แบบรายงานโครงการวิจัย
โดยใช้เงินรายได้คณะวิศวกรรมศาสตร์
ประจำปี 2555

ชื่อโครงการ

เครื่องวัดสิ่งแวดล้อมระยะไกลอัจฉริยะ
(Intelligent long rang environment measurer machine)

ผู้รับผิดชอบโครงการ

หัวหน้าโครงการวิจัย รศ.ดร.อรรถสิทธิ์ หล่าสกุล

หน่วยงาน

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

พระจอมเกล้าเจ้าคุณทหารลาดกระบัง

แบบรายงานโครงการวิจัย
โดยใช้เงินรายได้คณะวิศวกรรมศาสตร์
ประจำปี 2555

ชื่อโครงการ

เครื่องวัดสิ่งแวดล้อมระยะไกลอัจฉริยะ

(Intelligent long rang environment measurer machine)

ผู้รับผิดชอบโครงการ

หัวหน้าโครงการวิจัย รศ.ดร.อรรถสิทธิ์ หล่าสกุล

หน่วยงาน

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

พระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทคัดย่อ

วิธีการควบคุมอุปกรณ์ระยะไกลผ่านการสื่อสารแบบไร้สายนับได้ว่ามีประโยชน์ต่อการนำใช้มางานกันเป็นอย่างมากในหลายประเทศในปัจจุบันได้มีการคิดทั้งอุปกรณ์และโปรแกรมสำหรับตัวอุปกรณ์นั้นออกมาอย่างต่อเนื่อง อุปกรณ์ไร้สายที่เป็นที่รู้จักและมีชื่อเสียงมาก ตัวอย่าง เช่น Xbee, ZigBee [1][2][6][7][8] เหล่านี้เป็นต้น ซึ่งอุปกรณ์เหล่านี้ส่วนมากก็จะใช้รับส่งข้อมูลกันในระยะใกล้สามารถต่อเชื่อมกันหลายตัว ราคาต่อตัวก็ไม่สูงนัก ประกอบกับมีซอฟแวร์ที่ออกแบบไว้เฉพาะให้ใช้งานได้อย่างมีประสิทธิภาพ ทำให้ผู้ใช้สามารถนำมาประยุกต์ใช้งานได้ทันที แต่อุปกรณ์นั้นหากนำมาใช้งานในการส่งระยะทางไกลๆ ก็จะมีรุ่นที่ผลิตขึ้นมาเฉพาะใช้งานในระยะไกลเช่นกัน แต่ก็จะมีราคาต่อตัวสูงขึ้นมากซึ่งในปกติของการรับส่งข้อมูลระยะไกลๆ ที่อยู่นอกเหนือเครือข่ายของระบบโทรศัพท์ปัจจุบันหรือที่เป็นรังผึ้ง (GSM) แล้วนั้น ที่เป็นไปได้อย่างเดียวที่สามารถใช้งานได้ก็คือการใช้ระบบของวิทยุสื่อสาร เพื่อการใช้งานจริงบนระยะทางที่ห่างของแต่ละตัววัดเซนเซอร์ที่ห่างไกลกันมากๆ ดังนั้นในงานวิจัยนี้จึงนำเสนอการออกแบบซอร์ฟแวร์ โดยจะทดลองทดสอบการทำงานกับอุปกรณ์รับส่งระยะใกล้ก่อน แทนการใช้งานกับระบบวิทยุจริง (ซึ่งต้องใช้ค่าใช้จ่ายสูงมากๆ) โครงงานวิจัยจะนำเสนอการออกแบบโปรแกรม ที่สามารถนำมาใช้ในการส่งแบบจุดต่อจุดที่มีประสิทธิภาพ สามารถใช้งานกับระบบการสื่อสารแบบวิทยุโดย อุปกรณ์ที่ใช้ทดสอบจะเป็น Xbee/ZigBee (โดยตั้งโหมดการทำงานแบบกระจายสัญญาณเหมือนวิทยุสื่อสาร) คุณสมบัติเด่นของโปรแกรมนี้ก็คือ จะสามารถกวาดหาเส้นทางหลักและเส้นสำรองในการสื่อสารได้ทั้งไปและกลับ กรณีเส้นทางหลักมีปัญหาสามารถระบุถึงจุดที่มีปัญหาได้ ซึ่งจะทำให้การนำไปประยุกต์ใช้ได้รับความรวดเร็ว และแม่นยำในการใช้งานขจัดจุดอ่อนของการใช้งานการับส่งข้อมูลผ่านระบบวิทยุสื่อสารที่มีมานาน ทำให้ระบบมีความน่าเชื่อถือมากขึ้นนั่นเอง

ABSTRACT

This research proposed programming software for remote control based wireless communications technology (Walky Talky). Because of, in the last research (in 2554) name “Long Rang Eviroment Monitoring via Radio System” had some backdraw of data-unstable. In order to overcome that problem, the new idea for software control is proposed. In this research, command/data can be sent remotely by using Xbee instead of radio communication (for experimental). All of measurer modul can be act as repeater station, scanning to decise the path that command/data should be continue sent to the desire measurer module. This make communication system has more stable then before. Software was implemented on microcontroller (dsPIC30F4011) [3] with Xbee module interfacing (setting to operate as radio broadscat mode). So this software can be applied to the real radio system in the future. For using, just as the last research, any data obtained by all measurer modules can be monitor by user on general personal computer easily.

สารบัญ

หัวข้อ	หน้า

สารบัญตารางและสารบัญรูป

ชื่อตาราง **หน้า**

บทที่ 1
บทนำ

สืบเนื่องมาจาก การที่ผู้วิจัย ได้นำเสนองานวิจัยเพื่อขอทุนสนับสนุนจากเงินรายได้คณะวิศวฯ ในปี 2552 เรื่อง "เครื่องวัดระยะไกลผ่านวิทยุสื่อสาร" ซึ่งเป็นงานวิจัยสร้างเครื่องวัดอุณหภูมิระยะไกลโดยใช้การสื่อสารทางวิทยุสื่อสาร โดยนำเสนอการสร้างอุปกรณ์ตรวจวัดสภาพสิ่งแวดล้อม (ในงานวิจัยเลือกการวัดอุณหภูมิเป็นตัวอย่าง) ที่มีขนาดเล็กที่เรียกว่าเป็นตัวลูก (Client) จำนวนหลายๆตัวที่สามารถนำไปติดตั้งในพื้นที่ห่างไกลหลายๆตำแหน่งที่ต้องการวัด โดยการใช้งานคือผู้ใช้เองสามารถใช้คอมพิวเตอร์ส่วนบุคคลทั่วไปที่ติดตั้งโปรแกรมเฉพาะที่ทำขึ้น ทำการต่อเชื่อมกับเครื่องวัดตัวแม่ (Main) ของตัวเอง สั่งงานเพื่อวัดค่าจากตัวลูกที่ต้องการได้ทันที งานวิจัยแรกนี้มีจุดเด่นคือสามารถทำงานได้ในพื้นที่ ที่สัญญาณโทรศัพท์เข้าไปไม่ถึง เช่น พื้นที่ภูเขาได้ เพราะในงานวิจัยได้ใช้วิทยุสื่อสารเป็นสื่อในการส่งคำสั่งหรือรับข้อมูลจากตัวลูก ต่อมาในปี 54 ผู้วิจัยได้ทำการพัฒนาต่อเนื่องกับงานวิจัยนี้อีกครั้ง โดยนำเสนอรูปแบบของการรับและส่งข้อมูล อยู่บนคลื่นวิทยุสื่อสารเช่นเดิม หากแต่เพิ่มความสามารถให้สามารถรับส่งได้ไกลมากขึ้นโดยไม่จำเป็นต้องเพิ่มกำลังส่งของเครื่องวิทยุ ซึ่งใช้วิธีกำหนดให้ตัวลูกแต่ละตัวสามารถทำตัวเองเป็นตัวทวนสัญญาณเพื่อส่งหรือรับสัญญาณต่อกันเป็นทอดๆไปได้ ดังนั้นถึงแม้ตัวแม่จะมีกำลังเครื่องส่งสัญญาณที่แรงไม่พอถึงตัวลูกที่ต้องการติดต่อก็สามารถทำงานได้ โดยการพัฒนานี้ได้เสนอขอทุนวิจัยจาก สำนักคณะกรรมการวิจยแห่งชาติของปีงบประมาณ 2554 ซึ่งงานวิจัยนี้ได้ดำเนินการกว่า 70-80% แล้ว กำหนดส่งตามหมายกำหนดการคือเดือน กันยายน พ.ศ.2554 นี้ และจากการวิจัยทั้งสองเรื่องต่อเนื่องกันที่ผ่านมาทำให้ผู้วิจัยทราบถึงจุดที่ควรจะพัฒนาต่อเนื่องต่อไปอีกให้ระบบมีประสิทธิภาพมากขึ้นไปอีกขั้น เพราะปัญหาที่พบส่วนหนึ่งคือในการติดต่อด้วยวิทยุสื่อสารนั้น คลื่นความถี่เหล่านี้จะเป็นคลื่นที่มีโอกาศถูกรบกวนได้ง่ายเพราะเป็นคลื่นที่บุคคลอื่นก็สามารถใช้งานได้ทำให้ข้อมูลผิดพลาดได้ (เหตุสุดวิสัย) ซึ่งก็สามารถแก้ปัญหาได้ในระดับหนึ่งคือการส่งซ้ำจนกว่าจะได้ข้อมูลที่ถูกต้อง แต่ปัญหาที่สำคัญอีกอย่างก็คือหากในเส้นการรับส่งนั้น มีตัวลูกตัวใดเสียหายไม่ทำงาน ก็จะทำให้ระบบนั้นทำงานไม่ได้เลย อันนี้จึงเป็นเหตุผลที่ผู้วิจัยได้มีแนวคิดของการพัฒนาจุดนี้คือ การทำให้ตัวลูกทุกตัวนอกจากจะทำตัวเองเป็นสถานีทวนสัญญาณดังที่ผ่านมาแล้ว ยังต้องสามารถสแกนหาตัวลูกตัวอื่นที่ตนเองสามารถติดต่อได้เพื่อกำหนดเส้นทางที่สั้นที่สุดให้ข้อมูลหรือคำสั่งนั้นสามารถส่งผ่านตัวมันไปสู่ตัวลูกตัวต่อไปได้ ทำให้ข้อมูลจากตัวแม่สามารถผ่านไปสู่ตัวลูกที่อยู่ห่างไกลได้ ถึงแม้ว่าตัวลูกบางตัวจะเสียหายไปบ้าง เพราะการแสกนของตัวลูกแต่ละตัวนั้นจะสแกนหาตัวลูกตัวอื่นที่ทำงานได้และเส้นทางที่ใกล้สุดสู่ตัวลูกเป้าหมายนั่นเอง อันนี้ก็จะทำให้ระบบวัดสิ่งแวดล้อมฉลาดมากขึ้น มีความน่าเชื่อถือมากขึ้นนั่นเอง.

วัตถุประสงค์ของโครงการวิจัย

ในการทำการวิจัยครั้งนี้ วัตถุประสงค์ก็คือ ออกแบบและสร้าง เครื่องเครื่องวัดสิ่งแวดล้อมระยะไกล อัฉริยะ ที่สามารถมีคุณสมบัติพื้นฐานดังนี้

<u>ส่วนของซอฟร์แวร์</u>

- สร้างซอฟร์แวร์ของตัวแม่ในลักษณะของ การใช้งานที่ง่ายๆไม่ซับซ้อน (GUI) [9] สำหรับผู้ใช้เพื่ออ่านและเก็บข้อมูลสิ่งแวดล้อม (ตัวอย่างเช่น อุณหภูมิ[5]) ได้ ที่สามารถใช้งานกับเครื่องคอมพิวเตอร์ทั่วไปได้ (PC) ผ่านการสื่อสารแบบอนุกรม [4].
- สร้างซอฟร์แวร์ของตัวลูกที่มีฟังก์ชั่นการทำงาน ฉลาดมากขึ้น และสามารถแก้ไขข้อมูลเริ่มต้นได้อย่างสะดวก นำไปสู่การใช้งานจริงในระบบวิทยุระยะไกลจริงๆต่อไป

<u>ส่วนฮาร์ดแวร์</u>

- สร้างตัวเครื่องตัวแม่ ของเครื่อง วัดสิ่งแวดล้อมระยะไกลอัฉริยะ ที่มีขนาดเล็กและต่อใช้งานกับคอมพิวเตอร์ได้อย่างสะดวก
- สร้างตัวเครื่องตัวลูก ที่มีส่วนของตัววัดอุณหภูมิ และส่วนควบคุมที่จำเป็นอื่นๆ (เช่น อุปกรณ์ รีเลย์)
- ทั้งตัวหลักคือตัวแม่และตัวลูกจะออกแบบให้มีขนาดเล็ก, ติดตั้งใช้งานได้ง่าย และใช้แบตเตอร์รี่ได้

ขอบเขตของโครงการวิจัย

ทำการสร้างชุดต้นแบบระบบ โดยประกอบด้วย

- ชุดควบคุมหลักที่ใช้ติดตั้งกับ คอมพิวเตอร์ (PC) หรือตัวแม่ (Main)
  จำนวน 1 เครื่อง
- ชุดติดตั้งเซนเซอร์ที่ทำงานเป็นตัวตรวจวัดอุณหภูมิและความชื่น หรือที่เรียกว่าตัวลูก (Client) จำนวน 4 เครื่อง
- ซอฟร์แวร์ ควบคุมการทำงานทั้งระบบ จำนวน 1 ชุด

โดยทั้งนี้ ระบบจะมีความสามารถในการส่งคำสั่งไปหรืออ่านข้อมูลเซนเซอร์จากตัวลูกใดๆ โดยผ่านสัญญาณแบบไร้สาย (ในงานวิจัยเลือกใช้ อุปกรณ์ Xbee เพื่อให้เหมาะสมกับงบประมาณที่เสนอขอ) เพื่อนำมาแสดงผลที่คอมพิวเตอร์ (PC) ของผู้ใช้ได้ ดังตามวัตถุประสงค์ที่ได้กล่าวมา

ระยะเวลาดำเนินโครงการ

ระยะเวลาดำเนินการ 1 ปี

ประโยชน์ที่คาดว่าจะได้รับ

ดังที่ได้กล่าวมาในหัวข้อที่ 9 ว่าเป็นโครงการวิจัยที่ต่อยอดจากผลงานวิจัยที่ได้ทำมาแล้ว ดังนั้นในงานวิจัยนี้ได้นำข้อที่ควรปรับปรุงเพื่อให้งานวิจัยสามารถนำไปใช้งานได้จริงอย่างมีประสิทธิภาพ ซึ่งผลการวิจัยนี้ก็จะทำให้ ความน่าเชื่อถือ (ข้อมูล), ความคงทน (ทำงานได้แม้มีตัวลูกบางตัวไม่ทำงาน) มีประสิทธิภาพมากขึ้น ก็จะทำให้สามารถนำไปใช้ในการตรวจวัดสภาพสิ่งแวดล้อมได้อย่างมีประสิทธิภาพมากขึ้นรวมทั้งการบำรุงรักษาที่ไม่ศูนย์เปล่านั่นเอง ซึ่งสอดคล้องกันกับหัวข้อวิจัยของ สจล. (หัวข้อที่ 4) และสอดคล้องกันกับยุทธศาสตร์การพัฒนาประเทศตามแผนพัฒนาเศรษฐกิจและสังคมแห่งชาติ ฉบับที่ 10 (หัวข้อที่ 3) ตามลำดับ.

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง/การทบทวนวรรณกรรม

นอกเหนือจากงานวิจัยของผู้วิจัยเองทั้งสองงานวิจัยที่เกี่ยวเนื่องกันดังที่ได้กล่าวมาข้างต้นแล้ว ในรูปจุดประสงค์แบบเดียวกับการใช้งานนี้ยังไม่พบปรากฏ ในทั้งในและต่างๆประเทศ แต่หากมีงานที่ประยุกต์เอาระบบวิทยุในการส่งรับข้อมูลไปใช้งานบ้าง ดังเช่น “เครื่องควบคุมระบบอุปกร์แบบไร้สายระยะไกล ผ่านวิทยุสื่อสาร” ดังปรากฏใน www.thaiamp.com ซึ่งเป็นลักษณะของการใช้งานประยุกต์คนละแบบ.

รูปที่ 1 เครื่องส่งสัญญาณวิทยุควบคุมระยะไกล

บทที่ 3
วิธีดำเนินการวิจัยและผลการวิจัย

3.1 แผนงานระยะต่างๆของการดำเนินงานวิจัย

ระยะเวลาวิจัยรวม หนึ่งปี ดังแสดงตารางช่วงเวลาการทำงาน ในแต่ละส่วน ในตารางที่ 1

หมายเหตุ เดือนที่หนึ่ง หมายถึง เดือนที่นับจากเดือนที่ได้รับอนุมัติโครงงานวิจัย และเดือนที่สิบสอง หมายถึง เดือนสุดท้ายของการทำโครงงานวิจัย

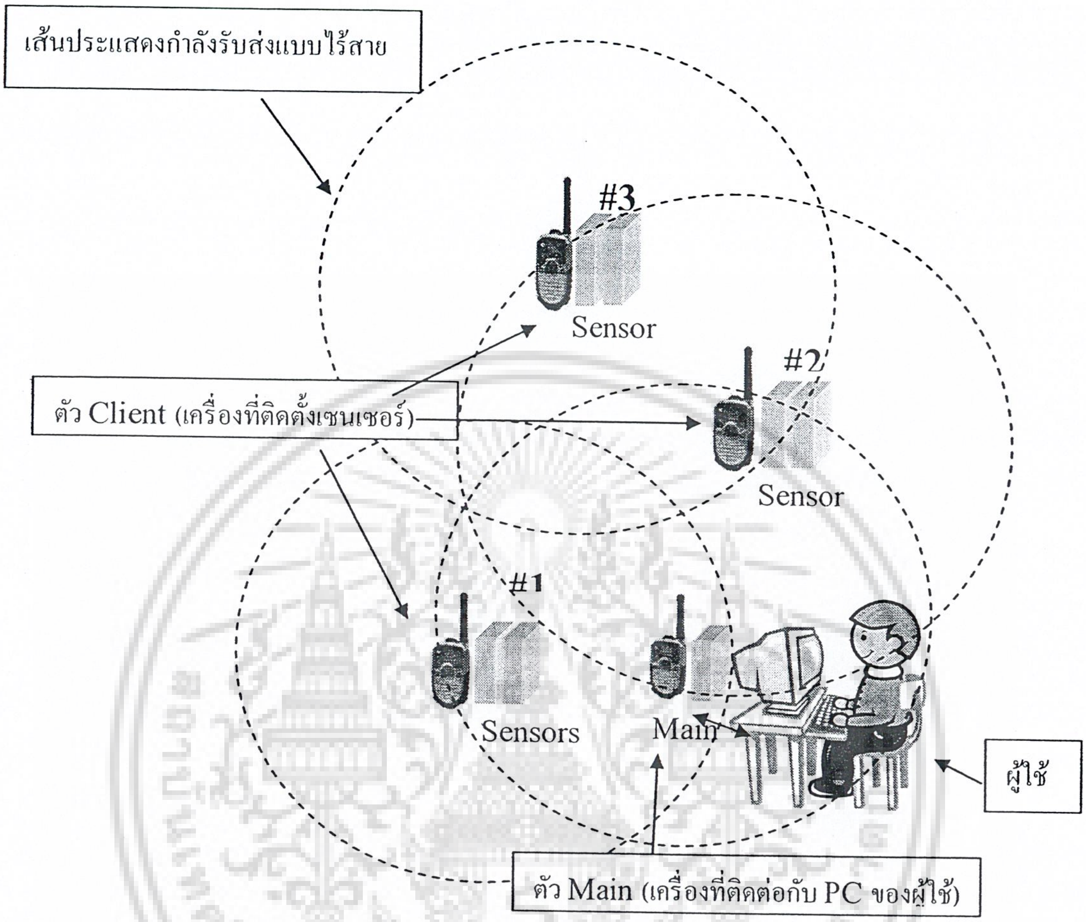
การดำเนินงาน	ระยะเวลา											
	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.
ศึกษาออกแบบวิธีการแก้ปัญหาการหาเส้นทางเดินของข้อมูลทั้งขาไปและขากลับให้ได้ข้อสรุปที่ดีที่สุด												
นำผลที่ได้จากข้อสรุปในช่วงแรกไปสร้างซอฟร์แวร์ โดยออกแบบฮาร์ดแวร์ของตัวแม่และตัวลูกให้สอดคล้องกัน												
ทดสอบซอฟร์แวร์บนฮาร์ดแวร์ที่ออกแบบไว้และแก้ไขให้สมบูรณ์												
ทดลองเก็บข้อมูลสรุปข้อดีเสียและทำรายงานการวิจัย												

ตารางที่ 1 แสดงช่วงเวลาการทำงาน

งานวิจัยนี้ได้แบ่งออกเป็นสองส่วน คือส่วนของ ฮาร์ดแวร์และซอฟร์แวร์ ดังนี้

3.2 แนวคิดใหม่ของขบวนการสื่อสาร

ทำการศึกษาและพัฒนาแนวคิดของการส่งคำสั่งหรือรับข้อมูล แบบใหม่ โดยจากรูปที่ 1 แสดงถึงภาพรวมของการใช้งานระบบวัดสภาพสิ่งแวดล้อมระยะไกลมาตฐาน



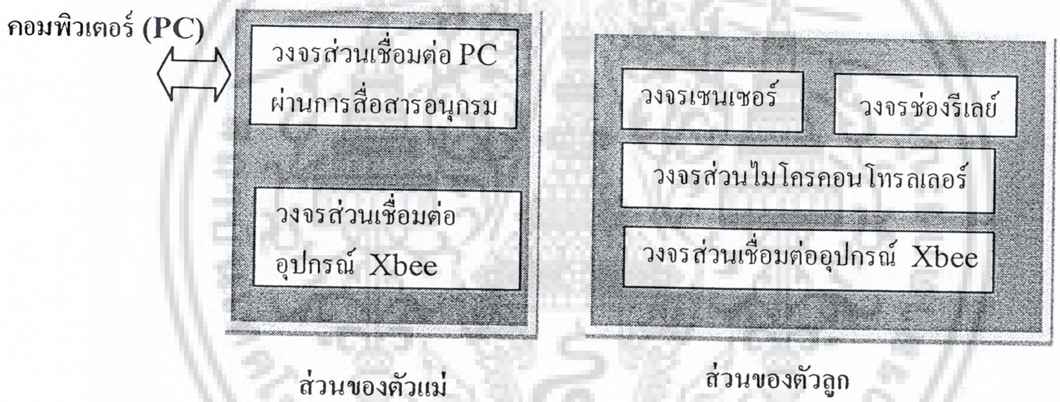
รูปที่ 2 แสดงระบบของการส่งคำสั่งและรับข้อมูลของงานวิจัยที่ผ่านมา

ในรูปที่ 2 แสดงการติดต่อกันแบบเป็นทอดๆ โดยทุกตัวของตัวลูกจะทำหน้าที่เป็นสถานีทวนสัญญาณ หากตัวแม่ไม่ได้ประสงค์จะอ่านค่าผลการวัดของตนเอง ตัวอย่างเช่น ตัวแม่ต้องการอ่านข้อมูลอุณหภูมิของตัวลูกหมายเลข 3 จะเห็นว่าสัญญาณตัวแม่ไม่สามารถส่งไปถึงตัวลูกหมายเลข 3 ได้ ก็ต้องส่งคำสั่งอ่านอุณหภูมิไปที่หมายเลข 2 ก่อน จากนั้นตัวลูกหมายเลข 2 ก็ส่งคำสั่งนั้นต่อไปที่หมายเลข 3 เมื่อตัวหมายเลข 3 ทราบว่าเป็นการอ่านค่าอุณหภูมิของตนเอง ก็จะส่งข้อมูลนั้นกลับมาตามเส้นทางเดิมคือผ่านมาที่หมายเลข 2 และกลับสู่ตัวแม่ต่อไป ดังนั้นในงานวิจัยที่ผ่านมาคำสั่งที่ออกจากตัวแม่ก็จำเป็นจะต้องกำหนดเส้นทางไว้ก่อน และหากในเส้นทางเหล่านั้น มีการเสียหายของตัวลูกตัวใดตัวหนึ่งก็จะไม่สามารถรับส่งข้อมูลกันได้ ซึ่งเป็นข้อเสียที่นำมาเป็นประเด็นแก้ไขปัญหาในงานวิจัยนี้นั่นเอง ในงานวิจัยนี้ได้ปรับปรุงให้มีความสามารถของตัวลูกให้สามารถส่งข้อมูลหรือคำสั่งเป็นทอดๆไปได้(เหมือนเป็นตัวทวนสัญญาณ) และขณะเดียวกันก็สามารถเปลี่ยนเส้นทางให้ข้อมูล

สามารถรับส่งกันได้แม้มีตัวลูกในเส้นทางเสียหายไป(หาเส้นทางอื่นเอง ตามที่โปรแกรมไว้) ทำให้ประสิทธิภาพดีขึ้น ไม่เสียเวลาหากต้องมีการเดินทางไปแก้ไขตัวลูกที่เสียหายเพราะไปแก้ไขได้อย่างถูกต้องถูกตัวที่เสียหายนั่นเอง

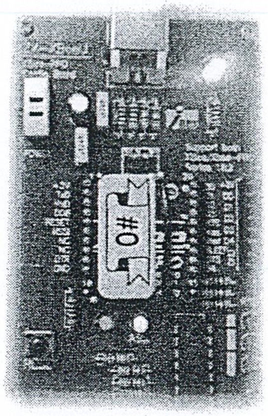
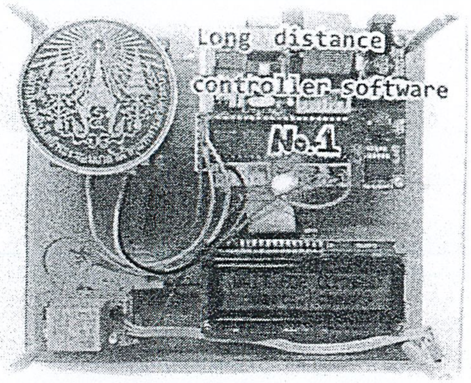
3.3 การสร้างส่วนของฮาร์ดแวร์

ทำการสร้างฮาร์แวร์เพื่อทดสอบ ซึ่งมีรูปแบบเบื้องต้นสามารถแสดงได้ดังรูปของบล็อกไออะแกรม ดังรูปที่ 5 ส่วนตัวลูกจะใช้ไมโครคอนโทรลเลอร์ (dsPIC30F4011) เป็นตัวควบคุม โดยต่อร่วมกันกับชุดสื่อสารที่เป็น Xbee (Pro) แทนวิทยุสื่อสาร เนื่องจากมีราคาแพง (และให้เหมาะสมกับงบประมาณที่เสนอขอ) ระบบจำเป็นต้องใช้ตัวลูกหลายตัวเพื่อพิสูจน์แนวคิดข้างต้น ดังนั้นตัวฮาร์ดแวร์ตัวแม่และตัวลูกเบื้องต้น จึงสามารถแสดงได้ดังรูปที่ 5



รูปที่ 3 แสดงถึงบล็อกฮาร์แวร์เบื้องต้นของตัวแม่และตัวลูก

ซึ่งจากรูปที่ 3 จะเห็นได้ว่าในส่วนของตัวแม่นั้นจะไม่มีไมโครคอนโทรลเลอร์อยู่ด้วย ก็เพราะว่าในการทดสอบนี้ยังไม่มีการใช้กับวิทยุจริงๆ ดังนั้นจึงสามารถจะใช้เครื่องโปรแกรมตัว Xbee แทนได้ (ZX-XbeeU) เพราะจะมีส่วนที่เชื่อมต่อกับ PC (RS-232C) เช่นเดียวกัน ในการควบคุมการเชื่อมต่อระหว่างตัวลูกอื่นๆ (ผ่าน Xbee) กับไมโครคอมพิวเตอร์ของผู้ใช้ รูปของตัวแม่และตัวลูกที่ใช้ในการทดลองในงานวิจัยนี้ก็แสดงดังรูปที่ 4

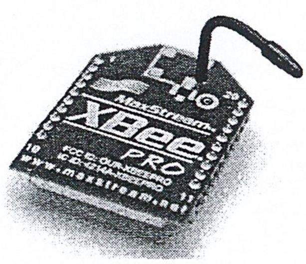
 

รูปที่ 4 แสดงตัวเครื่องของตัวแม่และตัวลูก

3.3.1 ส่วนโมดูลการรับส่งวิทยุ (Xbee Module)

ส่วนนี้ หน้าที่การทำงานก็คือเป็นส่วน ใช้รับส่งข้อมูลที่ได้จากเซนเซอร์ส่งออกอากาศสื่อสารกันระหว่างโมดูลของตัวลูกด้วยกันหรือตัวลูกกับตัวแม่ โดยปกติแล้วระบบที่ออกแบบนี้จะใช้งานกับระบบวิทยุสื่อสารจริงๆ (Walky Talky) ซึ่งมีกำลังส่งประมาณอย่างน้อยก็ 5 วัตต์ แต่เนื่องจากว่างบประมาณในโครงการนี้มีน้อยมากไม่พอที่จะต้องจัดซื้อเครื่องรับส่งวิทยุจริงๆ และจำเป็นต้องใช้หลายตัวด้วย ผู้วิจัยเลยต้องมองหาอุปกรณ์อื่นๆที่สามารถทำงานได้เช่นกัน ก็ได้เลือกเอา Xbee ซึ่งราคาต่อตัวไม่สูงมากนักและสามารถกำหนดให้ทำงานในแบบส่งกระจายเสียงได้ นั่นหมายความว่าสามารถนำมาใช้แทนวิทยุได้นั่นเอง เพียงแต่จะมีระยะรับส่งที่ใกล้ (ประมาณ 300 เมตร) แต่ก็เพียงพอเพราะว่า ในงานวิจัยนี้ต้องการทดสอบการทำงานของระบบว่าเป็นไปดังที่ต้องการหรือไม่เท่านั้น ยังไม่สามารถออกไปใช้งานจริงได้ นั่นเอง

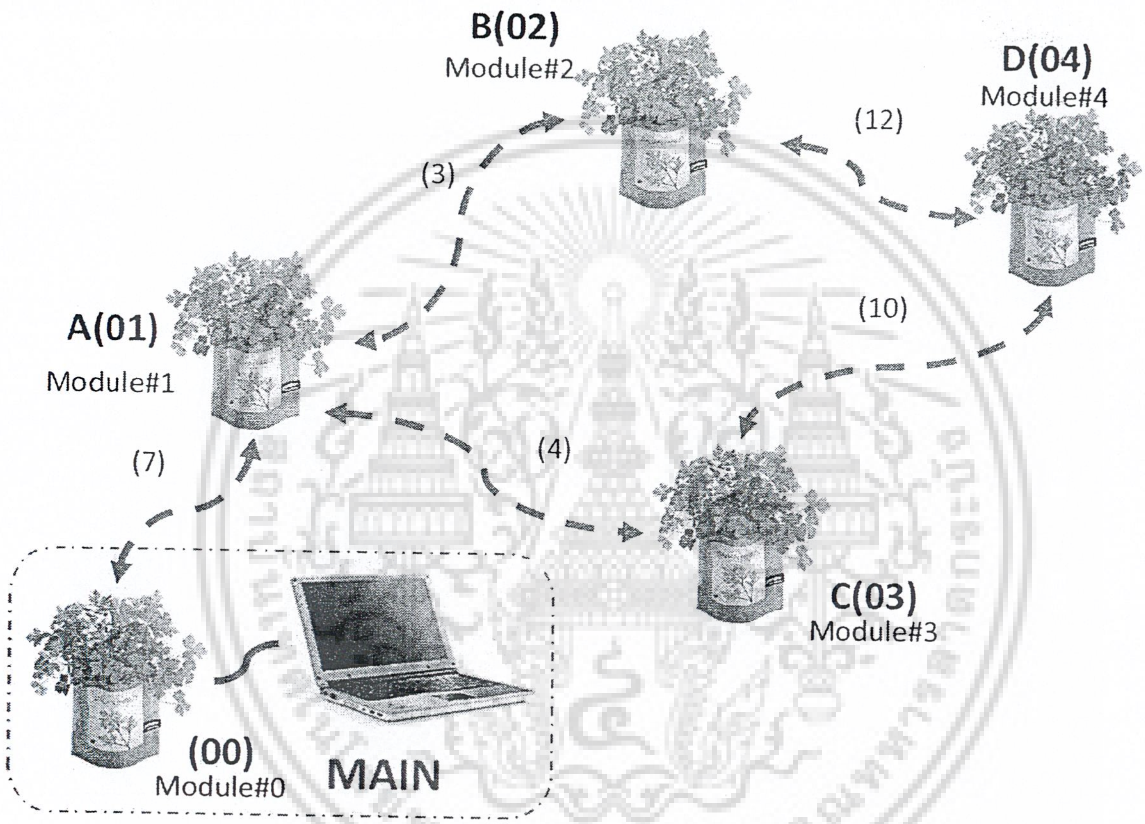
ตัวของ Xbee จะมีให้เลือกใช้งานได้หลายเบอร์ ผู้วิจัยเลือกเบอร์ ProXbee ซึ่งมีการใช้งานที่ง่ายและหาซื้อได้ง่าย สามารถกำหนดการทำงานด้วยโปรแกรม X-CTU ได้อย่างสะดวกเป็นโปรแกรมที่ให้มากับตัว Xbee อยู่แล้ว โดยกำหนดเป็นแบบกระจายเสียง (Boardscasting) หมายถึง ส่งกระจายไปทุกทิศทางแบบไม่เจาะจงผู้รับ ซึ่งเป็นเหมือนการส่งวิทยุทั่วไป โดยตัวรับก็จะมีการตรวจสอบด้วยโปรแกรมของตัวลูกเอง โมดูลที่ใช้งานแสดงดังรูปที่ 5



รูปที่ 5 แสดงโมดูล Xbee ที่ใช้งาน

3.3.2 การทำงานของเครื่องแม่กับ PC

ส่วนนี้จะเป็นฮาร์ดแวร์เครื่องโปรแกรม Xbee (ZX-XbeeU) ที่ต่อเชื่อมกับคอมพิวเตอร์ มีหน้าที่ติดต่อรับ-ส่งข้อมูล หรือส่งคำสั่งจากตัวเครื่องคอมพิวเตอร์ ไปยังตัวลูก (Client) โดยรับคำสั่งผ่านทางหน้าจอคอมพิวเตอร์ของผู้ใช้ ทำการส่งคำสั่งไปยังอุปกรณ์ XBee-PRO โดยดังรูปที่ 6 ในส่วนของ Module#0 นั่นก็คือตัวแม่นั่นเอง



รูปที่ 6 แสดงตัวแม่ (Module#0) ที่เชื่อมต่อกับ PC เพื่อรับคำสั่งส่งไปยังตัวลูกอื่นๆ

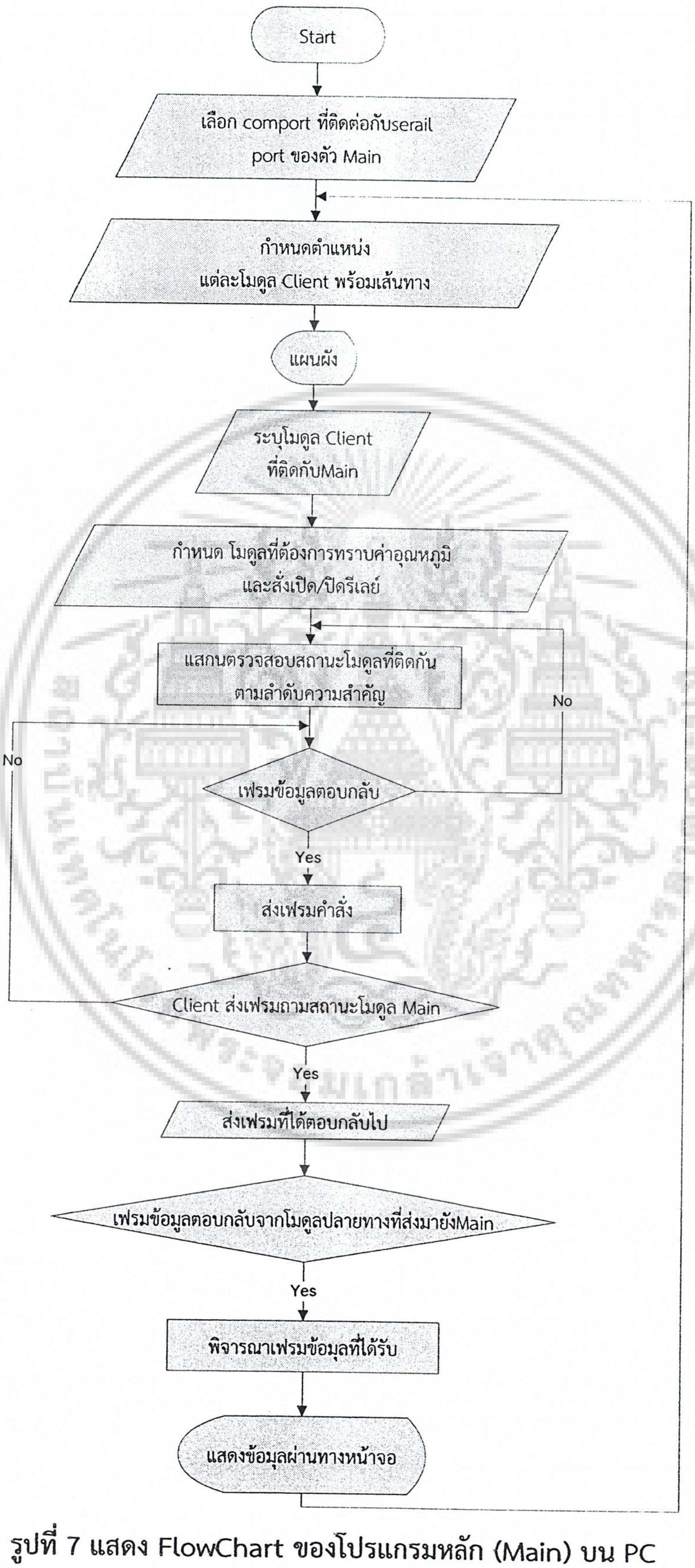
อนึ่ง ในส่วนของ XBee-PRO ที่ได้จัดหามานั้นเป็นแบบ XBee-PRO Series 1 ซึ่งสามารถกำหนดให้ทำงานในโหมด Broadcast ได้จึงได้นำโหมดการทำงานนี้มาใช้เพื่อให้ทุกตัวที่อยู่ในรัศมีการสื่อสารที่สามารถติดต่อกันได้โดยตรง จากการทำงานของโหมดดังกล่าวทำให้ Client แต่ละตัวสามารถมีเส้นทางการติดต่อสื่อสารกันได้หลายหลายเส้นทาง ดังนั้นการติดต่อสื่อสารกันระหว่างโมดูลก็ต้องทำการเลือกเส้นทางที่ดีที่สุดให้กับ Client ทุกตัว ซึ่งเส้นทางการส่งข้อมูลจะถูกกำหนดโดยUser จากนั้นโปรแกรมใน Main จะทำการติดต่อกับโมดูลตามเส้นทางหลักของ User แต่หากเส้นทางหลักที่ได้เลือกเกิดพบปัญหา โปรแกรมจะทำการเปลี่ยนเส้นทางให้ใหม่

3.3.3 โปรแกรมส่วนของ PC (เขียนด้วย C#, GUI)

โปรแกรมหลัก (Main) (เขียนด้วย C# บน PC) โปรแกรมส่วนนี้ทำหน้าที่เป็น GUI เพื่อติดต่อกับผู้ใช้ในการสั่งงานควบคุมต่างๆ ต่อโมดูลตัวลูก เช่น การอ่านผลอุณหภูมิ สั่งรีเลย์ทำงาน โดยมีการจัดรูปแบบคำสั่งเพื่อติดต่อกับตัววัดที่ต้องการติดต่อด้วยอย่างอัตโนมัติ

จากการทำงานในโหมด Broadcastทำให้ Client แต่ละตัวสามารถมีเส้นทางการติดต่อสื่อสารกันได้หลายหลายเส้นทาง ดังนั้นบนโปรแกรมหลักจึงต้องทำการเลือกเส้นทางที่ดีที่สุดให้กับ Client ทุกตัว โดยโปรแกรมหลักจะพิจารณาเส้นทางที่เหมาะสมจากเส้นทางที่ User กำหนดให้ ซึ่งเส้นทางที่ User กำหนดให้มีทั้งเส้นทางหลัก และเส้นทางสำรองอื่น โปรแกรมจะทำการจัดลำดับความสำคัญของเส้นทางต่างๆจากนั้นจะเลือกส่งสัญญาณไปตามลำดับความสำคัญ หากเส้นทางหลักเกิดพบปัญหา โปรแกรมหลักจะทำการเปลี่ยนเส้นทางการสื่อสารให้ใหม่ตามลำดับความสำคัญของเส้นทางนั้นๆ จึงทำให้การส่งข้อมูลประสบความสำเร็จทุกครั้ง เพื่อป้องกันการสูญหายของข้อมูล ทุกครั้งที่จะส่งข้อมูล จึงต้องมีการตรวจสอบสถานะของโมดูลตัวที่ต้องการจะส่งข้อมูลไปก่อนเสมอ หากโมดูลนั้นพร้อมใช้งาน ก็จะทำการส่งข้อมูลไปยังตัวนั้นทันที การตรวจสอบนั้น นอกจากเป็นการป้องกันการสูญหายของข้อมูลแล้ว ยังเป็นการตรวจสอบสถานะของแต่ละโมดูลไปในตัวด้วย ส่วนการส่งข้อมูลกลับก่อนที่ตัว Main จะรับข้อมูลกลับก็จะต้องถูกตรวจสอบสถานะจาก Client ด้วยเช่นกัน

ในรูปแบบของคำสั่งจากด้านบนก็จะถูกจัดการสร้างขึ้นเองจากส่วนของโปรแกรมบน PC ซึ่งมีการเชื่อมต่อกับโมดูลเบอร์ศูนย์ (#0) อยู่แล้วโดยโมดูลเบอร์ศูนย์ จะมี MY_ID=0x00 คำสั่งดังกล่าวสามารถเขียนในรูป Flowchart ได้ดังแสดงในรูปที่ 7 ดังนี้



รูปที่ 7 แสดง FlowChart ของโปรแกรมหลัก (Main) บน PC

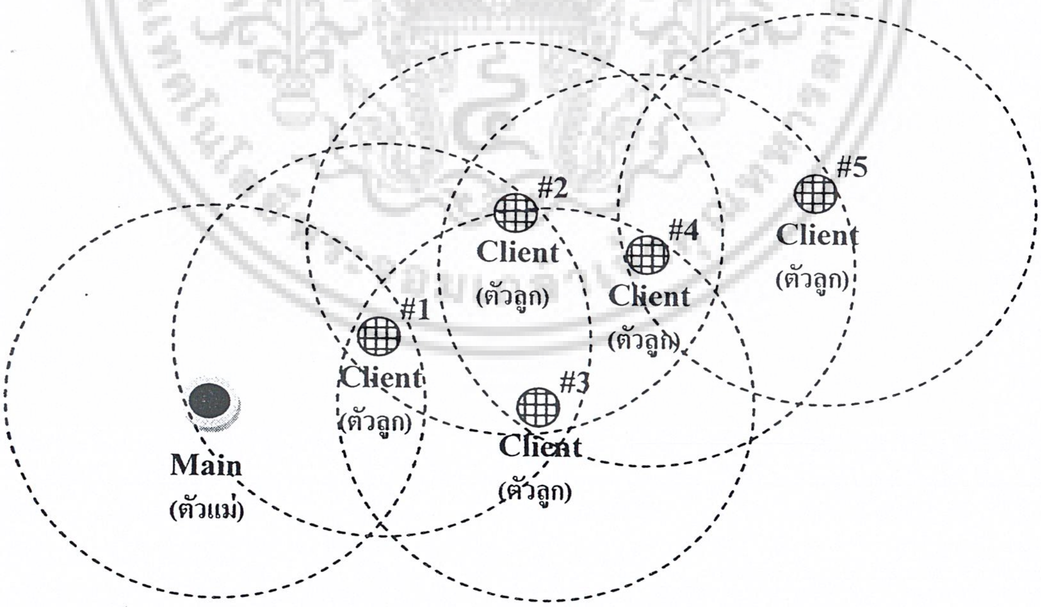
3.3.4 ส่วนซอร์ฟแวร์บนไมโครคอนโทรลเลอร์ (ตัวเครื่องลูกที่ใช้งาน)

เนื่องจากในส่วนโปรแกรมตัวลูกนี้จะมีความสำคัญมากเพราะต้องมีการส่งต่อสัญญาณเป็นช่วงๆ คล้ายเครื่องทวนสัญญาณและยังต้องมีการแสกนหาเส้นทางอื่นเพื่อให้ส่งข้อมูลได้เมื่อเส้นทางหลักมีปัญหา ดังนั้นจึงข้อเริ่มจากแนวคิดก่อนจะไปเป็นโปรแกรมต่อไป

3.3.4.1 แนวคิดการออกแบบซอฟร์แวร์

จากรูปแบบในรูปที่ 8 จะเห็นว่าได้มีการกำหนดเส้นทางจากตัวแม่ไปสู่ตัวลูกแต่ละตัวไว้แล้ว ซึ่งกรณีนี้หากตัวลูกที่เป็นตัวทวนสัญญาณระหว่างทางตัวใดตัวหนึ่งเสีย ก็จะทำให้ไม่สามารถใช้งานได้กับตัวอื่นๆ ที่อยู่ไกลออกไปได้เลย ดังนั้นแนวคิดใหม่เบื้องต้นที่นำเสนอก็คือ ทำให้ตัวลูกแต่ละตัวมีความฉลาดมากขึ้น โดยกำหนดให้มีการสแกนหาตัวลูกตัวอื่นที่อยู่รอบตัวเองที่สัญญาณตัวเองสามารถส่งไปถึงได้ว่ามีตัวใดใช้งานได้ และตัวที่ใช้งานได้นั้นตัวใดใกล้กับตัวเป้าหมายที่ตัวแม่ต้องการติดต่อด้วยมากที่สุด ก็เลือกตัวลูกนั้นในการส่งผ่านคำสั่งหรือข้อมูลออกไป แต่หากไม่มีตัวใดทำงานเลยก็ทำการเก็บหมายเลขของตัวที่เสียเหล่านั้นกลับสู่ตัวแม่ให้ทราบ เพื่อผู้ใช้จะได้วางแผนการแก้ไขต่อไปภายหลัง ดังนั้นเส้นทางของการเดินทางคำสั่งจากตัวแม่ไปสู่ตัวลูกเป้าหมายและเส้นทางข้อมูลกลับจากตัวเป้าหมายไปสู่ตัวแม่อาจไม่เป็นเส้นทางเดียวกันก็ได้ เพราะอาจมีตัวใดไม่ทำงานในระหว่างการเดินทางกลับของข้อมูลก็อาจเป็นได้ แต่ข้อมูลก็สามารถกลับมาได้เพราะใช้ตัวลูกตัวอื่นที่อยู่ใกล้เคียงนั่นอง

ซึ่งขอแสดงตัวอย่างเบื้องต้น ดังนี้ โดยกำหนดให้รูปแบบของการวางตัวลูกในระบบวัดสิ่งแวดล้อมเป็นดังรูปที่ 8 ดังนี้

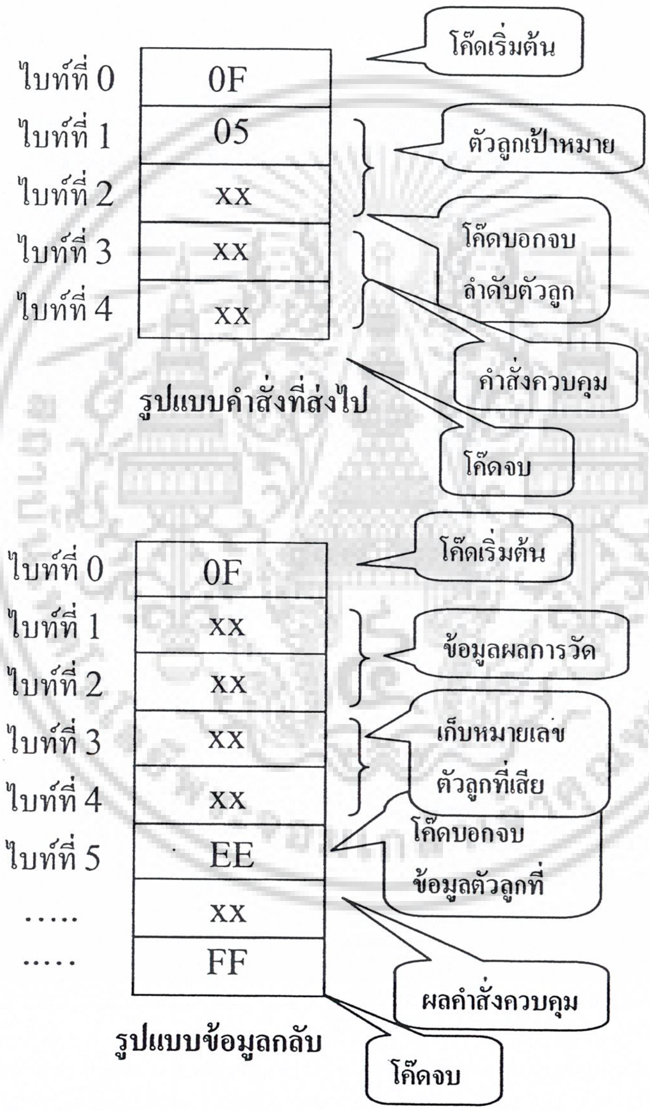


รูปที่ 8 แสดงตัวอย่างการวางตำแหน่งของตัวแม่และตัวลูกจำนวน 5 ตัว

จากรูปที่ 8 สมมุติเราต้องการส่งคำสั่งจากตัวแม่ (main) ไปอ่านอุณหภูมิตัวลูก (Client) ตัวที่ 5 เราจะมีเส้นทางการเดินทาง ที่สั้นที่สุดที่ควารจะเป็นดังนี้

ตัวแม่ → ตัวลูก #1 → ตัวลูก #2 → ตัวลูก #4 → ตัวลูก #5

ในแนวคิดใหม่นี้ รูปแบบของคำสั่งก็จะสั้นลงเพราะไม่จำเป็นต้องบอกเส้นทางการเดินทางไปสู่ตัวเป้าหมายว่าผ่านตัวลูกตัวใดบ้าง (ตัวลูกแต่ละตัวจะทำการค้นหาเส้นทางเอง) ดังนั้นรูปแบบคำสั่งที่ควรจะเป็นในเบื้องต้น ก็แสดงดังรูปที่ 9



รูปที่ 9 แสดงรูปแบบคำสั่งของแนวคิดใหม่ และรูปบบข้อมูลกลับในกรณีของข้อมูลอุณหภูมิได้

ในรูปที่ 9 นั้น รูปแบบคำสั่งที่ส่งไปจะมีเพียง 5 ไบท์เท่านั้น เพราะจะมีเพียงตัวลูกเป้าหมายเท่านั้นที่เป็นกุญแจสำคัญในการนำไปคำนวณหาเส้นทางของตัวลูกอื่นๆ แต่ในส่วนของข้อมูลที่กลับมาจะมีขนาดเปลี่ยนแปลงไปตามสถานการณ์ที่เกิดขึ้นคือในไบท์ที่เก็บหมายเลขของตัวลูกที่ไม่ทำงานก็จะ

มีจำนวนมากน้อยตามเป็นจริง (โค๊ดเริ่มต้น, โค๊ดจบ ต่างๆ เป็นค่าที่กำหนดขึ้นเพื่อการตรวจสอบเท่านั้น และจะมีค่าไม่ตรงกับข้อมูลอยู่แล้ว)

การทำงานของระบบตัวอย่างสามารถอธิบาย ตัวลูกแต่ละตัวลูกจะมีการเลือกเส้นทางดังนี้

ตัวลูกตัวที่ 1 (#1) มีขอบเขตของสัญญาณที่สามารถติดต่อได้คือ ตัวแม่, ตัวลูก #2, ตัวลูก #3 เมื่อวัดระยะเส้นทางจากตัวเป้าหมาย (ตัวลูก #5) มาสู่ตัวลูกที่ติดต่อได้เหล่านี้ จะได้ ตัวลูก #2 ที่ใกล้ตัวเป้าหมายมากที่สุดและตัวลูก #2 นี้ก็ย่อมมีความน่าจะเป็นที่จะสามารถติดต่อกับตัวลูกตัวอื่นอีกเพื่อไปถึงตัวเป้าหมายได้ ดังนั้น ตัวลูก #1 นี้จึงเลือก ตัวลูก #2 เป็นเส้นทางที่มีความสำคัญสูงสุดและ ตัวลูก #3 เป็นเส้นทางเลือกลำดับรองลงมา หากตัวลูก #2 ไม่ทำงาน

ตัวลูกตัวที่ 2 (#2) มีขอบเขตของสัญญาณที่สามารถติดต่อได้คือ ตัวลูก #1,ตัวลูก #3 และตัวลูก #4 และเมื่อวัดระยะเส้นทางจากตัวเป้าหมายมาสู่ตัวลูกที่ติดต่อได้เหล่านี้ จะได้ตัวลูก #4 ที่มีระยะทางใกล้ที่สุด ดังนั้น ตัวลูก #2 นี้ก็จะเลือก ตัวลูก #4 เป็นลำดับต้นในการส่งผ่านข้อมูล แต่หากตัวนี้ไม่ทำงาน (เสีย) ก็จะเลือก ตัวลูก #3 แทน ซึ่งจะเห็นได้ว่า ตัวลูก #3 นี้ มีขอบเขตของสัญญาณที่ติดต่อได้คือ ตัวลูก #1,#2 และ #4 และ ตัวลูก #4 นี้มีเส้นทางสั้นสุด จึงถูกเลือกให้ส่งสัญญาณต่อไป คำสั่งจึงสามารถส่งต่อไปได้ แต่หาก ตัวลูก #4 ไม่ทำงานอีก ตัวลูก #3 ก็จะทำการส่งข้อมูลหมายเลขที่เสีย (ที่สะสมมา คือ ตัวลูก #2, #4) กลับไปสู่ตัวแม่คือหาเส้นทางที่สั้นที่สุด ซึ่งก็คือ ตัวลูก #1 นั่นเอง ซึ่งก็จะเป็นเส้นทางใหม่ใช้ส่งข้อมูลกลับไปสู่ผู้ใช้งานต่อไป

ตัวลูกตัวที่ 4 (#4) มีขอบเขตของสัญญาณที่สามารถติดต่อได้คือ ตัวลูก #2,ตัวลูก #3 และ ตัวลูก #5 ซึ่งกรณีนี้ ก็จะทำการติดต่อไปที่ ตัวลูก #5 เลยเพราะเป็นตัวเป้าหมาย แต่หากติดต่อไม่ได้เพราะ ตัวลูก #5 ไม่ทำงานก็จะรายงานผลอันนี้ กลับสู่ตัวแม่ ด้วยหลักการอันเดียวกัน

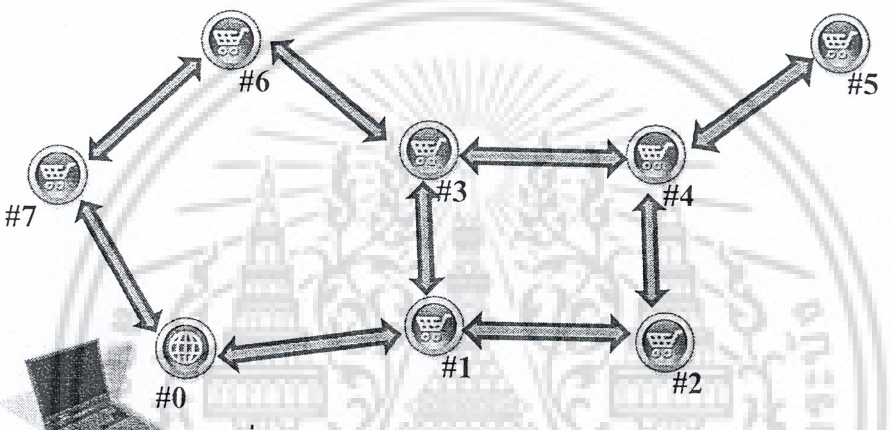
ตัวลูกตัวที่ 5 (#5) มีขอบเขตของสัญญาณที่สามารถติดต่อได้คือ ตัวลูก #4 และเป็นตัวเป้าหมายเอง จึงทำการเก็บค่าอุณหภูมิและส่งกลับสู่ตัวแม่ กลับผ่านมาที่ ตัวลูก #4 ด้วยหลักการอันเดียวกัน

ด้วยแนวคิดนี้จะเห็นได้ว่าเราได้ความน่าเชื่อถือของระบบมากขึ้น อีกทั้งนอกจากจะทราบถึงข้อมูลอุณหภูมิที่ต้องการอ่านแล้วยังทราบถึงตัวลูกที่เสียหายได้ด้วย (หากมีตัวที่เสียหาย) ทำให้สามารถวางแผนแก้ไขได้ง่าย เพราะในสภาวะจริงนั้น ตัวลูกจะอยู่ห่างไกลกันมากและมีหลายตัว การเดินทางเข้าไปตรวจสอบในแต่ละตัวจึงเป็นเรื่องของการใช้ค่าใช้จ่ายที่สูง ดังนั้นการไม่เสียเที่ยวของการไปแก้ไขอุปกรณ์ข้อมูลว่าตัวลูกตัวใดเสียจึงเป็นสิ่งสำคัญ และอีกอย่างคือในการส่งคำสั่งนั้นในรูปแบบเดิมนั้นต้องมีการกำหนดเส้นทางตายตัวก่อน ดังนั้น หากเส้นทางต้องผ่านตัวลูกจำนวนมากก็จะมีขนาดของคำสั่งที่ยาวมากตามไปด้วย ซึ่งไม่เป็นผลดีเพราะการส่งคำสั่งด้วยคลื่นวิทยุนั้นจะเกิดการรบกวนได้ง่ายจากเหตุสุดวิสัย เช่นการทับซ้อนของความถี่ผู้อื่นที่ใช้งานพร้อมกัน ดังนั้นขนาดของคำสั่งควรจะสั้นมากๆ ซึ่งในแนวคิดใหม่ในรูปแบบของคำสั่งจะมีการบอกเพียงว่าจะติดต่อกับหมายเลขตัวลูกเบอร์อะไร เมื่อมีการส่งไปแล้วตัวลูกแต่ละตัวก็จะค้นหาเส้นทางที่ดีที่สุดเองอัตโนมัติ ซึ่งในส่วนของการค้นหานี้ก็สามารถทำได้ง่ายโดย เพราะในการใช้งานระบบในขั้นตอนแรกก็ต้องมีกาวางตำแหน่งของตัวลูกแต่ละตัวบนแผนที่ก่อนอยู่แล้ว นั่นหมายความว่าตัวลูกแต่ละตัวก็รู้ถึงตำแหน่งของตัวเองในแผนที่และตัวลูกแต่ละตัวก็ทราบได้ว่าตัวลูกตัวอื่นที่อยู่รอบข้างที่ตนสามารถติดต่อได้มีตัวหมายเลขใดบ้าง ข้อมูลตรงนี้ของตัวลูกแต่ละตัวก็จะไม่เหมือนกัน ส่วนระยะห่างของตัวเป้าหมายสู่ตัวลูกแต่ละตัวก็สามารถคำนวณได้ง่าย เพราะแต่ละตัวจะมีตำแหน่งบนแผนที่อยู่แล้วสามารถคำนวณโดยหลัก

ตรีโกณมิติได้ นั่นเอง เพียงแต่จุดที่อาจเป็นจุดอ่อนบ้าง ก็คือจะมีการติดต่อระหว่างตัวลูกกันเองมากครั้งขึ้นเพราะต้องมีการตรวจสอบว่าตัวลูกใดเสียหรือไม่ นั่นเอง

ตัวลูกจะต้องออกแบบให้มีขนาดเล็กที่สุด และออกแบบให้ใช้พลังงานน้อยที่สุด เพราะต้องถูกนำไปติดตั้งในพื้นที่ห่างไกล พลังงานจึงต้องใช้จากแบตเตอร์รี่เท่านั้น ในส่วนของตัวแม่ จะมีขนาดที่เล็กเช่นกันเพราะไม่ต้องมีส่วนของแบตเตอร์รี่จ่ายแรงไฟ และทำให้ติดต่อกันในช่องอนุกรมซึ่งจะทำให้ประหยัดค่าใช้จ่ายมากขึ้นและขนาดจะเล็กลงมาก

3.3.4.2 ตัวอย่างการหารายชื่อของ Send list, Back list กรณีที่ 1



รูปที่ 10 แสดงตัวอย่าง แผนผังกรณีที่ 1

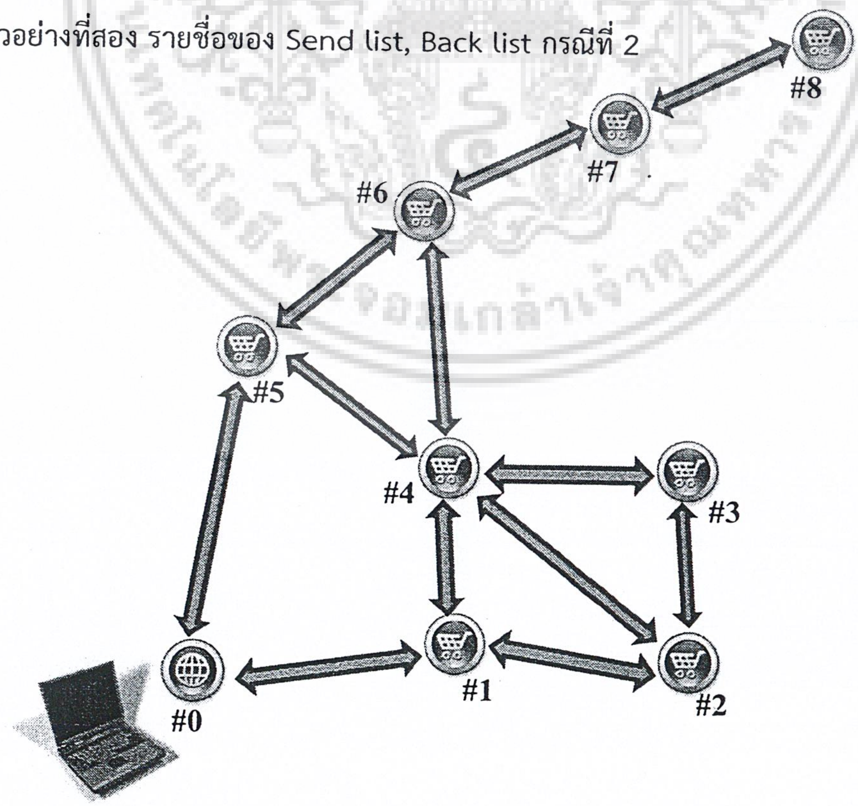
ไป (คำสั่ง) Send list		กลับ (ข้อมูล) Back list	
โมดูลเป้าหมาย	ลำดับความสำคัญของโมดูลทางผ่าน	โมดูลเป้าหมาย	ลำดับความสำคัญของโมดูลทางผ่าน
เบอร์ #0 (ตัวแม่)			
#6	#7, #1	#0	ไม่มีเพราะไม่ได้ส่งข้อมูลไปไหน
#3,#4,#2,#5	#1, #7		
เบอร์ #1 (โมดูลตัวลูก)			
#4,#5,#6,#7	#3, #2	#0	ไม่มีเพราะไม่ได้ใช้ตัวส่งผ่าน
เบอร์ #2(โมดูลตัวลูก)			
#5	#4	#0	#1,#4
เบอร์ #3(โมดูลตัวลูก)			
#2, #5	#4	#0	#1, #6
#7	#6		
เบอร์ #4(โมดูลตัวลูก)			
#5,#2	ไม่ต้องมีการส่งผ่านตัวใดอีก	#0	#3, #2
เบอร์ #5(โมดูลตัวลูก)			

#5	ไม่ต้องมีการส่งผ่านตัวใดอีก	#0	#4
เบอร์ #6(โมดูลตัวลูก)			
#2,#4,#5	#3	#0	#7, #3
เบอร์ #7(โมดูลตัวลูก)			
#2,#3,#4,#5	#6	#0	ไม่มีเพราะไม่ได้ส่งข้อมูลผ่านตัวใด

ตารางที่ 2 การเดินทางตามแผนผังกรณีที่ 1

จะเห็นได้ว่า จำนวนของตัวที่ผ่าน (ตัวที่จะถูกสแกน) เพื่อใช้สำหรับส่งคำสั่งต่อไปหรือรับข้อมูลส่งกลับ ของแต่ละโมดูลตัวลูกนั้น จะมีจำนวนที่ไม่เหมือนกัน ซึ่งบางตัวอาจไม่มีตัวอื่นที่ต้องส่งผ่านคำสั่งหรือข้อมูลก็เป็นได้ ทั้งนี้เนื่องจากว่า เราจะต้องพิจารณาถึงเส้นทางที่จำเป็นต้องผ่านเฉพาะความเป็นจริงเท่านั้นก่อน (ซึ่งวิธีการนี้จะลดภาระและเวลาที่ต้องเสียไปกับการสแกนผ่านตัวที่ไม่มีความหมายอย่างไม่จำเป็น) จากนั้นจึงสามารถกำหนดตัวที่จะถูกสแกนได้และเรียงตามลำดับความสำคัญด้วย เมื่อได้ค่ารายชื่อเหล่านี้แล้วก็สามารถนำไปเป็นข้อมูลของแต่ละโมดูลได้ โดยอาจโปรแกรมเข้าไปเลยหรือทำเป็นแผ่น SD CARD ใส่กับโมดูลนั้นๆ ก็ได้ซึ่งรายชื่อนี้น่าจะสร้างได้อย่างอัตโนมัติโดยการเขียนโปรแกรมรับแผนผังแล้วให้คอมพิวเตอร์จัดการหาให้ (ทำบน GUI คอมพิวเตอร์ภาษา C#) จากนั้นก็ค่อยบันทึกลง SD CARD ต่อไป

3.3.4.3 ตัวอย่างที่สอง รายชื่อของ Send list, Back list กรณีที่ 2



รูปที่ 11 แสดงตัวอย่าง แผนผังกรณีที่ 2

ไป (คำสั่ง) Send list		กลับ (ข้อมูล) Back list	
โมดูลเป้าหมาย	ลำดับความสำคัญของโมดูลทางผ่าน	โมดูลเป้าหมาย	ลำดับความสำคัญของโมดูลทางผ่าน
เบอร์ #0 (โมดูลตัวแม่)			
#4,#6,#7,#8	#5	#0	ไม่มีเพราะเป็นตัวแม่เอง
#2,#3	#1		
เบอร์ #1 (โมดูลตัวลูก)			
#3	#4, #2	#0	ไม่มีเพราะการส่งผ่าน
#6,#7,#8	#4		
เบอร์ #2(โมดูลตัวลูก)			
	ไม่มีเพราะไม่มีตัวที่ต้องผ่าน	#0	#1, #4
เบอร์ #3(โมดูลตัวลูก)			
	ไม่มีเพราะไม่มีตัวที่ต้องผ่าน	#0	#2, #4
เบอร์ #4(โมดูลตัวลูก)			
#7,#8	#6	#0	#1, #5
เบอร์ #5(โมดูลตัวลูก)			
#7,#8	#6	#0	ไม่มีเพราะไม่ต้องผ่านตัวใด
#2,#3	#4		
เบอร์ #6(โมดูลตัวลูก)			
#8	#7	#0	#5, #4
เบอร์ #7(โมดูลตัวลูก)			
	ไม่มีเพราะไม่มีตัวที่ต้องผ่าน	#0	#6
เบอร์ #8(โมดูลตัวลูก)			
	ไม่มีเพราะไม่มีตัวที่ต้องผ่าน	#0	#7

ตารางที่ 3 การเดินทางตามแผนผังกรณีที่ 2

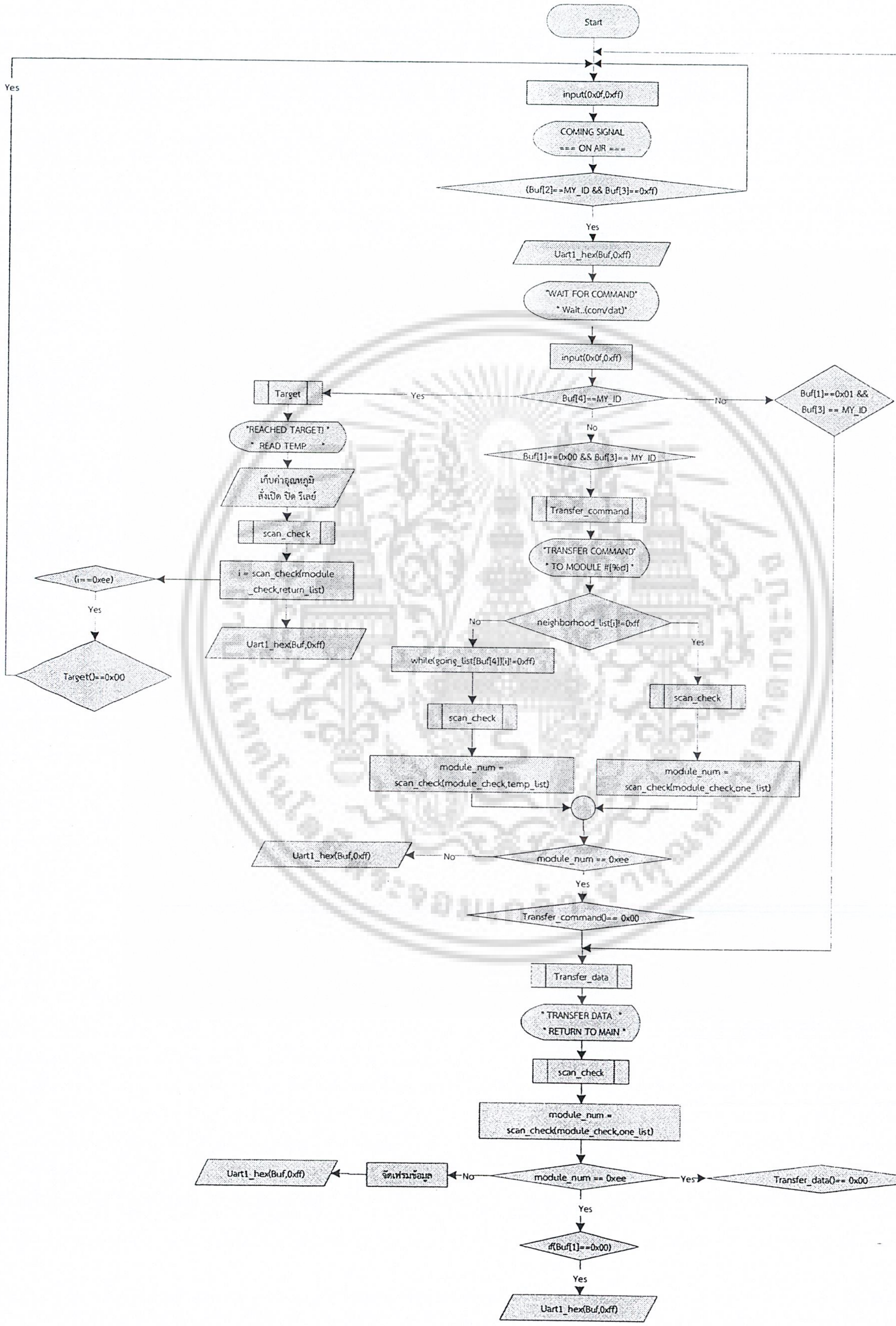
หมายเหตุ รายชื่อของตัวโมดูลที่ถูกสแกนเพื่อใช้ส่งผ่านคำสั่งหรือข้อมูล (ชื่อในช่อง ลำดับความสำคัญของโมดูลทางผ่าน) หมายถึงว่าเป็นรายชื่อโมดูลเพื่อจะใช้ผ่านไปถึงตัวเป้าหมาย (ชื่อในช่องโมดูลเป้าหมาย) ซึ่งมีการเรียงลำดับการสแกนไว้แล้ว ส่วนตัวเป้าหมายอื่นที่ไม่มีชื่อในช่อง (ช่องโมดูลเป้าหมาย) นี้นั้นก็หมายถึงใช้เบอร์นั้นในการส่งคำสั่งหรือรับข้อมูลไปเลยเพราะมันไม่จำเป็นต้องผ่านตัวใดไปนั่นเอง

ดังนั้น จะเห็นได้ว่าการวางตำแหน่งของตัวลูกแต่ละตัวจะทำให้มีผลต่อการสร้างรายชื่อของตัวที่ถูกสแกน ซึ่งยิ่งมีโครงข่ายที่มีการเชื่อมต่อได้หลายตัวก็จะยิ่งทำให้ระบบมีโอกาสใช้งานได้ดี คือรับส่งข้อมูลได้สำเร็จมาขึ้นทั้งนี้เพราะมีโอกาสเลือกเส้นทางอื่นในการไปและกลับได้มากขึ้นนั่นเอง

ดังนั้นโปรแกรมของตัวลูก (Client) #1, #2, #3, #4, โปรแกรมส่วนนี้ทำหน้าที่อยู่ 3 อย่าง

1. ทำหน้าที่ส่งต่อคำสั่ง เมื่อรับคำสั่งมาแล้วรู้ว่าไม่ใช่คำสั่งที่จะทำการอ่านค่าจากโมดูลตัวเอง ก็จะจัดรูปแบบของคำสั่งใหม่แล้วส่งคำสั่งนี้ต่อไปสู่โมดูลตัวอื่นตามเส้นทางที่เลือกจากตารางเส้นทางที่มีอยู่ประจำในแต่ละโหนด โดยจะพิจารณาตามลำดับความสำคัญของเส้นทาง โดยเลือกเส้นทางให้เป็นเส้นทางหลัก และเส้นทางสำรอง
2. ทำหน้าที่ส่งต่อข้อมูลกลับ ในช่วงตอนส่งข้อมูลกลับไปสู่ PC เมื่อรับข้อมูลที่จะส่งกลับมา (จากตัวโมดูลอื่น) ถึงตัวเอง ก็จะทำการส่งข้อมูลนั้นต่อให้กลับไปตามเส้นทางเดิมที่ส่งมาให้
3. ทำหน้าที่อ่านค่าอุณหภูมิของตอนเองตามคำสั่งที่ระบุมา พร้อมทั้งการอ่านสถานะของอุปกรณ์ตัวนั้นๆ จากนั้นก็จะจัดข้อมูลใหม่ส่งกลับไป (ตามเส้นทางที่เดิมที่ส่งข้อมูลมา ทำจึงสามารถส่งกลับได้อย่างถูกต้อง)

ไม่ว่าClient จะทำหน้าที่ใดก็ตาม ก่อนส่งสัญญาณต่อไปยังโมดูลถัดไป ต้องมีการตรวจสอบสถานะโมดูลตัวถัดไปก่อนเสมอ จากหลักการทำงานของ Client สมารถเขียนเป็น Flow chart ได้ดังนี้



รูปที่ 12 แสดง Flowchart ของโปรแกรมตัวลูก (Client)

3.4 การใช้งาน

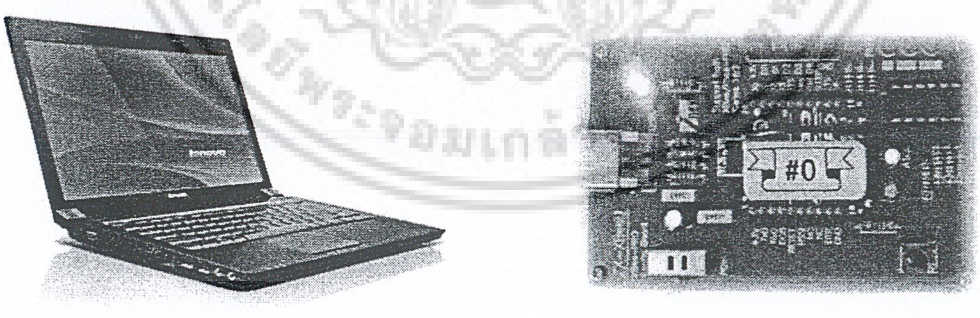
ในส่วนนี้ ก็จะได้แสดงการจัดเตรียมระบบ และการทดสอบใช้งานโดยกำหนดให้มีสถานะการที่แตกต่างกัน ทั้งนี้เพื่อจะได้พิสูจน์ถึงการทำงานของซอฟร์แวร์ทั้งหมด ที่ออกแบบไว้ว่าทำงานได้ดังที่มีจุดประสงค์ไว้

3.4.1 การจัดเตรียมระบบ

ในการใช้ในลำดับแรก เราจะต้องวางตัวลูกทั้งหมดก่อนว่าต้องวางไว้จุดใดบ้าง หลังจากนั้นก็ทำการจัดวางเส้นทางทั้งไปและกลับ ดังที่ได้แสดงให้เห็นมาแล้วในหัวข้อที่ผ่านมา ซึ่งเราก็จะได้ข้อมูลขอเส้นทางไปกลับในแต่ละเส้นสำหรับตัวลูกทุกตัว อันนี้จะเป็นข้อมูลที่ตัวลูกทุกตัวจะต้องมีไว้ (เป็นของแต่ละตัว ไม่สามารถใช้ร่วมกันได้) ส่วนตัวโปรแกรมหลักของตัวลูกของทุกตัวก็จะมีการทำงานที่เหมือนกัน และจากนั้นก็ทำการ โปรแกรมสู่ตัวลูกแต่ละตัวจนครบทุกตัว

3.4.1.1 อุปกรณ์ภาคส่งตัวแม่ (Main)

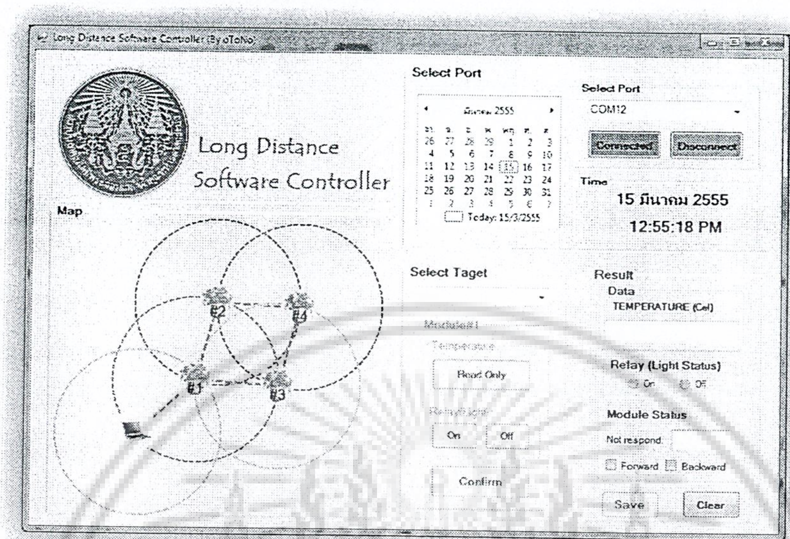
ในการทดลองนี้ผู้วิจัยจะใช้เป็นเครื่องแล็ปท็อป (Laptop) ที่ได้ทำการติดตั้งโปรแกรมใช้งานไว้เรียบร้อยแล้วเป็นส่วนที่ผู้ใช้งานต้องใช้สั่งการ กับ บอร์ด XBee-PRO Main ซึ่งจะใช้เป็นตัวแม่สำหรับติดต่อตัวลูกอื่นๆกับคอมพิวเตอร์ของผู้ใช้ ผ่านช่องสื่อสารอนุกรม (RS-232C) กรณีที่ตัวเครื่องคอมพิวเตอร์ไม่มีช่องสัญญาณอนุกรมนี้ก็สามารถนำอุปกรณ์เปลี่ยนหัวต่อแบบ USBtoRS232 มาใช้งานได้เช่นกัน แสดงดังรูปข้างล่าง



รูปที่ 13 แล็ปท็อป (Laptop)ผู้ใช้ กับ บอร์ด XBee-PRO Main (ด้านขวา)

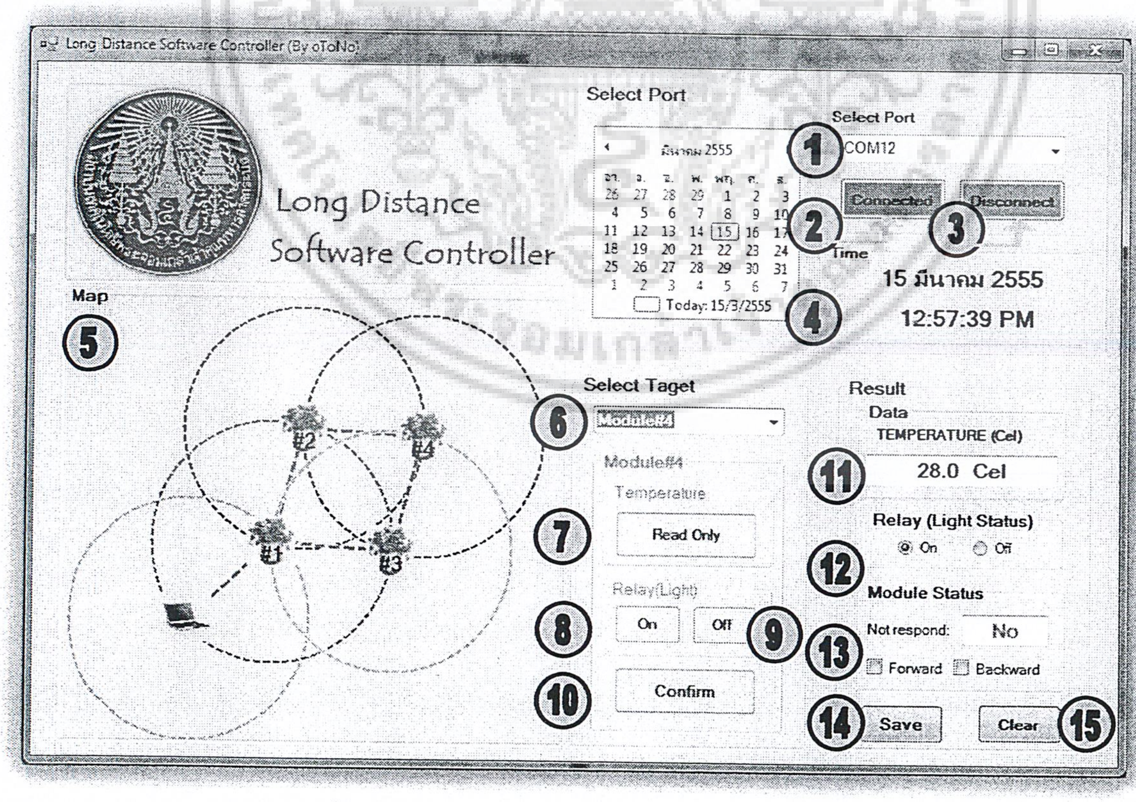
3.4.1.2 หน้าจอโปรมแกรม GUI

หน้าจอโปรแกรมควบคุม (ดังรูปที่ 14)



รูปที่ 14 หน้าจอโปรแกรมควบคุม

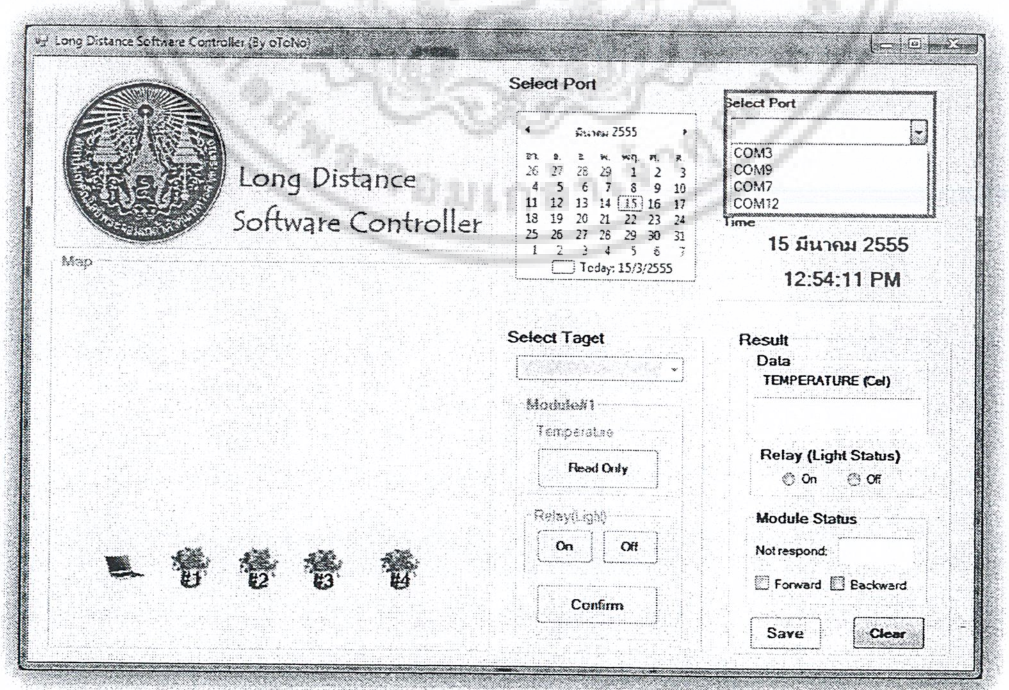
ขั้นตอนการใช้งานโปรแกรมควบคุม



รูปที่ 15 แสดงการใช้งานหน้าจอโปรแกรมควบคุม

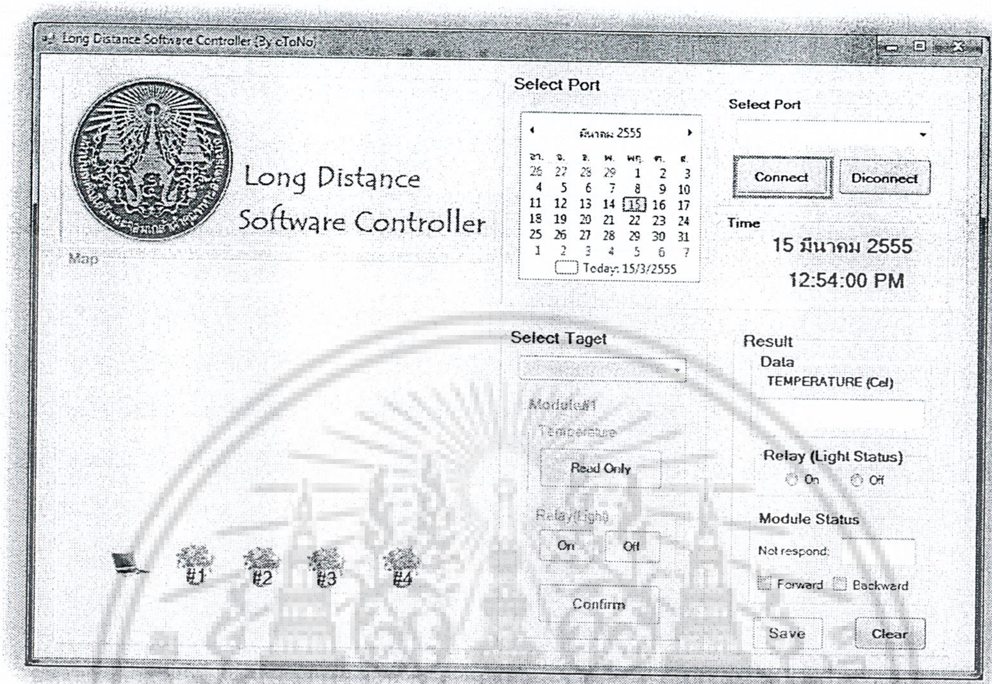
- หมายเลขที่ 1คือช่องเลือกหมายเลขพอร์ตให้ตรงกับพอร์ตอนุกรมอาร์เอส-232 (RS-232) ที่ติดต่อกับตัวส่งสัญญาณตัวMain
- หมายเลขที่ 2 คือปุ่ม Connect เพื่อเชื่อต่อกับ Comport ที่เลือกไว้
- หมายเลขที่ 3 คือปุ่ม Disconnect เมื่อไม่ต้องการเชื่อมต่อกับ
- หมายเลขที่ 4 คือส่วนที่แสดงวันที่ และเวลา ขณะที่มีการใช้งาอยู่
- หมายเลขที่ 5 คือส่วนที่ กำหนดตำแน่งและแสดงเส้นทางระหว่าโมดูลแต่ละตัว
- หมายเลขที่ 6 คือช่องที่ให้เลือกโมดูลเป้าหมายที่ต้องการอ่านอุณหภูมิและ เปิด/ปิด รีเลย์
- หมายเลขที่ 7 คือปุ่มสำหรับอ่านค่าอุณหภูมิของโมดูลเป้าหมาย
- หมายเลขที่ 8 คือปุ่มสำหรับเปิด รีเลย์ของโมดูลเป้าหมาย
- หมายเลขที่ 9 คือปุ่มสำหรับปิด รีเลย์ของโมดูลเป้าหมาย
- หมายเลขที่ 10 คือปุ่มยืนยันการเลือกจากหมายเลข7,8,9 เพื่อส่งคำสั่งไปยังโมดูลปลายทาง
- หมายเลขที่ 11 คือช่องที่แสดงค่าอุณหภูมิที่อ่านได้ของโมดูลเป้าหมาย
- หมายเลขที่ 12 คือช่องที่แสดงสถานะเปิด/ปิด รีเลย์ของโมดูลเป้าหมาย
- หมายเลขที่ 13 คือช่องที่แสดงสถานะโมดูลตัวที่เสียระหว่างไปและกลับยังโมดูลเป้าหมาย
- หมายเลขที่ 14 คือปุ่มสำหรับบันทึกค่าอุณหภูมิและสถานะเปิด/ปิด รีเลย์ของโมดูลเป้าหมาย
- หมายเลขที่ 15 คือปุ่มสำหรับล้างข้อมูลทั้งหมดที่ได้ทำมาก่อนหน้านี้

1) เปิดหน้าจอโปรแกรมควบคุม ไปที่ช่อง Select Port ให้เลือกหมายเลขพอร์ตให้ตรงกับพอร์ตอนุกรมอาร์เอส-232 (RS-232) ที่เชื่อมต่อไว้ (ดังรูปที่ 16)



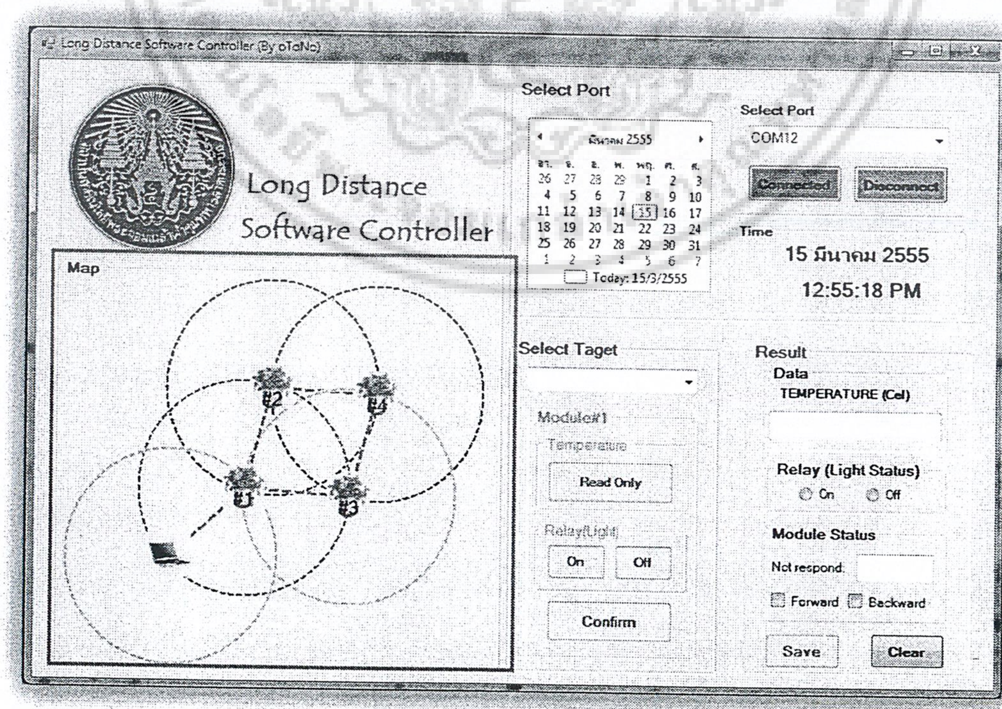
รูปที่ 16 แสดงการเลือกพอร์ตอนุกรมเพื่อเชื่อมต่อ

2) กดปุ่ม Connect เพื่อเชื่อมต่อ ระบบนี้จะทำงานก็ต่อเมื่อได้ทำการเชื่อมต่อระหว่างโปรแกรมการใช้งานกับชุดเครื่องวัดไร้สาย (ดังรูปที่ 17)



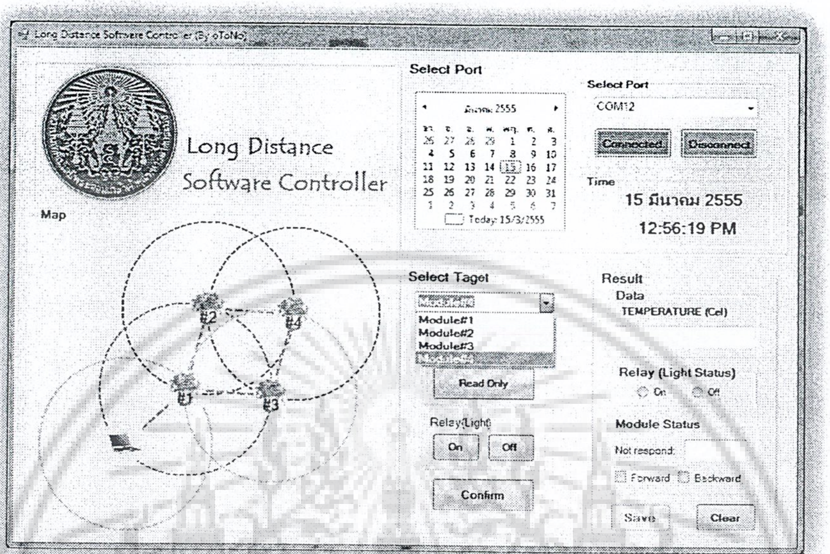
รูปที่ 17 แสดงการเชื่อมต่อเพื่อสั่งงานไมโครคอนโทรลเลอร์

3) กำหนดรูปแบบการวางตำแหน่งของโมดูลแต่ละตัวโดยสามารถกำหนดเองได้ในส่วนของMap



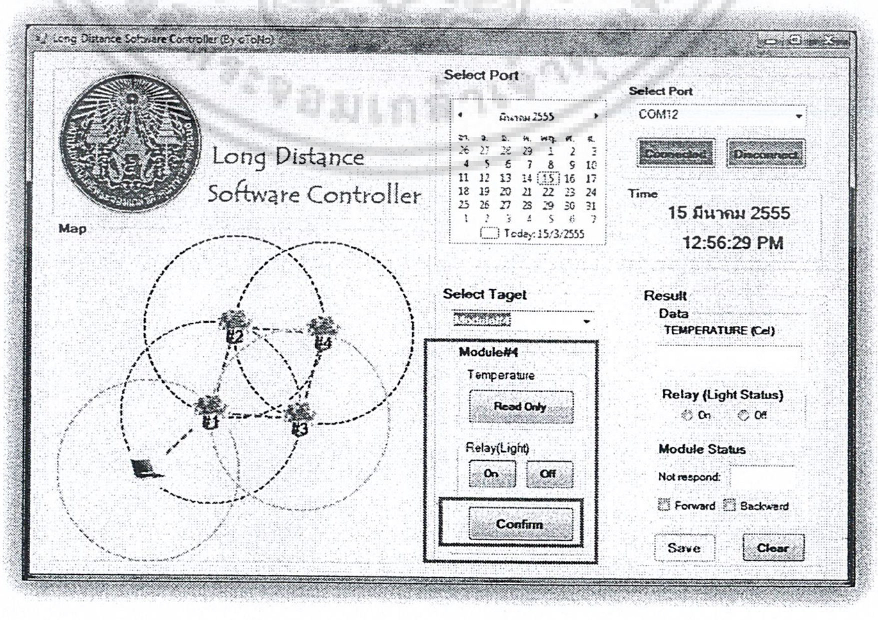
รูปที่ 18 แสดงการกำหนดรูปแบบการวางตำแหน่งของโมดูลแต่ละตัว

4.) การทดลองอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ในแต่ละโมดูลผ่านโปรแกรมควบคุม ทำการเลือกโมดูลปลายทางที่ต้องการจะทำการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ผ่านโปรแกรมควบคุม โดยไปที่ช่อง Select Target (ดังรูปที่ 19)



รูปที่ 19 แสดงการเลือกโมดูลปลายทางที่ต้องการจะทำการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์

5) หลังจากเลือกโมดูลปลายทาง (Module#4) ได้แล้วก็จะให้เลือกว่าจะทำการอ่านค่าอุณหภูมิเพียงอย่างเดียว (โดยสามารถกดที่ปุ่ม Read Only) หรือ จะทำการเปิด/ปิดรีเลย์ด้วย (โดยสามารถกดที่ปุ่ม On/Off ในช่อง Relay) และทำการเลือกปุ่ม Confirm เพื่อยืนยันการ หลังจากนั้นโปรแกรมควบคุมก็จะส่งค่าคำสั่งที่เราได้เลือกไว้แล้วผ่านทางโมดูล XBee-PRO ภาคส่ง ไปยังโมดูลปลายทางที่ต้องการ เพื่อทำการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ (ดังรูปที่ 20)

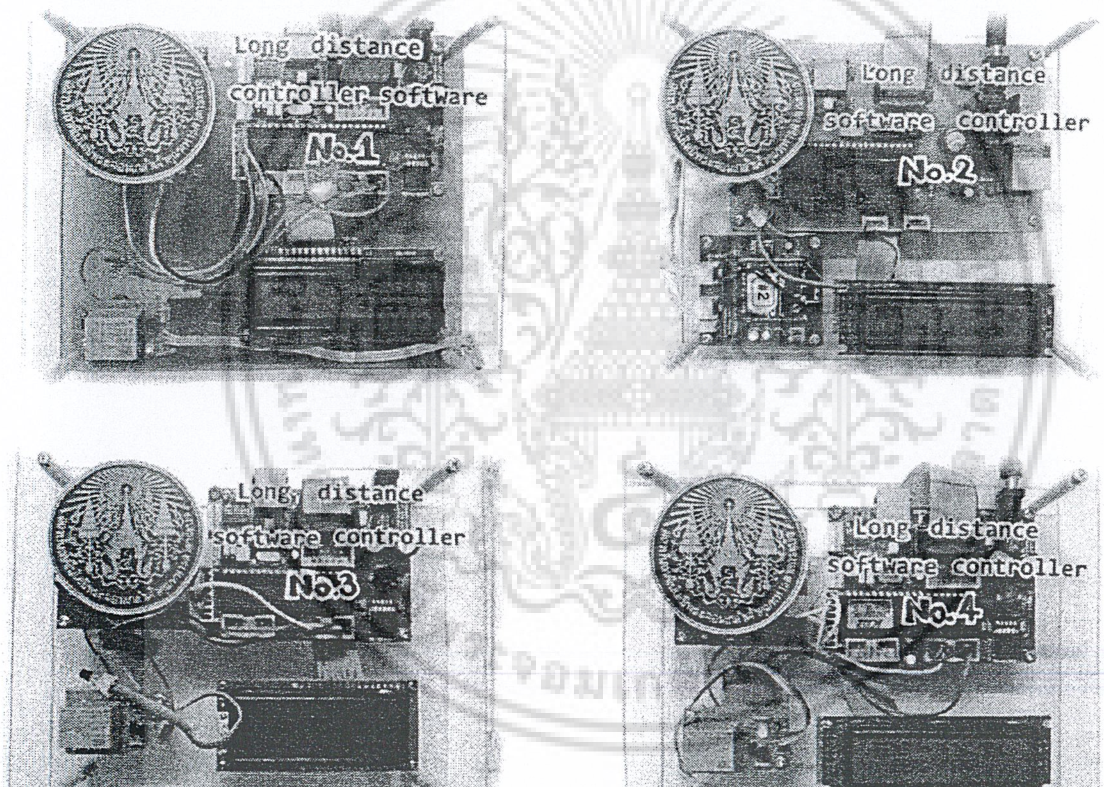


รูปที่ 20 แสดงการเลือกอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์

ที่ได้กล่าวมาเป็นส่วนของ โปรแกรมที่ติดตั้งบนคอมพิวเตอร์ของผู้ใช้และส่วนของตัวแม่ ซึ่งจะเห็นได้ว่ามีการทำงานที่ง่ายดาย เพราะผู้ใช้จะทำงานผ่านระบบ GUI นั่นเอง

3.4.1.3 อุปกรณ์ตัวลูก

ส่วนนี้ผู้วิจัยจะใช้เป็นโมเดลที่ตั้งชื่อเป็นชุดที่ 1, ชุดที่ 2, ชุดที่ 3, และชุดที่ 4 เพื่อความสะดวกของการตรวจสอบการเดินทางของข้อมูลหรือคำสั่งว่าเป็นไปตามที่กำหนดหรือไม่ ซึ่งทุกตัวได้สร้างตามที่ออกแบบไว้เหมือนกันทุกอย่าง เพียงแต่โปรแกรมของข้อมูลแต่ละตัว จะไม่เหมือนกันเท่านั้นเอง ดังแสดงดังรูปข้างล่างนี้



รูปที่ 21 โมเดลอุปกรณ์ชุดที่ 1, ชุดที่ 2, ชุดที่ 3 และชุดที่ 4

ทุกตัวก็จะมี LCD 16x2 เพื่อแสดงผลการทำงานของตนเองทำให้สามารถติดตามการเดินทางของข้อมูลได้สะดวกยิ่งขึ้น และทุกตัวก็จะมี รีเลย์กับชุดโมดูลวัดอุณหภูมิเหมือนกันทุกตัว หลังจากได้ทำการโปรแกรมลงบนตัวลูกทุกตัวแล้ว ก็พร้อมใช้งานได้ทันที

3.5 ขั้นตอนการทดสอบใช้งาน

เพื่อจะได้ทดสอบการทำงานของโปรแกรม ว่าสามารถทำงานได้ตามจุดประสงค์ จึงขอแบ่งการทดสอบออกเป็น 3 หัวข้อ โดยหัวข้อแรกเป็น การใช้งานกรณีที่โมดูลตัวลูกทุกตัวทำงานเป็นปกติ และในหัวข้อต่อมาจะทดสอบในกรณีที่มีตัวใดตัวหนึ่งเสีย ซึ่งจะเห็นได้ว่าเมื่อทดสอบในกรณีแรกจะสามารถส่งผ่านขอมูลหรือคำสั่งได้ปกติ ส่วนในหัวข้อที่สองจะเห็นได้ว่านอกจากจะส่งผ่านคำสั่งหรือข้อมูลได้แล้วยังสามารถบอกได้ว่าตัวใดเสีย ซึ่งอันนี้นับว่ามีประโยชน์มากในกรณีที่ต้องมีการเดินทางไปตรวจซ่อมตัวลูกจะได้ทราบตำแหน่งที่แน่นอนได้นั่นเอง แต่ทั้งนี้ในตอนท้ายของรายงานก็จะได้สรุปให้เห็น ข้อจำกัดที่เกิดขึ้นเพื่อการพัฒนาต่อไปในอนาคต และเช่นกันในหัวข้อที่สาม ก็จะแสดงให้เห็นถึงผลการทำงานเมื่ออุปกรณ์ตัวลูกเสียหายไปสองตัว สัญญาณไม่สามารถไปสู่ตัวเป้าหมายได้

โดยในการทดสอบนี้ก็ได้จัดรูปแบบของการวางตัวลูกเป็นไปดังแสดงในตัวหน้าจอ GUI ซึ่งเป็นลักษณะของการเชื่อมต่อโมดูล 4 ตัว โดยตัวแม่สามารถส่งผ่านคำสั่งไปสู่ตัวเป้าหมายโดยกำหนดตัวเป้าหมายเป็น ตัวที่ 4 ได้สองเส้นทาง

3.5.1 การทดลองอ่านค่าอุณหภูมิและเปิด/ปิดรีเลย์ในแต่ละโมดูลผ่านโปรแกรมควบคุม (กรณีแต่ละโมดูลไม่มีตัวเสีย)

จุดประสงค์

เพื่อตรวจสอบความถูกต้องในการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ในแต่ละโมดูลผ่านโปรแกรมควบคุม

วิธีการทดลอง

ทำการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ในแต่ละโมดูลผ่านโปรแกรมควบคุมและดูผลจากโปรแกรมควบคุมซึ่งจะแสดงในส่วนของ Result

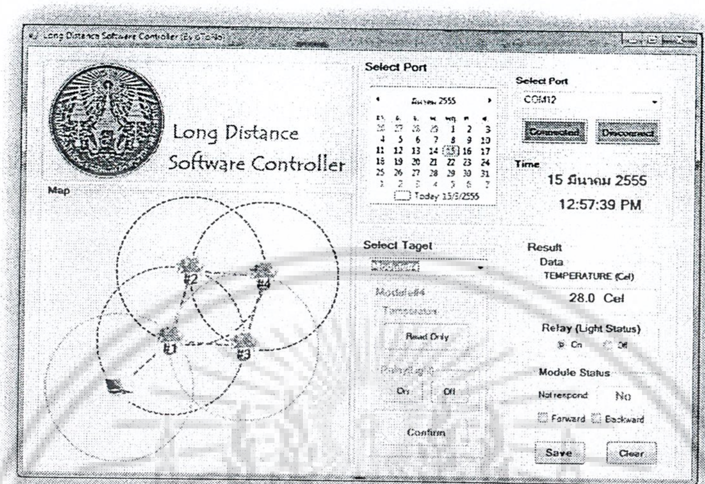
1) ทดลองโดยการ กดปุ่มRead Onlyเพื่ออ่านค่าอุณหภูมิ และกดปุ่ม On เพื่อสั่งเปิดการทำงานรีเลย์ ของโมดูลปลายทางตัวที่ 4 (Module#4) (ดังรูปที่ 22)

2) หลังจากที่ทดลองโดยการ กดปุ่ม Read Only เพื่ออ่านค่าอุณหภูมิ และกดปุ่ม On เพื่อสั่งเปิดการทำงานรีเลย์ ของโมดูลปลายทางตัวที่ 4 ในส่วนของโมดูลปลายทางก็จะมีการส่งค่าอุณหภูมิ และสถานะรีเลย์กลับมาแสดงผลผ่านโปรแกรมควบคุม

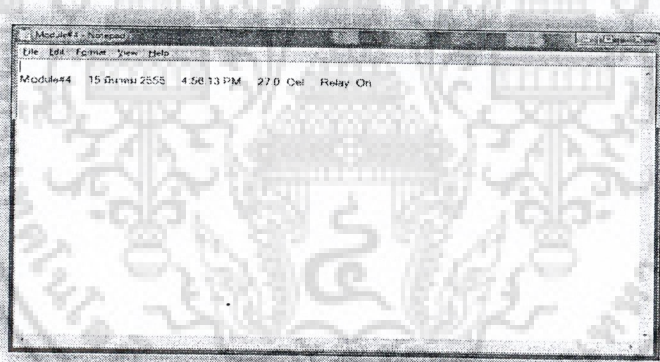
โดยค่าของอุณหภูมิที่อ่านได้จะแสดงในช่อง Temperature (27.0 °c), สถานะของรีเลย์ที่ช่อง Relay (Light Status) จะอยู่ในสถานะ On และจะแสดงโมดูลตัวที่เสียที่สแกนเจอในช่อง Module Status (ในกรณีนี้ไม่มีตัวเสียเกิดขึ้น) (ดังรูปที่ 24)

เมื่อต้องการที่จะทำการบันทึกข้อมูลอุณหภูมิและสถานะรีเลย์ที่ตรวดวัดได้ จากโมดูลปลายทาง (Module#4) ณ เวลาและวันที่ที่กดบันทึก โดยไปที่ช่อง Result ให้เลือกกดปุ่ม Save ข้อมูลที่บันทึกจะถูกเก็บเป็น Text file อยู่ใน Folder ที่สร้างเอาไว้ (ดังรูปที่ 23)

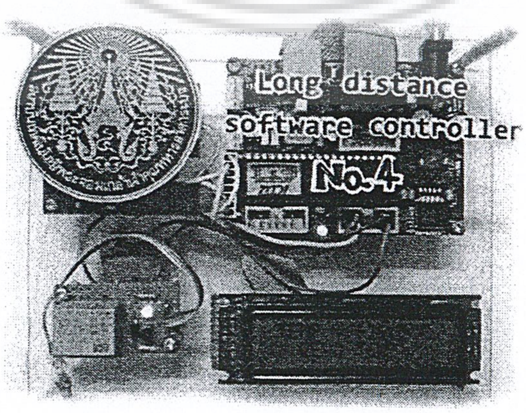
โดยในส่วนของโมดูลปลายทาง (Module#4) ที่เราต้องการอ่านค่าอุณหภูมิ และเปิดรีเลย์ เมื่อคำสั่งไปถึง ก็จะมีการแสดงค่าอุณหภูมิ ณ ขณะนั้น ผ่านทางจอ LCD และสถานะของรีเลย์ ก็จะเปลี่ยนเป็น On สามารถสังเกต ได้จากจะมีหลอด LED สีเขียวติดอยู่ (ดังรูปที่ 22)



รูปที่ 22 แสดงการค่าอุณหภูมิ และสถานะของรีเลย์ (Module#4) จากโปรแกรมควบคุม



รูปที่ 23 แสดงข้อมูลอุณหภูมิและสถานะรีเลย์ที่บันทึกได้จาก Module#4



รูปที่ 24 แสดงการค่าอุณหภูมิ และสถานะรีเลย์ ของModule#4

ผลการทดลอง

จากการทดลองเมื่อทำการอ่านค่าอุณหภูมิ และเปิดรีเลย์ของModule#4 จากโปรแกรมควบคุม โมดูลสามารถอ่านค่าอุณหภูมิ, ทำการเปิดรีเลย์ และส่งค่ากลับมาแสดงที่โปรแกรมควบคุมได้อย่างถูกต้อง

3.5.2 การทดลองอ่านค่าอุณหภูมิและเปิด/ปิดรีเลย์ในแต่ละโมดูลผ่านโปรแกรมควบคุม (กรณีที่โมดูลตัวที่ 2 เสีย)

จุดประสงค์

เพื่อตรวจสอบความถูกต้องในการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ในแต่ละโมดูลผ่านโปรแกรมควบคุมในกรณีที่โมดูลตัวที่ 2 เสียคำสั่ง, ข้อมูลอุณหภูมิและสถานะรีเลย์ยังสามารถที่จะไปยังโมดูลปลายทางและกลับมาแสดงผลที่โปรแกรมควบคุมได้หรือไม่

วิธีการทดลอง

ทำทดลองโดยปิดการทำงานของโมดูลตัวที่ 2 ไว้ ทำการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ในโมดูลปลายทาง (Module#4) ผ่านโปรแกรมควบคุมและสามรถดูผลจากโปรแกรมควบคุมซึ่งจะแสดงในส่วนของ Result

1) เปิดหน้าจอโปรแกรมควบคุม ไปที่ช่อง Select Port ให้เลือกหมายเลขพอร์ตให้ตรงกับพอร์ตอนุกรมอาร์เอส-232 (RS-232) ที่เชื่อมต่อไว้ กดปุ่ม Connect เพื่อเชื่อมต่อ ระบบนี้จะทำงานก็ต่อเมื่อได้ทำการเชื่อมต่อระหว่างโปรแกรมการใช้งานกับชุดเครื่องวัดไร้สายและกำหนดรูปแบบการวางตำแหน่งของโมดูลแต่ละตัว โดยสามารถกำหนดเองได้ ในส่วนของ Map

2) การทดลองอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ในแต่ละโมดูลผ่านโปรแกรมควบคุม (กรณีที่โมดูลตัวที่ 2 เสีย)

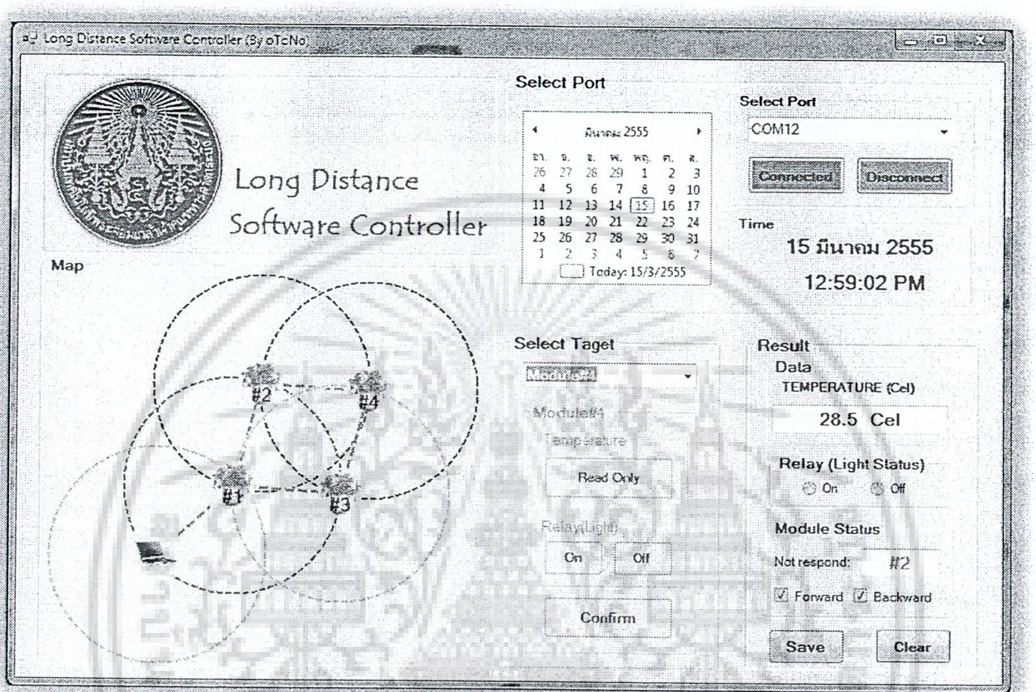
2.1) ทำการเลือกโมดูลปลายทาง (Module#4) ที่ต้องการจะทำการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ผ่านโปรแกรมควบคุม โดยไปที่ช่อง Select Target

หลังจากเลือกโมดูลปลายทาง (Module#4) ได้แล้วก็จะให้เลือกว่าจะทำการอ่านค่าอุณหภูมิเพียงอย่างเดียว (โดยสามารถกดที่ปุ่ม Read Only) หรือ จะทำการเปิด/ปิดรีเลย์ด้วย (โดยสามารถกดที่ปุ่ม On/Off ในช่อง Relay) และทำการเลือกปุ่ม Confirm เพื่อยืนยันการ หลังจากนั้นโปรแกรมควบคุมก็จะส่งค่าคำสั่งที่เราได้เลือกไว้แล้วผ่านทางโมดูล XBee-PRO ภาคส่ง ไปยังโมดูลปลายทางที่ต้องการ เพื่อทำการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์

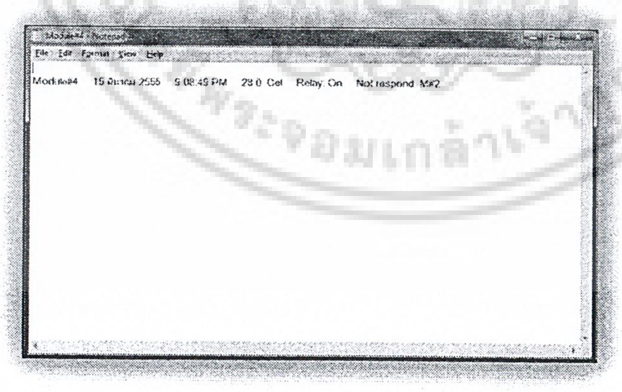
ทดลองโดยการ กดปุ่ม Read Only เพื่ออ่านค่าอุณหภูมิ และกดปุ่ม On เพื่อสั่งเปิดการทำงานรีเลย์ ของโมดูลปลายทางตัวที่ 4(Module#4)

2.2) หลังจากที่ทดลองโดยการ กดปุ่ม Read Only เพื่ออ่านค่าอุณหภูมิ และกดปุ่ม On เพื่อสั่งเปิดการทำงานรีเลย์ ของโมดูลปลายทางตัวที่ 4 ในส่วนของโมดูลปลายทางก็จะมีการส่งค่าอุณหภูมิและสถานะรีเลย์กลับมาแสดงผลผ่านโปรแกรมควบคุม

โดยค่าของอุณหภูมิที่อ่านได้จะแสดงในช่อง Temperature (28.0 °c), สถานะของรีเลย์ที่ช่อง Relay (Light Status) จะอยู่ในสถานะ On และจะแสดงโมดูลตัวที่เสียที่สแกนเจอในช่อง Module Status (ในกรณีนี้โมดูลตัวที่เสีย คือตัวที่ 2) (ดังรูปที่ 25)



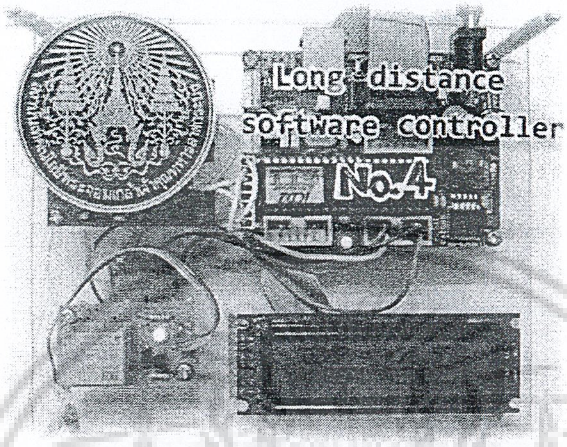
รูปที่ 25 แสดงค่าอุณหภูมิ, สถานะของรีเลย์และโมดูลตัวที่เสีย (โมดูลตัวที่2)



รูปที่ 26 แสดงข้อมูลอุณหภูมิ, สถานะรีเลย์และโมดูลตัวที่เสีย ที่บันทึกได้

2.3) เมื่อต้องการที่จะทำการบันทึกข้อมูลอุณหภูมิและสถานะรีเลย์ที่ตรวจวัดได้ จากโมดูลปลายทาง (Module#4) ณ เวลาและวันที่ที่กดบันทึก โดยไปที่ช่อง Result ให้เลือกกดปุ่มSaveข้อมูลที่บันทึกจะถูกเก็บเป็น Text file อยู่ใน Folder ที่สร้างเอาไว้ (ดังรูปที่ 26)

2.4) โดยในส่วนของโมดูลปลายทาง (Module#4) ที่เราต้องการอ่านค่าอุณหภูมิ และเปิดรีเลย์ เมื่อคำสั่งไปถึง ก็จะมีการแสดงค่าอุณหภูมิ ณ ขณะนั้น ผ่านทางจอ LCD และสถานะของรีเลย์ ก็จะเปลี่ยนเป็น On สามารถสังเกต ได้จากจะมีหลอดLED สีเขียวติดอยู่ (ดังรูปที่ 27)

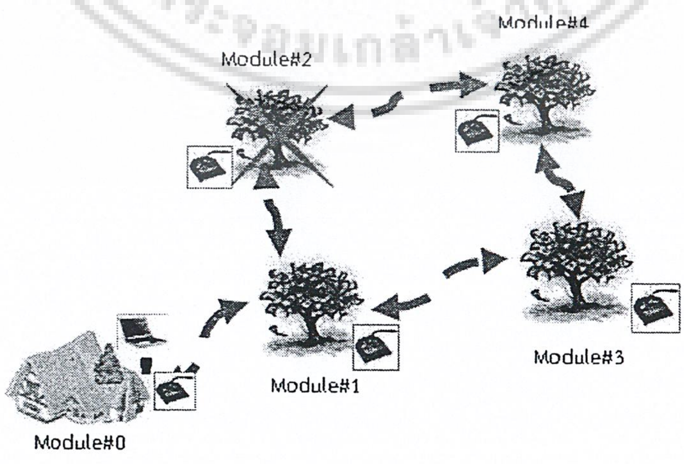


รูปที่ 27 แสดงการค่าอุณหภูมิ และสถานะรีเลย์ ของModule#4

ผลการทดลอง

จากการทดลองเมื่อทำการอ่านค่าอุณหภูมิ และเปิดรีเลย์ของModule#4 จากโปรแกรมควบคุม โดยในกรณีที่โมดูลตัวที่ 2 เสีย โมดูลสามารถอ่านค่าอุณหภูมิ, ทำการเปิดรีเลย์ และส่งค่ากลับมาแสดงที่โปรแกรมควบคุมได้อย่างถูกต้อง

เนื่องจากโมดูลแต่ละตัวจะทำหน้าที่ส่งต่อคำสั่ง เมื่อรับคำสั่งมาแล้วรู้ว่าไม่ใช่คำสั่งที่จะทำการอ่านค่าจากโมดูลตัวเอง ก็จะจัดรูปแบบของคำสั่งใหม่แล้วส่งคำสั่งนี้ต่อไปสู่โมดูลตัวอื่นตามเส้นทางที่เลือกจากตารางเส้นทางที่มีอยู่ประจำในแต่ละโหนด โดยจะพิจารณาตามลำดับความสำคัญของเส้นทาง โดยเลือกเส้นทางให้เป็นเส้นทางหลักและเส้นทางสำรอง (ดังรูปที่ 28)



รูปที่ 28 แสดงการพิจารณาเส้นทางใหม่เมื่อมีโมดูลบางตัวเสีย

3.5.3 การทดลองอ่านค่าอุณหภูมิและเปิด/ปิดรีเลย์ในแต่ละโมดูลผ่านโปรแกรมควบคุม (กรณีที่โมดูตัวที่ 2 และตัวที่ 3 เสีย)

จุดประสงค์

เพื่อตรวจสอบความถูกต้องในการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ในแต่ละโมดูลผ่านโปรแกรมควบคุมในกรณีที่โมดูลตัวที่ 2 และตัวที่ 3เสียคำสั่ง, ข้อมูลอุณหภูมิและสถานะรีเลย์ยังสามารถที่จะไปยังโมดูลปลายทางและกลับมาแสดงผลที่โปรแกรมควบคุมได้หรือไม่

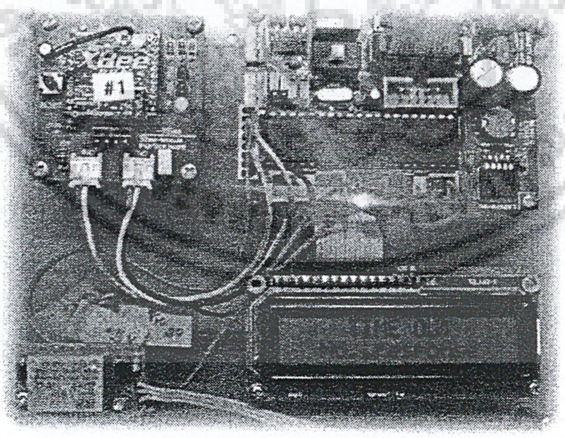
วิธีการทดลอง

ทำทดลองโดยปิดการทำงานของโมดูลตัวที่ 2 และตัวที่ 3 ไว้แล้วทำการอ่านค่าอุณหภูมิ และเปิด/ปิดรีเลย์ในโมดูลปลายทาง (Module#4) ผ่านโปรแกรมควบคุมและสามรถดูผลจากโปรแกรมควบคุมซึ่งจะแสดงในส่วนของ Result

1) เลือกหมายเลขพอร์ตที่จะทำการเชื่อมต่อกำหนดรูปแบบการวางตำแหน่งของโมดูลแต่ละตัว และเลือกหมายเลขโมดูลปลายทาง (Module#4)

2) หลังจากที่ทดลองโดยการ กดปุ่ม Read Only เพื่ออ่านค่าอุณหภูมิ และกดปุ่ม On เพื่อสั่งเปิดการทำงานรีเลย์ ของโมดูลปลายทางตัวที่ 4

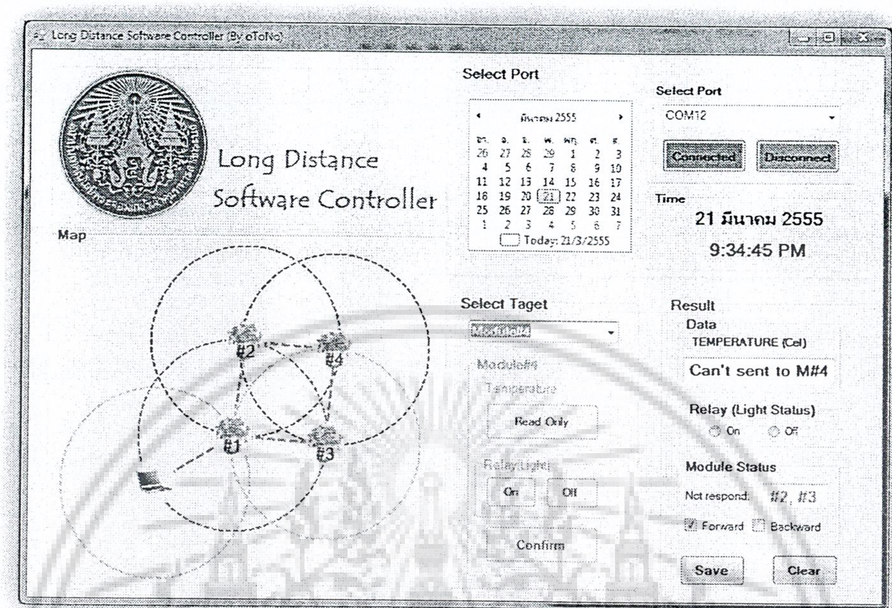
แต่ในกรณีนี้ โมดูลตัวที่ 2 และตัวที่ 3 เสีย จึงทำให้โมดูลตัวตัวที่ 1 ไม่สามารถที่จะสแกนเจอโมดูลตัวที่ 2 และ 3 ได้ โดยในระหว่างที่โมดูลตัวที่ 1 สแกนหาโมดูลตัวแรกจะรอการตอบกลบมาเป็นเวลาประมาณ 5 วินาที เมื่อไม่มีการตอบกลับมา ก็จะเกิด Time Out (หมดเวลา) และทำการสแกนหาโมดูลตัวถัดไป (ดังรูปที่ 29)



รูปที่ 29 แสดงโมดูลตัวที่ 1 สแกนหาโมดูลตัวที่ 2 และ 3 จนหมดเวลา

3) เนื่องจากโมดูลตัวที่ 2 และ 3 เสีย ทำให้คำสั่งจากโมดูลตัวที่ 1 ไม่สามารถเดินทางไปยัง โมดูลปลายทาง (Module#4) ได้ โปรแกรมควบคุมจึงแสดงผลที่ช่อง Module Status คือโมดูลตัวที่ 2และ 3

เสียอีกทั้งยังแจ้งให้ผู้ใช้งานทราบว่าไม่สามารถ อ่านค่าอุณหภูมิและสถานะรีเลย์ของโมดูลปลายทางได้ (ดังรูปที่ 30)

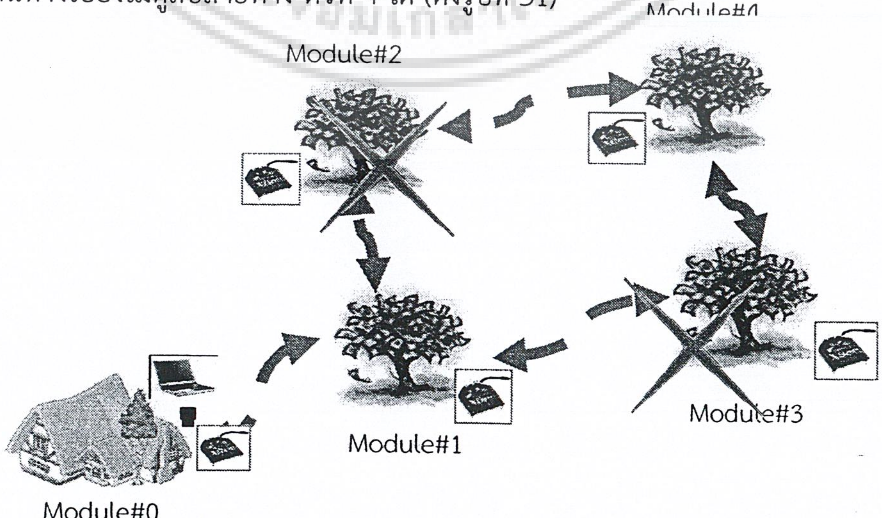


รูปที่ 30 แสดงค่าอุณหภูมิ, สถานะของรีเลย์และโมดูลตัวที่เสีย (ตัวที่ 2 และ 3)

ผลการทดลอง

จากการทดลองเมื่อทำการอ่านค่าอุณหภูมิ และเปิดรีเลย์ของ Module#4 จากโปรแกรมควบคุม โดยในกรณีที่โมดูลตัวที่ 2 และ 3 เสีย จะไม่สามารถอ่านค่าอุณหภูมิและสถานะรีเลย์ของโมดูลปลายทางได้

เนื่องจากการเดินทางจากโมดูลตัวที่ 1 ไปยังโมดูลปลายทาง ตัวที่ 4 จะต้องเดินทางผ่านโมดูลตัวที่ 2 หรือโมดูลตัวที่ 3 ตัวใดตัวหนึ่ง แต่ในกรณีนี้โมดูลทั้งสองตัว เสียพร้อมกัน จึงส่งผลให้โมดูลตัวที่ 1 ไม่สามารถเดินทางไปยังโมดูลปลายทาง ตัวที่ 4 ได้ (ดังรูปที่ 31)



รูปที่ 31 แสดงการพิจารณาเส้นทางใหม่เมื่อมีโมดูลบางตัวเสีย

บทที่ 4
อภิปรายผลการวิจัยและวิจารณ์

จากผลการทดสอบนั้น สามารถทำงานได้อย่างที่ตั้งจุดประสงค์ไว้ทำให้มีความมั่นใจว่าผลงานนี้จะสามารถนำไปสู่การใช้งานจริงกับระบบวิทยุ ดังที่ได้กล่าวไว้ นั่นหมายความว่า ปัญญหาใดๆที่จะเกิดขึ้นในการใช้งานจริง ก็จะต้องทำการแก้ไขต่อไปอีก แต่ปัญหาก็จะน้อยลงมาก เพราะในด้านขั้นตอนการทำงานโปรแกรมนั้น จากผลการทดสอบก็สามารถทดสอบทำได้แล้ว เพียงแต่เป็นการทดสอบกับการส่งในระยะใกล้ ซึ่งหากต้องในใช้งานในอุปกรณ์วิทยุจริงๆ ปัญหาใหม่ๆต่างๆ อาจเกิดขึ้นได้ เช่นการทับซ้อนของความถี่ที่ใช้งานเป็นต้น ซึ่งต้องทำการแก้ไขต่อไป เป็นกรณีๆ ไปนั้นเอง

บทที่ 5
สรุปและข้อเสนอแนะ

จากการทดสอบแล้วนั้น จะเห็นได้ว่าตัวเครื่องสามารถทำงานได้ดังจุดประสงค์และจะทำให้เกิดความคล่องตัวมากขึ้นไปอีกหากทุกส่วนถูกนำมาวางบนแผ่นวงจรพิมพ์เดียวกัน ซึ่งเป็นงานที่จะต้องมีการพัฒนาต่อไปนั่นเอง และถึงแม้ตัวเครื่องจะสามารถทำงานได้ดังจุดประสงค์ของการออกแบบซอฟร์แวร์ไว้แล้ว แต่โครงการวิจัยนี้ ยังไม่ได้นำไปสู่การใช้งานกับอุปกรณ์ที่รับส่งด้วยวิทยุอย่างจริงจังที่จะต้องมีการส่งด้วยระยะที่ไกลและในพื้นที่ลักษณะต่างๆ ซึ่งสภาวะแวดล้อมจะแตกต่างกันโดยสิ้นเชิง เช่น อาจต้องมีสัญญาณรบกวนช่องความถี่ที่ใช้ในการส่ง เหล่านี้เป็นต้น ดังนั้นผู้วิจัยจึงได้นำเสนองานวิจัยต่อยอดงานวิจัยชิ้นนี้ต่อไปโดยเสนอขอทุนจาก กระทรวงวิทยาศาสตร์ (วช.) ซึ่งหากได้รับการสนับสนุนก็จะเป็นการนำผลงานนี้ไปสู่การใช้งานได้อย่างจริงจังต่อไป

บรรณานุกรม

[1] ZigBee เบื้องต้น
http://www.bloggang.com/viewblog.php?id=zol&date=11-02-2011&group=24&gblog=11

[2] Xbee Basic Configuration in Network Application
http://www.thaieasyelec.com/Embedded-Electronics-Application/Xbee-Basic-Configuration-in-Network-Application.html, [ออนไลน์]

[3] นคร ภักดีชาติ และคณะ. คู่มือการทดลอง dsPIC Microcontroller เบื้องต้นด้วยโปรแกรมภาษาซี . กรุงเทพ : บริษัท อินโนเวตีฟ เอ็กพอริเมนต์ จำกัด

[4] Serial Port
http://riverplusblog.com/2011/06/18/serial-communication/

[5] DS1820 Digital Thermometer Replacements datasheet at
www.datasheetcatalog.org/datasheet/maxim/DS1820-DS1820S.pdf, [ออนไลน์]

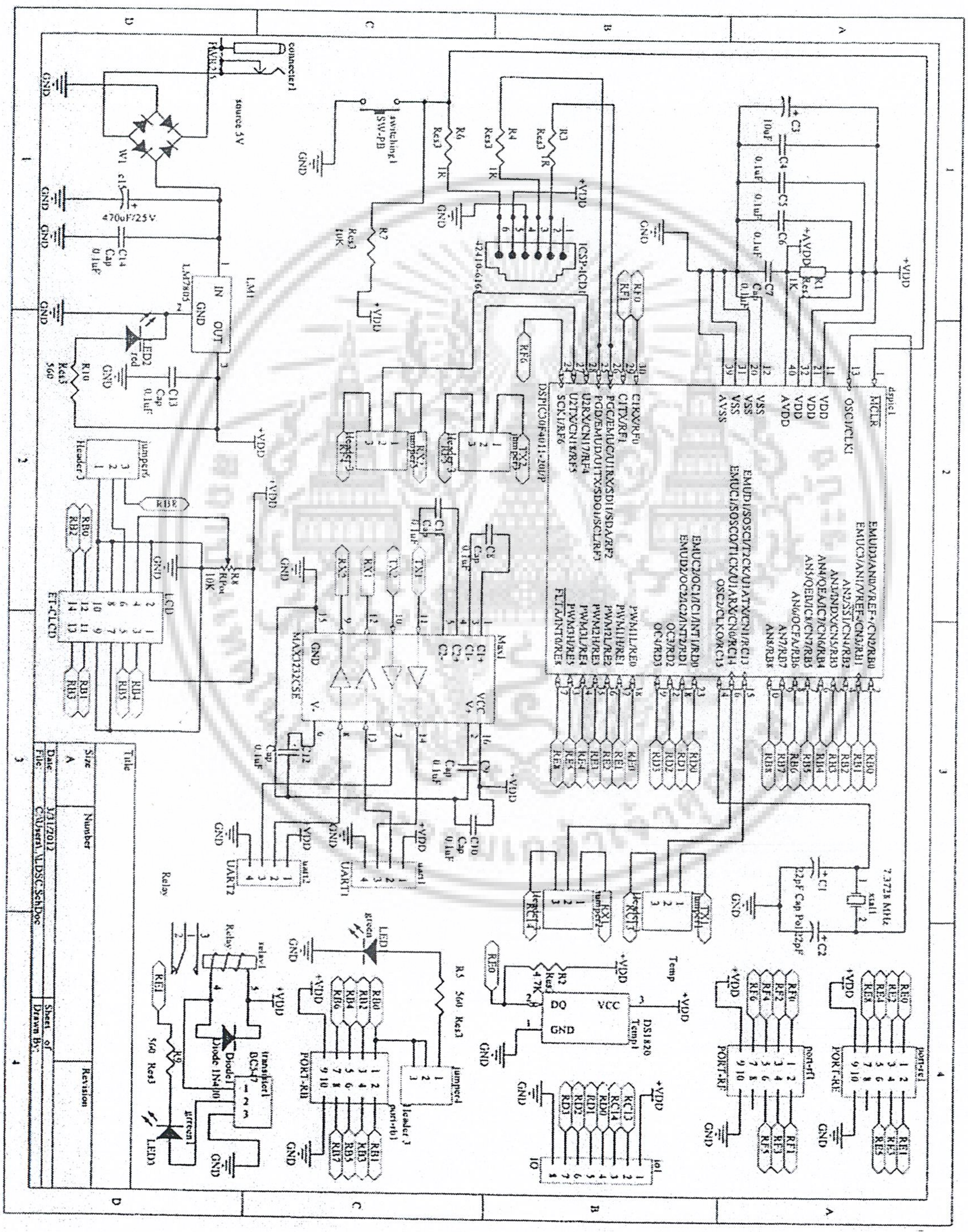
[6] ZigBee IEEE 802.15.4 Standard datasheet at
http://www.thaitelecomkm.org/TTE/topic/attach/Bluetooth_and_Zigbee/index.php, [ออนไลน์]

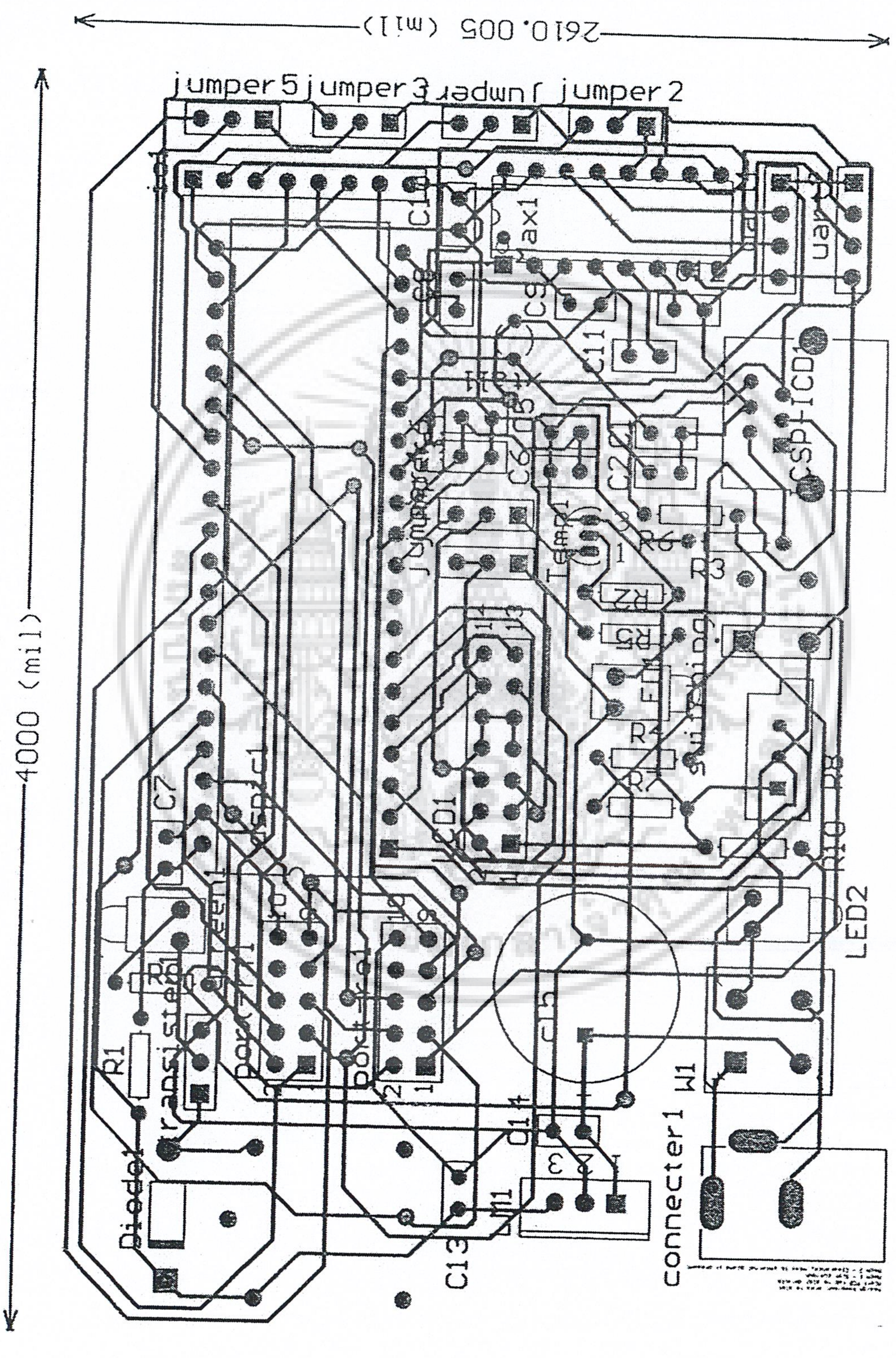
[7] ZigBee Technology and IEEE 802.15.4 Standard datasheet at
http://zigbeeyoyo.blogspot.com/2007/08/zigbee.html, [ออนไลน์]

[8] Zigbee and Xbee BASIC
http://www.thaieasyelec.com/index.php?lay=show&ac=article&Id=538707975&Ntype=1, [ออนไลน์]

[9] สัจจะ จรัสรุ่งรวีวร. 2550. คู่มือ Visual C# 2005 ฉบับสมบูรณ์. ครั้งที่ 1. นนทบุรี: ไอดีซี อินโฟ ดิสทริบิวเตอร์ เซ็นเตอร์

ภาคผนวก ก.
วงจรรวมของโครงงานบางส่วน (ทั้งหมดอยู่ในแผ่นข้อมูลแล้ว)




2610.005 (mil)
4000 (mil)
jumper5
jumper3
jumper2
LED2
connecter1
Diode1
C13
C7
R1
R2
R3
R4
R5
R6
C11
C15
C14
W1
D8

ภาคผนวก ข.

โปรแกรมควบคุม

ส่วนของโปรแกรมบนตัวลูก แสดงเฉพาะบางส่วน (โปรแกรมสมบูรณ์อยู่ในแผ่นข้อมูลแล้ว)

```
/*
 * File       : main.c
 * Purpose    : Module Receive/Send Data/Command Test
 * Author     : ATTASIT LASAKUL
 * Company    : KMITL.
 * WWW        : www.kmitl.ac.th
 * Date       : 08/03/2012
*/
//----------------------------------------------:Includes
#include <p30f4011.h>    // generic header file for dsPIC
#include "uart.h"        // uart module
#include <string.h>      // String.h standard header
#include "stdio.h"
//-----------------------------------
/*
                    คำสั่ง
   -----------
  |  0x0F     |  <------- โค๊ดเริ่มต้น
   -----------
   -----------
  |  0x00     |  <------- ระหัสบอกเป็นคำสั่ง
   -----------
  | Send ID   |  <------- ID ของผู้ส่งคำสั่งมา
   -----------
  | Rev ID    |  <------- ID ของตัวโมดูลส่งผ่านคำสั่ง
   -----------
  | Desire ID |  <----- (ตัวนี้เป็นตัวที่ต้องการอ่านอุณหภูมิ)
   -----------
  |  Relay0   |  <----- สี่บิทบนใช้กำหนดสถานะรี เลย์และสี่บิทล่างบอกว่าจะอ่านอย่างเดียวหรือไม่----------
   -----------
  |  ID       |  <------ เป็นจุดเริ่มเลข ID ของตัวที่พบว่าเสีย
   -----------
  |  ID       |
   -----------
  |  0xFF     |  <----- จบ
   -----------
```

ข้อมูลกลับ

```
 -----------
| 0x0F    | <------- โค๊ดเริ่มต้น
 -----------
| 0x01    | <--------- โค๊ดบอกเป็น ข้อมูล
 -----------
| ID ผู้ส่ง       | <-----------หมายเลขโมดูลที่ส่งข้อมูลมา
 -----------
 -----------
| ID ผู้ผ่าน      | <--------- หมายเลขโมดูลที่ส่งผ่านข้อมูลต่อสู่ตัวแม่
 -----------
| Return?  | <--------  ข้อมูลบอกการย้อนกลับ
 _______________
| TEMP(f)   | <------- ID ของผู้ที่จะถูกส่งต่อไป
 -----------
| TEMP(b)   | <------- ID ของตัวที่ต้องการอ่านค่าอุณหภูมิ
 -----------
 -----------
| Relay0   | <----- ลี่บิทล่างบอกว่าจะแสดงสถานะอย่างเดียว-----------
 -------------
| ID      | <------- แสดงหมายเลขตัวที่เสีย
 -----------
| 0xFF    | <----- จบ
 -----------
//--------------------------------
*/
// Setup Configuration For ET-BASE dsPIC30F4011
_FOSC(CSW_FSCM_OFF & XT_PLL16);                                    //
Disable Clock Switching,Enable Fail-Salf Clock
                    // Clock Source = Primary XT + (PLL x 16)
_FWDT(WDT_OFF);
              // Disable Watchdog
_FBORPOR(PBOR_OFF & PWRT_64 & MCLR_EN);              // Disable Brown-Out ,Power
ON = 64mS,Enable MCLR
_FGS(CODE_PROT_OFF);                                 // Code Protect OFF
//------------------------------------------:Calc Baud Rate Generator
#define Fcy           29491200.0          // Fosc 7.3728MHz*16/4
#define BAUD_RATE       9600.0              // Baud Rate 9600 bps
#define BAUD_RATE_GEN   (Fcy/(16.0*BAUD_RATE))-1  // Baud Rate Generator
#define KeyEnter        13                  // Key Code
```

```
/* End Configuration For ET-BASE dsPIC30F4011 */
//----  (Relay#0)
#define TRIS_Rly0 TRISEbits.TRISE1
#define Relay0  LATEbits.LATE1
//---- LED (Relay#1)
#define TRIS_Rly1 TRISEbits.TRISE2
#define Relay1  LATEbits.LATE2
//---- LED ---
#define TRIS_LED_Rly0 TRISEbits.TRISE3
#define LED_Relay0  LATEbits.LATE3
//---- LED (Relay#1)
#define TRIS_LED_Rly1 TRISEbits.TRISE4
#define LED_Relay1  LATEbits.LATE4


// Character LCD Interface Pins
#define TRIS_DATA_PIN_4     TRISBbits.TRISB0        // Direction D4
#define TRIS_DATA_PIN_5     TRISBbits.TRISB1        // Direction D5
#define TRIS_DATA_PIN_6     TRISBbits.TRISB2        // Direction D6
#define TRIS_DATA_PIN_7     TRISBbits.TRISB3        // Direction D7
#define TRIS_RS             TRISBbits.TRISB4        // Direction RS
#define TRIS_E              TRISBbits.TRISB5        // Direction E

#define DATA_PIN_4          LATBbits.LATB0          // RB0 = D4 LCD
#define DATA_PIN_5          LATBbits.LATB1          // RB1 = D5 LCD
#define DATA_PIN_6          LATBbits.LATB2          // RB2 = D6 LCD
#define DATA_PIN_7          LATBbits.LATB3          // RB3 = D7 LCD
#define RS_PIN              LATBbits.LATB4          // RB4 = RS LCD
#define E_PIN               LATBbits.LATB5          // RB5 = E  LCD

/* Display ON/OFF Control */
#define DON                 0x0F                    // Display on
#define DOFF                0x0B                    // Display off
#define CURSOR_ON           0x0F                    // Cursor on
#define CURSOR_OFF          0x0D                    // Cursor off
#define BLINK_ON            0x0F                    // Cursor Blink
#define BLINK_OFF           0x0E                    // Cursor No Blink

/* Cursor or Display Shift */
#define SHIFT_CUR_LEFT      0x13            // Cursor shifts to the left
#define SHIFT_CUR_RIGHT     0x17            // Cursor shifts to the right
```

```
#define SHIFT_DISP_LEFT     0x1B                    // Display shifts to the left
#define SHIFT_DISP_RIGHT    0x1F                    // Display shifts to the right

/* Function Prototypes */

void Initial_4bitLCD(void);                         // Initial LCD Interface
void SetCGRamAddr(unsigned char);
void SetDDRamAddr(unsigned char);                   // Set Cursor Address
void WriteCmdLCD(unsigned char);                    // Write Command
void WriteDataLCD(unsigned char);                   // Write Data
void PutsLCD(unsigned char*);                       // Print Message
void Delay_tW_LCD(void);                            // Enable Pulse Delay
void Busy_LCD(void);                                // Wait LCD Busy
void Delay(unsigned long int);                      // Delay Time Function
void PutsLCD2line(int,char[]);
void input(unsigned char, unsigned char);
void Show_data(unsigned char *show);
unsigned char Target(void);
unsigned char Transfer_command(void);
unsigned char  Transfer_data(void);
void Uart1_hex(unsigned char[],unsigned char);
unsigned char scan_check(unsigned char[],unsigned char[]);
unsigned char wait(void);
unsigned char tempresbit(void);
void tempwrbyte(unsigned char dat);
char temprdbyte(void);
char rdtemp(void);
/*  Function RS-232 */
void Uart1_PrintStr(unsigned char *str_uart);
void Delay_MS(unsigned int ms);
void init_uart1(void);
void init_uart2(void);
//--- For DS1820
#define TRIS_DS1820     TRISEbits.TRISE0                        // Direction E0
#define DOUT_DS1820     LATEbits.LATE0          // OUT
#define DIN_DS1820      PORTEbits.RE0           // IN
char buf_Tempc[20]={'\0','\0','\0','\0','\0'            // Set buffer for display Temp
                    ,'\0','\0','\0','\0','\0'
                    ,'\0','\0','\0','\0','\0'
                    ,'\0','\0','\0','\0','\0'};
```

```
//-----------------------:Global variables
unsigned char Buf[40];            // Received Command/data is stored in array Buf
unsigned char Buf_ch[40];

unsigned char module_check[4]={0x0f,0x00,0x00,0xff};
unsigned char lose_module[5]={0xff,0xff,0xff,0xff,0xff};//เก็บ โมดูลที่เสียของแต่ละช่วง ชั่วคราว
unsigned char tempb;
unsigned char tempf;
unsigned char r_dat;

//#define MY_ID          0x01   //กำหนดเลขหมายตัวเองหมายเลข'1'
#define MY_ID            0x02   //กำหนดเลขหมายตัวเองหมายเลข'2'
//#define MY_ID          0x03   //กำหนดเลขหมายตัวเองหมายเลข'3'
//#define MY_ID          0x04   //กำหนดเลขหมายตัวเองหมายเลข'4'
//------ Module #2 ---------------------------------//
unsigned char going_list[5][3] = {          0x00,0x00,0x00,
                                  0x00,0x00,0x00,
                                  0x00,0x00,0x00,
                                  0x00,0x00,0x00,
                        0x00,0x00,0x00};  //ใส่โมดูลที่ต้องส่งผ่านไปสู่ตัวเป้าหมาย(ไม่ใช่โมดูลข้างเคียง)
                                          //กรณีนี้คือ #4 เท่านั้น อย่าลืมปิดท้ายด้วย 0xff

unsigned char return_list[3] = { 0x01,0xff,0xff};   //ใส่โมดูลที่ต้องส่งผ่านข้อมูลไปสู่ตัวแม่(ไม่ใช่โมดูล
ข้างเคียง) แต่หากเป็นตัวติดกับตัวแม่ก็ใส่ 0x00 อย่าลืมปิดท้ายด้วย 0xff
unsigned char neighborhood_list[3] = {0x04,0xff,0xff}; //ใส่ตัวโมดูลข้างเคียงส่งคำสั่ง  (ใส่ 0xff ไว้
เป็นระหัสท้ายข้อมูลด้วย)
/*
//------ Module #3 ---------------------------------//
unsigned char going_list[5][3] = {          0x00,0x00,0x00,
                                  0x00,0x00,0x00,
                                  0x00,0x00,0x00,
                                  0x00,0x00,0x00,
                        0x00,0x00,0x00};  //ใส่โมดูลที่ต้องส่งผ่านไปสู่ตัวเป้าหมาย(ไม่ใช่โมดูลข้างเคียง)
                                          //กรณีนี้คือ #4 เท่านั้น อย่าลืมปิดท้ายด้วย 0xff

unsigned char return_list[3] = { 0x01,0xff,0xff};   //ใส่โมดูลที่ต้องส่งผ่านข้อมูลไปสู่ตัวแม่(ไม่ใช่โมดูล
ข้างเคียง) แต่หากเป็นตัวติดกับตัวแม่ก็ใส่ 0x00 อย่าลืมปิดท้ายด้วย 0xff
unsigned char neighborhood_list[3] = {0x04,0xff,0xff}; //ใส่ตัวโมดูลข้างเคียงส่งคำสั่ง  (ใส่ 0xff ไว้
เป็นระหัสท้ายข้อมูลด้วย)
//------ Module #4 ---------------------------------//
```

```
unsigned char going_list[5][3] = {           0x00,0x00,0x00,
                                           0x00,0x00,0x00,
                                           0x00,0x00,0x00,
                                           0x00,0x00,0x00,
                                  0x00,0x00,0x00};  //ใส่โมดูลที่ต้องส่งผ่านไปสู่ตัวเป้าหมาย(ไม่ใช่โมดูลข้างเคียง)
                                                 //กรณีนี้คือ #4 เท่านั้น อย่าลืมปิดท้ายด้วย 0xff

unsigned char return_list[3] = { 0x02,0x03,0xff};  //ใส่โมดูลที่ต้องส่งผ่านข้อมูลไปสู่ตัวแม่(ไม่ใช่โมดูลข้างเคียง) แต่หากเป็นตัวติดกับตัวแม่ก็ใส่ 0x00 อย่าลืมปิดท้ายด้วย 0xff
unsigned char neighborhood_list[3] = {0xff,0xff,0xff}; //ใส่ตัวโมดูลข้างเคียงส่งคำสั่ง  (ใส่ 0xff ไว้เป็นระหัสท้ายข้อมูลด้วย)
*/
//==============================================================
//------------------------:Main Program:----------------------
//==============================================================
int main(void)
{
  unsigned char x;
        Delay(2000000);
        TRIS_DS1820  = 1;          //กำหนดเป็น อินพุท สำหรับอุณหภูมิ
   Initial_4bitLCD();              // Initial LCD 4 Bit Interface
//กำหนด รีเลย์ตัวที่หนึ่งและสองให้ใช้ช่องพอร์ท E2,E3
 TRIS_Rly0 = 0;            // Set direction control RF2 Output
 Relay0 = 1;               // Clear RF2 กำหนดให้ตัวรีเลย์อยู่ในสภาวะ OFF ก่อน
 TRIS_Rly1 = 0;            // Set direction control RF3 Output
 Relay1 = 1;               // Clear RF3 กำหนดให้ตัวรีเลย์อยู่ในสภาวะ OFF ก่อน
 TRIS_LED_Rly0 = 0;
 TRIS_LED_Rly1 = 0;
 LED_Relay0 = 1;
 LED_Relay1 = 1;
 //เริ่มแสดงผลจอ LCD
  PutsLCD2line(0," PROGRAM START  ");
  PutsLCD2line(1,"  WAITTING..... ");
   Delay(5000000);
   Relay0 = 0;
   Relay1 = 0;
   init_uart1();           // Initialize the UART1 9600 bps
  tempb=0;
  tempf=0;
  Delay(3000000);
```

```
        //====================== START HERE ==========================
Loop:
  for(;;)
        {
                init_uart1(); //just for sure
                PutsLCD2line(0,"WAIT FOR COMMAND");
                PutsLCD2line(1," Wait..(check) ");
      input(0x0f,0xff);  // input (หัวข้อมูล, ท้ายข้อมูล)
       PutsLCD2line(0," COMMING SIGNAL ");
                PutsLCD2line(1," === ON AIR === ");
                    Delay(3000000);
  ////////////รับมาแยกแยะเพื่อทำงานในแต่ละกรณี///////////////
 if (Buf[2]==MY_ID && Buf[3]==0xff){
                                PutsLCD2line(0," CHECK COMMAND ");
                                PutsLCD2line(1," RECEIVED..... ");
                                Delay(5000000);

      Uart1_hex(Buf,0xff);  //ส่งคำสั่งตรวจเช็ดสะท้อนกลับคืนไปและลบข้อมูลคำสั่งตรวจสอบ   Buf
                                  //แสดงผลรอรับคำสั่งหรือข้อมูล
                                PutsLCD2line(0,"WAIT FOR COMMAND");
                                PutsLCD2line(1," Wait..(com/dat)");
                  for(x=0;x<40;x++) Buf[x]=0x00; //ล้างบัฟเฟอร์รอรับข้อมูลต่อไป
                  input(0x0f,0xff);  //รอรับคำสั่งหรือข้อมูลเข้ามาอีกครั้ง
                        if(Buf[4]==MY_ID)
                             {
if (Target()== 0x00) goto Loop;  //เป็นตัวสุดท้ายเป้าหมาย ฉะนั้นต้องอ่านอุณหภูมิกลับทันที หากกลับ
ไม่ได้ก็ให้ไปเริ่มต้นใหม่
                          }else if( Buf[1]==0x00 && Buf[3]== MY_ID)
                                {
if (Transfer_command()== 0x00) //หากส่งคำสั่งต่อไปไม่ได้ก็ให้กลับคืนสู่ตัวแม่
                        {
if(Transfer_data()==0x00) goto Loop; //กลับไม่ได้ก็ให้ไปเริ่มต้นใหม่
                             }
                                } else if( Buf[1]==0x01 && Buf[3] == MY_ID)
                                    {
if(Transfer_data()==0x00) goto Loop;  //ส่งข้อมูลต่อไป หากไปไม่ได้ก็ให้ไปเริ่มต้นใหม่
                              }
                  }
            }
 return 0;
}
```

ส่วนที่สองเป็นส่วนของโปรแกรม GUI (C#) แสดงเฉพาะส่วนต้นเท่านั้น ส่วนที่เหลือจะอยู่ในแผ่นข้อมูล

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;          //ใส่เพื่อใช้งาน RS-232 Port ได้
using System.IO;
using System.Drawing.Drawing2D;

namespace GUI_LDSC
{
    public partial class Form1 : Form
    {
        int counter = new int();
        int end_counter = new int();
        byte[] tbyte = new byte[10];  // สำหรับไว้ส่งข้อมูล
        byte[] bbyte = new byte[12];  //สำหรับไว้รับข้อมูล

        string data_mess;

        int wait_end;
        int time_out = new int();

        /// ////////ตัวแปรสำหรับ Map
        int radius = 100;               /// กำหนดกขอบเขตรัศมีของแต่ละโหนด (ที่ใช้ในการคำนาณ)
        Size size = new Size(200, 200);  /// กำหนดกขอบเขตรัศมีของแต่ละโหนด (ที่ใช้ในการแสดง)
        int posX1, posY1, posX2, posY2, posX3, posY3, posX4, posY4, posX5, posY5;
        Point pointClicked, pointMoveTo, Center1, Center2, Center3, Center4,
Center5,rad1,rad2,rad3,rad4,rad5;

        ////////////////////// กำหนดสีปากกา ////////////////////////
        Pen p = new Pen(Color.WhiteSmoke, 4), p0 = new Pen(Color.Red, 3), p1 = new
Pen(Color.Green, 2), p2 = new Pen(Color.Purple, 2),
            p3 = new Pen(Color.YellowGreen, 2), p4 = new Pen(Color.Blue, 2), p5 = new
Pen(Color.Orange, 2);
```

```
        public Form1()
        {
            InitializeComponent();
        }

  private void Form1_Load(object sender, EventArgs e)
        {
            serialPort1.DataReceived                    +=                    new
SerialDataReceivedEventHandler(serialPort1_DataReceived); //กำหนดช่องเลือกการติดต่อ

            string[] comStr = SerialPort.GetPortNames();
            int i = 0;
            foreach (string port in comStr)
            {
                comboBox1.Items.Add(comStr[i]);
                i++;
            }
            comboBox2.Items.Add(gB_1.Text);
            comboBox2.Items.Add(gB_2.Text);
            comboBox2.Items.Add(gB_3.Text);
            comboBox2.Items.Add(gB_4.Text);
            gB_1.Visible = true;
            gB_1.Enabled = false;
            counter = 0;
            end_counter = 12;
            time_out = 0;    //กำหนดค่าเริ่มต้นของการรอสัญญาณ ใช้ใน Timer1
            wait_end = 60;   //กำหนดระยะเวลาที่ทำการรอสัญญาณกลับ

            timer_DateTime.Start();
            //timer1.Enabled = false;
            //timer2.Enabled = false;
            //timer3.Enabled = false;
            //Graphics g = gB_5.CreateGraphics();
            p0.DashStyle = DashStyle.Dash;
            p1.DashStyle = DashStyle.Dash;
            p2.DashStyle = DashStyle.Dash;
            p3.DashStyle = DashStyle.Dash;
            p4.DashStyle = DashStyle.Dash;
            p5.DashStyle = DashStyle.Dash;
        }
```

```
private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    while (counter < end_counter)
    {
        bbyte[counter] = (byte)serialPort1.ReadByte();
        counter = counter + 1;

    }
    serialPort1.DiscardInBuffer(); //Clear the UART TX Buffer
    serialPort1.DiscardInBuffer(); //Clear the UART RX Buffer
}

private void timer_DateTime_Tick(object sender, EventArgs e)
{
    date.Text = DateTime.Now.ToLongDateString();
    time.Text = DateTime.Now.ToLongTimeString();
}

private void timer1_Tick(object sender, EventArgs e)
{
    time_out = time_out + 1;

    textBox8.Text = "";
    textBox9.Text = "";
    textBox10.Text = "";
    textBox11.Text = "";
    textBox12.Text = "";
    textBox13.Text = "";
    textBox14.Text = "";
    textBox15.Text = "";
    textBox16.Text = "";
    textBox17.Text = "";
    textBox18.Text = "";
    textBox19.Text = "";

    textBox8.Text = bbyte[0].ToString();
    textBox9.Text = bbyte[1].ToString();
    textBox10.Text = bbyte[2].ToString();
    textBox11.Text = bbyte[3].ToString();
    textBox12.Text = bbyte[4].ToString();
    textBox13.Text = bbyte[5].ToString();
```

```
textBox14.Text = bbyte[6].ToString();
textBox15.Text = bbyte[7].ToString();
textBox16.Text = bbyte[8].ToString();
textBox17.Text = bbyte[9].ToString();
textBox18.Text = bbyte[10].ToString();
textBox19.Text = bbyte[11].ToString();

if (gB_1.Text == comboBox2.Text)
    data_mess += gB_1.Text + "    " + date.Text + "    " + time.Text + "    " +
Temp.Text + "    " + groupBox12.Text + "  " + bbyte[5].ToString() + "    " + bbyte[6].ToString()
+ "\r\n";

else if (gB_2.Text == comboBox2.Text)
    data_mess += gB_2.Text + "    " + date.Text + "    " + time.Text + "    " +
Temp.Text + "    " + groupBox12.Text + "  " + bbyte[5].ToString() + "    " + bbyte[6].ToString()
+ "\r\n";

else if (gB_3.Text == comboBox2.Text)
    data_mess += gB_3.Text + "    " + date.Text + "    " + time.Text + "    " +
Temp.Text + "    " + groupBox12.Text + "  " + bbyte[5].ToString() + "    " + bbyte[6].ToString()
+ "\r\n";

else
    data_mess += gB_4.Text + "    " + date.Text + "    " + time.Text + "    " +
Temp.Text + "    " + groupBox12.Text + "  " + bbyte[5].ToString() + "    " + bbyte[6].ToString()
+ "\r\n";

//timer1.Enabled = false;
}
private void timer2_Tick(object sender, EventArgs e)
{
    if (bbyte[1] ==0x01  && bbyte[2] ==0x01 && bbyte[3] == 0x00)
    {
        if (bbyte[7] == 0x01 || bbyte[7] == 0x00) radioButton1.Checked = true;
        else radioButton1.Checked = false;
        if (bbyte[7] == 0x02) radioButton2.Checked = true;
        else radioButton2.Checked = false;
        Temp.Text = bbyte[5].ToString() + "." + bbyte[6].ToString() + "  Cel";
        textBox20.Text = "No";
        checkBox1.Checked = false;
        checkBox2.Checked = false;
```

```
        if (bbyte[8] != 0x00 && bbyte[9] == 0xFF)
        {
            if (bbyte[8] < 0x0a)
            {
                textBox20.Text = "#" + bbyte[8].ToString();
                checkBox1.Checked = true;
            }
            else
            {
                textBox20.Text = "#" + (bbyte[8]-0x0a).ToString();
                checkBox2.Checked = true;
            }
        }

        if (bbyte[8] != 0x00 && bbyte[9] != 0x00 && bbyte[10] == 0xFF)
        {
            if (bbyte[8] == (bbyte[9]-0x0a))
            {
                textBox20.Text = "#" + bbyte[8].ToString();
                checkBox1.Checked = true;
                checkBox2.Checked = true;
            }
            else
            {
                Temp.Text = "Can't sent to M#4";
                textBox20.Text = "#" + bbyte[8].ToString() + ", " + "#" + bbyte[9].ToString();
                radioButton1.Checked = false;
                radioButton2.Checked = false;
                checkBox1.Checked = true;
            }
        }
        timer1.Enabled = false;
        timer2.Enabled = false;
    }
    else
    {
        Temp.Text = "  WAIT.... " + time_out.ToString();
        if (time_out == wait_end)
        {
            Temp.Text = " Time Out      ";
```

```
                time_out = 0;
                timer1.Enabled = false;
                timer2.Enabled = false;
            }
        }

    }

    private void check_data_Tick(object sender, EventArgs e)
    {
        counter = 0;
        if (bbyte[2] == 0x00 && bbyte[3] == 0xFF)
            {
                tbyte[0] = 0x0F;
                textBox1.Text = tbyte[0].ToString();
                tbyte[1] = 0x00;
                textBox2.Text = tbyte[1].ToString();
                tbyte[2] = 0x01;
                textBox3.Text = tbyte[2].ToString();
                tbyte[3] = 0xFF;
                textBox4.Text = tbyte[3].ToString();

                tbyte[0] = byte.Parse(textBox1.Text);
                tbyte[1] = byte.Parse(textBox2.Text);
                tbyte[2] = byte.Parse(textBox3.Text);
                tbyte[3] = byte.Parse(textBox4.Text);

                serialPort1.Write(tbyte, 0, 7);
                //counter = 0;
                check_data.Enabled = false;
            }

    }

    private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
    {
        //for (int i = 0; i < 15; i++)
        //{ bbyte[i] = 0x00; }
        Temp.Text = "";
        textBox20.Text = "";
        timer2.Enabled = true;
```

```
//Temp.Text = " Device Ready!  ";

tbyte[0] = 0x0F;
textBox1.Text = tbyte[0].ToString();
tbyte[1] = 0x00;
textBox2.Text = tbyte[1].ToString();
tbyte[2] = 0x01;
textBox3.Text = tbyte[2].ToString();
tbyte[3] = 0xFF;
textBox4.Text = tbyte[3].ToString();

tbyte[0] = byte.Parse(textBox1.Text);
tbyte[1] = byte.Parse(textBox2.Text);
tbyte[2] = byte.Parse(textBox3.Text);
tbyte[3] = byte.Parse(textBox4.Text);
serialPort1.Write(tbyte, 0, 7);
//counter = 0;

tbyte[0] = 0x0F;
textBox1.Text = tbyte[0].ToString();
tbyte[1] = 0x00;
textBox2.Text = tbyte[1].ToString();
tbyte[2] = 0x00;
textBox3.Text = tbyte[2].ToString();
tbyte[3] = 0x01;
textBox4.Text = tbyte[3].ToString();
tbyte[4] = 0x00;
textBox5.Text = tbyte[4].ToString();
tbyte[5] = 0x00;
textBox6.Text = tbyte[5].ToString();
tbyte[6] = 0xFF;
textBox7.Text = tbyte[6].ToString();

data_mess = "";

tbyte[0] = byte.Parse(textBox1.Text);
tbyte[1] = byte.Parse(textBox2.Text);
tbyte[2] = byte.Parse(textBox3.Text);
tbyte[3] = byte.Parse(textBox4.Text);
tbyte[4] = byte.Parse(textBox5.Text);
tbyte[5] = byte.Parse(textBox6.Text);
```

```
tbyte[6] = byte.Parse(textBox7.Text);

if (gB_1.Text == comboBox2.Text)
{
    tbyte[4] = 0x01;
    textBox5.Text = tbyte[4].ToString();

    gB_1.Enabled = true;
    btnModu1_Read.Enabled = true;
    btnModu1_On.Enabled = true;
    btnModu1_Off.Enabled = true;
    //btnConf1.Enabled = true;

    gB_1.Visible = true;
    gB_2.Visible = false;
    gB_3.Visible = false;
    gB_4.Visible = false;
}
else if (gB_2.Text == comboBox2.Text)
{
    tbyte[4] = 0x02;
    textBox5.Text = tbyte[4].ToString();

    btnModu2_Read.Enabled = true;
    btnModu2_On.Enabled = true;
    btnModu2_Off.Enabled = true;
    //btnConf2.Enabled = true;

    gB_1.Visible = false;
    gB_2.Visible = true;
    gB_3.Visible = false;
    gB_4.Visible = false;
}
else if (gB_3.Text == comboBox2.Text)
{
    tbyte[4] = 0x03;
    textBox5.Text = tbyte[4].ToString();

    btnModu3_Read.Enabled = true;
    btnModu3_On.Enabled = true;
    btnModu3_Off.Enabled = true;
```

```
            //btnConf3.Enabled = true;

            gB_1.Visible = false;
            gB_2.Visible = false;
            gB_3.Visible = true;
            gB_4.Visible = false;
        }
        else
        {
            tbyte[4] = 0x04;
            textBox5.Text = tbyte[4].ToString();

            btnModu4_Read.Enabled = true;
            btnModu4_On.Enabled = true;
            btnModu4_Off.Enabled = true;
            //btnConf4.Enabled = true;

            gB_1.Visible = false;
            gB_2.Visible = false;
            gB_3.Visible = false;
            gB_4.Visible = true;
        }

        btnConf1.Enabled = false;
        btnConf2.Enabled = false;
        btnConf3.Enabled = false;
        btnConf4.Enabled = false;


    }
```