส้นทางจะบันทึกลงใน **Routing Table** จากนั้นข้อความทุกประเภทจะเข้าไปยังส่วน **Estimator** เพื่อทำการคำนวณคุณภาพของเส้นทางกับโหนดผู้ดูแล นอกจากนั้นยังต้องตรวจสอบว่าเกิดลูปในเส้นทางที่ส่งข้อมูลหรือไม่ หากข้อความที่เข้ามาไม่ตกอยู่ในค่ายประเภทที่กล่าวมาจะถูก **Filter** เพื่อละทิ้งข้อความที่ไม่ใช่ข้อมูลหรือเป็นข้อความซ้ำ แล้วส่งต่อไปยัง **Forwarding Queue**

2.6.4 ข้อความในการส่งและรับของโปรโตคอล XMesh

2.6.4.1 TinyOS Multihop message

ข้อมูลเพิ่มเติม routing ใน TinyOS ข้อมูลมีขนาดใหญ่ที่สุดได้ 55 ไบต์

ตารางที่ 2.2 แสดงโครงสร้างข้อความในโปรโตคอล XMesh

Component	Description
Header	Composed of standard TOS header and XMesh multihop header
Payload	User data
CRC	CRC check (Refer to Appendix E)

ตารางที่ 2.3 แสดงโครงสร้างแพ็กเกจในโปรโตคอล XMesh

Type	Name	Description	Component
uint16_t	addr	Destination address (next hop)	TinyOS Header (MICA2). The MICAz header has 5 additional bytes.
uint8_t	type	AM type: defines type of message	
uint8_t	group	AM group	
uint8_t	length	Remaining bytes in message, N/I CRC	
uint16_t	sourceaddr	Address of mote that sent the message	XMesh multihop header
uint16_t	originaddr	Address of mote that originated the message	
uint16_t	seqno	Message sequence number.	
uint8_t	socket	Application ID	
uint8_t	data	Payload	Payload size is set with the TOSH_DATA_LENGTH define. Default size is 29 bytes
uint16_t	crc	CRC check (Refer to Appendix E)	CRC

2.6.4.2 XMesh Message API

การส่งข้อความใน XMesh มีด้วยกัน 2 รูปแบบ

2.6.4.2.1 Upstream คือการส่งข้อความจากโหนดไปหาสถานีฐาน

2.6.4.2.2 Downstream คือจากสถานีฐานไปโหนด โดยการจะ Downstream ได้นั้นจำเป็นต้องได้รับข้อความจาก Upstream อย่างน้อย 1 ครั้ง

Quality of Services (QoS) แบ่งได้เป็น 2 แบบ ดังนี้

- Link level acknowledgement เน้นในเรื่องของการใช้พลังงานให้ต่ำที่สุดแต่ไม่รับประกันการส่งข้อมูลสำเร็จทุกครั้ง
- End-to-End acknowledgement ถ้ามีการส่งซ้ำจะมีการตัดสินใจที่จะเลือกส่งไปยังโหนดที่มีค่าสัญญาณสูง

2.6.4.3 XMeshBase Packet Structure

จะมีฟิลด์แรกและฟิลด์สุดท้ายที่ใช้ในการเปิดปิดแพ็กเกจ ฟิลด์นั่นชื่อ Ssynch ซึ่งจะมีค่า 0x7E(01111111) ฟิลด์ต่อมาคือ Packet Type จะบอกว่าต้องการการตอบกลับหรือไม่ต้องการตอบกลับ ฟิลด์ต่อไปคือ Tiny OS message Header ประกอบด้วย addr คือ สถานีปลายทาง AM type คือ ชนิดของข้อมูล (Rte,DatUP,Hlth,AckDwn) AM group ชื่อกลุ่ม length ความยาวของข้อมูล sourceaddr สถานีต้นทาง Originaddr สถานีจุดเริ่มต้น (ในกรณีที่มีการส่งผ่านข้อมูลสถานีต้นทางจะเปลี่ยนเรื่อยตามโหนดผู้ดูแล) seqno ลำดับของข้อความ socket บอกชนิดของข้อมูล (ข้อมูลจากเซนเซอร์,ข้อมูลบอกสุขภาพของโหนด) payload ข้อมูล CRC มีการทำ hash function เพื่อเช็คความถูกต้อง

ตารางที่ 2.4 แสดงโครงสร้างของข้อมูล

Sync 0x7E	Type 0x42	Data TOS_Msg Header	Data Payload	CRC 0xAFE2	Sync 0x7E

Byte #	Field	Description	
0	Ssynch byte	Always 0x7E	
1	Packet Type	Type	Description
		P_PACKET_NO_ACK (0x42)	- User packet with no ACK required.
		P_PACKET_ACK (0x41)	– User packet. ACK required. Includes a prefix byte. Receiver must send a P_ACK response with prefix byte as contents.
		P_ACK (0x40)	– The ACK response to a P_PACKET_ACK packet. Includes the prefix byte as its contents.
		P_UNKNOWN (0xFF)	– An unknown packet type.
2...n-1	Payload Data	In most cases will be a TinyOS Message of varying length, which is described below.	
n	SYNC_BYTE	Always 0x7E	

บทที่ 3
ระบบปฏิบัติการ TinyOS

ในปัจจุบันระบบปฏิบัติการสำหรับตัวตรวจจับ ระบบปฏิบัติการ TinyOS เป็นระบบปฏิบัติการที่ได้รับการยอมรับจากผู้ผลิตทั่วไป เนื่องจากเป็นระบบปฏิบัติการประเภทโอเพนซอร์ส (Open source) จึงสามารถนำไปปรับแต่งให้เหมาะสมกับแต่ละอุปกรณ์ของแต่ละบริษัทผู้ผลิตได้ นอกจากนั้นยังเป็นระบบปฏิบัติการที่มีลักษณะเชิงคอมโพเนนท์ (Component-based operating system) ทำงานภายใต้สภาพแวดล้อมที่ขับเคลื่อนด้วยเหตุการณ์ (Event-based operating environment) ซึ่งได้รับการออกแบบมาสำหรับระบบเครือข่ายแบบฝังตัว (Networked embedded systems) โดยเฉพาะ เนื่องด้วยข้อจำกัดของตัวตรวจจับที่มีทรัพยากร อาทิ หน่วยความจำ หน่วยประมวลผล ที่จำกัดกว่าอุปกรณ์เครือข่ายอื่นๆ ทั่วไป และลักษณะการทำงานที่ใช้เป็นตัวรับรู้เหตุการณ์จากเซ็นเซอร์ตลอดเวลา นั่นคือเมื่อมีเหตุการณ์เกิดขึ้นก็จะทำให้เกิดการทำงานอย่างหนึ่งอย่างใดของตัวตรวจจับนั่นเอง

สำหรับเนื้อหาในบทนี้จะอ้างอิงถึงระบบปฏิบัติการ TinyOS รุ่น 2.0 ขึ้นไป โดยเริ่มต้นจะกล่าวถึงภาพรวมของระบบปฏิบัติการ TinyOS ว่ามีลักษณะ การทำงานที่เหมาะสมกับระบบเครือข่ายตรวจจับแบบไร้สายอย่างไร จากนั้นจะกล่าวถึงแพลตฟอร์มและการจัดโครงสร้างของอุปกรณ์ (Platforms and hardware abstraction) หน่วยจัดการประมวลผล (Scheduler) การเริ่มต้นการทำงานของระบบปฏิบัติการ (Booting and initialization) การบริการในลักษณะเสมือน (Virtualization) ตัวจับเวลา (Timer) การติดต่อสื่อสาร (Communication) การตรวจจับ (Sensing) การตัดสินใจในการจัดสรรทรัพยากร (Arbitration) การจัดการพลังงาน (Power management) และโปรโตคอลของเครือข่าย (Network protocols)

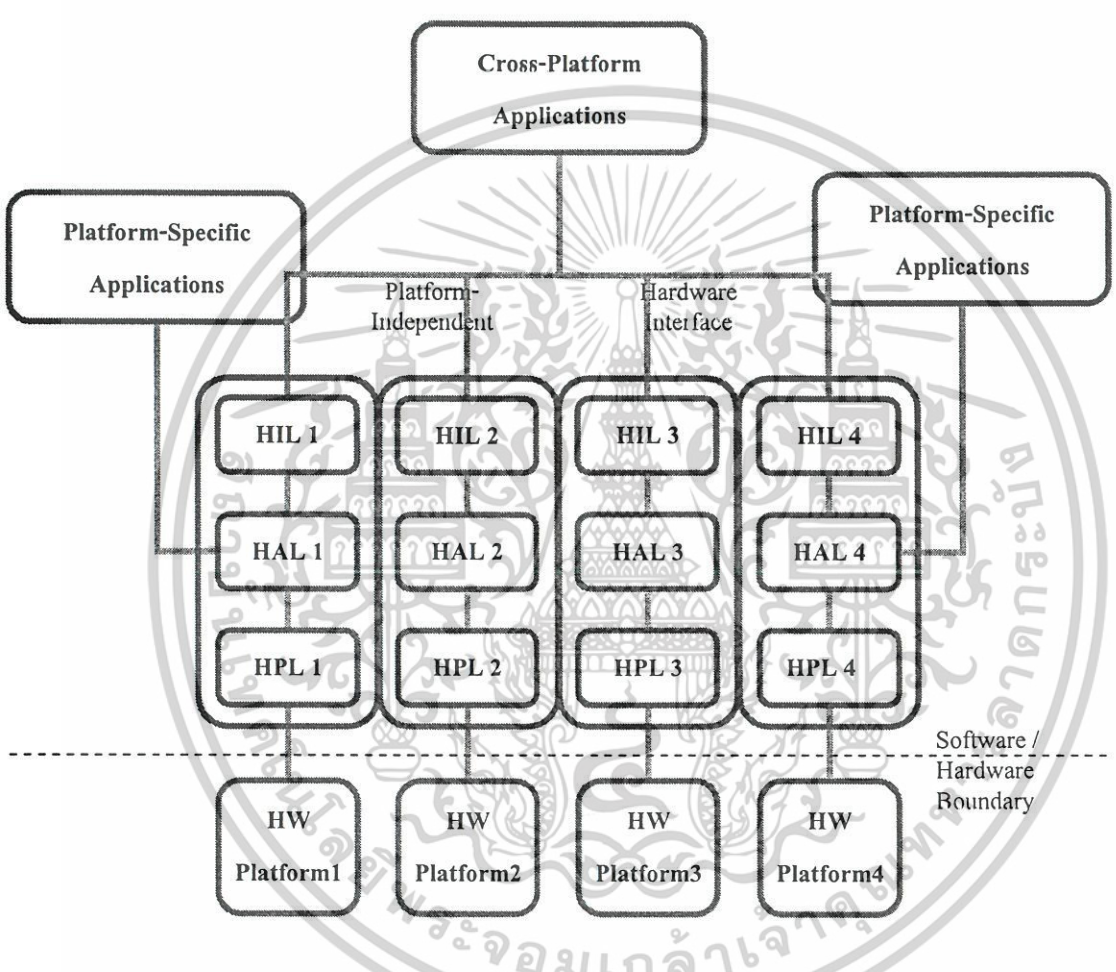
3.1 แพลตฟอร์มและการจัดโครงสร้างของอุปกรณ์ (Platforms and Hardware Abstraction)

3.1.1 แพลตฟอร์ม (Platform)

สำหรับระบบปฏิบัติการ TinyOS หมายถึงกลุ่มของชิป (Chip) และโค้ดโปรแกรมที่เชื่อมต่อกัน ตัวอย่างเช่น แพลตฟอร์ม IRIS ซึ่งเป็นแพลตฟอร์มของอุปกรณ์ในชุด Wireless Sensor Network Classroom Kit ประกอบด้วยหน่วยประมวลผลรุ่น XM2110CA ซึ่งมีพื้นฐานมาจากไมโครคอนโทรลเลอร์ ATmega1281 สำหรับโปรแกรมสำหรับชิป

(Chip code) จะอยู่ใน tos/chips สำหรับตำแหน่งที่อยู่ของแพลตฟอร์มนั้นมักมีไฟล์นามสกุล .platform อยู่ เพื่อเป็นตัวเลือกที่จะส่งเข้าไปยังคอมไพเลอร์ของ nesC

สำหรับความเป็นนามธรรมของส่วนอุปกรณ์ใน TinyOS ส่วนใหญ่แล้วจะแสดงออกเป็น 3 ลำดับขั้น เรียกว่า HAA (Hardware Abstraction Architecture) ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 แสดง Hardware Abstractioin Architecture

3.1.2 HPL (Hardware Presentation Layer)

คือระดับชั้น (เลเยอร์) โปรแกรมที่ติดต่อกับส่วนอุปกรณ์ (hardware) ผ่านอินเตอร์เฟสของ nesC โดยทั่วไปแล้วคอมโพเนนท์ใน HPL จะไม่มีสถานะอื่นนอกจากสถานะของส่วนอุปกรณ์ของตัวเองเท่านั้น และส่วนใหญ่แล้วคอมโพเนนท์ใน HPL มักมีคำนำหน้าว่า 'Hpl' แล้วต่อด้วยชื่อของชิป อาทิ คอมโพเนนท์ใน HPL ของชิปรุ่น CC1000 จะมีชื่อว่า HplCC1000

3.1.3 HAL (Hardware Abstraction Layer)

มีความเป็นนามธรรมสูงกว่า HPL (higher-level abstractions) ทำให้คอมโพเนนท์ในระดับนี้สามารถใช้งานได้สะดวกกว่า HPL แต่ยังคงไว้ซึ่งการทำงานระหว่างส่วนอุปกรณ์ที่สมบูรณ์ และส่วนใหญ่แล้วคอมโพเนนท์ใน HAL มักมีคำนำหน้าเป็นชื่อของชิป อาทิ คอมโพเนนท์ใน HPL ของชิปรุ่น CC1000 จะมีชื่อว่า CC1000

3.1.4 HIL (Hardware Independent Layer)

มีความเป็นอิสระจากส่วนอุปกรณ์ (Hardware independent) ทำให้คอมโพเนนท์ในระดับนี้ มักไม่สนับสนุนการทำงานระหว่างอุปกรณ์อย่างสมบูรณ์เท่ากับคอมโพเนนท์ใน HAL และคอมโพเนนท์ใน HIL จะไม่มีคำนำหน้ากำกับ เพื่อแสดงถึงความเป็นธรรมนามที่แอพพลิเคชั่นสามารถใช้งานได้หลายแพลตฟอร์ม อาทิ คอม-โพเนนท์ใน HIL ของชิปรุ่น CC1000 บทแพลตฟอร์ม mica2 มีชื่อว่า ActiveMessageC

3.2 หน่วยจัดการประมวลผล (Scheduler)

ใน TinyOS จะมีหน่วยจัดการประมวลผลที่ทำงานภายใต้นโยบาย non-preemptive FIFO (First in, First out) โดยในแต่ละงาน (Task) จะมีตำแหน่งที่ถูกจองไว้ในคิวของงาน (Task queue) เป็นของตนเอง และงานชิ้นนั้นสามารถทำงานได้เพียงครั้งเดียว การล้มเหลวจะเกิดขึ้นก็ต่อเมื่องานดังกล่าวเคยทำงานแล้วเท่านั้น หากคอมโพเนนท์ต้องการสั่งให้ทำงานชิ้นเดิมหลายครั้งจะต้องกำหนดสถานะให้เพื่อให้สามารถทำงานซ้ำได้ (ดูรายละเอียดเกี่ยวกับการโปรแกรม งาน (Task) ได้ในบทที่ 4)

3.3 การเริ่มต้นการทำงานของระบบปฏิบัติการ (Booting and Initialization)

อินเตอร์เฟส StdControl ซึ่งทำหน้าในการกำหนดค่าเริ่มต้นให้กับระบบถูกแบ่งออกเป็น 3 อินเตอร์เฟสคือ Init, Scheduler และ Boot โดยอินเตอร์เฟส Init ใช้เริ่มต้นการทำงาน (Initialization) อินเตอร์เฟส Scheduler ใช้เริ่มต้นและหยุดการทำงานของคอมโพเนนท์ และอีกอินเตอร์เฟสหนึ่งคือ Boot ใช้ในการแจ้งเตือนว่าตัวตรวจจับกำลังเริ่มต้นการทำงาน ซึ่งแตกต่างจากอินเตอร์เฟส StdControl ในระบบปฏิบัติการรุ่นก่อนหน้านี้ที่ทำหน้าที่ทั้งสามจากอินเตอร์เฟสเพียงตัวเดียว

สำหรับอินเตอร์เฟส Init จะมีคำสั่ง init ดังแสดงในตารางที่ 3.1

ตารางที่ 3.1 แสดงอินเตอร์เฟส Init

```
interface Init {
  command error_t init();
}
```

อินเตอร์เฟส Init เป็นแบบซิงโครนัส ทำให้สามารถกำหนดลำดับการเริ่มต้นการทำงานให้ได้ เนื่องจากการกำหนดค่าแตกต่างจากการดำเนินการแบบอื่นที่สามารถแทรกการดำเนินกันได้ แต่การกำหนดค่านั้นจะทำเป็นลำดับ (Sequential) ดังนั้นหากกระบวนการเริ่มต้นการทำงานต้องรอคอยการขัดจังหวะ (Interrupt) หรือเหตุการณ์อื่น กระบวนการเริ่มต้นการทำงานจะต้องรอจนกว่าเหตุการณ์เหล่านั้นจะเสร็จสิ้นเช่นกัน นอกจากนั้นคอมโพเนนท์อื่นจะไม่สามารถเริ่มต้นการทำงานได้หากกระบวนการกำหนดค่าเริ่มต้นยังไม่เสร็จสิ้น

สำหรับอินเตอร์เฟส Scheduler ใช้สำหรับการเริ่มต้นและการควบคุมการประมวลผลงาน ดังแสดงในตารางที่ 3.2

ตารางที่ 3.2 แสดงอินเตอร์เฟส Scheduler

```
interface Scheduler {
  command void init();
  command bool runNextTask(bool sleep);
  command void taskLoop();
}
```

จากตารางที่ 3.2 คำสั่ง init เป็นคำสั่งในการกำหนดค่าคิวงาน (Task queue) และโครงสร้างข้อมูลของหน่วยจัดการประมวลผล คำสั่ง runNextTask จะรับค่าบูลีน sleep เพื่อระบุว่าหน่วยจัดการประมวลผลควรทำอย่างไรหากไม่มีงานให้ประมวลผล และจะส่งค่ากลับเป็นบูลีนที่ระบุว่าได้ทำงานนั้นหรือไม่ หากบูลีน sleep มีค่าเป็นเท็จ (False) คำสั่งจะส่งค่ากลับเป็นเท็จโดยทันที แต่หากบูลีน sleep มีค่าเป็นจริง (True) คำสั่งจะยังไม่ส่งค่ากลับจนกว่างานจะได้รับการประมวลผลและคำสั่งดังกล่าวจะบอกให้หน่วยประมวลผลกลาง (CPU) หยุดพักการทำงานจนกว่าจะมีงานใหม่เข้ามา และคำสั่ง taskLoop จะบอกหน่วยจัดการประมวลผลให้เข้าสู่การวนประมวลผลงานแบบไม่มีที่สิ้นสุด

สำหรับอินเตอร์เฟส Boot จะมีตัวรับเหตุการณ์ booted ดังแสดงในตารางที่ 3.3 ซึ่งจะถูกกระตุ้นเมื่อการเริ่มต้นของตัวตรวจจับเสร็จสิ้น

ตารางที่ 3.3 แสดงอินเตอร์เฟส Boot

```
interface Boot {
  event void booted();
}
```

3.4 การบริการในลักษณะเสมือน (Virtualization)

เนื่องจาก TinyOS ได้รับการพัฒนาขึ้นจาก nesC รุ่น 1.2 ซึ่งได้เพิ่มคุณสมบัติใหม่คือ Generic หรือคอมโพเนนท์ที่สามารถสร้างออปเจ็คได้ ทำให้ TinyOS สามารถนำโครงสร้างข้อมูล เช่น เวกเตอร์ (Vector) หรือ คิว (Queue) ซึ่งพัฒนาโดยใช้รูปแบบ Generic modules กลับมาใช้ใหม่ได้ นอกจากนั้นคุณสมบัติ Generic configurations ยังทำให้สร้างบริการ (Service) ที่บรรจุการเชื่อม (Wiring) ความสัมพันธ์เชิงซับซ้อนให้กับเครื่องลูกข่ายที่ต้องการได้ ในทางปฏิบัติแล้ว บริการดังกล่าวนั้นสามารถให้บริการในลักษณะเสมือน (Virtualized) ได้ โดยโปรแกรมจะสร้างออปเจ็คของคอมโพเนนท์บริการที่ได้จัดเตรียมอินเตอร์เฟสที่จำเป็น และบริการนี้จะทำการเชื่อมแบบเบื้องหลังให้โดยอัตโนมัติ

3.5 ตัวจับเวลา (Timer)

สำหรับอินเตอร์เฟสที่เกี่ยวข้องกับเวลาที่ระบบปฏิบัติการ TinyOS ได้จัดเตรียมไว้มีดังแสดงในตารางที่ 3.4

ตารางที่ 3.4 แสดงอินเตอร์เฟสเกี่ยวกับเวลาที่ TinyOS จัดเตรียมไว้ให้

```
interface Counter< precision_tag, size_type >
interface Alarm< precision_tag, size_type >
interface BusyWait< precision_tag, size_type >
interface LocalTime< precision_tag >
interface Timer< precision_tag >
```

อินเตอร์เฟส LocalTime และ Timer ส่วนใหญ่จะถูกเรียกใช้โดยแอพพลิเคชั่นส่วนผู้ใช้ สำหรับอินเตอร์เฟส Alarm, BusyWait และ Counter จะถูกเรียกใช้โด TinyOS และคอมโพเนนท์ที่ผู้ใช้สร้างขึ้น สำหรับรายละเอียดของแต่ละอินเตอร์เฟสมีดังต่อไปนี้

3.5.1 อินเตอร์เฟส Counter

อินเตอร์เฟส Counter จะส่งค่ากลับเป็นเวลาปัจจุบัน โดยอินเตอร์เฟสดังกล่าวได้จัดเตรียมคำสั่ง และตัวรับเหตุการณ์เพื่อจัดการกรณีค่าตัวนับกว้างเกินกว่าที่ระบบกำหนด (Overflow) สำหรับโครงสร้างอินเตอร์เฟสของ Counter ดังแสดงในตารางที่ 3.5

ตารางที่ 3.5 แสดงโครงสร้างของอินเตอร์เฟส Counter

```
interface Counter<precision_tag,size_type>
{
  async command size_type get();
  async command bool isOverflowPending();
  async command void clearOverflow();
  async event void overflow();
}
```

จากตารางที่ 3.5 คำสั่ง get จะส่งค่ากลับเป็นเวลาปัจจุบัน คำสั่ง isOverflow-Pending จะส่งค่ากลับเป็นจริงหากมีการกำหนดสถานะตัวนับกว้างเกินกว่าที่ระบบกำหนดไว้สำหรับตัวนับนี้ หากเป็นกรณีอื่นจะส่งค่ากลับเป็นเท็จ คำสั่ง clearOverflow ยกเลิกตัวนับกว้างเกินกว่าที่ระบบกำหนดไว้ที่อยู่ในระหว่างดำเนินการ โดยล้างค่า และตัวรับเหตุการณ์ overflow จะถูกกระตุ้นเมื่อตัวนับกว้างเกินกว่าที่ระบบกำหนด

3.5.2 อินเตอร์เฟส Alarm

อินเตอร์เฟส Alarm เป็นส่วนเพิ่มเติมของ Counter โดยคำสั่งและตัวรับเหตุการณ์ทั้งหมดในอินเตอร์เฟสนี้เป็นอะซิงโครนัส (Asynchronous) สำหรับอินเตอร์เฟส Alarm ได้จัดเตรียมคำสั่งพื้นฐานและคำสั่งเพิ่มเติม ดังแสดงในตารางที่ 3.6

ตารางที่ 3.6 แสดงโครงสร้างอินเตอร์เฟส Alarm

```
interface Alarm<precision_tag,size_type>
{
  // basic interface
  async command void start( size_type dt );
  async command void stop();
  async event void fired();

  // extended interface
  async command bool isRunning();
  async command void startAt( size_type t0, size_type dt );
  async command size_type getNow();
  async command size_type getAlarm();
}
```

จากตารางที่ 3.6 คำสั่ง start จะยกเลิกสัญญาณแจ้ง (Alarm) ก่อนหน้าและกำหนดว่าจะแจ้งอีกเมื่อใดโดยใช้พารามิเตอร์เวลา dt ที่รับเข้ามา โดยสัญญาณแจ้งจะเรียกได้เพียงครั้งเดียวและจากนั้นจะหยุดลง คำสั่ง stop จะยกเลิกสัญญาณแจ้งก่อนหน้า ตัวรับเหตุการณ์ fired จะถูกกระตุ้นเมื่อสัญญาณแจ้งได้หมดช่วงเวลาลง คำสั่ง isRunning จะส่งค่ากลับเป็นจริงหากสัญญาณแจ้งได้เริ่มต้นและไม่สามารถยกเลิกหรือไม่สามารถเรียกได้ หรือส่งค่ากลับเป็นเท็จในกรณีอื่น คำสั่ง startAt เป็นคำสั่งที่เรียกใช้ภายในระบบ โดยจะยกเลิกสัญญาณแจ้งก่อนหน้าและกำหนดว่าจะแจ้งอีกเมื่อใดโดยใช้พารามิเตอร์เวลา t0 + dt คำสั่งนี้จะช่วยให้ยอมรับกรณีเกิดความล่าช้าเป็นเวลา t0 ก่อนที่จะมีการเรียกคำสั่งนี้ คำสั่ง getNow จะส่งค่ากลับเป็นเวลาปัจจุบันในความกว้างและความเที่ยงตรงของสัญญาณแจ้ง และคำสั่ง getAlarm จะส่งค่ากลับเป็นเวลาที่สัญญาณแจ้งในขณะนั้นจะเรียกหรือเวลาที่สัญญาณแจ้งก่อนหน้าถูกกำหนดให้เรียก คำสั่งนี้สามารถใช้ร่วมกับคำสั่ง startAt เพื่อกำหนดสัญญาณแจ้งจากสัญญาณแจ้งก่อนหน้า เช่นเดียวกับ startAt(getAlarm(), dt) โดย

การเรียกใช้รูปแบบนี้จะถูกใช้ในตัวรับเหตุการณ์ fired เพื่อสร้างสัญญาณแจ้งที่สามารถแจ้งเป็นเวลา (Periodic)

3.5.3 อินเตอร์เฟส BusyWait

อินเตอร์เฟส BusyWait อนุญาตให้เกิดความล่าช้าแบบซิงโครนัสเป็นเวลาสั้นๆ ได้ สำหรับโครงสร้างอินเตอร์เฟสของ BusyWait ดังแสดงในตารางที่ 3.7

ตารางที่ 3.7 แสดงโครงสร้างอินเตอร์เฟส BusyWait

```
interface BusyWait<precision_tag,size_type>
{
  async command void wait( size_type dt );
}
```

จากตารางที่ 3.7 คำสั่ง wait จะยับยั้ง (block) สัญญาณแจ้งเป็นเวลาอย่างน้อย dt

3.5.4 อินเตอร์เฟส LocalTime

อินเตอร์เฟส LocalTime จะให้ค่าเวลาปัจจุบัน แต่จะไม่สนใจกรณีค่าตัวนับกว้างเกินกว่าที่ระบบกำหนด สำหรับโครงสร้างอินเตอร์เฟสของ LocalTime ดังแสดงในตารางที่ 3.8

ตารางที่ 3.8 แสดงโครงสร้างอินเตอร์เฟส LocalTime

```
interface LocalTime<precision_tag>
{
  async command uint32_t get();
}
```

จากตารางที่ 3.8 คำสั่ง get จะส่งค่ากลับเป็นเวลาปัจจุบัน

3.5.5 อินเตอร์เฟส Timer

สำหรับคำสั่งและตัวรับเหตุการณ์ของอินเตอร์เฟส Timer เป็นแบบซิงโครนัส โดยอินเตอร์เฟสนี้รองรับทั้งการจับเวลาเพียงครั้งเดียวและแบบช่วงเวลา สำหรับโครงสร้างอินเตอร์เฟสของ Timer ดังแสดงในตารางที่ 3.9

ตารางที่ 3.9 แสดงโครงสร้างอินเตอร์เฟส Timer

```
interface Timer<precision_tag>
{
  // basic interface
  command void startPeriodic( uint32_t dt );
  command void startOneShot( uint32_t dt );
  command void stop();
  event void fired();

  // extended interface
  command bool isRunning();
  command bool isOneShot();
  command void startPeriodicAt( uint32_t t0, uint32_t dt );
  command void startOneShotAt( uint32_t t0, uint32_t dt );
  command uint32_t getNow();
  command uint32_t gett0;
  command uint32_t getdt();
}
```

จากตารางที่ 3.9 คำสั่ง startPeriodic จะยกเลิกตัวจับเวลาที่ทำงานก่อนหน้าและกำหนดว่าจะแจ้งอีกเมื่อใดโดยใช้พารามิเตอร์เวลา dt ที่รับเข้ามา โดยตัวจับเวลาจะทำงานเป็นช่วงเวลา dt จนกว่าจะหมดเวลา คำสั่ง startOneShot จะทำเหมือนกับคำสั่ง start-Periodic(dt) เพียงแต่ตัวจับเวลาจะทำงานเพียงครั้งเดียวเท่านั้นแล้วหยุดลง คำสั่ง stop จะยกเลิกตัวจับเวลาที่ทำงานก่อนหน้า คำสั่ง จะส่งสัญญาณให้กับตัวจับเวลา โดยจะส่งเพียงครั้งเดียวหรือเป็นช่วงเวลาก็ได้ คำสั่ง isRunning จะส่งค่ากลับเป็นจริงหากตัวจับเวลาได้เริ่มต้นไปแล้ว และยังไม่ถูกยกเลิก และยังไม่ได้เคยถูกเรียกจากกรณีตัวจับเวลาเป็นแบบครั้งเดียว (One-shot timer) (กรณีตัวจับเวลาเป็นแบบช่วงเวลา (Periodic timer) คำสั่ง isRunning จะส่งค่ากลับเป็นจริงเสมอจนกว่าตัวจับเวลาจะถูกยกเลิก) คำสั่ง isOneShot จะส่งค่ากลับเป็นจริงหากตัวจับเวลาเป็นแบบครั้งเดียวหรือจะส่งค่ากลับเป็นเท็จหากตัวจับเวลาเป็นแบบช่วงเวลา คำสั่ง startPeriodicAt จะยกเลิกตัวจับเวลาที่ทำงานก่อนหน้าและกำหนดว่าจะแจ้งอีกเมื่อใดโดยใช้พารามิเตอร์เวลา t0 + dt โดยตัวจับเวลาจะแจ้งทุกช่วงเวลา dt จนกว่าจะหยุดลง สำหรับ d0 นั้นมีลักษณะเช่นเดียวกับอินเตอร์เฟส Alarm คำสั่ง startOneShotAt จะทำงานคล้ายกับคำสั่ง startPeriodicAt เพียงแต่จะทำงานเพียงครั้งเดียว คำสั่ง getNow จะส่งค่ากลับเป็นเวลาปัจจุบันในความกว้างและความเที่ยงตรงของตัวจับเวลา คำสั่ง gett0 และ getdt จะส่งค่ากลับเป็นเวลาที่ตัวจับเวลาเริ่มทำงานก่อนหน้าและความล่าช้าหรือช่วงเวลาของตัวจับเวลาที่ทำงานก่อนหน้า

3.6 การติดต่อสื่อสาร (Communication)

ใน TinyOS ชนิดของบัฟเฟอร์ของข้อความคือ message_t โดยทุกบัฟเฟอร์ของข้อความจะเข้าถึงผ่านอินเตอร์เฟส ตัวอย่างเช่นจะดึงค่าที่อยู่ปลายทางของ AMPacket ที่มีชื่อว่า msg จะมีการเรียกคือ AMPacket.destination(msg)

Active messages เป็นส่วนประกอบหนึ่งใน HIL โดยคอมโพเนนท์ ActiveMessageC ได้กำหนดอินเตอร์เฟสของเครือข่ายให้เป็นมาตรฐาน อาทิ แพลตฟอร์ม mica2 ได้กำหนด CC1000 เป็นระดับชั้น Active message ของ ActiveMessageC ในขณะที่แพลตฟอร์ม TMote ได้กำหนด CC2420 เป็นระดับชั้น Active message ของ ActiveMessageC

สำหรับการติดต่อสื่อสารของ Active message จะกระทำผ่าน Generic component ได้แก่ AMSenderC, AMReceiverC, AMSnooperC, และ AMSnoopingReceiverC

3.7 การตรวจจับ (Sensing)

การตรวจจับ ถือเป็นส่วนสำคัญของการประยุกต์ใช้เครือข่ายตรวจจับแบบไร้สาย เนื่องจาก ชนิดของตัวตรวจจับที่แตกต่างกันสามารถทำให้เกิดความหลากหลายของอินเตอร์เฟสที่ติดต่อกับอุปกรณ์เหล่านั้น แต่อินเตอร์เฟสที่แตกต่างกันก็จะพึ่งพาแอพพลิเคชั่นสำหรับจัดการข้อมูลของตัวตรวจจับร่วมกัน แสดงให้เห็นว่าตัวตรวจจับก็ได้มีอินเตอร์เฟสที่เป็นกลางในการได้เข้ามาซึ่งข้อมูล (Data acquisition) รวมอยู่ด้วย ใน TinyOS จะมีการจัดเตรียมไว้ทั้งอินเตอร์เฟสที่ขึ้นกับและไม่ขึ้นกับตัวตรวจจับ

ใน TinyOS ประกอบด้วยอินเตอร์เฟสที่สามารถใช้ได้อิสระจากตัวตรวจจับ ซึ่งครอบคลุมการทำงานทั่วไป เรียกว่า Source and Sink Independent Drivers (SID)

3.7.1 Split-Phase Small Scalar I/O

กลุ่มอินเตอร์เฟสนี้จะใช้กับการรับส่งข้อมูลที่มีขนาดเล็กและไม่ค่อยมีความสำคัญมากนัก ดังแสดงในตารางที่ 3.10

ตารางที่ 3.10 แสดงโครงสร้างของกลุ่มอินเตอร์เฟสสำหรับการรับส่งข้อมูลขนาดเล็ก

```
interface Read<val_t> {
  command error_t read();
  event void readDone( error_t result, val_t val );
}

interface Write<val_t> {
  command error_t write( val_t val );
  event void writeDone( error_t result );
}

interface ReadWrite<val_t> {
```

ตารางที่ 3.11 แสดงโครงสร้างของกลุ่มอินเตอร์เฟสสำหรับการรับส่งข้อมูลขนาดเล็ก (ต่อ)

```
  command error_t read();
  event void readDone( error_t result, val_t val );

  command error_t write( val_t val );
  event void writeDone( error_t result );
}
```

คอมโพเนนท์ที่การอ่านและเขียนข้อมูลสัมพันธ์กัน เช่น อ่านแล้วค่อยเขียนข้อมูล ซึ่งมีโอกาสที่จะเกิดข้อผิดพลาดในขั้นตอนใดขั้นตอนหนึ่ง ควรใช้อินเตอร์เฟส ReadWrite แต่หากต้องการคอมโพเนนท์ที่ให้บริการอ่านและเขียนข้อมูลในช่วงเวลาเดียวกันได้ ก็ให้ใช้อินเตอร์เฟส Read หรือ Write ที่เป็นอิสระกัน นอกจากนั้น หากพารามิเตอร์ result จากตัวรับเหตุการณ์ Read.readDone และ ReadWrite.readDone ไม่ใช่ SUCCESS ค่าของพารามิเตอร์ val จะต้องเป็นศูนย์ และหากการเรียกคำสั่ง Read.read ได้รับค่าที่ส่งกลับเป็น SUCCESS แต่ไม่ได้เรียกตัวรับเหตุการณ์ Read.readDone คำสั่ง Read.read จะต้องไม่ส่งค่ากลับเป็น SUCCESS

ตัวอย่างการนำไปใช้ ได้แก่ การวัดอุณหภูมิ แรงดันไฟฟ้า ภาพ และการอ่าน ADC

3.7.2 Split-Phase Large Scalar I/O

ในกรณีที่อ่านหรือเขียนข้อมูลขนาดใหญ่ ควรรับส่งผ่านการอ้างอิงตำแหน่งหน่วยความจำแทน สำหรับกลุ่มของอินเตอร์เฟสที่รับข้อมูลผ่านการอ้างอิงตำแหน่งหน่วยความจำจะแสดงในตารางที่ 3.11 โดย SID จะทำหน้าที่จัดการพื้นที่เก็บข้อมูลแทน (Takes ownership) ตั้งแต่การเรียกคำสั่ง read หรือ write จนกระทั่งตัวรับเหตุการณ์ readDone หรือ writeDone ได้รับการกระตุ้น

ตารางที่ 3.12 แสดงโครงสร้างของกลุ่มอินเตอร์เฟสสำหรับการรับส่งข้อมูลขนาดใหญ่

```
interface ReadRef<val_t> {
  command error_t read( val_t* val );
  event void readDone( error_t result, val_t* val );
}

interface WriteRef<val_t> {
  command error_t write( val_t* val );
  event void writeDone( error_t result, val_t* val );
}

interface ReadWriteRef<val_t> {
  command error_t read( val_t* val );
  event void readDone( error_t result, val_t* val );

  command error_t write( val_t* val );
  event void writeDone( error_t result, val_t* val );
}
```

เช่นเดียวกับกลุ่มอินเตอร์เฟสสำหรับการรับส่งข้อมูลขนาดเล็ก ที่กล่าวไปในหัวข้อที่ผ่านมา หากพารามิเตอร์ result จากตัวรับเหตุการณ์ ReadRef.readDone และ Read-WriteRef.readDone ไม่ใช่ SUCCESS ค่าของพารามิเตอร์ val จะต้องเป็นศูนย์ นอกจากนั้น คอมโพเนนท์ที่การอ่านและเขียนข้อมูลสัมพันธ์กัน เช่น อ่านแล้วค่อยเขียนข้อมูล ซึ่งมีโอกาสที่จะเกิดข้อผิดพลาดในขั้นตอนใดขั้นตอนหนึ่ง ควรใช้อินเตอร์เฟส ReadWriteRef แต่หากต้องการคอมโพเนนท์ที่ให้บริการอ่านและเขียนข้อมูลในช่วงเวลาเดียวกันได้ก็ให้ใช้อินเตอร์เฟส ReadRef หรือ WriteRef ซึ่งอินเตอร์เฟสทั้งสองเป็นอิสระต่อกัน ตัวอย่างการนำไปใช้ ได้แก่ การอ่านข้อมูลทิศทางการเคลื่อนที่ในสองระนาบพร้อมกัน

3.7.3 Single-Phase Scalar I/O

ในบางอุปกรณ์อาจต้องเก็บสถานะต้องการให้อุปกรณ์ทำงานได้อย่างรวดเร็ว กรณีนี้จำเป็นต้องการทำงานแบบ single-phase แทนการทำงานแบบ split-phase ตัวอย่างเช่น การเก็บ MAC address ของโหนด สำหรับกลุ่มของอินเตอร์เฟสดังกล่าวแสดงในตารางที่ 3.12

ตารางที่ 3.13 แสดงโครงสร้างของกลุ่มอินเตอร์เฟสสำหรับการเก็บข้อมูลที่มีขนาดเล็ก

```
interface Get<val_t> {
  command val_t get();
}

interface Set<val_t> {
  command void set( val_t val );
}

interface GetSet<val_t> {
  command val_t get();
  command void set( val_t val );
}
```

แต่หากเป็นข้อมูลที่ขนาดใหญ่ ควรใช้การอ้างอิงตำแหน่งของหน่วยความจำแทน ซึ่งกลุ่มของอินเตอรเฟสดังกล่าวแสดงในตารางที่ 3.13

ตารางที่ 3.14 แสดงโครงสร้างของกลุ่มอินเตอร์เฟสสำหรับการเก็บข้อมูลที่มีขนาดใหญ่

```
interface GetRef<val_t> {
  command error_t get( val_t* val );
}

interface SetRef<val_t> {
  command error_t set( val_t* val );
}

interface GetSetRef<val_t> {
```

ตารางที่ 3.15 แสดงโครงสร้างของกลุ่มอินเตอร์เฟสสำหรับการเก็บข้อมูลที่มีขนาดใหญ่ (ต่อ)

```
  command error_t get( val_t* val );
  command error_t set( val_t* val );
}
```

3.7.4 Notification-Based Scalar I/O

ตัวตรวจจับบางประเภทจะใช้ในลักษณะการแจ้งเตือน (Notification) มากกว่าเป็นการร้องขอข้อมูล อาทิ ตัวตรวจจับการเคลื่อนที่ passive-IR (PIR) ตัวตรวจจับเสียงหรือควัน เป็นต้น TinyOS จึงได้จัดเตรียมอินเตอร์เฟสเพื่อรองรับการทำงานรูปแบบนี้ในชื่อ Notify ดังแสดงในตารางที่ 3.14 โดยอินเตอร์เฟสดังกล่าวเหมาะสมสำหรับการรับส่งข้อมูลที่ไม่ค่อยมีความสำคัญมากนัก หากต้องการรับส่งข้อมูลที่มีความสำคัญสูงอาจต้องใช้อินเตอร์เฟสที่เฉพาะกับแพลตฟอร์มหรืออุปกรณ์นั้นๆ

ตารางที่ 3.16 โครงสร้างอินเตอร์เฟส Notify

```
interface Notify<val_t> {
  command error_t enable();
  command error_t disable();
  event void notify( val_t val );
}
```

จากตารางที่ 3.14 คำสั่ง enable และ disable เป็นการเปิดหรือปิดการแจ้งเตือนเหตุการณ์ สำหรับพารามิเตอร์ val ของตัวรับเหตุการณ์ notify มีไว้เพื่อใช้กำหนดในอินเตอร์เฟส Read

3.7.5 Split-Phase Streaming I/O

ในบางกรณี ตัวตรวจจับอาจจะต้องอ่านข้อมูลอย่างต่อเนื่อง เช่น การรับฟังเสียง การรับรู้การเคลื่อนไหวอยู่ตลอดเวลา ดังนั้นในอุปกรณ์ดังกล่าวอาจเตรียมอินเตอร์เฟส ReadStream เพื่อไว้สำหรับการอ่านข้อมูลอย่างต่อเนื่อง ดังแสดงในตารางที่ 3.15

ตารางที่ 3.17 แสดงโครงสร้างอินเตอร์เฟส ReadStream

```
interface ReadStream<val_t> {
  command error_t postBuffer( val_t* buf, uint16_t count );
  command error_t read( uint32_t usPeriod );
  event void bufferDone( error_t result,
                         val_t* buf, uint16_t count );
  event void readDone( error_t result );
}
```

จากตารางที่ 3.15 หลังจากเรียกคำสั่ง postBuffer ไดร์เวอร์จะนำข้อมูลใส่เข้าคิวเพื่อการเขียนลงหน่วยความจำ สำหรับขนาดของหน่วยความจำจะถูกชี้ตำแหน่งโดยพารามิเตอร์ buf ต้องมีขนาดอย่างน้อยเท่ากับขนาดของตัวชี้บนโหนดรวมกับขนาดของพารามิเตอร์ count

หลังจากเขียนข้อมูลลงบัฟเฟอร์แล้ว ลูกข่ายจะสามารถเรียกคำสั่ง read ด้วยช่วงเวลาที่ระบุในพารามิเตอร์ จากนั้นไดร์เวอร์จะเริ่มต้นใส่บัฟเฟอร์ลงไปในคิว หลังจากเสร็จสิ้นแล้วจะส่งสัญญาณไปยังตัวรับเหตุการณ์ bufferDone ลูกข่ายอาจเรียกคำสั่ง postBuffer หลังจากคำสั่ง read เพื่อที่จะเตรียมพื้นที่เก็บข้อมูลใหม่เพื่อที่จะใช้งานอนาคต

ถ้าอุปกรณ์ไม่สามารถเก็บข้อมูลเข้าบัฟเฟอร์ได้ จะต้องส่งสัญญาณไปยังตัวรับเหตุการณ์ readDone ด้วยโค้ดที่แสดงถึงข้อผิดพลาด หากข้อผิดพลาดเกิดขึ้นในขณะอ่านข้อมูล อุปกรณ์จะต้องส่งสัญญาณไปยังตัวรับเหตุการณ์ readDone เช่นกัน โดยก่อนเรียกจะต้องส่งสัญญาณไปยังตัวรับเหตุการณ์ bufferDone เพื่อเคลียรบัฟเฟอร์ นอกจากนั้นหากอยู่ในระหว่างที่ส่งสัญญาณไปยังตัวรับเหตุการณ์ readDone การเรียกคำสั่ง postBuffer จะต้องส่งค่ากลับเป็น FAIL

สำหรับอินเตอร์เฟสสำหรับการเขียนข้อมูล ดังแสดงในตารางที่ 3.16

ตารางที่ 3.18 แสดงโครงสร้างอินเตอร์เฟส WriteStream

```
interface WriteStream<val_t> {
  command error_t postBuffer( val_t* buf, uint16_t count );
  command error_t write( uint32_t period );
  event void bufferDone( error_t result,
                         val_t* buf, uint16_t count );
  event void writeDone( error_t result );
}
```

สำหรับการทำงานของคำสั่ง postBuffer และตัวรับเหตุการณ์ bufferDone จะเหมือนกับอินเตอร์เฟส ReadStream ส่วนคำสั่ง write และตัวรับเหตุการณ์ writeDone จะมีการทำงานคล้ายกับอินเตอร์เฟส ReadStream เช่นกัน

3.8 การจัดการพลังงาน (Power management)

การจัดการพลังงานใน TinyOS จะแบ่งออกเป็นสองส่วนด้วยกัน คือส่วนของไมโครคอนโทรลเลอร์และส่วนของอุปกรณ์ สำหรับส่วนของไมโครคอนโทรลเลอร์คิดจากจำนวนที่อุปกรณ์หรือแหล่งของการขัดจังหวะทำงานอยู่ สำหรับส่วนของอุปกรณ์จะจัดการผ่านตัวตัดสินใจ (ดูรายละเอียดในหัวข้อ การตัดสินใจในการจัดสรรทรัพยากร)

3.9 การตัดสินใจในการจัดสรรทรัพยากร (Arbitration)

เพื่อที่จะสนับสนุนการจัดสรรทรัพยากรอย่างมีประสิทธิภาพ TinyOS 2.0 ได้จัดเตรียมอินเตอร์เฟสของทรัพยากร (Resource interface) ซึ่งเป็นคอมโพเนนท์ใช้ร้องขอและได้มาซึ่งมาทรัพยากร โดยมีผู้ตัดสินใจ (Arbiter) ซึ่งทำหน้าที่จัดเตรียมเกณฑ์การตัดสินใจในการเข้าถึงระหว่างลูกข่าย สำหรับการการตัดสินใจบางครั้ง ผู้ตัดสินใจจะจัดเรียมเกณฑ์ด้านการจัดการพลังงาน (Power management) เช่น หากทรัพยากรดังกล่าวไม่มีการเรียกใช้อาจสั่งให้ปิดตัวลงเพื่อประหยัดพลังงาน เป็นต้น

3.10 โปรโตคอลของเครือข่าย (Network Protocols)

TinyOS ได้จัดเตรียมสองโปรโตคอล คือ Dissemination และ Collection โดย Dissemination มีความน่าเชื่อถือในนำส่งข้อมูลขนาดเล็ก (ขนาดน้อยกว่า 20 ไบต์) ไปยังทุกโหนดในเครือข่าย ในขณะที่ Collection จะมีกระบวนการเราติ้งทรี (routing tree)

บทที่ 4
การพัฒนาโปรแกรมสำหรับตัวตรวจจับแบบไร้สาย

ในบทนี้จะกล่าวถึงการวิเคราะห์และออกแบบระบบ และการออกแบบฐานข้อมูล โดยเริ่มต้นจะกล่าวความต้องการของระบบที่จะใช้ศึกษาการทำงานของตัวตรวจจับแบบไร้สาย และการรับส่งข้อมูลในเครือข่าย จากนั้นจะกล่าวถึงออกแบบแผนภาพยูสเคส แผนภาพแอ็กทีวิตี แผนภาพดีพลอยเมนต์ และสุดท้ายจะกล่าวถึงการออกแบบฐานข้อมูลที่ใช้เก็บข้อมูลและแสดงผลในโปรแกรมที่ชุดพัฒนาจัดมาให้

4.1 ความต้องการของระบบที่ใช้ศึกษา

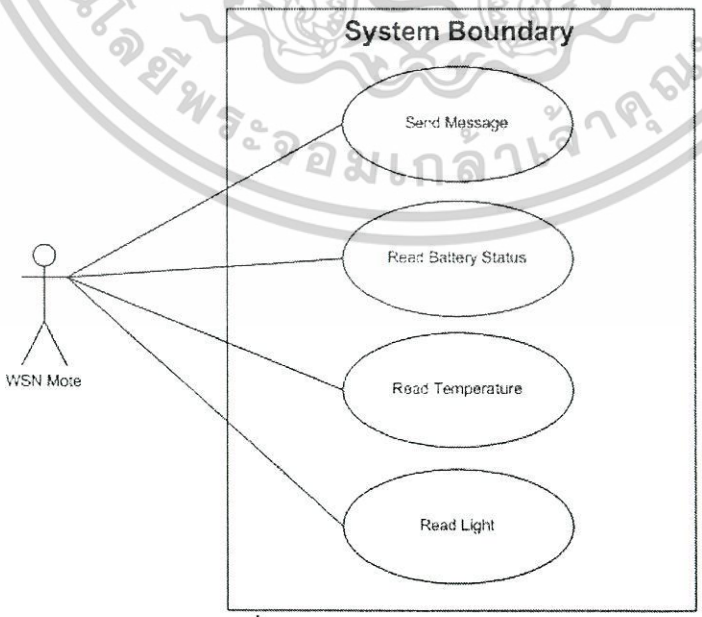
สำหรับโปรแกรม ที่จะใช้ศึกษาการทำงานของตัวตรวจจับแบบไร้สาย แบบการรับส่งข้อมูลในระบบเครือข่ายตัวตรวจจับแบบไร้สาย

- สามารถอ่านอุณหภูมิ ความเข้มของแสง และพลังงานแบตเตอรี่ที่เหลืออยู่ได้
- สามารถรับส่งข้อมูลในเครือข่ายได้

4.2 การวิเคราะห์ระบบ

4.2.1 แผนภาพยูสเคส (Use Case Diagrams)

จากความต้องการของระบบ สามารถนำมาสร้างเป็นแผนภาพยูสเคส ซึ่งอธิบายการทำงานของระบบ สำหรับแผนภาพยูสเคสของระบบแสดงได้ดังรูปที่ 4.1



รูปที่ 4.1 แผนภาพยูสเคส

จากแผนภาพยูสเคส จะประกอบไปด้วย

4.2.1.1 แอ็กเตอร์

เป็นส่วนที่แสดงถึงบุคคลที่เกี่ยวข้องกับระบบ สำหรับแผนภาพยูส-เคสนี้ประกอบไปด้วยแอ็กเตอร์ ดังนี้

- WSN Mote เป็นอุปกรณ์ตัวตรวจจับแบบไร้สาย ที่ทำหน้าที่ในการอ่านเซ็นเซอร์อุณหภูมิและแสงในบริเวณโดยรอบ และรับส่งข้อมูลระหว่างโหนดภายในเครือข่าย นอกจากนั้นยังสามารถอ่านสถานะแบตเตอรี่ของตนเองได้

4.2.1.2 ยูสเคส

เป็นส่วนที่แสดงถึงฟังก์ชั่นที่ระบบกระทำได้ สำหรับรายละเอียดของแต่ละยูสเคสสามารถอธิบายได้จากตารางที่ 4.1 ถึง 4.4

ตารางที่ 4.1 แสดงคำอธิบายยูสเคส Send Message

ชื่อยูสเคส	Send Message
แอ็กเตอร์ปฐมภูมิ	WSN Mote
รายละเอียด	ส่งข้อมูลไปยังโหนดถัดไปภายในเครือข่ายเดียวกัน
เหตุการณ์ที่กระตุ้น	
ลำดับการทำงานปกติ	1. สร้างข้อความที่ต้องการส่งออก 2. ส่งข้อมูลไปยังโหนดถัดไป 3. ส่งค่ากลับว่าส่งข้อความได้สำเร็จ
ลำดับการทำงานทางเลือก	

ตารางที่ 4.2 แสดงคำอธิบายยูสเคส Read Battery Status

ชื่อยูสเคส	Read Battery Status
แอ็กเตอร์ปฐมภูมิ	WSN Mote
รายละเอียด	อ่านค่าแบตเตอรี่
เหตุการณ์ที่กระตุ้น	เมื่อครบกำหนดเวลาที่ตั้งไว้
ลำดับการทำงานปกติ	1. อ่านค่าแบตเตอรี่ 2. เขียนข้อมูลเข้าแพ็คเก็ต
ลำดับการทำงานทางเลือก	1.1 หากมีการอ่านข้อมูลของเซ็นเซอร์อยู่ จะไม่อ่านค่าแบตเตอรี่

ตารางที่ 4.3 แสดงคำอธิบายยูสเคส Read Temperature

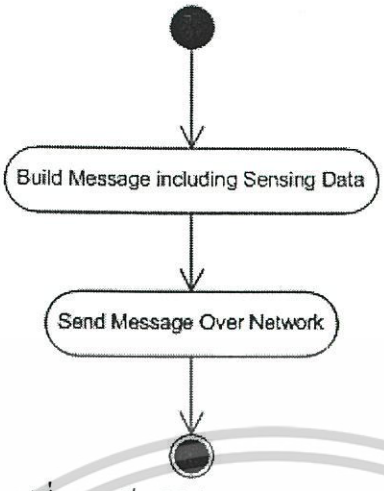
ชื่อยูสเคส	Read Temperature
แอ็กเตอร์ปฐมภูมิ	WSN Mote
รายละเอียด	อ่านค่าอุณหภูมิบริเวณโดยรอบเซ็นเซอร์
เหตุการณ์ที่กระตุ้น	เมื่ออ่านอ่านค่าแบตเตอรี่เรียบร้อยแล้ว
ลำดับการทำงานปกติ	1. อ่านค่าอุณหภูมิ 2. เขียนข้อมูลเข้าแพ็คเก็ต
ลำดับการทำงานทางเลือก	1.1 หากมีการอ่านข้อมูลของเซ็นเซอร์อยู่ จะไม่อ่านค่าอุณหภูมิ

ตารางที่ 4.4 แสดงคำอธิบายยูสเคส Read Light

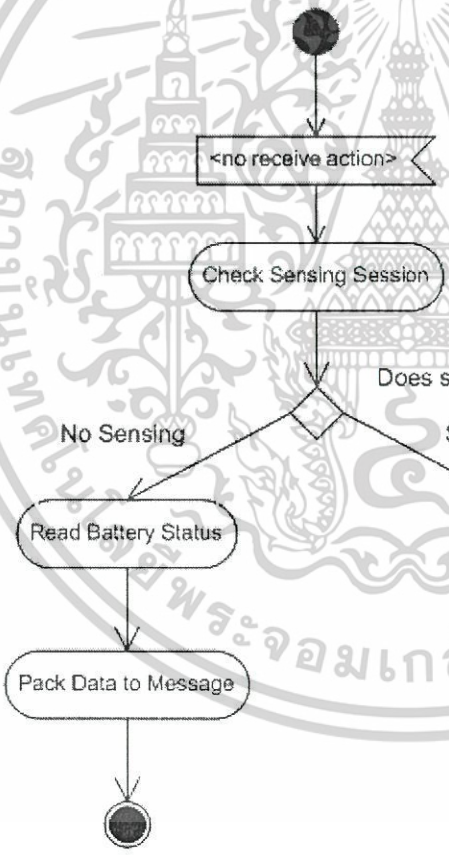
ชื่อยูสเคส	Read Light
แอ็กเตอร์ปฐมภูมิ	WSN Mote
รายละเอียด	อ่านค่าพลังงานแสงบริเวณโดยรอบเซ็นเซอร์
เหตุการณ์ที่กระตุ้น	เมื่ออ่านอ่านค่าแสงเรียบร้อยแล้ว
ลำดับการทำงานปกติ	1. อ่านค่าความเข้มของแสง 2. เขียนข้อมูลเข้าแพ็คเก็ต
ลำดับการทำงานทางเลือก	1.1 หากมีการอ่านข้อมูลของเซ็นเซอร์อยู่ จะไม่อ่านค่าความเข้มของแสง

4.2.2 แผนภาพแอ็กทิวิตี (Activity Diagrams)

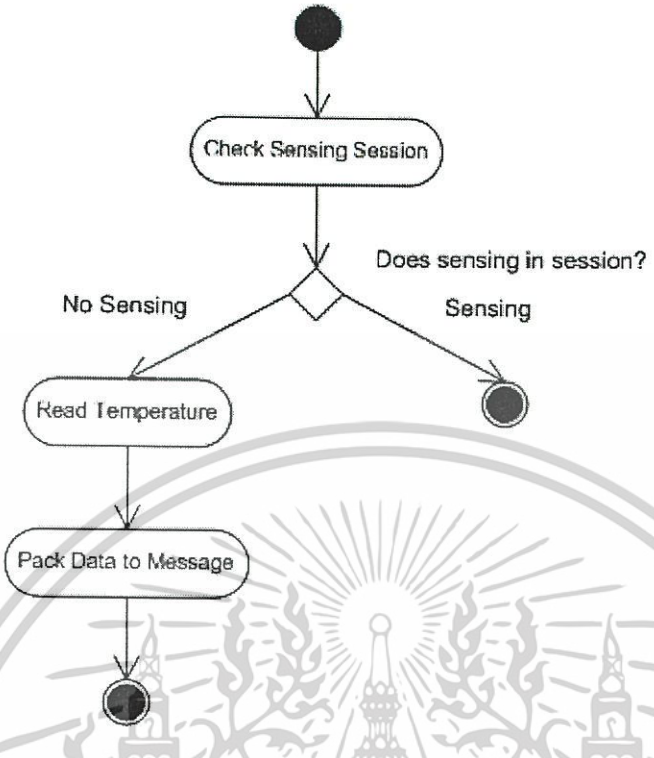
แผนภาพแอ็กทิวิตี จะแสดงขั้นตอนการทำงานของยูสเคสในรูปแบบกิจกรรม (activity) จากคำอธิบายของแต่ละยูสเคส สามารถวาดแผนภาพแอ็กทิวิตีได้ดังนี้



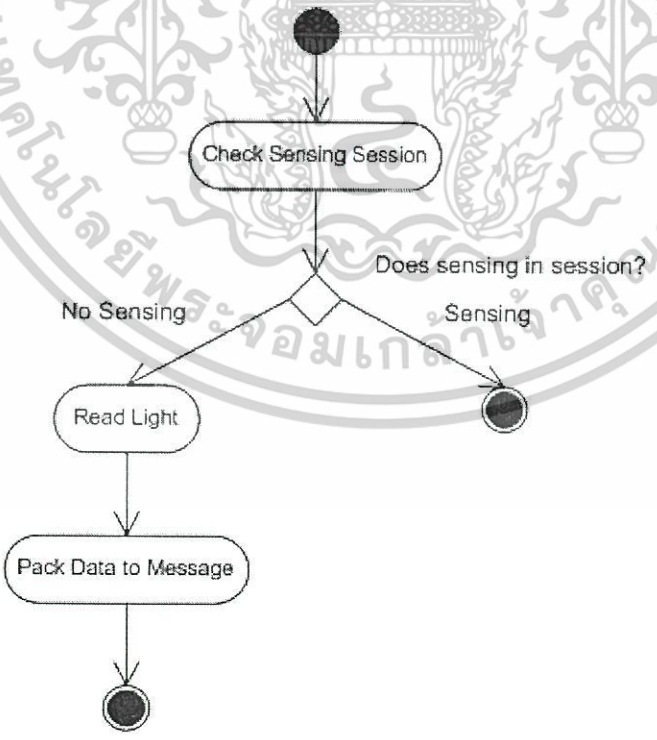
รูปที่ 4.2 แอ็กทิวิตี Send Message



รูปที่ 4.3 แอ็กทิวิตี Read Battery Status



รูปที่ 4.4 แอ็กทิวิตี Read Temperature

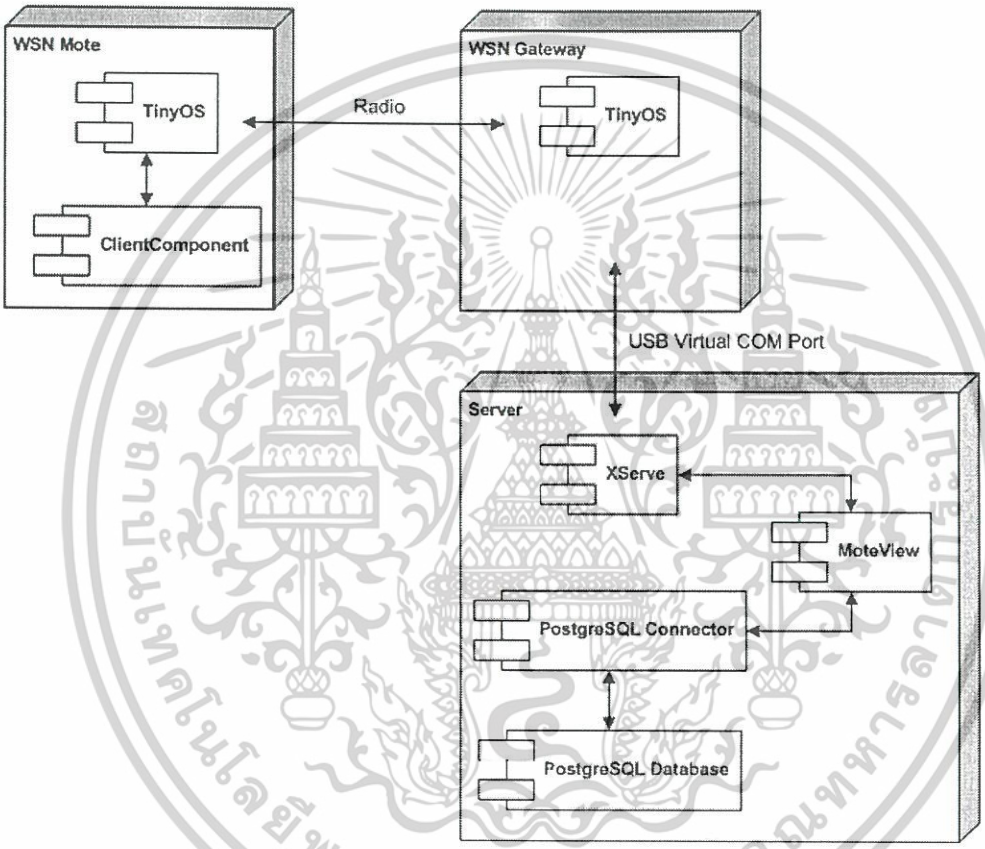


รูปที่ 4.5 แอ็กทิวิตี Read Light

4.3 การออกแบบระบบ

4.3.1 แผนภาพดีพลอยเมนต์ (Deployment Diagrams)

เมื่อระบบได้รับการพัฒนาเสร็จสิ้นแล้ว ช่วงถัดไปจะเป็นการติดตั้งระบบเพื่อศึกษาการทำงานตามที่ได้กล่าวไปแล้วในตอนต้นของบทนี้ โดยแต่ละคอมโพเนนท์จะได้รับการติดตั้งในสถานที่ต่างๆ ดังแสดงในรูปที่ 4.6



รูปที่ 4.6 แผนภาพดีพลอยเมนต์

จากรูปที่ 4.6 จะมีส่วนประกอบดังนี้

4.3.1.1 WSN Mote

เป็นอุปกรณ์ตัวตรวจจับแบบไร้สาย ทำหน้าที่อ่านค่าอุณหภูมิ ความเข้มของแสงจากเซ็นเซอร์ และพลังงานแบตเตอรี่ที่เหลืออยู่ จากนั้นจึงส่งให้ WSN Gateway ผ่านทาง WSN Mote ตัวอื่นไปเรื่อยๆ จนถึง WSN Gateway โดย WSN Mote มีคอมโพเนนท์ดังต่อไปนี้

- ClientComponent เป็นโค้ดที่พัฒนาขึ้น เพื่อให้ตัวตรวจจับอ่านค่าอุณหภูมิ ความเข้มของแสงจากเซ็นเซอร์ และพลังงานแบตเตอรี่ที่เหลืออยู่
- TinyOS เป็นระบบปฏิบัติการที่ฝังกับตัวตรวจจับ โดยจะมีไลบารี่ของชุดคำสั่งต่างๆ ที่จำเป็นต่อการอ่านข้อมูลและรับส่งข้อความ (message) ภายในเครือข่ายเดียวกัน

4.3.1.2 WSN Gateway

เป็นอุปกรณ์ที่ทำหน้าที่รับส่งข้อความระหว่างเครือข่าย หรือระหว่างเครื่องแม่ข่าย โดย WSN Gateway มีคอมโพเนนท์ TinyOS ซึ่งทำหน้าที่ส่วนหนึ่งของคอมโพเนนท์ TinyOS ใน WSN Mote

4.3.1.3 Server

เป็นเครื่องแม่ข่ายที่ติดตั้งคอมโพเนนท์ต่างๆ เพื่อให้สามารถรับข้อมูลจาก WSN Gateway มาจัดเก็บ ประมวลผล และแสดงผล สำหรับ Server มีคอมโพเนนท์ดังต่อไปนี้

- XServe ให้บริการในการค้นหาเส้นทางภายในเครือข่ายและการรับส่งข้อมูลกับแอพพลิเคชั่นในส่วนอื่น
- MoteView เป็นส่วนที่เชื่อมระหว่าง XServe กับ PostgreSQL Connector ใช้แสดงผลข้อมูลที่อ่านจากตัวตรวจจับแล้วส่งต่อเข้าสู่ฐานข้อมูล (PostgreSQL Database) ผ่าน PostgreSQL Connector
- PostgreSQL Database เป็นระบบฐานข้อมูล
- PostgreSQL Connector เป็นตัวเชื่อมต่อระหว่าง MoteView กับ PostgreSQL Database เพื่อให้ MoteView สามารถส่งต่อข้อมูลเข้าสู่ฐานข้อมูลได้

4.3 การออกแบบฐานข้อมูล

สำหรับตารางในการเก็บข้อมูล จะประกอบไปด้วย 3 ตาราง ดังต่อไปนี้

4.4.1 ตาราง wsnstudy

ตาราง wsnstudy เป็นตารางบันทึกข้อมูลที่ส่งเข้ามาจากโหนดในเครือข่าย โดยตารางดังกล่าวมีโครงสร้างดังแสดงในตารางที่ 4.5

ตารางที่ 4.5 แสดงรายละเอียดของตาราง wsnstudy

ชื่อฟิลด์	ประเภทข้อมูล	รายละเอียด	คีย์หลัก	ค่าว่าง
result_time	timestamp without time zone	เวลาที่รายการนี้เข้าตาราง		ได้
nodeid	integer	หมายเลขโหนดที่ส่งข้อมูล		ได้
parent	integer	หมายเลขโหนด		ได้
voltage	integer	พลังงานของแบตเตอรี่ที่เหลืออยู่		ได้
temp	integer	อุณหภูมิที่เซ็นเซอร์อ่านได้		ได้
light	integer	แสงที่เซ็นเซอร์อ่านได้		ได้

4.4.2 ตาราง wsnstudy_i

ตาราง wsnstudy_i เป็นตารางบันทึกข้อมูลของโหนดที่ส่งข้อมูลเข้ามาที่อุปกรณ์เกต์เวย์ โดยตารางดังกล่าวมีโครงสร้างดังแสดงในตารางที่ 4.6

ตารางที่ 4.6 แสดงรายละเอียดของตาราง wsnstudy_i

ชื่อฟิลด์	ประเภทข้อมูล	รายละเอียด	คีย์หลัก	ค่าว่าง
nodeid	integer	หมายเลขโหนดที่ส่งข้อมูล		ได้
x_coord	integer			ได้
y_coord	integer			ได้
z_coord	integer			ได้
mote_name	text	ชื่ออุปกรณ์ของโหนดที่ส่งข้อมูล		ได้
calib	bytea			ได้
board_id	integer	หมายเลขของเซ็นเซอร์บอร์ด		ได้

4.4.3 ตาราง wsnstudy_l

ตาราง wsnstudy_l เป็นตารางบันทึกข้อมูลล่าสุดที่ส่งเข้ามาจากโหนดในเครือข่าย ซึ่งจะใช้ในการแสดงผลบนโปรแกรม MoteView โดยตารางดังกล่าวมีโครงสร้างเช่นเดียวกับตาราง wsnstudy (ดูรายละเอียดเพิ่มเติมเกี่ยวกับโปรแกรม MoteView ได้จากภาคผนวก ค)

บทที่ 5
การศึกษาการทำงานของเครือข่ายตรวจจับแบบไร้สาย

ในการศึกษาการทำงานของเครือข่ายตรวจจับแบบไร้สาย จะแบ่งออกเป็น 2 ส่วน คือ การศึกษากระบวนการตั้งแต่ตัวตรวจจับอ่านข้อมูล แล้วส่งต่อไปยังอุปกรณ์เกตเวย์ของเครือข่าย และการศึกษาการทำงานของโปรโตคอลที่ค้นหาเส้นทาง (Routing protocol) ในสถานการณ์ต่างๆ อาทิ โหนดระหว่างที่ใช้รับส่งข้อมูลปิดตัวลง หรือ เส้นทางการรับส่งข้อมูลถูกกีดขวางทำให้ไม่สามารรถรับส่งต่อในเส้นทางดังกล่าวได้ เป็นต้น

สำหรับการทดลองนั้นจะแบ่งออกเป็น 2 หัวข้อ ดังนี้

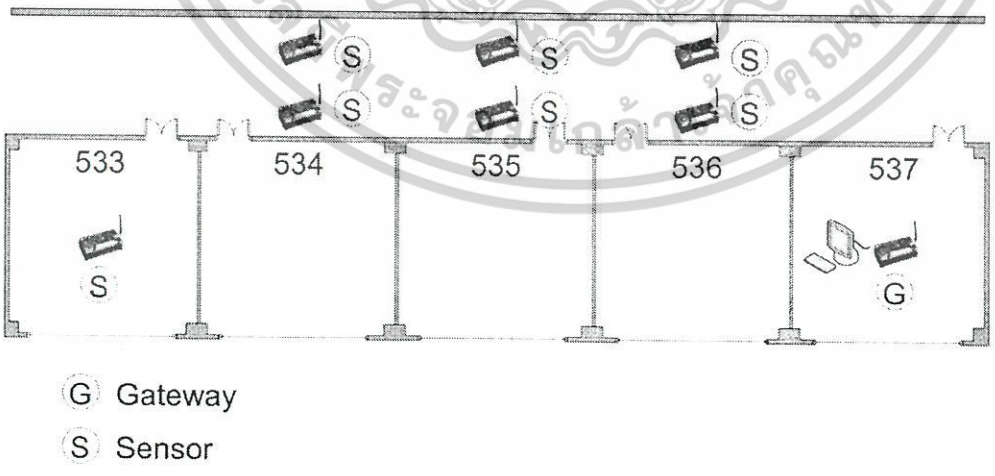
การทดลองที่ 1 การศึกษากระบวนการรับส่งข้อมูลจากต้นทางไปยังอุปกรณ์เกตเวย์แบบหลายโหนด (Multi-hops)

การทดลองที่ 2 การศึกษาการทำงานของโปรโตคอลที่ค้นหาเส้นทางในกรณีเกิดความผิดปกติขึ้นในเส้นทางที่รับส่งข้อมูล

โดยทั้งสองการทดลองนั้นจะใช้อุปกรณ์ในทดลองดังนี้

- ตัวตรวจจับแบบไร้สาย จำนวน 7 ตัว
- อุปกรณ์เกตเวย์ของเครือข่ายพร้อมสายเชื่อมต่อชนิดยูเอสบี จำนวน 1 ชุด
- คอมพิวเตอร์ที่ติดตั้ง XServe และ MoteView

สำหรับการเชื่อมต่อจะเป็นไปตามรูปที่ 5.1



รูปที่ 5.1 แสดงการจัดวางอุปกรณ์ในการทดลอง

5.1 การทดลองที่ 1 การศึกษากระบวนการรับส่งข้อมูลจากต้นทางไปยังเกตเวย์แบบหลายช่วง (Multi-hops)

5.1.1 วัตถุประสงค์ของการทดลอง

เพื่อต้องการศึกษากระบวนการรับส่งข้อมูลจากต้นทางไปยังอุปกรณ์เกตเวย์แบบหลายโหนดในสภาวการณ์ปกติ

5.1.2 สมมติฐาน

โปรโตคอลของเครือข่ายสามารถทำให้รับส่งข้อมูลในเส้นทางที่เร็วที่สุด

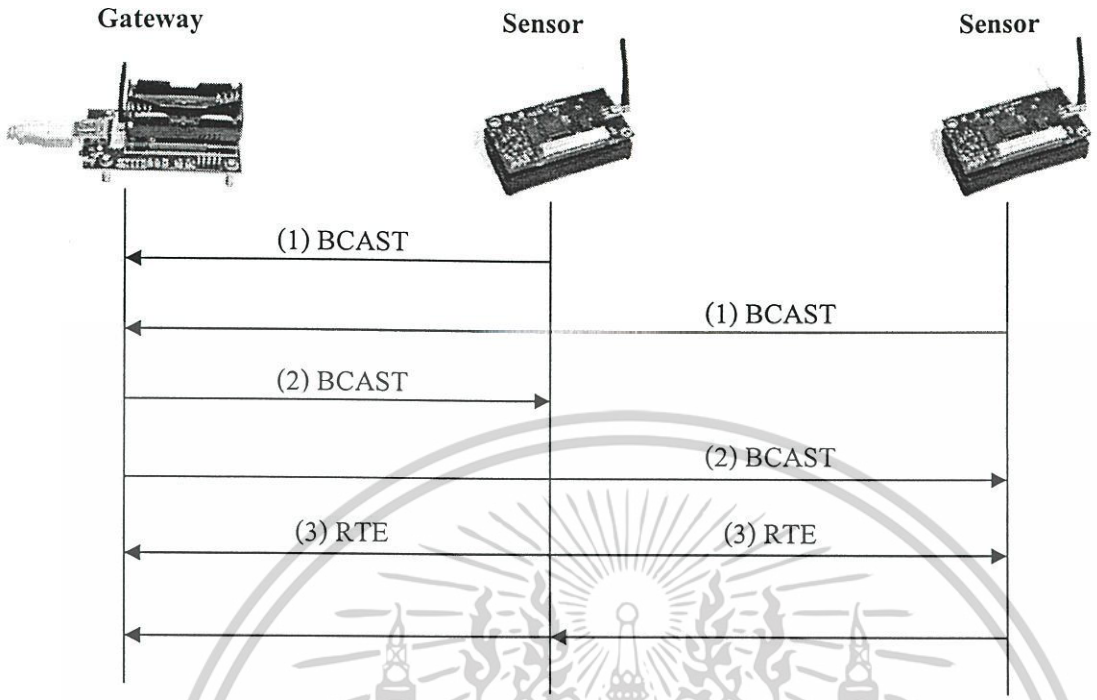
5.1.3 ขั้นตอนในการทดลอง

1. ติดตั้งโปรแกรมที่ใช้ในการทดลองในตัวตรวจจับที่ต้องการให้ตรวจจับ 1 ตัว
2. เชื่อมต่อคอมพิวเตอร์เข้ากับอุปกรณ์เกตเวย์ และจัดวางตัวตรวจจับดังแสดงในรูปที่ 5.1
3. เปิดตัวตรวจจับทุกตัว และเปิดโปรแกรม MoteView
4. เชื่อมต่อโปรแกรม MoteView แล้วสังเกตผลลัพธ์บนหน้าจอ (ดูรายละเอียดเพิ่มเติมเกี่ยวกับโปรแกรม MoteView ได้จากภาคผนวก ค.)

5.1.4 ผลลัพธ์ในการทดลอง

หลังจากการทดลองพบว่า เมื่อเริ่มต้นเครือข่ายแต่ละโหนดภายในเครือข่ายพยายามสื่อสารกับอุปกรณ์เกตเวย์ โดย

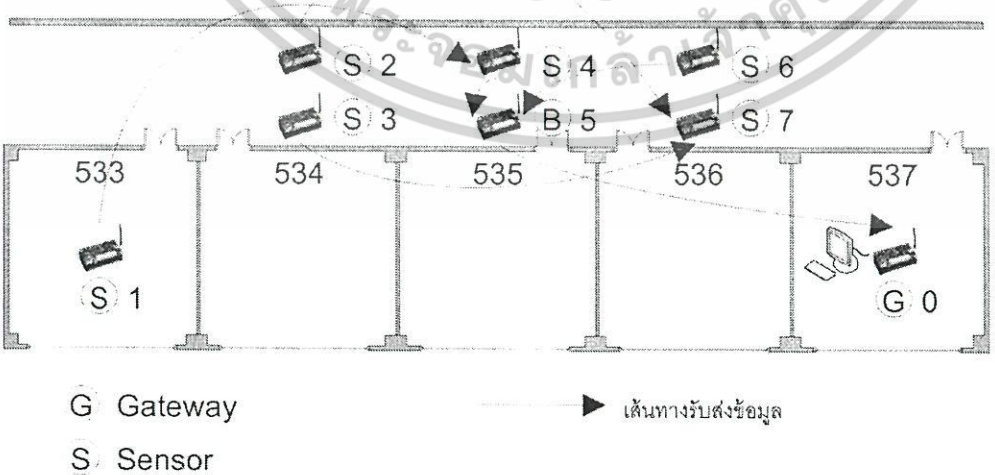
1. พยายามประกาศเฟรมชนิดบรอดคาส (broadcast) ชื่อ BCAST
2. เมื่ออุปกรณ์เกตเวย์ ได้รับจะตอบกลับด้วยเฟรมชนิดยูนิคาส (unicast) ชื่อ AckDwn และในขณะเดียวกันอุปกรณ์เกตเวย์ก็จะพยายามสื่อสารกับโหนดอื่นๆ ภายในเครือข่าย โดยส่งเฟรม BCAST เช่นกัน
3. เมื่อโหนดใดๆ ในเครือข่ายได้รับเฟรม AckDwn แล้ว จะประกาศเฟรมชนิดบรอดแคส ชื่อ RTE
4. ในกรณีที่โหนดอื่นที่อยู่ไกลจากอุปกรณ์เกตเวย์ ได้รับเฟรม RTE อยู่แล้วพบว่าสามารถติดต่อกับอุปกรณ์เกตเวย์โดยผ่านโหนดที่ประกาศเฟรม RTE ออกมาได้ ก็จะเปลี่ยนเส้นทางยังมีโหนดดังกล่าวแทน จากเหตุการณ์ดังกล่าวสรุปได้ดังรูปที่ 5.2



รูปที่ 5.2 แสดงกระบวนการเริ่มต้นการติดต่อสื่อสาร

เมื่อการเรียนรู้เป็นที่เรียบร้อย แต่ละโหนดจะส่งข้อมูลเข้ามาที่อุปกรณ์เกตเวย์โดยใช้เฟรมชนิดยูนิคาส ซึ่งเส้นทางการรับส่งข้อมูลและข้อมูลที่รับส่งนั้นแสดงดังรูปที่ 5.3 ถึง 5.8 ตามลำดับ

<u>หมายเหตุ</u> ลำดับเวลาในรูปที่ 5.4 ถึง 5.5 และ 5.6 ถึง 5.8 จะไม่ตรงกัน เนื่องจากการกำหนดค่าของโปรแกรมที่ต่างกัน



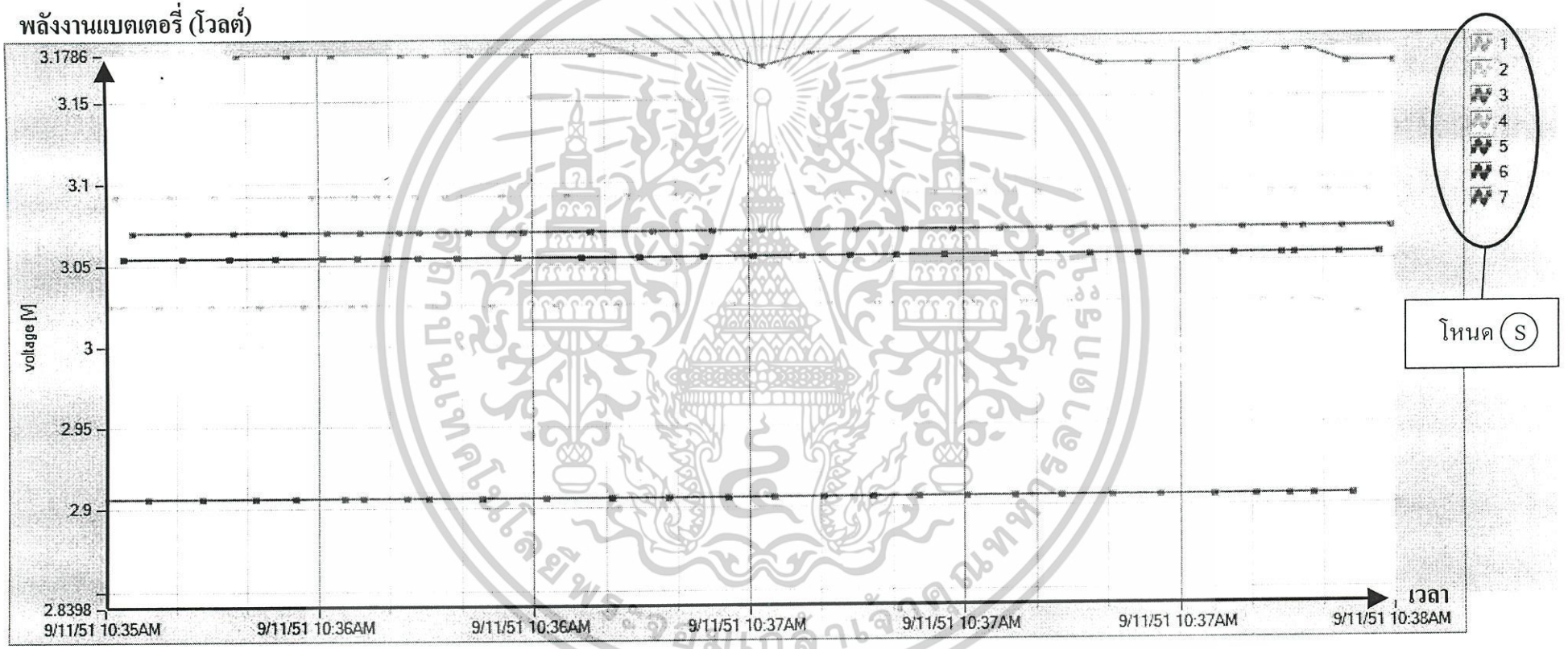
รูปที่ 5.3 แสดงเส้นทางรับส่งข้อมูลของโหนดในเครือข่าย

Id	voltage	temp	light	adc2	adc3	adc4	adc5	adc6	Time
1	2.84V	24.44C	875	1.26V	0.64V	0.46V	0.45V	0.46V	11/9/2551 10:38:01
2	3.09V	27.95C	900	1.67V	1.83V	1.61V	1.5V	1.47V	11/9/2551 10:38:05
3	3.17V	27.69C	868	2.01V	1.94V	1.49V	1.21V	1.23V	11/9/2551 10:38:08
4	3.02V	27.95C	598	1.39V	1.42V	1.23V	1.12V	1.19V	11/9/2551 10:38:04
5	2.91V	28.2C	681	2.21V	1.91V	1.63V	1.52V	1.59V	11/9/2551 10:38:03
6	3.05V	28.12C	692	1.33V	1.84V	1.91V	1.93V	1.95V	11/9/2551 10:38:06
7	3.07V	28.97C	684	1.31V	1.11V	1.22V	1.34V	1.56V	11/9/2551 10:38:07

รูปที่ 5.4 ตารางสรุปข้อมูลที่แต่ละโหนดส่งมาที่เครื่องแม่ข่าย

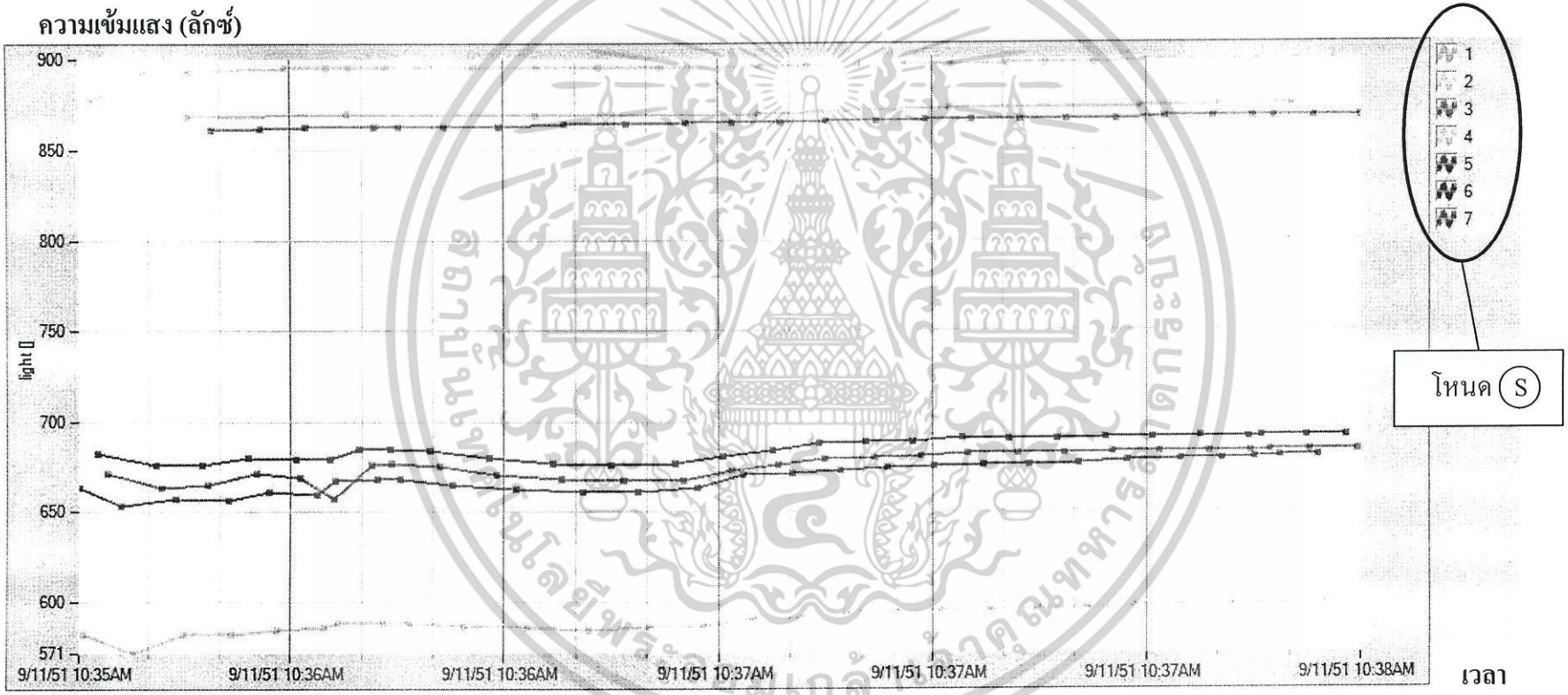
Id	health_pkts	node_pkts	forwarded	dropped	retries	battery	power_sum	board_id	quality_tx	quality_rx	path_cost	parent_rssi	Time
1	6.34%	47.76%	19.03%	33.21%	244.03%	2.8v	0mAHr	145	100%	0%	79	0	11/9/2551 10:36:53
2	4.26%	82.98%	0%	17.02%	29.79%	3v	0mAHr	145	46.67%	66.67%	14	0	11/9/2551 10:37:06
3	5%	65%	0%	35%	0%	3.1v	0mAHr	145	100%	100%	8	3	11/9/2551 10:36:28
4	2.47%	48.15%	41.98%	9.88%	35.8%	3v	0mAHr	145	86.67%	100%	5	0	11/9/2551 10:37:49
5	2.63%	51.32%	39.47%	9.21%	7.89%	2.9v	0mAHr	145	100%	100%	4	0	11/9/2551 10:37:38
6	4.35%	84.78%	0%	15.22%	0%	3v	0mAHr	145	100%	100%	8	18	11/9/2551 10:37:55
7	2.13%	41.49%	50%	8.51%	48.94%	3v	0mAHr	145	100%	100%	8	6	11/9/2551 10:38:06

รูปที่ 5.5 ตารางสรุปข้อมูลการทำงานของแต่ละโหนดในเครือข่าย



หมายเหตุ จุดบนเส้นเป็นจุดที่มีการวัดข้อมูล

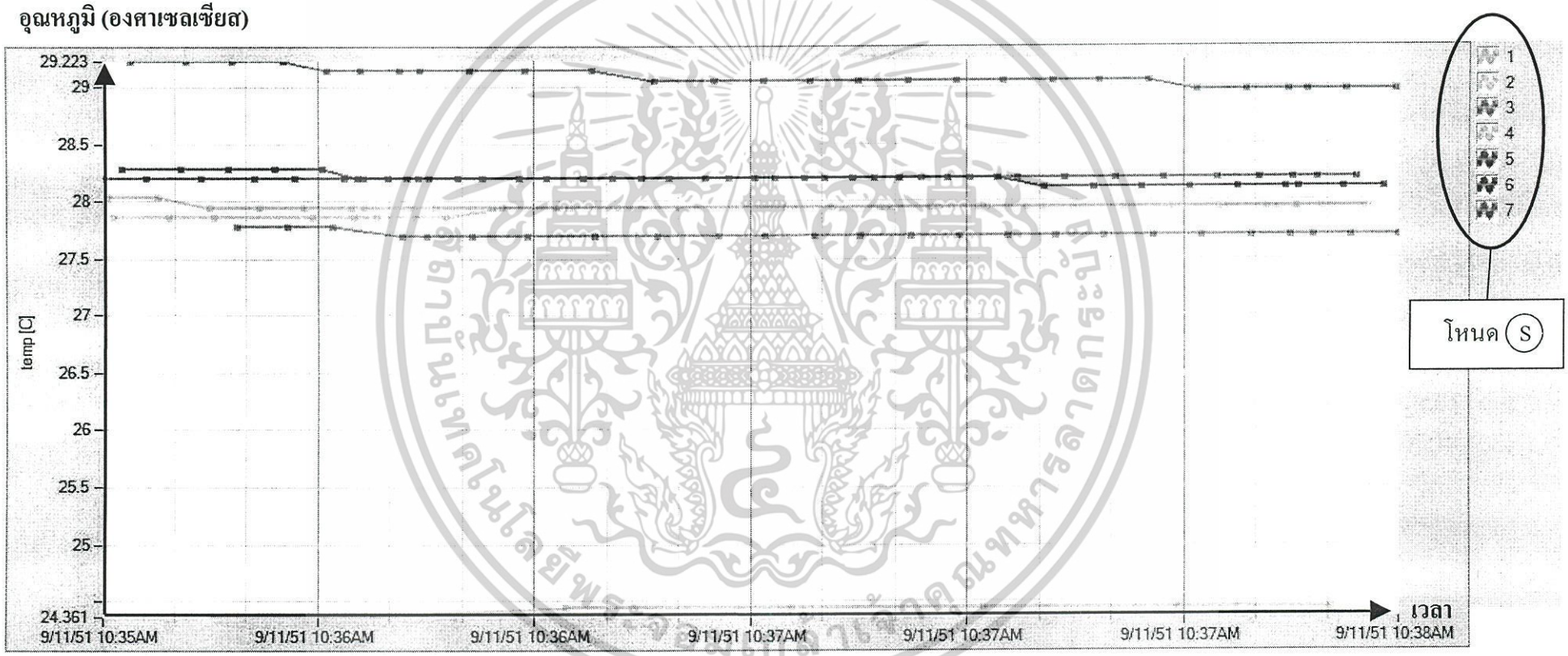
รูปที่ 5.6 กราฟแสดงค่าพลังงานของแบตเตอรี่ที่เหลืออยู่ของแต่ละโหนด



หมายเหตุ 1. จุดบนเส้นเป็นจุดที่มีการวัดข้อมูล

2. เนื่องจากตัวตรวจจับที่ 1 ถึง 3 ได้ถูกวางที่มีแสงสว่างมาก ความเข้มแสงที่วัดได้จึงสูงกว่าตัวตรวจจับที่ 4 ถึง 7 ตัวอื่นๆ ที่ถูกวางอยู่บริเวณที่มีแสงสว่างน้อย

รูปที่ 5.7 กราฟแสดงค่าความเข้มของแสงที่วัดได้ในบริเวณของแต่ละโหนด



หมายเหตุ 1. จุดบนเส้นเป็นจุดที่มีการวัดข้อมูล

2. เนื่องจากตัวตรวจจับที่ 1 ถูกวางในห้องปฏิบัติการที่เปิดเครื่องปรับอากาศ อุณหภูมิที่วัดได้จึงต่ำกว่าตัวตรวจจับตัวอื่นๆ ที่ถูกวางอยู่บริเวณทางเดินหน้าห้องปฏิบัติการ

รูปที่ 5.8 กราฟแสดงค่าอุณหภูมิที่วัดได้ในบริเวณของแต่ละโหนด

5.2 การทดลองที่ 2 การศึกษาการทำงานของโปรโตคอลค้นหาเส้นทางในกรณีเกิดความผิดปกติขึ้นในเส้นทางที่รับส่งข้อมูล

5.2.1 วัตถุประสงค์ของการทดลอง

เพื่อต้องการศึกษาการทำงานของโปรโตคอลค้นหาเส้นทาง ในการหาเส้นทางเพื่อส่งข้อมูลไปยังอุปกรณ์เกตเวย์ในกรณีที่เกิดปัญหาขึ้นระหว่างการรับส่งข้อมูล ทำให้เส้นทางรับส่งข้อมูลเดิมไม่สามารถใช้งานได้

5.2.2 สมมติฐาน

โปรโตคอลค้นหาเส้นทางสามารถปรับเปลี่ยนเส้นทางในการรับส่งข้อมูลได้

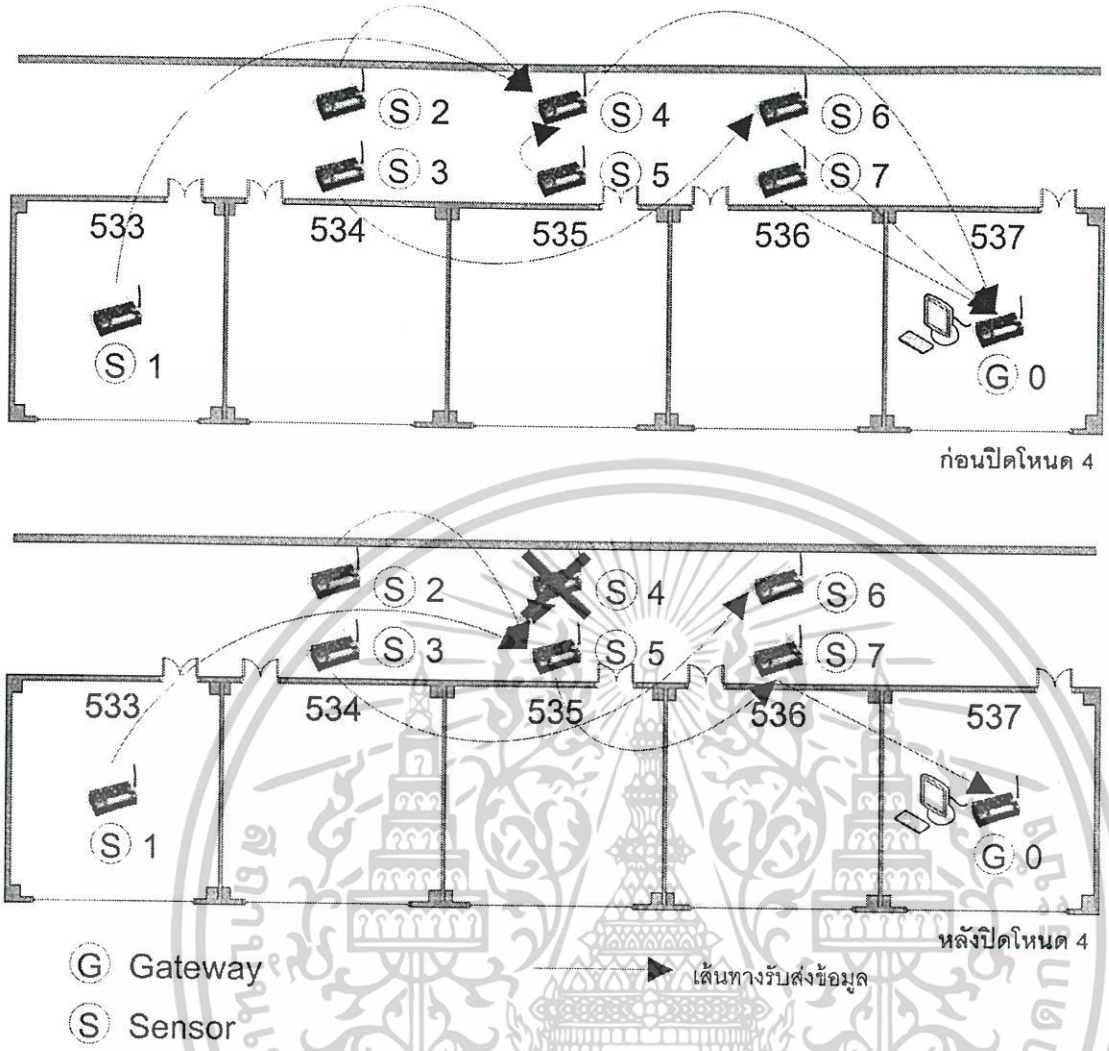
5.2.3 ขั้นตอนในการทดลอง

1. ติดตั้งโปรแกรมที่ใช้ในการทดลองในตัวตรวจจับที่ต้องการให้ตรวจจับ 1 ตัว
2. เชื่อมต่อคอมพิวเตอร์เข้ากับอุปกรณ์เกตเวย์ และจัดวางตัวตรวจจับดังแสดงในรูปที่ 4.1
3. เปิดตัวตรวจจับทุกตัว และเปิดโปรแกรม MoteView
4. เชื่อมต่อโปรแกรม MoteView แล้วสังเกตผลลัพธ์บนหน้าจอ (ดูรายละเอียดเพิ่มเติมเกี่ยวกับโปรแกรม MoteView ได้จากภาคผนวก ค.)
5. ทดลองซ้ำอีก 2 ครั้ง แล้วสรุปผลการทดลอง

5.2.4 ผลลัพธ์ในการทดลอง

เริ่มต้นการรับส่งข้อมูลก็สามารถดำเนินได้โดยปกติ โดยมีเส้นทางจากการทดลอง 1 ใน 3 ครั้ง ดังแสดงในรูปที่ 5.9 (บน) แต่หลังจากได้ปิดสวิตช์โหนด 4 แล้ว ประมาณ 5-6 วินาที โหนดที่ส่งข้อมูลผ่านโหนด 4 จะทราบว่าโหนด 4 นั้นได้ปิดตัวลงแล้ว จึงพยายามหาเส้นทางใหม่ โดยใช้กระบวนการเดียวกับการเริ่มต้นของเครือข่าย ดังที่ได้กล่าวไปในผลลัพธ์ของการทดลองที่ 5.1 สำหรับเส้นทางในการรับส่งข้อมูลหลังจากโหนด 4 จะเป็นดังแสดงในรูปที่ 5.9 (ล่าง)

ข้อสังเกตบางประการจากการทดลอง คือ แต่ละโหนดจะพยายามไม่รับโหลดในการรับข้อมูลจากโหนดอื่นเพื่อส่งต่อเกิน 3 โหนด หากเกินกว่านั้นโหนดที่จะส่งข้อมูลเข้ามาจะพยายามเปลี่ยนเส้นทางไปส่งให้โหนดอื่น



รูปที่ 5.9 แสดงเส้นทางรับส่งข้อมูลของโหนดในเครือข่ายก่อนและหลังการปิดโหนด 4

5.2.5 สรุปผลการทดลอง

จากการทดสอบสรุปได้ว่า โปรโตคอลดังกล่าวสามารถปรับเปลี่ยนเส้นทางในการรับส่งข้อมูลได้โดยใช้เวลาในการรับส่งไม่นานนัก และเส้นทางที่เปลี่ยน คือที่โหนด 4 ไปเป็นโหนดที่ 5 ถือว่ามีความเหมาะสม

บทที่ 6
การพัฒนาระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สาย

ในบทนี้จะกล่าวถึงการวิเคราะห์ ออกแบบระบบ และการออกแบบฐานข้อมูล โดยเริ่มต้นจะกล่าวถึงความต้องการของระบบที่จะใช้ในการพัฒนาระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สาย โดยมุ่งเน้นการเฝ้าติดตามสภาพแวดล้อมภายในอาคาร จากนั้นจะกล่าวถึงการออกแบบแผนภาพยูสเคส และการออกแบบฐานข้อมูล

6.1 ความต้องการของระบบที่ใช้ศึกษา

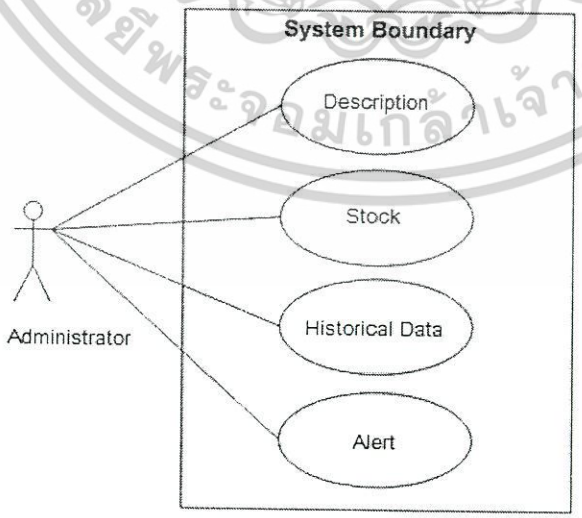
ภาษาที่ใช้ในการพัฒนาระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สาย

- Java(JSP,Servlet,JSTL)
- JavaScipt(excanvas.js,jquery.flot.js, jquery.js)
- Postgre Database

6.2 การวิเคราะห์ระบบ

6.2.1 แผนภาพยูสเคส (Use Case Diagrams)

จากความต้องการของระบบ สามารถนำมาสร้างเป็นแผนภาพยูสเคส ซึ่งอธิบายการทำงานของระบบ สำหรับแผนภาพยูสเคสของระบบแสดงได้ดังรูปที่ 6.1



รูปที่ 6.1 แผนภาพยูสเคส

จากแผนภาพยูสเคส จะประกอบไปด้วย

6.2.1.1 แอ็กเตอร์

เป็นส่วนที่แสดงถึงบุคคลที่เกี่ยวข้องกับระบบ สำหรับแผนภาพยูส-เคสนี้ประกอบไปด้วยแอ็กเตอร์ ดังนี้

- Administrator ผู้ดูแลระบบสามารถเข้าไปจัดการรายละเอียดเกี่ยวกับอุปกรณ์ในระบบคลัง , เข้าไปสร้างระบบอธิบายเครือข่ายตรวจจับแบบไร้สาย , เข้าไปสร้างตารางและกราฟย้อนหลังได้ และ เข้าไปให้ระบบสร้างการตรวจจับเหตุการณ์ที่สนใจได้

6.2.1.2 ยูสเคส

เป็นส่วนที่แสดงถึงฟังก์ชั่นที่ระบบกระทำได้ สำหรับรายละเอียดของแต่ละยูสเคสสามารถอธิบายได้จากตารางที่ 6.1 ถึง 6.4

ตารางที่ 6.1 แสดงคำอธิบายยูสเคส Desciption

ชื่อยูสเคส	Description
แอ็กเตอร์ปฐมภูมิ	Administrator
รายละเอียด	ใช้สร้างสร้างระบบอธิบายเครือข่ายตรวจจับแบบไร้สาย
เหตุการณ์ที่กระตุ้น	
ลำดับการทำงานปกติ	1. กำหนดจำนวนอุปกรณ์ที่ต้องใช้ในระบบ 2. เพิ่มรายละเอียดของอุปกรณ์เข้าไปในระบบ
ลำดับการทำงานทางเลือก	

ตารางที่ 6.2 แสดงคำอธิบายยูสเคส Stock

ชื่อยูสเคส	Stock
แอ็กเตอร์ปฐมภูมิ	Administrator
รายละเอียด	ใช้จัดการระบบคลังอุปกรณ์ในระบบ
เหตุการณ์ที่กระตุ้น	
ลำดับการทำงานปกติ	1. สามารถเพิ่ม ,ลด,แก้ไข รายละเอียดต่างๆของอุปกรณ์
ลำดับการทำงานทางเลือก	

ตารางที่ 6.3 แสดงคำอธิบายยูสเคส Historical Data

ชื่อยูสเคส	Historical Data
แอ็กเตอร์ปฐมภูมิ	Administrator
รายละเอียด	แสดงผลข้อมูลย้อนหลัง
เหตุการณ์ที่กระตุ้น	
ลำดับการทำงานปกติ	1. เลือกเงื่อนไขในการสร้างข้อมูลย้อนหลัง 2. แสดงผลข้อมูลในรูปแบบตาราง 3. แสดงผลข้อมูลในรูปแบบกราฟ
ลำดับการทำงานทางเลือก	

ตารางที่ 6.4 แสดงคำอธิบายยูสเคส Alert

ชื่อยูสเคส	Alert
แอ็กเตอร์ปฐมภูมิ	Administrator
รายละเอียด	ส่งข้อความแจ้งเตือนเมื่อเกิดเหตุการณ์
เหตุการณ์ที่กระตุ้น	
ลำดับการทำงานปกติ	1. กำหนดเงื่อนไขที่จะทำให้เกิดเหตุการณ์แจ้งเตือน 2. แจ้งเตือนเมื่อเกิดเหตุการณ์ตามที่กำหนดไว้ 3. บันทึกการแจ้งเตือนลงฐานข้อมูล
ลำดับการทำงานทางเลือก	

6.3 การออกแบบฐานข้อมูล

สำหรับตารางในการเก็บข้อมูล จะประกอบไปด้วย 3 ตาราง ดังต่อไปนี้

6.3.1 ตาราง login

ตาราง login เป็นตารางที่ใช้ในการตรวจสอบสิทธิ์ในการเข้าใช้ส่วนต่างๆของระบบ โดยตารางดังกล่าวมีโครงสร้างดังแสดงในตารางที่ 6.5

ตารางที่ 6.5 แสดงรายละเอียดของตาราง login

ชื่อฟิลด์	**ประเภทข้อมูล**	**รายละเอียด**	**คีย์หลัก**	**ค่าว่าง**
Username	Varchar	ยูสเซอร์เนม	ใช่	ไม่ได้
Password	Varchar	พาสเวิร์ด		ไม่ได้

6.3.2 ตาราง stock

ตาราง stock เป็นตารางที่ใช้ในการดูจำนวนอุปกรณ์ทั้งหมดและจำนวนที่ใช้อยู่และจำนวนคงเหลือของอุปกรณ์ทั้งหมด โดยตารางดังกล่าวมีโครงสร้างดังแสดงในตารางที่ 6.6

ตารางที่ 6.6 แสดงรายละเอียดของตาราง stock

ชื่อฟิลด์	ประเภทข้อมูล	รายละเอียด	คีย์หลัก	ค่าว่าง
devicename	Varchar	ชื่อของอุปกรณ์	ใช่	ไม่ได้
devicetype	Varchar	ชนิดของอุปกรณ์		ไม่ได้
Description	Varchar	รายละเอียดของอุปกรณ์		ได้
Total	Int	อุปกรณ์ทั้งหมด		ไม่ได้
Used	Int	จำนวนที่ใช้อยู่		ไม่ได้

6.3.3 ตาราง xbw_da100_results

ตาราง xbw_da100_results เป็นตารางบันทึกข้อมูลที่ส่งเข้ามาจากโหนดในเครือข่าย โดยตารางดังกล่าวมีโครงสร้างดังแสดงในตารางที่ 6.7

ตารางที่ 6.7 แสดงรายละเอียดของตาราง xbw_da100_results

ชื่อฟิลด์	ประเภทข้อมูล	รายละเอียด	คีย์หลัก	ค่าว่าง
result_time	timestamp without time zone	เวลาที่รายการนี้เข้าตาราง		ได้
Nodeid	Integer	หมายเลขโหนดที่ส่งข้อมูล		ได้
Parent	Integer	หมายเลขโหนด		ได้
Voltage	Integer	พลังงานของแบตเตอรี่ที่เหลืออยู่		ได้
Temp	Integer	อุณหภูมิที่เซ็นเซอร์อ่านได้		ได้
Light	Integer	แสงที่เซ็นเซอร์อ่านได้		ได้

6.3.4 ตาราง alert

ตาราง alert เป็นตารางที่ใช้เก็บเงื่อนไขที่ใช้ในการตรวจสอบข้อมูล โดยตารางดังกล่าวมีโครงสร้างดังแสดงในตารางที่ 6.8

ตารางที่ 6.8 แสดงรายละเอียดของตาราง alert

ชื่อฟิลด์	ประเภทข้อมูล	รายละเอียด	คีย์หลัก	ค่าว่าง
Id	Int	ลำดับในการเช็ค	ใช่	ไม่ได้
Value	Double	ค่าตัวเลขที่ ใช้ตรวจสอบ		ไม่ได้
Type	Varchar	ชนิดของข้อมูล (light,temp,voltage)		ไม่ได้
Condition	Integer	มากกว่า , น้อยกว่า		ไม่ได้
Message	Integer	Warning , Critical		ไม่ได้

6.3.5 ตาราง History Alert

ตาราง History Alert เป็นตารางที่ใช้เก็บข้อมูลที่แจ้งเตือนไปแล้ว โดยตารางดังกล่าวมีโครงสร้างดังแสดงในตารางที่ 6.9

ตารางที่ 6.9 แสดงรายละเอียดของตาราง History Alert

ชื่อฟิลด์	ประเภทข้อมูล	รายละเอียด	คีย์หลัก	ค่าว่าง
Result_time	Timestamp	เวลา		ไม่ได้
Node_id	Integer	หมายเลขโหนด	ใช่	ไม่ได้
Voltage	Integer	พลังงานของแบตเตอรี่ที่เหลืออยู่		ได้
Temp	Integer	อุณหภูมิที่เซ็นเซอร์อ่านได้		ได้
Light	Integer	แสงที่เซ็นเซอร์อ่านได้		ได้
Motename	Varchar	ชื่อของอุปกรณ์		ไม่ได้

6.4 การตีความข้อมูลจากตัวตรวจจับ

6.4.1 การแปลงข้อมูลดิบเป็นหน่วยองศาเซลเซียส

a = 0.001307050

b = 0.000214381

c = 0.000000093

Rthr = 10000 * (1023-adc) / adc (6.1)

Temperature Celcius = 1/(a+b*log(Rthr)+c*pow(log(Rthr),3)) – 273.15 (6.2)

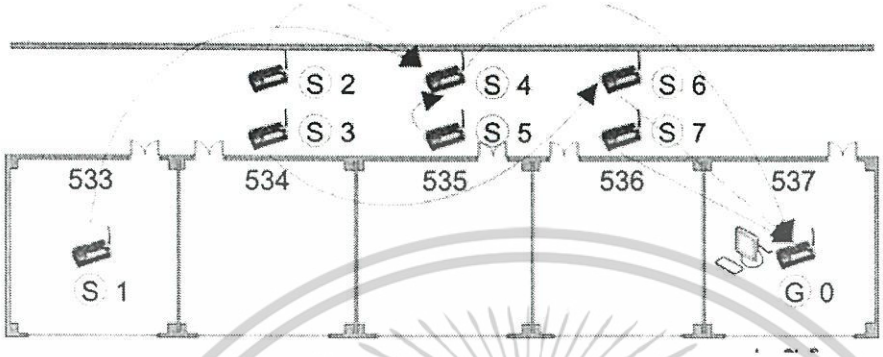
adc คือค่าที่ตัวตรวจจับเก็บลงฐานข้อมูลเทเบิ้ลชื่อ xbw_da100_results ในฟิลด์ชื่อ Temp

6.4.2 การแปลงข้อมูลดิบเป็นหน่วยโวลต์

Voltage volt = 1252.325 / voltage (6.3)

Voltage คือค่าที่ตัวตรวจจับเก็บลงในฐานข้อมูลเทเบิ้ลชื่อ xbw_da100_results ในฟิลด์ชื่อ Voltage

บทที่ 7
การทดสอบระบบสังเกตการณ์ตรวจจับแบบไร้สาย



รูปที่ 7.1 แสดง Topology ของระบบเฝ้าติดตามสภาพแวดล้อมในอาคาร

7.1 วัตถุประสงค์ของการพัฒนาระบบ

แอพพลิเคชั่นนี้พัฒนาขึ้นเพื่อใช้สร้างระบบสังเกตการณ์ตรวจจับแบบไร้สาย และเฝ้าสังเกตการณ์สภาพแวดล้อมภายในอาคาร ทำการตรวจจับอุณหภูมิ ความสว่างของแสง และระดับพลังงานในแบตเตอรี่แล้วแสดงผลผ่านเว็บ และสามารถดูข้อมูลย้อนหลังในรูปแบบกราฟและตารางได้

7.2 สมมติฐาน

สามารถดูข้อมูลย้อนหลังในรูปแบบกราฟและตารางได้ และ สามารถเฝ้าติดตามให้มีการแจ้งเตือนได้

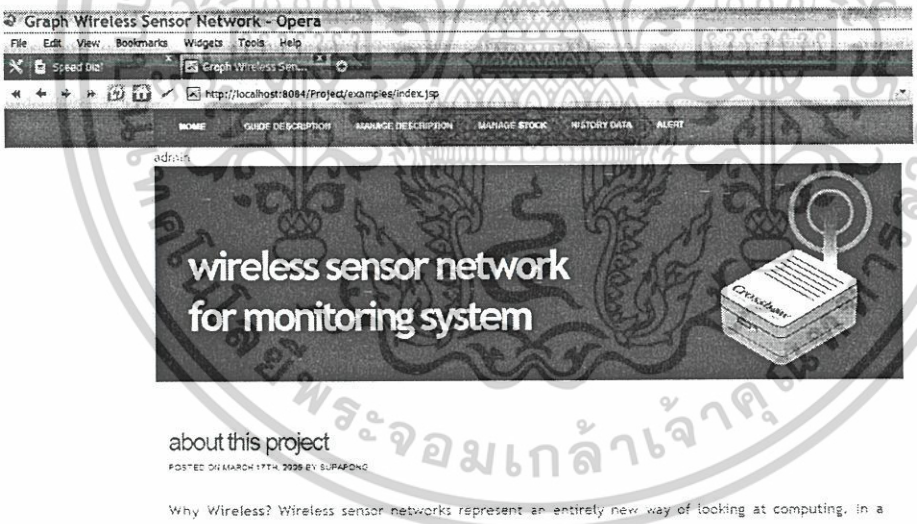
7.3 ขั้นตอนการทดสอบ

7.3.1 ฟังก์ชั่น login

เป็นฟังก์ชั่นที่ใช้ในการตรวจสอบผู้ใช้งานระบบว่ามีสิทธิ์ในการเข้าใช้ระบบหรือไม่

รูปที่ 7.2 หน้า Login

รูปที่ 7.3 Login ไม่ผ่าน



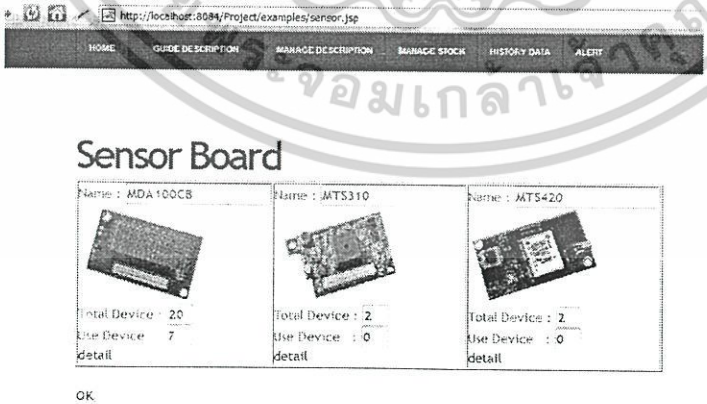
รูปที่ 7.4 Login ผ่านแล้วมาที่หน้า index

7.3.2 ฟังก์ชั่น Description

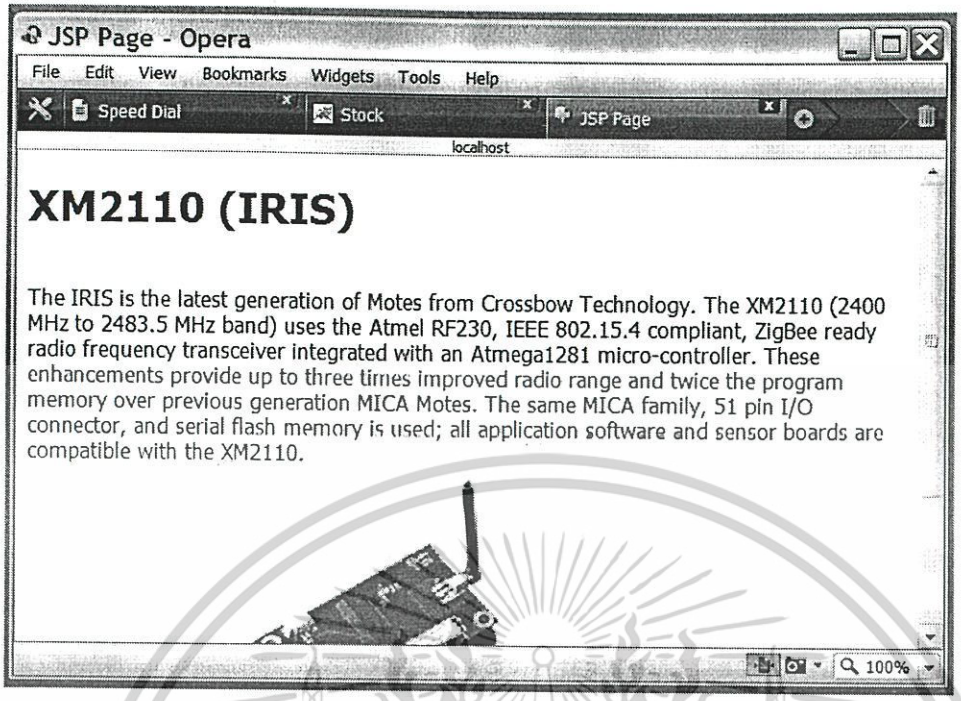
เป็นฟังก์ชั่นที่ช่วยอธิบายรายระเอียดของระบบสังเกตการณ์เครือข่ายตรวจจับแบบไร้สาย ชนิด,จำนวนอุปกรณ์ และชื่อเพื่อใช้สื่อความหมายของอุปกรณ์ให้ผู้ใช้เข้าใจ สามารถทำการแก้ไข ปรับปรุง ลบข้อมูลได้

รูปที่ 7.5 หน้าให้ใส่จำนวนอุปกรณ์เกตเวย์ที่ต้องการใช้

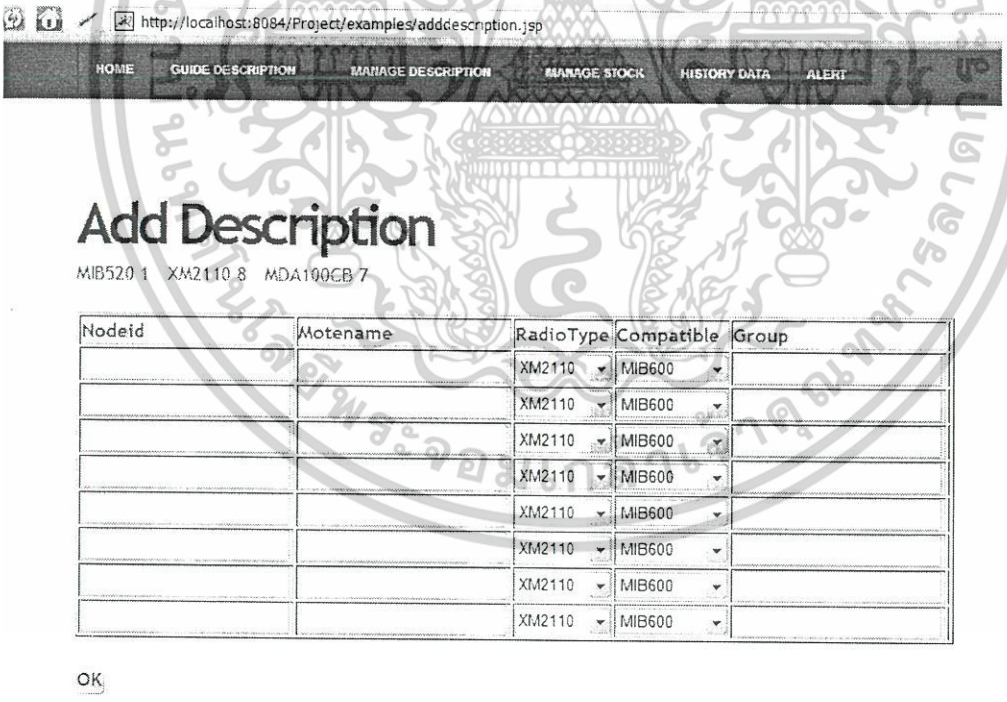
รูปที่ 7.6 หน้าให้ใส่จำนวนเรดิโอโปรเซสเซอร์ที่ต้องการใช้



รูปที่ 7.7 หน้าให้ใส่จำนวนเซนเซอร์ที่ต้องการใช้



รูปที่ 7.8 หน้าแสดงรายละเอียดหลังจากกดลิงค์



รูปที่ 7.9 หน้าเพิ่มรายละเอียดของอุปกรณ์

Description

NodeID	Motename	Group	Radio	Compatible
0	Gateway	Floor 5	XM2110 detail	MIB520 detail
1	room 533	Floor 5	XM2110 detail	MDA100CB detail
2	room 534 far	Floor 5	XM2110 detail	MDA100CB detail
3	room 534 near	Floor 5	XM2110 detail	MDA100CB detail
4	room 535 far	Floor 5	XM2110 detail	MDA100CB detail
5	room 535 near	Floor 5	XM2110 detail	MDA100CB detail
6	room 536 far	Floor 5	XM2110 detail	MDA100CB detail
7	room 537	Floor 5	XM2110 detail	MDA100CB detail

Add Update Delete

รูปที่ 7.10 หน้าแสดงรายละเอียดของอุปกรณ์

Update Description

NodeID	Motename	Group	Radio	Compatible
0	Gateway	Floor 5	XM2110	MIB520
1	room 533	Floor 5	XM2110	MDA100CB
2	room 534 far	Floor 5	XM2110	MDA100CB
3	room 534 near	Floor 5	XM2110	MDA100CB
4	room 535 far	Floor 5	XM2110	MDA100CB
5	room 535 near	Floor 5	XM2110	MDA100CB
6	room 536 far	Floor 5	XM2110	MDA100CB
7	room 537	Floor 5	XM2110	MDA100CB

OK

รูปที่ 7.11 หน้าปรับเปลี่ยนรายละเอียดของอุปกรณ์

Delete Description

NodeID	Motename	Group	Radio	Compatible	check
0	Gateway	Floor 5	XM2110	MIB520	☐
1	room 533	Floor 5	XM2110	MDA100CB	☐
2	room 534 far	Floor 5	XM2110	MDA100CB	☐
3	room 534 near	Floor 5	XM2110	MDA100CB	☐
4	room 535 far	Floor 5	XM2110	MDA100CB	☐
5	room 535 near	Floor 5	XM2110	MDA100CB	☐
6	room 536 far	Floor 5	XM2110	MDA100CB	☐
7	room 537	Floor 5	XM2110	MDA100CB	☐

OK

รูปที่ 7.12 หน้าลบรายละเอียดของอุปกรณ์

7.3.3 ฟังก์ชั่น Stock

เป็นฟังก์ชั่นที่ใช้จัดการจำนวนอุปกรณ์ที่มีอยู่ และสามารถทำการแก้ไข ปรับปรุง ลบ ชนิด,จำนวนอุปกรณ์ได้

Stock Device

DeviceName	DeviceType	Description	Total	Use
MIB600 detail	Gateway	MIB600 provides Ethernet (10/100 Base-T) connectivity to IRIS and MICA family Motes	5	0
MIB520 detail	Gateway	MIB520 provides USB connectivity to the IRIS and MICA family of Motes	10	1
XM2110 detail	Processor+Radio Board	IRIS (2400 MHz to 2483.5 MHz band) uses the Atmel RF230, IEEE 802.15.4 compliant, ZigBee	20	8
MPR2400 detail	Processor+Radio Board	MICAz (2400 MHz to 2483.5 MHz band) uses the Chipcon CC2420, IEEE 802.15.4 compliant, ZigBee	5	0
MTS310 detail	Sensor	Light, Temperature, Acoustic, Sounder, Dual-Axis Accel and Dual-Axis Mag Sensor Board	2	0
MTS420 detail	Sensor	Light, Temperature, Humidity, Barometric Pressure, Seismic and GPS	2	0
MDA100CB detail	Sensor	Light,Temperature,Prototype Area Sensor/DAQ Board	20	7

Change Add Del

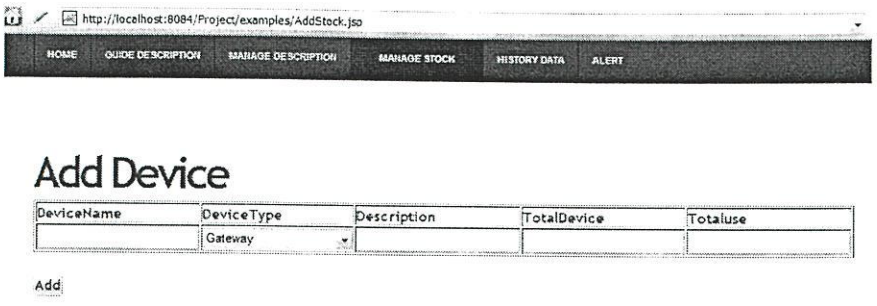
รูปที่ 7.13 หน้าแสดงรายละเอียดและจำนวนของอุปกรณ์

Change

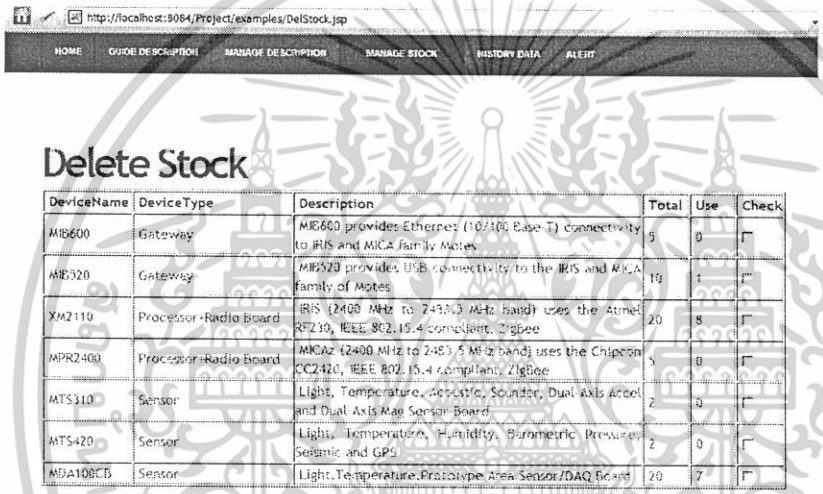
DeviceName	DeviceType	Description	TotalDevice	Totaluse
MIB600	Gateway	MIB600 provides Ethe	5	0
MIB520	Gateway	MIB520 provides USB	10	1
XM2110	Processor+Radio Board	IRIS (2400 MHz to 24	20	8
MPR2400	Processor+Radio Board	MICAz (2400 MHz to	5	0
MTS310	Sensor	Light, Temperature	2	0
MTS420	Sensor	Light, Temperature, I	2	0
MDA100CB	Sensor	Light,Temperature,Pr	20	7

OK

รูปที่ 7.14 หน้าปรับเปลี่ยนรายละเอียดและจำนวนของอุปกรณ์



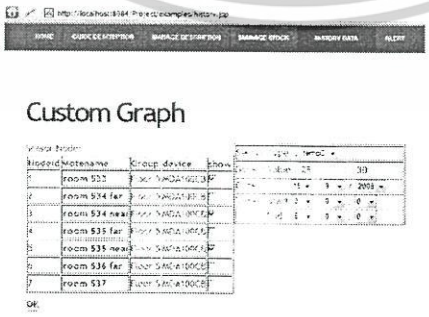
รูปที่ 7.15 หน้าเพิ่มรายละเอียดและจำนวนของอุปกรณ์



รูปที่ 7.16 หน้าลบรายละเอียดและจำนวนของอุปกรณ์

7.3.4 ฟังก์ชั่น Historical Data

เป็นฟังก์ชั่นที่ใช้ในการแสดงข้อมูลย้อนหลังในรูปแบบของกราฟและตาราง

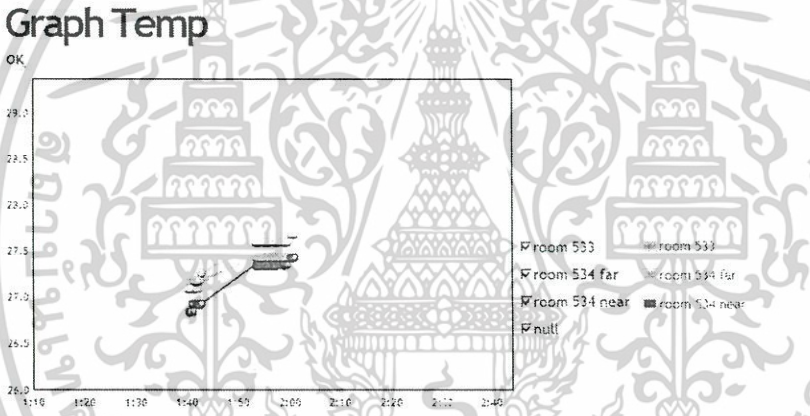


รูปที่ 7.17 หน้าเลือกข้อมูลที่จะนำไปแสดงผลในรูปแบบกราฟและตาราง

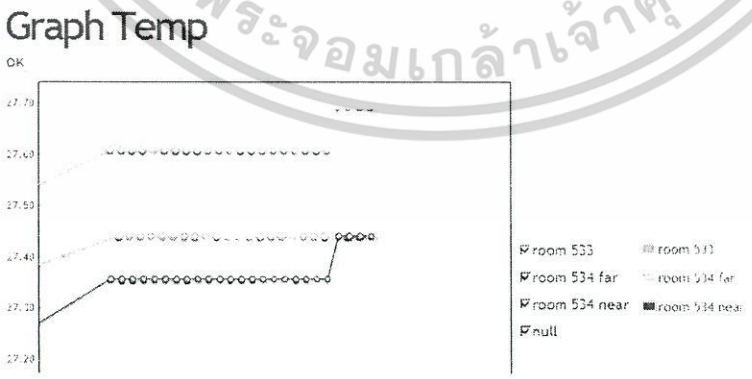
Table Temp

Nodeid	Motename	Result_time	Temp
1	room 533	2008-09-19 01:40:45.203	27.704751163207 31
1	room 533	2008-09-19 01:40:45.281	27.704751163207 31
1	room 533	2008-09-19 01:41:06.421	27.704751163207 31
1	room 533	2008-09-19 01:41:06.453	27.704751163207 31
1	room 533	2008-09-19 01:41:06.531	27.704751163207 31
1	room 533	2008-09-19 01:41:06.593	27.704751163207 31
1	room 533	2008-09-19 01:41:24.296	27.704751163207 31
1	room 533	2008-09-19 01:41:24.375	27.704751163207 31
1	room 533	2008-09-19 01:41:24.437	27.704751163207 31
1	room 533	2008-09-19 01:41:24.515	27.704751163207 31

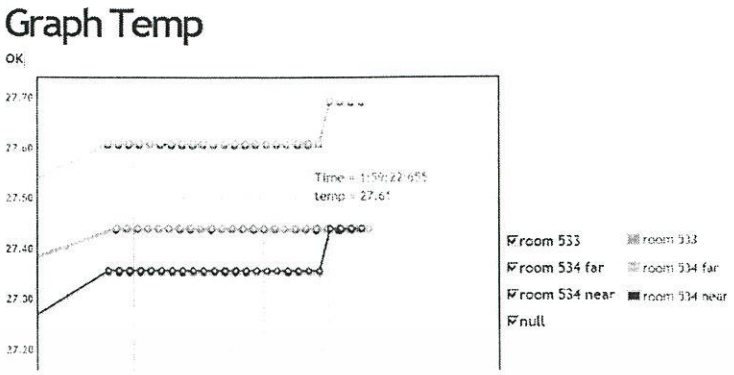
รูปที่ 7.18 หน้าแสดงข้อมูลในรูปแบบตาราง



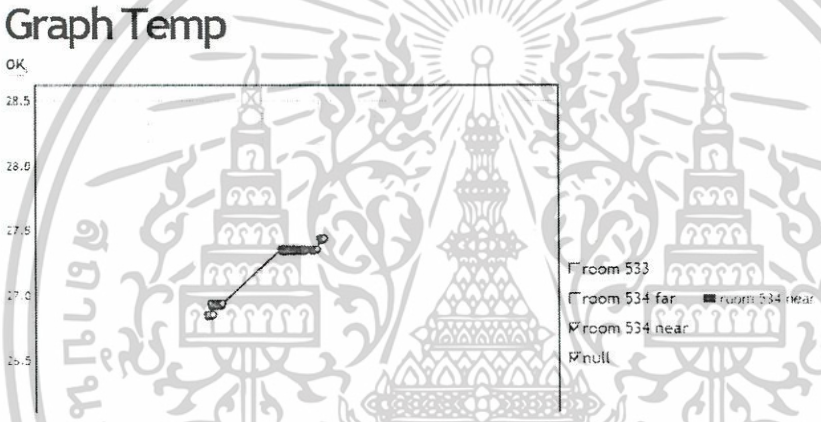
รูปที่ 7.19 หน้าแสดงข้อมูลในรูปแบบกราฟ



รูปที่ 7.20 ฟังก์ชั่นซูม



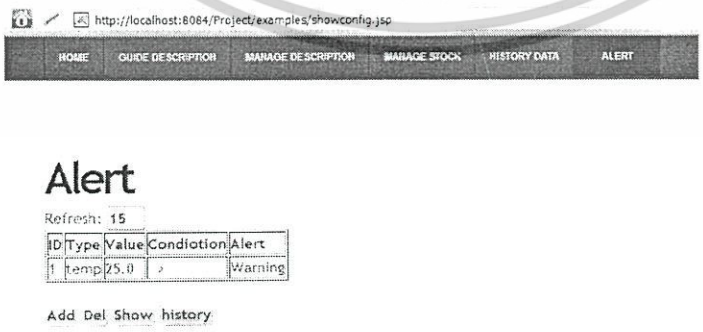
รูปที่ 7.21 ฟังก์ชั่นแสดงผลข้อมูล



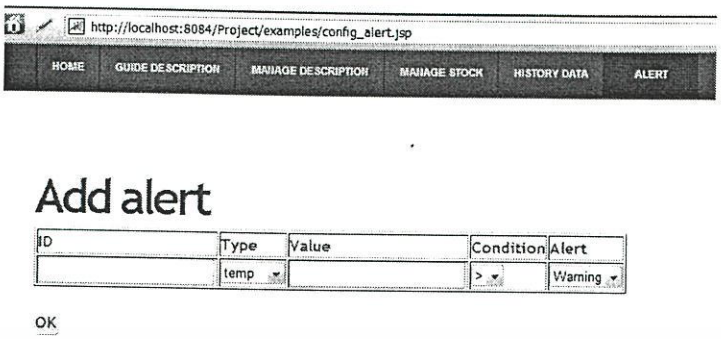
รูปที่ 7.22 ฟังก์ชั่นเลือกแสดงผลข้อมูล

7.3.5 ฟังก์ชั่น Alert

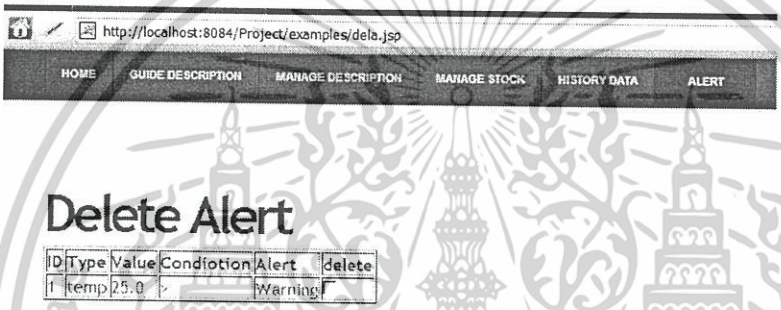
เป็นฟังก์ชั่นที่ใช้ในการแจ้งเตือนเมื่อเกิดเหตุการณ์



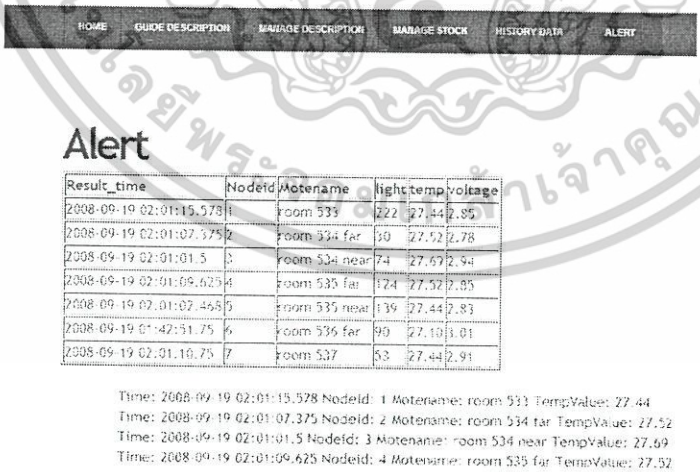
รูปที่ 7.23 หน้าแสดงเหตุการณ์ที่จะทำให้เกิดการแจ้งเตือน



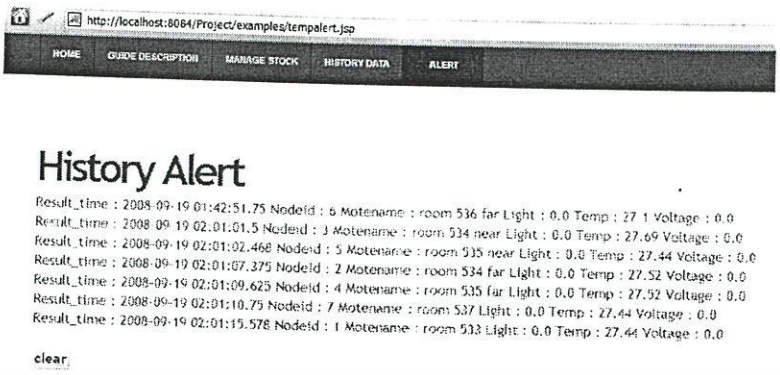
รูปที่ 7.24 หน้าเพิ่มเหตุการณ์ที่จะทำให้เกิดการแจ้งเตือน



รูปที่ 7.25 หน้าลบเหตุการณ์ที่จะทำให้เกิดการแจ้งเตือน



รูปที่ 7.26 หน้าแสดงเหตุการณ์การแจ้งเตือน



รูปที่ 7.27 หน้าแสดงเหตุการณ์ย้อนหลังการแจ้งเตือน

7.4 สรุปผลการทดสอบ

โปรแกรมสามารถสร้างข้อมูลย้อนหลังในรูปแบบตารางและกราฟ และทำการแจ้งเตือนได้

7.5 ปัญหาและอุปสรรค

ปัญหาในการแปลงแปลงหน่วย ปัญหาในการใช้ภาษา javascript กับ jsp servlet ปัญหาการขาดความเข้าใจการจะพัฒนาอุปกรณ์เหล่านี้ให้เป็นระบบ

บทที่ 8
สรุปผลการวิจัย และข้อเสนอแนะ

8.1 สรุปผลการศึกษาเครือข่ายตรวจจับแบบไร้สาย

เครือข่ายตรวจจับแบบไร้สาย (Wireless Sensor Network) เป็นเครือข่ายไร้สายที่ประกอบด้วยอุปกรณ์ที่ทำงานอย่างเป็นอิสระ โดยใช้ตัวตรวจจับทำงานร่วมกันในการเฝ้าสังเกตการณ์สภาพแวดล้อม เช่น อุณหภูมิ เสียง การสั่นสะเทือน ความกดดัน การเคลื่อนไหว หรือของเสีย ในสถานที่แตกต่างกัน แล้วส่งข้อมูลไปยังสถานีฐานซึ่งทำหน้าที่เป็นประตูทางออกไปยังผู้ใช้งาน ผ่านทางโหนดต่างๆ (หรือที่เรียกว่าโมท (Mote)) ในเครือข่าย โดยโหนดทำงานภายใต้สภาพแวดล้อมที่จำกัด อาทิ พลังงาน หน่วยความจำ แบนด์วิดธ์ (bandwidth) และความสามารถในการติดต่อสื่อสาร

ระบบปฏิบัติการ TinyOS เป็นระบบปฏิบัติการประเภทโอเพนซอร์ส (Open source) สำหรับตัวตรวจจับแบบไร้สายที่ผู้ผลิตส่วนใหญ่เลือกใช้ มีลักษณะเชิงคอมโพเนนท์ ทำงานภายใต้สภาพแวดล้อมที่ขับเคลื่อนด้วยเหตุการณ์ ซึ่งได้รับการออกแบบมาสำหรับระบบเครือข่ายแบบฝังตัว เนื่องจากข้อจำกัดด้านทรัพยากร อาทิ หน่วยความจำ หน่วยประมวลผล ที่จำกัดกว่าอุปกรณ์เครือข่ายอื่นๆ ทั่วไป และลักษณะการทำงานที่ใช้เป็นตัวรับรู้เหตุการณ์จากเซ็นเซอร์ตลอดเวลา นั่นคือ เมื่อมีเหตุการณ์เกิดขึ้นก็จะทำให้เกิดการทำงานอย่างหนึ่งอย่างใดของตัวตรวจจับนั่นเอง

ในรายงานฉบับนี้ได้นำเสนอการศึกษาและการทดลองการทำงานของตัวตรวจจับแบบระบบเครือข่ายตัวตรวจจับแบบไร้สายในกรณีการรับส่งข้อมูลทั่วไป และกรณีเกิดปัญหาในเส้นทางที่รับส่งข้อมูล ซึ่งแสดงให้เห็นว่าโปรโตคอลที่มีนั้นสามารถปรับเปลี่ยนเส้นทางในการรับส่งข้อมูลได้ แต่จากการสังเกตพบว่าการทำงานนั้นอาจไม่ได้เป็นไปตามมาตรฐานของโปรโตคอลที่อ้างอิงมา เช่น ZigBee Routing Protocol ที่เมื่อบริษัท Crossbow Technology นำมาใช้ในชุดพัฒนาอุปกรณ์สำหรับเครือข่ายตัวตรวจจับแบบไร้สาย อาจมีการดัดแปลงในบางส่วนไป ซึ่งจุดนี้จะต้องใช้เวลาในการศึกษาเพิ่มเติม เพื่อที่จะได้เข้าใจในการทำงานที่มากขึ้นและถูกต้องยิ่งขึ้นสืบไป

8.2 สรุปผลการพัฒนาระบบสังเกตการณ์ตรวจจับแบบไร้สาย

สามารถเฝ้าติดตามสภาพแวดล้อมภายในอาคารโดยการตรวจจับอุณหภูมิ ความสว่างของแสง พลังงานแบตเตอรี่ในอุปกรณ์ แล้วมีข้อความแจ้งเตือนเมื่อเกิดเหตุการณ์ที่เรากำหนดไว้ และสามารถเลือกดูข้อมูลย้อนหลังในรูปแบบกราฟและตาราง

8.3 ข้อเสนอแนะ

สามารถเพิ่มโมดูลที่ใช้ร่วมกับ ตัวตรวจจับ ซึ่งมี ฟังก์ชั่นในการ วัดค่าความเข้มของเสียง, วัดค่าทำความแร่งในหน่วยแกน x แกน y, ความชื้น ในเรื่องของการเพิ่มการ sensor monitoring การเพิ่มโมดูลในการใช้งาน GPS เพื่อทำ location mapping

บรรณานุกรม

Behrouz A. Forouzan and Sophia Chung Fegan. **Data Communications and Networking, 4/e.** Boston : McGraw-Hill, 2007

Crossbow Technology. 2007. **MoteConfig User's Manual.** [Online]. Available : http://www.xbow.com/Support/Support_pdf_files/MoteConfig_Users_Manual.pdf

Crossbow Technology. 2007. **MoteView User's Manual.** [Online]. Available : http://www.xbow.com/Support/Support_pdf_files/MoteView_Users_Manual.pdf

Crossbow Technology. 2007. **MoteWorks Getting Started Guide.** [Online]. Available : http://www.xbow.com/Support/Support_pdf_files/MoteWorks_Getting_Started_Guide.pdf

Crossbow Technology. 2007. **XServe User's Manual.** [Online]. Available : http://www.xbow.com/Support/Support_pdf_files/XServe_User_Manual.pdf

David Gay and et. al. **The nesC Language: A Holistic Approach to Networked Embedded Systems.** [Online]. Available : http://nescc.sourceforge.net/papers/nesc-pldi-2003.pdf

Ian D. Chakeres and Elizabeth M. Belding-Royer. "**AODV Routing Protocol Implementation Design.**" *Proceedings of the International Workshop on Wireless Ad Hoc Networking (WWAN)*, Tokyo, Japan, March 2004.

LAN-MAN Standards Committee of the IEEE Computer Society. 2006. **IEEE Standard for Information Technology— Telecommunications and Information Exchange between Systems— Local and Metropolitan Area Networks—Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).** [Online]. Available : http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf

Philip Levis. 2006. **TinyOS Programming.** [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf

Sinem Coleri Ergen. 2004. **ZigBee/IEEE 802.15.4 Summary.** [Online]. Available : http://pages.cs.wisc.edu/~suman/courses/838/papers/zigbee.pdf

ภาคผนวก

ภาคผนวก ก

ผก. 1 การพัฒนาโปรแกรมสำหรับตัวตรวจจับด้วยภาษา nesC

ในการพัฒนาโปรแกรมสำหรับตัวตรวจจับแบบไร้สายสำหรับอุปกรณ์ในชุด Wireless Sensor Network Classroom Kit จะพัฒนาโดยการใช้ภาษาโปรแกรม nesC สำหรับ nesC (Network embedded system C) เป็นภาษา C ชนิดหนึ่งที่ได้รับการปรับปรุงให้มีลักษณะเป็นคอมโพเนนท์ (Component-based) เพื่อการพัฒนาระบบเครือข่ายแบบฝังตัว (Networked embedded systems) โดยการทำงานจะทำงานบนระบบปฏิบัติการ TinyOS ซึ่งกล่าวไว้แล้วในบทที่ผ่านมา

สำหรับเนื้อหาในบทนี้ จะกล่าวถึงหลักการพัฒนาโปรแกรมสำหรับตัวตรวจจับ โดยใช้ภาษา nesC และโปรแกรมที่ได้รับการพัฒนาขึ้นเพื่อใช้ในการทดลองในครั้งนี้

1. ภาษาการโปรแกรม nesC

สำหรับภาษา nesC เป็นภาษา C ชนิดหนึ่งที่ได้รับการปรับปรุงเพื่อการพัฒนาระบบเครือข่ายแบบฝังตัว อาทิ ระบบเครือข่ายตรวจจับแบบไร้สาย โดยได้เพิ่มเติมแนวคิดของคอมโพเนนท์ ความเหมาะสมในการทำงานภายในสภาพแวดล้อมที่มีทรัพยากร อาทิ พลังงาน หน่วยความจำ อยู่จำกัด และการรองรับการทำงานภายใต้ระบบปฏิบัติการ TinyOS

2. การกำหนดรายละเอียดให้กับคอมโพเนนท์ (Component Specification)

ในแต่ละคอมโพเนนท์จะประกอบไปด้วยสองส่วน คือ ฟังก์ชั่นที่ได้จัดเตรียมให้ (provides) และฟังก์ชั่นที่เรียกใช้งาน (uses) มาเชื่อมต่อกัน (wiring) สำหรับการรวมกลุ่มของฟังก์ชั่น สามารถทำได้โดยการใช้อินเตอร์เฟส (คีย์เวิร์ด interface) โดยมักนำไปใช้ในการรวมกลุ่มของการบริการอย่างหนึ่งอย่างใด อาทิ การคำนวณเวลา การส่งข้อมูลจากตัวตรวจจับ ดังตัวอย่างที่ 1 เป็นต้น

ตัวอย่างที่ 1 การกำหนดโครงสร้างอินเตอร์เฟส

```
interface Timer {
  command result_t start(char type, uint32_t interval);
  command result_t stop();
  event result_t fired();
}
interface Send {
  command result_t send(TOS_Msg *msg, uint16_t length);
  event result_t sendDone(TOS_Msg *msg, result_t success);
}
```

อินเตอร์เฟสในภาษา nesC จะประกอบไปด้วยคำสั่ง (Command) และตัวรับเหตุการณ์ (Event) ดังตัวอย่างอินเตอร์เฟส `Timer` ที่แสดงอยู่ในรูปที่ 4.1 ได้กำหนดคำสั่งชื่อ `start` และ `stop` และตัวรับเหตุการณ์ชื่อ `fired` นอกจากนั้นอินเตอร์เฟสยังสามารถรับตัวแปร (Parameter) เข้าไปได้ การกำหนดอินเตอร์เฟสนั้นจะช่วยให้นำคอมโพเนนท์กลับมาใช้งานใหม่ได้ และมีความยืดหยุ่นต่อการเปลี่ยนแปลง

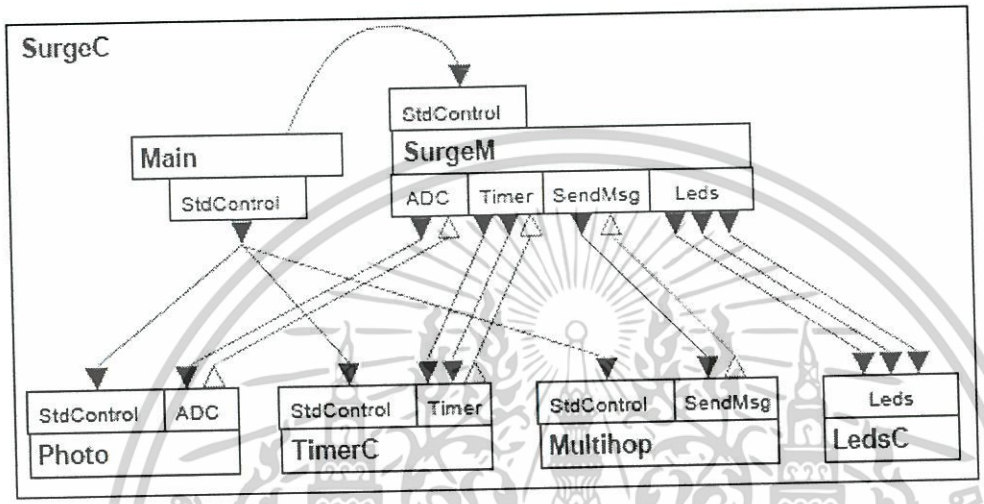
3. การพัฒนาคอมโพเนนท์ (Component Implementation)

คอมโพเนนท์ในภาษา nesC มี 2 ประเภท คือ โมดูล (Module) และคอนฟิกูเรชั่น (Configuration) สำหรับโมดูลจะเป็นส่วนของการจัดเตรียมโค้ดโปรแกรม ไม่สามารถสร้างออป-เจ็คจากโมดูลได้ (เนื่องจากโมดูลมีลักษณะเป็นซิงเกิลตัน (singleton)) ส่วนคอนฟิกูเรชั่นจำใช้ในการเชื่อมต่อกับคอมโพเนนท์และอินเตอร์เฟสอื่น ดังนั้นคอมโพเนนท์ประเภทโมดูลจึงใช้ในการพัฒนาคอมโพเนนท์ของส่วนอุปกรณ์ (hardware) ก็เพราะผู้เรียกใช้จะไม่สามารถสร้างออปเจ็คได้อย่างอิสระ ดังแสดงในตัวอย่างที่ 2 สำหรับรูปนี้เป็นส่วนหนึ่งของโค้ดโปรแกรม `SurgeM` โดยได้กำหนดคำสั่ง StdControl.init ซึ่งจะเรียกในช่วงการเริ่มต้น โดยทำการเรียก (คียเวิร์ด call) การเริ่มต้นการจับเวลา (Timer.start) และตัวจัดการเหตุการณ์จับเวลา (Timer.fired) โดยทำการเรียกข้อมูลจากตัวตรวจจับ (ADC.getData) และการดึงข้อมูลจากตัวตรวจจับ (ADC.dataReady)

ตัวอย่างที่ 2 การพัฒนาคอมโพเนนท์

```
module SurgeM {
  provides interface StdControl;
  uses interface ADC;
  uses interface Timer;
  uses interface Send;
}
implementation {
  uint16_t sensorReading;
  command result_t StdControl.init() {
    return call Timer.start(TIMER_REPEAT, 1000);
  }
  event result_t Timer.fired() {
    call ADC.getData();
    return SUCCESS;
  }
  event result_t ADC.dataReady(uint16_t data) {
    sensorReading = data;
    ... send message with data in it ...
    return SUCCESS;
  }
  ...
}
```

สำหรับโปรแกรมที่พัฒนาโดยภาษา nesC จะได้รับการอธิบายโดย Top-level configuration ซึ่งเป็นตัวเชื่อมต่อคอมโพเนทท์ที่ใช้งานทั้งหมดเข้าด้วยกัน ดังแสดงในรูปที่ 1 ซึ่งการเรียกนั้นก็มีอยู่ 2 ลักษณะด้วยกัน คือ หลายๆ สิ่งเรียกใช้สิ่งเดียวกัน (“fan-in”) และสิ่งหนึ่งเรียกใช้หลายสิ่ง (“fan-out”) อย่างตัวอย่างคอมโพเนนท์ SurgeC ดังแสดงในรูปที่ 1 StdControl.init ในMain ได้เรียกใช้ถึง 4 ส่วนด้วยกัน คือ SurgeM, Photo, TimerC และ Multihop



รูปที่ ผก. 1 คอนฟิกูเรชั่น SurgeC

คอมโพเนนท์ส่วนใหญ่ใน TinyOS มักมีเพียงออปเจ็คเดียว แต่ในบางครั้งก็อาจต้องสร้างหลายออปเจ็คขึ้นใน ซึ่งในภาษา nesC อาจสร้าง Abstract component ซึ่งอาจมีตัวแปรด้วยก็ได้ สำหรับคอมโพเนนท์ประเภทนี้จะถูกสร้างขึ้นในช่วงเวลาคอมไพล์ ดังแสดงในรูปที่ผก. 1

ตัวอย่างที่ 3 การสร้างและใช้ Abstract component

```
abstract module QueuedSend(int maxAttempts) { ... }
configuration Multihop {
  provides interface Send;
}
implementation {
  components MultihopM, QueuedSend(10) as newQueue, ... ;
  Send = MultihopM.Send;
  MultihopM.QueuedSendMsg -> newQueue.Send;
  ...
}
```

4. การทำงานในเวลาเดียวกันในภาษา nesC (Concurrency in nesC)

การทำงานในเวลาเดียวกัน (Concurrency) ถือได้ว่าเป็นส่วนสำคัญของ nesC โดยปกติเหตุการณ์หรือคำสั่งใดๆ อาจถูกเรียกโดยตรงหรือผ่านการขัดจังหวะ (interrupt) ในการที่จะจัดการการทำงานในเวลาเดียวกันนี้ nesC จะต้องใช้ Atomic sections (บล็อกโค้ด atomic) และ Tasks (คีย์เวิร์ด task) ดังแสดงในตัวอย่างที่ 4 จากตัวอย่างนี้ Timer.fired และ ADC.data Ready เป็นอะซิงโครนัสโค้ด

ตัวอย่างที่ 4 การใช้ Atomic sections และ Tasks ใน SurgeM

```
module SurgeM { ... }
implementation {
  bool busy;
  norace uint16_t sensorReading;
  event result_t Timer.fired() {
    bool localBusy;
    atomic {
      localBusy = busy;
      busy = TRUE;
    } if (!localBusy)
        call ADC.getData();
    return SUCCESS;
  }
  task void sendData() { // send sensorReading
    adcPacket.data = sensorReading;
    call Send.send(&adcPacket, sizeof adcPacket.data);
    return SUCCESS;
  }
  event result_t ADC.dataReady(uint16_t data) {
    sensorReading = data;
    post sendData();
    return SUCCESS;
  }
}
```

จากตัวอย่างที่ผ่านมา ยังมีบางกรณีที่มีโอกาสเกิดไม่ความเสถียรในการปรับปรุงข้อมูล (race-condition) ตัวอย่างเช่นตัวแปร sensorReading โดยตัวแปรดังกล่าวมีการระบุคีย์เวิร์ด norace เพื่อระงับข้อผิดพลาดเกี่ยวกับไม่ความเสถียรในการปรับปรุงข้อมูล หรือหลีกเลี่ยงการป้องกันการเข้าถึงตัวแปร sensorReading ทั้งหมดโดยใช้บล็อกโค้ด atomic ก็ได้

5. อินเตอร์เฟสที่มีการรับตัวแปร (Parameterized Interfaces)

ใน TinyOS อินเตอร์เฟสที่มีการรับค่าตัวแปรจะถูกใช้เพื่อสร้าง Active messages สำหรับ Active messages นั้นตัวแปรจะมีความหมายถึงตัวเลขในการระบุว่าตัวจัดการใดจะถูกเรียกใช้ ดังแสดงในตัวอย่างที่ 5

ตัวอย่างที่ 5 แสดง Active Messages ที่อินเตอร์เฟสมีการรับค่าตัวแปร

```
module GenericComm { // id is the Active Message ID
  provides interface Send[uint8_t id];
  provides interface Receive[uint8_t id];
}
implementation {
  TOS_Msg *msg;
  command result_t
    Send.send[uint8_t id](uint8_t length, TOS_Msg *data)
  { data->amId = id; msg = data; ... }
  void sendComplete(TOS_Msg *packet) {
    signal Send.sendDone[msg->amId](msg, SUCCESS);
  }
  ...
}
configuration Multihop { ... }
implementation {
  components QueuedSend(10) as newQueue, GenericComm,
  ...
  newQueue.RealSend -> GenericComm.Send[42];
}
```

ภาคผนวก ข

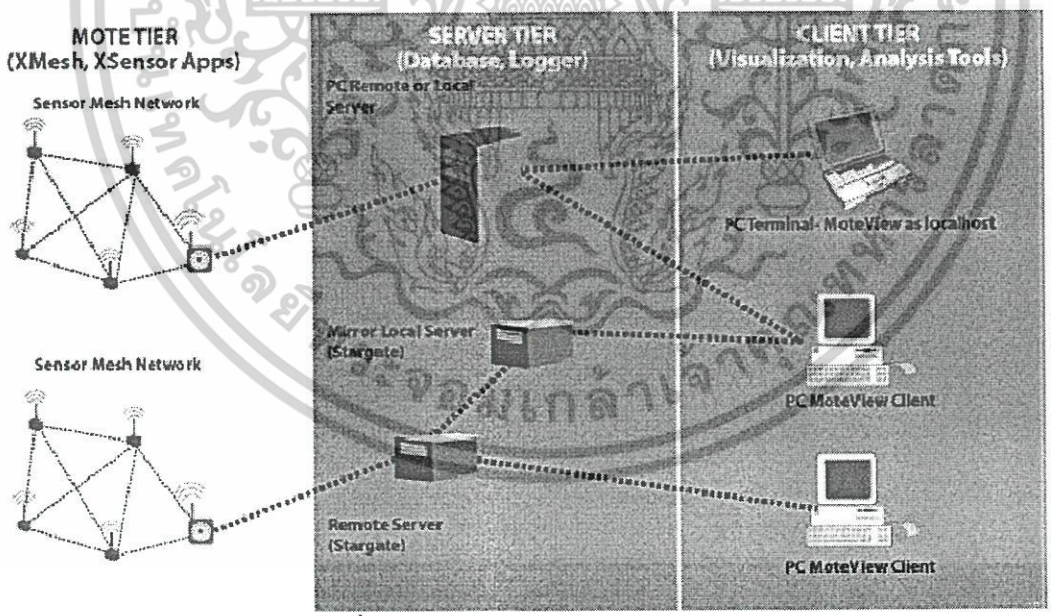
ผข. 1 การติดตั้งโปรแกรม MoteWorks 2.0 และ MoteView 2.0

โปรแกรม MoteWorks 2.0 และ MoteView 2.0 เป็นโปรแกรมที่มาพร้อมกับชุด Wireless Sensor Network Classroom Kit ได้รับการพัฒนาโดยบริษัท Crossbow Technology จำกัด สำหรับแต่ละโปรแกรมมีรายละเอียดดังนี้

1. MoteWorks 2.0

MoteWorks เป็นแพลตฟอร์มใช้ในการสร้างและควบคุม ตรวจตราการทำงานของระบบเครือข่ายตรวจจับแบบไร้สาย ทั้งในส่วนฮาร์ดแวร์และซอฟท์แวร์ หากมองในมุมมองโครงร่างของเครือข่าย ดังแสดงในรูปที่ 1 จะสามารถแบ่งออกเป็น 3 ส่วน ดังนี้

- **Mote Tier** ในส่วนนี้โปรแกรม XMesh จะทำหน้าที่ควบคุมการไหลของข้อมูลของทุกโหนดในเครือข่ายไปยังเครื่องแม่ข่าย
- **Server Tier** เป็นส่วนที่จัดการ แปลงและบัฟเฟอร์ข้อมูลจากเครือข่ายตรวจจับแบบไร้สาย โดยผ่านโปรแกรม XServe
- **Client Tier** เป็นส่วนที่แสดงผลจากข้อมูลที่ได้มาจากเครือข่ายตรวจจับแบบไร้สาย โดยผ่านโปรแกรม MoteView



รูปที่ ผข. 1 โครงร่างของ XMesh

สำหรับโปรแกรม MoteWorks 2.0 สามารถแบ่งออกเป็น 2 ส่วนดังนี้

1.1 Development Environment

- Cygwin เป็นโปรแกรมจำลองสภาพแวดล้อมของระบบปฏิบัติการลีนุกซ์
- MoteConfig 2.0 และ OTAP เป็นโปรแกรมในการกำหนดค่าและติดตั้งโค้ดโปรแกรมที่ได้รวมกับ XMesh/TinyOS เป็นเฟิร์มแวร์ลงในโหนด
- XSniffer เป็นโปรแกรมในการดูข้อมูลดิบของเฟรมที่รับส่งในเครือข่าย
- Programmer's Notepad 2 เป็นสภาพแวดล้อมและเครื่องมือที่ช่วยในการพัฒนาโปรแกรม (IDE) ใช้สร้าง แก้ไขและคอมไพล์ไฟล์โปรแกรม
- GraphViz 2.6 เป็นเครื่องมือในการเขียนกราฟ
- TortoiseCVS 1.8.22 เป็นโปรแกรมที่ช่วยการจัดการเวอร์ชั่นโค้ดโปรแกรมสำหรับทีมพัฒนาโปรแกรม
- PuTTY Utilities เป็นโปรแกรมที่ใช้ในการเข้าถึงโหนดจากระยะไกล เช่น ผ่าน Telnet, SSH
- WinMerge 2.4.6.0 เป็นเครื่องมือในการแบ่งแยกความแตกต่างระหว่างไฟล์และรวมไฟล์บนระบบปฏิบัติการวินโดว์

1.2 Compilers

- nesC 1.2 เป็นคอมไพเลอร์ของภาษา nesC ซึ่งเป็นภาษาที่ใช้พัฒนาโปรแกรมสำหรับโหนด
- คอมไพเลอร์ AVR-GCC สำหรับไมโครคอนโทรลเลอร์ Atmel Atmega128
- ตัวตรวจหาข้อบกพร่อง GDB และ AVR Insight

2. MoteView 2.0

MoteView เป็นส่วนที่ติดต่อระหว่างผู้ใช้งานกับเครือข่ายตรวจจับแบบไร้สาย โดยโปรแกรมดังกล่าวได้จัดเตรียมเครื่องมือในการติดตั้งและตรวจตราโหนดในเครือข่าย นอกจากนั้นยังช่วยในการเชื่อมต่อกับฐานข้อมูล วิเคราะห์ และแสดงผลข้อมูลที่ตัวตรวจจับอ่านได้ในรูปกราฟ

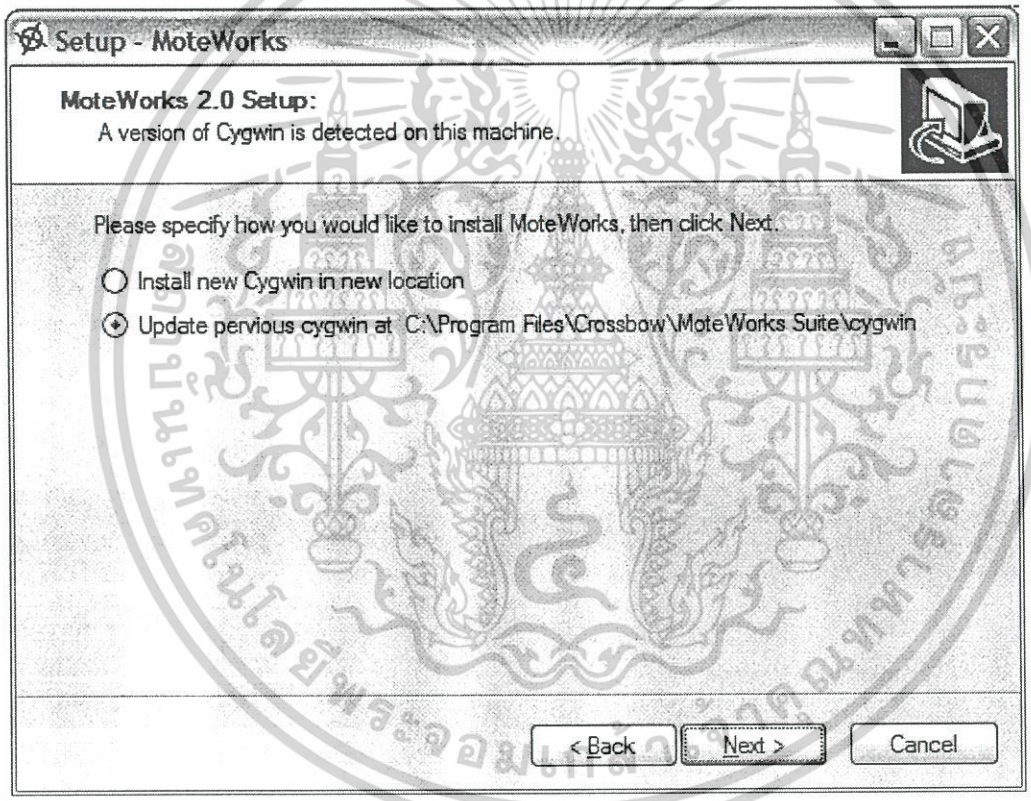
สำหรับรายละเอียดของการติดตั้งแต่ละโปรแกรมมีดังนี้

2.1 ระบบปฏิบัติการที่รองรับ

ระบบปฏิบัติการที่รองรับของ MoteWorks 2.0 และ MoteView 2.0 ได้แก่ระบบปฏิบัติการ Windows XP, Windows Server 2003 (ในที่นี้ติดตั้งบน Windows XP Professional with SP2)

3. การติดตั้ง MoteWorks 2.0

1. ดับเบิ้ลคลิกไฟล์ MoteWorks_2.0.F_Setup.exe ในโฟลเดอร์ *<CD_DRIVE>:/SeniorProject/Softwares/*
2. ในหน้า Welcome to the MoteWorks setup Wizard ให้คลิก Next
3. หากตัวติดตั้งพบโปรแกรม Cygwin ในคอมพิวเตอร์เครื่องที่ทำการติดตั้งอยู่ก่อนแล้ว จะมีหน้าต่างดังแสดงในรูปที่ 2 หากต้องการติดตั้งโปรแกรม Cygwin ที่มากับตัวติดตั้งนี้ในตำแหน่งใหม่ให้เลือก "Install new Cygwin in new location" แต่หากต้องการติดตั้งทับไฟล์โปรแกรมเดิมที่มีอยู่แล้ว ให้เลือก "Update previous cygwin ..." แล้วให้คลิก Next



รูปที่ ผข. 2 แสดงหน้าต่างถามผู้ติดตั้งให้เลือกระหว่างติดตั้ง Cygwin ในตำแหน่งใหม่หรือติดตั้งทับไฟล์ของ Cygwin เดิมที่มีอยู่

4. ในหน้า License Agreement ให้เลือก I accept the agreement แล้วคลิก Next
5. ในหน้า Select Destination Location ให้ระบุตำแหน่งที่จัดเก็บไฟล์โปรแกรม โดยตำแหน่งนั้นไม่ควรมีช่องว่างระหว่างชื่อ เช่น Documents and Settings เป็นต้น แล้วคลิก Next

6. ในหน้า Select MoteWorks Components ให้เลือก Full installation แล้วคลิก Next
7. ในหน้า Ready to Install ให้คลิก Install เพื่อทำการติดตั้งโปรแกรม
8. รอจนกว่ากระบวนการติดตั้งจะเสร็จสิ้น หากมีหน้าต่างตัวติดตั้งอื่นขึ้นมาให้ทำการติดตั้งส่วนประกอบนั้น
9. ในหน้า Completing the MoteWorks Setup Wizard ให้คลิก Finish เพื่อเสร็จสิ้นกระบวนการติดตั้ง

4. การติดตั้ง MoteView 2.0

1. ดับเบิ้ลคลิกไฟล์ MoteView_2.0.F_Setup.exe ในโฟลเดอร์ *<CD_DRIVE>:/SeniorProject/Softwares/*
2. ในหน้า Select Destination Location ให้ระบุตำแหน่งที่จัดเก็บไฟล์โปรแกรม โดยตำแหน่งนั้นไม่ควรมีช่องว่างระหว่างชื่อ เช่น Documents and Settings เป็นต้น แล้วคลิก Next
3. ในหน้า Select Start Menu Folder ให้ระบุตำแหน่งที่จัดเก็บไอคอน แล้วคลิก Next
4. ในหน้า Select Additional Tasks ให้คลิก Next
5. ในหน้า Ready to Install ให้คลิก Install เพื่อทำการติดตั้งโปรแกรม
6. รอจนกว่ากระบวนการติดตั้งจะเสร็จสิ้น หากมีหน้าต่างตัวติดตั้งอื่นขึ้นมาให้ทำการติดตั้งส่วนประกอบนั้น
7. ในหน้า Completing the MoteView Setup Wizard ให้คลิก Finish เพื่อเสร็จสิ้นกระบวนการติดตั้ง

5. การติดตั้ง MIB520 USB Gateway

สำหรับอุปกรณ์เกตเวย์รุ่น MIB 520 นั้นเชื่อมต่อกับคอมพิวเตอร์ผ่านพอร์ต USB โดยเสมือนการเชื่อมต่อผ่านพอร์ต COM (Virtual COM Port) จึงต้องติดตั้งไดร์เวอร์เพิ่มเติมสำหรับอุปกรณ์ดังกล่าว โดยการติดตั้งมีขั้นตอนดังนี้

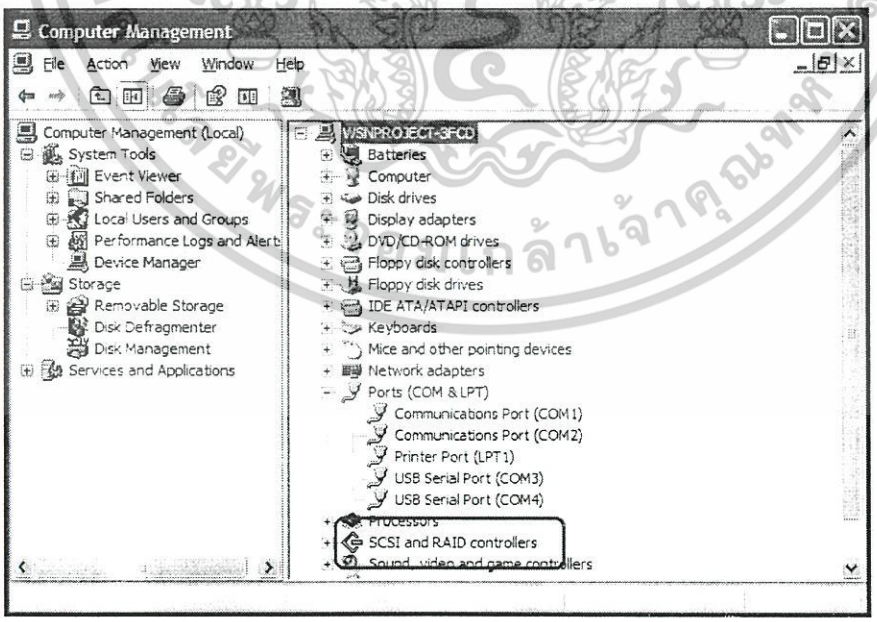
ข้อควรระวัง หาก MIB520 ต่อกับหน่วยประมวลผลและแผงควบคุมการทำงานของการรับส่งคลื่นวิทยุ (Processor / Radio Boards) ควรถอดถ่านออกก่อนเสียบ MIB520 เข้ากับคอมพิวเตอร์

1. เสียบสาย USB เข้ากับคอมพิวเตอร์และ MIB520 ระบบปฏิบัติการจะเรียกหาไดร์เวอร์ของอุปกรณ์ หากไม่พบจะขึ้นหน้าต่าง Found New Hardware Wizard ขึ้นมา

2. ในหน้า Welcome to The New Hardware Wizard เลือกตัวเลือก No, not this time แล้วคลิกปุ่ม Next
3. ในหน้า What do you want the wizard to do? เลือกตัวเลือก Install from a list or specific location (Advanced) แล้วคลิกปุ่ม Next
4. ในหน้า Please choose your search and installation options ให้เลือก Search for the best driver in these locations เอาเครื่องหมายหน้าตัวเลือก Search removable media (floppy, CD-ROM ...) ออก และทำเครื่องหมายหน้าตัวเลือก Include this location in this search จากนั้นให้ Browse ไปยังไดเร็กทอรี่ *<CD_DRIVE>:/SeniorProject/Softwares/USB Drivers* แล้วคลิก Next
5. ทำขั้นตอนที่ 1 ถึง 4 ซ้ำ จนกว่าการติดตั้งไดร์เวอร์จะเสร็จสิ้นสมบูรณ์

เมื่อติดตั้งเสร็จสิ้นจะมี Virtual COM Port จำนวน 2 พอร์ต โดยพอร์ตที่ x จะใช้สำหรับการโปรแกรม และพอร์ตที่ x+1 จะใช้สำหรับการติดต่อสื่อสารกับโหนด หากต้องการดูว่าขณะนี้เป็นหมายเลขพอร์ตอะไรบ้าง ให้ทำดังนี้

1. คลิกขวาที่ My Computer แล้วเลือกเมนู Manage หน้าต่าง Computer Management จะปรากฏขึ้น
2. คลิก Device Manager ภายใต้ System Tools แล้วสังเกตหมายเลขพอร์ตในหัวข้อ Ports (COM & LPT) ดังแสดงในรูปที่ผข. 3



รูปที่ ผข. 3 แสดงหมายเลขพอร์ตที่เชื่อมต่อกับ MIB520 USB Gateway

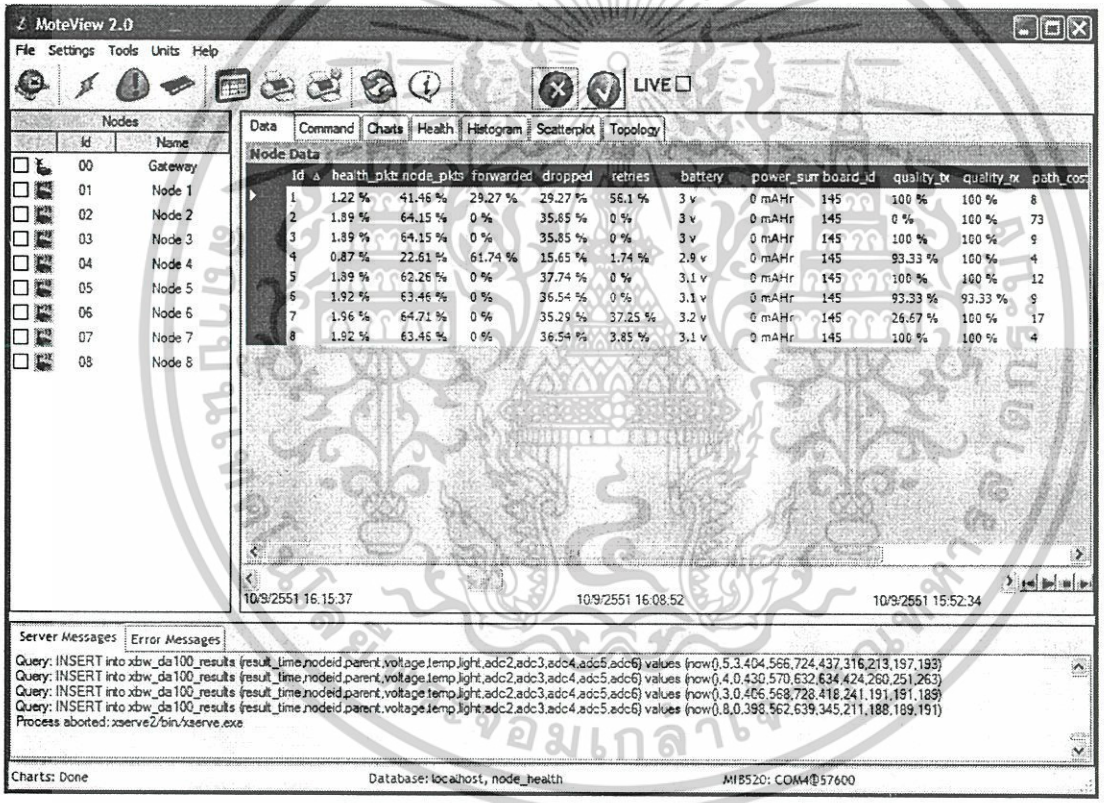
ภาคผนวก ค

ผค. 1 การใช้งานโปรแกรม MoteView

โปรแกรม MoteView 2.0 เป็นโปรแกรมที่มาพร้อมกับชุด Wireless Sensor Network Classroom Kit ได้รับการพัฒนาโดยบริษัท Crossbow Technology จำกัด โดย MoteView เป็นส่วนที่ติดต่อระหว่างผู้ใช้งานกับเครือข่ายตรวจจับแบบไร้สาย ซึ่งดังกล่าวได้จัดเตรียมเครื่องมือในการติดตั้งและตรวจตราโหนดในเครือข่าย นอกจากนั้นยังช่วยในการเชื่อมต่อกับฐานข้อมูล วิเคราะห์และแสดงผลข้อมูลที่ตัวตรวจจับอ่านได้ในรูปกราฟ

1. การเปิดใช้งานโปรแกรม

1. ให้ไปที่ Start | All Programs | Crossbow | MoteView 2.0F หน้าต่างโปรแกรมจะปรากฏขึ้นดังแสดงในรูปที่ 1



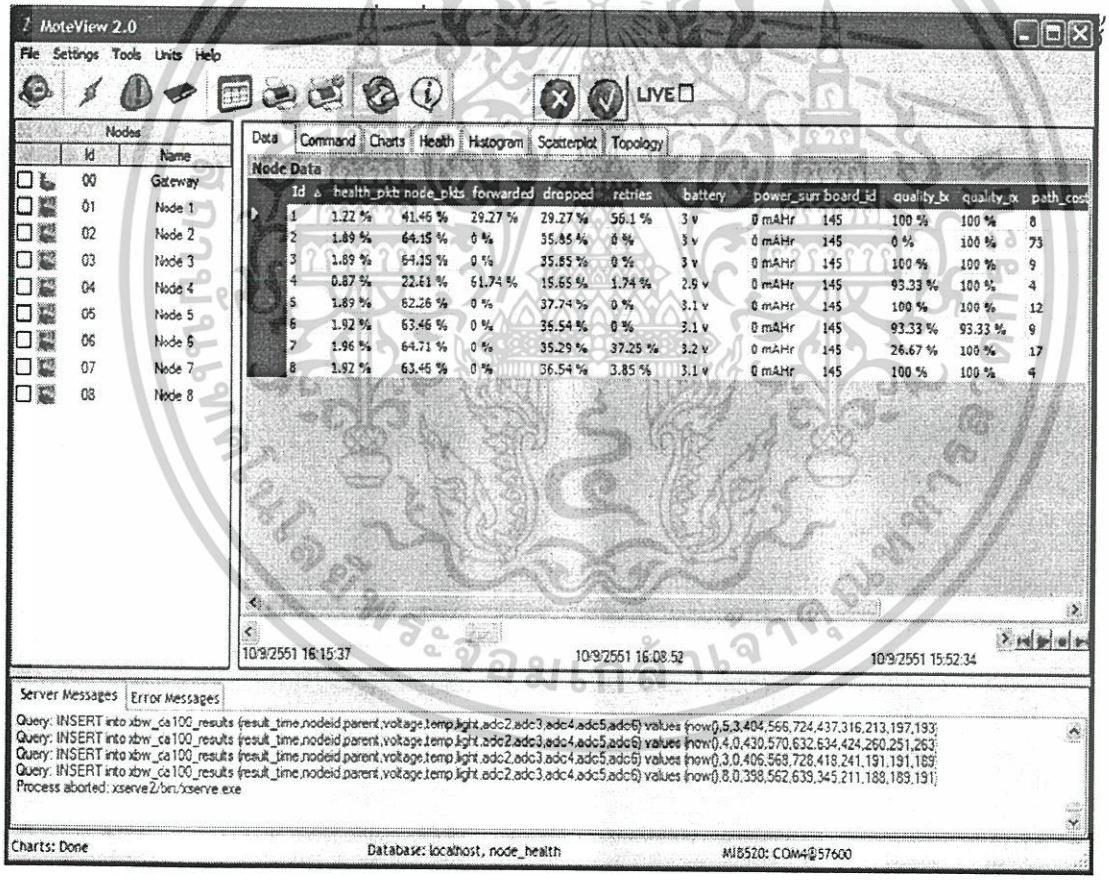
รูปที่ ผค. 1 หน้าต่าง MoteView

2. การเชื่อมต่อกับเครือข่ายตัวตรวจจับแบบไร้สาย

1. จากหน้าต่าง MoteView ให้คลิกปุ่ม  หน้าต่าง Connect to WSN จะปรากฏขึ้น

2. ในแท็ป Mote ให้เลือก Acquire Live Data ในหัวข้อ 1.Select Operation Mode และเลือก Custom ในหัวข้อ 2.Select Acquisition Type เมื่อเสร็จแล้วคลิก Next
3. ในแท็ป Gateway ให้เลือกในช่อง InterfaceBoard, Hostname/IP, Port ในหัวข้อ 3.Select Gateway ตามอุปกรณ์เกตเวย์ที่ใช้งาน เมื่อเสร็จแล้วคลิก Next

   <u>หมายเหตุ</u> หากเลือก Interface Board เป็น MIB520 ให้เลือก Port เป็นพอร์ตที่ x+1 โดยหมายเลขพอร์ตนั้นสามารถดูได้ Device Manager ใน Computer Management
4. ในแท็ป Database หากมีการเปลี่ยนแปลงค่าในการเชื่อมต่อกับฐานข้อมูล PostgreSQL ให้คลิก Edit หากไม่ได้แก้ไขค่าที่ตั้งไว้ให้คลิก Next
5. ในแท็ป Sensor ให้ทำเครื่องหมายหน้า View Alternate Table แล้วเลือกตารางที่สัมพันธ์กับข้อมูลที่จะส่งมาจากโหนดภายในเครือข่าย เมื่อเสร็จแล้วคลิก Finish



รูปที่ ผค. 2 แสดงการทำงานของ MoteView

7. หากต้องการหยุดต้องกด