การพัฒนาระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

นางสาวปิยนุช เนตรคุณ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชานิวเคลียร์เทคโนโลยี ภาควิชาวิศวกรรมนิวเคลียร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

DEVELOPMENT OF A GAMMA RADIATION MONITORING SYSTEM VIA LOCAL AREA COMPUTER NETWORK

Miss Piyanooch Nedkun

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Nuclear Technology

Department of Nuclear Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2011

หัวข้อวิทยานิพนธ์	การพัฒนาระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น
โดย	นางสาวปิยนุช เนตรคุณ
สาขาวิชา	นิวเคลียร์เทคโนโลยี
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์อรรถพร ภัทรสุมันต์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

……. ประธานกรรมการ
(รองศาสตราจารย์นเรศร์ จันทน์ขาว)

…….. อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์อรรถพร ภัทรสุมันต์)

…….กรรมการ
(ผู้ช่วยศาสตราจารย์สุวิทย์ ปุณณชัยยะ)

……. กรรมการ
(อาจารย์เดโช ทองอร่าม)

……. กรรมการภายนอกมหาวิทยาลัย
(ดร.ศรินรัตน์ วงษ์ลี)

ปิยนุช เนตรคุณ: การพัฒนาระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น. (DEVELOPEMENT OF A GAMMA RADIATION MONITORING SYSTEM VIA LOCAL AREA COMPUTER NETWORK) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ.อรรถพร ภัทรสุมันต์, 81 หน้า.

ได้พัฒนาระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายท้องถิ่นโดยใช้ระบบตรวจวัดรังสีแกมมาที่ควบคุมการทำงานด้วยไมโครคอนโทรลเลอร์ ระบบนี้ประกอบด้วยสถานีลูกข่ายและสถานีแม่ข่ายสำหรับสำหรับตรวจวัดรังสีแกมมา โดยสถานีลูกข่ายฯ จะทำหน้าที่ตรวจวัด และส่งข้อมูลการตรวจวัดรังสีแกมมาไปยังสถานีแม่ข่ายฯ เมื่อมีคำสั่งจากสถานีแม่ข่ายฯ ส่วนสถานีแม่ข่ายฯ จะทำหน้าที่เป็นศูนย์กลางในการควบคุมการทำงานของสถานีลูกข่ายฯ รับข้อมูลการวัดรังสีแกมมาจากลูกข่าย จัดเก็บผลการวัดในฐานข้อมูลและแสดงผลการวัดรังสีแกมมาของสถานีลูกข่ายฯในรูปเว็บเพจ

ผลการทดสอบพบว่าระบบที่พัฒนาขึ้นนี้สามารถแสดงผลการตรวจวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นได้อย่างถูกต้องโดยมีความแตกต่างจากค่ามาตรฐานไม่เกิน ±10% ระบบยังสามารถเก็บข้อมูลไว้ในฐานข้อมูลและแสดงผลในรูปเว็บเพจได้ อีกทั้งยังสามารถเพิ่มจำนวนสถานีลูกข่ายได้ตามต้องการ

ภาควิชา วิศวกรรมนิวเคลียร์ ลายมือชื่อนิสิต.....

สาขาวิชา นิวเคลียร์เทคโนโลยี ลายมือชื่อ อ.ที่ปรึกษา.....

ปีการศึกษา 2554

5170384021 : MAJOR NUCLEAR TECHNOLOGY

KEYWORDS : GAMMA RADIATION MONITORING /LOCAL AREA COMPUTER NETWORK

PIYANOOCH NEDKUN: DEVELOPEMENT OF A GAMMA RADIATION MONITORING SYSTEM VIA LOCAL AREA COMPUTER NETWORK. ADVISOR: ASST. PROF. ATTAPORN PATTARASUMUNT, 81 pp.

A gamma radiation monitoring system via local area network was developed using a microcontroller-controlled gamma-ray measurement system. This system consisted of sub and central gamma-ray measurement stations. The substation measured and displayed gamma-ray level when requested by the central station. The central station functioned as the control unit for controlling the substation, receiving data, recording and reporting all measurement data on the web page.

From performance testing of the developed system, it was found that the system could measure the gamma-ray level correctly with the deviation from standard value less of than ±10%. The system could record data on the database and display all data on the web page. The number of substations of the developed system could be increased as needed.

Department : Nuclear Engineering Student's Signature .....

Field of Study : Nuclear Technology Advisor's Signature .....

Academic Year : 2011

กิตติกรรมประกาศ

ผู้วิจัยขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์อรรถพร ภัทรสุมันต์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ผู้คอยให้คำปรึกษา ชี้แนะ ทำให้วิทยานิพนธ์สำเร็จลุล่วงไปได้ด้วยดี

ขอกราบขอบพระคุณ รองศาสตราจารย์นเรศร์ จันทน์ขาวประธานกรรมการสอบ ผู้ช่วยศาสตราจารย์ สุวิทย์ ปุณณชัยยะ อาจารย์เดโช ทองอร่าม และดร.ศรินรัตน์ วงษ์ลี ผู้เป็นกรรมการสอบวิทยานิพนธ์ ที่ได้ตรวจสอบและให้คำแนะนำในการแก้ไขวิทยานิพนธ์

ขอขอบคุณ อาจารย์ทุกท่านในภาควิชาวิศวกรรมนิวเคลียร์ ที่กรุณาให้ความรู้ คอยแนะนำ และตอบข้อสงสัย ในการศึกษาของผู้เขียน

ขอขอบคุณ คุณวสันต์ สร้อยหล้า ที่คอยแนะนำและตอบข้อสงสัยในการเขียนโปรแกรมควบคุมไมโครคอนโทลเลอร์ผ่านระบบเครือข่ายท้องถิ่น

ขอขอบคุณ คุณณัฐพงศ์ จิตรจักร ที่ช่วยให้คำแนะนำในเรื่องการนำเสนอข้อมูลในรูปแบบของเว็บเพจ

ขอขอบคุณ คุณเฉลิมพงษ์ โพธิ์ลี้ ที่ช่วยให้คำแนะนำในการสอบเทียบมาตรฐานระบบวัดรังสีแกมมา

ขอขอบคุณเพื่อน ๆ พี่ ๆ และน้อง ๆ ที่ภาควิชาวิศวกรรมนิวเคลียร์ทุกคนที่คอยให้กำลังใจ ให้การสนับสนุน และช่วยเหลือในงานวิจัยมาโดยตลอด

ท้ายนี้ขอกราบขอบพระคุณอย่างสูงยิ่งต่อ บิดา มารดา ซึ่งให้การสนับสนุนและเป็นกำลังใจในการศึกษาของผู้เขียนให้สำเร็จลุล่วงได้ด้วยดี

สารบัญ

หน้า

บทคัดย่อภาษาไทย..... ง

บทคัดย่อภาษาอังกฤษ..... จ

กิตติกรรมประกาศ..... ฉ

สารบัญ..... ช

สารบัญตาราง..... ญ

สารบัญภาพ..... ฎ

บทที่

1 บทนำ..... 1

1.1 ความเป็นมาและความสำคัญของปัญหา..... 1

1.2 วัตถุประสงค์ของการวิจัย..... 2

1.3 ขอบเขตของการวิจัย..... 2

1.4 ขั้นตอนและวิธีดำเนินการวิจัย..... 2

1.5 ประโยชน์ที่คาดว่าจะได้รับ..... 2

1.6 เอกสารและงานวิจัยที่เกี่ยวข้อง..... 3

2. แนวคิดและทฤษฎีระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น..... 4

2.1 หัววัดรังสี..... 4

2.2 ระบบวัดรังสี..... 7

2.3 หน่วยแสดงผล..... 8

2.4 แหล่งกำเนิดรังสีแกมมา..... 8

2.5 คุณสมบัติของรังสีแกมมา..... 9

2.6 เครือข่ายคอมพิวเตอร์..... 10

2.7 ความรู้เบื้องต้นเกี่ยวกับ TCP/IP 11

2.8 เครื่องมือตรวจวัดเพื่อเฝ้าระวังทางด้านรังสี..... 12

2.9 ระบบเฝ้าระวังรังสีแกมมาที่ใช้กันอยู่ในปัจจุบัน..... 13

3 การพัฒนาระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น..... 16

3.1 วัสดุและอุปกรณ์ที่ใช้ในการวิจัย..... 16

3.2 การออกแบบระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น..... 16

3.2.1 สถานีสำหรับการตรวจวัดปริมาณรังสีแกมมา..... 16

บทที่ หน้า

3.2.2 สถานีแม่ข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น..... 22
3.3 โปรแกรมควบคุมการทำงานของระบบเฝ้าระวังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น..... 22
3.3.1 โปรแกรมควบคุมการทำงานของสถานีลูกข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น..... 22
3.3.2 โปรแกรมควบคุมการทำงานของสถานีแม่ข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น..... 24
3.3.3 โปรแกรมสำหรับการนำเสนอข้อมูลในรูปแบบของเว็บเพจ..... 27
4 การทดสอบการทำงานของระบบและผลการทดสอบ..... 28
4.1 การทดสอบระบบการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์..... 28
4.2 การทดสอบการรับส่งข้อมูลที่ได้จากการวัดรังสีแกมมาโดยส่งข้อมูลผ่านระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่าย..... 31
4.3 การทดสอบการนำเสนอข้อมูลในรูปแบบของเว็บเพจ..... 36
5 สรุปผลการวิจัยและข้อเสนอแนะ..... 38
5.1 สรุปผลการวิจัย..... 38
5.1.1 การทดสอบระบบการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์..... 38
5.1.2 การทดสอบการรับส่งข้อมูลที่ได้จากการวัดรังสีแกมมาโดยส่งข้อมูลผ่านระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่าย..... 38
5.1.3 การสอบเทียบมาตรฐานระบบวัดรังสีแกมมา..... 39
5.1.4 การทดสอบการนำเสนอข้อมูลในรูปแบบของเว็บเพจ..... 39
5.1.5 ข้อเสนอแนะ..... 39
รายการอ้างอิง..... 40
ภาคผนวก..... 41
ภาคผนวก ก หัววัดรังสีไกเกอร์มูลเลอร์ ของบริษัท Ludlum รุ่น 44-7..... 42

บทที่ หน้า

ภาคผนวก ข MODEL 2200 scaler/ratemeter..... 43
ภาคผนวก ค ไมโครคอนโทรลเลอร์ ET-dsPIC33WEB V1.0..... 44
ภาคผนวก ง โปรแกรมควบคุมการทำงานของสถานีลูกข่าย..... 47
ภาคผนวก จ โปรแกรมควบคุมการทำงานของสถานีแม่ข่าย..... 69
ประวัติผู้เขียนวิทยานิพนธ์..... 81

สารบัญตาราง

ตารางที่		หน้า
3.1	รายละเอียดแสดงการเชื่อมต่อระหว่างไมโครคอมพิวเตอร์กับเครื่องสเกลเลอร์/เรตมิเตอร์……	19
4.1	ผลการแสดงการทดสอบการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์……	30
4.2	ผลการการทดสอบการรับส่งข้อมูลที่ได้จากการวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นโดยส่งข้อมูลผ่านทางระบบเครือข่ายท้องถิ่นเข้าสู่สถานีแม่ข่าย……	33
4.3	ผลการสอบเทียบมาตรฐานระบบวัดรังสีแกมมา……	34
4.4	ผลการทดสอบความถูกต้องของระบบ……	36

สารบัญภาพ

ภาพที่		หน้า
2.1	เครื่อง GM counter และหัววัดรังสีรูปแบบต่าง ๆ.....	6
2.2	TCP layer ……..……...	11
3.1	ลักษณะระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น.....	17
3.2	แผนภาพแสดงการทำงานของสถานีลูกข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น……..……..…….	18
3.3	หัววัดรังสีไกเกอร์มูลเลอร์ ของบริษัท Ludlum รุ่น 44-7……..……...	18
3.4	เครื่องสเกลเลอร์/เรตมิเตอร์ โมเดล 2200 ของบริษัท Ludlum ……..	18
3.5	เครื่องสเกลเลอร์/เรตมิเตอร์ เชื่อมต่อกับไมโครคอนโทรลเลอร์ผ่านพอร์ตอนุกรม……...	19
3.6	การเชื่อมต่อ RS232 กับบอร์ดไมโครคอนโทรลเลอร์……	20
3.7	โครงสร้างบอร์ด ET-dsPIC33WEB V1.0……..……	20
3.8	การเชื่อมต่อ max232 กับไมโครคอนโทรลเลอร์……	21
3.9	โครงสร้างโมดูลสื่อสาร Ethernet รุ่น ET-MINI ENC28J6……..	21
3.10	โฟล์วชาร์ตแสดงขั้นตอนการทำงานของโปรแกรมของระบบลูกข่ายฯ…….	23
3.11	โฟล์วชาร์ตแสดงขั้นตอนการทำงานของโปรแกรมของระบบแม่ข่ายฯ…….	25
3.12	ภาพแสดงรายระเอียดหน้าจอของโปรแกรมของผู้ควบคุมสถานีแม่ข่ายฯ.....	26
3.13	โฟล์วชาร์ตแสดงขั้นตอนการทำงานของโปรแกรมของการนำเสนอข้อมูลในรูปแบบของเว็บเพจ……..	27
4.1	แผนภาพการจัดระบบการทดสอบความแม่นยำในการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์……	29
4.2	การจัดระบบการทดสอบความแม่นยำในการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์..……....	29
4.3	เครื่องสเกลเลอร์/เรตมิเตอร์และหน้าจอคอมพิวเตอร์ที่แสดงผลการวัดรังสีแกมมา.....…….....	30
4.4	การจัดระบบการทดสอบการรับส่งข้อมูลที่ได้จากการวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นโดยส่งข้อมูลผ่านทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่ายฯ..……	32

ภาพที่ หน้า

4.5 ภาพแสดงการจัดระบบการทดสอบการรับส่งข้อมูลที่ได้จากการวัดปริมาณรังสีแกมมาผ่านเครือข่ายท้องถิ่นโดยส่งข้อมูลผ่านทางระบบเครือข่ายท้องถิ่นเข้าสู่สถานีแม่ข่าย…….. 32
4.6 การจัดอุปกรณ์ในการสอบเทียบมาตรฐานระบบวัดรังสีแกมมา…… 34
4.7 กราฟแสดงความสัมพันธ์ระหว่าง อัตราปริมาณรังสีที่คำนวณจากต้นกำเนิดรังสีมาตรฐานกับอัตราปริมาณรังสีที่วัดได้…….. 35
4.8 การรายงานผลการวัดปริมาณรังสีแกมมาในรูปแบบเว็บเพจ..... 37
4.9 ภาพหน้าจอการค้นหาข้อมูลย้อนหลัง……... 37

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันมีการพัฒนาและนำเทคโนโลยีทางด้านรังสี มาใช้ประโยชน์อย่างมากมาย ทั้งด้านการแพทย์ อุตสาหกรรม การเกษตร รวมถึงการผลิตพลังงานจากโรงไฟฟ้านิวเคลียร์ ซึ่งเป็นที่ทราบกันอยู่แล้วว่าในบริเวณที่เกี่ยวข้องกับสารกัมมันตรังสีหรือดำเนินกิจกรรมเกี่ยวเนื่องกับการใช้รังสีนั้นย่อมมีปริมาณรังสีสูงกว่าบริเวณอื่น ๆ ไม่ว่าจะเป็นบริเวณตัวอาคารหรือบริเวณภายนอกรอบอาคาร จึงมีความจำเป็นอย่างยิ่งที่ต้องมีการเฝ้าระวังและตรวจวัดปริมาณรังสีในบริเวณเหล่านั้นอยู่เสมอ โดยการใช้เครื่องมือวัดทางรังสีตรวจวัดและเก็บข้อมูลโดยตรง แต่หากบริเวณที่ทำการตรวจวัดนั้นมีปริมาณรังสีสูงและเป็นอันตรายต่อผู้ปฏิบัติงานรวมทั้งมีพื้นที่หลายจุดที่ต้องทำการควบคุมและเฝ้าระวัง อาจส่งผลให้การทำงานของผู้ปฏิบัติงานเกิดความล่าช้าและเมื่อมีอุบัติเหตุจากรังสีเกิดขึ้นอาจก่อให้เกิดความยุ่งยากในการปฏิบัติงาน ซึ่งจะเป็นการดีอย่างยิ่งหากสามารถสร้างระบบเพื่ออำนวยความสะดวกในการเฝ้าระวังและการเก็บรวบรวมข้อมูลที่ได้มารวมไว้ ณ จุดเดียวโดยไม่ต้องอาศัยผู้ปฏิบัติงานเดินทางเข้าไปในบริเวณดังกล่าวเพื่อเก็บข้อมูล ทำให้ผู้ปฏิบัติงานมีความปลอดภัยมากขึ้น

การเก็บรวบรวมข้อมูล การสื่อสารข้อมูล การเก็บรักษาข้อมูล และการประมวลผลข้อมูลในปัจจุบัน พบว่าองค์กรจำนวนมากมีการเก็บรวบรวบข้อมูลไว้ที่ศูนย์กลางเพื่อความสะดวกในการติดตามข้อมูล ตรวจสอบข้อมูล การใช้ข้อมูลรวมทั้งโปรแกรม จึงมีการวางโครงสร้างระบบเครือข่ายคอมพิวเตอร์ขึ้นมาในระดับองค์กรเพื่อให้การทำงานมีความสะดวกยิ่งขึ้น ซึ่งในปัจจุบันมีการเพิ่มขีดความสามารถในการทำงานของระบบเครือข่ายคอมพิวเตอร์ และความต้องการใช้งานก็เพิ่มสูงตามไปด้วย

ด้วยเหตุผลดังกล่าวจึงเป็นที่มาของงานวิจัยนี้ ที่มุ่งเน้นพัฒนาระบบเฝ้าระวังปริมาณรังสีโดยอาศัยการเชื่อมโยงข้อมูลผ่านเครือข่ายคอมพิวเตอร์ ซึ่งจะทำให้ผู้ปฏิบัติงานนั้นสามารถรับข้อมูลที่ได้จากการวัดปริมาณรังสีโดยไม่ต้องเข้าไปยังจุดต่าง ๆ หรือบริเวณที่มีความเสี่ยง ทำให้เกิดความสะดวกและปลอดภัยในการปฏิบัติงานมากยิ่งขึ้น อีกทั้งยังสามารถเข้าถึงข้อมูลได้สะดวกและรวดเร็วโดยการประมวลผลและรายงานผลข้อมูลโดยผ่านระบบเครือข่ายคอมพิวเตอร์

1.2 วัตถุประสงค์ของการวิจัย

เพื่อพัฒนาระบบเฝ้าระวังรังสีแกมมาที่สามารถส่งข้อมูลผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น (local area network)

1.3 ขอบเขตของการวิจัย

1. ออกแบบและพัฒนาระบบเฝ้าระวังรังสีแกมมาที่สามารถส่งข้อมูลการตรวจวัดรังสีผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น
2. พัฒนาโปรแกรมควบคุมการทำงานของระบบการส่งข้อมูลผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น การประมวลผลข้อมูล การเก็บข้อมูลบนฐานข้อมูล และการแสดงผลข้อมูลบนไมโครคอมพิวเตอร์ในรูปแบบของเว็บเพจ ได้แก่ ระดับรังสี วัน เวลาของแต่ละสถานีลูกข่าย

1.4 ขั้นตอนและวิธีดำเนินการวิจัย

1. ศึกษาค้นคว้าทฤษฎี ข้อมูลและผลงานวิจัยที่เกี่ยวข้อง
2. ศึกษา ออกแบบอุปกรณ์เชื่อมโยงระหว่างระบบวัดรังสีกับระบบควบคุมที่สามารถส่งข้อมูลผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น
3. ออกแบบและพัฒนาโปรแกรมบนระบบควบคุมให้มีความสามารถในการส่งข้อมูลการวัดปริมาณรังสีจากระบบตรวจวัดรังสีผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น การประมวลผลข้อมูล การเก็บข้อมูลบนฐานข้อมูลและการแสดงผลข้อมูลบนคอมพิวเตอร์ในรูปแบบของเว็บเพจ
4. ทดสอบและปรับปรุงการทำงานของระบบที่พัฒนาขึ้น
5. วิเคราะห์สรุปผลผลการวิจัยและเขียนวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ได้ระบบเฝ้าระวังรังสีแกมมาที่สามารถเฝ้าระวังรังสีแกมมาได้อย่างต่อเนื่องผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น และแสดงผลบนคอมพิวเตอร์ในรูปแบบของเว็บเพจ ซึ่งมีประโยชน์อย่างมากในการป้องกันอันตรายจากรังสี

1.6 เอกสารและงานวิจัยที่เกี่ยวข้อง

1. ปี พ.ศ.2539 ธนากร อรัญศิริ ได้ทำงานวิจัยเรื่องการพัฒนาระบบเชื่อมโยงสัญญาณสำหรับเฝ้าระวังทางรังสีโดยใช้เครือข่ายวิทยุสื่อสาร (Development of an interfacing system for radiation surveillance using a radio communication network) ซึ่งเป็นวิทยานิพนธ์ปริญญามหาบัณฑิต โดยการพัฒนาระบบเชื่อมโยงสัญญาณทางด้านระบบเฝ้าระวังทางรังสี โดยใช้เครือข่ายวิทยุสื่อสาร มีวัตถุประสงค์เพื่อพัฒนาระบบรายงานข้อมูลวัดปริมาณรังสีเป็นกิจวัตร จากเครื่องวัดปริมาณรังสีเครือข่ายทั่วภูมิภาคของประเทศ รวมทั้งส่งสัญญาณเตือนการตรวจพบปริมาณรังสีที่ผิดปกติโดยอัตโนมัติผ่านเครือข่ายวิทยุสื่อสารเชื่อมโยงมายังสถานีแม่ข่ายที่ศูนย์ควบคุมปริมาณรังสีในสิ่งแวดล้อม ด้วยการจำลองระบบเครือข่ายวิทยุสื่อสารของการไฟฟ้าฝ่ายผลิตแห่งประเทศไทย [1]
2. ปี พ.ศ.2549 กิตติศักดิ์ ชัยสรรค์ ได้ทำงานวิจัยเรื่องการพัฒนาระบบเฝ้าระวังรังสีแกมมาในสิ่งแวดล้อมผ่านเครื่อข่ายโทรศัพท์เคลื่อนที่ (Development of an environmental gamma radiation monitoring system via mobile telephone network) โดยพัฒนาระบบเฝ้าระวังรังสีแกมมาในสิ่งแวดล้อมผ่านเครือข่ายโทรศัพท์เคลื่อนที่ ด้วยการนำระบบตรวจวัดรังสีแกมมา ไมโครคอมพิวเตอร์และโทรศัพท์เคลื่อนที่ มาพัฒนาเป็นระบบเฝ้าระวังรังสีที่ประกอบด้วยสถานีลูกข่ายและสถานีแม่ข่ายระวังรังสีแกมมาในสิ่งแวดล้อม [2]
3. ปี 2541 Ch. Wedekind,G. Schilling, M. Gruttmuller, K. Becker [3] ได้เสนองานวิจัยใน Applied Radiation and Isotopes เรื่อง Gamma-radiation monitoring network at sea. เป็นระบบเฝ้าระวังปริมาณการปนเปื้อนของสารกัมมันตรังสีในน้ำทะเล โดยการวัดปริมาณรังสีแกมมาในน้ำทะเลในพื้นที่ต่างๆและเลือกใช้หัววัดที่สามารถแสดงปริมาณรังสีแกมมา และแสดงสเปกตรัมของรังสีแกมมาได้ โดยข้อมูลของการวัดรังสีจะถูกส่งผ่านระบบเครือข่ายเข้าสู่ระบบหลัก ซึ่งข้อมูลดังกล่าวจะถูกประมวลผลและจำลองเหตุการณ์เพื่อหาตำแหน่งของแหล่งกำเนิดหากมีความผิดปกติเกิดขึ้น [3]

บทที่ 2

แนวคิดและทฤษฎีระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

2.1 หัววัดรังสี [4]

กระบวนการตรวจวัดรังสีของหัววัดรังสี อาศัยโครงสร้างของหัววัดรังสีซึ่งประกอบด้วยตัวกลางอันตรกิริยาจากรังสีและส่วนสร้างสัญญาณพัลส์ การแบ่งชนิดของหัววัดรังสีจะแบ่งตามประเภทของตัวกลาง (detection media) ที่ใช้กับหัววัดรังสี 3 ประเภท คือ

ก. หัววัดชนิดบรรจุก๊าซ (Gas-filled detector)

ข. หัววัดชนิดซิลทิเลชัน (Scintillation detector)

ค. หัววัดชนิดสารกึ่งตัวนำ (Semiconductor detector)

อย่างไรก็ตามหัววัดรังสีที่ใช้ตัวกลางที่แตกต่างเหล่านี้แบ่งย่อยตามคุณลักษณะการทำงานของตัวกลาง สถานะของตัวกลาง และลักษณะโครงสร้างของตัวกลางในกรณีเฉพาะของหัววัดรังสีแต่ละชนิดดังนี้

2.1.1 หัววัดรังสีชนิดบรรจุก๊าซ แบ่งประเภทของหัววัดรังสีออกตามคุณลักษณะการทำงานชองก๊าซที่บรรจุและทำให้ปริมาณคู่ของไอออนต่างปริมาณกันแบ่งได้เป็น

ก. หัววัดรังสีแบบไอออไนเซชัน (Ionization chamber detector)

ข. หัววัดรังสีแบบพรอพอร์ชันแนล (Proportional detector)

ค. หัววัดรังสีแบบไกเกอร์ (Geiger – Muller detector)

และยังมีการสร้างหัววัดชนิดที่ใช้ก๊าซเป็นตัวกลางในการวัดรังสีในลักษณะภาชนะปิดไม่สนิทและปล่อยให้ก๊าซไหลผ่านโครงสร้างของหัววัดรังสี เรียกว่า Gas flow detector (counter) เช่น หัววัดที่มีการจัดรูปทรงการวัดรังสีแบบ 2π และ 4π หรือหัววัดรังสีขนาดเล็กที่หน้าต่างบางสำหรับวัดรังสีพลังงานต่ำ

2.1.2 หัววัดรังสีชนิดซินทิลเลชัน แบ่งประเภทของรังสีตามสถานะของตัวกลาง เช่น ของแข็ง ของเหลว และก๊าซ เป็นต้น อันตรกิริยาในตัวกลางจะทำให้เกิดประกายแสง (scintillated light) ส่งไปยังโฟโตแคโทดของหลอดทวีคูณอิเล็กตรอน (photomultiplier tube, PMT) เปลี่ยนเป็นสัญญาณไฟฟ้า แบ่งได้เป็น

ก. หัววัดรังสีแบบซินทิลเลชันแบบของแข็ง (Solid scintillation detector)

ข. หัววัดรังสีแบบซินทิลเลชันแบบของเหลว (Liquid scintillation detector)

ค. หัววัดรังสีแบบซินทิลเลชันแบบก๊าซ (Gas scintillation detector)

3 หัววัดรังสีชนิดสารกึ่งตัวนำ แบ่งประเภทของหัววัดรังสีตามเทคนิคการสร้างรอยต่อ P–N สำหรับการวัดรังสีให้มีบริเวณไวต่อรังสีที่เหมาะสมกับคุณสมบัติของรังสีที่มีประจุไฟฟ้าในตัว และรังสีประเภทคลื่นแม่เหล็กไฟฟ้าแบ่งได้เป็น

3.1 หัววัดรังสีสารกึ่งตัวนำสำหรับวัดอนุภาคที่มีประจุไฟฟ้า (Charged-particle detector)

3.1.1 หัววัดรังสีชนิดสารกึ่งตัวนำแบบดิฟฟิวจังชัน (Silicon diffused junction detector, SDJ)

3.1.2 หัววัดรังสีชนิดสารกึ่งตัวนำแบบเซอร์เฟสแบริเออร์ (Silicon surface barrier detector, SSB)

3.1.3 หัววัดรังสีชนิดสารกึ่งตัวนำแบบอิมแพลนต์พลานา (Passivated implanted planar silicon detector, PIPS)

3.2 หัววัดรังสีชนิดสารกึ่งตัวนำสำหรับวัดโฟตอน หรือรังสีคลื่นแม่เหล็กไฟฟ้า (Photon detector)

3.2.1 หัววัดรังสีชนิดสสารกึ่งตัวนำแบบลิเทียมดริฟท์ (Lithium-drifted detector, Si(Li) หรือ Ge(Li))

3.2.2 หัววัดรังสีชนิดสารกึ่งตัวนำแบบเจอร์มาเนียมบริสุทธิ์ (High–purity germanium detector, HPGe)

ในงานวิจัยนี้เลือกใช้หัววัดรังสีชนิดบรรจุก๊าซแบบไกเกอร์ (Geiger–Muller detector) ในการทดสอบระบบ

Geiger Muller counter (GM counter) [5]

เครื่องสำรวจรังสีนิยมใช้หัววัด GM counter ในการวัดรังสีหรืออนุภาคอย่างกว้างขวาง เนื่องจากใช้งานง่าย ราคาไม่แพง เหมาะสำหรับเป็นเครื่องมือในการสำรวจสารกัมมันตรังสี โดยเฉพาะในงานป้องกันอันตรายจากรังสี GM counter (ภาพที่ 2.1) มีหลายชนิดขึ้นกับการใช้งาน ได้แก่

• Thin-end window GM counter เป็นหัววัดรังสีที่มีปลายหัววัดด้านหนึ่งทำด้วยแผ่นไมกาบางๆ ซึ่งส่วนมากจะใช้วัดรังสีแอลฟา และบีตาพลังงานต่ำ สามารถวัดรังสีแกมมาได้เช่นกัน

• Side- window GM counter จะมีปลอกผนังด้านนอกเป็นอะลูมิเนียมหรือสแตนเลส ซึ่งสามารถเลื่อนปิดเปิดได้เพื่อวัดรังสีตามที่ต้องการ เมื่อผนังเลื่อนปิดคลุมหัววัดจะใช้วัดรังสีแกมมา เมื่อผนังเลื่อนเปิดจะใช้วัดรังสีบีตา หัววัดชนิดนี้เหมาะสำหรับใช้วัดรังสีบีตาพลังงานสูง รังสีแกมมา และรังสีเอกซ์

• Pancake shaped probe จะใช้ในการวัดรังสี อัลฟา บีตาและแกมมา เหมือนชนิด thin-end window แต่จะมี sensitivity สูงกว่าชนิด end-window เนื่องจากมีพื้นที่ในการวัดรังสีมากกว่าประมาณ 3 เท่า

ภาพที่ 2.1 เครื่อง GM counter และหัววัดรูปแบบต่าง ๆ

2.2 **ระบบวัดรังสี** [4]

เป็นกระบวนการนำผลที่ได้จากหัววัดรังสีไปสร้างเป็นข้อมูลการวัดและวิเคราะห์ผลการวัดรังสีเพื่อแสดงผลในหน่วยวัดรังสีมาตรฐานประกอบด้วย 2 ส่วนย่อย

- อุปกรณ์วัดรังสี ทำหน้าที่จัดสัญญาณไฟฟ้าจากหัววัดรังสีด้วยกระบวนการทางอิเล็กทรอนิกส์ เพื่อจัดข้อมูลให้อยู่ในรูปแบบที่ต้องการ เช่น การวัดปริมาณรังสี การวิเคราะห์สเปกตรัมนิวเคลียร์ เป็นต้น จึงจำเป็นต้องมีเทคนิคในการจัดอุปกรณ์วัดให้จัดการข้อมูลวัดในรูปแบบที่ต้องการ

- การแสดงผลการวัดรังสี ผลการวัดรังสีในหน่วยมาตรฐานจะต้องมีการวิเคราะห์ และแปรผลข้อมูลที่วัดได้จากกระบวนการวัดรังสีจึงต้องการเทคนิคด้านประเมินผลการวัดรังสี เพื่อให้ผลการวัดมีความถูกต้อง ความแม่นยำ และมีความน่าเชื่อถือ ทั้งนี้เนื่องจากในกระบวนการวัดรังสีมีตัวแปรที่มีผลต่อค่านับวัดรังสีหลายอย่าง เช่น ประสิทธิภาพของการวัดรังสี รังสีรบกวนจากภายนอก ความแปรปรวนของการแผ่รังสี และความไม่เสถียรของระบบวัดรังสี เป็นต้น

ระบบวัดรังสีโดยทั่วไปแล้วสามารถแบ่งหน้าที่การทำงานของกลุ่มอุปกรณ์การวัดเฉพาะหน้าที่ในการจัดการสัญญาณได้เป็น 4 ส่วนหลักคือ

- ส่วนสร้างข้อมูลการวัดรังสี เป็นส่วนการทำงานที่ประกอบด้วยหัววัดรังสีที่ได้รับการไบอัสด้วยแหล่งจ่ายไฟฟ้าศักดาสูง ทำหน้าที่ตรวจจับรังสีและสร้างปริมาณสัญญาณไฟฟ้า ที่เป็นสัดส่วนต่อการถ่ายโอนพลังงานของรังสีในตัวกลางของหัววัดรังสี
- ส่วนเลือกข้อมูลวัดรังสี เป็นส่วนการทำงานที่ประกอบด้วยอุปกรณ์ขยายสัญญาณและอุปกรณ์เฉพาะหน้าที่ในการตัดสัญญาณที่ไม่ต้องการ หรือเลือกช่วงสัญญาณที่ต้องการวิเคราะห์ ทั้งในรูปแบบของขนาดความสูงของสัญญาณพัลส์และช่วงเวลาที่แตกต่างกันของสัญญาณลอจิกที่ได้รับการเปลี่ยนหลังเกิดอันตรกิริยาในหัววัดรังสี ข้อมูลจากส่วนการทำงานนี้จะส่งเข้าการวิเคราะห์และเก็บข้อมูลต่อไป
- ส่วนเก็บข้อมูล เป็นส่วนการทำงานที่ประกอบด้วยอุปกรณ์สำหรับทำหน้าที่สะสมข้อมูล จำนวนนับรังสีรวม และการนับรังสีเฉพาะค่าพลังงาน ได้แก่ อุปกรณ์นับรังสี หรืออุปกรณ์อ่านค่าอัตรานับรังสี หรืออุปกรณ์วิเคราะห์ขนาดความสูงของสัญญาณพัลส์ เป็นต้น อุปกรณ์เหล่านี้จะทำการเก็บข้อมูลจำนวนนับรังสีเพื่อแสดงผลชั่วคราวเท่านั้น ในส่วนนี้อาจมีการจัดการข้อมูลแสดงผลให้อยู่ในรูปสเปกตรัมพลังงานได้

- ส่วนควบคุมการทำงานของระบบและส่วนจัดเก็บข้อมูลถาวร ในส่วนของอุปกรณ์ควบคุมการทำงานของระบบจะจัดการด้านการควบคุมระยะเวลาวัดรังสี ควบคุมการวิเคราะห์สัญญาณพัลส์ของการวัดรังสีเฉพาะกลุ่มหรือควบคุมตำแหน่งการวิเคราะห์พลังงานให้คงที่ อุปกรณ์ควบคุมเหล่านี้จะส่งสัญญาณควบคุมไปยังอุปกรณ์การทำงานส่วนต่าง ๆ ตามความเหมาะสมอีกส่วนหนึ่ง คืออุปกรณ์บันทึกข้อมูลถาวร ได้แก่ เครื่องเขียนกราฟ เครื่องพิมพ์ผล เป็นต้น

อุปกรณ์การวัดรังสีในส่วนการทำงานต่าง ๆ จะได้รับการออกแบบให้ทำการจัดการสัญญาณเฉพาะหน้าที่ ดังนั้นการจัดระบบวัดรังสีเพื่อให้ได้ข้อมูลการวัดตามกระบวนการใช้งานที่ต้องการ ไม่ว่าจะเป็นการประยุกต์การวัดรังสีด้านการวิเคราะห์ธาตุ งานควบคุมทาง อุตสาหกรรม งานด้านการแพทย์ และงานวิจัยค้นคว้าทางวิทยาศาสตร์ เป็นต้น จะต้องเลือกอุปกรณ์ต่าง ๆ เพื่อจัดระบบวัดให้ได้รูปแบบของข้อมูลตามที่ต้องการ

2.3 **หน่วยแสดงผล** [5]

Scaler and Timer

สัญญาณที่ผ่านการคัดเลือกจาก SCA จะถูกส่งมาบันทึกค่าเป็นตัวเลขด้วยเครื่อง scaler โดยมี timer เป็นตัวควบคุมเวลาในการนับวัด

Analog ratemeter

เป็นเครื่องสำหรับแสดงค่านับวัดเฉลี่ยต่อหน่วยเวลา โดยค่านับวัดที่แสดงบนจอจะแปรตามค่าเฉลี่ยของจำนวนนับรังสีที่วัดได้

2.4 **แหล่งกำเนิดรังสีแกมมา** [2]

เพื่อการเฝ้าระวังรังสีให้ครอบคลุมบริเวณที่จะมีโอกาสได้รับผลกระทบทางรังสีจากรังสีแกมมา ต้องทราบถึงแหล่งกำเนิดรังสีแกมมาในสิ่งแวดล้อม ซึ่งมี 2 แหล่ง คือ

1. แหล่งกำเนิดรังสีจากธรรมชาติ คือ พื้นดิน (Terrestrial sources of radiation) ในพื้นดินมีสารกัมมันตรังสีที่มีค่าครึ่งชีวิตยาว ปลดปล่อยรังสี และกำเนิดขึ้นมาพร้อมกับโลก ได้แก่ โปแตสเซียม-40, รูบิเดียม-87, ยูเรเนียม-238 และทอเรียม-232 โดย

โปแตสเซียม-40 เป็นสารกัมมันตรังสีหลักในการปลดปล่อยรังสีแกมมาสู่สิ่งแวดล้อม สำหรับยูเรเนียมและทอเรียมเป็นสารกัมมันตรังสีที่ปลดปล่อยรังสีแกมมาสู่สิ่งแวดล้อมในระดับต่ำแต่เป็นต้นทางของสายกำเนิดสารกัมมันตรังสีอีกเป็นจำนวนมากที่มีการสลายตัวอย่างต่อเนื่องไปสิ้นสุดที่ไอโซโทปเสถียร เรียกว่า อนุกรม ได้แก่ อนุกรมทอเรียม อนุกรมยูเรเนียม และอนุกรมแอคติเนียม ซึ่งสารกัมมันตรังสีบางตัวในแต่ละอนุกรมจะปลดปล่อยรังสีแกมมาสู่สิ่งแวดล้อมด้วย ดังนั้นบริเวณที่เป็นแหล่งกำเนิดรังสีแกมมาตามธรรมชาติจึงต้องมีการติดตามเฝ้าระวังแบบต่อเนื่องเพื่อไม่ให้ผู้ที่อาศัยอยู่บริเวณนั้นได้รับปริมาณรังสีสะสมสูงจะทำให้เกิดอันตราย

2. แหล่งกำเนิดรังสีที่มนุษย์สร้างขึ้น ได้แก่ การใช้สารกัมมันตรังสีในทางการแพทย์ การใช้ในทางอุตสาหกรรม การทดลองอาวุธนิวเคลียร์ การผลิตพลังงานโดยนิวเคลียร์ กิจกรรมเหล่านี้ เพื่อให้ผู้ปฏิบัติงานทางรังสีและมนุษย์ที่อาศัยบริเวณใกล้เคียงได้รับปริมาณรังสีที่น้อยที่สุดเท่าที่จะเป็นไปได้และต้องไม่เกินเกณฑ์ที่กำหนด ดังนั้นจึงจำเป็นที่จะต้องมีการเฝ้าระวังรังสีแกมมา โดยเฉพาะแหล่งกำเนิดรังสีขนาดใหญ่ที่มนุษย์สร้างขึ้น ซึ่งต้องการเฝ้าระวังและมีความรวดเร็วในการแจ้งเตือนเมื่อรังสีแกมมาเกินเกณฑ์ที่กำหนด

2.5 **คุณสมบัติของรังสีแกมมา** [5]

รังสีแกมมามีคุณสมบัติเป็นคลื่นแม่เหล็กไฟฟ้า เคลื่อนที่ด้วยความเร็วเท่าแสง สามารถเลี้ยวเบน แทรกสอดได้ และสามารถแสดงพฤติกรรมเป็นอนุภาคได้ จึงเรียกว่าเป็น โฟตอน พลังงานของรังแกมมา หรือคลื่นแม่เหล็กไฟฟ้า จะเพิ่มขึ้นตามความถี่ที่เพิ่มขึ้น หรือความยาวคลื่นที่สั้นลงตามสมการ

$$E = \frac{hc}{\lambda} \quad \text{.....(1)}$$

โดย h = ค่าคงที่ของ Planck = 6.6×10^{-34} J.s

C = ความเร็วคลื่นแม่เหล็กไฟฟ้า = 3.8×10^{8} m/s

λ = ความยาวคลื่นแม่เหล็กไฟฟ้า (m)

เมื่อรังสีแกมมาแสดงพฤติกรรมเป็นอนุภาค จะมีโมเมนตัม = h/λ มีอำนาจทะลุวัตถุได้สูง แต่ความเข้มรังสีจะลดลงตามระยะทางในอากาศ เป็นสัดส่วนผกผันกำลังสอง (inverse square law)

โดยมีความสัมพันธ์ตามสมการคณิตศาสตร์ดังนี้

$$\frac{I_1}{I_2} = \frac{r_2^2}{r_1^2} \qquad \text{.....(2)}$$

I^1, I^2 = ความเข้มรังสีที่ระยะ r^1, และ r^2 ตามลำดับ

r^1, r^2 = ระยะห่างจากต้นกำเนิดรังสี

2.6 เครือข่ายคอมพิวเตอร์ [6]

2.6.1 เครือข่ายท้องถิ่น

เครือข่ายท้องถิ่น (Local area network, LAN) หมายถึง เครือข่ายคอมพิวเตอร์ขนาดเล็กที่เป็นของคนเฉพาะกลุ่ม ปกติจะเป็นเครือข่ายที่มีขอบเขตอยู่ภายในอาคาร หรือ กลุ่มอาคารที่อยู่ติดกันมีระยะทางไม่เกิน 2-3 กิโลเมตร เหมาะสำหรับการใช้เชื่อมต่อคอมพิวเตอร์ส่วนบุคคล หรือเครื่องคอมพิวเตอร์ขนาดเล็กของพนักงานในองค์กรเข้าด้วยกัน โดยมีวัตถุประสงค์หลักคือ การใช้อุปกรณ์ส่วนกลางร่วมกัน เครือข่ายท้องถิ่นมีลักษณะเฉพาะที่แตกต่างจากระบบอื่น ๆ 3 ประการ คือ

(1) ขนาด

(2) เทคโนโลยีที่ใช้ในการรับ–ส่งข้อมูล

(3) รูปแบบการจัดโครงสร้างของระบบ

เทคโนโลยีที่ใช้ในการรับ-ส่งข้อมูลบนเครือข่ายท้องถิ่นโดยปกติจะใช้เพียงสายเคเบิลเส้นเดียวซึ่งจะเชื่อมต่อทั้งระบบเข้าด้วยกัน เครือข่ายปกติมีความเร็ว 10 ล้านบิตต่อวินาที (Mbps) หรือ 100 Mbps มีระยะเวลาในการรอคอยเฉลี่ยเพื่อส่งข้อมูลไม่เกิน 1000 ส่วนล้านวินาที ($x10^{-3}$s) และมีโอกาสที่จะเกิดข้อผิดพลาดน้อยมาก

มาตรฐาน IEEE 802.3

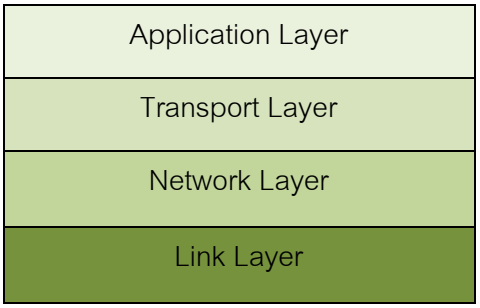
IEEE 802.3 หรือ อีเทอร์เน็ต (Ethernet) เป็นเครือข่ายที่มีความเร็วสูงการส่งข้อมูล 10 เมกกะบิตต่อวินาที IEEE ได้กำหนดมาตรฐานอีเทอร์เน็ตซึ่งทำงานที่ความเร็ว 10 เมกกะบิตต่อวินาทีไว้หลายประเภทตามชนิดสายสัญญาณ

2.7 ความรู้เบื้องต้นเกี่ยวกับ TCP/IP [7]

TCP (Transmission Control Protocal) เป็นโปรโตคอลที่รับประกันการรับส่งข้อมูลระหว่างโฮสต์ กล่าวคือโปรโตคอลมีกลไกในการตรวจสอบและยืนยันว่ามีข้อมูลจากต้นทางไปจนถึงปลายทางเสมอ และหากข้อมูลถึงปลายทางก็จะมีข้อมูลตอบรับว่าถึงปลายทางแล้ว หากไม่มีสัญญาณตอบรับก็แสดงว่าข้อมูลไม่ถึงปลายทาง ดังนั้นแอพพลิเคชันที่มีความสำคัญจึงเลือกโปรโตคอลนี้ในการรับส่งข้อมูล กระบวนการรับส่งข้อมูลแบบนี้เองที่เป็นจุดขายของ TCP ที่เด่นและไม่มีใครมาแข่งได้ เพราะในการทำงานของคอมพิวเตอร์ข้อมูลทุกตัวอักษรมีความสำคัญอย่างยิ่งยวดเท่าเทียมกัน การที่มีตัวอักษรหายไปแม้เพียงตัวเดียวก็อาจจะทำให้โปรแกรมทำงานผิดพลาดได้เลยทั้งหมดทันที ดังนั้นแอพพลิเคชันที่ทราบความสำคัญในจุดนี้จึงเลือกมาใช้ TCP ในการสื่อสารกันแพร่หลาย ประกอบกับการที่โปรโตคอล TCP อยู่บนเลเยอร์ของ IP ซึ่งสามารถส่งข้อมูลได้ทั่วในโลก

2.7.1 การแบ่งชั้น (Layering)

TCP/IP เป็นชุดโปรโตคอลที่ประกอบด้วยโปรโตคอลย่อยหลายตัว โดยแต่ละตัวก็จะทำหน้าที่แต่ละชั้นหรือเลเยอร์ (Layer) ซึ่งรับผิดชอบและแปลความหมายของข้อมูลในแต่ละระดับของการสื่อสาร ซึ่งในภาพรวมแล้ว TCP/IP แบ่งออกเป็น 4 เลเยอร์ดังภาพที่ 2.2



ภาพที่ 2.2 TCP layer

หน้าที่ความรับผิดชอบแต่ละเลเยอร์มีดังนี้

1. Link Layer ในเลเยอร์นี้จะเป็นดีไวซ์ไดรเวอร์ที่ทำงานอยู่บนระบบปฏิบัติการแต่ละระบบทำหน้าที่รับผิดชอบในการส่งข้อมูลตั้งแต่ระดับกายภาพสัญญาณไฟฟ้า จนถึงการแปลความ

จากระดับสัญญาณไฟฟ้าจนเป็นข้อมูลทางคอมพิวเตอร์ โปรโตรคอลระดับนี้ได้แก่ Ethernet และ SLIP (Serial Line Internet Protocal)

2. Network Layer รับผิดชอบการรับ-ส่งข้อมูลในเน็ทเวิร์ก ส่งต่อข้อมูลไปถึงจุดหมายปลายทาง โปรโตคอลระดับนี้ ได้แก่ IP, ICMP, IGMP

3. Transport Layer รับผิดชอบในการรับส่งข้อมูลระหว่างโฮสต์หนึ่งไปยังอีกโฮสต์หนึ่ง และจะส่งข้อมูลขึ้นไปให้ Application Layer นำไปใช้งานต่อ มีโปรโตคอลที่จัดอยู่ในเลเยอร์นี้คือ TCP และ UDP ซึ่งมีลักษณะในการรับส่งข้อมูลที่แตกต่างกันออกไป

4. Application Layer เป็นเลเยอร์ที่แอพพลิเคชันเรียกใช้โปรโตคอลระดับล่าง ๆ ลงไปเพื่อวัตถุประสงค์ที่แตกต่างกัน เช่น FTP (File Tranfer Protocal) ใช้สำหรับรับส่งแฟ้มข้อมูลระหว่างโฮสต์ เป็นต้น

2.8 เครื่องมือตรวจวัดเพื่อเฝ้าระวังทางด้านรังสี [2]

เครื่องมือวัดปริมาณรังสีที่ทำการติดตั้งตามสถานีตรวจวัดปริมาณรังสี จะมีลักษณะแตกต่างกันไปตามวัตถุประสงค์ในการตรวจวัด ซึ่งอาจต้องการวัดรังสีแกมมาในบรรยากาศ ปริมาณรังสีในน้ำ ปริมาณรังสีจากปล่องควัน และปริมาณรังสีบริเวณใกล้พื้นที่ตั้งของโรงงานไฟฟ้านิวเคลียร์ เป็นต้น เครื่องวัดรังสีแต่ละประเภทจะเลือกใช้หัววัดชนิดต่าง ๆ เช่น หัววัดไกเกอร์แบบวัดฟลักซ์สูงและฟลักซ์ต่ำ หัววัดโซเดียมไอโอไดด์ หัววัดไอออไนเซชันแชมเบอร์ ในงานวัดรังสีรวมและใช้หัววัดรังสีกึ่งตัวนำชนิด HPGe ในการระบุชนิดไอโซโทปรังสี โดยสามารถแบ่งชนิดของเครื่องมือการตรวจวัดรังสีในสิ่งแวดล้อมนี้เป็นกลุ่มใหญ่ได้ 2 กลุ่ม คือ

1. Nuclide Specific Monitor เป็นระบบตรวจวัดรังสีซึ่งออกแบบให้วัดรังสีแอลฟา บีตา และแกมมา และสามารถวิเคราะห์ชนิดของไอโซโทปและปริมาณรังสี โดยวัดปริมาณรังสีจากสิ่งแวดล้อม และการสูบอากาศผ่านเส้นเทปกระดาษกรองแบบต่อเนื่อง บางเครื่องอาจออกแบบให้วัดเฉพาะไอโซโทปของไอโอดีน -131 และซีเซียม 137 ซึ่งจะพบในการรั่วไหลของรังสีจากแท่งเชื้อเพลิงนิวเคลียร์หรือออกแบบให้วัดเฉพาะทริเตรียมในบริเวณปฏิบัติการด้านเครื่องเร่งอนุภาค หรือเครื่องปฏิกรณ์ปรมาณูแบบฟิวชัน เป็นต้น

2. Dose Rate Monitor เป็นระบบตรวจวัดรังสีซึ่งออกแบบให้วัดปริมาณรังสีรวมในสภาพแวดล้อม โดยสามารถเลือกชนิดของหัววัดตามความไวในการตรวจวัดรังสี หรือ

ออกแบบให้สามารถวัดปริมาณรังสีรวมในช่วงพลังงานต่าง ๆ โดยเป็นเครื่องที่ติดตั้งภายในอาคาร ภายนอกอาคาร รวมทั้งบริเวณที่ต้องการวัดปริมาณรังสี

2.9 ระบบเฝ้าระวังรังสีแกมมาที่ใช้กันอยู่ในปัจจุบัน [8]

ในประเทศไทยมีการตรวจวัดปริมาณรังสีด้วยระบบเฝ้าตรวจปริมาณรังสีแกมมาในอากาศที่สำนักงานปรมาณูเพื่อสันติ (ปส.) และภูมิภาคต่าง ๆ ทั่วประเทศเพื่อเฝ้าตรวจระดับรังสีจากการทดลองระเบิดนิวเคลียร์และการดำเนินงานของสถานปฏิบัติการทางรังสีในประเทศและต่างประเทศที่อาจเป็นอันตรายต่อประชาชนและสิ่งแวดล้อม โดยการตรวจวัดรังสีแกมมาในอากาศแบบเรียลไทม์และส่งข้อมูลผลการตรวจวัดออนไลน์มายังศูนย์ข้อมูลเฝ้าตรวจกัมมันตภาพรังสี ณ สำนักงานปรมณูเพื่อสันติภาพ นอกจากนี้หากมีการตรวจวัดระดับรังสีแกมมาที่ผิดปกติได้ ระบบจะมีการเตือนให้ทราบเพื่อดำเนินการตรวจสอบ และประสานงานกับงานที่เกี่ยวข้องต่อไป

การตรวจวัดระดับรังสีแกมมาในอากาศ (Ambient Gamma Dose Rate) เป็นวิธีการที่นิยมใช้ในการเฝ้าตรวจหรือเฝ้าระวังอันตรายจากรังสีที่เกิดจากกิจกรรมของมนุษย์ เนื่องจากว่ากิจกรรมทางด้านรังสีขนาดใหญ่ที่มีการรั่วไหลของสารรังสีสู่สิ่งแวดล้อมนั้น สารรังสีจะแพร่กระจายไปได้ไกลโดยใช้ตัวกลาง คือ อากาศชั้นบน ซึ่งสารรังสีจะค้างอยู่ในบรรยากาศชั้นบนได้เป็นเวลานาน จึงสามารถถูกพัดพาไปได้เป็นระยะทางไกลจากจุดกำเนิด นอกจากนี้รังสีแกมมาเป็นรังสีที่ได้จากการสลายตัวของนิวไคลด์รังสีเกือบทั้งหมดโดยเฉพาะนิวไคลด์รังสีที่เกิดจากปฏิกิริยาฟิชชัน การตรวจวัดระดับรังสีแกมมาในอากาศนี้มีเครื่องวัดติดตั้งประจำสถานีตามภูมิภาคต่าง ๆ ของประเทศไทย ได้แก่

- สำนักงานปรมาณูเพื่อสันติ กรุงเทพฯ
- มหาวิทยาลัยเชียงใหม่ จังหวัดเชียงใหม่
- มหาวิทยาลัยขอนแก่น จังหวัดขอนแก่น
- มหาวิทยาลัยทักษิณ จังหวัดสงขลา
- มหาวิทยาลัยพะเยา จังหวัดพะเยา
- มหาวิทยาลัยอุบลราชธานี จังหวัดอุบลราชธานี
- สถานีวิจัยและฝึกอบรม วน. เกษตรตราด จังหวัดตราด
- สถานีวิจัยเพื่อการพัฒนาชายฝั่งอันดามัน จังหวัดระนอง

การรายงานผล

1. จะรายงานผล 2 ช่องทาง คือ
   1.1 จอ LED หน้าสำนักงานปรมาณูเพื่อสันติ
   1.2 เวปไซต์ของสำนักงานปรมาณูเพื่อสันติ (www.oaep.go.th)
2. ข้อมูลจะมีการปรับปรุงแบบวันต่อวัน
3. รายงานในหน่วยนาโนซีเวิร์ตต่อชั่วโมง (nSv/hr)
4. ค่าที่วัดได้หากต่ำกว่า 200 nSv/hr ถือว่าระดับรังสีแกมมาในอากาศอยู่ในระดับปกติที่วัดได้ทั่วไปในธรรมชาติ
5. หากค่าที่วัดได้สูงจนอาจเป็นอันตรายต่อประชาชนและสิ่งแวดล้อม สำนักงานปรมาณูเพื่อสันติและจะมีมาตรการรองรับเพื่อให้ประชาชนและสิ่งแวดล้อมปลอดภัยจากอันตรายของรังสี

ระบบตรวจวัดปริมาณรังสีแกมมาในอากาศ ใช้หัววัดรังสีแบบบรรจุด้วยแก๊ส (Gas-Filled Detector) ชนิด Geiger-Muller Counter และหัววัดรังสี Nal ทำการวัดปริมาณรังสีแกมมาเฉลี่ยในแต่ละชั่วโมง ส่งข้อมูลผลการตรวจวัดออนไลน์มายังศูนย์ข้อมูลเฝ้าตรวจกัมมันตภาพรังสี ณ สำนักงานปรมาณูเพื่อสันติภาพ และส่งสัญญาณเตือนภัยเมื่อปริมาณรังสีแกมมาเกินระดับธรรมชาติที่มีค่าเฉลี่ย 200 nGy/hr

ในประเทศญี่ปุ่น พื้นที่ซึ่งทางรัฐบาลจัดไว้สำหรับการพัฒนาด้านอุตสาหกรรมนิวเคลียร์ที่เมืองโตไก (Tokai) จะต้องมีแผนกเฉพาะกิจที่ทำหน้าที่ควบคุมและป้องกันผลกระทบทางรังสีต่อสิ่งแวดล้อม (Environmental Protection Section) ซึ่งขึ้นอยู่กับกองสุขภาพและระดับความปลอดภัยทางรังสี (Health and Safety Division) มีการวางสถานีเครือข่าย รวมทั้งสุ่มเก็บตัวอย่างพืชผักและสิ่งของปัจจัยในชีวิตประจำวันของประชากรในรัศมีควบคุมมาทำการตรวจวัดและประเมินระดับรังสีตามจุดต่าง ๆ สถานีเครือข่ายสำหรับการตรวจวัดปริมาณรังสีที่จัดไว้จะรายงานผลข้อมูลการวัดเข้าศูนย์ประมวลข้อมูลอัตโนมัติ (Environmental Data Automatic Processing System) โดยแต่ละสถานีจะติดตั้งอุปกรณ์และเครื่องมือวัดชนิดของรังสีและสารรังสีที่แตกต่างกัน หน่วยงานด้านความปลอดภัยจะต้องรายงานระดับรังสีที่วัดได้เป็นประจำทุก 3 เดือนหรือทุก ๆ 1 ปี ต่อคณะกรรมการควบคุมความปลอดภัยทางนิวเคลียร์ (Nuclear Safety Commission) และเผยแพร่ต่อสาธารณะชน [9]

ในประเทศสหรัฐอเมริกาบริเวณที่เป็นที่ตั้งของโรงไฟฟ้านิวเคลียร์ จำเป็นต้องมีการติดตั้งสถานีตรวจวัดระดับรังสีอันเป็นไปตามข้อบังคับของ NRC (Nuclear Regulatory Commission) ในการเดินเครื่องปฏิกรณ์นิวเคลียร์ซึ่งจะต้องมีการรายงานผลทางระดับของรังสีในสิ่งแวดล้อมบริเวณพื้นที่ใกล้เคียง (Off-site Radiological Environment Monitoring Reports) ทุก 6 เดือน หรือ 3 เดือน ต่อคณะกรรมการควบคุมการเดินเครื่องปฏิกรณ์นิวเคลียร์ โดยมีการแบ่งพื้นที่ควบคุมออกเป็น 2 โซน ตามกฎข้อบังคับที่ (10C FR 100.3) ดังนี้

1. โซนเอ็กซ์คลูซัน (Exclusion Zone) เป็นพื้นที่โดยรอบสถานที่ตั้งเครื่องปฏิกรณ์นิวเคลียร์ ซึ่งไม่อนุญาตให้มีผู้อยู่อาศัยถาวร ขึ้นอยู่กับขนาดกำลังของเครื่องปฏิกรณ์นิวเคลียร์และผลการคำนวณปริมาณรังสีจากไอโซโทปรังสีของไอโอดีน (I-131) ที่ตรวจพบในต่อมไทรอยด์ (Total Thyroid Dose)
2. โซนที่มีประชากรอยู่น้อย (Low population zone, LPZ) เป็นพื้นที่รอบนอกของโซนเอ็กคลูซัน

การตรวจวัดระดับรังสีในบริเวณที่มีการเดินเครื่องปฏิกรณ์นิวเคลียร์ อาจจะมีการส่งข้อมูลเข้าศูนย์ควบคุมสภาพอากาศและสิ่งแวดล้อมร่วมกับการตรวจวัดรังสีตามภูมิภาคต่าง ๆ และการสำรวจรังสีแบบเคลื่อนที่ (Mobile) ทั้งภาคพื้นดินและบนอากาศ ซึ่งการตรวจวัดระดับรังสีในลักษณะนี้มักจะใช้การสื่อสารด้วยระบบวิทยุ [10]

บทที่ 3

การพัฒนาระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

3.1 วัสดุและอุปกรณ์ที่ใช้ในการวิจัย

3.1.1 เครื่องสเกลเลอร์/เรตมิเตอร์ โมเดล 2200 ของบริษัท Ludlum

3.1.2 หัววัดรังสีไกเกอร์มูลเลอร์ รุ่น 44-7 ของบริษัท Ludlum

3.1.3 ไมโครคอนโทรลเลอร์ ET-dsPIC33WEB V1.0

3.1.4 จอแสดงผล Character LCD ขนาด 16 ตัวอักษร 2 บรรทัด

3.1.5 เครื่องโปรแกรม PIC/dsPIC รุ่น ET-PGMPIC USB หรือรุ่นอื่นๆ พร้อมสายเชื่อมต่อ USB

3.1.6 แหล่งจ่ายไฟสำหรับบอร์ด ET-dsPIC33WEB V1.0 สามารถจ่ายแรงดันไฟฟ้า 7-12 V กระแส 850 mA

3.1.7 คอมพิวเตอร์ชนิดพกพา หรือตั้งโต๊ะซึ่งมีพอร์ตเชื่อมต่อ Ethernet LAN

3.1.8 สาย LAN

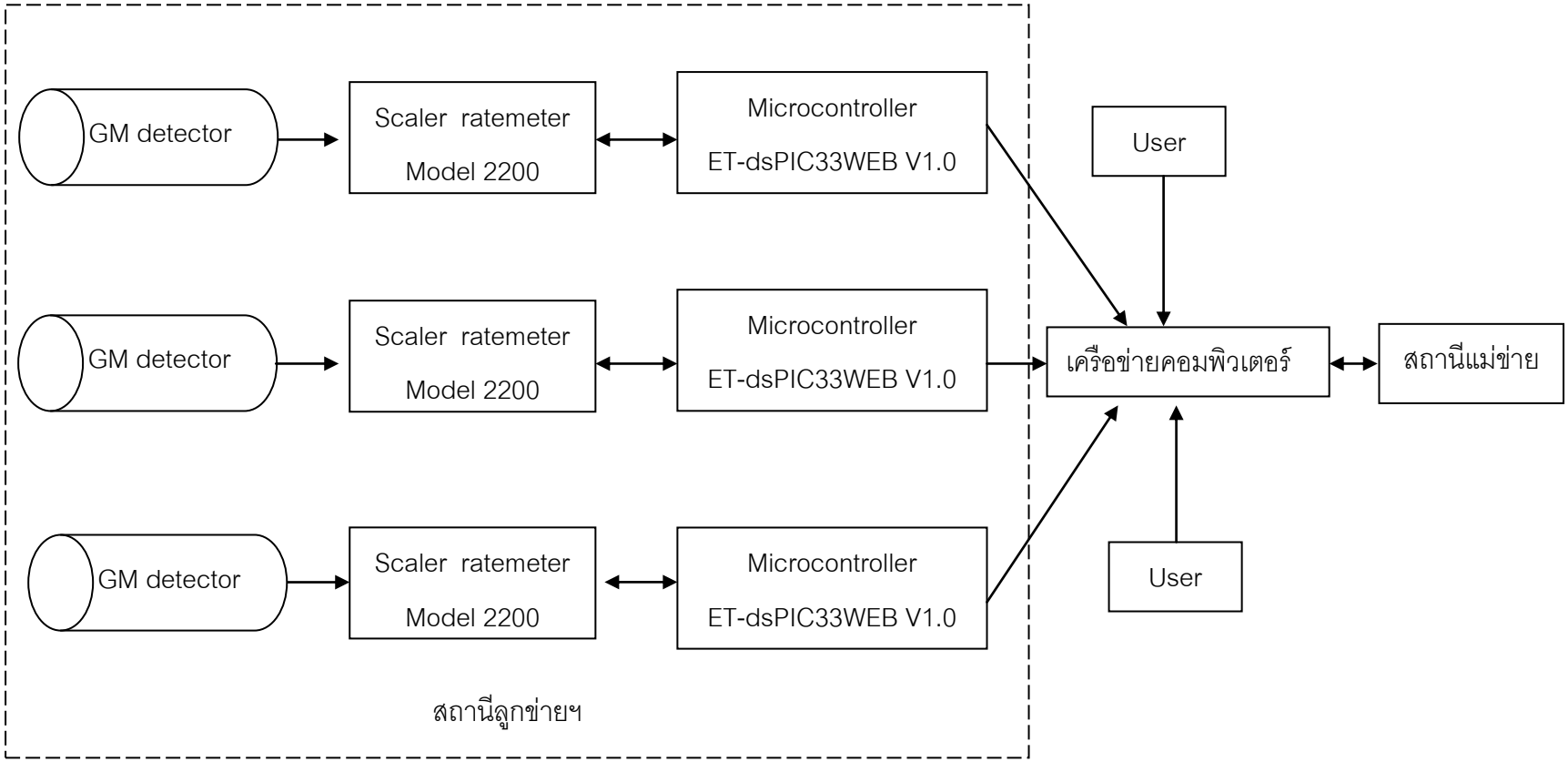
3.2 การออกแบบระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

ระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นที่พัฒนาขึ้น ประกอบด้วยส่วนประกอบสำคัญ 3 ส่วน คือ สถานีลูกข่ายสำหรับระบบตรวจวัดปริมาณรังสีแกมมาผ่านเครือข่ายท้องถิ่น สถานีแม่ข่ายสำหรับระบบตรวจวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นและโปรแกรมควบคุมการทำงานของระบบ ดังแสดงในภาพที่ 3.1 แต่ละส่วนจะมีรายละเอียดดังนี้

3.2.1 สถานีลูกข่ายสำหรับการตรวจวัดปริมาณรังสีแกมมา

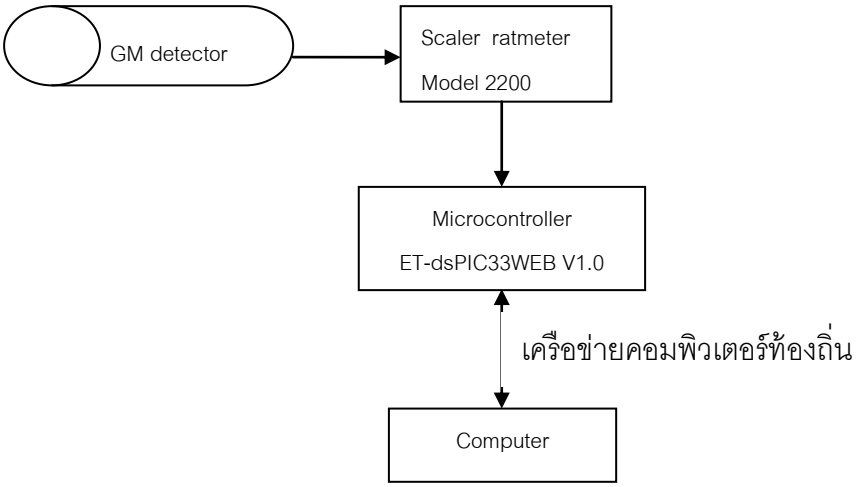
สถานีสำหรับการตรวจวัดปริมาณรังสีแกมมาที่ได้ออกแบบ มีส่วนประกอบดังนี้

1. หัววัดรังสีไกเกอร์มูลเลอร์ของบริษัท Ludlum รุ่น 44-7 (รายละเอียดภาคผนวก ก)
2. เครื่องวัดรังสี เครื่องสเกลเลอร์/เรตมิเตอร์ของบริษัท Ludlum รุ่น 2200 (รายละเอียดภาคผนวก ข.)
3. ไมโครคอนโทรลเลอร์ (รายละเอียดภาคผนวก ค)หรือไมโครคอมพิวเตอร์



ภาพที่ 3.1 ลักษณะระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

สถานีลูกข่ายฯ จะทำหน้าที่วัดปริมาณรังสีแกมมา และส่งข้อมูลการนับวัดรังสีแกมมาจากเครื่องสเกลเลอร์/เรตมิเตอร์ผ่านพอร์ตอนุกรม ดังแสดงรายละเอียดในภาพที่ 3.2 ในการควบคุมการทำงานของสถานีลูกข่ายฯ จะใช้ภาษาเบสิกในการเขียนโปรแกรมควบคุมการทำงานของสถานีลูกข่ายฯ



ภาพที่ 3.2 แผนภาพแสดงการทำงานของสถานีลูกข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

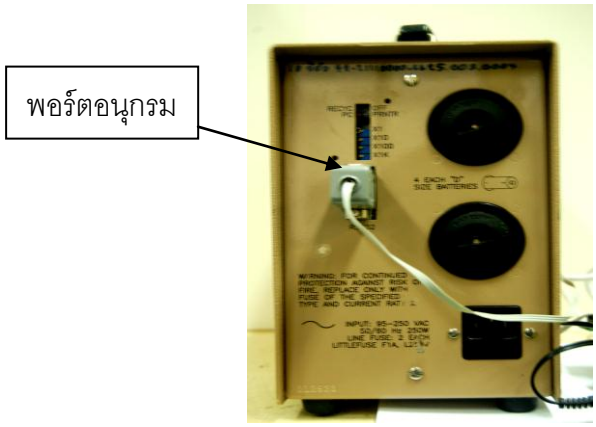
ภาพที่ 3.3 หัววัดรังสีไกเกอร์มูลเลอร์ ของบริษัท Ludlum รุ่น 44-7

ภาพที่ 3.4 เครื่องสเกลเลอร์/เรตมิเตอร์ โมเดล 2200 ของบริษัท Ludlum

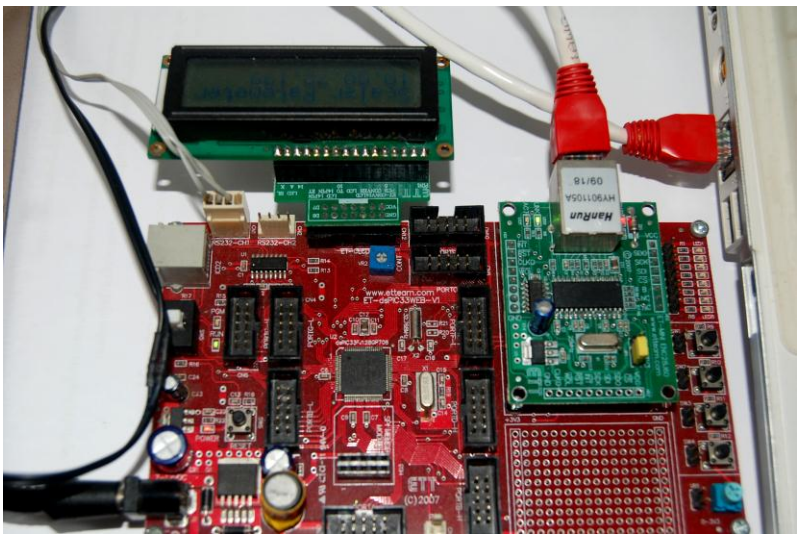
การเชื่อมต่อเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์ เชื่อมต่อกันทางพอร์ตอนุกรม ดังแสดงในภาพที่ 3.5 มีรายละเอียดในการเชื่อมต่อในตารางที่ 3.1 อัตราเร็วในการรับส่งข้อมูล 2400 บิต ต่อวินาที กลุ่มบิตข้อมูลมีจำนวน 8 บิต ไม่ใช้พาริตีในการตรวจสอบข้อมูล และจำนวนบิตสิ้นสุดข้อมูลมีจำนวน 1 บิต

ตารางที่ 3.1 รายละเอียดแสดงการเชื่อมต่อระหว่างไมโครคอมพิวเตอร์กับเครื่องสเกลเลอร์/เรตมิเตอร์

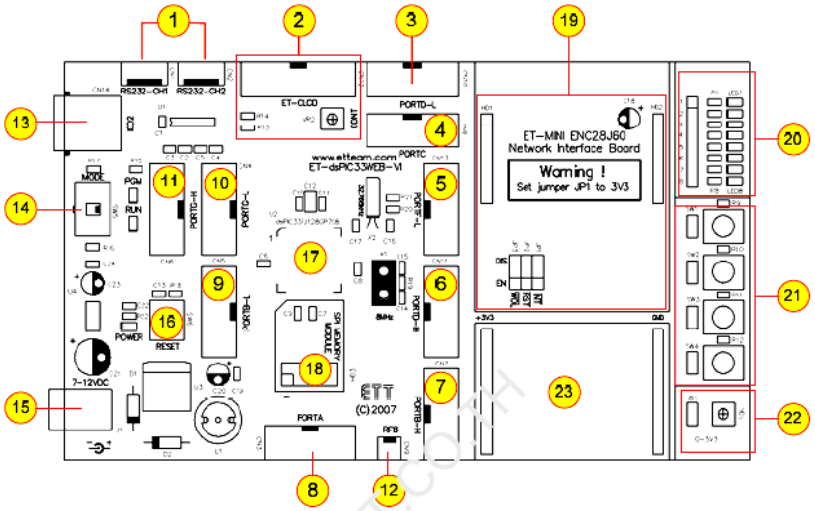
เครื่องสเกลเลอร์/เรตมิเตอร์		ไมโครคอมพิวเตอร์	
ลำดับขา	สายสัญญาณ	ลำดับขา	สายสัญญาณ
2	TD	2	TD
3	RD	3	RD
5	GND	5	GND
7	CTR	7	CTR
8	RTS	8	RTS



ภาพที่ 3.5 เครื่องสเกลเลอร์/เรตมิเตอร์ เชื่อมต่อกับไมโครคอนโทรลเลอร์ผ่านพอร์ตอนุกรม



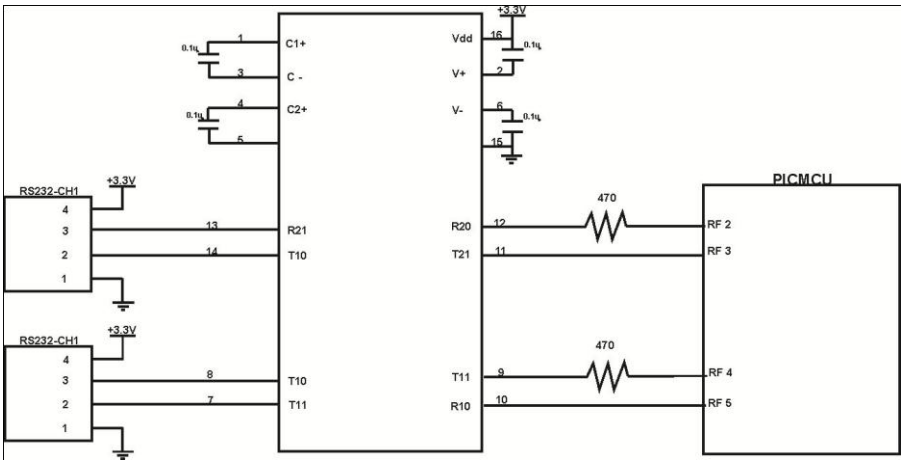
ภาพที่ 3.6 การเชื่อมต่อ RS232 กับบอร์ดไมโครคอนโทรลเลอร์



ภาพที่ 3.7 โครงสร้างบอร์ด ET-dsPIC33WEB V1.0

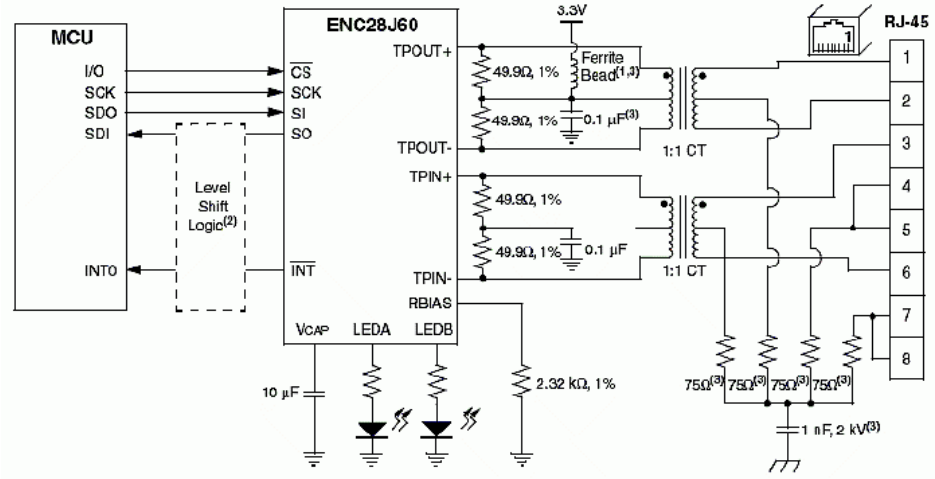
หมายเลข 1 พอร์ตเชื่อมต่อสัญญาณแบบ RS-232 จำนวน 2 พอร์ต มีวงจรการเชื่อมต่อดังต่อไปนี้

การติดต่อสื่อสารแบบ RS-232 สำหรับการติดต่อสื่อสารในโหมดนี้ จะต้องมีไอซีคอยทำหน้าที่เปลี่ยนข้อมูลให้อยู่ในรูปแบบอนุกรม ซึ่งทาง ETT ผู้ผลิตบอร์ดได้เตรียมไว้ในบอร์ดไมโครคอนโทรลเลอร์เรียบร้อยแล้ว เบอร์ IC-RS232 ที่ใช้งาน คือ max232 มีการเชื่อมต่อกับไมโครคอนโทรลเลอร์ สำหรับช่อง RS-232 ที่เลือกใช้ในที่นี้ คือ RS232-CH1 (หมายเลข 1) โดยการเชื่อมต่อแสดงในภาพที่ 3.8



ภาพที่ 3.8 การเชื่อมต่อ max232 กับไมโครคอนโทรลเลอร์

หมายเลข 19 ขั้วสัญญาณเชื่อมต่อกับโมดูลสื่อสาร Ethernet รุ่น ET-MINI ENC28J60



ภาพที่ 3.9 โครงสร้างโมดูลสื่อสาร Ethernet รุ่น ET-MINI ENC28J60

การติดต่อสื่อสารแบบ TCP-IP เป็นโปรโตคอลการสื่อสารอีกรูปแบบหนึ่ง ที่มีความนิยมกันอย่างกว้างขวางเนื่องจากความแน่นอนในการรับส่งข้อมูล มีความถูกต้องแม่นยำสูง IC ที่จะใช้ในการถอดรหัส เพื่อติดต่อสื่อสารระหว่างไมโครคอนโทรลเลอร์และคอมพิวเตอร์นั้นคือเบอร์ ENC28J60 ดังแสดงในภาพที่ 3.9 ของทาง microchip ซึ่งทางบริษัท ETT ได้จัดทำเป็น mini-module ไว้ใช้ประกอบกับบอร์ด dspic33 จึงเป็นการสะดวกในการพัฒนาโปรแกรม

3.2.2 สถานีแม่ข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

สถานีแม่ข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นที่ออกแบบและพัฒนาขึ้นนี้มีหน้าที่ ควบคุมการทำงานของระบบลูกข่ายฯ จัดเก็บผลการตรวจวัดปริมาณรังสีแกมมาไว้ในฐานข้อมูล และแสดงผลในรูปแบบของเว็บเพจ สถานีแม่ข่ายฯ นี้ใช้ Microsoft Visual Basic ในการพัฒนาโปรแกรมควบคุม

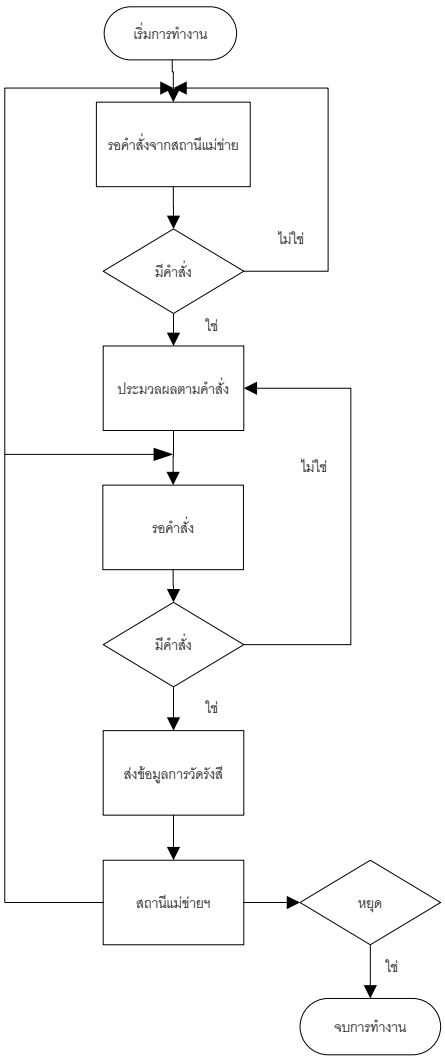
3.3 โปรแกรมควบคุมการทำงานของระบบเฝ้าระวังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

โปรแกรมการควบคุมการทำงานของระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นแบ่งออกเป็น 3 ส่วน คือ

- โปรแกรมสำหรับสถานีลูกข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น
- โปรแกรมสำหรับสถานีแม่ข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น
- โปรแกรมสำหรับการนำเสนอข้อมูลในรูปแบบของเว็บเพจ

3.3.1 โปรแกรมควบคุมการทำงานของสถานีลูกข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

เครื่องสเกลเลอร์/เรตมิเตอร์จะทำการเชื่อมต่อกับไมโครคอนโทรลเลอร์ผ่านทางพอร์ตอนุกรม เครื่องสเกลเลอร์/เรตมิเตอร์จะส่งข้อมูลที่ได้จากการวัดปริมาณรังสีแกมมาผ่านไมโครคอนโทรลเลอร์ทางพอร์ตอนุกรม ไมโครคอนโทรลเลอร์จะส่งข้อมูลการวัดรังสีผ่านระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่ายฯ โดยการโปรแกรมด้วยภาษาเบสิกที่มีโฟลว์ชาร์ตแสดงการทำงานดังภาพที่ 3.10 และมีรายละเอียดโปรแกรมในภาคผนวก ง



ภาพที่ 3.10 โฟล์วชาร์ตแสดงขั้นตอนการทำงานของโปรแกรมของสถานีลูกข่ายฯ

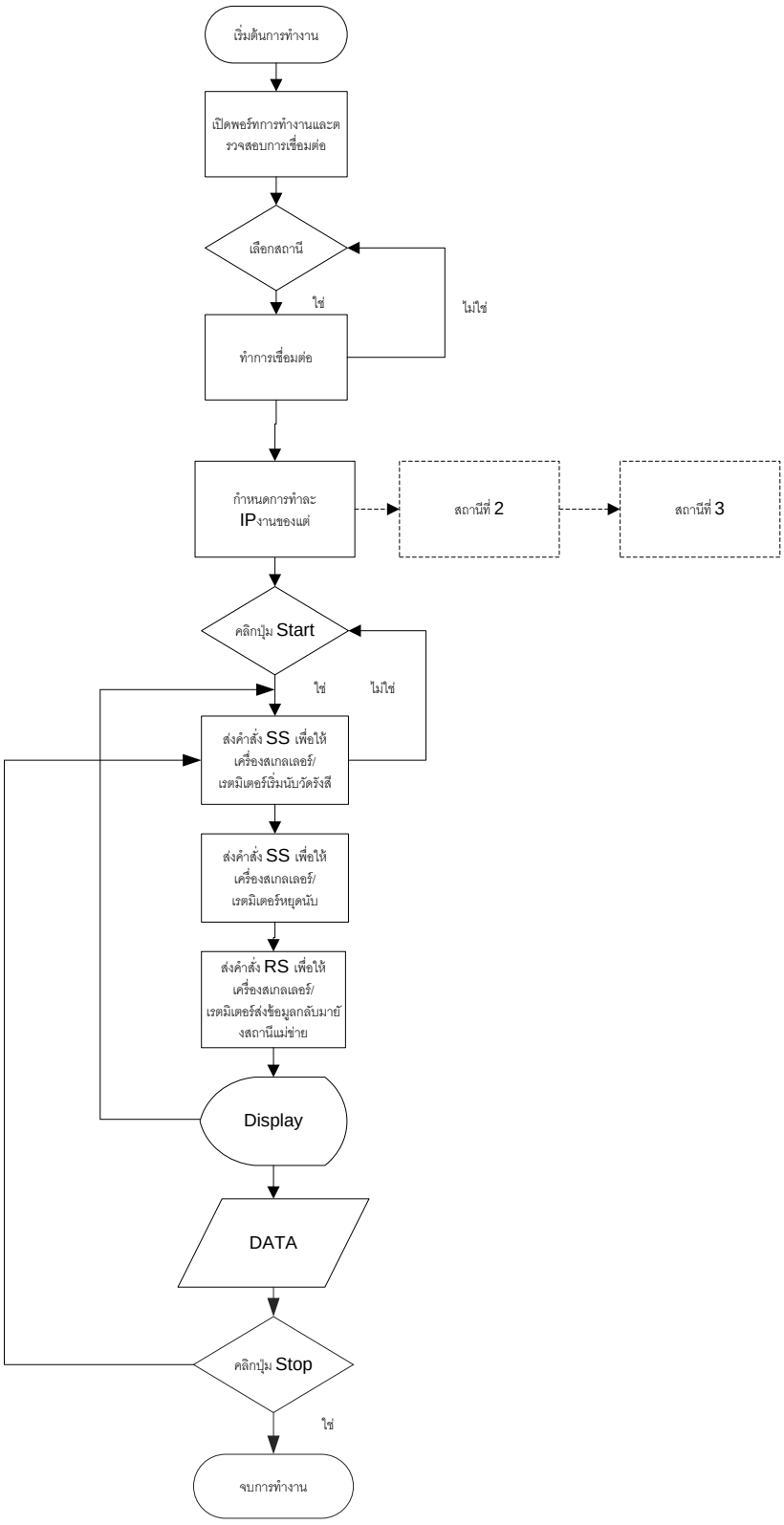
ขั้นตอนในการทำงานของโปรแกรมของสถานีลูกข่ายฯ ดังนี้

1. เริ่มต้นการทำงาน
2. เครื่องสเกลเลอร์/เรตมิเตอร์รอคำสั่งเริ่มนับจากสถานีแม่ข่ายฯ ที่จะส่งผ่านมาทางไมโครคอนโทรลเลอร์
3. เมื่อมีคำสั่ง SS ส่งมายังเครื่องสเกลเลอร์/เรตมิเตอร์ เครื่องสเกลเลอร์/เรตมิเตอร์จะเริ่มทำการวัดรังสี

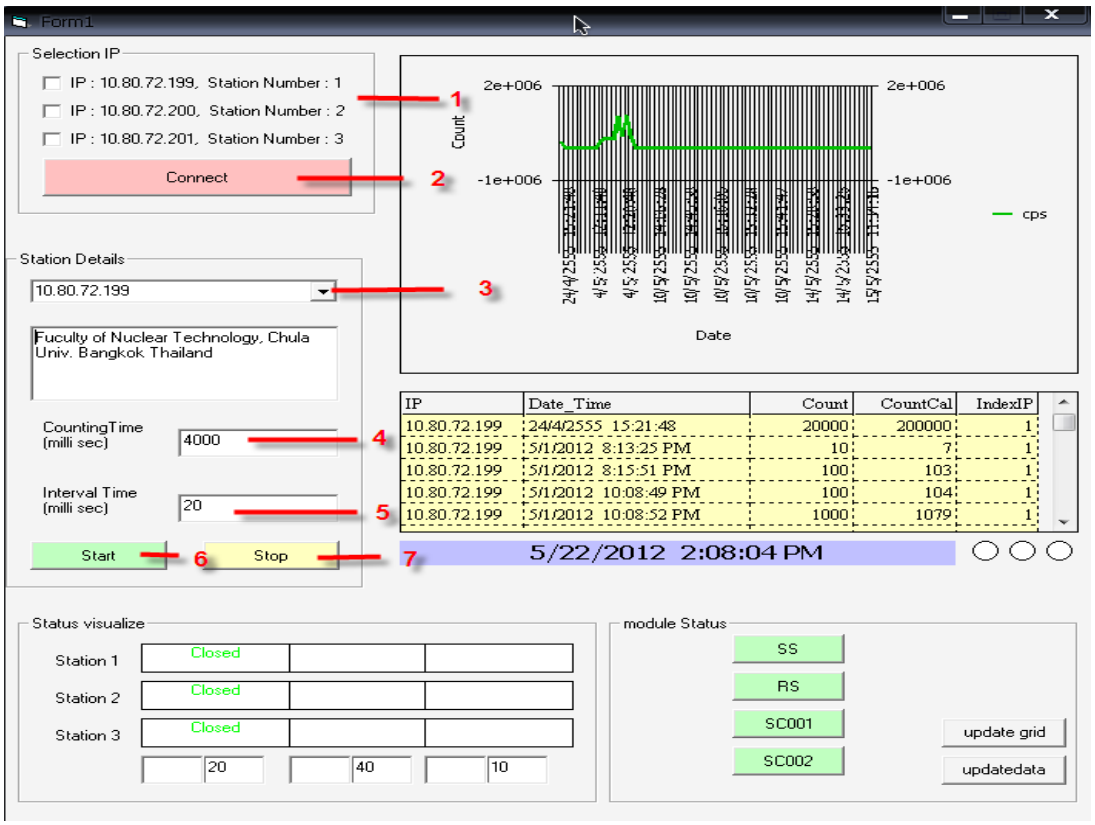
4. เครื่องสเกลเลอร์/เรตมิเตอร์จะทำการวัดรังสีครบเวลาที่กำหนดไว้ซึ่งเวลาที่กำหนดจะถูกกำหนดโดยสถานีแม่ข่ายฯ
5. เมื่อครบเวลาที่กำหนดจะมีคำสั่ง SS ส่งไปยังเครื่องสเกลเลอร์/เรตมิเตอร์ อีกครั้งเพื่อให้เครื่องสเกลเลอร์/เรตมิเตอร์จะหยุดวัดรังสี
6. เมื่อมีคำสั่ง RS ส่งไปยังเครื่องสเกลเลอร์/เรตมิเตอร์ เครื่องวัดจะส่งข้อมูลการวัดรังสีให้สถานีแม่ข่ายฯ
7. เครื่องสเกลเลอร์/เรตมิเตอร์จะเริ่มต้นการทำงานอีกครั้งตามเวลาที่ถูกกำหนดโดยสถานีแม่ข่ายฯ
8. ระบบทำงานอย่างต่อเนื่อง
9. เมื่อมีคำสั่งหยุดการทำงาน ระบบจะหยุดส่งข้อมูล
10. สิ้นสุดการทำงาน

3.3.2 โปรแกรมควบคุมการทำงานของสถานีแม่ข่ายระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

สถานีแม่ข่ายจะเป็นตัวควบคุมการทำงานของเครื่องสเกลเลอร์/เรตมิเตอร์ และยังทำหน้าที่เป็นตัวรับข้อมูล เก็บบันทึกผลข้อมูลในรูปแบบของฐานข้อมูล ทั้งยังแสดงผลบนไมโครคอมพิวเตอร์ และเจ้าหน้าที่ผู้ควบคุมสามารถควบคุมการทำงานของระบบวัดรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นผ่านสถานีแม่ข่ายฯ ได้โดยตรง ซึ่งการทำงานของสถานีแม่ข่ายฯ มีโฟลว์ชาร์ตการทำงานดังภาพที่ 3.11 ซึ่งมีรายละเอียดของโปรแกรมดังในภาคผนวก จ. และมีรายละเอียดหน้าจอของโปรแกรมของผู้ควบคุมสถานีแม่ข่ายฯ ดังภาพที่ 3.12



ภาพที่ 3.11 โฟล์วชาร์ตแสดงขั้นตอนการทำงานของโปรแกรมของระบบแม่ข่ายฯ



ภาพที่ 3.12 ภาพแสดงรายละเอียดหน้าจอของโปรแกรมของผู้ควบคุมสถานีแม่ข่ายฯ

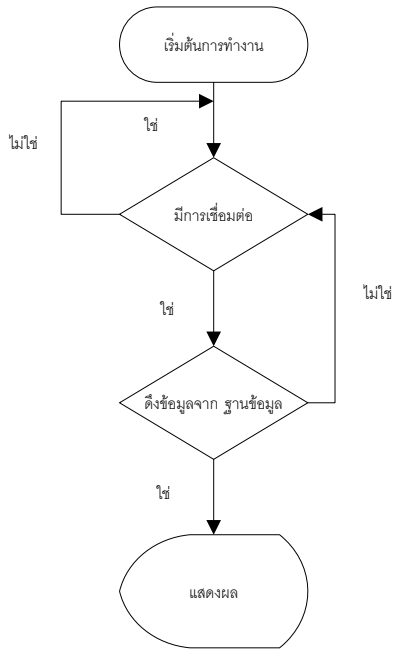
ขั้นตอนในการทำงานของโปรแกรมของผู้ควบคุมสถานีแม่ข่ายฯ ดังนี้

1. เริ่มต้นการทำงาน
2. ตรวจสอบสถานะการเชื่อมต่อ (หมายเลข 1)
   - เลือก IP address ของแต่ละสถานีที่ต้องการเฝ้าระวังตรวจวัดระดับรังสีแกมมา
3. ทำการเชื่อมต่อสถานีตรวจวัด (หมายเลข 2)
4. เลือกตั้งค่าการทำงานของแต่ละสถานี (หมายเลข 3)
   ก. ช่วงเวลาในการตรวจวัด (หมายเลข 4)
   ข. รอบเวลาในการวัดแต่ละครั้ง (หมายเลข 5)
5. เลือก Start ระบบจะทำการเชื่อมต่อการรับส่งข้อมูลผ่านทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่น (หมายเลข 6)
6. ส่งสัญญาณคำร้องขอไปยังเครื่องวัดรังสีผ่านระบบทางระบบเครือข่ายท้องถิ่น
7. จัดเก็บข้อมูลลงฐานข้อมูล

8. สั่ง Stop เพื่อหยุดการทำงาน (หมายเลข 7)

3.3.3 โปรแกรมสำหรับการนำเสนอข้อมูลในรูปแบบของเว็บเพจ

เมื่อระบบเฝ้ารังสีแกมมาผ่านเครื่อข่ายคอมพิวเตอร์ท้องถิ่นทำการวัดปริมาณรังสี ปริมาณรังสีที่วัดได้จะถูกเก็บบันทึกไว้เป็นฐานข้อมูลซึ่งใช้ Microsoft Access ในการเก็บรวบรวมข้อมูล และระบบสามารถนำเสนอข้อมูลในรูปแบบของเว็บเพจได้ ในการนำเสนอข้อมูลในรูปแบบของเว็บเพจ ใช้โปรแกรม Microsoft Visual Studio ในการดึงข้อมูลจากโปรแกรม Microsoft Access ปริมาณรังสีที่ได้จะถูกประมวลผล และนำมาเสนอบนเว็บเพจ หน้าเว็บเพจสร้างด้วยโปรแกรม Dreamweaver CS 5 มีโฟล์วชาร์ตแสดงขั้นตอนการทำงานของโปรแกรมของการนำเสนอข้อมูลในรูปแบบของเว็บเพจ ดังแสดงในภาพที่ 3.13



ภาพที่ 3.13 โฟล์วชาร์ตแสดงขั้นตอนการทำงานของโปรแกรมของการนำเสนอข้อมูลในรูปแบบของเว็บเพจ

บทที่ 4

การทดสอบการทำงานของระบบและผลการทดสอบ

การดำเนินงานวิจัยประกอบด้วยการออกแบบระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น และการพัฒนาโปรแกรมคำสั่งการเชื่อมต่อสัญญาณพร้อมทั้งแสดงสงผลและนำข้อมูลที่ได้มานำเสนอในรูปแบบของเว็บเพจ มีลำดับขั้นตอนการทดสอบดังนี้

1. การทดสอบการรับส่งข้อมูลระหว่างเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์
2. การทดสอบการรับส่งข้อมูลที่ได้จากการวัดรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นโดยส่งข้อมูลผ่านทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นท้องถิ่นเข้าสู่สถานีแม่ข่ายฯ
3. การสอบเทียบมาตรฐานระบบวัดรังสีแกมมา
4. การทดสอบการทำงานของระบบวัดรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น
5. การทดสอบระบบการนำเสนอข้อมูลในรูปแบบของเว็บเพจ

4.1 การทดสอบระบบการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์

การทดสอบระบบการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์เป็นการทดสอบความถูกต้องในการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์ โดยมีรายละเอียดการทดลองดังนี้

4.1.1 เครื่องมือและอุปกรณ์การทดลอบ

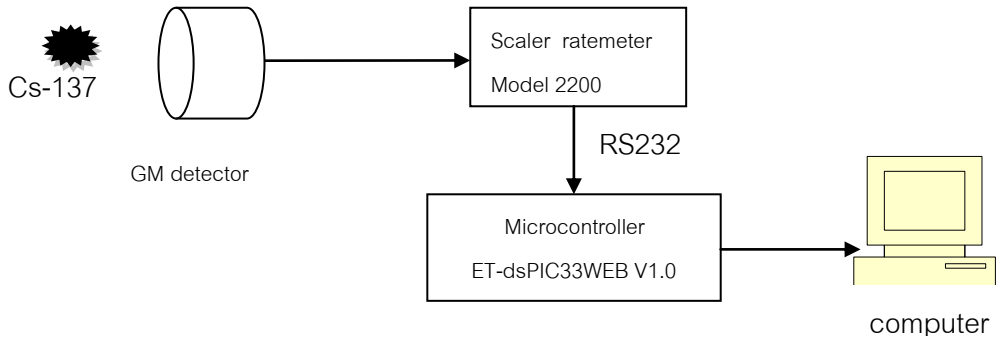
1. เครื่องสเกลเลอร์/เรตมิเตอร์ โมเดล 2200 ของบริษัท Ludlum
2. ไมโครคอนโทรลเลอร์ ET-dsPIC33WEB V1.0
3. หัววัดรังสีไกเกอร์มูลเลอร์ ของบริษัท Ludlum รุ่น 44-7

4. ต้นกำเนิดรังสี Cs-137 ความแรงรังสี 92μCi

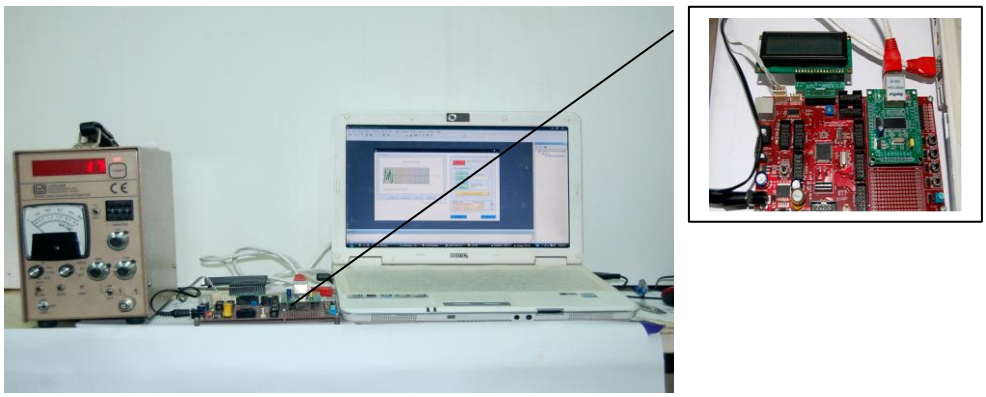
5. คอมพิวเตอร์โน๊ตบุ๊ค 1 เครื่อง

4.1.2 ขั้นตอนการทดสอบความถูกต้องในการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์

จัดระบบตามภาพที่ 4.1 จากนั้นทำการทดสอบการทำงานโดยการใช้ โปรแกรม Microsoft Visual Basic เขียนโปรแกรมในการทดสอบ โดยส่งรหัส SS เพื่อสั่งให้เครื่องสเกลเลอร์/เรตมิเตอร์นับรังสี ส่งรหัส SS เพื่อทดสอบการหยุดการนับรังสี และส่งรหัส RS เพื่อให้สเกลเลอร์/เรตมิเตอร์ ส่งข้อมูลผลการนับรังสีผ่านพอร์ตอนุกรม



ภาพที่ 4.1 แผนภาพการจัดระบบการทดสอบความถูกต้องในการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์

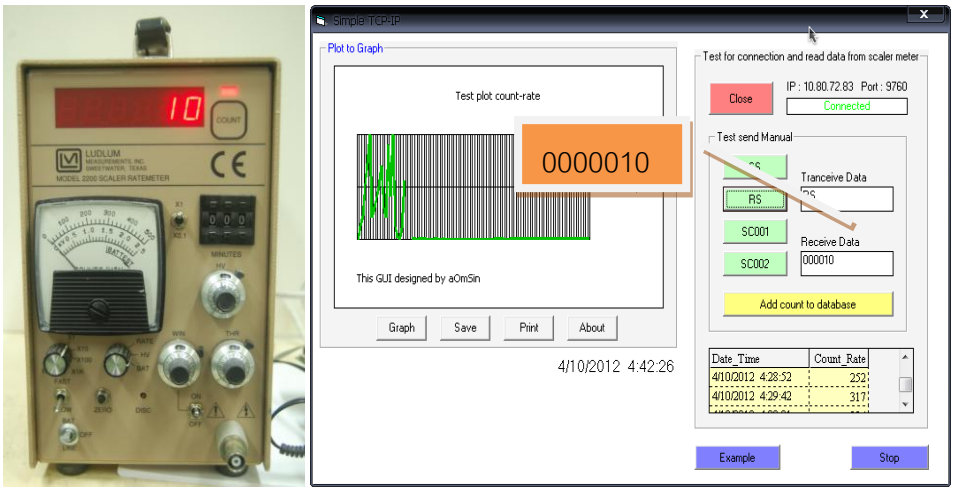


ภาพที่ 4.2 การจัดระบบการทดสอบความถูกต้องในการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์

จากการทดสอบการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์ได้ข้อมูลดังแสดงในตารางที่ 4.1

ตารางที่ 4.1 ผลการแสดงการทดสอบการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทรลเลอร์

ครั้งที่	ผลการนับรังสีแกมมาที่แสดงบนเครื่องสเกลเลอร์/เรตมิเตอร์	ผลการนับรังสีแกมมาที่แสดงบนไมโครคอมพิวเตอร์
1	10	10
2	15	15
3	17	17
4	22	22
5	23	23
6	27	27
7	31	31
8	34	34
9	37	37
10	40	40



ภาพที่ 4.3 เครื่องสเกลเลอร์/เรตมิเตอร์และหน้าจอคอมพิวเตอร์ที่แสดงผลการวัดรังสีแกมมา

จากข้อมูลในตารางที่ 4.1 พบการรับส่งข้อมูลระหว่างเครื่องสเกลเลอร์/เรตมิเตอร์สามารถและไมโครคอนโทรลเลอร์ ทำได้อย่างถูกต้องไม่พบความผิดพลาดใด

4.2 การทดสอบการรับส่งข้อมูลที่ได้จากการวัดรังสีแกมมาโดยส่งข้อมูลผ่านระบบเครือข่ายท้องถิ่นเข้าสู่สถานีแม่ข่ายฯ

การทดสอบระบบการรับส่งข้อมูลที่ได้จากการวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นโดยส่งข้อมูลผ่านทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่าย เป็นการทดสอบความถูกต้องในการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์ผ่านทางไมโครคอนโทรลเลอร์ โดยไมโครคอนโทรลเลอร์จะเชื่อมต่อกับสถานีแม่ข่ายฯ และส่งผ่านข้อมูลทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่น โดยมีรายละเอียดการทดสอบดังนี้

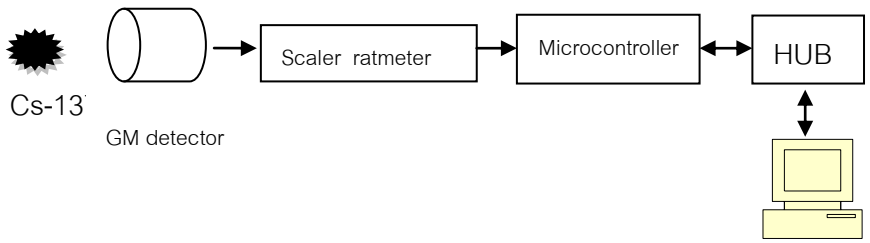
4.2.1 เครื่องมือและอุปกรณ์การทดสอบ

1. เครื่องสเกลเลอร์/เรตมิเตอร์ โมเดล 2200 ของบริษัท Ludlum
2. ไมโครคอนโทรลเลอร์ ET-dsPIC33WEB V1.0
3. หัววัดรังสีไกเกอร์มูลเลอร์ ของบริษัท Ludlum รุ่น 44-7
4. ต้นกำเนิดรังสี Cs-137 ความแรงรังสี 92 μCi
5. คอมพิวเตอร์สถานีแม่ขาย
6. สาย LAN(RJ-45) สายตรง
7. เราท์เตอร์

4.2.2 ขั้นตอนการทดสอบระบบการรับส่งข้อมูลที่ได้จากการวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น โดยส่งข้อมูลผ่านทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่ายฯ

จัดระบบการทดสอบดังภาพที่ 4.4 จากนั้นทำการทดสอบการทำงานโดยการใช้โปรแกรม Microsoft Visual Basic เขียนโปรแกรมเพื่อใช้ในการทดสอบการรับส่งข้อมูลผ่านระบบเครือข่ายคอมพิวเตอร์ท้องถิ่น ซึ่งมีขั้นตอนการทดสอบดังนี้

1. ทำการเลือกสถานีที่ลูกข่ายฯ (สถานีลูกข่ายฯ อาจมีมากกว่า 1 สถานี)
2. ทำการเชื่อมต่อโปรแกรมระบบระหว่างสถานีลูกข่ายฯ และสถานีแม่ข่ายฯ
3. ทำการตั้งค่าการทำงานของสถานีลูกข่ายฯ ในที่นี้เรากำหนดให้เครื่องสเกลเลอร์/เรตมิเตอร์วัดปริมาณรังสีทุก ๆ 60 วินาที และทำการเก็บข้อมูลหลังจากการวัดปริมาณรังสีทุก ๆ 30 วินาที และทำงานเริ่มการทำงานใหม่อย่างต่อเนื่องทุก ๆ 60 วินาทีจนกว่าจะมีการสั่งหยุดการทำงาน
4. กดสตาร์ทที่หน้าจอของตัวโปรแกรมควบคุมการทำงานที่สถานีแม่ข่ายฯ เพื่อเริ่มต้นการทำงาน



ภาพที่ 4.4 การจัดระบบการทดสอบการรับส่งข้อมูลที่ได้จากการวัดปริมาณรังสี แกมมาผ่านเครือข่ายท้องถิ่นโดยส่งข้อมูลผ่านทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่ายฯ

ภาพที่ 4.5 ภาพแสดงการจัดระบบการทดสอบการรับส่งข้อมูลที่ได้ จากการวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น โดยส่งข้อมูลผ่านทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่ายฯ

จากการทดสอบการรับส่งข้อมูลที่ได้จากการวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น โดยส่งข้อมูลผ่านทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่าย ได้ผลดัง ตารางที่ 4.2

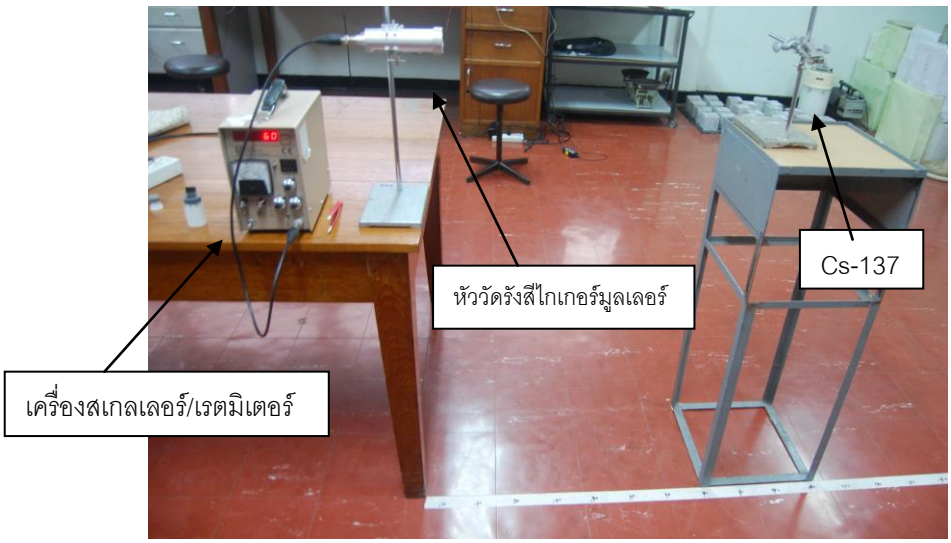
ตารางที่ 4.2 ผลการทดสอบการรับส่งข้อมูลที่ได้จากการวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่โดยส่งข้อมูลผ่านทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่าย

Date_Time	จำนวนรังสีที่ได้จากการทดสอบระบบ	จำนวนรังสีจากหน้าจอของเครื่องสเกลเลอร์/เรตมิเตอร์
4/4/2012 10:04:50 PM	0	0
4/10/2012 4:29:42 PM	317	317
4/10/2012 4:28:52 PM	252	252
4/10/2012 4:28:02 PM	220	220
4/10/2012 4:27:12 PM	192	192
4/10/2012 4:26:22 PM	192	192

จากข้อมูลในตารางที่ 4.2 พบว่าระบบสามารถรับส่งข้อมูลที่ได้จากการวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นโดยส่งข้อมูลผ่านทางระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่ายฯ ระบบสามารถแสดงผลได้อย่างถูกต้อง ไม่มีความผิดพลาดใดๆ

4.2.3 การสอบเทียบมาตรฐานระบบวัดรังสีแกมมา

ในการสอบเทียบมาตรฐานระบบวัดรังสีแกมมา ใช้ต้นกำเนิดรังสีมาตรฐาน Cs-137 ความแรงรังสี 92 μCi เพื่อนำไปคำนวนปริมาณรังสีที่ระยะต่าง ๆ ดังแสดงในตารางที่ 4.3 ในการสอบเทียบมาตรฐานระบบวัดรังสีแกมมาจัดอุปกรณ์ดังภาพที่ 4.6



ภาพที่ 4.6 การจัดอุปกรณ์ในการสอบเทียบมาตรฐานระบบวัดรังสีแกมมา

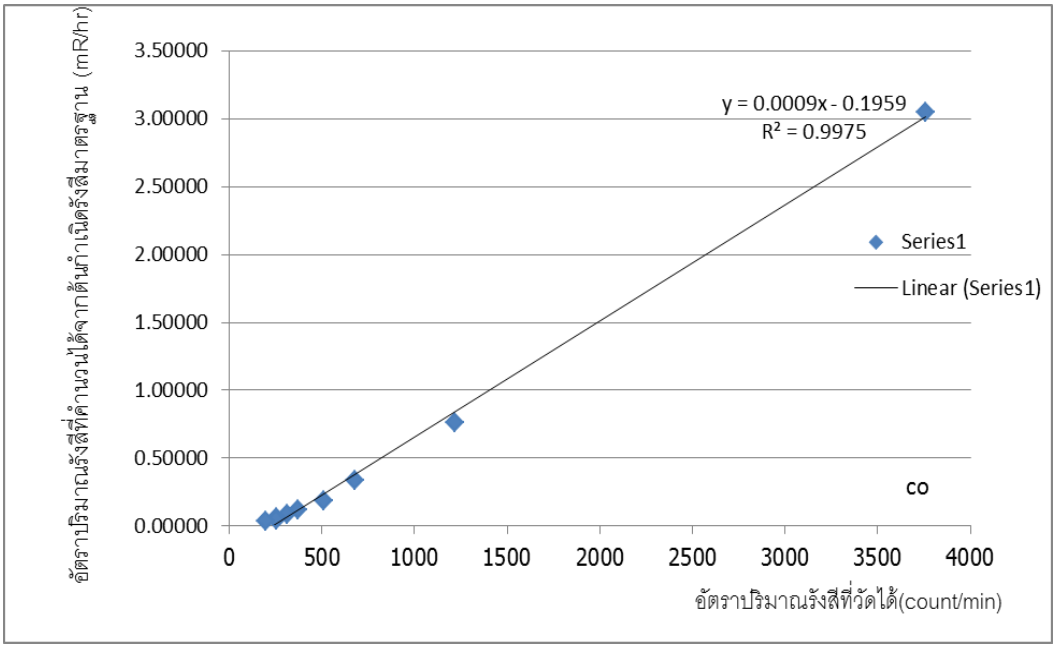
การสอบเทียบมาตรฐานระบบวัดรังสีแกมมาสอบเทียบที่ระยะห่างระหว่างต้นกำเนิดรังสีและหัววัดรังสีตั้งแต่ 10 ถึง 100 เซนติเมตรโดยเพิ่มระยะห่างครั้งละ 10 เซนติเมตร ผลการทดสอบแสดงใน ตารางที่ 4.3

ตารางที่ 4.3 แสดงผลการสอบเทียบมาตรฐานระบบวัดรังสีแกมมา

ระยะห่าง (cm)	อัตราปริมาณรังสีที่คำนวณได้จากต้นกำเนิดรังสีมาตรฐาน (mR/hr)	อัตราปริมาณรังสีที่วัดได้ (count/min)
10	3.04590	3761
20	0.76148	1220
30	0.33843	628
40	0.19037	510
50	0.12184	371
60	0.08461	314

ตารางที่ 4.3 ผลการสอบเทียบมาตรฐานระบบวัดรังสีแกมมา(ต่อ)

70	0.06216	258
80	0.04759	255
90	0.03760	196
100	0.03046	169



ภาพที่ 4.7 กราฟแสดงความสัมพันธ์ระหว่าง อัตราปริมาณรังสีที่คำนวณจากต้นกำเนิดรังสีมาตรฐาน (mR/hr) กับอัตราปริมาณรังสีที่วัดได้ (count/min)

4.2.4 การทดสอบความถูกต้องของระบบวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น

เพื่อทำการทดสอบความถูกต้องของการวัดปริมาณรังสีแกมมาของระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น ภายหลังจากได้นำสมการความสัมพันธ์ที่ได้จากการสอบเทียบใส่ลงในโปรแกรมควบคุมการทำงานของสถานีแม่ข่ายฯ จึงต้องมีการทดสอบการวัด

ปริมาณรังสีแกมมาของระบบทั้งหมด โดยการวัดอัตราปริมาณรังสีแกมมาจากต้นกำเนิดรังสีมาตรฐานที่ระยะต่าง ๆ ผลการวัด ดังแสดงในตารางที่ 4.4

ตารางที่ 4.4 ผลการทดสอบความถูกต้องของระบบ

อัตราปริมาณรังสีที่คำนวณจากต้นกำเนิดรังสีมาตรฐาน (mR/hr)	อัตราปริมาณรังสีที่วัดได้ (mR/hr)	ความแตกต่าง (mR/hr)	เปอร์เซ็นต์ความแตกต่าง
3.04590	3.1296	0.0837	2.75
0.33843	0.3683	0.0298	8.83
0.19037	0.2064	0.0160	8.42
0.12184	0.1299	0.0081	6.62
0.08461	0.0918	0.0072	8.50

จากผลการทดลองพบว่า ระบบสามารถแสดงค่าที่ได้จากวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นในหน่วย mR/hr ได้อย่างถูกต้อง โดยมีเปอร์เซ็นต์ค่าความแตกต่างไม่เกิน ± 10 %

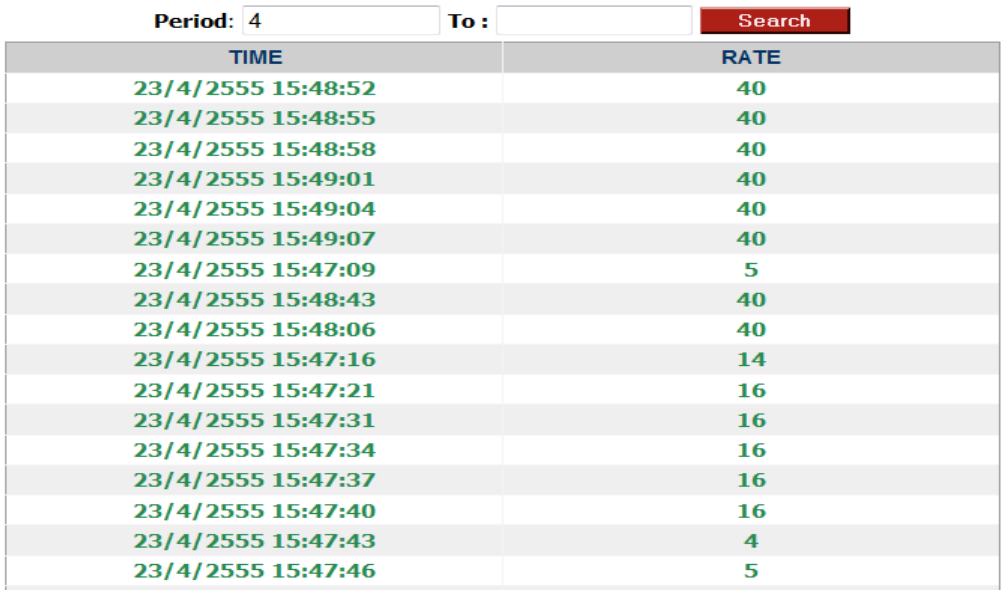
4.3 การทดสอบการนำเสนอข้อมูลในรูปแบบเว็บเพจ

การทดสอบระบบการนำเสนอข้อมูลในรูปแบบของเว็บเพจเป็นการทดสอบการนำเสนอผลการวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นในรูปแบบเว็บเพจ โดยนำข้อมูลจากฐานข้อมูลมาแสดงผล ในการแสดงผลใช้ Microsoft Visual Basic ดึงข้อมูลจากฐานข้อมูล และ Dream weaver CS5 ในการพัฒนาหน้าเว็บเพจ

จากการทดสอบระบบการแสดงข้อมูลในรูปแบบของเว็บเพจพบว่า สามารถแสดงข้อมูลได้อย่างถูกต้อง ดังแสดงในภาพที่ 4.8

ภาพที่ 4.8 ภาพหน้าจอแสดงการรายงานผลการวัดปริมาณรังสีแกมมาในรูปแบบเว็บเพจ

นอกจากนี้ผู้ใช้ยังสามารถเลือกแสดงผลการวัดปริมาณรังสีแกมมาย้อนหลังได้ โดยการใส่ข้อมูลวัน เดือน ปี ที่ต้องการทราบดังภาพที่ 4.9



ภาพที่ 4.9 ภาพหน้าจอแสดงการค้นหาข้อมูลย้อนหลัง

บทที่ 5

สรุปผลและเสนอแนะ

5.1 สรุปผลการวิจัย

ระบบเฝ้าระวังรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่นที่ ได้ทำการพัฒนาสถานีลูกข่ายฯ ให้สามารถเคลื่อนย้ายไปบริเวณที่ต้องการได้อย่างสะดวก มีความแม่นยำในการส่งข้อมูลให้ผู้ควบคุมระบบสามารถควบคุมการทำงานของสถานีลูกข่ายฯ ได้จากสถานีแม่ข่ายฯ และยังสามารถแสดงผลปริมาณรังสีแกมมาในรูปแบบเว็บเพจได้ อีกทั้งยังสามารถเรียกดูข้อมูลย้อนหลังจากฐานข้อมูลที่บันทึกไว้ได้ โดยผลการทดสอบการทำงานของระบบสรุปได้ดังนี้

5.1.1 การทดสอบระบบการรับส่งข้อมูลจากเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทลเลอร์

ก. จากการทดสอบการเชื่อมต่อการรับส่งข้อมูลระหว่างเครื่องสเกลเลอร์/เรตมิเตอร์กับไมโครคอนโทลเลอร์พบว่า เครื่องสเกลเลอร์/เรตมิเตอร์สามารถส่งข้อมูลการนับวัดรังสีแกมมาที่ทำการตรวจวัดผ่านพอร์ตอนุกรมได้อย่างถูกต้อง

ข. จากการทดสอบการเชื่อมต่อระหว่างไมโครคอนโทลเลอร์กับคอมพิวเตอร์ผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น พบว่า ไมโครคอนโทลเลอร์สามารถส่งผลการวัดรังสีจากระบบวัดรังสีแกมมาได้อย่างถูกต้อง

5.1.2 การทดสอบการรับส่งข้อมูลที่ได้จากการวัดรังสีแกมมาโดยส่งข้อมูลผ่านระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่ายฯ

จากการทดสอบการรับส่งข้อมูลที่ได้จากการวัดรังสีแกมมาโดยส่งข้อมูลผ่านระบบเครือข่ายคอมพิวเตอร์ท้องถิ่นเข้าสู่สถานีแม่ข่ายฯ พบว่า สถานีลูกข่ายฯ สามารถส่งข้อมูลการวัดรังสีแกมมาไปแสดงผลบนหน้าจอของไมโครคอมพิวเตอร์ได้ถูกต้อง

5.1.3 การสอบเทียบมาตรฐานระบบวัดรังสีแกมมา

ก. ความสัมพันธ์ระหว่างอัตราปริมาณรังสีที่คำนวณจากต้นกำเนิดรังสีมาตรฐาน (mR/hr) กับอัตราปริมาณรังสีที่วัดได้ (count/min) มีความสัมพันธ์ที่เป็นเส้นตรง y=0.0009x-0.1959

ข. การทดสอบความถูกต้องของระบบวัดปริมาณรังสีแกมมาผ่านเครือข่ายคอมพิวเตอร์ท้องถิ่น พบว่า ระบบสามารถวัดปริมาณรังสีแกมมาได้อย่างถูกต้อง โดยมีค่าความแตกต่างจากค่ามาตรฐานไม่เกิน ±10%

5.1.4 การทดสอบการนำเสนอข้อมูลในรูปแบบของเว็บเพจ

จากการทดสอบการนำเสนอข้อมูลในรูปแบบของเว็บเพจ พบว่าระบบสามรถนำข้อมูลปริมาณรังสีแกมมา แสดงบนหน้าเว็บเพจได้ และยังสามารถเรียกดูข้อมูลปริมาณรังสีย้อนหลังในแต่ละช่วงเวลาได้

5.1.5 ข้อเสนอแนะ

1. ควรมีระบบไฟสำรองสำหรับระบบฯ
2. ในอนาคตควรมีการพัฒนาให้ระบบสามารถใช้งานในระบบอินเตอร์เน็ตได้
3. ในกรณีที่ต้องการติดตั้งระบบนอกพื้นที่ ที่ระบบเครือข่ายไม่สามารถเข้าได้ ควรมีระบบ อื่นรองรับด้วย เช่น การติดต่อผ่านทางโทรศัพท์ หรือ GPRS

รายการอ้างอิง

[1] ธนากร อรัญศิริ. การพัฒนาระบบเชื่อมโยงสัญญาณสำหรับเฝ้าระวังทางรังสีโดยใช้เครือข่ายวิทยุสื่อสาร. วิทยานิพนธ์ปริญญามหาบัณฑิต, สาขาวิชานิวเคลียร์เทคโนโลยี ภาควิชานิวเคลียร์เทคโนโลยี คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2539.

[2] กิตติศักดิ์ ชัยสรรค์. การพัฒนาระบบเฝ้าระวังรังสีแกมมาในสิ่งแวดล้อมผ่านเครือข่ายโทรศัพท์เคลื่อนที่. วิทยานิพนธ์ปริญญามหาบัณฑิต, สาขาวิชานิวเคลียร์เทคโนโลยี ภาควิชานิวเคลียร์เทคโนโลยี คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2549.

[3] Ch. Wedekind,G. Schilling, M. Gruttmuller, K. Becker. Gamma-radiation monitoring network at sea, Applied Radiation and Isotopes, Germany, 1998.

[4] สุวิทย์ ปุณณชัยยะ, เอกสารการสอนวิชา 2111-606 การตรวจวัดรังสีนิวเคลียร์และอุปกรณ์วัดนิวเคลียร์ ภาควิชานิวเคลียร์เทคโนโลยี คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

[5] จิราภรณ์ โตเจริญชัย, ภาวนา ภู่สุวรรณ เทคโนโลยีทางเวชศาสตร์นิวเคลียร์ ฉบับแก้ไขปรับปรุง : พิมพ์ครั้งที่ 2, กรุงเทพ : พี.เอ.ลีฟวิ่ง, 2545.

[6] ANDREW S.TANENBAUM, ยุทธ์ สว่างวรรณ (แปล), เครือข่ายคอมพิวเตอร์. เพียร์สัน เอ็ดดูเคชัน อินโดไชน่า, 2547

[7] เรืองไกร รังสิผล. เจาะระบบ TCP/IP. กรุงเทพฯ : โปรวิชั่น, 2544.

[8] สำนักงานปรมาณูเพื่อสันติ กระทรวงวิทยาศาสตร์และเทคโนโลยี. ระบบวัดระดับรังสีแกมมาในอากาศ [ออนไลน์]. แหล่งที่มา : http://www.oaep.go.th
[10 พฤษภาคม 2555]

[9] Environmental radiation monitoring around Tokai reprocessing plant. Environmental protection section, health and safety division, Tokai works, power reactor and Nuclear fuel Development Corporation.

[10] Lamarsh, Jonh R. Introduction to nuclear engineering. 2nd ed. California : Addsion-Wesley, publishing, 1983.

ภาคผนวก

ภาคผนวก ก.

หัววัดรังสีไกเกอร์มูลเลอร์ ของบริษัท Ludlum **รุ่น** 44-7

INDICATED USE: Alpha beta gamma survey; Sample counting
DETECTOR: End window halogen quenched G-M
WINDOW: 1.7 ± 0.3 mg/cm² mica
WINDOW AREA:
Active - 6 cm²
Open - 5 cm²
EFFICIENCY(4pi geometry) : Typically 2%-^{14}C; 10%-^{90}Sr/^{90}Y; 7%-^{239}Pu
SENSITIVITY : Typically 2100 cpm/mR/hr (*^{137}Cs gamma*)
ENERGY RESPONSE: Energy dependant
DEAD TIME : Typically 200 microseconds
COMPATIBLE INSTRUMENTS : General purpose survey meters, ratemeters, and scalers
OPERATING VOLTAGE : 900 volts
CONNECTOR: Series "C" (*others available*)
CONSTRUCTION: Anodized aluminum housing
TEMPERATURE RANGE: -4° F(-20° C) to 122° F(50° C)
May be certified for operation from -40° F(-40° C) to 150° F(65° C)
SIZE: 1.8" (4.6 cm) Diameter X 5.8" (14.7 cm)L
WEIGHT: 1 lb (0.5kg)

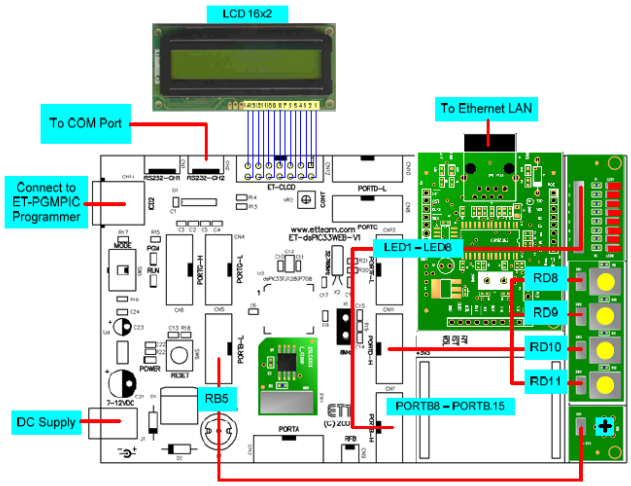
ภาคผนวก ข.

MODEL 2200 Scaler/Ratemeter, SCA

INDICATED USE: Single channel analyzing, gross sample counting
COMPATIBLE DETECTORS: G-M, proportional, scintillation
CONNECTOR: Series “C” (*others available*)
SCALER: 6 digit LED display with dimmer control providing a range of 0 - 999999 counts (*controlled by COUNT and HOLD buttons*)
SCALER LINEARITY: Reading within ±2% of true value
TIMER: Thumb wheel adjustment from 0 - 99 minutes with selectable divisions of X0.1, X1, X10, or EXT for manual timing
RATEMETER: 0 - 500,000 cpm total range
METER DIAL: 0 - 500 cpm, 0 - 2.5 kV, BAT TEST
MULTIPLIERS: X1, X10, X100, X1000
LINEARITY: Reading within ±10% of true value
RESPONSE: Toggle switch for FAST (2.2 seconds), or SLOW (22 seconds) from 10% to 90% of final reading
ZERO: Pushbutton to zero meter
HIGH VOLTAGE: Adjustable from 200 - 2500 volts (*will support 60 megohm scintillation loads*)
THRESHOLD: Voltage sensitive, adjustable from 1.00 - 10.00
WINDOW: Adjustable from 0 to 10.0 (*can be enabled or disabled*)
DISCRIMINATOR: Adjustable from 2 - 100 mV at threshold setting of 1.00
DATA OUT: 15 pin connector allowing for recorder, printer, or software interface.
METER: 2.5" (6.4 cm) arc, 1 mA movement analog type
POWER: 95 - 135 VAC (*178 - 240 VAC available)*, 50-60 Hz single phase (*less than 100 mA*) or 4 ea. “D” cell batteries
CHARGER: Activated when control switch is in CHG position (*use with rechargeable batteries only*)
BATTERY LIFE: Typically 20 hours with alkaline batteries (*battery condition can be checked on meter*)
CONSTRUCTION: Aluminum housing with beige polyurethane enamel paint and anodized aluminum front panel
TEMPERATURE RANGE: 5°F (-15°C) to 122°F (50°C)
May be certified to operate from -40°F (-40°C) to 150°F (65°C)
SIZE: 8.5" (21.6 cm)H X 5" (12.7 cm)W X 9.3" (23.5 cm)D
WEIGHT: 8.6 lbs (3.9kg)
Part No.: 48-165

ภาคผนวก ค.

ไมโครคอนโทรลเลอร์ ET-dsPIC33WEB V1.0



รูปที่ แสดงการเชื่อมต่อสัญญาณต่างๆ เพื่อทำการทดลอง

ET-dsPIC33WEB V1.0 เป็นบอร์ดไมโครคอนโทรลเลอร์ในตระกูล dsPIC ของบริษัท Microchip โดยได้นำเอาไมโครคอนโทรลเลอร์ที่ประมวลผลข้อมูลแบบ 16 บิต เบอร์ dsPIC33FJ128GP708 มาพัฒนาเป็นบอร์ดใช้งาน ซึ่งคุณสมบัติเด่นของ dsPIC33FJ128GP708 คือ หน่วยประมวลผลสัญญาณดิจิตอล (Digital Signal Processing) และ ทรัพยากรต่างๆ ดังต่อไปนี้

หน่วยประมวลผล (CPU)

- ความเร็วในการประมวลผล 40 MIPS (16 Bit Data / 24 Bit Instruction Code)
- ฮาร์ดแวร์รองรับการคูณข้อมูล 16 x 16 บิต โดยใช้เวลาเพียง 1 ไซเคิลคำสั่ง
- ฮาร์ดแวร์รองรับการหารข้อมูล 32-bit x 16 บิต
- C Compiler ถูกออกแบบให้มีความกระชับ Optimized Instruction Set
- รองรับการ Interrupt มากถึง118 Vector Interrupt จาก 63 แหล่ง 7 Priority Level Program
- รองรับการ DMA กับ Peripheral Hardware ได้ 8 ช่อง พร้อม DMA Buffer 2Kbyte

ระบบ (System)

- แหล่งกำเนิดสัญญาณนาฬิกาสามารถเลือกได้ ทั้งจากภายในและภายนอก
- มีวงจร Power-Up Timer และ Oscillator Start-Up
- มีระบบตรวจสอบสัญญาณนาฬิกา (Fail-Safe Clock Monitor)
- ระบบ Watchdog Timer ที่ใช้แหล่งสัญญาณนาฬิกาแบบ RC oscillator ที่แยกจากส่วนอื่นๆ
- ทำงานที่แรงดันระดับ 3.0 ถึง 3.6 โวลต์
- I/O Pin 4mA Sink สามารถเชื่อมต่อกับสัญญาณ 5VTTL ได้ (5V Tolerant)
- รองรับโหมดการทำงานแบบ Run, Idle และ Sleep modes
- สามารถปรับเปลี่ยนโหมดการทำงานของสัญญาณนาฬิกาได้หลากหลายเพื่อประสิทธิภาพ และ ให้สอดคล้องกับการดูแลจัดการในเรื่องของพลังงาน

คุณสมบัติทางด้านสัญญาณอนาลอก (Analog Features)

- โมดูลแปลงสัญญาณ Analog to Digital ความละเอียด 10-bit จำนวน 24 ช่อง และสามารถโปรแกรมเป็น 12Bit ได้ 2 ช่อง ความเร็วในการ Sampling สัญญาณสูงสุด 1.1 MSPS

คุณสมบัติโดยทั่วไปของ MCU เบอร์ dsPIC33FJ128GP708

- น่วยความจำโปรแกรมแบบ Flash Memory ขนาด 128 K Byte
- หน่วยความจำข้อมูล SRAM ขนาด 16 K Byte
- I/O Ports ใช้งานจำนวน 69 บิต(รวม Peripheral Function ต่างๆ)
- โมดูลการสื่อสาร UART จำนวน 2 ช่อง
- โมดูลการสื่อสารแบบ SPI จำนวน 2 ช่องรองรับทั้ง Master และ Slave Modes
- โมดูลการสื่อสารแบบ I2C จำนวน 2 ช่องรองรับทั้ง Master และ Slave Modes
- โมดูลการสื่อสารแบบ CAN จำนวน 2 ช่อง
- โมดูล Timer ขนาด 16-bit จำนวน 9 ช่อง และสามารถจับคู่ใช้งานเป็น Timer ขนาด 32 Bit ได้พร้อมกันจำนวน 4 ช่อง
- โมดูล Capture , Compare / PWM จำนวน 8 ชุด
- ระบบฮาร์ดแวร์ RTCC, Real-Time Clock Calendar with Alarms ภายใน

- โมดูล ADC ขนาด 10Bit จำนวน 24 ช่อง และสามารถโปรแกรมค่าเป็น 12Bit ได้ 2 ช่อง
- ระบบการสื่อสารแบบขนาน DCI(Data Converter Interface) จำนวน 1 ช่อง

คุณสมบัติโดยทั่วไปของบอร์ด ET-dsPIC33WEB V1.0

- ใช้ไมโครคอนโทรเลอร์ dsPIC33FJ128GP708 ขนาด 80 PIN
- สัญญาณนาฬิกาคริสตอลออสซิลเลเตอร์ขนาด 8 MHz (สามารถใช้ PLL รันได้ถึง 40 MHz)
- สัญญาณนาฬิกาคริสตอลออสซิลเลเตอร์ขนาด 32.768KHz สำหรับ RTC
- I/O Port ขนาด 10 PIN (จัดเรียงตามมาตรฐานของ อีทีที) จำนวน 9 พอร์ต
- ชุดวงจร Line Driver RS232 จำนวน 2 พอร์ต
- พอร์ตสำหรับต่อ LCD เรียงตามมาตรฐานของ อีทีที (14Pin ET-CLCD) จำนวน 1 พอร์ต
- ขั้วต่อสัญญาณดาวน์โหลดโปรแกรมแบบ ICD2 และ สวิตช์ตัดต่อสัญญาณ Run / Program
- วงจร LED สำหรับใช้ทดลองเอาต์พุตแบบ Digital จำนวน 8 ช่อง
- วงจรสวิตช์ Push-Button สำหรับใช้ทดลองอินพุตแบบ Digital จำนวน 4 ช่อง
- วงจรสร้างแรงดัน 0-3.3V จากตัวต้านทานปรับค่าได้ สำหรับทดลองโมดูล A/D จำนวน 1 ช่อง
- พอร์ตสำหรับเชื่อมต่อกับโมดูล Ethernet รุ่น ET-MINI ENC28J60(ใช้ SPI1)
- พอร์ตเชื่อมต่อกับหน่วยความจำ EEPROM 25LCxxx จำนวน 1 ช่อง(ใช้ SPI2)
- ชุด Regulate แบบ Switching สำหรับแปลงไฟ DC Input ให้เป็น 5V และ 3.3 V
- LED สถานะสำหรับ Power(แดง),Program(แดง) และ Run(เขียว)
  ขั้วต่อแรงดันไฟ VCC และ GND ใช้ได้กับไฟ 7-12 VDC

ภาคผนวก ง.

โปรแกรมควบคุมการทงานของสถานีลูกข่าย

(Main.c)

```
#define THIS_IS_STACK_APPLICATION
#include "TCPIP Stack/TCPIP.h"
#include "MainDemo.h"
#include "usart1.h"
#include "Ratemeter.h"
APP_CONFIG AppConfig;
static unsigned short wOriginalAppConfigChecksum;    // Checksum of the ROM defaults for AppConfig
BYTE AN0String[8];
int __C30_UART = 2;
static void InitAppConfig(void);
static void InitializeBoard(void);
//static void ProcessIO(void);
/********************************************************************************************/
int main(void)
{
            //static DWORD t = 0;
            static DWORD dwLastIP = 0;
  //static char  letter;
            InitializeBoard();
  _init_uart1(SelectBuadrate(2400));
  _uart1_puts("test USART1\n\r");
            #if defined(USE_LCD)
            init_lcd();
            lcd_gotoxy(0,0);
            DelayMs(100);
            strcpypgm2ram((char*)LCDText, "Scaler Ratemeter  "
              "              ");
            LCDUpdate();
            #endif
  TickInit();
            InitAppConfig();
  StackInit();
  DisplayIPValue(AppConfig.MyIPAddr);     // Print to UART
  {
    StackTask();
    StackApplications();
                        if(dwLastIP != AppConfig.MyIPAddr.Val)
```

```
                {
                        #if defined(STACK_USE_UART)
                                _uart1_puts((ROM char*)"\r\n ** Re new IP Address ** \r\n");
                        #endif
                        dwLastIP = AppConfig.MyIPAddr.Val;
            DisplayIPValue(AppConfig.MyIPAddr);
                }
        RateMeterTask();
            }
}
void DisplayIPValue(IP_ADDR IPVal)
{
   BYTE IPDigit[4];
            BYTE i;
#ifdef USE_LCD
            BYTE j;
            BYTE LCDPos=16;
#endif
            for(i = 0; i < sizeof(IP_ADDR); i++)            {
              uitoa((WORD)IPVal.v[i], IPDigit);
                #ifdef USE_LCD
                        for(j = 0; j < strlen((char*)IPDigit); j++)
                        {
                                LCDText[LCDPos++] = IPDigit[j];
                        }
                        if(i == sizeof(IP_ADDR)-1)
                                break;
                        LCDText[LCDPos++] = '.';
                #else
                        if(i == sizeof(IP_ADDR)-1)
                                break;
                #endif
            }
            #if defined(STACK_USE_UART)
        _uart1_puts((ROM char*)" - ");
                _uart1_puts((char *)LCDText);
        _uart1_puts((ROM char*)"\r\n");
            #endif
            #ifdef USE_LCD
                if(LCDPos < 32u)
                        LCDText[LCDPos] = 0;
                LCDUpdate();
            #endif
```

```
}
//static void ProcessIO(void)
//{
//}
static void InitializeBoard(void)
{
        PLLFBD=0x26;            // PLL = 40 (M=38 + default 2)   40MIPS
        CLKDIVbits.PLLPOST=0;           // N1=0 + default 2
        CLKDIVbits.PLLPRE=0;            // N2=0 + default 2
        //OSCTUN=0;                                     // Tune FRC oscillator, if FRC is used
    //Disable Watch Dog Timer
        RCONbits.SWDTEN=0;
    //Clock switching to incorporate PLL
        __builtin_write_OSCCONH(0x03);          // Initiate Clock Switch to Primary
                                                // Oscillator with PLL (NOSC=0b011)
        __builtin_write_OSCCONL(0x01);          // Start clock switching
        while (OSCCONbits.COSC != 0b011);       // Wait for Clock switch to occur
    // Wait for PLL to lock
        while(OSCCONbits.LOCK!=1) {};
        // Port I/O
        AD1PCFGHbits.PCFG23 = 1;    // Make RA7 (BUTTON1) a digital input
        AD1PCFGHbits.PCFG20 = 1;    // Make RA12 (INT1) a digital input for MRF24WB0M PICtail Plus interrupt
        // ADC
    AD1CHS0 = 0;                                // Input to AN0 (potentiometer)
        AD1PCFGLbits.PCFG5 = 0;                 // Disable digital input on AN5 (potentiometer)
        AD1PCFGLbits.PCFG4 = 0;                 // Disable digital input on AN4 (TC1047A temp sensor)
        // ADC
        AD1CON1 = 0x84E4;                       // Turn on, auto sample start, auto-convert, 12 bit mode (on parts with a
12bit A/D)
        AD1CON2 = 0x0404;                       // AVdd, AVss, int every 2 conversions, MUXA only, scan
        AD1CON3 = 0x1003;                       // 16 Tad auto-sample, Tad = 3*Tcy
        AD1CSSL = 1<<5;                                 // Scan pot
    #if defined(ENC_CS_TRIS)
            ENC_CS_IO = 1;
            ENC_CS_TRIS = 0;
    #endif
    #if defined(EEPROM_CS_TRIS)
            EEPROM_CS_IO = 1;
            EEPROM_CS_TRIS = 0;
    #endif
    #if defined(EEPROM_CS_TRIS)
            XEEInit();
    #endif
```

```
}
static ROM BYTE SerializedMACAddress[6] = {MY_DEFAULT_MAC_BYTE1, MY_DEFAULT_MAC_BYTE2,
MY_DEFAULT_MAC_BYTE3, MY_DEFAULT_MAC_BYTE4, MY_DEFAULT_MAC_BYTE5, MY_DEFAULT_MAC_BYTE6};
static void InitAppConfig(void)
{
        // Start out zeroing all AppConfig bytes to ensure all fields are  deterministic for checksum generation
        memset((void*)&AppConfig, 0x00, sizeof(AppConfig));
        AppConfig.Flags.bIsDHCPEnabled = TRUE;
        AppConfig.Flags.bInConfigMode  = TRUE;
        memcpypgm2ram((void*)&AppConfig.MyMACAddr, (ROM void*)SerializedMACAddress,
sizeof(AppConfig.MyMACAddr));
        AppConfig.MyIPAddr.Val          = MY_DEFAULT_IP_ADDR_BYTE1 | MY_DEFAULT_IP_ADDR_BYTE2<<8ul |
MY_DEFAULT_IP_ADDR_BYTE3<<16ul | MY_DEFAULT_IP_ADDR_BYTE4<<24ul;
        AppConfig.DefaultIPAddr.Val     = AppConfig.MyIPAddr.Val;
        AppConfig.MyMask.Val            = MY_DEFAULT_MASK_BYTE1 | MY_DEFAULT_MASK_BYTE2<<8ul |
MY_DEFAULT_MASK_BYTE3<<16ul | MY_DEFAULT_MASK_BYTE4<<24ul;
        AppConfig.DefaultMask.Val       = AppConfig.MyMask.Val;
        AppConfig.MyGateway.Val         = MY_DEFAULT_GATE_BYTE1 | MY_DEFAULT_GATE_BYTE2<<8ul |
MY_DEFAULT_GATE_BYTE3<<16ul | MY_DEFAULT_GATE_BYTE4<<24ul;
        AppConfig.PrimaryDNSServer.Val  = MY_DEFAULT_PRIMARY_DNS_BYTE1 |
MY_DEFAULT_PRIMARY_DNS_BYTE2<<8ul  | MY_DEFAULT_PRIMARY_DNS_BYTE3<<16ul  |
MY_DEFAULT_PRIMARY_DNS_BYTE4<<24ul;
        AppConfig.SecondaryDNSServer.Val = MY_DEFAULT_SECONDARY_DNS_BYTE1 |
MY_DEFAULT_SECONDARY_DNS_BYTE2<<8ul  | MY_DEFAULT_SECONDARY_DNS_BYTE3<<16ul  |
MY_DEFAULT_SECONDARY_DNS_BYTE4<<24ul;
        wOriginalAppConfigChecksum = CalcIPChecksum((BYTE*)&AppConfig, sizeof(AppConfig));
}
```

(Retmeter.c)

```
#ifndef __RATEMETER_C__
#define __RATEMETER_C__
#include "TCPIPConfig.h"
#include "TCPIP Stack/TCPIP.h"
#include "Ratemeter.h"
#include "usart1.h"
#define  SS_COMMAND   "SS"
#define  SO_COMMAND   "SO"
#define  SC_COMMAND   "SC"
#define  RS_COMMAND   "RS"
#define  RL_COMMAND   "RL"
#define  RM_COMMAND   "RM"
#define  RC_COMMAND   "RC"
#define  TCP_CLOSE    "CC"
#define RATEMETER_PORT                                  9760
static WORD w,w1,z;
static TCP_SOCKET    MySocket;
static enum _TCPServerState
{
        SM_HOME = 0,
        SM_LISTENING = 1,
        SM_RS = 2,
        SM_RL = 3,
        SM_RM = 4,
        SM_RC = 5,
        SM_CC = 6,
        SM_SO = 7,
        SM_SC = 8,
        SM_SS = 9
} TCPServerState = SM_HOME;
static BYTE     UART1_FREE = 0;
static enum _UART1_State
{
        UART1_HOME      = 0,
        UART1_LISTENING = 1,
        UART1_COMPLETE  = 2
} UART1_STATE = UART1_HOME;
static enum _CAPTURE_State
{
        CAPTURE_HOME      = 0,
        CAPTURE_START     = 1,
        CAPTURE_COMPLETE  = 2
```

```
} CAPTURE_STATE = CAPTURE_HOME;
volatile BYTE  letter,i;
volatile BYTE  uartTemp[256];
volatile BYTE  uartCount;
volatile BYTE  * uartIndex;
void SS_command(void);
void SO_command(void);
void SC_command(void);
void RS_command(void);
void RL_command(void);
void RM_command(void);
void RC_command(void);
void SS_Listening(void);
void RateMeterTask(void)
{
        if((TCPServerState == SM_LISTENING) && (UART1_FREE == 1))
        {
                //if(_uart1_ready()){
                        SS_Listening();
                //}
        }
        switch(TCPServerState){
                case SM_HOME:
                        if(!TCPIsConnected(MySocket)){
                MySocket = TCPOpen(0, TCP_OPEN_SERVER,RATEMETER_PORT, TCP_PURPOSE_DEFAULT);
                                if(MySocket == INVALID_SOCKET){
           TCPServerState = SM_HOME;
        }else{
                                                TCPServerState = SM_LISTENING;
                                }
                                UART1_FREE = 0       ;
                        }else{
                                TCPServerState = SM_LISTENING;
                          UART1_FREE = 0;
                        }
                        break
                case SM_LISTENING:
                        z = TCPIsGetReady(MySocket);
                        if(z){
        w = TCPFind(MySocket, 0x0A, 0, FALSE);
                          if(w != 0xFFFFu)
                                {
```

```
                    w1 = TCPFindROMArray(MySocket, (ROM BYTE*)SO_COMMAND, sizeof(SO_COMMAND)-1,
0,TRUE);
                        if( ( w1 != 0u) ){
                {__asm__ volatile ("nop");}
                    }else{
                                    TCPGetArray(MySocket,(BYTE*)command,z);
                                    TCPDiscard(MySocket);
                                    TCPServerState = SM_SO;break;
                }

                    w1 = TCPFindROMArray(MySocket, (ROM BYTE*)RS_COMMAND, sizeof(RS_COMMAND)-1,
0,TRUE);
                        if( ( w1 != 0u) ){
                {__asm__ volatile ("nop");}
                    }else{
                                    TCPGetArray(MySocket,(BYTE*)command,z);
                                    TCPDiscard(MySocket);
                                    TCPServerState = SM_RS;break;
                }

                    w1 = TCPFindROMArray(MySocket, (ROM BYTE*)RL_COMMAND, sizeof(RL_COMMAND)-1,
0,TRUE);
                        if( ( w1 != 0u) ){
                {__asm__ volatile ("nop");}
                    }else{
                                    TCPGetArray(MySocket,(BYTE*)command,z);
                                    TCPDiscard(MySocket);
                                    TCPServerState = SM_RL;break;
                }

                    w1 = TCPFindROMArray(MySocket, (ROM BYTE*)RM_COMMAND, sizeof(RM_COMMAND)-1,
0,TRUE);
                        if( ( w1 != 0u) ){
                {__asm__ volatile ("nop");}
                    }else{
                                    TCPGetArray(MySocket,(BYTE*)command,z);
                                    TCPDiscard(MySocket);
                                    TCPServerState = SM_RM;break;
                }

                    w1 = TCPFindROMArray(MySocket, (ROM BYTE*)RC_COMMAND, sizeof(RC_COMMAND)-1,
0,TRUE);
                        if( ( w1 != 0u) ){
```

```
                {__asm__ volatile ("nop");}
                                }else{
                                                TCPGetArray(MySocket,(BYTE*)command,z);
                                                TCPDiscard(MySocket);
                                                TCPServerState = SM_RC;break;
                        }
                                        w1 = TCPFindROMArray(MySocket, (ROM BYTE*)TCP_CLOSE,
sizeof(TCP_CLOSE)-1, 0,TRUE);
                                        if( ( w1 != 0u) ){
                        {__asm__ volatile ("nop");}
                                        }else{
                                                TCPGetArray(MySocket,(BYTE*)command,z);
                                                TCPDiscard(MySocket);
                                                TCPServerState = SM_CC;break;
                }
                                        w1 = TCPFindROMArray(MySocket, (ROM BYTE*)SS_COMMAND,
sizeof(SS_COMMAND)-1, 0,TRUE);
                                        if( ( w1 != 0u) ){
                        {__asm__ volatile ("nop");}
                                        }else{
                                                TCPGetArray(MySocket,(BYTE*)command,z-1);
                    TCPDiscard(MySocket);
                                                TCPServerState = SM_SS;break;
                }
                                        w1 = TCPFindROMArray(MySocket, (ROM BYTE*)SC_COMMAND,
sizeof(SC_COMMAND)-1, 0,TRUE);
                                        if( ( w1 != 0u) ){
                        {__asm__ volatile ("nop");}
                                        }else{
                                                TCPGetArray(MySocket,(BYTE*)command,z);
                                                TCPServerState = SM_SC;break;
                }
                                        //TCPGetArray(MySocket,(BYTE*)agrument,z-1);
                //TCPPutArray(MySocket,(BYTE*)agrument,z-1);
                                        TCPDiscard(MySocket);
                                                //TCPFlush(MySocket);
                TCPServerState = SM_LISTENING;
                }
                                }
                                break;
        case SM_SO:
                                        if( UART1_FREE == 0){
                                                SO_command();
```

```
                    }else{
                        TCPServerState = SM_LISTENING;
                    }
    break;
case SM_SC:
                    if( UART1_FREE == 0){
                        SC_command();
                    }else{
                        TCPDiscard(MySocket);
                        TCPServerState = SM_LISTENING;
                    }
    break;
case SM_RS:
                    if( UART1_FREE == 0){
                        RS_command();
                    }else{
                        TCPServerState = SM_LISTENING;
                    }
    break;
case SM_RL:
                    if( UART1_FREE == 0){
                        RL_command();
                    }else{
                        TCPServerState = SM_LISTENING;
                    }
    break;
case SM_RM:
                    if( UART1_FREE == 0){
                        RM_command();
                    }else{
                        TCPServerState = SM_LISTENING;
                    }
    break;
case SM_RC:
                    if( UART1_FREE == 0){
                        RC_command();
                    }else{
                        TCPServerState = SM_LISTENING;
                    }
    break;
case SM_CC:
                if( UART1_FREE == 0){
                    TCPPutROMString(MySocket, (ROM BYTE*)"\n\r Close TCP >> ");
```

```
            TCPPutString(MySocket,(BYTE*)command);
                                        TCPFlush(MySocket);
                                        TCPClose(MySocket);
            TCPServerState = SM_HOME;
                                    }else{
                                        TCPServerState = SM_LISTENING;
                                    }
        break;
      case SM_SS:
                                        //SS_command();
        _uart1_puts("S");
        _uart1_puts("S");
        _uart1_putc(0x0A);
                                    if(UART1_FREE==1)
                                    {
                                        UART1_STATE = UART1_HOME;
                                        UART1_FREE = 0;
                                        TCPServerState = SM_LISTENING;
                                    }else{
                        if(_uart1_ready()){
                        do{
                        letter = _uart1_getc();
                                }while( _uart1_ready());
                                        }

                                        UART1_STATE = UART1_HOME;
                                        UART1_FREE = 1;
          TCPServerState = SM_HOME;
                                    }
        break;
                        default : break;
            }
}
void SS_command(void)
{
  switch(UART1_STATE){
      case UART1_HOME :
            if(_uart1_ready()){
              do{
                letter = _uart1_getc();
             }while( _uart1_ready());
            }
        _uart1_puts("S");
```

```
    _uart1_puts("S");
    _uart1_putc(0x0A);
    UART1_STATE = UART1_LISTENING;
    uartCount = 0;
    break;
case UART1_LISTENING :
                        switch(CAPTURE_STATE){
                            case CAPTURE_HOME :
                                if(_uart1_ready()){
                                        uartIndex = uartTemp;
            letter = _uart1_getc();
                                    *uartIndex = letter;
                                        uartIndex++;
            uartCount++;
                                        CAPTURE_STATE = CAPTURE_START;
                                    }else{
                                        break;
                                    }
                            case CAPTURE_START :
                                if(_uart1_ready()){
                        letter = _uart1_getc();
                                      *uartIndex = letter;
                                      uartIndex++;
                                      uartCount++;
                                        if(letter==0x0A){
                                            CAPTURE_STATE = CAPTURE_COMPLETE;
                                        }else{
                                            CAPTURE_STATE = CAPTURE_START;
                                            break;
                                        }
                                }else{
                                        break;
                                    }
                            case CAPTURE_COMPLETE :
                                    uartIndex = uartTemp;
        uartCount = 0;
        TCPPutArray(MySocket,(BYTE*) uartTemp, uartCount-1);
        TCPPutROMString(MySocket, (ROM BYTE*)"\n");
                                    TCPFlush(MySocket);
         CAPTURE_STATE = CAPTURE_HOME;
                                    UART1_STATE = UART1_COMPLETE;
                                    break;
                        }
```

```
            break;
        case UART1_COMPLETE :
            UART1_STATE = UART1_HOME;
                TCPServerState = SM_LISTENING;
            break;
    }
}
void SC_command(void)
{
    switch(UART1_STATE){
        case UART1_HOME :
                                        z = TCPIsGetReady(MySocket);
                                        if(z){
                                        w = TCPFind(MySocket, 0x0A, 0, FALSE);
                                                if(w != 0xFFFFu){
                                                TCPGetArray(MySocket,(BYTE*)agrument,z-1);
                                                }
                                        TCPDiscard(MySocket);
                                        }
                if(_uart1_ready()){
                    do{
                        letter = _uart1_getc();
                    }while( _uart1_ready());
                }
            _uart1_puts(SC_COMMAND);
            _uart1_puts((char *)agrument);
            _uart1_putc(0x0A);

            UART1_STATE = UART1_LISTENING;
            uartCount = 0;
            break;
        case UART1_LISTENING :
                                         switch(CAPTURE_STATE){
                                                case CAPTURE_HOME :
                                                    if(_uart1_ready()){
                                                                uartIndex = uartTemp;
                        letter = _uart1_getc();
                                                            *uartIndex = letter;
                                                                uartIndex++;
                        uartCount++;
                                                                CAPTURE_STATE = CAPTURE_START;
                                                            }else{
                                                                break;
```

```
                                                    }
                                                case CAPTURE_START :
                                                    if(_uart1_ready()){
                                                letter = _uart1_getc();
                                                     *uartIndex = letter;
                                                     uartIndex++;
                                                     uartCount++;
                                                          if(letter==0x0A){
                                                                CAPTURE_STATE = CAPTURE_COMPLETE;
                                                          }else{
                                                                CAPTURE_STATE = CAPTURE_START;
                                                                break;
                                                          }
                                                    }else{
                                                          break;
                                                    }

                                                case CAPTURE_COMPLETE :
                                                          uartIndex = uartTemp;
                    uartCount = 0;
                    TCPPutArray(MySocket,(BYTE*) uartTemp, uartCount-1);
                    TCPPutROMString(MySocket, (ROM BYTE*)"\n");
                                                     TCPFlush(MySocket);
                     CAPTURE_STATE = CAPTURE_HOME;
                                                          UART1_STATE = UART1_COMPLETE;
                                                          break;
                                              }
            break;
        case UART1_COMPLETE :
            UART1_STATE = UART1_HOME;
                TCPServerState = SM_LISTENING;
            break;
    }
}
void SO_command(void)
{
    switch(UART1_STATE){
        case UART1_HOME :
                if(_uart1_ready()){
                   do{
                      letter = _uart1_getc();
                   }while( _uart1_ready());
                }
```

```
    // _uart1_puts(SO_COMMAND);
    _uart1_puts("S");
    _uart1_puts("O");
    _uart1_putc(0x0A);
    UART1_STATE = UART1_LISTENING;
    uartCount = 0;
    break;
case UART1_LISTENING :
                        switch(CAPTURE_STATE){
                            case CAPTURE_HOME :
                                if(_uart1_ready()){
                                        uartIndex = uartTemp;
            letter = _uart1_getc();
                                    *uartIndex = letter;
                                        uartIndex++;
            uartCount++;
                                        CAPTURE_STATE = CAPTURE_START;
                                    }else{
                                        break;
                                    }
                            case CAPTURE_START :
                                if(_uart1_ready()){
                        letter = _uart1_getc();
                                      *uartIndex = letter;
                                      uartIndex++;
                                      uartCount++;
                                        if(letter==0x0A){
                                            CAPTURE_STATE = CAPTURE_COMPLETE;
                                        }else{
                                            CAPTURE_STATE = CAPTURE_START;
                                            break;
                                        }
                                }else{
                                        break;
                                    }
                            case CAPTURE_COMPLETE :
                                    uartIndex = uartTemp;
        uartCount = 0;
        TCPPutArray(MySocket,(BYTE*) uartTemp, uartCount-1);
        TCPPutROMString(MySocket, (ROM BYTE*)"\n");
                                TCPFlush(MySocket);
          CAPTURE_STATE = CAPTURE_HOME;
                                    UART1_STATE = UART1_COMPLETE;
```

```
                                                    break;
                                            }
            break;

        case UART1_COMPLETE :
            UART1_STATE = UART1_HOME;
                TCPServerState = SM_LISTENING;
            break;
    }
}
void RS_command(void)
{
    switch(UART1_STATE){
        case UART1_HOME :
                if(_uart1_ready()){
                    do{
                        letter = _uart1_getc();
                    }while( _uart1_ready());
                }
            // _uart1_puts(SO_COMMAND);
            _uart1_puts("R");
            _uart1_puts("S");
            _uart1_putc(0x0A);
            UART1_STATE = UART1_LISTENING;
            uartCount = 0;
            break;
        case UART1_LISTENING :
                                        switch(CAPTURE_STATE){
                                                case CAPTURE_HOME :
                                                    if(_uart1_ready()){
                                                                uartIndex = uartTemp;
                        letter = _uart1_getc();
                                                            *uartIndex = letter;
                                                                uartIndex++;
                        uartCount++;
                                                                CAPTURE_STATE = CAPTURE_START;
                                                            }else{
                                                                break;
                                                            }
                                                case CAPTURE_START :
                                                    if(_uart1_ready()){
                                            letter = _uart1_getc();
                                                                *uartIndex = letter;
```

```
                                                uartIndex++;
                                                uartCount++;
                                                        if(letter==0x0A){
                                                                CAPTURE_STATE = CAPTURE_COMPLETE;
                                                        }else{
                                                                CAPTURE_STATE = CAPTURE_START;
                                                                break;
                                                        }
                                                }else{
                                                        break;
                                                }

                                        case CAPTURE_COMPLETE :
                                                uartIndex = uartTemp;
                uartCount = 0;
                TCPPutArray(MySocket,(BYTE*) uartTemp, uartCount-1);
                TCPPutROMString(MySocket, (ROM BYTE*)"\n");
                                                TCPFlush(MySocket);
                 CAPTURE_STATE = CAPTURE_HOME;
                                                UART1_STATE = UART1_COMPLETE;
                                                break;
                                        }
            break;
        case UART1_COMPLETE :
            UART1_STATE = UART1_HOME;
                TCPServerState = SM_LISTENING;
            break;
    }
}
void RL_command(void)
{
    switch(UART1_STATE){
        case UART1_HOME :
                if(_uart1_ready()){
                    do{
                        letter = _uart1_getc();
                    }while( _uart1_ready());
                }
            _uart1_puts("R");
            _uart1_puts("L");
            _uart1_putc(0x0A);
            UART1_STATE = UART1_LISTENING;
            uartCount = 0;
```

```
    break;
case UART1_LISTENING :
                                switch(CAPTURE_STATE){
                                        case CAPTURE_HOME :
                                            if(_uart1_ready()){
                                                        uartIndex = uartTemp;
                    letter = _uart1_getc();
                                                    *uartIndex = letter;
                                                        uartIndex++;
                    uartCount++;
                                                        CAPTURE_STATE = CAPTURE_START;
                                                    }else{
                                                        break;
                                                    }
                                        case CAPTURE_START :
                                            if(_uart1_ready()){
                                  letter = _uart1_getc();
                                                     *uartIndex = letter;
                                                     uartIndex++;
                                                     uartCount++;
                                                        if(letter==0x0A){
                                                                CAPTURE_STATE = CAPTURE_COMPLETE;
                                                        }else{
                                                                CAPTURE_STATE = CAPTURE_START;
                                                                break;
                                                        }
                                            }else{
                                                        break;
                                                    }
                                        case CAPTURE_COMPLETE :
                                                    uartIndex = uartTemp;
        uartCount = 0;
        TCPPutArray(MySocket,(BYTE*) uartTemp, uartCount-1);
        TCPPutROMString(MySocket, (ROM BYTE*)"\n");
                                         TCPFlush(MySocket);
         CAPTURE_STATE = CAPTURE_HOME;
                                                    UART1_STATE = UART1_COMPLETE;
                                                    break;
                                }
    break;
case UART1_COMPLETE :
    UART1_STATE = UART1_HOME;
        TCPServerState = SM_LISTENING;
```

```
            break;
    }
}
void RM_command(void)
{
    switch(UART1_STATE){
        case UART1_HOME :
                if(_uart1_ready()){
                    do{
                        letter = _uart1_getc();
                    }while( _uart1_ready());
                }
            _uart1_puts("R");
            _uart1_puts("M");
            _uart1_putc(0x0A);
            UART1_STATE = UART1_LISTENING;
            uartCount = 0;
            break;
        case UART1_LISTENING :
                                switch(CAPTURE_STATE){
                                                case CAPTURE_HOME :
                                                    if(_uart1_ready()){
                                                                    uartIndex = uartTemp;
                                letter = _uart1_getc();
                                                            *uartIndex = letter;
                                                                    uartIndex++;
                                uartCount++;
                                                                    CAPTURE_STATE = CAPTURE_START;
                                                            }else{
                                                                    break;
                                                            }
                                                case CAPTURE_START :
                                                    if(_uart1_ready()){
                                            letter = _uart1_getc();
                                                                *uartIndex = letter;
                                                                uartIndex++;
                                                                uartCount++;
                                                                    if(letter==0x0A){
                                                                            CAPTURE_STATE = CAPTURE_COMPLETE;
                                                                    }else{
                                                                            CAPTURE_STATE = CAPTURE_START;
                                                                            break;
                                                                    }
```

```
                                                    }else{
                                                                        break;
                                                    case CAPTURE_COMPLETE :
                                                                        uartIndex = uartTemp;
                    uartCount = 0;
                    TCPPutArray(MySocket,(BYTE*) uartTemp, uartCount-1);
                    TCPPutROMString(MySocket, (ROM BYTE*)"\n");
                                                    TCPFlush(MySocket);
                     CAPTURE_STATE = CAPTURE_HOME;
                                                                        UART1_STATE = UART1_COMPLETE;
                                                                        break;
                                                }
                break;
            case UART1_COMPLETE :
                UART1_STATE = UART1_HOME;
                    TCPServerState = SM_LISTENING;
                break;
        }
}
void RC_command(void)
{
    switch(UART1_STATE){
        case UART1_HOME :
                if(_uart1_ready()){
                    do{
                        letter = _uart1_getc();
                    }while( _uart1_ready());
                }
            _uart1_puts("R");
            _uart1_puts("C");
            _uart1_putc(0x0A);
            UART1_STATE = UART1_LISTENING;
            uartCount = 0;
            break;
        case UART1_LISTENING :
                                        switch(CAPTURE_STATE){
                                                    case CAPTURE_HOME :
                                                        if(_uart1_ready()){
                                                                        uartIndex = uartTemp;
                            letter = _uart1_getc();
                                                    *uartIndex = letter;
                                                                        uartIndex++;
                            uartCount++;
```

```
                                                CAPTURE_STATE = CAPTURE_START;
                                            }else{
                                                break;
                                            }
                                        case CAPTURE_START :
                                            if(_uart1_ready()){
                                    letter = _uart1_getc();
                                                *uartIndex = letter;
                                                uartIndex++;
                                                uartCount++;
                                                if(letter==0x0A){
                                                    CAPTURE_STATE = CAPTURE_COMPLETE;
                                                }else{
                                                    CAPTURE_STATE = CAPTURE_START;
                                                    break;
                                                }
                                            }else{
                                                break;
                                            }
                                        case CAPTURE_COMPLETE :
                                                uartIndex = uartTemp;
                uartCount = 0;
                TCPPutArray(MySocket,(BYTE*) uartTemp, uartCount-1);
                TCPPutROMString(MySocket, (ROM BYTE*)"\n");
                                            TCPFlush(MySocket);
                 CAPTURE_STATE = CAPTURE_HOME;
                                                UART1_STATE = UART1_COMPLETE;
                                                break;
                                    }
            break;

        case UART1_COMPLETE :
            UART1_STATE = UART1_HOME;
                TCPServerState = SM_LISTENING;
            break;
    }
}
void SS_Listening(void)
{
  switch(UART1_STATE){
      case UART1_HOME :
              if(_uart1_ready()){
              uartCount = 0;
```

```
            }
        UART1_STATE = UART1_LISTENING;

    case UART1_LISTENING :
                                switch(CAPTURE_STATE){
                                    case CAPTURE_HOME :
                                        if(_uart1_ready()){
                                                uartIndex = uartTemp;
                    letter = _uart1_getc();
                                            *uartIndex = letter;
                                                uartIndex++;
                    uartCount++;
                                                CAPTURE_STATE = CAPTURE_START;
                                            }else{
                                                break;
                                            }
                                    case CAPTURE_START :
                                        if(_uart1_ready()){
                                  letter = _uart1_getc();
                                             *uartIndex = letter;
                                             uartIndex++;
                                             uartCount++;
                                                if(letter==0x0A){
                                                        CAPTURE_STATE = CAPTURE_COMPLETE;
                                                }else{
                                                        CAPTURE_STATE = CAPTURE_START;
                                                        break;
                                                }
                                        }else{
                                                break;
                                            }

                                    case CAPTURE_COMPLETE :
                                                uartIndex = uartTemp;
            uartCount = 0;
            TCPPutArray(MySocket,(BYTE*) uartTemp, uartCount-1);
            TCPPutROMString(MySocket, (ROM BYTE*)"\n");
                                         TCPFlush(MySocket);
             CAPTURE_STATE = CAPTURE_HOME;
                                                UART1_STATE = UART1_COMPLETE;
break;
                                }
        break;
```

```
        case UART1_COMPLETE :
            UART1_STATE = UART1_HOME;
                TCPServerState = SM_LISTENING;
                                        UART1_FREE = 0;
            break;
    }
}
#endif      //#if defined(STACK_USE_RATEMETER_SERVER)
```

ภาคผนวก จ.

โปรแกรมควบคุมการทำงานของสานีแม่ข่าย

From 1

```
Option Explicit
Global ConnMyDB As New ADODB.Connection
Global RS As New ADODB.Recordset
Global DS As New ADODB.Recordset
Global XS As New ADODB.Recordset
Global Statement As String
Global blnNewData As Boolean'
Public Sub OpenDataBase()
On Error GoTo Err_Handler
Dim DB_File As String
   DB_File = App.Path
   If Right$(DB_File, 1) <> "\" Then DB_File = DB_File & "\"
   DB_File = DB_File & "data.mdb"
   Set ConnMyDB = New ADODB.Connection
   ConnMyDB.ConnectionString = _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=" & DB_File & ";" & _
      "Persist Security Info=False"
   ConnMyDB.Open
   Exit Sub
Err_Handler:
   MsgBox "Error : " & Err.Number & " " & Err.Description
   End
End Sub
Public Sub CloseDataBase()
   If ConnMyDB.State = adStateOpen Then '
      ConnMyDB.Close
      Set ConnMyDB = Nothing
   End If
End Sub
```

From 2

```
Option Explicit
Dim buffer1 As String
Dim buffer2 As String
Dim buffer3 As String
Dim index1 As Integer
Dim index2 As Integer
Dim index3 As Integer
```

```
            Function TimeDelay(Delay As Double)
  Dim PauseTime, Start
  PauseTime = Delay
  Start = Timer
  Do While Timer < Start + PauseTime
    DoEvents
  Loop
End Function
Sub SetupScreen()
    txtStation.Text = ""
    txtDistance.Text = ""
    txtTime.Text = ""
    txtCycleTime.Text = ""
    cmbIP.Clear
End Sub
Sub RecordToScreen(IDd As String)
Set RS = New Recordset
    Statement = "select ID,IP,Station,Distance,Time,Cycle  from Status where ID=" & IDd
    RS.Open Statement, ConnMyDB, adOpenForwardOnly, , adCmdText
    txtStation.Text = "" & RS("Station")
    txtDistance.Text = "" & RS("Distance")
    txtTime.Text = "" & RS("Time")
    txtCycleTime.Text = "" & RS("Cycle")
    cmbIP.Text = RS("IP")
 'RS.Close
Set RS = Nothing
End Sub
Sub DisplayIP()
Dim Add$
    Set DS = New ADODB.Recordset
    Statement = "SELECT * FROM Status ORDER BY IP"
    Set DS = ConnMyDB.Execute(Statement, , adCmdText)
    cmbIP.Clear
    Do Until DS.EOF '
       Add$ = "" & DS("IP")
       cmbIP.AddItem Add$
       DS.MoveNext
    Loop
    DS.Close
    Set DS = Nothing
End Sub
Private Sub cmbIP_Click()
    Dim Index As String
```

```
    If cmbIP.Text = "10.80.72.199" Then
        Index = 1
    ElseIf cmbIP.Text = "10.80.72.200" Then
        Index = 2
    ElseIf cmbIP.Text = "10.80.72.201" Then
        Index = 3
    End If
    Call RecordToScreen(Index)
    Call updateFlexGrid(Index)
End Sub
Private Sub updateFlexGrid(IPnumber As String)
    Dim chrtArray()
    Set XS = New ADODB.Recordset
    If XS.State <> adStateClosed Then
        XS.Close
    End If
    XS.CursorLocation = adUseClient
        Statement = "SELECT IP, Date_Time, Count, CountCal, IndexIP FROM Count_Table where IndexIP=" & IPnumber
    'Set XS = ConnMyDB.Execute(Statement, , adCmdText)
    XS.Open Statement, ConnMyDB, adOpenDynamic, adLockOptimistic
    If XS.RecordCount = 0 Then
        MsgBox "No Data in Data-Base"
        myGrid.Clear
        Exit Sub
    End If
    Set myGrid.Recordset = XS
        myGrid.ColWidth(0) = 1200
        myGrid.ColWidth(1) = 2200
        myGrid.ColWidth(2) = 1000
        myGrid.ColWidth(3) = 1000
        myGrid.ColWidth(4) = 800
        myGrid.Refresh
    ReDim chrtArray(1 To XS.RecordCount, 1 To 2)
    MSChart1.ShowLegend = True
    MSChart1.chartType = VtChChartType2dLine
    'MSChart1.Title.Text = "Test plot count-rate"
    MSChart1.Plot.Axis(VtChAxisIdX).AxisTitle.Text = "Date"
    MSChart1.Plot.Axis(VtChAxisIdY).AxisTitle.Text = "Count"
    'MSChart1.FootnoteText = "This GUI Test Plot for multi Station"
    Dim X As Integer
    For X = 1 To XS.RecordCount
        chrtArray(X, 1) = XS!Date_Time
        chrtArray(X, 2) = XS!CountCal
```

```
        XS.MoveNext
    Next X
        With MSChart1
        .ChartData = chrtArray
        .ColumnCount = 1
        .ColumnLabelCount = 1
        .ColumnLabel = "cps"
        .Column = 1
        .Refresh
    End With
End Sub
Private Sub cmdConnect_Click()
    If cmdConnect.Caption = "Connect" Then
        If Check1 Then
            Call WinsockStation1.Connect("10.80.72.199", 9760)
        End If
        If Check2 Then
            Call WinsockStation2.Connect("10.80.72.200", 9760)
        End If
        If Check3 Then
            Call WinsockStation3.Connect("10.80.72.201", 9760)
        End If
        If Check1 Or Check2 Or Check3 Then
            cmdConnect.Caption = "Close"
        End If
    Else
        WinsockStation1.Close
        WinsockStation2.Close
        WinsockStation3.Close
        cmdConnect.Caption = "Connect"
    End If
End Sub
Private Sub cmdStart_Click()
    'Set RS = New Recordset
    'Statement = "select Cycle  from Status where ID=1"
    'RS.Open Statement, ConnMyDB, adOpenForwardOnly, , adCmdText
    'txtCycleTime1.Text = "" & RS("Cycle")
    'RS.Close
    'Statement = "select Cycle  from Status where ID=2"
    'RS.Open Statement, ConnMyDB, adOpenForwardOnly, , adCmdText
    'txtCycleTime2.Text = "" & RS("Cycle")
    'RS.Close
    'Statement = "select Cycle  from Status where ID=3"
```

```
    'RS.Open Statement, ConnMyDB, adOpenForwardOnly, , adCmdText
    'txtCycleTime3.Text = "" & RS("Cycle")
    'RS.Close
    If cmbIP.Text = "10.80.72.199" And Check1 Then
        Shape1.BackColor = vbGreen
        If WinsockStation1.State = 7 Then
            WinsockStation1.SendData CStr("SS" + vbLf)
            txtTX1.Text = CStr("SS" + vbLf)
        End If
        txtTime1.Text = txtTime.Text
        txtCycleTime1.Text = txtCycleTime.Text
        TimerAutoread1.Interval = txtTime1
        TimerAutoread1.Enabled = True
        index1 = 0
    End If
    If cmbIP.Text = "10.80.72.200" And Check2 Then
        If WinsockStation2.State = 7 Then
            WinsockStation2.SendData CStr("SS" + vbLf)
            txtTX2.Text = CStr("SS" + vbLf)
        End If
        txtTime2.Text = txtTime.Text
        txtCycleTime2.Text = txtCycleTime.Text
        TimerAutoread2.Interval = txtTime2
        TimerAutoread2.Enabled = True
        index2 = 0
    End If
    If cmbIP.Text = "10.80.72.201" And Check3 Then
        If WinsockStation3.State = 7 Then
            WinsockStation3.SendData CStr("SS" + vbLf)
            txtTX3.Text = CStr("SS" + vbLf)
        End If
        txtTime3.Text = txtTime.Text
        txtCycleTime3.Text = txtCycleTime.Text
        TimerAutoread3.Interval = txtTime3
        TimerAutoread3.Enabled = True
        index3 = 0
    End If
End Sub
Private Sub cmdStop_Click()
    If cmbIP.Text = "10.80.72.199" Then
        If WinsockStation1.State = 7 Then
            WinsockStation1.SendData CStr("SS" + vbLf)
            txtTX1.Text = CStr("SS" + vbLf)
```

```
        End If
        TimerAutoread1.Enabled = False
        Shape1.BackColor = vbBlack
    End If
    If cmbIP.Text = "10.80.72.200" Then
        If WinsockStation2.State = 7 Then
            WinsockStation2.SendData CStr("SS" + vbLf)
            txtTX2.Text = CStr("SS" + vbLf)
        End If
        TimerAutoread2.Enabled = False
        Shape2.BackColor = vbBlack
    End If
    If cmbIP.Text = "10.80.72.201" Then
        If WinsockStation3.State = 7 Then
            WinsockStation3.SendData CStr("SS" + vbLf)
            txtTX3.Text = CStr("SS" + vbLf)
        End If
        TimerAutoread3.Enabled = False
        Shape3.BackColor = vbBlack
    End If
End Sub
Private Sub Command1_Click()
    updateFlexGrid (1)
End Sub
Private Sub Form_Load()
    OpenDataBase
    Call SetupScreen
    Call DisplayIP
    Call RecordToScreen(1)
    Call updateFlexGrid(1)
    Set RS = New Recordset
    Statement = "select Cycle  from Status where ID=1"
    RS.Open Statement, ConnMyDB, adOpenForwardOnly, , adCmdText
    txtCycleTime1.Text = "" & RS("Cycle")
    RS.Close
    Statement = "select Cycle  from Status where ID=2"
    RS.Open Statement, ConnMyDB, adOpenForwardOnly, , adCmdText
    txtCycleTime2.Text = "" & RS("Cycle")
    RS.Close
    Statement = "select Cycle  from Status where ID=3"
    RS.Open Statement, ConnMyDB, adOpenForwardOnly, , adCmdText
    txtCycleTime3.Text = "" & RS("Cycle")
    RS.Close
```

```
    index1 = 0
    index2 = 0
    index3 = 0
End Sub
Private Sub TimerAutoread1_Timer()
    Shape1.BackColor = vbBlack
    'after start wait until = txtTime1.text --> Stop
    If WinsockStation1.State = 7 Then
        WinsockStation1.SendData CStr("SS" + vbLf)
        txtTX1.Text = CStr("SS" + vbLf)
    End If
    'wait for 2 sec before read
    TimeDelay 2
    Shape1.BackColor = vbRed
  If WinsockStation1.State = 7 Then
        WinsockStation1.SendData CStr("RS" + vbLf)
        txtTX1.Text = CStr("RS" + vbLf)
    End If
    'wait for 2 sec after read
    TimeDelay 2
    Shape1.BackColor = vbBlack
    'Start again
    If WinsockStation1.State = 7 Then
        WinsockStation1.SendData CStr("SS" + vbLf)
        txtTX1.Text = CStr("SS" + vbLf)
    End If
    TimeDelay 2
    Shape1.BackColor = vbGreen
End Sub
Private Sub TimerAutoread2_Timer()
    Shape2.BackColor = vbBlack
    'after start wait until = txtTime1.text --> Stop
    If WinsockStation2.State = 7 Then
        WinsockStation2.SendData CStr("SS" + vbLf)
        txtTX2.Text = CStr("SS" + vbLf)
    End If
    'wait for 2 sec before read
    TimeDelay 2
    Shape2.BackColor = vbRed
    If WinsockStation2.State = 7 Then
        WinsockStation2.SendData CStr("RS" + vbLf)
        txtTX2.Text = CStr("RS" + vbLf)
    End If
```

```
    'wait for 2 sec after read
    TimeDelay 2
    Shape2.BackColor = vbBlack
    'Start again
    If WinsockStation2.State = 7 Then
       WinsockStation2.SendData CStr("SS" + vbLf)
       txtTX2.Text = CStr("SS" + vbLf)
    End If
    TimeDelay 2
    Shape2.BackColor = vbGreen
End Sub
Private Sub TimerAutoread3_Timer()
    Shape3.BackColor = vbBlack
    'after start wait until = txtTime1.text --> Stop
    If WinsockStation3.State = 7 Then
       WinsockStation3.SendData CStr("SS" + vbLf)
       txtTX3.Text = CStr("SS" + vbLf)
    End If
    'wait for 2 sec before read
    TimeDelay 2
    Shape2.BackColor = vbRed
    If WinsockStation3.State = 7 Then
       WinsockStation3.SendData CStr("RS" + vbLf)
       txtTX3.Text = CStr("RS" + vbLf)
    End If
    'wait for 2 sec after read
    TimeDelay 2
    Shape3.BackColor = vbBlack
    'Start again
    If WinsockStation3.State = 7 Then
       WinsockStation3.SendData CStr("SS" + vbLf)
       txtTX3.Text = CStr("SS" + vbLf)
    End If
    TimeDelay 2
    Shape3.BackColor = vbGreen
End Sub
Private Sub TimerDateTime_Timer()
    lbDateTime = "" & Date & "  " & Time
End Sub
Private Sub TimerWinsock1_Timer()
    If WinsockStation1.State = 0 Then sendstatus1.Caption = "Closed"
    If WinsockStation1.State = 1 Then sendstatus1.Caption = "Open"
    If WinsockStation1.State = 2 Then sendstatus1.Caption = "Listening"
```

```
    If WinsockStation1.State = 3 Then sendstatus1.Caption = "Connection Pending"
    If WinsockStation1.State = 4 Then sendstatus1.Caption = "Resolving Host"
    If WinsockStation1.State = 5 Then sendstatus1.Caption = "Host Resolved"
    If WinsockStation1.State = 6 Then sendstatus1.Caption = "Connecting"
    If WinsockStation1.State = 7 Then sendstatus1.Caption = "Connected"
    If WinsockStation1.State = 8 Then sendstatus1.Caption = "No Carrier"
    If WinsockStation1.State = 9 Then sendstatus1.Caption = "Error"
End Sub
Private Sub TimerWinsock2_Timer()
    If WinsockStation2.State = 0 Then sendstatus2.Caption = "Closed"
    If WinsockStation2.State = 1 Then sendstatus2.Caption = "Open"
    If WinsockStation2.State = 2 Then sendstatus2.Caption = "Listening"
    If WinsockStation2.State = 3 Then sendstatus2.Caption = "Connection Pending"
    If WinsockStation2.State = 4 Then sendstatus2.Caption = "Resolving Host"
    If WinsockStation2.State = 5 Then sendstatus2.Caption = "Host Resolved"
    If WinsockStation2.State = 6 Then sendstatus2.Caption = "Connecting"
    If WinsockStation2.State = 7 Then sendstatus2.Caption = "Connected"
    If WinsockStation2.State = 8 Then sendstatus2.Caption = "No Carrier"
    If WinsockStation2.State = 9 Then sendstatus2.Caption = "Error"
End Sub
Private Sub TimerWinsock3_Timer()
    If WinsockStation3.State = 0 Then sendstatus3.Caption = "Closed"
    If WinsockStation3.State = 1 Then sendstatus3.Caption = "Open"
    If WinsockStation3.State = 2 Then sendstatus3.Caption = "Listening"
    If WinsockStation3.State = 3 Then sendstatus3.Caption = "Connection Pending"
    If WinsockStation3.State = 4 Then sendstatus3.Caption = "Resolving Host"
    If WinsockStation3.State = 5 Then sendstatus3.Caption = "Host Resolved"
    If WinsockStation3.State = 6 Then sendstatus3.Caption = "Connecting"
    If WinsockStation3.State = 7 Then sendstatus3.Caption = "Connected"
    If WinsockStation3.State = 8 Then sendstatus3.Caption = "No Carrier"
    If WinsockStation3.State = 9 Then sendstatus3.Caption = "Error"
End Sub
Private Sub updatedatabase_Click()
    XS.AddNew
    XS.Fields("IP") = "10.80.72.199"
    XS.Fields("Date_Time") = lbDateTime
    XS.Fields("Count") = Val(txtRX1)
    XS.Fields("CountCal") = Val((0.0014 * txtRX1) - 0.0198)
    XS.Fields("IndexIP") = "1"
    'y = 1.0836x - 5.0423
    XS.Update
End Sub
Private Sub WinsockStation1_DataArrival(ByVal bytesTotal As Long)
```

```
    ' Dim y1, y2, y3, ytest As Double
    ' ytest = 0.001
    If index1 < txtCycleTime1 Then
        WinsockStation1.GetData buffer1$, vbString
        Beep
        txtRX1.Text = buffer1$
        XS.AddNew
        XS.Fields("IP") = "10.80.72.199"
        XS.Fields("Date_Time") = lbDateTime
        XS.Fields("Count") = Val(txtRX1)
        XS.Fields("CountCal") = Val((0.0009 * txtRX1) - 0.1959)
        XS.Fields("IndexIP") = "1"
            ' This section for define Alarm
       ' If (Val((0.0014 * txtRX1) - 0.0198)) > ytest Then
        '    MsgBox "No Data in Data-Base"
        'End If
        XS.Update
        index1 = index1 + 1
        updateFlexGrid (1)
    Else
        If WinsockStation1.State = 7 Then
            WinsockStation1.SendData CStr("SS" + vbLf)
            txtTX1.Text = CStr("SS" + vbLf)
        End If
        TimerAutoread1.Enabled = False
        Shape1.BackColor = vbBlack
    End If
End Sub
Private Sub WinsockStation2_DataArrival(ByVal bytesTotal As Long)
    If index2 < txtCycleTime2 Then
        WinsockStation2.GetData buffer2$, vbString
        Beep
        txtRX2.Text = buffer2$
        XS.AddNew
        XS.Fields("IP") = "10.80.72.200"
        XS.Fields("Date_Time") = lbDateTime
        XS.Fields("Count") = Val(txtRX2)
        XS.Fields("CountCal") = Val((1.0836 * txtRX2) - 5.0423)
        XS.Fields("IndexIP") = "2"
        XS.Update
        index2 = index2 + 1
        updateFlexGrid (2)
    Else
```

```
        If WinsockStation2.State = 7 Then
            WinsockStation2.SendData CStr("SS" + vbLf)
            txtTX2.Text = CStr("SS" + vbLf)
        End If
        TimerAutoread2.Enabled = False
        Shape2.BackColor = vbBlack
    End If
End Sub
Private Sub WinsockStation3_DataArrival(ByVal bytesTotal As Long)
    If index3 < txtCycleTime3 Then
        WinsockStation3.GetData buffer3$, vbString
        Beep
        txtRX3.Text = buffer3$
        XS.AddNew
        XS.Fields("IP") = "10.80.72.201"
        XS.Fields("Date_Time") = lbDateTime
        XS.Fields("Count") = Val(txtRX3)
        XS.Fields("CountCal") = Val((1.0836 * txtRX3) - 5.0423)
        XS.Fields("IndexIP") = "3"
        XS.Update
        index3 = index3 + 1
        updateFlexGrid (3)
    Else
        If WinsockStation3.State = 7 Then
            WinsockStation3.SendData CStr("SS" + vbLf)
            txtTX3.Text = CStr("SS" + vbLf)
        End If
        TimerAutoread3.Enabled = False
        Shape3.BackColor = vbBlack
    End If
End Sub
'' This part for Manual Mode
Private Sub cmdRS_Click()
    If WinsockStation1.State = 7 Then
        WinsockStation1.SendData CStr("RS" + vbLf)
        txtTX1.Text = CStr("RS" + vbLf)
    End If
    If WinsockStation2.State = 7 Then
        WinsockStation2.SendData CStr("RS" + vbLf)
        txtTX2.Text = CStr("RS" + vbLf)
    End If
    If WinsockStation3.State = 7 Then
        WinsockStation3.SendData CStr("RS" + vbLf)
```

```
        txtTX3.Text = CStr("RS" + vbLf)
      End If
    End Sub
    Private Sub cmdSC001_Click()
      If WinsockStation1.State = 7 Then
        WinsockStation1.SendData CStr("SC001" + vbLf)
        txtTX1.Text = CStr("SC001" + vbLf)
      End If
      If WinsockStation2.State = 7 Then
        WinsockStation2.SendData CStr("SC001" + vbLf)
        txtTX2.Text = CStr("SC001" + vbLf)
      End If
      If WinsockStation3.State = 7 Then
        WinsockStation3.SendData CStr("SC001" + vbLf)
        txtTX3.Text = CStr("SC001" + vbLf)
      End If
    End Sub
    Private Sub cmdSC002_Click()
      If WinsockStation1.State = 7 Then
        WinsockStation1.SendData CStr("SC002" + vbLf)
        txtTX1.Text = CStr("SC002" + vbLf)
      End If
      If WinsockStation2.State = 7 Then
        WinsockStation2.SendData CStr("SC002" + vbLf)
        txtTX2.Text = CStr("SC002" + vbLf)
      End If
      If WinsockStation3.State = 7 Then
        WinsockStation3.SendData CStr("SC002" + vbLf)
        txtTX3.Text = CStr("SC002" + vbLf)
      End If
    End Sub
' in this case we send SS + (vblf = 0A)
    Private Sub cmdSS_Click()
      If WinsockStation1.State = 7 Then
       WinsockStation1.SendData CStr("SS" + vbLf)
        txtTX1.Text = CStr("SS" + vbLf)
      End If
      If WinsockStation2.State = 7 Then
        WinsockStation2.SendData CStr("SS" + vbLf)
        txtTX3.Text = CStr("SS" + vbLf)
      End If
    End Sub
```

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวปิยนุช เนตรคุณ เกิดวันที่ 10 พฤษภาคม พ.ศ.2528 ที่จังหวัดกาฬสินธุ์ สำเร็จการศึกษาระดับปริญญาวิทยาศาสตรบัณฑิต สาขาฟิสิกส์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี ในปีการศึกษา 2550 และในปีการศึกษา 2551 ได้เข้าศึกษาระดับปริญญามหาบัณฑิตที่ภาควิชาวิศวกรรมนิวเคลียร์ สาขาวิชานิวเคลียร์เทคโนโลยี คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย