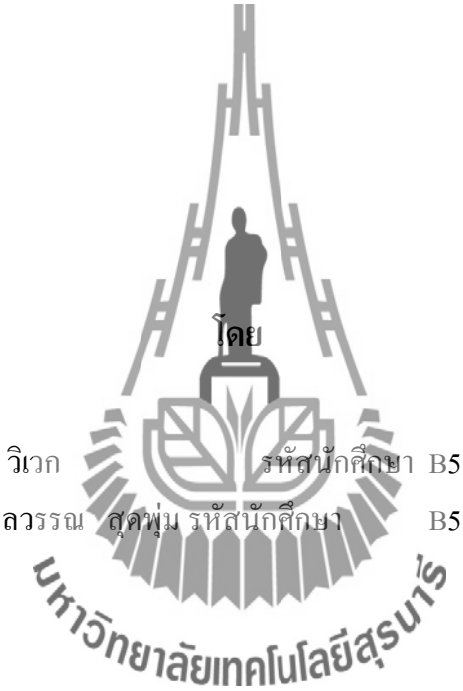
ระบบเก็บข้อมูลระหว่างการเดินรถของรถไฟ



โดย

นายวิศรุต วิเวก รหัสนักศึกษา B5103690

นางสาวกมลวรรณ สุดพุ่ม รหัสนักศึกษา B5106226

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 427499 โครงงานวิศวกรรมโทรคมนาคม

หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2545

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ประจำภาคการศึกษาที่ 1 ปีการศึกษา 2554

ระบบรักษาความปลอดภัยของรถไฟ

คณะกรรมการสอบโครงงาน

(อาจารย์ ดร. สมศักดิ์ วาณิชอนันต์ชัย)
กรรมการ/อาจารย์ที่ปรึกษาโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร. มนต์ทิพย์ภา อุฑารสกุล)
กรรมการ

(ผู้ช่วยศาสตราจารย์ เรืออากาศเอก ดร. ประโยชน์ คำสวัสดิ์)
กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ปิยาภรณ์ กระฉอดนอก)
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับรายงานโครงงานฉบับนี้ เป็นส่วนหนึ่งของการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม วิชา 427499 โครงงานวิศวกรรมโทรคมนาคม ประจำปีการศึกษา 2553

โครงงาน **ระบบเก็บข้อมูลระหว่างการเดินรถของรถไฟ**

จัดทำโดย 1. นายวิศรุต วิเวก รหัสประจำตัว B5103690
2. นางสาวกมลวรรณ สุดพุ่ม รหัสประจำตัว B5106226
อาจารย์ที่ปรึกษา อาจารย์ ดร.สมศักดิ์ วาณิชอนันต์ชัย
สาขาวิชา วิศวกรรมโทรคมนาคม
ภาคการศึกษาที่ 1/ 2554

บทคัดย่อ
(Abstract)

โครงงาน ระบบเก็บข้อมูลระหว่างการเดินรถของรถไฟ เป็นการนำเทคโนโลยีโดยใช้เครือข่ายไร้สาย (Zigbee) มาประยุกต์และควบคุมด้วยไมโครคอนโทรลเลอร์ เพื่อเก็บข้อมูลทางด้านสถิติและด้านความปลอดภัยของรถไฟ

ซึ่งในโครงงานจะต้องเขียนโปรแกรมไมโครคอนโทรลเลอร์ให้มีการบันทึกเวลาทุกครั้งที่ที่กั้นรถไฟมีการ ขึ้น-ลง โดยจะส่งข้อมูลนั้นผ่านทาง Wireless Sensor Network นอกจากนี้ข้อมูลนี้จะถูกส่งมายัง ไมโครคอนโทรลเลอร์เพื่อบันทึกไว้เป็นฐานข้อมูลและสามารถที่จะเปิดดูข้อมูลได้โดยสามารถกลับไปดูข้อมูลย้อนหลังได้

กิตติกรรมประกาศ

(Acknowledgement)

จากการที่คณะจัดทำรายงานได้รับมอบหมายให้ทำโครงงานเรื่อง ระบบเก็บข้อมูลระหว่างการเดินรถของรถไฟ ทำให้คณะผู้จัดทำได้รับประโยชน์และความรู้เพิ่มมากขึ้น เกี่ยวกับการใช้เครือข่ายไร้สาย Zigbee , ไมโครคอนโทรลเลอร์เบอร์ MSP430 G2231, ไมโครคอนโทรลเลอร์เบอร์ P89V51RD2 ตลอดจนการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ได้เป็นอย่างดี บัดนี้โครงงานเรื่องระบบรักษาความปลอดภัยของรถไฟ สามารถเสร็จลุล่วงไปได้ด้วยดีเนื่องด้วยคณะผู้จัดทำได้รับความอนุเคราะห์ช่วยเหลือในด้านต่างๆ จนทำให้สามารถดำเนินงานสำเร็จลุล่วงไปได้ด้วยดี จึงใคร่ขอขอบพระคุณบุคคลดังรายนามต่อไปนี้

1. อาจารย์ ดร. สมศักดิ์ วาณิชอนันต์ชัย อาจารย์ที่ปรึกษาโครงงานที่ให้คำแนะนำ ให้คำปรึกษา และดูแลการทำโครงงานอย่างใกล้ชิดตลอดการทำโครงงาน

2. อาจารย์ วิชัย ศรีสุรักษ์ อาจารย์ที่ให้คำแนะนำด้านการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์

3. ผู้ช่วยศาสตราจารย์ เรืออากาศเอก ดร. ประโยชน์ คำสวัสดิ์ ที่ให้ยืมและอำนวยความสะดวกในเรื่องอุปกรณ์ที่ใช้ในการทำโครงงานครั้งนี้

คณาจารย์ทุกท่าน คุณพ่อ คุณแม่ และเพื่อนๆ ปริญญาตรีสาขาวิชาวิศวกรรมโทรคมนาคมทุกคนที่ให้ความช่วยเหลือมาโดยตลอด

โครงงานนี้จะไม่สามารถสำเร็จลุล่วงไปได้หากปราศจากแรงสนับสนุนจากบุคคลดังรายนามข้างต้น ทางคณะผู้จัดทำจึงขอขอบพระคุณทุกๆท่านเป็นอย่างสูงมา ณ โอกาสนี้ หากมีข้อผิดพลาดประการใด คณะผู้จัดทำใคร่ขออภัยมา ณ ที่นี้ด้วย

นายวิศรุต วิเวก

นางสาวกมลวรรณ สุดพุ่ม

สารบัญ

	หน้า
บทคัดย่อ……....	ก
กิตติกรรมประกาศ…….	ข
สารบัญ……	ค
สารบัญภาพ……...	จ
สารบัญตาราง……	ฉ
บทที่ 1 บทนำ	
1.1 บทนำ…….	1
1.2 หลักการและเหตุผล……...	3
1.3 วัตถุประสงค์……...	4
1.4 ขอบเขตงาน……	4
1.5 ผลที่คาดว่าจะได้รับ……	4
บทที่ 2 มาตรฐาน IEEE 802.15.4 / ZigBee	
2.1 มาตรฐาน IEEE 802.15.4 / ZigBee คืออะไร……...	5
2.2 โปรโตคอลของ ZigBee (ZigBee Protocol)……	6
2.2.1 ขั้นตอนการทำงานของโพรโทคอล ZigBee……..	7
2.3 โครงสร้างของเครือข่าย (ZigBee Topologies)……...	8
2.3.1 โครงสร้างเครือข่ายแบบดาว (Star Topology)……..	9
2.3.2 โครงสร้างเครือข่ายแบบเมช (Mesh Topology)……...	10
2.3.3 โครงสร้างเครือข่ายแบบกลุ่มต้นไม้ (Cluster Tree Topology)…….	11
บทที่ 3 การต่อไมโครคอนโทรลเลอร์กับ Xbee	
3.1 การต่อไมโครคอนโทรลเลอร์กับ Xbee เบื้องต้น……	12
3.2 Xbee ที่เลือกใช้……..	13
3.3 ไมโครคอนโทรลเลอร์ที่เลือกใช้…….	16
3.3.1 ไมโครคอนโทรลเลอร์ที่ฝั่งส่งข้อมูล MSP430 G2231……..	17
3.3.2 ไมโครคอนโทรลเลอร์ที่ฝั่งรับข้อมูล P89V51RD2……...	18

3.3.2.1 การโหลดโปรแกรม P89V51RD2 ด้วย Flash Magic 5…… 20

หน้า

3.4 การใช้โปรแกรม Hyperterminal สำหรับติดต่อกับไมโครคอลโทรลเลอร์ 23
3.5 UART,TTL, RS232,MAX232,MAX3232 คืออะไร…….. 26
บทที่ 4 วิธีการดำเนินโครงงาน
4.1 ลักษณะการทำงานโดยรวมของระบบ…… 33
4.1.1 ทำไมจึงส่งข้อมูล serial number ของ Xbee……... 34
4.2 การ Configuration Xbee……. 34
4.2.1 การตั้งค่า XBee ให้ทำงานเป็น Coordinator…….. 35
4.2.2 การตั้งค่า XBee ให้ทำงานเป็น End Device……... 36
4.2.3 การทดสอบผลการ Configuration……. 38
4.3 การทำงานของฝั่งส่ง…… 38
4.4 การทำงานของฝั่งรับ……... 40
4.4.1 RTC DS1307…… 43
4.4.2 EEPROM…….. 43
4.5 ความสามารถในการเก็บข้อมูล…… 44
บทที่ 5 ผลการทดสอบ
5.1 การทดสอบการบันทึกข้อมูลซ้ำ…….. 45
5.2 การทดสอบการรับ-ส่งข้อมูลและบันทึกเวลา…….. 46
บทที่ 6 บทสรุป
6.1 สรุปผลการดำเนินโครงการ…… 48
6.2 ข้อเสนอแนะ……... 48
ประวัติผู้เขียน..... 49
บรรณานุกรม..... 50
ภาคผนวก
โปรแกรม code_tx.c……. 51
โปรแกรม recive.asm…… 58
โปรแกรมย่อย 00_Systm.sub…… 63
โปรแกรมย่อย 10_D1307.sub……... 70

โปรแกรมย่อย EEPROM.sub…… 81
ตัวอย่างกรณีศึกษาอุบัติเหตุ…….. 94

สารบัญตาราง

หน้า

ตารางที่ 2.1 กลุ่มความถี่ของมาตรฐาน IEEE 802.15.4 (Frequency Band)……... 5
ตารางที่ 3.1 เปรียบเทียบ Xbee แบบต่างๆ……. 13
ตารางที่ 3.2 แสดงรายละเอียด PIN ของ Xbee……... 15
ตารางที่ 4.1 แสดงการเชื่อมต่อ MSP430 G2231 เข้ากับ Xbee…….. 40
ตารางที่ 4.2 แสดงการเชื่อมต่อ MCU-P89V51RD2 เข้ากับอุปกรณ์..... 41
ตารางที่ 4.3 แสดงความสามารถในการเก็บข้อมูลเมื่อกำหนดหน่วยความจำแตกต่างกัน……. 44

สารบัญรูปภาพ

หน้า

รูปที่ 1.1 แสดงการเปรียบเทียบเทคโนโลยีไร้สาย……. 2

รูปที่ 2.1 แสดงช่องสัญญาณตามมาตรฐาน ZigBee……... 5

รูปที่ 2.2 แสดงโครงสร้างโปรโตคอลของ ZigBee…… 7

รูปที่ 2.3 แสดงโครงสร้างของเครือข่ายแบบต่างของ Xbee……... 8

รูปที่ 2.4 โครงสร้างเครือข่ายแบบ Star…….. 9

รูปที่ 2.5 โครงสร้างเครือข่ายแบบ Mesh……... 10

รูปที่ 2.6 โครงสร้างเครือข่ายแบบ Cluster Tree…… 11

รูปที่ 3.1 การต่อไมโครคอนโทรลเลอร์กับ Xbee เบื้องต้น…… 12

รูปที่ 3.2 รูปแบบของสายอากาศ Xbee…….. 14

รูปที่ 3.3 Xbee Pro Serie2 รุ่น XBP24-BWIT-004…… 14

รูปที่ 3.4 โครงสร้างของ Xbee……... 15

รูปที่ 3.5 XBee Explorer Regulated และ XBee Explorer Dongle……. 16

รูปที่ 3.6 MSP430G2 LunchPAD…….. 17

รูปที่ 3.7 บอร์ด MCS51 รุ่น SUT-V5x: MCU-P89V51RD2……. 18

รูปที่ 3.9 PIN ต่างๆของ MCU-P89V51RD2…… 19

รูปที่ 3.10 ระดับสัญญาณ TTL 5 V และ 3 V…… 26

รูปที่ 3.11 การสื่อสารอนุกรมแบบ Synchronize……... 27

รูปที่ 3.12 การสื่อสารอนุกรมแบบ Asynchronize……. 27

รูปที่ 3.13 แนวคิดวิธีเชื่อมต่อระหว่างอุปกรณ์…….. 28

รูปที่ 3.14 ระดับสัญญาณ TTL และ RS232…….. 28

รูปที่ 3.15 การสื่อสารระหว่างคอมพิวเตอร์กับ MCU…… 29

รูปที่ 3.16 วงจรแปลงระดับสัญญาณ TTL 0 - 5V เป็น RS 232……. 29

รูปที่ 3.17 วงจรแปลงระดับสัญญาณ TTL 0 - 3.3V เป็น RS 232…….. 30

รูปที่ 3.18 ขาของ คอนเน็กเตอร์ DB9…… 30

รูปที่ 3.19 การเชื่อมต่อสาย DB9……... 31

หน้า

รูปที่ 3.20 การสื่อสารระหว่างคอมพิวเตอร์กับ MCU…… 32
รูปที่ 3.20 การสื่อสารระหว่าง MCU…… 32
รูปที่ 4.1 Flow chart ของฝั่งส่ง..... 33
รูปที่ 4.2 Flow chart ของฝั่งรับ…….. 34
รูปที่ 4.3 โครงข่ายไร้สายของ Xbee ในโครงงาน..... 35
รูปที่ 4.4 ตัวอย่างรูปการหา SH, SL, MODEM…….. 35
รูปที่ 4.4 แสดงขั้นตอนการตั้งค่า XBee ให้ทำงานเป็น Coordinator…….. 36
รูปที่ 4.5 แสดงขั้นตอนการตั้งค่า XBee ให้ทำงานเป็น End Device…….. 37
รูปที่ 4.6 วิธีการเชื่อมต่อในการ Configuration…….. 37
รูปที่ 4.7 การทดสอบการ Configuration ด้วยแท็บ Terminal…… 38
รูปที่ 4.8 แสดงการส่งข้อมูลใช้ MSP430 G2231ผ่านทาง Hyperterminal……. 39
รูปที่ 4.9 แสดงการเชื่อมต่อ MSP430 G2231 เข้ากับ Xbee…… 40
รูปที่ 4.10 แสดงการเชื่อมต่อบอร์ด SUT-V5x: MCU-P89V51RD2 เข้ากับอุปกรณ์. 41
รูปที่ 4.11 แสดงข้อมูลและเวลาที่รับได้..... 42
รูปที่ 4.12 แสดงข้อมูลและเวลาที่รับได้ที่ถูกบันทึก..... 42
รูปที่ 4.13 RTC DS1307…… 43
รูปที่ 4.14 EEPROM…….. 44
รูปที่ 5.1 การทดสอบการบันทึกข้อมูลซ้ำ..... 45
รูปที่ 5.2 การทดสอบการบันทึกข้อมูลซ้ำ…….. 46
รูปที่ 5.3 การทดสอบการรับ-ส่งข้อมูลและบันทึกเวลา…… 47

บทที่ 1
บทนำ

1.1 บทนำ

เทคโนโลยีเครือข่ายไร้สาย(Wireless LAN) ในปัจจุบันมีมากมายหลายรูปแบบ เช่น GSM, CDMA, WiFi, WiMAX , BlueTooth ฯลฯ ซึ่งแต่ละรูปแบบมีอัตราการรับข้อมูลที่สูงต่ำแตกต่างกันไป และเป็นระบบการสื่อสารข้อมูลที่มีความคล่องตัวมาก ซึ่งอาจจะนำมาใช้ทดแทนหรือเพิ่มต่อกับระบบเครือข่ายแลนใช้สายแบบดั้งเดิม โดยใช้การส่งคลื่นความถี่วิทยุในย่านวิทยุ RF และ คลื่นอินฟราเรด ในการรับและส่งข้อมูลระหว่างคอมพิวเตอร์แต่ละเครื่อง ผ่านอากาศ , ทะลุกำแพง , เพดานหรือสิ่งก่อสร้างอื่นๆ โดยปราศจากความต้องการของการเดินสาย นอกจากนั้นระบบเครือข่ายไร้สายก็ยังมีคุณสมบัติครอบคลุมทุกอย่างเหมือนกับระบบ LAN แบบใช้สาย ที่สำคัญก็คือการที่มันไม่ต้องใช้สายทำให้การเคลื่อนย้ายการใช้งานทำได้โดยสะดวก ไม่เหมือนระบบ LAN แบบใช้สาย ที่ต้องใช้เวลาและการลงทุนในการปรับเปลี่ยนตำแหน่งการใช้งานเครื่องคอมพิวเตอร์ ซึ่งงานทางด้านไมโครคอนโทรลเลอร์ จะมีความเกี่ยวข้องกับการสื่อสารทั้งสิ้น เช่น การสร้างเครือข่ายของระบบหนึ่ง ๆ , การติดต่อสื่อสารใช้งานอุปกรณ์ RF Module และ ที่คุ้นเคยกันดีก็คือการสื่อสารเพื่อใช้งานติดต่อกับอุปกรณ์อื่น ๆ ผ่าน Interface ต่าง ๆ เช่น RS232(UART) , I2C , Ethernet , LAN , TCP/IP , USB ฯลฯ ผู้ที่เคยเขียน software ที่เกี่ยวข้องกับการสื่อสารต่าง ๆ จะมีความเข้าใจเกี่ยวกับโปรโตคอลสื่อสาร สามารถเรียนรู้ การสื่อสารแบบอื่น ๆ ได้ไม่ยาก การเขียน software ลักษณะการ รับ stream data เพื่อมาเก็บใน buffer แล้วทำการ encapsulate , de-capsulate ข้อมูล (เช่น การเขียนโปรแกรมทางด้าน network security , UART , I2C ฯลฯ) แล้วนำข้อมูลไปใช้งาน เป็นสิ่งจำเป็นสำหรับงานทางด้านการติดต่อสื่อสารเกือบทุกรูปแบบ และ แสดงการเปรียบเทียบเทคโนโลยีไร้สายแบบต่างๆ ได้ดังนี้

	ZigBee™ 802.15.4	Bluetooth™ 802.15.1	Wi-Fi™ 802.11b	GPRS/GSM 1XRTT/CDMA
Application Focus	Monitoring & Control	Cable Replacement	Web, Video, Email	WAN, Voice/Data
System Resource	4KB-32KB	250KB+	1MB+	16MB+
Battery Life (days)	100-1000+	1-7	.1-5	1-7
Nodes Per Network	255/65K+	7	30	1,000
Bandwidth (kbps)	20-250	720	11,000+	64-128
Range (meters)	1-75+	1-10+	1-100	1,000+
Key Attributes	Reliable, Low Power, Cost Effective	Cost, Convenience	Speed, Flexibility	Reach, Quality

รูปที่ 1.1 แสดงการเปรียบเทียบเทคโนโลยีไร้สาย

1.2 หลักการและเหตุผล

ZigBee เป็นมาตรฐานสากล IEEE 802.15.4 สำหรับการสื่อสารแบบไร้สาย ที่มีอัตราการรับส่งข้อมูลต่ำ ใช้พลังงานต่ำ และด้วยความก้าวหน้าทางด้านเทคโนโลยีจึงทำให้อุปกรณ์มีราคาถูก โดยมีจุดประสงค์ก็เพื่อให้สามารถสร้างระบบที่เรียกว่า Wireless Sensor Network ได้ ซึ่งระบบนี้ จะสามารถทำงาน ในร่ม กลางแจ้ง ทนแดด ทนฝน และใช้กำลังไฟฟ้าน้อย

ในปัจจุบันเทคโนโลยีเกี่ยวกับเครื่องกั้นถนน-รถไฟยังไม่เจริญมากนัก โดยเฉพาะในเขตที่ห่างไกลจากตัวเมืองส่วนมากแล้วจะไม่มีสถานีควบคุมอยู่ในบริเวณนั้นและเมื่อเกิดอุบัติเหตุเกิดขึ้น เราไม่ทราบว่าอุบัติเหตุนั้นเกิดขึ้นจากความประมาทของผู้ขับขี่ยานพาหนะหรือมีความผิดพลาดในการกั้นทางรถไฟ ทางคณะผู้จัดทำจึงได้พัฒนา Wireless Sensor Network เพื่อมาเก็บข้อมูลในการเดินทางและการเกิดอุบัติเหตุของรถไฟ

1.2.1 สภาพการจราจรและขนส่งของประเทศไทยอาศัยทางรถยนต์และทางรถไฟในปริมาณที่เพิ่มขึ้นสูงมาก โดยเฉพาะโครงสร้างพื้นฐานทางรถไฟ ซึ่งจากข้อมูลของการรถไฟแห่งประเทศไทย (รฟท.) ปี พ.ศ. 2550 พบว่า เส้นทางการเดินรถไฟในปัจจุบันมีโครงข่ายรถไฟทั้งประเทศรวม 4,043 กิโลเมตร ผ่านจังหวัดต่างๆทั่วประเทศรวม 47 จังหวัด มีจุดตัดผ่านทางรถไฟทั้งสิ้น 2,449 แห่ง ในจำนวนนี้เป็นทางตัดผ่านที่ได้รับอนุญาตจากการรถไฟ ฯประมาณ 1,914 แห่งและเป็นทางลักผ่านที่ไม่ได้รับอนุญาตจากการรถไฟ ฯประมาณ 535 แห่ง ทั้งนี้การรถไฟ ฯ ได้จัดทำการติดตั้งระบบป้องกันอุบัติเหตุหรือเครื่องกั้นถนน (รวมประเภทไฟเตือนอัตโนมัติ) แล้วประมาณ 637 แห่ง แต่สถิติขบวนรถไฟชนกับยานพาหนะบริเวณทางตัดผ่านยังคงมีระดับมาก ซึ่งจะเห็นได้ว่าปัญหาอุบัติเหตุบริเวณจุดตัดทางรถไฟกับถนนนับวันจะทวีความรุนแรงและเกิดบ่อยครั้งก่อให้เกิดความสูญเสียต่อชีวิตและทรัพย์สินของประชาชนและประเทศ

1.2.2 สำหรับสถิติอุบัติเหตุจุดตัดทางรถไฟกับถนน ปี พ.ศ.2548-2550 มีดังนี้

ปี พ.ศ.	จำนวน	บาดเจ็บ	เสียชีวิต
2548	99	133	40
2549	97	87	25
2550	94	119	31
รวม	290	339	96

1.2.3 ปัจจัยหลักของการเกิดอุบัติเหตุบริเวณจัดตัดทางรถไฟกับถนน สำหรับรถไฟทางไกลประกอบไปด้วยปัจจัยด้านคน ยานพาหนะ ถนนและสิ่งแวดล้อม ซึ่งปัจจัยด้านสิ่งแวดล้อม โดยเฉพาะปัจจัยด้านลักษณะทางกายภาพบริเวณจุดตัดทางรถไฟนั้น เป็นปัจจัยหลักที่มีผลต่อการเกิดอุบัติเหตุบริเวณจุดตัดทางรถไฟกับถนนมากที่สุด มาตรการในการเก็บบันทึกปัญหาอุบัติเหตุ

บริเวณจัดตัดทางรถไฟกับถนนเป็นวิธีการที่ดี เพื่อยืนยันข้อมูลว่าอุบัติเหตุเกิดขึ้นจากสาเหตุใด เพื่อที่จะได้นำไปปรับปรุงระบบการจราจรทางบกให้เป็นไปอย่างปลอดภัย สะดวกและรวดเร็ว รวมทั้งมีระบบการเดินรถไฟทั่วประเทศต่อไปด้วย

1.3 วัตถุประสงค์

1. เพื่อสร้าง Wireless Sensor Network ให้สามารถรับส่งข้อมูลได้
2. เพื่อพัฒนาการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์และบันทึกข้อมูล
3. เพื่อศึกษาการเขียนโปรแกรม ควบคุมการรับ-ส่งข้อมูลอัตโนมัติโดยปราศจากมนุษย์ได้
4. สามารถเข้าถึงฐานข้อมูลและดูข้อมูลในอดีตได้
5. เพื่อที่จะนำข้อมูลที่ได้ไปพัฒนาระบบ แก้ปัญหา และวิเคราะห์การเกิดอุบัติเหตุได้

1.4 ขอบเขตงาน

1. ศึกษาการเขียนโปรแกรม ควบคุมบอร์ดไมโครคอนโทรลเลอร์ ในการรับ-ส่งข้อมูล
2. ศึกษา ZigBee มาตรฐาน IEEE 802.15.4 ในการสร้าง Wireless Sensor Network และสามารถติดต่อสื่อสารกันได้
3. ศึกษาการเข้าสู่ฐานข้อมูล

1.5. ผลที่คาดว่าจะได้รับ

1. ได้เครื่องมือที่สามารถใช้งานได้จริง
2. ข้อมูลที่ได้มีความแม่นยำและถูกต้อง
3. สามารถนำความรู้ที่ได้ไปประกอบวิชาชีพในอนาคตได้
4. สามารถนำไปแก้ไขปัญหาและ วิเคราะห์สาเหตุของการเกิดอุบัติเหตุได้
5. อำนวยความสะดวกแก่ผู้ใช้งานได้อย่างมีประสิทธิภาพ
6. โปรแกรมที่สร้างขึ้นมานี้สามารถนำไปดัดแปลง แก้ไข เพื่อประยุกต์ใช้แก้ปัญหาเฉพาะกรณีๆ ได้

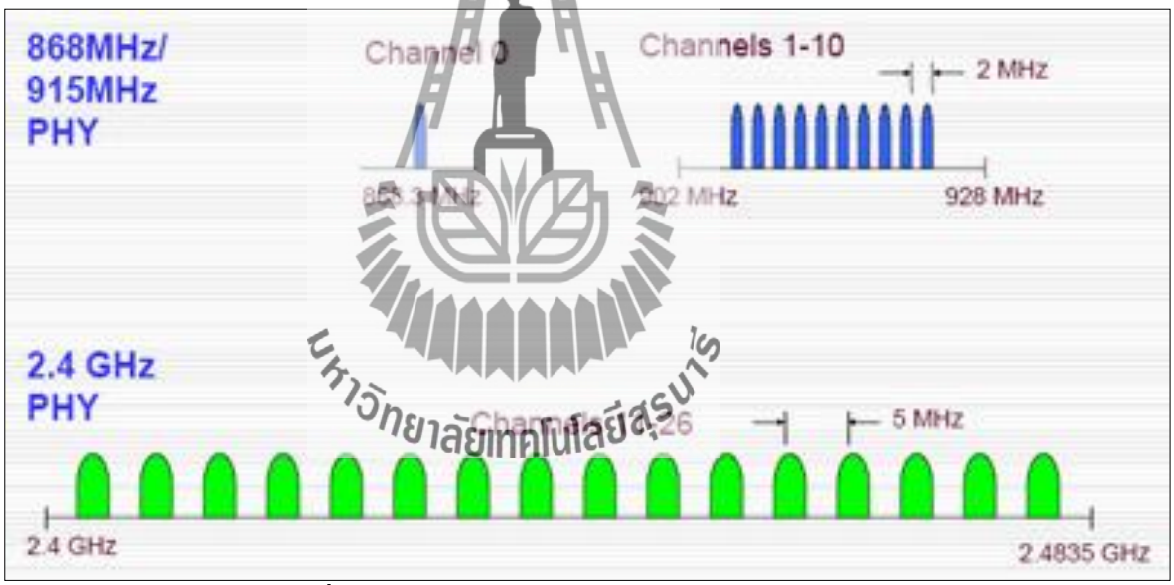
บทที่ 2

มาตรฐาน IEEE 802.15.4 / ZigBee

2.1 มาตรฐาน IEEE 802.15.4 / ZigBee คืออะไร

ZigBee มาตรฐานสากล กำหนดโดย ZigBee Alliance เป็น การสื่อสารแบบไร้สาย ที่มีอัตราการรับส่งข้อมูลต่ำ ใช้พลังงานต่ำ และราคาถูก จุดประสงค์ก็เพื่อให้สามารถสร้างระบบที่เรียกว่า Wireless Sensor Network ได้ ซึ่งระบบนี้ จะสามารถทำงาน ในร่ม กลางแจ้ง ทนแดด ทนฝนและอยู่ได้ด้วยแบตเตอรี่ก้อนเล็ก (เช่นถ่าน AA 2 ก้อน) นานเป็นเดือนเป็นปี

ZigBeeกำหนด ย่านความถี่ใช้งานตามมาตรฐานไว้ 3 ย่านความถี่คือ ย่าน 2.4 GHz , ย่าน 915 MHzและย่าน 868 MHzโดยแต่ละย่านจะมีช่องสัญญาณ 16 ช่อง , 10 ช่องและ 1 ช่อง ตามลำดับ ส่วนอัตรารับส่งข้อมูล (ทางอากาศ) จะอยู่ที่ 250 Kbps , 40 Kbps , 20 Kbps ตามลำดับ



รูปที่ 2.1 แสดงช่องสัญญาณตามมาตรฐาน ZigBee

ตารางที่ 2.1 กลุ่มความถี่ของมาตรฐาน IEEE 802.15.4 (Frequency Band)

BAND	COVERRAGE	DATA RATE	CHANNELS
2.4 GHz	WorldWide	250 Kbps	16
915 MHz	Americas	40 Kbps	10
868 MHz	Europe	20 Kbps	1

ชนิดอุปกรณ์ของ ZigBeeมีอยู่ 2 ชนิดคือ แบบ Physical Device และ Logical Device

แบบ Physical Deviceมี 2 ประเภท คือ

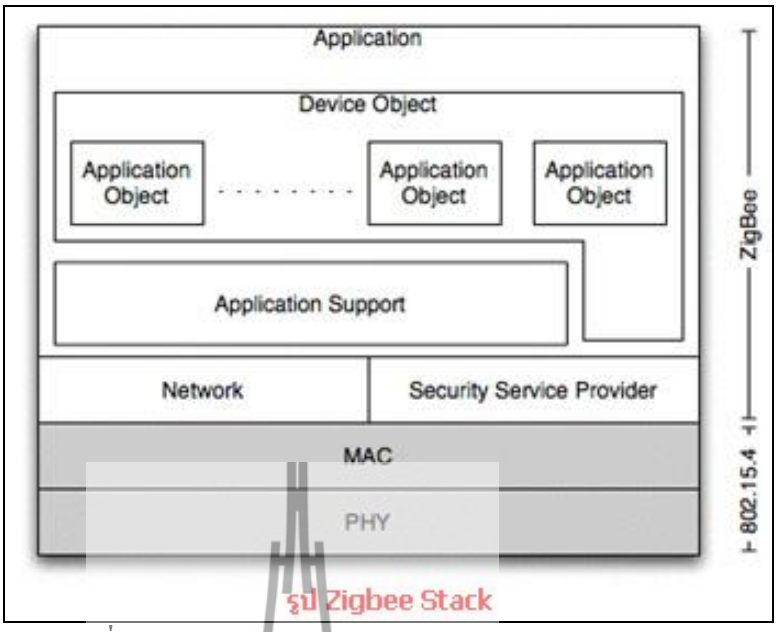
- Full Function Device : FFD เป็นเราเตอร์ที่เป็นสื่อกลางในการส่งข้อมูลจากอุปกรณ์อื่นๆ ใช้พลังงานจาก power line ทำงานได้ในทุก Topology และสามารถทำเป็นจุดเชื่อมต่อกันได้
- Reduced Function Device : RFD เหมาะแก่การเชื่อมต่อภายในเครือข่ายใช้พลังงานจากแบตเตอรี่ ไม่สามารถถ่ายทอดข้อมูลจากอุปกรณ์อื่นๆ ได้ทำได้ง่ายในเครือข่ายที่เป็นแบบ star

แบบ Logical Deviceมี 3 ประเภท คือ

- ZigBee Coordinators เป็นจุดที่ประสานเชื่อมต่อกัน ทำหน้าที่ในการจัดเก็บข้อมูลในเครือข่าย
- ZigBee Routers ทำหน้าที่จัดการเส้นทางของข้อความที่ส่งผ่านภายในโครงข่ายระหว่างคู่ของโนดใด ๆ
- ZigBee End Devices เป็นโนดที่อยู่ในส่วนของผู้ใช้งาน โดยสามารถเป็นได้ทั้งแบบ RFD และ FFD

2.2 โปรโตคอลของ ZigBee (ZigBee Protocol)

สถาปัตยกรรมของ ZigBee Stack จะเป็นไปตาม OSI seven-layer model โปรโตคอล ZigBee จะใช้เฉพาะในส่วนของชั้นโปรแกรมประยุกต์ (Application Layer) ชั้นสนับสนุนโปรแกรมประยุกต์ (Application Support Layer) และชั้นเครือข่าย (Network Layer) เท่านั้น ในส่วนของชั้นควบคุมการเข้าถึงตัวกลาง (MAC Layer) และชั้นกายภาพ (Physical Layer) จะใช้ตามมาตรฐาน IEEE 802.15.4 ซึ่งทำงานในเรื่องของระดับกำลังสัญญาณ คุณภาพของการเชื่อมต่อ (Link Quality) การควบคุมการเข้าถึง (Access Control) และการรักษาความปลอดภัย เป็นต้น โดยมีโครงสร้างของโปรโตคอล ZigBee เป็นดังนี้



รูปที่ 2.2 แสดงโครงสร้างโปรโตคอลของ ZigBee

ชั้นโปรแกรมประยุกต์ (Application Layer) มีส่วนที่เรียกว่า โครงประกอบของโปรแกรมประยุกต์ (Application Framework) โดยมีออบเจกของอุปกรณ์ ZigBee (ZigBee Device Object หรือ ZDO) ทำหน้าที่ควบคุมการเข้าถึงและการใช้งานในชั้นโปรแกรมประยุกต์ ส่วนในชั้นสนับสนุนโปรแกรมประยุกต์ (Application Support Layer) นั้น ทำหน้าที่ในการสร้างเฟรมของชั้นโปรแกรมประยุกต์และทำหน้าที่ในการรับส่งข้อมูลรวมถึงการจัดการด้านต่างๆที่เกี่ยวกับชั้นโปรแกรมประยุกต์ และส่วนของชั้นเครือข่าย ทำหน้าที่ในการค้นหาเส้นทางของข้อมูลจากต้นทางไปยังปลายทางที่อยู่ภายในเครือข่ายเดียวกันหรือต่างเครือข่ายกัน โครงสร้างชั้นโปรโตคอลของ ZigBee

2.2.1 ขั้นตอนการทำงานของโพรโทคอล ZigBee

- ขั้นตอนการทำงานของ ZigBee coordinator

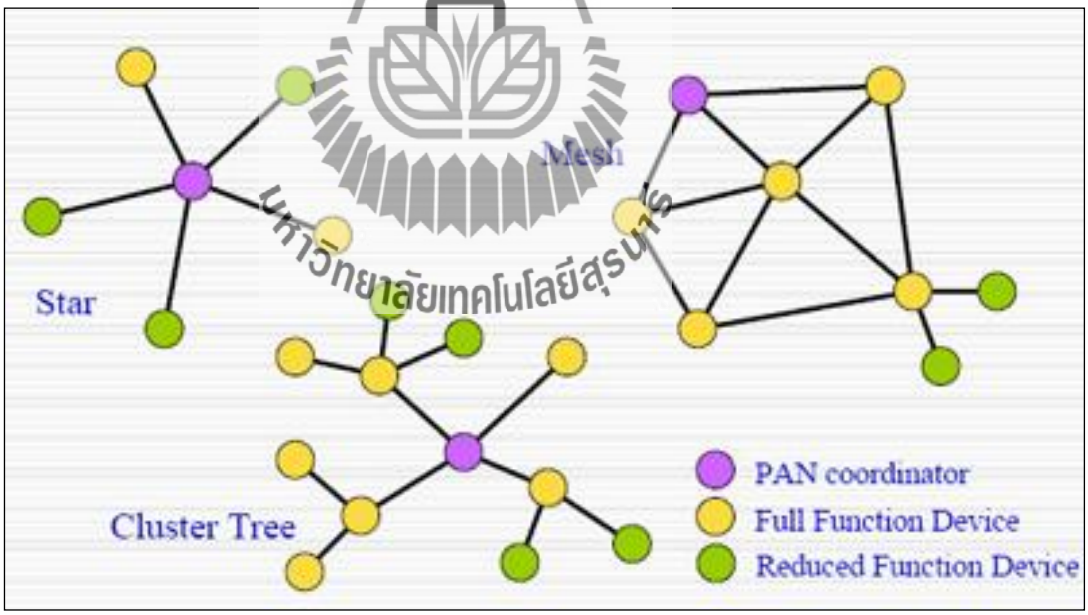
ZigBee coordinator จะเริ่มต้นเครือข่ายโดยการตรวจสอบการใช้ช่องสัญญาณวิทยุภายในบริเวณรอบๆถ้ามีช่องสัญญาณที่ไม่ถูกใช้โดย coordinator ตัวอื่นก็สามารถเริ่มต้นเครือข่ายได้ หลังจากนั้น coordinator ก็จะทำหน้าที่เป็นศูนย์กลางของเครือข่าย รองรับการเข้าร่วมเครือข่ายของZigBee end-device และรองรับการร้องขออื่นๆ ตามมาตรฐานด้วยเช่นกันในโครงงานนี้ coordinator รองรับการเข้าร่วมเครือข่าย การออกจากเครือข่ายและการร้องขอการ Binding เท่านั้น

- ขั้นตอนการทำงานของ ZigBee end-device

ZigBee end-device จะเริ่มต้นการทำงานโดยการร้องขอการเข้าร่วมเครือข่ายไปยัง coordinator ประจำเครือข่ายนั้นๆ โดยการตรวจสอบผ่านช่องสัญญาณต่างๆ ว่า coordinator ใช้ช่องสัญญาณใดอยู่เมื่อเข้าร่วมเครือข่ายแล้ว end-device จึงสามารถทำการร้องขอคำสั่งอื่นๆ ผ่านทาง coordinator ได้ เช่นการส่งข้อความทั่วไป (Message), การร้องขอการ Binding (Binding request), การขอออกจากเครือข่าย

2.3 โครงสร้างของเครือข่าย (ZigBee Topologies)

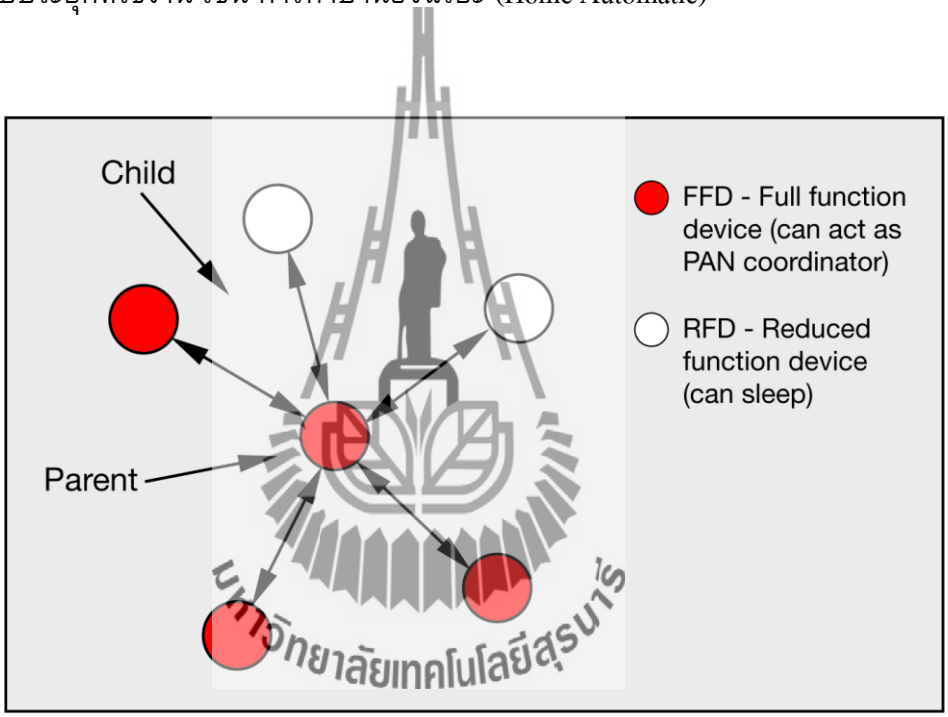
โพรโทคอล ZigBeeถูกออกแบบมาเฉพาะในส่วนของ Application layer, Application support layer และ Network layer เท่านั้น แต่ใช้ MAC layer และ Physical layer ตามมาตรฐาน IEEE 802.15.4 รูปแบบเครือข่ายตามมาตรฐาน IEEE 802.15.4 แบ่งออกเป็น 3 แบบ ได้แก่ โครงสร้างแบบดาว (Star Topology) โครงสร้างแบบเมช (Mesh Topology) และโครงสร้างแบบกลุ่มของต้นไม้ (Cluster Tree) แสดงได้ดังรูป



รูปที่ 2.3 แสดงโครงสร้างของเครือข่ายแบบต่างของ Xbee

2.3.1 โครงสร้างเครือข่ายแบบดาว (Star Topology)

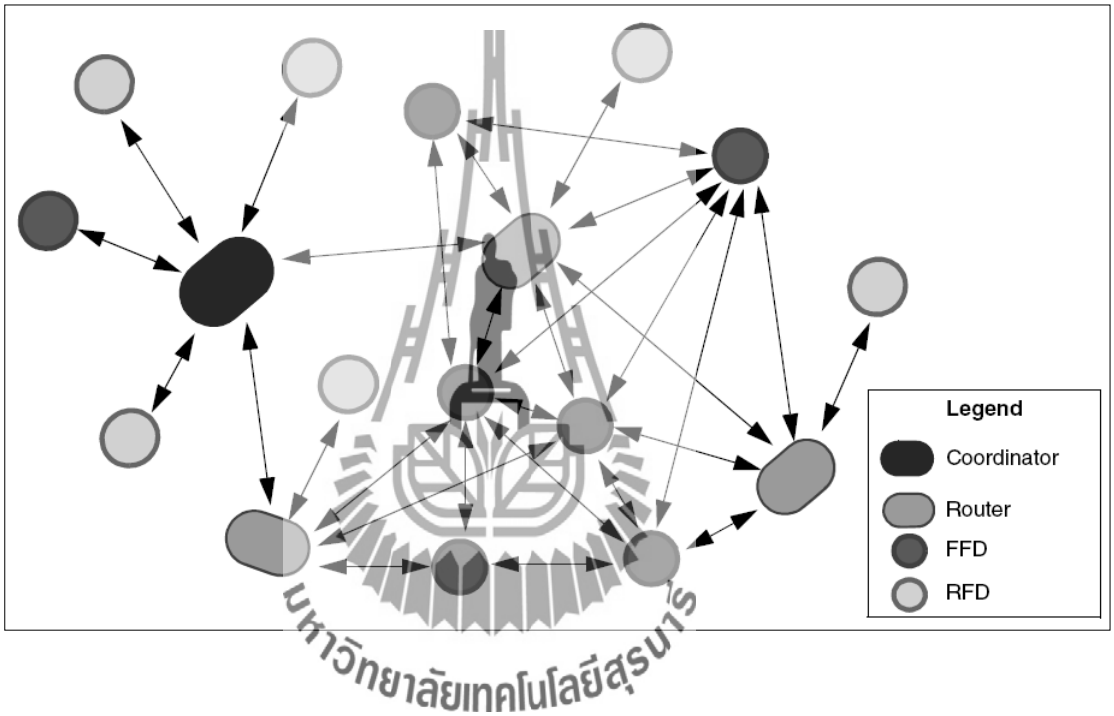
ในโครงสร้างเครือข่ายแบบดาวการสื่อสารจะประกอบด้วยและ End Device ที่ทำหน้าที่เป็นอุปกรณ์ปลายทางหนึ่งตัวขึ้นไป และCoordinator ที่ทำหน้าที่เป็นส่วนควบคุมส่วนกลาง (Single Central Controller) หรือเรียกว่าตัวประสานการเชื่อมต่อของเครือข่ายไร้สายส่วนบุคคล ที่เป็นศูนย์กลางในการส่งข้อมูลของเครือข่าย หนึ่งตัวขึ้นไป โดย End Device ทุกตัวจะติดต่อกับ Coordinator เท่านั้น หาก End Device ต้องการส่งข้อมูลไปยัง End Device ตัวอื่น ก็จะต้องส่งไปยัง Coordinator แล้ว Coordinator จะส่งต่อไปยัง End Device เป้าหมาย ยกตัวอย่างการนำโครงข่ายแบบนี้ไปประยุกต์ใช้งาน เช่น การทำบ้านอัจฉริยะ (Home Automatic)



รูปที่ 2.4 โครงสร้างเครือข่ายแบบ Star

2.3.2 โครงสร้างเครือข่ายแบบเมช (Mesh Topology)

ในโครงสร้างแบบเมชนี้ จะมีหนึ่งตัวประสานการเชื่อมต่อของเครือข่ายไร้สายส่วนบุคคล ทุกอุปกรณ์สามารถสื่อสารกับทุกอุปกรณ์อื่นๆได้ในระยะการส่งข้อมูลที่สามารถส่งถึงดังรูปที่ 2 .7 ซึ่งข้อดีของระบบนี้ คือ สามารถลดการส่งข้อมูลภายใน (Message Latency) และเพิ่มความน่าเชื่อถือ (Reliability) ให้กับระบบ ยกตัวอย่างการนำโครงข่ายแบบนี้ไปประยุกต์ใช้งานเช่น การควบคุมภายในโรงงานและการดูแลติดตาม (Monitoring) เครือข่ายเซ็นเซอร์ไร้สาย

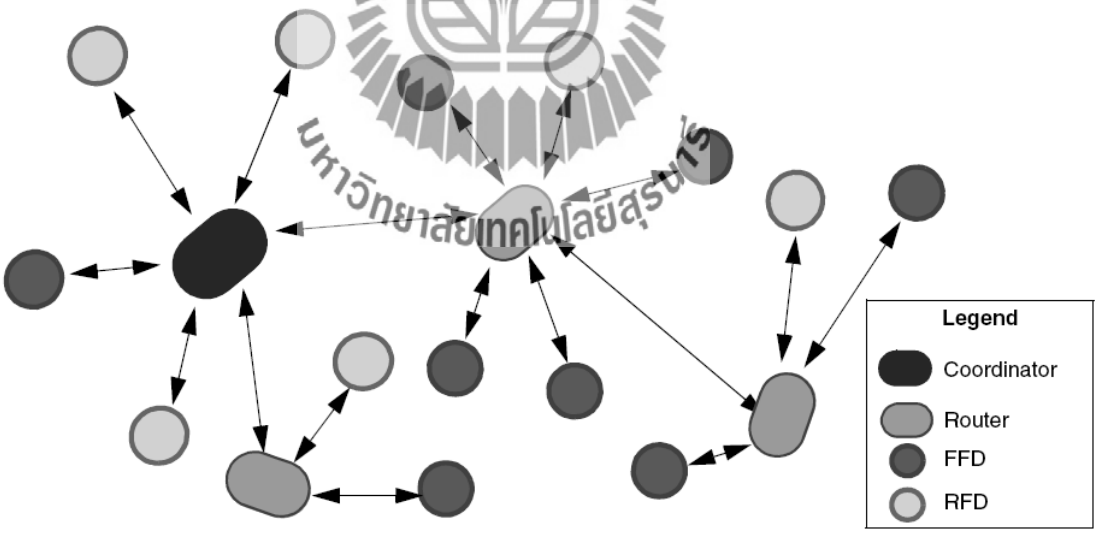


รูปที่ 2.5 โครงสร้างเครือข่ายแบบ Mesh

2.3.3 โครงสร้างเครือข่ายแบบกลุ่มต้นไม้ (Cluster Tree Topology)

เครือข่ายแบบกลุ่มของต้นไม้เป็นรูปแบบพิเศษของเครือข่ายจุดต่อจุด ซึ่งอุปกรณ์ส่วนมากจะเป็น FDD และ RFD ที่เชื่อมต่อกับเครือข่ายแบบกลุ่มของต้นไม้ที่โหนดสุดท้ายของสาขาสำหรับ FDD บางอุปกรณ์นั้นสามารถเป็นตัวประสานการเชื่อมต่อและมีการประสานเวลา (Synchronization) กับอุปกรณ์อื่นๆ และตัวประสานการเชื่อมต่อเหล่านี้เป็นตัวประสานการเชื่อมต่อของเครือข่ายส่วนบุคคล โดยตัวประสานการเชื่อมต่อของเครือข่ายไร้สายส่วนบุคคลจะสร้างกลุ่มชุดแรกด้วยตัวมันเองที่หัวกลุ่ม (Cluster Head หรือ CLH) ด้วยหมายเลขกลุ่ม (Cluster ID หรือ CLD) ที่มีค่าหมายเลขเป็น 0 ดังรูปที่ 2.8

การเลือกหมายเลขของเครือข่ายส่วนบุคคลจะเลือกหมายเลขที่ยังไม่ถูกใช้และกระจาย (Broadcast) เฟรมเบคอน (Beacon) ไปยังอุปกรณ์ข้างเคียง อุปกรณ์ที่ได้รับเบคอนอาจจะร้องขอเข้าร่วมเครือข่ายที่ CLH ถ้าตัวประสานเวลาการเชื่อมต่อของเครือข่ายส่วนบุคคลอนุญาต มันจะบันทึกอุปกรณ์ใหม่นั้นเป็นโหนดลูก (Child Node) และอุปกรณ์ตัวใหม่จะบันทึก CLH เป็นโหนดผู้ปกครอง (Parent Node) ข้อดีของโครงสร้างกลุ่มของต้นไม้ คือ เป็นการครอบคลุมพื้นที่ที่มีการต้องการเข้าร่วมในเครือข่ายมาก



รูปที่ 2.6 โครงสร้างเครือข่ายแบบ Cluster Tree

บทที่ 3
การต่อไมโครคอนโทรลเลอร์กับ Xbee

3.1 การต่อไมโครคอนโทรลเลอร์กับ Xbee เบื้องต้น

Xbee เป็นอุปกรณ์ที่มี Microcontroller และ RF IC อยู่ภายใน ทำหน้าที่เป็น อุปกรณ์ transceiver (อุปกรณ์รับ-ส่งสัญญาณ) แบบ แบบ Half Duplex ย่านความถี่ 2.4 Ghz มีการจัดการโดยใช้พลังงานต่ำ ใช้งานง่าย มี interface ที่ใช้รับและส่งข้อมูลกับ Xbee เป็น UART (TTL) ซึ่งสำหรับทางด้านไมโครคอนโทรลเลอร์ เรานำขาที่ใช้ติดต่อสื่อสาร UART ของ Xbee ต่อเข้ากับ UART ของ ไมโครคอนโทรลเลอร์ ได้โดยตรง

ถ้าใช้ Microcontroller ที่ใช้ไฟเลี้ยง 3.3 Volt สามารถต่อ Xbee เข้ากับ Microcontroller ได้โดยตรง โดยที่ Xbee จะใช้ DI(ขา3) ซึ่งเป็น Rx ของ Xbee , DO(ขา2) ซึ่งเป็น Tx ของ Xbee , Supply Voltage ใช้ขา VCC(ขา1) ต่อกับ 3.3 Volt และ GND(ขา10) ต่อกับ Ground ในเบื้องต้นเพียงเท่านี้ก็สามารถเขียนโปรแกรมต่อได้แล้ว

รูปที่ 3.1 การต่อไมโครคอนโทรลเลอร์กับ Xbee เบื้องต้น

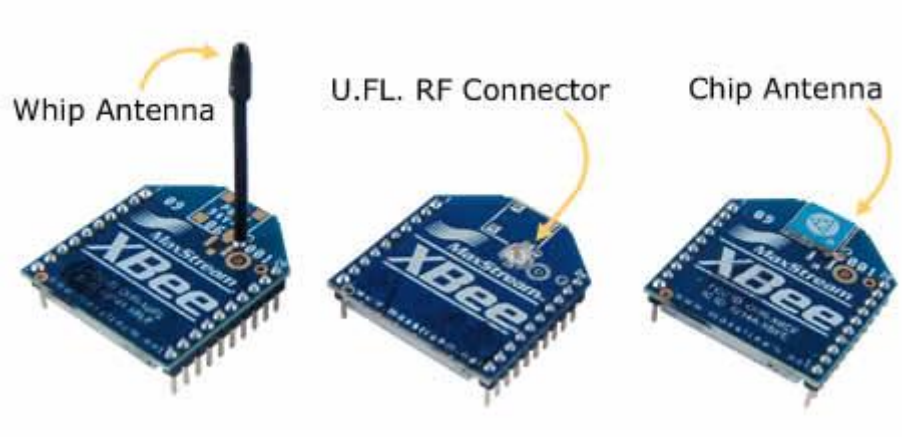
เนื่องจาก Xbee กินไฟเลี้ยง 3.3 V ดังนั้นหากใช้งานร่วมกับไมโครคอนโทรลเลอร์ ที่ใช้ไฟเลี้ยง 5V จะต้องใช้การแก้ปัญหาเรื่องระดับสัญญาณ Logice ระหว่าง 3.3 V / 5V สำหรับการออกแบบใช้ Serial Port UART แบบ Virtual Com Port (RS232 to USB) สามารถใช้ Serial to USB Dongle หรือใช้วงจรแปลงสัญญาณ RS232 to TTL ที่สร้างได้ด้วย IC MAX232

3.2 Xbee ที่เลือกใช้

Xbee มีขายตามท้องตลาดมีมากมายหลายรุ่นหลายแบบโดยแต่ละแบบก็แตกต่างและเหมาะสมกับงานที่ต่างกันไป ซึ่งสามารถแสดงรายละเอียดเบื้องต้นดังนี้

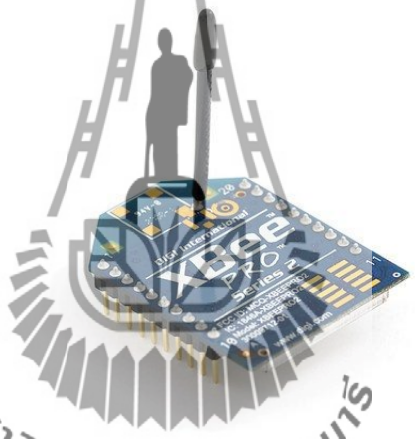
ตารางที่ 3.1 เปรียบเทียบ Xbee แบบต่างๆ

Feature	Series1	Series2	Series1 Pro	Series2 Pro
Power Input	3.3V @ 50mA	3.3V @ 40mA	3.3V @ 215mA	3.3V @ 295mA
Max data rate (Air)	250kbps	250kbps	250kbps	250kbps
Power Output	1mW output (+0dBm)	2mW output (+3dBm)	60mW output (+18dBm)	50mW output (+17dBm)
Distance	300ft (100m) range	400ft (120m) range	1 mile (1500m) range	1 mile (1600m) range
Antenna	Wire,Chip,UFL,SMA	Wire,Chip,UFL,SMA	Wire,Chip,UFL,SMA	Wire,Chip,UFL,SMA
Peripheral	6 10-bit ADC input pins 8 digital IO pins	6 10-bit ADC input pins 8 digital IO pins	6 10-bit ADC input pins 8 digital IO pins	6 10-bit ADC input pins 8 digital IO pins
Upgrade Firmware	Local	over-air configuration(ZB)	Local	over-air configuration(ZB)
Network	Point to point and multi-point networks	Point to point / multi-point / Mesh Network	Point to point and multi-point networks	Point to point / multi-point / Mesh Network



รูปที่ 3.2 รูปแบบของสายอากาศ Xbee

ในโครงงานนี้ผู้จัดทำได้เลือกใช้ Xbee Pro Serie2 รุ่น XBP24-BWIT-004

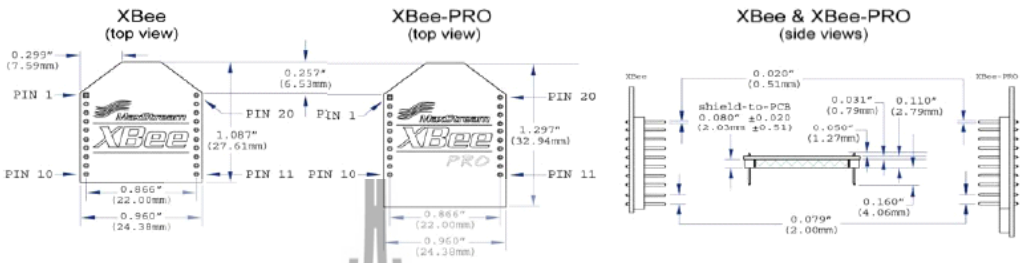


รูปที่ 3.3 Xbee Pro Serie2 รุ่น XBP24-BWIT-004

Features:

- 3.3V @ 295mA
- 250kbps Max data rate
- 50mW output (+17dBm)
- 1 mile (1600m) range
- Built-in antenna
- Fully FCC certified
- 6 10-bit ADC input pins

- 8 digital IO pins
- 128-bit encryption
- Local or over-air configuration
- AT or API command set



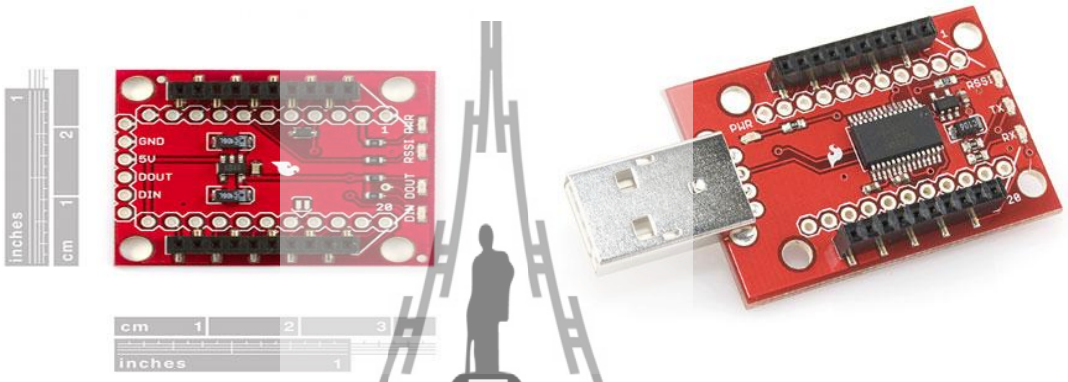
รูปที่ 3.4 โครงสร้างของ Xbee

ตารางที่ 3.2 แสดงรายละเอียด PIN ของ Xbee

Pin #	Name	Direction	Description
1	VCC		Power supply
2	DOUT	Output	UART Data Out
3	DIN / $\overline{\text{CONFIG}}$	Input	UART Data In
4	DIO12	Either	Digital I/O 12
5	$\overline{\text{RESET}}$	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI / DIO10	Either	PWM Output 0 / RX Signal Strength Indicator / Digital IO
7	PWM / DIO11	Either	Digital I/O 11
8	[reserved]	-	Do not connect
9	$\overline{\text{DTR}}$ / SLEEP_RQ/ DIO8	Either	Pin Sleep Control Line or Digital IO 8
10	GND	-	Ground
11	DIO4	Either	Digital I/O 4
12	$\overline{\text{CTS}}$ / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / $\overline{\text{SLEEP}}$ / DIO9	Output	Module Status Indicator or Digital I/O 9
14	[reserved]	-	Do not connect
15	Associate / DIO5	Either	Associated Indicator, Digital I/O 5
16	$\overline{\text{RTS}}$ / DIO6	Either	Request-to-Send Flow Control, Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0 / Commissioning Button	Either	Analog Input 0, Digital IO 0, or Commissioning Button

เหตุผลที่เลือกใช้ มีการพิจารณาจากเหตุผล 3 ด้านคือ

- เลือกใช้รุ่น PRO เนื่องจากกำลังส่งจะอยู่ในช่วง 50-60 mW โดยจะมีระยะประมาณ 1500 เมตร ซึ่งเหมาะสมกับส่งข้อมูลไปยังรถไฟในขณะที่อยู่ไกล
- เลือกสายอากาศแบบ Wire Antenna เพราะ ระยะและความเสถียร จะได้ตาม Spec
- ในโครงงานนี้ได้ใช้อุปกรณ์เสริมคือ XBee Explorer Regulated เพื่อแปลงระดับแรงดัน 5 V to 3 V และ XBee Explorer Dongle

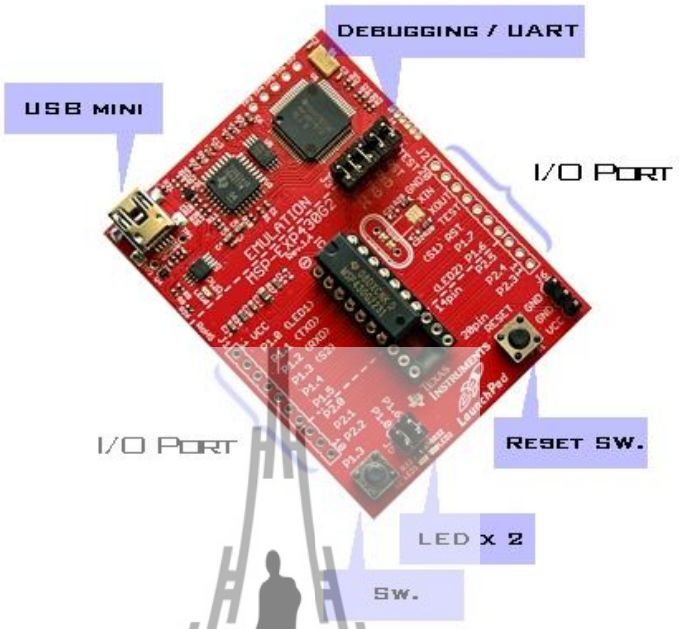


รูปที่ 3.5 XBee Explorer Regulated และ XBee Explorer Dongle

3.3 ไมโครคอนโทรลเลอร์ที่เลือกใช้

ในโครงงานนี้ผู้จัดทำได้ใช้ไมโครคอนโทรลเลอร์ 2 ตัว คือที่ฝั่งรับข้อมูลและฝั่งส่งข้อมูล โดยใช้ไมโครคอนโทรลเลอร์เบอร์ P89V51RD2 และไมโครคอนโทรลเลอร์เบอร์ MSP430 G2231 ตามลำดับ ด้วยเหตุผลดังต่อไปนี้

3.3.1 ไมโครคอนโทรลเลอร์ที่ฝังส่งข้อมูล MSP430 G2231



รูปที่ 3.6 MSP430G2 LunchPAD

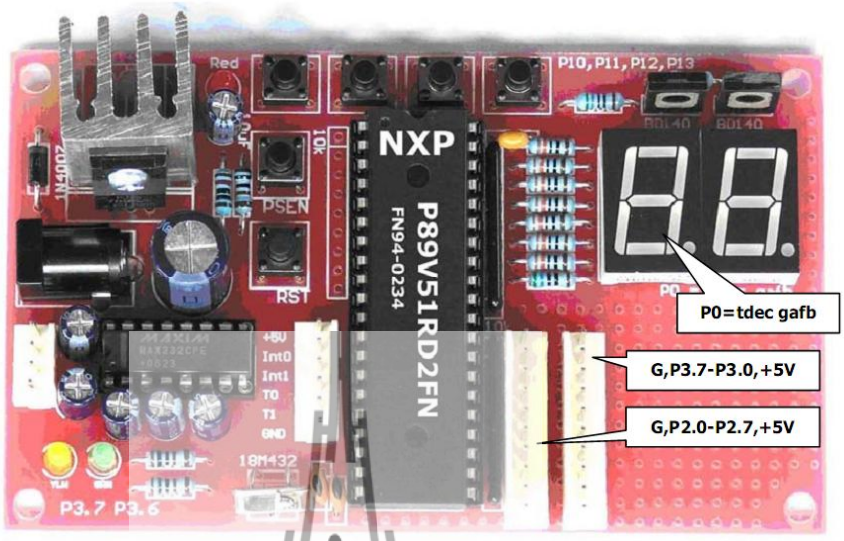
MSP430 LaunchPad เป็นบอร์ดทดลอง/พัฒนาของทาง Texas ซึ่งมาพร้อมกับวงจรดีบักผ่านพอร์ต USB แบบ JTAG 2 สาย ที่เรียกว่า Spy Bi-Wire ซึ่งเป็นเทคโนโลยีใหม่ในการดีบักและโปรแกรมไมโครคอนโทรลเลอร์ในวงจรด้วยสายสัญญาณเพียง 2 เส้น

ในส่วนของตัวไมโครคอนโทรลเลอร์ในชุด MSP430 LaunchPad นั้นมีชิปในอนุกรม Value line ของ MSP430 คือเบอร์ MSP430G2231 ซึ่งมีความจุของน่วยความจำแบบแฟลช 2 กิโลไบต์, แรม 128 ไบต์, 10 GPIO, ไทเมอร์ 16 บิต, WDT, BOR, I2C/SPI, Internal Temp Sensor, A/D 8 ช่อง ความละเอียด 10 บิต

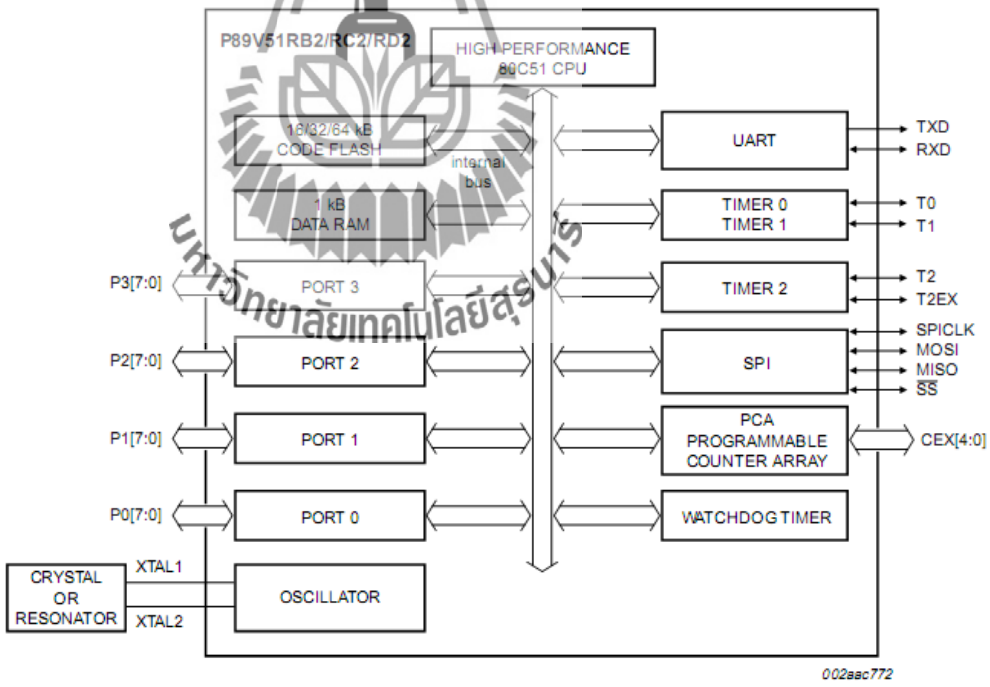
เหตุผลที่เลือกใช้

- ราคาถูก
- ประหยัดพลังงานมาก
- จ่ายแรงดัน 3.3v สามารถต่อกับ Xbee ได้โดยไม่ต้องมีการแปลงระดับแรงดัน
- สามารถรับส่งข้อมูลผ่านทาง USB ทำให้ง่ายต่อการทดลอง
- ออกแบบมาเพื่อให้ใช้งานขาพอร์ตของไมโครคอนโทรลเลอร์ได้อย่างเต็มที่ มีจุดต่อที่ใช้งานสะดวก

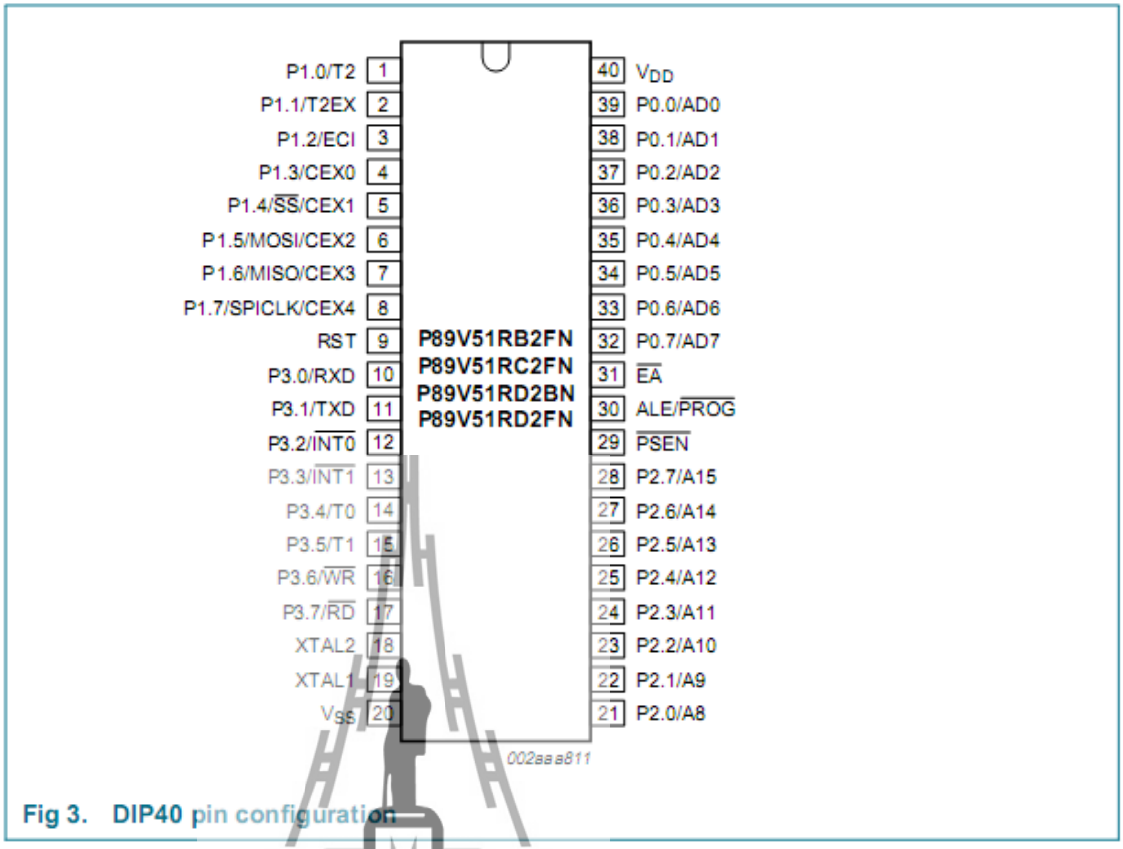
3.3.2 ไมโครคอนโทรลเลอร์ที่ฝังรับข้อมูล P89V51RD2



รูปที่ 3.7 บอร์ด MCS51 รุ่น SUT-V5x: MCU-P89V51RD2



รูปที่ 3.8 โครงสร้างภายใน MCU-P89V51RD2



รูปที่ 3.9 PIN ต่างๆของ MCU-P89V51RD2

ใช้บอร์ด MCS51 รุ่น “SUT-V5x: MCU-P89V51RD2” ที่ออกแบบและสร้างโดยทีมงานห้องปฏิบัติการไมโครคอนโทรลเลอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี โดยรวมประกอบด้วยส่วนต่างๆดังนี้

พอร์ตสำหรับสื่อสารอนุกรม RS232 เพื่อโหลดโปรแกรมและแสดงผลการทำงานของ MCU

พอร์ต P3 และ P2 สำหรับต่อใช้งานเป็น IO ทั่วไป

สวิตซ์ P1.0, P1.1,P1.2 และ P1.3 เพื่อป้อนลอจิกอินพุตให้ MCU

ส่วนแสดงผล LED ต่อกับ P3.6 และ P3.7 แบบแอกทีฟโลว์

สวิตซ์ Reset และ PSEN เพื่อเตรียมโหลดโปรแกรมให้กับ MCU

เหตุผลที่เลือกใช้

- มี IO port มากเหมาะสำหรับนำไปต่อยอดในอนาคต
- สื่อสารทางพอร์ตอนุกรม RS232 โดยมีการแปลงสัญญาณ RS232 to TTL ไว้แล้ว

- เป็นไมโครคอนโทรลเลอร์ที่มีความชำนาญ
- ทนความร้อนได้สูง

3.3.2.1 การโหลดโปรแกรม P89V51RD2 ด้วย Flash Magic 5

Flash Magic เป็นโปรแกรมสำหรับใช้ Download HEX File ให้กับหน่วยความจำโปรแกรมภายในตัวของไมโครคอนโทรลเลอร์ ตระกล MCS51 ของ Philips ในกลุ่มเบอร์ที่รองรับการ Download ข้อมลด้วยวิธีการแบบ ISP Download (In System Programming Download) ซึ่งได้รับการพัฒนาขึ้นโดย Embedded System Academy ซึ่งไดรับการสนับสนุนจาก Philips โดยผู้ใช้สามารถทำาการ Download โปรแกรมตัวนี้
มาใช้งานได้ฟรีโดยไม่เสียค่าใช้จ่ายใดๆ จากเว็ปไซต์ของ WWW.ESACADEMY.COM โดยในปัจจุบัน จะสนับสนุนการใช้งานร่วมกับไมโครคอนโทรลเลอร์ของ Philips ไดหลายเบอร์ เช่น

- 89C51RX2 เช่น 89C51RA2XX,89C51RB2XX,89C51RC2XX,89C51RD2XX
- 89C60X2,89C61X2
- 89C51RX2H เช่น 89C51RB2HXX,89C51RC2HXX,89C51RD2HXX
- 89C66X เช่น 89C660,89C662,89C664,89C668,89C6699
- 89C51RX+ เช่น 89C51RB+,89C51RC+,89C51RD+
- XA-G39,XA-G49
- 89LPC9XX เช่น 89LPC901,02,03,04,07,08,12,13,14,20,21,22,30,31,32,35
- 89LV51RD2
- 89V51RD2

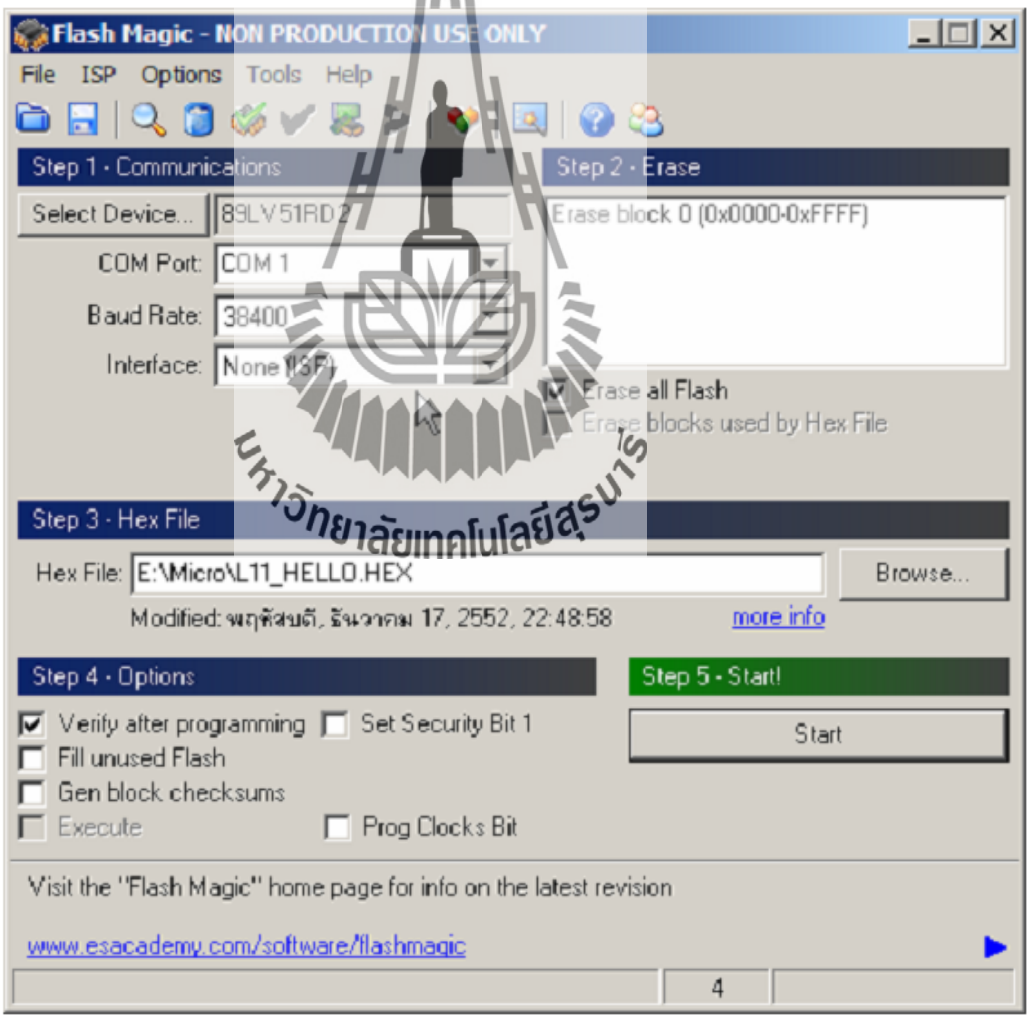
โดยการทำางานของโปรแกรม Flash Magic นั้นจะสนับสนุนการ Download โปรแกรมแบบ HEX File จากคอมพิวเตอร์ PC ผานทางพอร์ตสื่อสารอนุกรมแบบ RS232 ไปให้กับหน่วยความจำโปรแกรม Flash Memory ภายในตัว CPU ตระกล MCS51 เฉพาะที่ผลิตขนโดย Philips เท่านั้น โดยการ Download สามารถ ทำได้ทั้งแบบ Manual และ อัตโนมัติ (Auto Download) ซึ่งในกรณีของการ Download แบบอัตโนมัตินั้น จะต้องออกแบบวงจรสำหรับเลือกโหมดการทำงานของ CPU โดยใช้สัญญาณ RTS และ DTR ของพอร์ตสื่อสารอนุกรม RS232 ได้ด้วย โดยจะใชสัญญาณ DTR สาหรบทำหน้าที่ควบคมการทำงานของสัญญาณ

Reset ของ CPU และใช้สัญญาณ RTS สำหรับกำหนดสภาวะโลจิกให้กับขาสัญญาณ PSEN ของ CPU

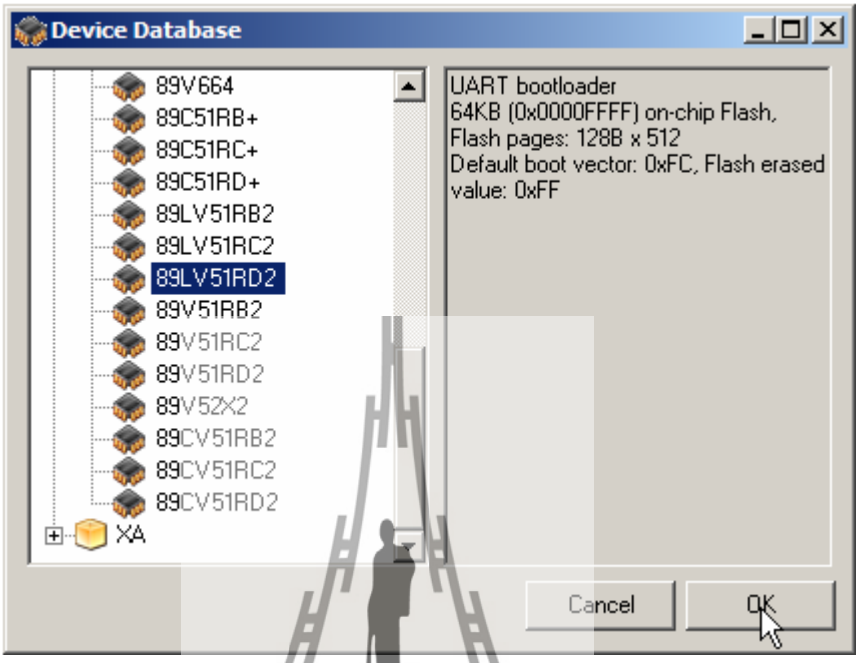
การโหลดโปรแกรมเข้า MCU ด้วยโปรแกรม Flash Magic

การโหลดโปรแกรมเข้า MCU เพื่อทำการรันนั้นจะมีข้อแตกต่างกันไปตามแต่ว่าใช้บอร์ดอะไรและใช้ MCU เบอร์ไหน สำหรบบอร์ด SUT-V5x นี้ใช้โปรแกรมช่วยโหลดชื่อ Flash Magic V5.4 และมีขั้นตอนการทำงานดังนี้

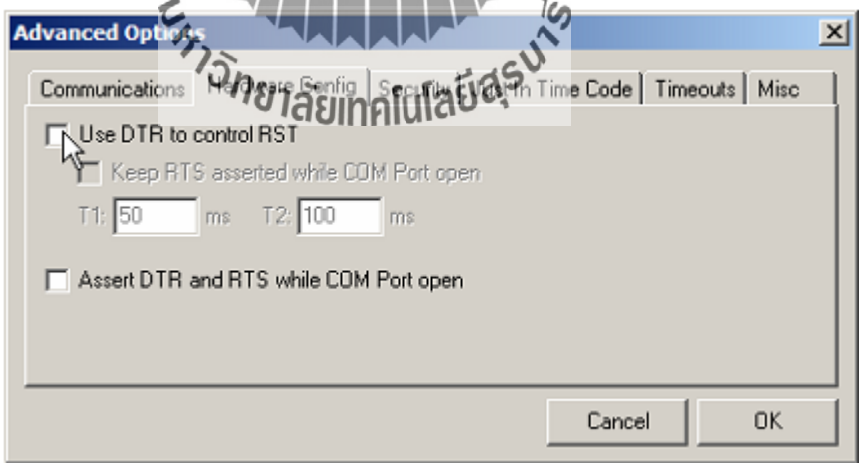
1. เขียนชุดคำสั่งดวยโปรแกรม Assembly หรอ C หรอ Basic คอมไพล์เพื่อให้ได้ Hex File
2. เรียกโปรแกรม Flash Magic V5.4



3. เลือก Select Device = 89V51RD2



4. ไปที่Option->Advance ที่หัวข้อ Hardware Config->Disable Use DTR to Control RST->Ok



5. ตั้งค่า พารามิเตอร์ต่างๆ

- COM Port COM1
- Baud Rate 9600

- Interface ISP
- Select Eras All
- Select Verify after programming

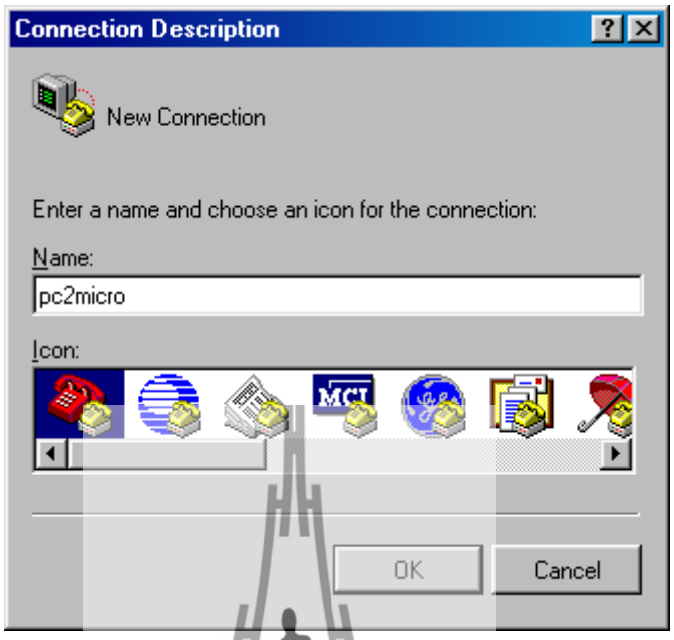
6. Step3: Hex File เลือกไฟล์ที่ต้องการให้ทำงาน ในที่นี้คือ E:\Micro\Hello.Hex
7. ที่บอร์ดไมโครกดสวิตซ์ Reset ค้างไว้
8. ที่ Flash Magic กดปุ่ม Start รอจนกว่าขึ้น Massage Box “Reset the device into ISP mode now” ปล่อยปุ่ม Reset ที่บอร์ดไมโครคอนโทรลเลอร์

9. ที่บอร์ดไมโครกดสวิตซ์ Reset อีกครั้งเพื่อส่ง RUN ให้โปรแกรมทำงาน
10. หากมีการแก้ไขโปรแกรมเมื่อคอมไพล์แล้ว ขั้นตอนการโหลดให้ทำซ้ำขั้นตอนที่ 6 เป็นต้นมา

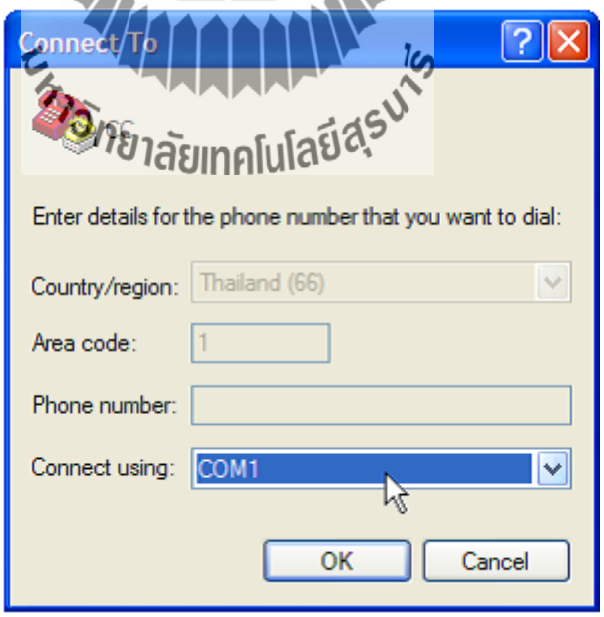
หมายเหตุ
ในกรณีที่ใช้ Hex File ที่ได้จากการแปลค่าส่งของ SXAS1 นั้น จะมีบรรทัดวางเกิดขึ้นใน Hex File ด้วย 1 บรรทัด ซึ่งจะไม่สามารถใช้กับโปรแกรมของ Flash Magic ได้ทันที แต่จะตองทำการแก้ไข Hex File นั้น โดยการเขาไปตัดบรรทัดวางในสวนเริ่มต้นของ Hex File ออกแล้วสั่งบันทึก Hex File นั้นใหม่เสียก่อน จึงจะสามารถ Download Hex File นั้น เพื่อใช้งานกับโปรแกรมของ Flash Magic ได้ตามปรกติ

3.4 การใช้โปรแกรม Hyperterminal สำหรับติดต่อกับไมโครคอลโทรลเลอร์

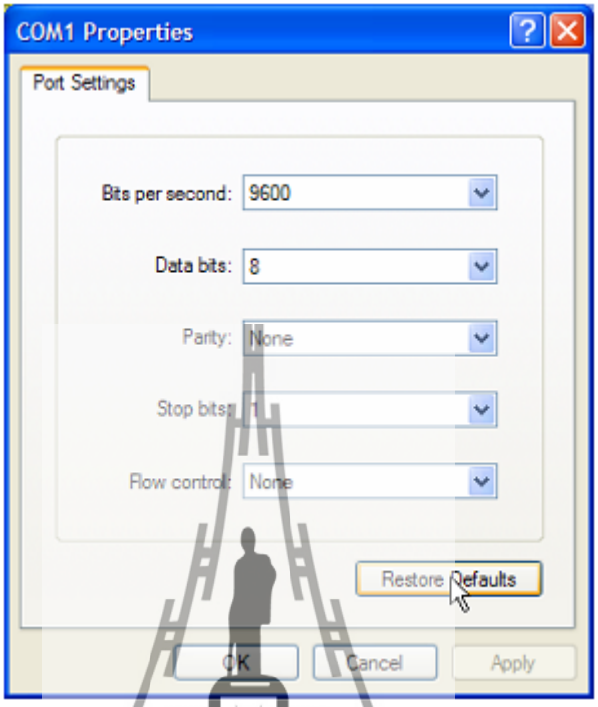
1. เราสามารถเรียกโปรแกรม Hyperterminal ได้โดย เริ่มจากคลิก
   Start –> Program –> Accessories –> Communications –> HyperTerminal
2. ที่หน้าต่าง Connection Description ในช่อง Name ให้ใส่ชื่อที่เราต้องการ และช่อง Icon ก็เลือก icon ที่เราชอบ



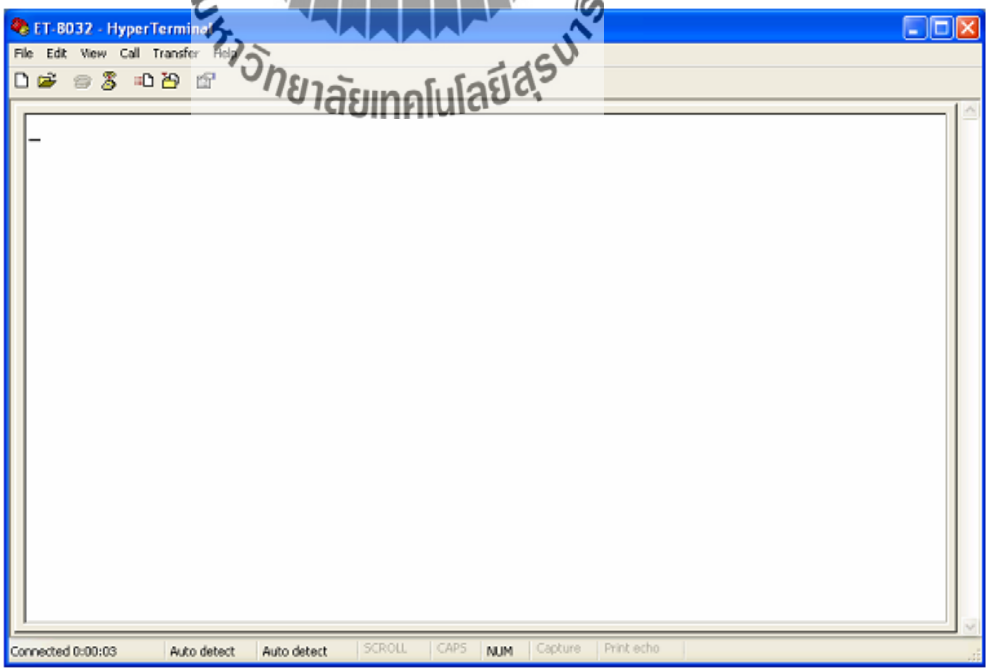
3. หน้าต่าง Connect to ในช่อง Connect using ให้เลือก comport ที่เราใช้ติดต่อกับ microcontroller ช่องอื่นๆ ปล่อยไว้ตามนั้น



6. เลือก Boud rate ให้ตรงกับ Boud rate ของ microcontroller



5. คลิก OK ก็จะขึ้นหน้าจอใช้งานตามรูป และ บอร์ดจะ connect กับ comport ให้เราโดยอัตโนมัติ ลองดูตามรูปครับ

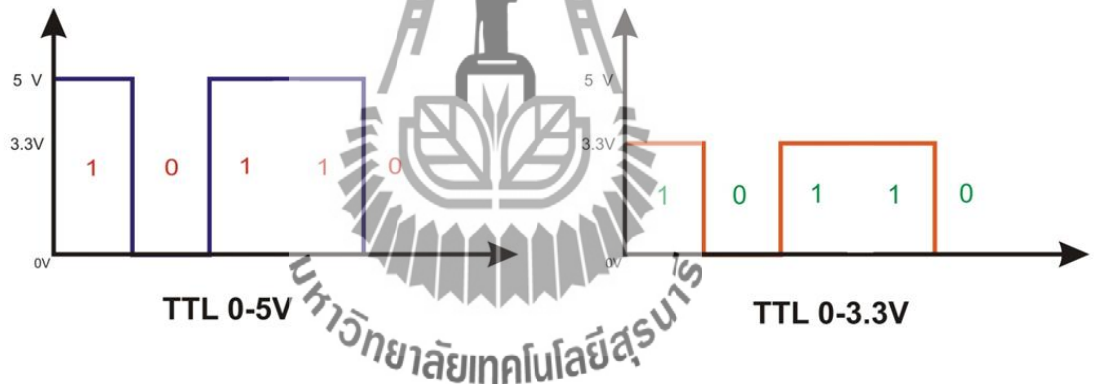


3.5 UART,TTL, RS232,MAX232,MAX3232 คืออะไร

ทั้งหมดนี้ล้วนแต่เป็นรูปแบบการสื่อสารข้อมูล และสื่อสารข้อมูลเหล่านี้ เป็นที่นิยมใช้กันอย่างมาก และ มีการใช้เชื่อมต่อ เพื่อสื่อสารกับอุปกรณ์ต่าง ๆ มากมาย เช่น การสื่อสารกับคอมพิวเตอร์ , Xbee , RFID , GPS , GSM Module , RF Module ฯลฯ จึงต้องมีการอธิบาย เพื่อให้เข้าใจและนำไปใช้ได้อย่างถูกต้อง

TTL (Transistor-Transistor Logic)

TTL เป็นระดับแรงดันที่ถูกกำหนดขึ้นในยุคแรกๆเพื่อใช้ระหว่าง Transistor กับ Transistor ภายในวงจรรวม(IC) ดังนั้น TTL จะใช้ระดับแรงดัน อยู่ที่ 0 – 5 V แต่ในปัจจุบันมีอุปกรณ์หลายเบอร์ที่ทำงานในช่วง 0 – 3.3 V ซึ่งผู้ใช้ควรตรวจสอบจาก Datasheet ของอุปกรณ์ที่ใช้เสียก่อนว่าเป็นระดับแรงดันแบบใด เพราะหากใช้ผิดประเภทจะทำให้อุปกรณ์เสียหาย

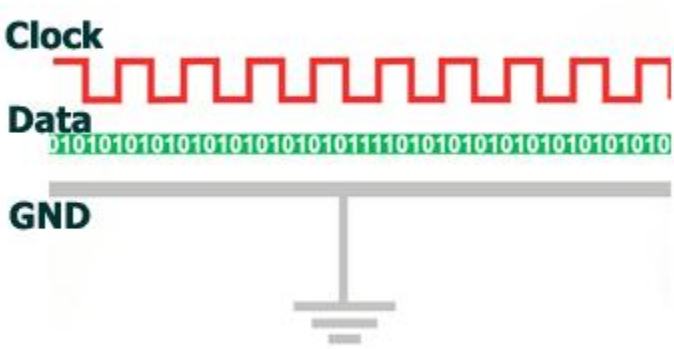


รูปที่ 3.10 ระดับสัญญาณ TTL 5 V และ 3 V

UART

ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter หมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส (Asynchronous) ซึ่งเป็นส่วนหนึ่งในการสื่อสารอนุกรม แบบ Asynchronous แท้จริงแล้วการสื่อสารแบบอนุกรมจะแบ่งเป็น 2 แบบ คือ

1) การสื่อสารอนุกรมแบบ Synchronize เป็นรูปแบบที่ใช้วิธีส่งข้อมูล โดยใช้สัญญาณ Clock มาเป็นตัวกำหนดจังหวะ การรับส่งข้อมูล การส่งข้อมูลแบบนี้ เป็นการรับส่งที่ค่อนค่างมีคุณภาพ มีโอกาสที่ข้อมูลจะสูญหายระหว่างการส่งน้อย แต่มีข้อเสียคือ เป็นการสื่อสารแบบ Half Duplex ไม่สามารถรับและส่งข้อมูลในเวลาเดียวกัน

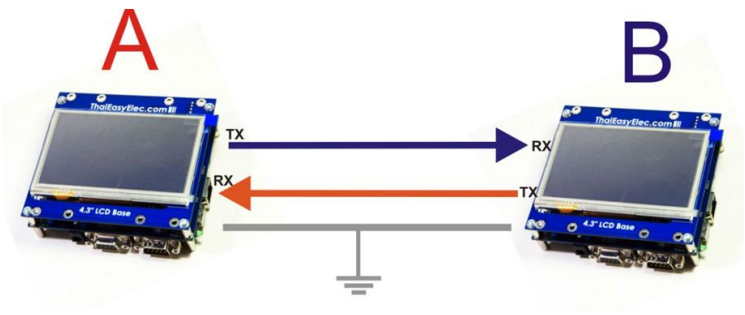


รูปที่ 3.11 การสื่อสารอนุกรมแบบ Synchronize

2) การสื่อสารอนุกรมแบบ Asynchronous เป็นการส่งข้อมูลที่ไม่ต้องใช้สัญญาณ Clock มาเป็นตัวกำหนดจังหวะการรับส่งข้อมูลแต่ ใช้วิธีกำหนด รูปแบบ Format การรับส่งข้อมูลขึ้นมาแทน และอาศัยการกำหนด ความเร็วของการรับ และ ส่ง ที่เท่ากันทั้งฝั่งรับและฝั่งส่ง ข้อดีของการใช้ Asynchronous คือสามารถสื่อสารแบบ Full Duplex รับ และ ส่งได้ในเวลาเดียวกัน แต่ Asynchronous มีโอกาสที่ข้อมูลจะสูญหายขณะรับส่งข้อมูล หรือ รับส่งข้อมูลผิดพลาดได้มากกว่าแบบ Synchronize สรุปกล่าวคือ UART (Universal Asynchronous Receiver Transmitter) หมายถึงรูปแบบการส่งข้อมูล ที่ถูกกำหนดขึ้นมาเพื่อใช้รับส่งข้อมูลแบบ Asynchronous โดยมีรูปแบบดังรูป

รูปที่ 3.12 การสื่อสารอนุกรมแบบ Asynchronize

เริ่มต้นจาก Start Bit เป็น Logic 0 จากนั้นจะตามด้วย Data ที่เราส่ง แล้วจะถูกปิดด้วย STOP Bit เป็น Logic 1



รูปที่ 3.13 แนวคิดวิธีเชื่อมต่อระหว่างอุปกรณ์

จากรูปแสดงถึงการเชื่อมต่อ ระหว่างบอร์ด SUN7 เพื่อส่ง Data หากันระหว่างบอร์ดด้วย

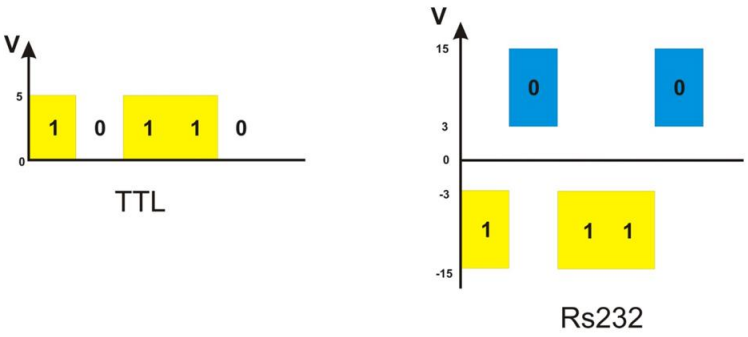
1. A ส่งข้อมูล ออกไปทางขา Tx ไปยัง B ซึ่งเป็นฝั่งรับ เพราะฉะนั้น ต้องต่อสายสัญญาณจากขา Tx ของ A ไปยังขา Rx ของ B
2. B ส่งข้อมูล ออกไปทางขา Tx ไปยัง A ซึ่งเป็นฝั่งรับ เพราะฉะนั้น ต้องต่อสายสัญญาณจากขา Tx ของ B ไปยังขา Rx ของ A
3. ต้องต่อ GND ของทั้ง A และ B ร่วมกันเพื่อทำให้ระดับแรงดันของทั้ง 2 บอร์ดมีจุดอ้างอิงเดียวกัน

RS232 (RecommendedStandard232)

RS232 คือ มาตรฐานการเชื่อมต่อข้อมูลแบบ Serial ใช้เพื่อเพิ่มระยะทางในการส่งข้อมูล แบบ Serial ให้สามารถส่งได้ระยะทางที่มากขึ้น โดยมีการเปลี่ยนระดับแรงดัน ของ Logic จากเดิมที่จะอยู่ในช่วง 0-5 V หรือ 0-3.3 V เป็นช่วง -15 ถึง 15 V โดยมีรายละเอียดดังนี้

Logic 0 ของ RS232 จะอยู่ในช่วง 3 ถึง 15V

Logic 1 ของ RS232 จะอยู่ในช่วง -3 ถึง -15V



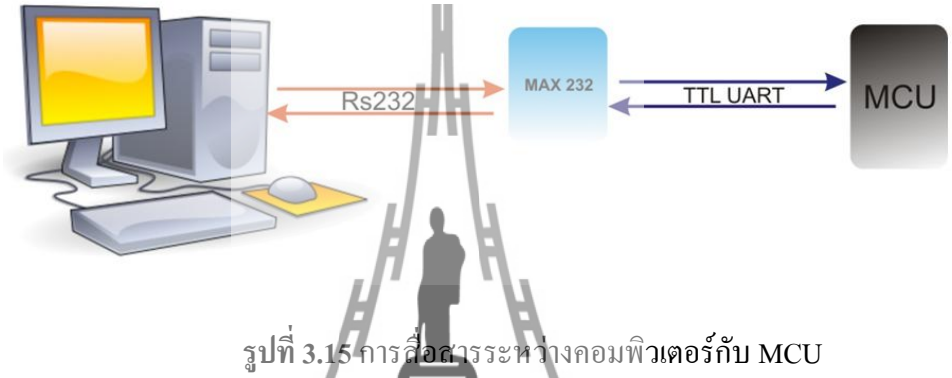
รูปที่ 3.14 ระดับสัญญาณ TTL และ RS232

จากรูปจะเห็นได้อย่างชัดเจนครับ ว่าทั้ง 2 อย่าง ส่ง Data เหมือนกัน แต่ระดับแรงดันที่ใช้ต่างกันมาก หากอุปกรณ์เป็น TTL แล้ว ไปต่อกับ RS232 ก็จะเกิดความเสียหายตามมาได้

การทำให้สัญญาณ TTL สามารถรับส่งข่อมูลกับ RS232

IC MAX 232

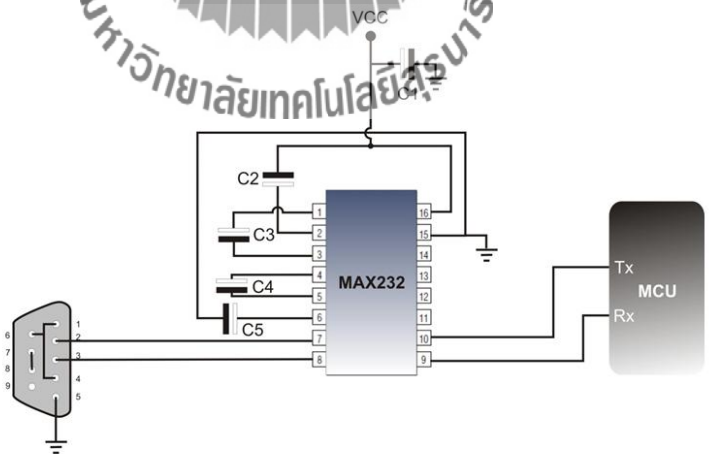
เป็น IC ที่ใช้เปลี่ยน TTL เป็น RS232 ในฝั่งส่ง และ เปลี่ยน RS232 เป็น TTL ในฝั่งรับ ดังรูป



รูปที่ 3.15 การสื่อสารระหว่างคอมพิวเตอร์กับ MCU

วิธีต่อใช้งาน MAX 232

วงจรนี้ใช้กับ TTL 0-5V เป็น RS 232



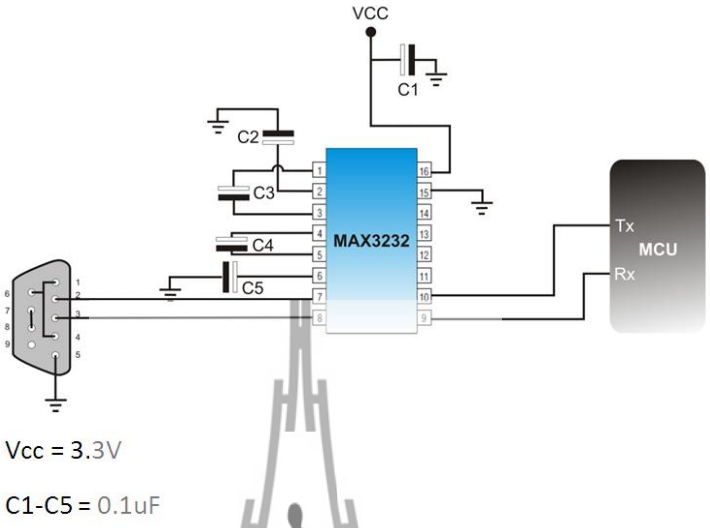
Vcc = 5V

C1 – C5 = 10 uF

รูปที่ 3.16 วงจรแปลงระดับสัญญาณ TTL 0 - 5V เป็น RS 232

วิธีต่อใช้งาน MAX 3232

วงจรนี้ใช้กับ TTL 0-3.3V เป็น RS 232



รูปที่ 3.17 วงจรแปลงระดับสัญญาณ TTL 0 - 3.3V เป็น RS 232

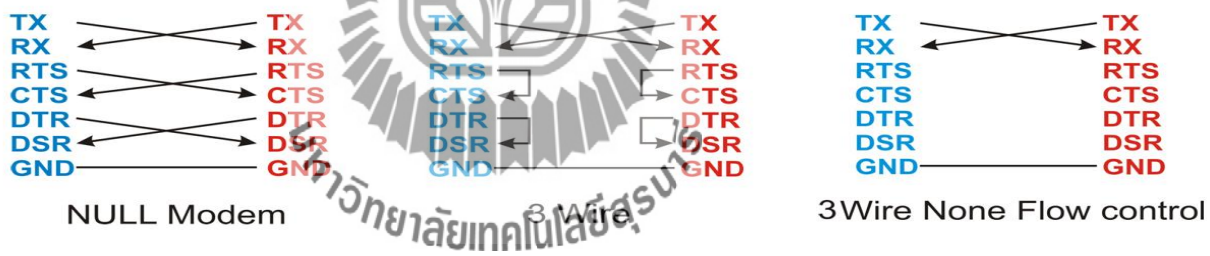
รูปที่ 3.18 ขาของ คอนเน็กเตอร์ DB9

ตารางที่ 3.3 ขาของ คอนเน็กเตอร์ DB9

DB-9M	Function	Abbreviation
Pin #1	Data Carrier Detect	CD
Pin #2	Receive Data	RD or RX or RXD
Pin #3	Transmitted Data	TD or TX or TXD
Pin #4	Data Terminal Ready	DTR
Pin #5	Signal Ground	GND
Pin #6	Data Set Ready	DSR
Pin #7	Requests To Send	RTS
Pin #8	Clear To Send	CTS
Pin #9	Ring Indicator	RI

การเชื่อมต่อสาย DB9

การเชื่อมต่อสาย DB9 โดยทั่วไปแบ่งได้เป็น 3 แบบดังรูป



รูปที่ 3.19 การเชื่อมต่อสาย DB9

TX = เป็นขาส่งข้อมูล

RX = เป็นขารับข้อมูล

RTS = เป็นขาที่ส่งสถานะไปยังตัวรับ ว่าต้องการส่งข้อมูล เมื่อต้องการส่งข้อมูล จะ ON จน กระทั้งส่ง Data ออกทางขา TX จนเสร็จจึงจะ OFF

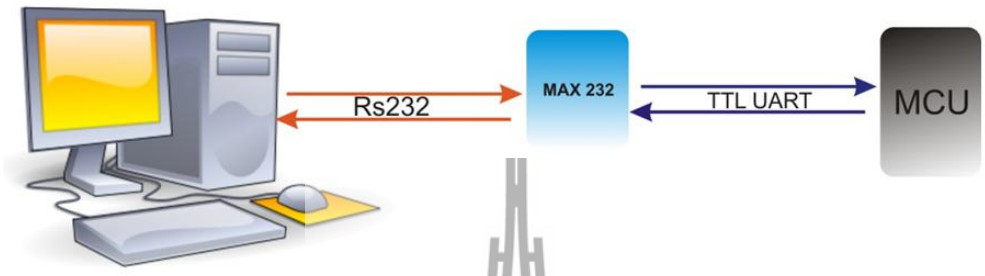
CTS = เป็นขาที่รอรับสถานะ จาก RTS ของอุปกรณ์ที่ต่ออยู่ด้วย

DTR = เป็นขาที่แสดงสถานะว่า Port นั้นเปิดอยู่หรือไม่

DSR = เป็นขาที่ใช้ตรวจเช็ค สถานะ DTR ของอุปกรณ์ที่เชื่อมต่ออยู่ด้วย

GND = Signal Ground

ตัวอย่างการนำไปใช้



รูปที่ 3.20 การสื่อสารระหว่างคอมพิวเตอร์กับ MCU

จากรูปเป็นตัวอย่างการเชื่อมต่อ Microcontroller (MCU) กับ PC เนื่องจาก Serial Port ของ PC เป็นมาตรฐาน RS232 แต่ MCU เป็น TTL จึงต้องใช้ MAX232 ปรับระดับแรงดันให้อยู่ในระดับเดียวกัน

รูปที่ 3.20 การสื่อสารระหว่าง MCU

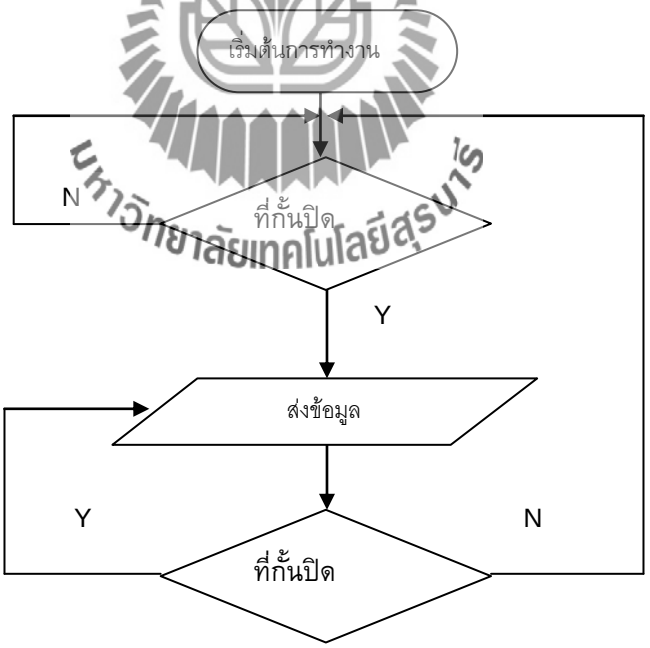
จากรูปเป็นการติดต่อกัน ระหว่าง MCU 2 ตัว สามารถต่อ Rx -> Tx , Tx -> Rx กันโดยตรงได้เลย เนื่องจาก ทั้ง 2 ตัวมีระดับแรงดันเป็น TTL เหมือนกัน

บทที่ 4
วิธีการดำเนินโครงงาน

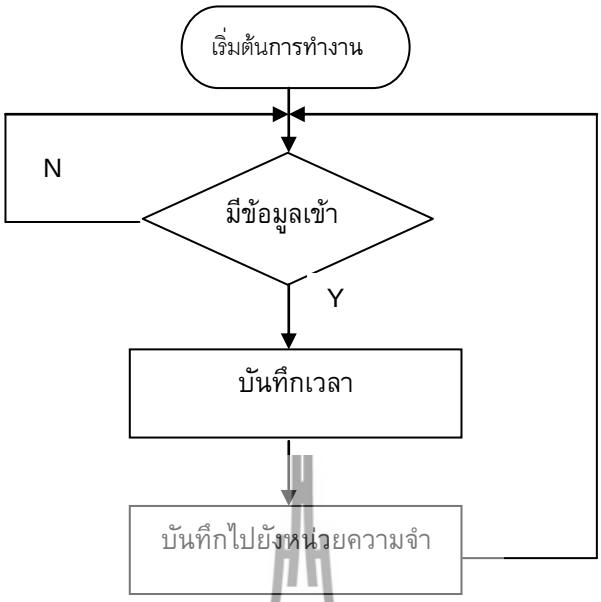
4.1 ลักษณะการทำงานโดยรวมของระบบ

ในโครงงานนี้จะแบ่งการทำงานออกเป็น 2 ส่วน คือฝั่งส่งและฝั่งรับซึ่งมีการทำงาน ดังนี้

- ที่ฝั่งส่งจะมีไมโครคอนโทรลเลอร์ที่ต่อกับ Xbee ติดตั้งอยู่บริเวณจุดตัดถนนกับรถไฟ ในขณะที่ที่กั้นรถไฟเปิดอยู่ที่ไมโครคอนโทรลเลอร์จะ Standby และในขณะที่ที่กั้นรถไฟลงสุดจะมีการสั่งไปยังไมโครคอนโทรลเลอร์เพื่อให้ไมโครคอนโทรลเลอร์ส่งข้อมูลรหัส serial number ของ Xbee ไปยัง Xbee ฝั่งรับ
- ที่ฝั่งรับจะมีไมโครคอนโทรลเลอร์ที่ต่อกับ Xbee ติดตั้งอยู่บนรถไฟ และรอข้อมูลที่จะส่งมายัง Xbee เพื่อมีข้อมูลส่งเข้ามา ไมโครคอนโทรลเลอร์จะสั่งให้ Real-Time Clock บันทึกเวลาลงไปด้วย โดยการบันทึกข้อมูลนั้นจะบันทึกไปยังหน่วยความจำภายนอก (EEprom) โดยข้อมูลที่บันทึกจะยังคงอยู่ถึงแม้ว่าจะเกิดอุบัติเหตุทำให้ไม่มีการจ่ายไฟก็ตาม โดยแสดงเป็น Flow chart ต่อไปนี้



รูปที่ 4.1 Flow chart ของฝั่งส่ง



รูปที่ 4.2 Flow chart ของฝั่งรับ

4.1.1 ทำไมจึงส่งข้อมูล serial number ของ Xbee

ในการส่ง Serial number ของ Xbee มีจุดประสงค์เพื่อให้สามารถยืนยันได้ว่ารถไฟกำลังผ่านที่กั้นรถไฟที่จุดไหนซึ่ง Serial number ของ Xbee จะสามารถบอกได้เนื่องจากเรามีฐานข้อมูล Serial number ของ Xbee ตัวนั้นๆว่าติดตั้งอยู่ที่จุดตัดถนนกับรถไฟใด

4.2 การ Configuration Xbee

จะเห็นว่าในการสั่งให้ไมโครคอนโทรลเลอร์ให้ส่งข้อมูลผ่านXbeeเราไม่ได้เขียนโปรแกรมเพื่อให้เป็นการบอกว่า จะให้ส่งไปยัง Xbee ตัวไหน แต่ก็สามารถส่งไปยัง Xbee ตัวที่ต้องการได้ทั้งนี้เพราะเราได้ทำการ Configuration Xbee

สำหรับโครงข่ายไร้สายของ Xbee ในโครงงานนี้ใช้แบบ star โดยมี Xbee ตัวส่งเป็น End Device อยู่บริเวณจุดตัดถนนกับรถไฟซึ่งสามารถจะมีมากกว่าหนึ่งตัวก็ได้ และมี Xbee ตัวรับเป็น Coordinator เพื่อรับข้อมูลจาก Xbee ตามจุดตัดต่างๆ ดังรูป

รูปที่ 4.3 โครงข่ายไร้สายของ Xbee ในโครงงาน

โดยมีขั้นตอนการ Configuration ดังนี้

รูปที่ 4.4 ตัวอย่างรูปการหา SH, SL, MODEM

4.2.1 การตั้งค่า XBee ให้ทำงานเป็น Coordinator

1 เข้าไปที่ Modem Configuration

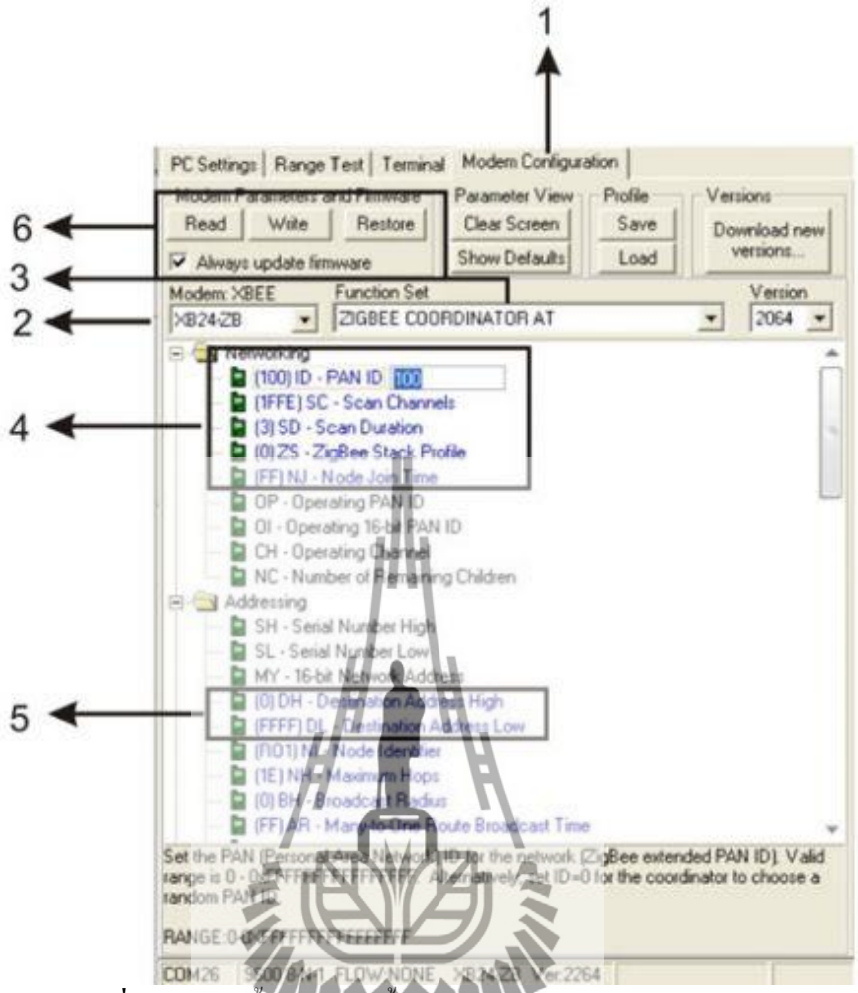
2 เลือก Modem XBee ให้ตรงตามรุ่นที่ใช้ ในที่นี้เลือก XBP24-B

3 เปลี่ยน Firmware ให้เป็น ZIGBEE COORDINATOR AT

4 ตั้ง PAN (Personal Area Network) สามารถตั้งได้ตามแต่ผู้ใช้จะกำหนด ในที่นี้ตั้ง เป็น 100

5 กำหนด Destination (จุดหมายที่ต้องการ รับส่งข้อมูลด้วย)

โดย ตั้งค่า DH = 13A200 , DL = 4064A748



รูปที่ 4.4 แสดงขั้นตอนการตั้งค่า XBee ให้ทำงานเป็น Coordinator

4.2.2 การตั้งค่า XBee ให้ทำงานเป็น End Device

1) เข้าไปที่ Modem Configuration

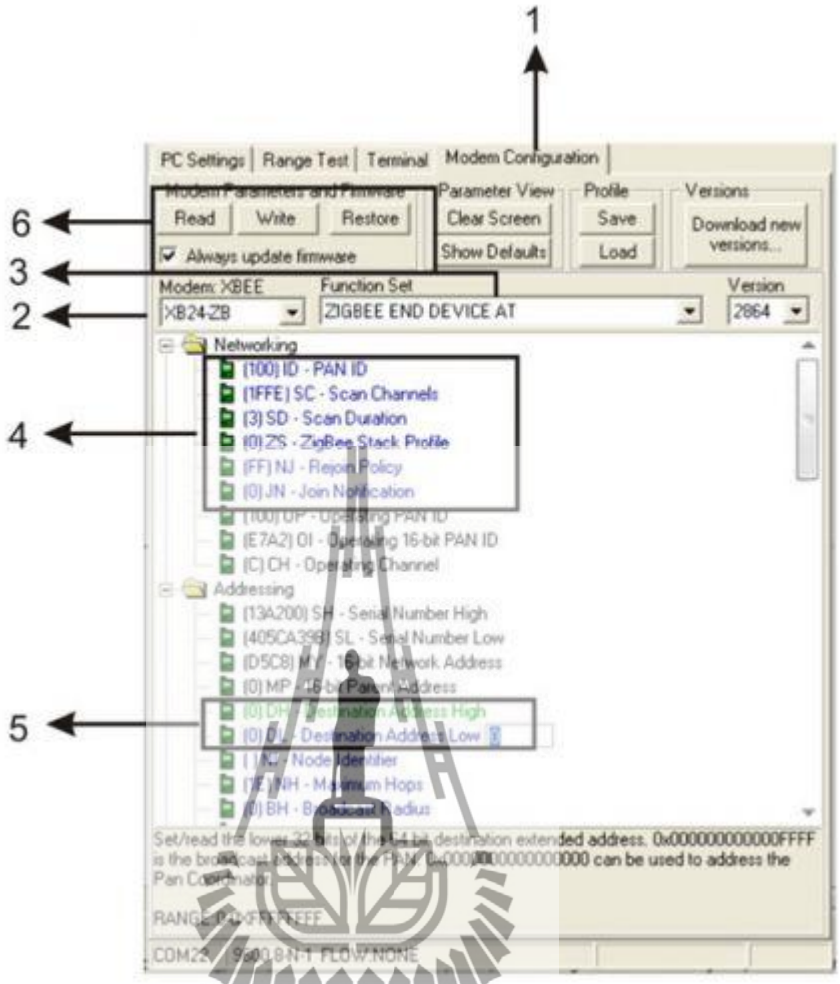
2) เลือก Modem XBee ให้ตรงตามรุ่นที่ใช้ ในที่นี้เลือก XBP24-B

3) เปลี่ยน Firmware ให้เป็น ZIGBEE END DEVICE AT

4) ตั้ง PAN (Personal Area Network) จะต้องตั้งให้เหมือนกับ Coordinator

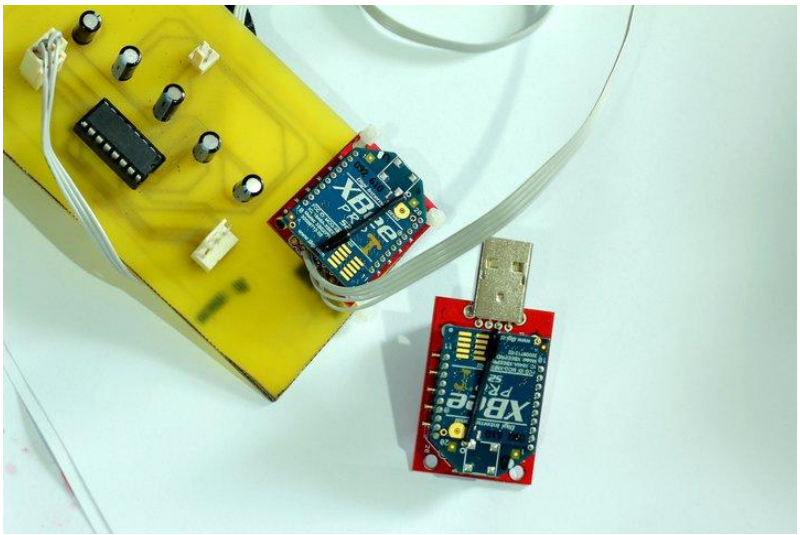
5) กำหนด Destination (จุดหมายที่ต้องการ รับส่งข้อมูลด้วย)

โดยตั้งค่า DH = 13A200 , DL = 4064A740



รูปที่ 4.5 แสดงขั้นตอนการตั้งค่า XBee ให้ทำงานเป็น End Device

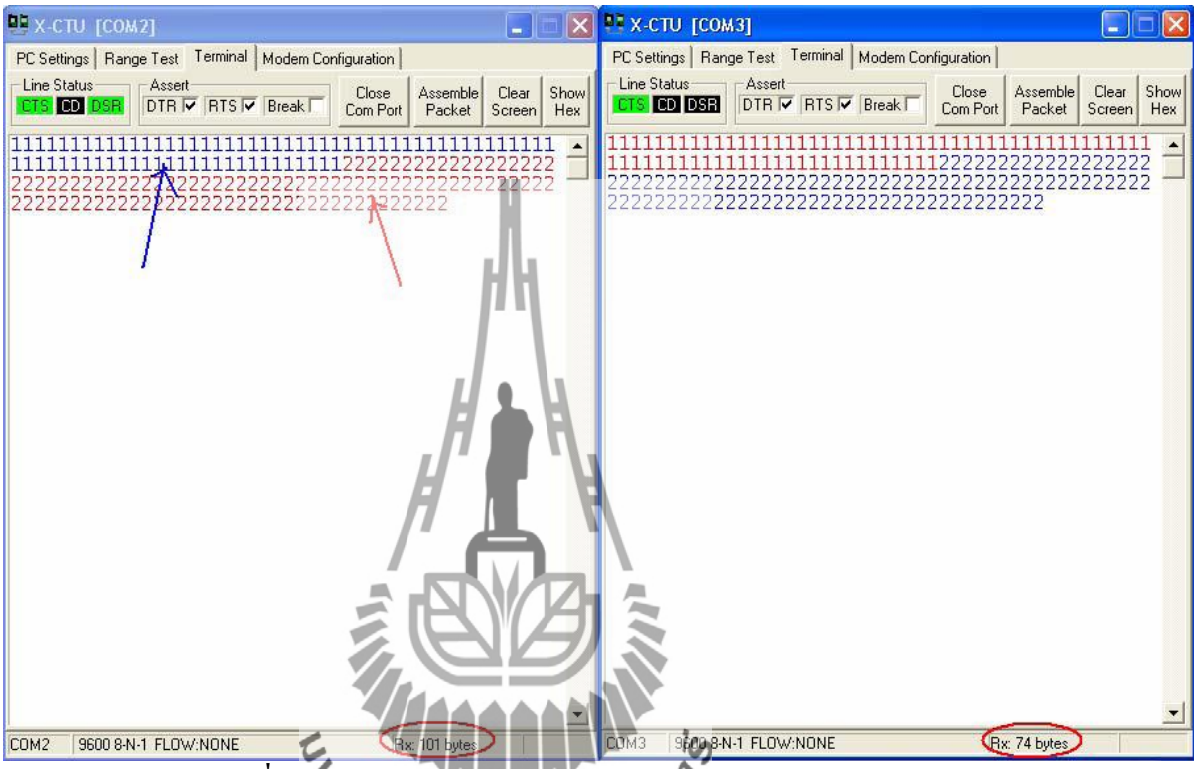
ในการ Configuration จะเชื่อมต่อกับคอมพิวเตอร์ โดยใช้ตัวแปลงดับสัญญาณจาก Serial port RS232 to TTL และใช้ USB Dongle ดังรูปที่ 4.6



รูปที่ 4.6 วิธีการเชื่อมต่อในการ Configuration

4.2.3 การทดสอบผลการ Configuration

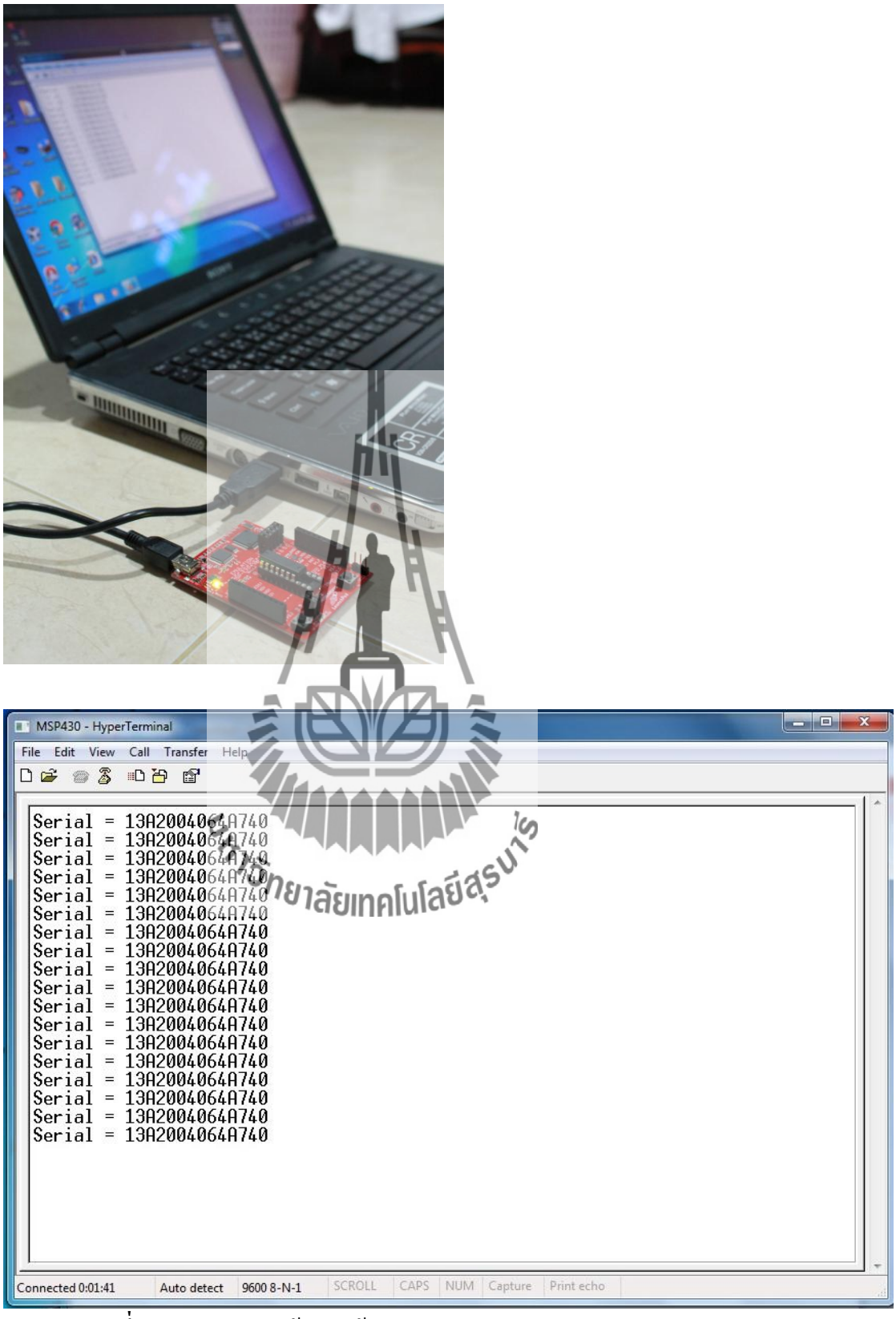
ไปที่แท็บ Terminal พิมพ์ข้อความในช่องฝั่งซ้าย(สีน้ำเงิน) จะเห็นข้อความนั้นมาแสดงที่ฝั่งขวา(สีแดง) ถ้าพิมพ์ข้อความในช่องฝั่งขวา(สีน้ำเงิน) จะเห็นข้อความนั้นมาแสดงที่ฝั่งซ้าย (สีแดง) แสดงว่าสามารถติดต่อกันได้แล้ว



รูปที่ 4.7 การทดสอบการ Configuration ด้วยแท็บ Terminal

4.3 การทำงานของฝั่งส่ง

ไมโครคอนโทรลเลอร์ที่ฝั่งส่งข้อมูลใช้ MSP430 G2231 โดยเขียนโปรแกรมให้ส่ง Serial number โดยส่งทีละตัวอักษรจนครบ โดยให้เว้นระยะเวลาประมาณ 5 วินาที และส่งออกตลอดเวลาเมื่อมีไฟเลี้ยงจ่ายให้ไมโครคอนโทรลเลอร์ โดยทดสอบส่งข้อมูลมายังคอมพิวเตอร์ทาง USB ด้วยโปรแกรม Hyperterminal ดังรูปที่ 4.6

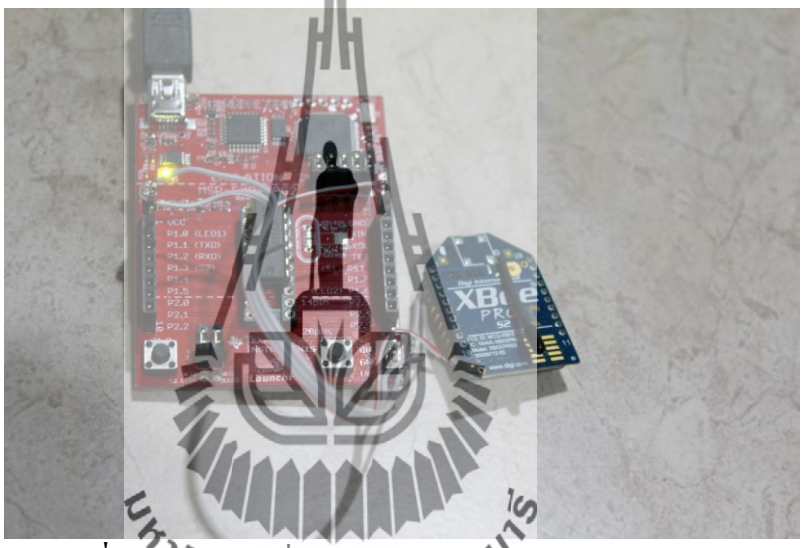


รูปที่ 4.8 แสดงการส่งข้อมูลใช้ MSP430 G2231ผ่านทาง Hyperterminal

หลังจากนั้นจะเปลี่ยนการส่งออกมายัง Xbee ซึ่งสามารถทำได้โดยการเชื่อมต่อ PIN ของ Xbee เข้ากับ Port ของ MSP430 ดังตารางที่4.1 ก็จะสามารถส่งข้อมูลออกผ่าน Xbee ได้

ตารางที่ 4.1 แสดงการเชื่อมต่อ MSP430 G2231 เข้ากับ Xbee

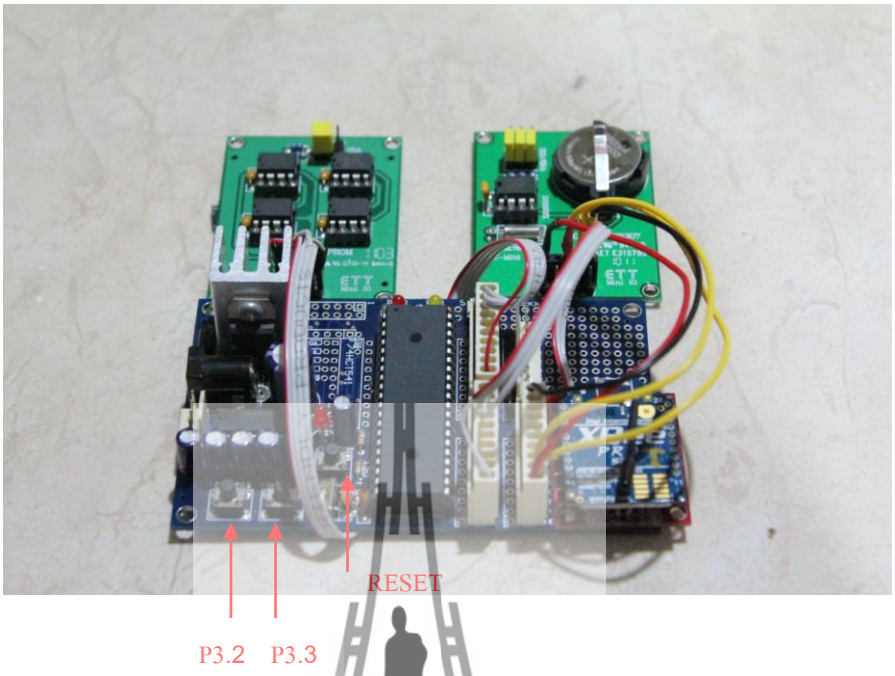
Port ของ MSP430	Xbee
Vcc	Vcc (PIN1)
P1.1 (Tx)	Din(PIN3)
GND	GND(PIN10)



รูปที่ 4.9 แสดงการเชื่อมต่อ MSP430 G2231 เข้ากับ Xbee

4.4 การทำงานของฝั่งรับ

ไมโครคอนโทรลเลอร์ฝั่งรับข้อมูลใช้ บอร์ด MCS51 รุ่น SUT-V5x: MCU-P89V51RD2 ต่อเข้ากับ Xbee และต่อเข้ากับ RTC DS1307 และ EEPROM โดยมีหลักการทำงานคือเมื่อรับข้อมูลมาจะบันทึกข้อมูลนั้นไปยัง EEPROM และเมื่อรับข้อมูลชุดนั้นเสร็จจะแสดงเวลาโดย RTC DS1307 และจะบันทึกเวลานั้นลงไปด้วย หากเวลามีการบันทึกข้อมูลจนเต็มหน่วยความจำ EEPROM จะมีการบันทึกข้อมูลทับข้อมูลเก่าไปเรื่อยๆ โดยสามารถเรียกดูข้อมูลโดยการกดปุ่ม P3.3 และสามารถล้างข้อมูลใน EEPROM โดยการกดปุ่ม P3.2 และสามารถล้างหน้าจอการแสดงผลด้วยการกดปุ่ม Reset

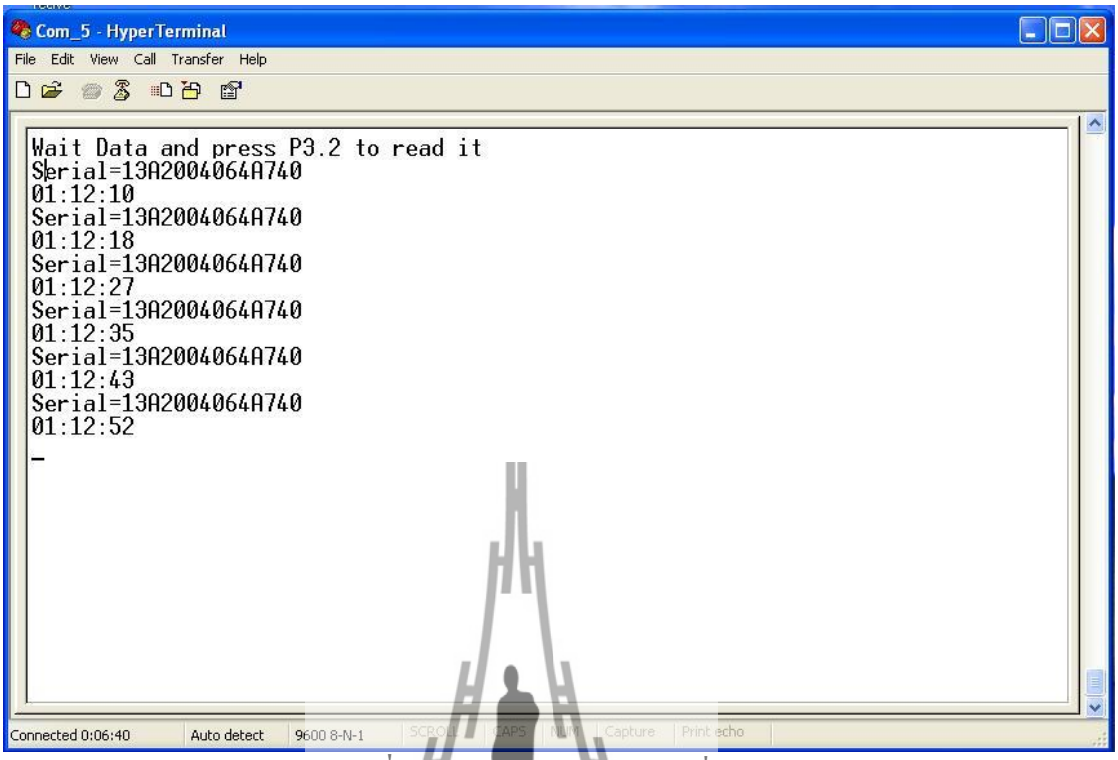


รูปที่ 4.10 แสดงการเชื่อมต่อบอร์ด SUT-V5x: MCU-P89V51RD2 เข้ากับอุปกรณ์

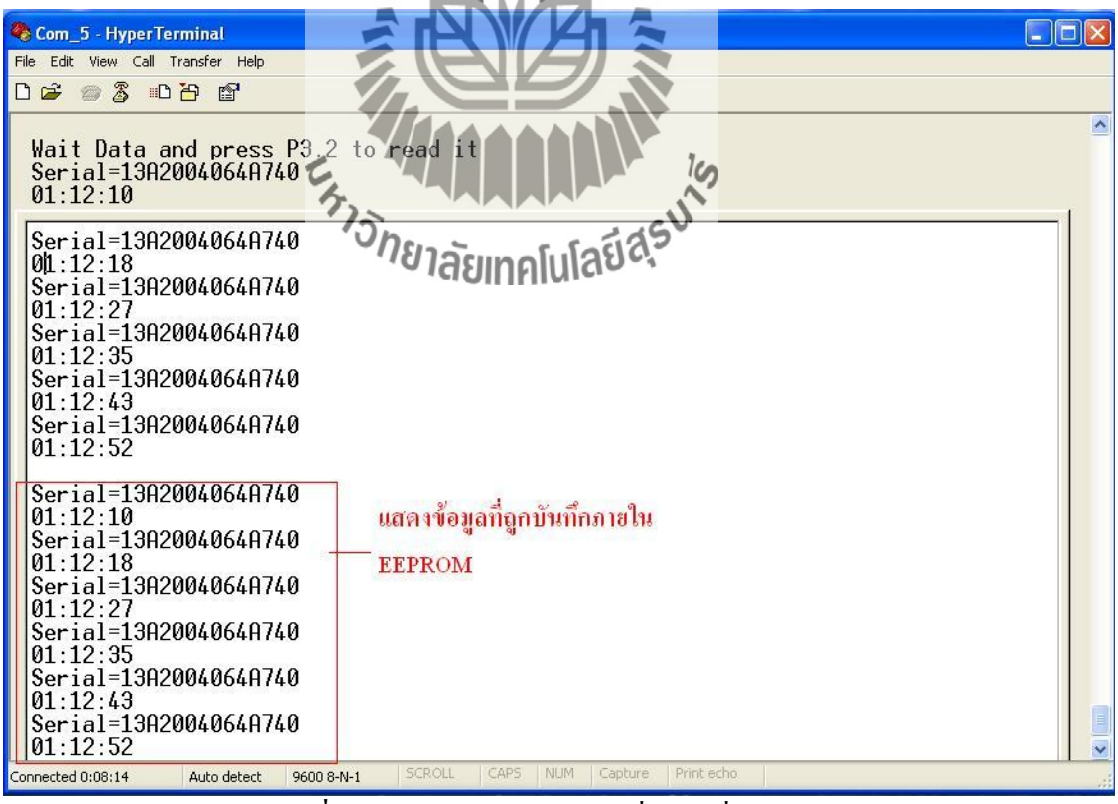
ตารางที่ 4.2 แสดงการเชื่อมต่อ MCU-P89V51RD2 เข้ากับอุปกรณ์

Port ของ P89V51RD2	Xbee	RTC DS1307	EEPROM
Vcc	Vcc (PIN1)	Vcc	Vcc
GND	GND(PIN10)	GND	GND
Rx	Dout	-	-
P1.0	-	SCL	-
P1.1	-	SDA	-
P2.6	-	-	SCL
P2.7	-	-	SDA

*<u>หมายเหตุ</u> ในต่อ Vcc ของ MCU เข้ากับ Xbee จะต้องผ่านการแปลงระดับแรงดันจาก **5v** มาเป็น**3v** ก่อน



รูปที่ 4.11 แสดงข้อมูลและเวลาที่รับได้



รูปที่ 4.12 แสดงข้อมูลและเวลาที่รับได้ที่ถูกบันทึก

4.4.1 RTC DS1307

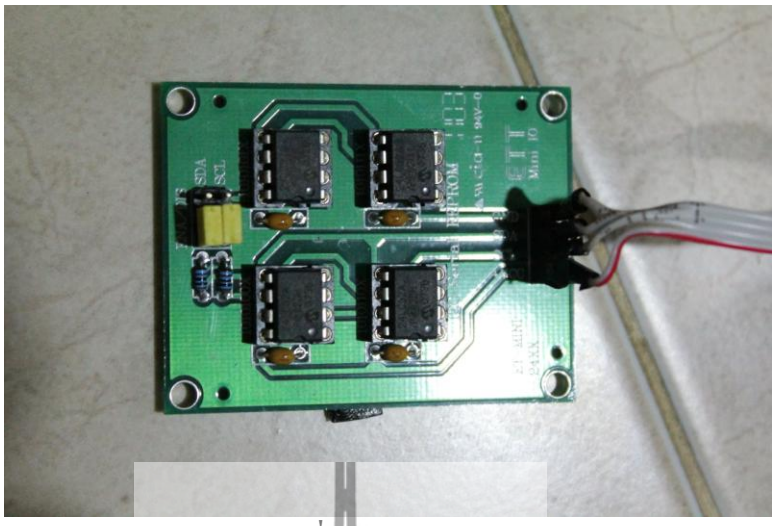
ระบบฐานเวลา เป็นสิ่งสำคัญที่สามารถนำไปใช้ในอุปกรณ์อีเล็กโทรนิกส์ได้หลากหลาย ภายในไมโครคอนโทรเลอร์เองก็มีไทเมอร์เพื่อใช้ในการจับเวลา หรือนำไปใช้เป็นฐานเวลาจริงได้เช่นกัน แต่เนื่องจากไมโครคอนโทรเลอร์สามารถทำงานได้ต่อเมื่อมีไฟเลี้ยงเท่านั้น ดังนั้นการใช้ไทเมอร์ของไมโครคอนโทรลเลอร์ สร้างฐานเวลาจริงจึงไม่ เหมาะสมในบางแอปพลิเคชัน

DS1307 เป็น IC ฐานเวลาของดัลลัสเซมิคอนดักเตอร์ (Dallas Semiconductor) มีบัสรับส่งข้อมูลแบบ I2C ซึ่งเป็นแบบ 2 wire สามารถสื่อสารได้ 2 ทิศทาง (bi-direction bus) ฐานเวลาของ DS1307 นั้นสามารถเก็บข้อมูล วินาที , นาที, ชั่วโมง, วัน, วันที่, เดือน และปี ได้ ระบบเวลาสามารถทำงานโหมดรูปแบบ 24 ชั่วโมง หรือ 12 ชั่วโมง AM/PM ก็ได้ ภายมีระบบตรวจจับแหล่งจ่ายไฟ โดยถ้าแหล่งจ่ายไฟหลักถูกตัดไป DS1307 สามารถสวิตซ์ไปใช้ไฟจากแบตเตอรี่ และทำงานต่อไป โดยที่ยังสามารถรักษาข้อมูลไว้ได้

รูปที่ 4.13 RTC DS1307

4.4.2 EEPROM

EEPROM (Electrically Erasable Programmable Read-Only Memory) หรืออีเอ็ปรอม คือ หน่วย ความจำรอม ที่ผู้ใช้สามารถลบหรือแก้ไขหรือเขียนซ้ำข้อมูลที่บรรจุอยู่ภายในได้ และสามารถกระทำซ้ำได้หลายครั้ง โดยข้อมูลจะยังคงอยู่แม้ว่าไม่มีไฟเลี้ยง ซึ่งหน่วยความจำ EEPROM ที่ได้ใช้ในโครงงานนี้มีความจุ 32KByte x 4 คือตั้งแต่ 0000H-3FFFH



รูปที่ 4.14 EEPROM

4.5 ความสามารถในการเก็บข้อมูล

ในการบันทึกข้อมูลและเวลา 1 รอบ ดังตัวอย่างข้างล่าง

Serial=13A2004064A740

01:12:10

ใช้ความจุ 32 Bytes (32 ตัวอักษร) โดย EEPROM สามารถใช้หน่วยความจำตั้งแต่ 0000H-3FFFH ซึ่งสามารถเก็บข้อมูลได้ดังตาราง

ตารางที่ 4.3 แสดงความสามารถในการเก็บข้อมูลเมื่อกำหนดหน่วยความจำแตกต่างกัน

ขนาดของข้อมูล	การกำหนดช่วงของหน่วยความจำ	ขนาดของหน่วยความจำ	สามารถเก็บข้อมูลได้ (ชุด)
32 Bytes	0000H-00FFH	256 Bytes	8
	0000H-0FFFH	4096 Bytes	128
	0000H-1FFFH	8192 Bytes	256
	0000H-3FFFH	19384 Bytes	512

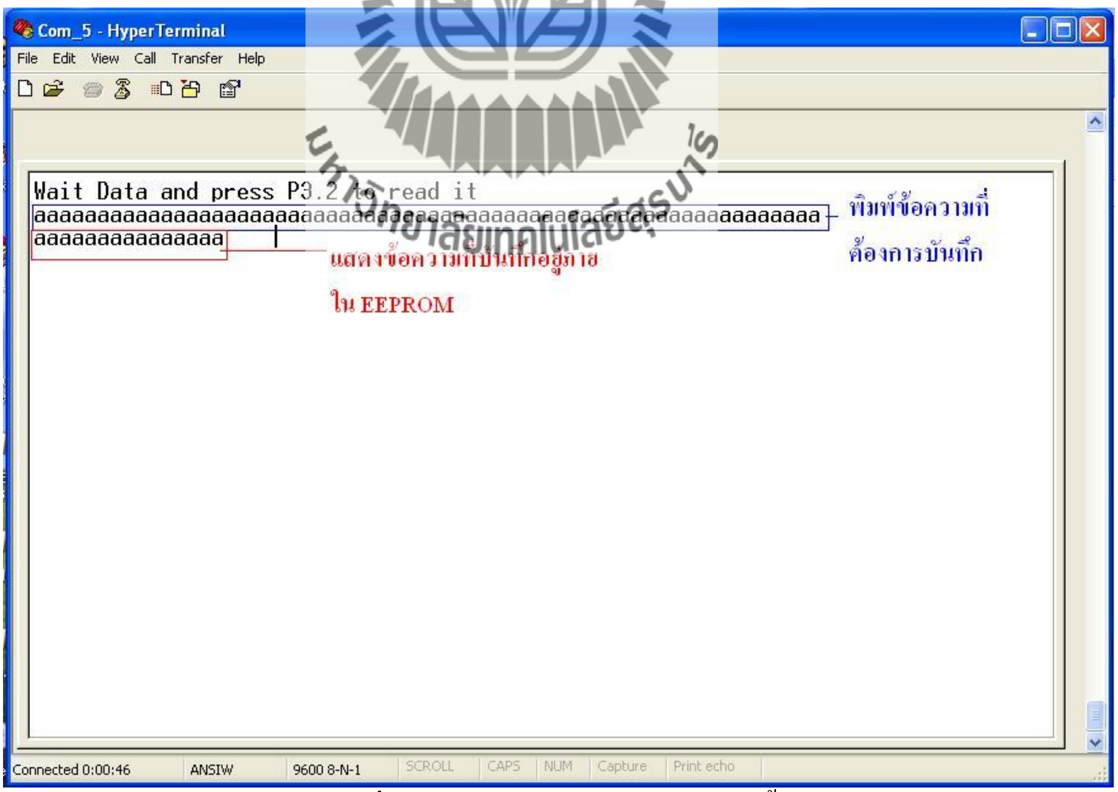
บทที่ 5
ผลการทดสอบ

ในบทนี้จะเสนอการทดสอบความสามารถในด้านต่างๆ ได้แก่ การทดสอบการบันทึกข้อมูลซ้ำและการทดสอบการรับ-ส่งข้อมูลและบันทึกเวลา

5.1 การทดสอบการบันทึกข้อมูลซ้ำ

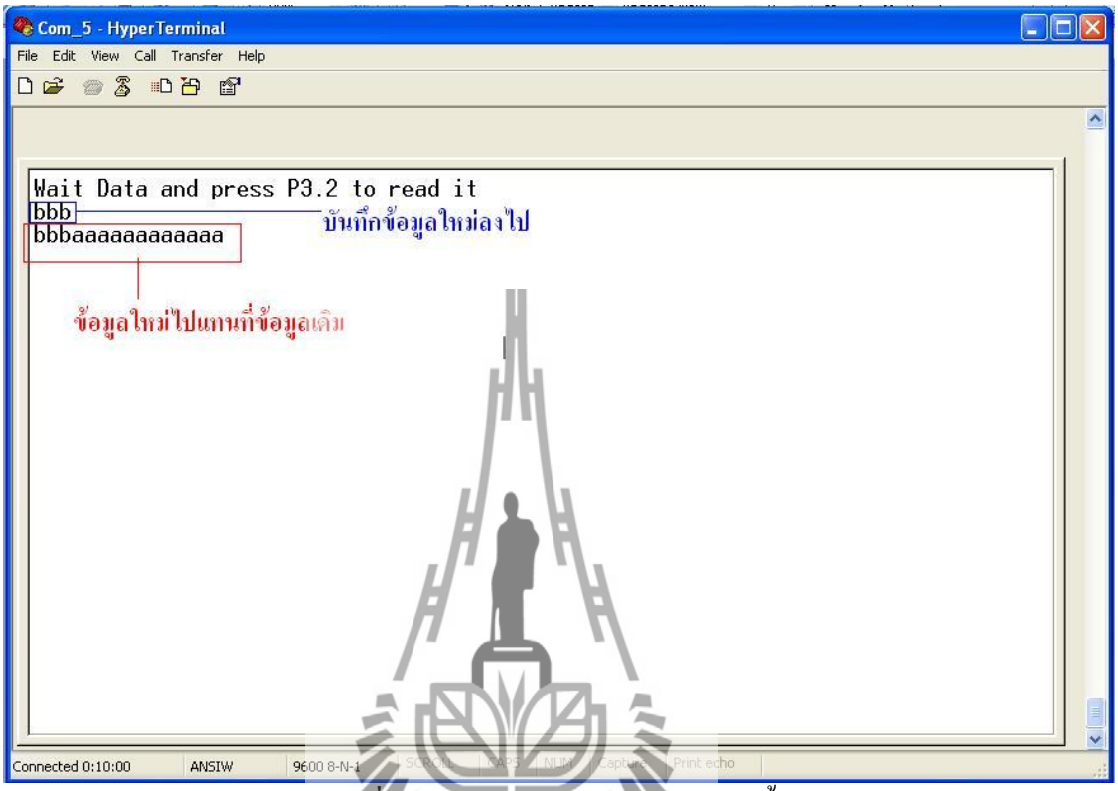
การทดสอบนี้จะทดสอบการบันทึกข้อมูลซ้ำเมื่อข้อมูลใน EEPROM เต็มและจะบันทึกข้อมูลใหม่แทนที่ข้อมูลเดิม โดยมีขั้นตอนทดสอบดังนี้

1. กำหนดช่วงของหน่วยความจำ 0000H-000FH เพื่อให้ง่ายต่อการทดสอบ
2. พิมพ์ aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa บน hyperterminal เพื่อให้บันทึกข้อความลงไป
3. กดปุ่ม P3.2 เพื่อดูข้อมูลภายใน EEPROM (จะต้องมี a 16 ตัว เนื่องจากจำกัดความจุของหน่วยความจำไว้ 16 Bytes)



รูปที่ 5.1 การทดสอบการบันทึกข้อมูลซ้ำ

4. พิมพ์ bbb บน hyperterminal เพื่อให้บันทึกข้อความลงไป

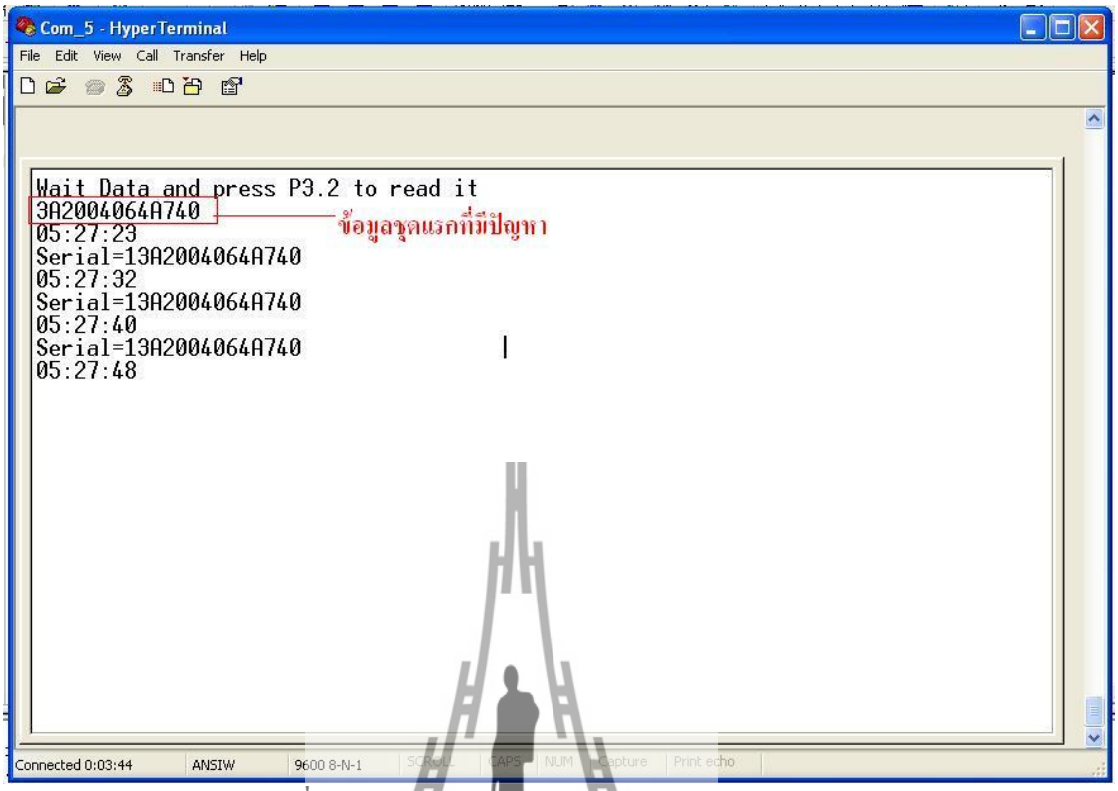


รูปที่ 5.2 การทดสอบการบันทึกข้อมูลซ้ำ

จะเห็นว่าข้อมูลที่ได้พิมพ์เพิ่มไปแทนที่ข้อมูลเดิมที่เก็บไว้ใน EEPROM เป็นที่เรียบร้อย

5.2 การทดสอบการรับ-ส่งข้อมูลและบันทึกเวลา

ในการทดสอบการรับ-ส่งข้อมูลและบันทึกเวลา สามารถรับส่งข้อมูลได้ปกติ แต่อาจจะมีปัญหาในการรับข้อมูลชุดแรกเนื่องจากฝั่งส่งมีการส่งข้อมูลตลอดเวลาจึงเป็นผลทำให้บางครั้งรับข้อมูลตั้งแต่ตัวแรกไม่ได้ แต่จะไม่มีผลต่อข้อมูลชุดต่อไป



รูปที่ 5.3 การทดสอบการรับ-ส่งข้อมูลและบันทึกเวลา

บทที่ 6
บทสรุป

เนื้อหาในบทนี้จะเป็นการสรุปผลการดำเนินโครงการ รวมไปถึงปัญหา แนวทางการแก้ไข และข้อเสนอแนะเพื่อเป็นประโยชน์ในการวิเคราะห์และออกแบบระบบต่อไป

6.1 สรุปผลการดำเนินโครงการ

โครงการนี้ได้นำ Xbee มาต่อเข้ากับไมโครคอนโทรลเลอร์โดยมีอุปกรณ์เสริมคือ RTC DS1307 และ EEPROM โดยทดลองสร้างระบบเก็บข้อมูลระหว่างการเดินรถของรถไฟ ได้เป็นผลสำเร็จตามเป้าหมายโดยระบบที่สร้างขึ้นมาสามารถบันทึกข้อมูล Serial number และเวลาที่เครื่องกั้น ถนน-รถไฟ ปิดได้

6.2 ข้อเสนอแนะ

1. ปัจจัยสำคัญที่มีผลต่อจำนวนรอบของชุดข้อมูลที่บันทึกได้ ถูกจำกัดด้วยขนาดของหน่วยความจำ EEPROM ในโครงการนี้ได้ใช้หน่วยความจำ EEPROM ที่มีความจุ 32 kBytes จำนวน 4 ตัว ซึ่งสามารถที่จะเพิ่มขนาดของหน่วยความจำให้มากขึ้นได้ โดยการเปลี่ยน IC EEPROM (24LCXX) ซึ่งมีความจุตั้งแต่ 64,128,256,512 kBytes

2. ในความเป็นจริงแล้วในวงการรถไฟจะเรียกชื่อของที่กั้นถนน -รถไฟ ด้วยสาย และ หลักกิโลเมตร โดยสามารถเปลี่ยนข้อมูลที่ส่งเป็นหลักกิโลเมตรก็ได้จะทำให้ขนาดของข้อมูลที่ส่งลดลงได้มาก

3. โปรแกรมที่สร้างขึ้นมานี้สามารถนำไปดัดแปลง แก้ไข เพื่อประยุกต์ใช้แก้ปัญหาเฉพาะกรณีๆ ได้

ประวัติผู้เขียน

นายวิศรุต วิเวก เกิดเมื่อวันที่ 21 เมษายน พ.ศ. 2533 ภูมิลำเนาอยู่ที่ ตำบลหนองจะบก อำเภอเมือง จังหวัดนครราชสีมา สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนราชสีมาวิทยาลัย อำเภอเมือง จังหวัดนครราชสีมา เมื่อปี พ.ศ. 2550 ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

นางสาวกมลวรรณ สุดพุ่ม เกิดเมื่อวันที่ 27 กรกฎาคม พ.ศ. 2532 ภูมิลำเนาอยู่ที่ ตำบลนกออก อำเภอเมือง จังหวัดนครราชสีมา สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนสุรนารีวิทยา อำเภอ เมือง จังหวัดนครราชสีมา เมื่อปี พ.ศ. 2550 ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

บรรณานุกรม

[1] **คู่มือปฏิบัติการไมโครโปรเซสเซอร์** , อาจารย์วิชัย ศรีสุรักษ์ สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยสุรนารี (ฉบับร่าง 6 ปีการศึกษา 2/2553)

[2] **ภาษาแอสเซมบลี สำหรับ MCS-51** , รศ.ธีรวัฒน์ ประกอบผล

[3] **เขียนโปรแกรมควบคุม Microcontroller ด้วยภาษา C, Assembly และ VB** , อภิชาติ ภู่พลับ

[4] **ภาษาซีสำหรับผู้เริ่มต้น** ,คะชา ชาญศิลป์

[5] **Zigbee and Xbee BASIC**, http://www.thaieasyelec.com/Embedded-Electronics-Application/Xbee-Basic-Configuration-in-Network-Application.html

[6] **การศึกษาจัดทำแผนแก้ไขอุบัติเหตุบริเวณจุดตัดทางรถไฟกับถนนสำหรับรถไฟทางไกล ,** สำนักงานนโยบายและแผนการขนส่งและจราจรกระทรงคมนาคม

ภาคผนวก

โปรแกรม code_tx.c

```
//*****************************************************************************
//   Software UART using Timer_A
//   Description: This example illustrates how to output characters
//   to an RS-232 serial port without a UART peripheral.
//                MSP430
//   Texas Instruments Inc.
//*****************************************************************************
#include <msp430.h>
#include <ti/mcu/msp430/csl/CSL.h>
//*****************************************************************************
;/    application definitions
//*****************************************************************************
* Bit period for 9600 Baud SW UART, assuming
 *    SMCLK = 1MHz,
 *    Timer_A input divider = 8 (=> timer freq = 125 KHz)
#define BITIME  (13 )          // 125 KHz /(13)
unsigned char bitCnt;          // number of bits to transmit
unsigned int  txByte;          // transmit buffer with start/stop bits
void putstr(char *str);        // transmit string
void transmit(unsigned char ch);       // transmit a character
//*****************************************************************************
;/   main
//*****************************************************************************
int i,j,k,l ;
int main(void)
{
   CSL_init();                 // Activate Grace-generated config
   while (1) {
```

```
        putstr("S");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
}
        putstr("e");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
}
        putstr("r");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
}
        putstr("i");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
}

        putstr("a");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
}
        putstr("l");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
```

```
}
                putstr("=");
    for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                        { }
}
                putstr("1");
    for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                        { }
}


                putstr("3");
    for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                        { }
}
                putstr("A");
    for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                        { }
}
                putstr("2");
    for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                        { }
}
                putstr("0");
    for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
```

```
                    { }
}
                    putstr("0");
      for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                    { }
}
                    putstr("4");
      for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                    { }
}
                    putstr("0");
      for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                    { }
}
                    putstr("6");
      for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                    { }
}

                    putstr("4");
      for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                    { }
}
                    putstr("A");
      for ( l = 0; l < 128; l++ )
```

```
{ for ( i = 0; i < 128; i++ )
                                { }
}
                        putstr("7");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
}
                        putstr("4");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
}
                        putstr("0");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
}
                        putstr("\n");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
}
                        putstr("\r");
        for ( l = 0; l < 128; l++ )
{ for ( i = 0; i < 128; i++ )
                                { }
}

        for (j = 0; j < 128; j++)
```

```
                    {
                    for ( i = 0; i < 128; i++ )
                    {}
                    }
  }
}
//*************************************************************************
;//   putstr
//*************************************************************************
void putstr(char *str)
{
  char *cp;
  for (cp = str; *cp != '\0'; cp++) {
    transmit(cp[0]);
  }
}
//*************************************************************************
;//   transmit
//*************************************************************************
*  Transmit specified character
 */
void transmit(unsigned char ch)
{
  bitCnt = 0xA;                  // Load Bit counter, 8data + ST/SP
  txByte = (unsigned int)ch | 0x100;  // Add mark stop bit to txByte
  txByte = txByte << 1;          // Add space start bit

  CCR0 = TAR + BITIME;           // Some time till first bit
  CCTL0 = OUTMOD0 + CCIE;        // TXD = mark = idle
  while (CCTL0 & CCIE);          // Wait for ISR to complete TX
```

```
}
//*******************************************************************************
;//    timer_A_ISR
//*******************************************************************************
*  Timer A0 interrupt service routine
 */
void timer_A_ISR(void)
{
   CCR0 += BITIME;                    // Schedule next interrupt
   if (bitCnt == 0) {
      CCTL0 &= ~CCIE;                 // All bits TXed, disable interrupt
   }
   else {
      if (txByte & 0x01) {
         CCTL0 &= ~OUTMOD2;           // TX Mark
      }
      else {
         CCTL0 |= OUTMOD2;            // TX Space
      }
      txByte = txByte >> 1;
      bitCnt--;
   }
}
```

โปรแกรม recive.asm

```
PortO_Ylw       EQU     P3.7
PortO_Grn       EQU     P3.6
AddL            EQU     60H
AddH            EQU     61H
M_AddL          EQU     65H
M_AddH          EQU     66H
;/*************************************************************************
;/ Test 24C32 EEPROM
;/*************************************************************************
;/* Clock Control Registers */;
;CKCON      EQU     08FH          ; Clock Control Register
;/* I2C-Bus Device Interface */;
I2C_SCL     EQU     P1.0          ; I2C-Bus Clock
I2C_SDA     EQU     P1.1          ; I2C-Bus Data
;/*************************************************************************
;/RTC
;/*************************************************************************
CKCON               EQU     08FH        ; Clock Control Register
WD_Reset            EQU     0A6H        ; Hardware Watch-dog Reset
WD_Prog             EQU     0A7H        ; Hardware Watch-Dog Program
PortO_I2C_Data      EQU     P2.0        ; Start bit
PortO_I2C_Clk       EQU     P2.1
Cpl_LED_Index       EQU     20H
Hour                EQU     21H
Minutes             EQU     22H
Seconds             EQU     23H
timeL               EQU     63H
timeH               EQU     64H
```

```
;/*****************************************************************************
        ORG     0000H
        JMP     0100H

        ORG     0023h
        JMP     Serial_Intr

        ORG     0100H
        MOV     SP,#2FH                 ; Initial Stack
        ANL     CKCON,#11111110B        ; Set Clock = Standard Mode (12 Clock)
        CALL    Initial_Serial
        CALL    Initial_RTC
;/*****************************************************************************
;/ Initial Bus for I2c
;/*****************************************************************************
    CLR     I2C_SCL             ; Initial Bus For Stop
    CLR     I2C_SDA
    SETB    I2C_SCL
    SETB    I2C_SDA             ; Stop Condition (SCL=1,SDA=1)
;/*****************************************************************************
        CALL    Send_ClearScreen

        SETB    EA
        SETB    ES

                MOV     DPTR,#Text1
                CALL    Send_Table
                CALL    Send_LineFeed
                MOV     AddH,#00H
                MOV     AddL,#00H
```

```
;/*****************************************************************************
;/Main_LOOP
;/*****************************************************************************

Main_LOOP:  JNB   P3.3,_show_key
            JNB   P3.2,reset_data
            JMP   Main_LOOP
;/*****************************************************************************
;/ show_key
;/*****************************************************************************
_show_key:  MOV   M_AddH,AddH
            MOV   M_AddL,AddL
            MOV   AddH,#00H
            MOV   AddL,#00H
            CALL  Send_LineFeed
Loop:       MOV   DPH,AddH
            MOV   DPL,AddL
            CALL  EEP_READ1
            CALL  SEND_ASCII
            INC   DPTR
            MOV   AddL,DPL
            MOV   AddH,DPH
            MOV   A,AddH
            CJNE  A,M_AddH,Loop
            MOV   A,AddL
            CJNE  A,M_AddL,Loop
            MOV   AddH,#00H
            MOV   AddL,#00H
            CALL  DELAY_1SEC
            JMP   Main_LOOP
```

```
;/*****************************************************************************
;/ Serial Int
;/*****************************************************************************
Serial_Intr:    CLR     ES
                JB      TI,_X_Srl_Intr
                MOV     A,SBUF
                MOV     SBUF,A
                ;CALL    Conv_1ASC2HEX
                MOV     DPH,AddH
                MOV     DPL,AddL
                CALL    EEP_WRITE
                MOV     A,SBUF                  ;check the end of data
                CJNE    A,#0DH,INC
                CALL    ReadShow_Time
                CALL    Send_LineFeed
                MOV     DPH,AddH
                MOV     DPL,AddL                ;
INC:            INC     DPTR
check_full:     MOV     A,DPH
                CJNE    A,#0FH,_X_Srl_Intr
                MOV     A,DPL
                CJNE    A,#0FFH,_X_Srl_Intr
                MOV     DPTR,#0000H

_X_Srl_Intr:    MOV     AddH,DPH
                MOV     AddL,DPL
                CLR     TI
                CLR     RI
                SETB    ES
                RETI
```

```
;/*****************************************************************************
;/ reset_data
;/*****************************************************************************
reset_data:     MOV   AddH,#00H
                MOV   AddL,#00H
                MOV   DPH,AddH
                MOV   DPL,AddL
loop1:          MOV   A,#' '
                MOV   SBUF,A
                CALL  EEP_WRITE
                INC   DPTR
                MOV   A,DPH
                CJNE  A,#0FH,loop1
                MOV   A,DPL
                CJNE  A,#0FFH,loop1
                MOV   DPTR,#0000H
                JMP   Main_LOOP
;/*****************************************************************************
Delay:  MOV   R1,#2
DLY00:DJNZ    Acc,DLY00
        DJNZ  B,DLY00
        DJNZ  R1,DLY00
        RET
Text1:          DB    'Wait Data and press P3.2 to read it',00H
;/*****************************************************************************
        $INCLUDE      "00_Systm.Sub"
        $INCLUDE      "EEPROM.sub"
        $INCLUDE      "10_D1307.sub"
;/*****************************************************************************
END
```

โปรแกรมย่อย 00_Systm.sub

```
;/*****************************************************************************
;/ Conv_2Hex2Ascii:
;/*****************************************************************************
;  input A      output timeH ,timeL
Conv_2Hex2Ascii:
        push    Acc
        ANL     A,#0FH
        MOV     DPTR,#HEXASC_TAB
        MOVC    A,@A+DPTR
        MOV     timeL,A
        pop     Acc
        ANL     A,#0F0H
        SWAP    A
        MOV     DPTR,#HEXASC_TAB
        MOVC    A,@A+DPTR
        MOV     timeH,A
        RET
;/*****************************************************************************;
Converse Serial Buffer@R1 to 2HEX@Acc
;/*****************************************************************************
Conv_2ASC2HEX:      PUSH    B
        CALL    Conv_1ASC2HEX
        SWAP    A
        MOV     B,A
        CALL    Conv_1ASC2HEX
        ORL     A,B
        POP     B
        RET
```

```
Conv_1ASC2HEX:      ;MOV   A,@R1
        MOV    C,Acc.6
        ANL    A,#0FH
        JNC    _X_Conv_1ASC2HEX
        ADD    A,#09H
        ANL    A,#0FH
_X_Conv_1ASC2HEX:  ;INC    R1
        RET

;/*************************************************************************;
Once Byte at Acc BCD2HEX Converstor
;/*************************************************************************
BCD2HEX:    PUSH   B
        MOV    B,A     ; Store at B
        ANL    A,#0FH
        PUSH   Acc     ; Store low of Acc
        MOV    A,B
        SWAP   A
        ANL    A,#0FH
        MOV    B,#10
        MUL    AB
        POP    B       ; Get Low of Acc
        ADD    A,B
        POP    B
        RET

;/*************************************************************************;
Once Byte at Acc Hex to BCD Converstor
;/*************************************************************************
HEX2BCD:    PUSH   06H     ;#For Output
```

```
        PUSH   07H      ;#For Output
        PUSH   DPH      ;#For Loop_Count
        MOV    B,A      ; Store A
        MOV            R6,#00  ; Clear Output
        MOV            R7,#00  ; Clear Output
        MOV            DPH,#8D         ; Data 8 Bits
HEXTOBCD01:     MOV            A,B      ; Bit Shift
        RLC            A
        MOV            B,A
HEXTOBCD02:     MOV            A,R7     ; Byte Add +0
        ADDC           A,Acc
        DA             A
        MOV            R7,A
        MOV            A,R6     ; Byte Add +1
        ADDC           A,Acc
        DA             A
        MOV            R6,A
        DJNZ           DPH,HEXTOBCD01 ; Loop Bit Shift
        MOV    B,R6    ; Load Output
        MOV    A,R7    ; Load Output
        POP    DPH
        POP    06H
        POP    07H
        RET
;/*****************************************************************************
; Initial Serial Port
;/*****************************************************************************
Initial_Serial:  MOV    SCON,#01010000B      ; Mode1 Serial Port
                 MOV    RCAP2H,#0FFH         ; Reload values for desired
                 MOV    RCAP2L,#0C4H         ; baud rate @18.432 9600=ffc4,4800=ff88
```

```
                MOV   T2CON,#00110100B      ; Timer 2 as baud rate generator
                RET                 ; and turn on Timer 2
;/*****************************************************************************
; Send Data to Serial Port
;/*****************************************************************************
SEND_3DEC:  CALL  HEX2BCD
                PUSH  Acc
                MOV   A,B
                CALL  Send_1HEX
                POP   Acc
SEND_2HEX:  PUSH  Acc
                SWAP  A
                CALL  SEND_1HEX
                POP   Acc
SEND_1HEX:  PUSH  DPH
                PUSH  DPL
                MOV   DPTR,#HEXASC_TAB
                ANL   A,#0FH
                MOVC  A,@A+DPTR
                POP   DPL
                POP   DPH
SEND_ASCII: CLR   ES
                CLR   TI
        MOV   SBUF,A
        MOV   timeL,A
        JNB   TI,$
        CLR   TI
        SETB  ES
        RET
HEXASC_TAB:         DB      '0123456789ABCDEF'
```

```
;/*****************************************************************************
; Send Line Feed
;/*****************************************************************************
Send_LineFeed: MOV   A,#0DH
               CALL  Send_ASCII
               MOV   A,#0AH
               CALL  Send_ASCII
               RET
;/*****************************************************************************
; Send Line Feed
;/*****************************************************************************
Send_ClearScreen:      MOV   A,#0CH
                       CALL  Send_ASCII
                       RET
;/*****************************************************************************
; Send data Table to Serial Start at DPTR to '00H'
;/*****************************************************************************
Send_Table:    CLR   A
               MOVC  A,@A+DPTR
               JZ    _X_Send_Table
               CALL  SEND_ASCII
               INC   DPTR
               JMP   SEND_TABLE
_X_Send_Table:RET
;/*****************************************************************************
;#*      Strat Display  for XTAL 24.00MHz
;/*****************************************************************************
DELAY_4SEC: PUSH  07H
               MOV   R7,#80  ; Delay 80*50MilliSec = 4000MilliSec
               JMP   _Start_DELAY_50M
```

```
DELAY_2SEC: PUSH   07H
            MOV    R7,#40  ; Delay 40*50MilliSec = 2000MilliSec
            JMP    _Start_DELAY_50M

DELAY_1SEC: PUSH   07H
            MOV    R7,#20  ; Delay 20*50MilliSec = 1000MilliSec
            JMP    _Start_DELAY_50M

DELAY_500M: PUSH   07H
            MOV    R7,#10  ; Delay 10*50MilliSec = 500MilliSec
            JMP    _Start_DELAY_50M

DELAY_250M: PUSH   07H
            MOV    R7,#5   ; Delay 250 MilliSec
            JMP    _Start_DELAY_50M

DELAY_200M: PUSH   07H
            MOV    R7,#4   ; Delay 100 MilliSec
            JMP    _Start_DELAY_50M

DELAY_150M: PUSH   07H
            MOV    R7,#3   ; Delay 150 MilliSec
            JMP    _Start_DELAY_50M

DELAY_100M: PUSH   07H
            MOV    R7,#2   ; Delay 100 MilliSec
            JMP    _Start_DELAY_50M

DELAY_50M:  PUSH   07H
            MOV    R7,#1   ; Delay 50 MilliSec
```

```
                JMP     _Start_DELAY_50M

_Start_DELAY_50M:   PUSH   B          ; 2MC ->
                    PUSH   Acc        ; 2MC ->
_DELAY_50M:         MOV    B,#200  ; 2MC -> Delay 50 MilliSec
_DELAY_250U:        MOV    A,#192  ; 1MC -> Delay 250 MicroSec
                    DJNZ   Acc,$   ; 2MC -> 192*0.651*2 = 249.9 MicroSec
                    DJNZ   B,_DELAY_250U          ; 2MC -> 249*200 = 50  MilliSec
                    DJNZ   R7,_DELAY_50M          ; 2MC -> 50*5   = 250 MilliSec
                    POP    Acc        ; 2MC ->
                    POP    B          ; 2MC ->
                    POP    07H        ; 2MC ->
                    RET

DELAY_25M:          PUSH   Acc
                    PUSH   B
                    MOV    B,#100
_DELAY_250MICRO:    MOV    A,#166  ; Delay 250 MicroSec
                    DJNZ   Acc,$   ; 166*0.5*3     = 249 MicroSec
                    DJNZ   B,_DELAY_250MICRO
                    POP    B
                    POP    Acc
                    RET
```

โปรแกรมย่อย 10_D1307.sub

```
;/*****************************************************************************
; ReadShow_Time:
;/*****************************************************************************
ReadShow_Time:      CALL   Read_RTC_Time
                    MOV    DPH,AddH
                    MOV    DPL,AddL
                    INC    DPTR
                    MOV    AddL,DPL
                    MOV    AddH,DPH
-------------------------------------------------------------------------------------------
                    MOV    A,Hour
                    CALL   Send_2HEX
                    MOV    A,Hour
                    CALL   Conv_2Hex2Ascii

                    MOV    A,timeH
                    MOV    DPH,AddH
                    MOV    DPL,AddL
                    CALL   EEP_WRITE
                    CALL   DELAY_150M
                    INC    DPTR
                    ;MOV   AddL,DPL
                    ;MOV   AddH,DPH
-------------------------------------------------------------------------------------------
                    ;MOV   DPH,AddH
                    ;MOV   DPL,AddL
                    MOV    A,timeL
                    CALL   EEP_WRITE
```

```
CALL   DELAY_150M
INC    DPTR
MOV    AddL,DPL
MOV    AddH,DPH
```

```
MOV    A,#':'
CALL   Send_ASCII
MOV    A,#':'

MOV    DPH,AddH
MOV    DPL,AddL
CALL   EEP_WRITE
CALL   DELAY_150M
INC    DPTR
MOV    AddL,DPL
MOV    AddH,DPH
```

```
MOV    A,Minutes
CALL   Send_2HEX
MOV    A,Minutes
CALL   Conv_2Hex2Ascii
MOV    A,timeH
MOV    DPH,AddH
MOV    DPL,AddL
CALL   EEP_WRITE
CALL   DELAY_150M
INC    DPTR
```

```
MOV   A,timeL
CALL  EEP_WRITE
CALL  DELAY_150M
INC   DPTR
MOV   AddL,DPL
MOV   AddH,DPH
```

```
MOV   A,#':'
CALL  Send_ASCII
MOV   A,#':'
MOV   DPH,AddH
MOV   DPL,AddL
CALL  EEP_WRITE
CALL  DELAY_150M
INC   DPTR
MOV   AddL,DPL
MOV   AddH,DPH
```

```
MOV   A,Seconds
CALL  Send_2HEX
MOV   A,Seconds
CALL  Conv_2Hex2Ascii

MOV   A,timeH
MOV   DPH,AddH
MOV   DPL,AddL
CALL  EEP_WRITE
CALL  DELAY_150M
INC   DPTR
```

```
                MOV     A,timeL
                CALL    EEP_WRITE
                CALL    DELAY_150M
                INC     DPTR
                MOV     AddL,DPL
                MOV     AddH,DPH
-------------------------------------------------------------------------------------------------------------
                MOV     A,#0DH
                MOV     DPH,AddH
                MOV     DPL,AddL
                CALL    EEP_WRITE

                INC     DPTR
                MOV     AddL,DPL
                MOV     AddH,DPH
-------------------------------------------------------------------------------------------------------------
                MOV     A,#0AH
                MOV     DPH,AddH
                MOV     DPL,AddL
                CALL    EEP_WRITE
-------------------------------------------------------------------------------------------------------------
                RET
;/*****************************************************************************
;       Read(Save) Data to RTC Time
;/*****************************************************************************
Read_RTC_Time:  MOV     DPL,#02H
                CALL    Read_Time
                MOV     Hour,A
                MOV     DPL,#01H
                CALL    Read_Time
```

```
        MOV   Minutes,A
        MOV   DPL,#00H
        CALL  Read_Time
        MOV   Seconds,A
        RET

INC_H:  MOV   DPL,#02H
        MOV   A,HOUR
        ADD   A,#01H
        DA    A
        CALL  WRITE_TIME
        CJNE  A,#23H,RUN
        CLR   A
RUN:    MOV   DPL,#0H       ; Hour 23(BCD)
        MOV   R7,#23H
        CALL  Initial_RTC
        RET

INC_M:  MOV   DPL,#01H
        MOV   A,MINUTES

        ADD   A,#01H
        DA    A
        CALL  WRITE_TIME
        CJNE  A,#59H,RUN1
        CLR   A

RUN1:   MOV   DPL,#01H      ; Min 59(BCD)
        MOV   R7,#59H
        MOV   A,#00
```

```
                CALL   Initial_RTC

                RET
;/*****************************************************************************
; Save_RTC_Time:
;/*****************************************************************************
Save_RTC_Time:  MOV    DPL,#02H
                MOV    A,Hour
                CALL   Write_Time
                MOV    DPL,#01H
                MOV    A,Minutes
                CALL   Write_Time
                MOV    DPL,#00H
                MOV    A,Seconds
                CALL   Write_Time
                RET
;/*****************************************************************************
;/*****************************************************************************
CONT_BYTE_RTC_W EQU    0D0H
CONT_BYTE_RTC_R EQU    0D1H
;Address 00H    Seconds 00–59
;Address 01H    Minutes 00–59
;Address 02H    Hours   1–12,00-23
;Address 03H    Day     01–07
;Address 04H    Date    01–31
;Address 05H    Month   01–12
;Address 06H    Year    00–99
;Address 07H    Control [OUT 0 0 SQWE 0 0 RS1 RS0]
;Address 08H-3FH        RAM 56x8  00      H–FFH
```

```
;/*****************************************************************************
;*          WRITE TIME TO RTC 1 BYTE
;*   INPUT  : DPL of Address  ,A   of Data
;/*****************************************************************************
WRITE_TIME:  MOV    B,A
WRITE_TIME00:        CLR     PortO_I2C_Data          ;Start bit
        CLR     PortO_I2C_Clk
_WR_TYPE_RTC:        MOV             A,#CONT_BYTE_RTC_W  ;Send control byte
        CALL            BYTE_RTC_WR
        SETB            PortO_I2C_Data
        SETB            PortO_I2C_Clk
        JB      PortO_I2C_Data,WRITE_TIME
        CLR     PortO_I2C_Clk
_WR_ADD_RTC:         MOV             A,DPL   ;Send address low
        CALL            BYTE_RTC_WR
        SETB            PortO_I2C_Data
        SETB            PortO_I2C_Clk
        JB      PortO_I2C_Data,WRITE_TIME
        CLR     PortO_I2C_Clk
_WR_DATA_RTC:        MOV     A,B             ;Send data
        CALL            BYTE_RTC_WR
        SETB            PortO_I2C_Data
        SETB            PortO_I2C_Clk
        JB      PortO_I2C_Data,WRITE_TIME
        CLR     PortO_I2C_Data
        CLR     PortO_I2C_Clk
        SETB            PortO_I2C_Clk           ;Stop bit
        SETB            PortO_I2C_Data
        RET
```

```
;/*****************************************************************************
;*          READ TIME FROM RTC 1 BYTE
;*   INPUT  : DPL of Address
;*   OUTPUT : A   of Data
;/*****************************************************************************
READ_TIME:   CLR     PortO_I2C_Data;Start bit
         CLR     PortO_I2C_Clk
_RD_TYPE_RTC:        MOV             A,#CONT_BYTE_RTC_W
         CALL            BYTE_RTC_WR
         SETB            PortO_I2C_Data
         SETB            PortO_I2C_Clk
         JB      PortO_I2C_Data,READ_TIME
         CLR     PortO_I2C_Clk
_RD_ADD_RTC:         MOV             A,DPL
         CALL            BYTE_RTC_WR
         SETB            PortO_I2C_Data
         SETB            PortO_I2C_Clk
         JB      PortO_I2C_Data,READ_TIME
         CLR     PortO_I2C_Clk
         SETB            PortO_I2C_Clk
         SETB            PortO_I2C_Data
         CLR     PortO_I2C_Data;Start bit
         CLR     PortO_I2C_Clk
_RD_ECHO_RTC:        MOV             A,#CONT_BYTE_RTC_R
         CALL            BYTE_RTC_WR
         SETB            PortO_I2C_Data
         SETB            PortO_I2C_Clk
         JB      PortO_I2C_Data,READ_TIME
_RD_DATA_RTC:        CLR     PortO_I2C_Clk
         CALL            BYTE_RTC_RD
```

```
        SETB        PortO_I2C_Data
        SETB        PortO_I2C_Clk
        CLR     PortO_I2C_Clk
        SETB        PortO_I2C_Clk  ;Stop bit
        SETB    PortO_I2C_Data
        RET
;/*****************************************************************************
;*      Byte RTC WRITE
;*      INPUT: Acc
;/*****************************************************************************
BYTE_RTC_WR:    PUSH        02H
        MOV         R2,#08H         ;Data 8 Bits
_BYTE_RTC_WR:   RLC     A
        MOV         PortO_I2C_Data,C
        SETB        PortO_I2C_Clk
        DB      0,0,0,0,0,0,0,0 ;8 NOP
        CLR     PortO_I2C_Clk
        DB      0,0,0,0,0,0,0,0 ;8 NOP
        DJNZ        R2,_BYTE_RTC_WR
        POP     02H
        RET

;/*****************************************************************************
;*      Byte RTC READ
;*   OUTPUT: Acc
;/*****************************************************************************
BYTE_RTC_RD:    PUSH    02H
        MOV     R2,#08H         ;Data 8 Bits
_BYTE_RTC_RD:   SETB    PortO_I2C_Clk
        DB      0,0,0,0,0,0,0,0 ;8 NOP
```

```
        MOV    C,PortO_I2C_Data
        CLR    PortO_I2C_Clk
        DB     0,0,0,0,0,0,0,0    ;8 NOP
        RLC    A
        DJNZ   R2,_BYTE_RTC_RD
        POP    02H
        RET

;/*************************************************************************
;         Initial Real Time Clock
;/*************************************************************************
Initial_RTC:    MOV    DPL,#00H        ; Sec 59(BCD)
        MOV    R7,#59H
        CALL   _Initial_RTC
        MOV    DPL,#01H        ; Min 59(BCD)
        MOV    R7,#59H
        CALL   _Initial_RTC
        MOV    DPL,#02H        ; Hour 23(BCD)
        MOV    R7,#23H

        CALL   _Initial_RTC
        MOV    DPL,#04H        ; Date 31(BCD)
        MOV    R7,#31H
        CALL   _Initial_RTC
        MOV    DPL,#05H        ; Mount 12(BCD)
        MOV    R7,#12H
_Initial_RTC:   CALL   READ_Time     ; Get Time
        CLR    C
        XCH    A,R7
        SUBB   A,R7     ; Max-Now => NC If (Max >= Now)
```

```
        JNC     _Initial_RTC_nSv        ; Not_Save
        MOV     A,#00   ; Save Start@1
        CALL    Write_Time      ; Save Data
_Initial_RTC_nSv:       RET
```

โปรแกรมย่อย EEPROM.sub

```
;/*****************************************************************************
;/* I2C-BUS EEPROM 24xx Write */;
;/* Write Data to I2C-EEPROM  */;
;/* Input  : DPTR (Address)   */;
;/*        : ACC (Data)       */;
;/*****************************************************************************
;   EEP_WRITE
;/*****************************************************************************
EEP_WRITE:    PUSH    B                ; Save Register
              PUSH    ACC              ; Save Data to Write
\
EEP_WR0:   CLR    I2C_SCL             ; Initial Bus For Start
           SETB   I2C_SDA
           SETB   I2C_SCL
           NOP                        ; TSU:STA Delay (5us)
           NOP
           NOP
           NOP
           NOP
           NOP
           CLR    I2C_SDA             ; Start Condition
           NOP                        ; THD:STA Delay (4us)
           NOP
           NOP
           NOP
        ;/* Ready to Start 1st Byte Send */;
           MOV    B,#8                ; Write Control Byte 8-Bit Count
           MOV    A,#10101000B        ; 1010-Slave100,Write
```

```
EEP_WR1:   CLR    I2C_SCL          ;/* Change Data to SDA Signal */;
           SETB   I2C_SDA          ; Release Data Line
           RLC    A                ; Write MSB First
           MOV    I2C_SDA,C        ; Write Bit Data to SDA Signal
           NOP
           NOP
           NOP
           ;
           SETB   I2C_SCL          ; Stobe Data to I2C-EEPROM
           NOP                     ; T-High Delay (4uS)
           NOP
           NOP
           NOP
           DJNZ   B,EEP_WR1        ; Loop to Write Control Byte
           ;
           CLR    I2C_SCL          ;*/ Falling 9'th Clock For ACK */;
           SETB   I2C_SDA          ; Release Data Line
           NOP
           NOP
           NOP
           NOP
           SETB   I2C_SCL          ;*/ Rising 9'th Clock for Get ACK */;
           NOP
           CLR    CY
           JNB    I2C_SDA,EEP_WR2  ; If ACK Jump
           LJMP   EEP_WR0          ; Restart Write Start Condition
           ;
EEP_WR2:   MOV    B,#8             ; Write Addr High-Byte Count
           MOV    A,DPH            ; Address High Byte
           ;
```

```
EEP_WR3:   CLR    I2C_SCL          ;/* Change Data to SDA Signal */;
           SETB   I2C_SDA                  ; Release Data Line
           RLC    A                        ; Write MSB First
           MOV    I2C_SDA,C                ; Write Bit Data to SDA Signal
           NOP
           NOP
           NOP
       ;
           SETB   I2C_SCL          ; Stobe Data to I2C-EEPROM
           NOP                     ; T-High Delay (4uS)
           NOP
           NOP
           NOP
       DJNZ   B,EEP_WR3            ; Loop to Write Addr High-Byte
       ;
           CLR    I2C_SCL          ;*/ Falling 9'th Clock For ACK */;
           SETB   I2C_SDA          ; Release Data Line
           NOP
           NOP
           NOP
           NOP
       SETB   I2C_SCL              ;*/ Rising 9'th Clock for Get ACK */;
       NOP
       CLR    CY
       JNB    I2C_SDA,EEP_WR4      ; If ACK Jump
       SETB   CY                   ; Not ACK From EEPROM
       POP    ACC
       LJMP   EEP_WR8
       ;
```

```
EEP_WR4:   MOV     B,#8                ; Write Addr Low-Byte Count
           MOV     A,DPL               ; Address Low Byte
       ;
EEP_WR5:   CLR     I2C_SCL             ;/* Change Data to SDA Signal */;
           SETB    I2C_SDA             ; Release Data Line
           RLC     A                   ; Write MSB First
           MOV     I2C_SDA,C           ; Write Bit Data to SDA Signal
           NOP
           NOP
           NOP
       ;
           SETB    I2C_SCL             ; Stobe Data to I2C-EEPROM
           NOP                         ; T-High Delay (4uS)
           NOP
           NOP
           NOP
           NOP
       DJNZ    B,EEP_WR5               ; Loop to Write Addr Low-Byte
       ;
       CLR     I2C_SCL                 ;*/ Falling 9'th Clock For ACK */;
       SETB    I2C_SDA                 ; Release Data Line
       NOP
       NOP
       NOP
       NOP
       SETB    I2C_SCL                 ;*/ Rising 9'th Clock for Get ACK */;
       NOP
       CLR     CY
       JNB     I2C_SDA,EEP_WR6         ; If ACK Jump
       SETB    CY                      ; Not ACK From EEPROM
       POP     ACC
```

```
        LJMP    EEP_WR8
        ;
EEP_WR6:  MOV     B,#8              ; Write Data 8-Bit Count
          POP     ACC               ; Restore Data For Write
        ;
EEP_WR7:  CLR     I2C_SCL           ;/* Change Data to SDA Signal */;
          SETB    I2C_SDA           ; Release Data Line
          RLC     A                 ; Write MSB First
          MOV     I2C_SDA,C         ; Write Bit Data to SDA Signal
          NOP
          NOP
          NOP
        ;
        SETB    I2C_SCL             ; Stobe Data to I2C-EEPROM
        NOP                         ; T-High Delay (4uS)
        NOP
        NOP
        NOP
        DJNZ    B,EEP_WR7           ; Loop to Write Data
        ;
        CLR     I2C_SCL             ;*/ Falling 9'th Clock For ACK */;
        SETB    I2C_SDA             ; Release Data Line
        NOP
        NOP
        NOP
        NOP
        SETB    I2C_SCL             ;*/ Rising 9'th Clock for Get ACK */;
        NOP
        CLR     CY
        JNB     I2C_SDA,EEP_WR8     ; If ACK Jump
```

```
        SETB    CY                      ; Not ACK From EEPROM
        ;
EEP_WR8:    CLR     I2C_SCL             ; Falling Stop Clock
        CLR     I2C_SDA
        NOP
        NOP
        SETB    I2C_SCL
        NOP                     ; TSU:STO (4us)
        NOP
        NOP
        NOP
EEP_WR9:    SETB    I2C_SDA             ; Stop Condition (SCL=1,SDA=1)
        JNB     I2C_SDA,EEP_WR9         ; Wait Until Stop
        POP     B                   ; Restore Register
        RET

;/*************************************************************************
;/* I2C-BUS EEPROM 24xx  Read */;
;/* Read Data From I2C-EEPROM */;
;/* -EEP_READ1 = Random Read  */;
;/*  Input  : DPTR (Address)  */;
;/*  Output : ACC (Data)      */;
;/* -EEP_READ2 = Current Read */;
;/*  Input  : None (Continue) */;
;/*  Output : ACC (Data)      */;
;/*************************************************************************
;  EEP_READ
;/*************************************************************************
EEP_READ1:  PUSH    B                   ; Save Register
EEP_RD0:    CLR     I2C_SCL             ; Initial Bus For Start
```

```
        SETB    I2C_SDA
        SETB    I2C_SCL
        NOP                     ; TSU:STA Delay (5us)
        NOP
        NOP
        NOP
        NOP
        CLR     I2C_SDA         ; Start Condition
        NOP                     ; THD:STA Delay (4us)
        NOP
        NOP
        NOP
        ;/* Ready to Start 1st Byte Send */;
        MOV     B,#8                    ; Write Control Byte 8-Bit Count
        MOV     A,#10101000B            ; 1010-Slave100,Write
        ;
EEP_RD1:    CLR     I2C_SCL             ;/* Change Data to SDA Signal */;
        SETB    I2C_SDA                 ; Release Data Line
        RLC     A                       ; Write MSB First
        MOV     I2C_SDA,C               ; Write Bit Data to SDA Signal
        NOP
        NOP
        NOP
        ;
        SETB    I2C_SCL             ; Stobe Data to I2C-EEPROM
        NOP                         ; T-High Delay (4uS)
        NOP
        NOP
        NOP
        DJNZ    B,EEP_RD1           ; Loop to Write Control Byte
```

```
        ;
        CLR    I2C_SCL              ;*/ Falling 9'th Clock For ACK */;
        SETB   I2C_SDA               ; Release Data Line
        NOP
        NOP
        NOP
        NOP
        SETB   I2C_SCL              ;*/ Rising 9'th Clock for Get ACK */;
        NOP
        CLR    CY
        JNB    I2C_SDA,EEP_RD2       ; If ACK Jump
        LJMP   EEP_RD0               ; Restart Read EEPROM
        ;
EEP_RD2:  MOV    B,#8                ; Write Addr High-Byte Count
        MOV    A,DPH                 ; Address High Byte
        ;
EEP_RD3:  CLR    I2C_SCL             ;/* Change Data to SDA Signal */;
        SETB   I2C_SDA               ; Release Data Line
        RLC    A                     ; Write MSB First
        MOV    I2C_SDA,C             ; Write Bit Data to SDA Signal
        NOP
        NOP
        NOP
        ;
        SETB   I2C_SCL               ; Stobe Data to I2C-EEPROM
        NOP                          ; T-High Delay (4uS)
        NOP
        NOP
        NOP
        DJNZ   B,EEP_RD3             ; Loop to Write Addr High-Byte
```

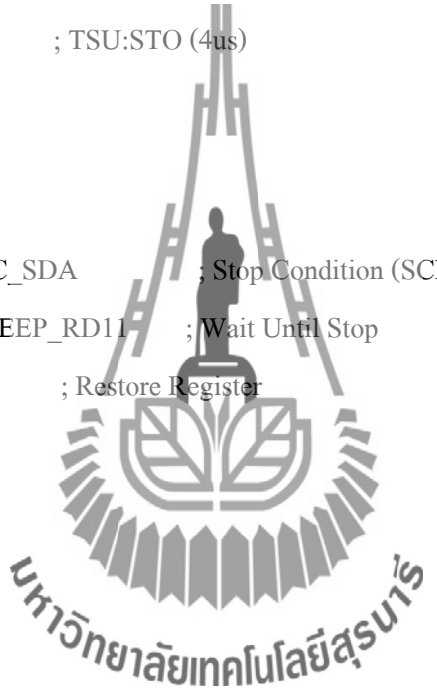
```
        ;
        CLR    I2C_SCL              ;*/ Falling 9'th Clock For ACK */;
        SETB   I2C_SDA              ; Release Data Line
        NOP
        NOP
        NOP
        NOP
        SETB   I2C_SCL              ;*/ Rising 9'th Clock for Get ACK */;
        NOP
        CLR    CY
        JNB    I2C_SDA,EEP_RD4      ; If ACK Jump
        SETB   CY                   ; Not ACK From EEPROM
        LJMP   EEP_RD10
        ;
EEP_RD4:  MOV    B,#8               ; Write Addr Low-Byte Count
        MOV    A,DPL                ; Address Low Byte
        ;
EEP_RD5:  CLR    I2C_SCL            ;/* Change Data to SDA Signal */;
        SETB   I2C_SDA              ; Release Data Line
        RLC    A                    ; Write MSB First
        MOV    I2C_SDA,C            ; Write Bit Data to SDA Signal
        NOP
        NOP
        NOP
        ;
        SETB   I2C_SCL              ; Stobe Data to I2C-EEPROM
        NOP                         ; T-High Delay (4uS)
        NOP
        NOP
        NOP
```

```
        DJNZ    B,EEP_RD5           ; Loop to Write Addr Low-Byte
        ;
        CLR     I2C_SCL             ;*/ Falling 9'th Clock For ACK */;
        SETB    I2C_SDA             ; Release Data Line
        NOP
        NOP
        NOP
        NOP
        SETB    I2C_SCL             ;*/ Rising 9'th Clock for Get ACK */;
        NOP
        CLR     CY
        JNB     I2C_SDA,EEP_RD6     ; If ACK Jump
        SETB    CY                  ; Not ACK From EEPROM
        LJMP    EEP_RD10
        ;
        ;/* Current Address Read Operation */;
EEP_READ2:  PUSH    B               ; Save Register
EEP_RD6:    CLR     I2C_SCL         ; Falling New Start Clock
        SETB    I2C_SDA             ; Initial Bus For Start
        SETB    I2C_SCL
        NOP                         ; TSU:STA Delay (5us)
        NOP
        NOP
        NOP
        NOP
        CLR     I2C_SDA             ; Start Condition
        NOP                         ; THD:STA Delay (4us)
        NOP
        NOP
        NOP
```

```
        ;
        MOV     B,#8                    ; Write Control Byte 8-Bit Count
        MOV     A,#10101001B            ; 1010-Slave100,Read
        ;
EEP_RD7:  CLR     I2C_SCL               ;/* Change Data to SDA Signal */;
        SETB    I2C_SDA                 ; Release Data Line
        RLC     A                       ; Write MSB First
        MOV     I2C_SDA,C               ; Write Bit Data to SDA Signal
        NOP
        NOP
        NOP
        ;
        SETB    I2C_SCL                 ; Stobe Data to I2C-EEPROM
        NOP                             ; T-High Delay (4uS)
        NOP
        NOP
        NOP
        DJNZ    B,EEP_RD7               ; Loop to Write Control Byte
        ;
        CLR     I2C_SCL                 ;*/ Falling 9'th Clock For ACK */;
        SETB    I2C_SDA                 ; Release Data Line
        NOP
        NOP
        NOP
        NOP
        SETB    I2C_SCL                 ;*/ Rising 9'th Clock for Get ACK */;
        NOP
        CLR     CY
        JNB     I2C_SDA,EEP_RD8         ; If ACK Jump
        SETB    CY                      ; Not ACK From EEPROM
```

```
        LJMP    EEP_RD10
        ;
EEP_RD8:    MOV     B,#8            ; Read Data 8-Bit Count
        CLR     CY
        CLR     A                   ; Clera Buffer For Read
        ;
EEP_RD9:    CLR     I2C_SCL          ;/* Wait to Receive SDA Signal */;
        SETB    I2C_SDA             ; Release Data Signal
        NOP
        NOP
        NOP
        NOP
        ;
        SETB    I2C_SCL             ; Read Data From I2C-EEPROM
        NOP
        MOV     C,I2C_SDA           ; Get Data From SDA Signal
        RLC     A
        NOP
        DJNZ    B,EEP_RD9           ; Loop to Read Data 8-Bit
        ;
        CLR     I2C_SCL            ;*/ Falling 9'th Clock For ACK */;
        SETB    I2C_SDA             ; Release Data Line
        NOP
        NOP
        NOP
        NOP
        SETB    I2C_SCL            ;*/ Rising 9'th Clock for Send ACK */;
        NOP
        NOP
        NOP
```

```
        NOP
        ;
EEP_RD10:  CLR    I2C_SCL            ; Falling Stop Clock
        CLR    I2C_SDA
        NOP
        NOP
        NOP
        SETB   I2C_SCL
        NOP                       ; TSU:STO (4us)
        NOP
        NOP
        NOP
EEP_RD11:  SETB   I2C_SDA           ; Stop Condition (SCL=1,SDA=1)
        JNB    I2C_SDA,EEP_RD11   ; Wait Until Stop
        POP    B                  ; Restore Register
        RET
```



ภาคผนวก

กรณีศึกษา

รถไฟชนรถบรรทุก จุดตัดทางรถไฟสถานีบ้านคูบัว

ทล.3338 กม.5+600

ข้อมูลและการตรวจสอบจุดเกิดเหตุ

เมื่อวันที่ 2 มีนาคม 2549 เวลา 08.00 น. หน่วยสืบสวนสาเหตุของอุบัติเหตุ (พื้นที่ศึกษา : กรุงเทพมหานครและปริมณฑล) ได้รับแจ้งจากศูนย์ปลอดภัยคมนาคม ว่าเกิดอุบัติเหตุรถไฟพุ่งชนรถบรรทุกบริเวณจุดตัดทางรถไฟบนทางหลวงหมายเลย 3338 บริเวณ กม.5+600 ([บริเวณสถานีรถไฟบ้านคูบัว อ.เมือง จ.ราชบุรี) ผลจากอุบัติเหตุทำให้มีผู้เสียชีวิต 6 ราย และบาดเจ็บ 26 ราย

ผลจากอุบัติเหตุ

1. ผู้เสียชีวิต 6 ราย
2. ผู้บาดเจ็บ 26 ราย
3. มูลค่าทรัพย์สินเสียหายกว่า 100 ล้านบาท

ข้อมูลจากการสอบถามผู้ประสบอุบัติเหตุ

<u>บุคคลที่ 1</u>

ผู้พบเห็นเหตุการณ์เป็นชาวบ้านที่ใช้เส้นทางที่เกิดเหตุและผ่านจุดเกิดเหตุอยู่เป็นประจำ ซึ่งได้ให้สัมภาษณ์มีใจความว่า “เวลาเกิดเหตุเป็นช่วงเช้า เวลาประมาณ 07.30 น. ซึ่งขณะนั้นเป็นเวลาที่รถไฟกำลังจะผ่านจุดตัดทางรถไฟ เมื่อตนมาถึงจุดเกิดเหตุเครื่องกั้นอัตมัตได้ถูกปิดลงแล้ว และมีรถยนต์จอดรอรถไฟผ่านทางอยู่ 3 คัน ในขณะนั้นเองรถบรรทุกที่จอดอยู่ด้านหลังตนได้แซงรถสวนทิศทางจราจรออกมาทางด้านขวา และฝ่าเครื่องกั้นซึ่งขณะที่รถบรรทุกกำลังเลี้ยวผ่านเครื่องกั้น ตนได้เห็นผู้ขับขี่รถบรรทุกได้บีบแตรและโบกมือให้รถจักรยานยนต์อีกคันหนึ่งซึ่งอยู่ฝั่งตรงข้ามของจุดตัดทางรถไฟให้หลบ จนรถไฟพุ่งชน ดังกล่าว”

<u>บุคคลที่ 2</u>

หลังเกิดเหตุ พขร. รถไฟได้ให้สัมภาษณ์แก่คณะทำงานมีใจความโดยสรุปว่า “ตนเป็นเวรผลัดที่ 2 ก่อเกิดการชน ตนได้ควบคุมรถไฟมาตามปกติ เมื่อถึงระยะประมาณ 1 กิโลเมตร ก่อนถึงจุดตัดทางรถไฟตนได้เปิดสัญญาณหวีด และได้ชะลอความเร็วของรถไฟลงเหลือประมาณ 80

กิโลเมตรต่อชั่วโมง ซึ่งเป็นหน้าที่ปฏิบัติโดยปกติของ พขร. เมื่อผ่านจุดตัดทางรถไฟ เมื่อถึงจุดเกิดเหตุตนได้เห็นรถบรรทุกข้ามฝ่าเครื่องกั้น และเกิดการชนในที่สุด”