



เครื่องสแกนลายนิ้วมือเพื่อเปิด - ปิดประตู



นายอรรถพล ศิลปกิจ วิศวกร ตรี	รหัสนักศึกษา	B4908067
นายชาญณรงค์ ประกอบดี วิศวกร ตรี	รหัสนักศึกษา	B5016839
นาย ภูริทัต สุธรรมมา วิศวกร ตรี	รหัสประจำตัว	B5007172

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 427494 โครงการวิศวกรรมโทรคมนาคม
และวิชา 427499 โครงการวิศวกรรมโทรคมนาคม
หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2546
สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี
ประจำภาคการศึกษาที่ 2 ปีการศึกษา 2553

การออกแบบสายอากาศแถวลำดับสะท้อนไมโครสตริปแบบสองลำคลื่น

คณะกรรมการสอบโครงการ

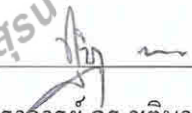


(ผู้ช่วยศาสตราจารย์ ดร.ปิยาภรณ์ กระจงนอก)
กรรมการ/อาจารย์ที่ปรึกษาโครงการ



(ผู้ช่วยศาสตราจารย์ ดร.รังสรรค์ ทองทา)
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี



(ผู้ช่วยศาสตราจารย์ ดร.ชุติมา พรหมมาก)
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้ยื่นรายงานโครงการฉบับนี้ เป็นส่วนหนึ่งของ
การศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม วิชา 427499 โครงการวิศวกรรม
โทรคมนาคม ประจำปีการศึกษา 2552

โครงการ เครื่องสแกนลายนิ้วมือเพื่อเปิด - ปิดประตู

โดย 1. นาย อรรถพล ศิลปกิจโกศล รหัสประจำตัว B4908067

2. นาย ชาญณรงค์ ประกอบดี รหัสประจำตัว B5016839

3. นาย ภูริทัต สุธรรมมา รหัสประจำตัว B5007172

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ร.อ. ดร. ประโยชน์ คำสวัสดิ์

สาขาวิชา วิศวกรรมโทรคมนาคม

ภาคการศึกษาที่ 2/2553

บทคัดย่อ

(Abstract)

โครงการนี้นำเสนอการออกแบบระบบเปิด - ปิดประตูด้วยการสแกนลายนิ้วมือ โดย การดำรงชีวิตในปัจจุบันนี้ ความก้าวหน้าทางด้านเทคโนโลยีเป็นปัจจัยหนึ่งที่ทำให้ชีวิตความมีศักยภาพของสังคมนั้น เช่น ในโรงงานอุตสาหกรรมมีการผลิตด้วยเครื่องจักรที่ทันสมัย หรือระบบการบริหารงานขององค์กร เป็นต้น เครื่องสแกนลายนิ้วมือก็เป็นเทคโนโลยีอีกทางเลือกหนึ่งที่น่าสนใจนำมาใช้งานภายในองค์กร เพราะประโยชน์จากลายนิ้วมือสามารถแสดงข้อมูลของบุคคลนั้นได้ ด้วยเหตุนี้จึงมีระบบที่นำเทคโนโลยีเครื่องสแกนลายนิ้วมือมาใช้ประโยชน์ในด้านต่างๆ เช่น การรักษาความปลอดภัย การบริหารทรัพยากรบุคคล

ผู้จัดทำจึงคิดปรับปรุงระบบการสแกนลายนิ้วมือเพื่อให้ได้ประโยชน์สูงสุด โดยจากการศึกษาความเป็นไปได้ของระบบ ทำให้เกิดแนวคิดในการใช้อุปกรณ์ดังนี้ 1. บอร์ด ไมโครคอนโทรลเลอร์ AVR 2. LCD 3. เครื่องสแกนลายนิ้วมือรุ่น fps-001 4. Electronic lock 5. ชุดรับส่งข้อมูลไร้สาย RS-232 และ 6. คอมพิวเตอร์ ระบบนี้เป็นการควบคุมแบบอัตโนมัติ โดยใช้บอร์ดไมโครคอนโทรลเลอร์ AVR เป็นตัวควบคุมหลักของระบบ โดยการใช้ประโยชน์ของระบบเครื่องสแกนลายนิ้วมือนี้ ผู้จัดทำมุ่งหวังเพื่อนำไปใช้ในการรักษาความปลอดภัยในองค์กร โดยประตูจะเปิดให้ก็ต่อเมื่อเป็นบุคคลที่มีข้อมูลอยู่ในระบบเท่านั้น และเพื่อให้ได้ระบบที่มีประสิทธิภาพมากขึ้นอีกทั้งยังประหยัดต้นทุนในการติดตั้งระบบ ด้วยการใช้เครื่องรับ - ส่งสัญญาณผ่านพอร์ต RS-232 แบบไร้สาย ในการส่งสัญญาณรับและส่งข้อมูลมาที่หน่วยประมวลผลกลาง เพื่อลดจำนวนอุปกรณ์ เช่น คอมพิวเตอร์ที่ตั้งติดตั้งไว้ตรงทางเข้าทุกจุดที่ต้องการเช่นเดิม

กิตติกรรมประกาศ (Acknowledgement)

การทำโครงการเรื่อง “เครื่องสแกนลายนิ้วมือเพื่อเปิด - ปิดประตู” ส่งผลให้คณะผู้จัดทำได้รับความรู้และประสบการณ์ในด้านต่างๆ มากมาย ไม่ว่าจะเป็นความรู้เกี่ยวกับหลักการทำงานของเครื่องสแกนลายนิ้วมือ การใช้งาน โปรแกรมต่างๆ เช่น ภาษาซี ฯ การใช้งานเครื่องอ่านลายนิ้วมือ Finger scan FPS-001 และหลักการทำงานของอุปกรณ์รับ - ส่งสัญญาณไร้สาย ขณะนี้โครงการดังกล่าวพร้อมทั้งรายงาน ได้สำเร็จแล้ว ซึ่งโครงการดังกล่าวนี้ได้รับความร่วมมือ คำปรึกษา ข้อเสนอแนะและการสนับสนุนจากบุคคลดังนี้

ผู้ช่วยศาสตราจารย์ ร.อ. ดร. ประโยชน์ คำสวัสดิ์ อาจารย์สาขา วิศวกรรมโทรคมนาคม ซึ่งท่านได้เป็นที่ปรึกษาโครงการดังกล่าวนี้ ข้าพเจ้าคณะผู้จัดทำโครงการทุกคน โกรธขอขอบพระคุณท่านอาจารย์เป็นอย่างยิ่งที่มีส่วนร่วมในการให้ข้อมูล คำแนะนำที่เป็นประโยชน์ต่อโครงการนี้ และเป็นที่ปรึกษาในการทำรายงานฉบับนี้จนเสร็จสมบูรณ์ ตลอดจนให้การดูแลและให้ความเข้าใจเกี่ยวกับการใช้งาน โปรแกรมภาษาซีและอุปกรณ์รับ-ส่งสัญญาณไร้สาย เป็นอย่างดีมาโดยตลอด ซึ่งข้าพเจ้าขอขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้ด้วย

นายชาญณรงค์ ประกอบดี

นายภูริทัต สุธรรมมา

นายอรรถพล สิลปกิจโกศล

คณะผู้จัดทำรายงาน

สารบัญ

เรื่อง	หน้า
บทคัดย่อ.....	ก
กิตติกรรมประกาศ.....	ข
สารบัญ.....	ค
สารบัญรูป.....	จ
บทที่ 1 บทนำ	
1.1 บทนำ.....	1
1.2 ความเป็นมาและความสำคัญ.....	1
1.3 วัตถุประสงค์ของโครงการ.....	2
1.4 ขอบเขตการทำงาน.....	2
1.5 ประโยชน์ที่คาดว่าจะได้รับ.....	2
1.6 ขั้นตอนการดำเนินงาน.....	2
บทที่ 2 ทฤษฎีและหลักการ	
2.1 บทนำ.....	3
2.2 ไมโครคอมพิวเตอร์.....	3
2.2.1 หน่วยประมวลผลกลาง.....	3
2.2.2 หน่วยความจำ.....	4
2.2.3 หน่วยอินพุต.....	4
2.2.4 หน่วยเอาต์พุต.....	4
2.3 ไมโครคอนโทรลเลอร์.....	5
2.4 เครื่องรับ – ส่งสัญญาณไร้สาย.....	5
2.4.1 Power supply.....	6
2.4.2 โหมดการทำงาน.....	8
2.4.3 ข้อเสนอแนะในการกำหนดค่าโครงร่าง.....	17
2.4.4 การเชื่อมต่อสัญญาณ RS232.....	19
2.4.5 ตัวอย่างการใช้งานโปรแกรม.....	20
2.4.6 ข้อสังเกตในการกำหนดโครงร่าง.....	24
2.5 อุปกรณ์ตรวจสอบภาพพิมพ์ลายนิ้วมือ.....	25
2.5.1 อุปกรณ์เครื่องสแกนลายนิ้วมือรุ่น FPS-001.....	25

สารบัญ (ต่อ)

เรื่อง	หน้า
2.6 รูปแบบการสื่อสาร.....	27
2.6.1 หลักการสื่อสาร.....	27
2.6.2 คำอธิบายคำสั่ง.....	29
บทที่ 3 การออกแบบโปรแกรมการทำภาพพิมพ์ลายนิ้วมือดิจิทัล	
3.1 บทนำ.....	32
3.2 ออกแบบ Hardware.....	33
3.3 ออกแบบ Software.....	34
3.4 การทำงาน.....	36
3.4.1 ขั้นตอนการใช้งานการเปิดประตู.....	36
3.4.2 ขั้นตอนการใช้งานการลงทะเบียนลายนิ้วมือ.....	40
บทที่ 4 การทดสอบเครื่องสแกนลายนิ้วมือเพื่อเปิด - ปิดประตู	
4.1 บทนำ.....	46
4.2 การทดสอบการเก็บข้อมูลผู้ใช้งาน.....	46
4.3 การทดสอบการเก็บข้อมูลและส่งข้อมูลผ่านเครื่องส่งสัญญาณไร้สาย...	48
4.4 การทดสอบการใช้งานเครื่องสแกนลายนิ้วมือเพื่อเปิด-ปิดประตูในการใช้งานจริง.....	49
บทที่ 5 สรุปผลและข้อเสนอแนะ	
5.1 บทนำ.....	52
5.2 สรุปผลการทดสอบเครื่องสแกนลายนิ้วมือเพื่อเปิด - ปิดประตู...	52
5.3 ข้อเสนอแนะ.....	52
ประวัติผู้เขียน.....	53
บรรณานุกรม.....	54
ภาคผนวก ก การใช้งานเบื้องต้น Code vision AVR.....	55
ภาคผนวก ข โปรแกรมคอมพิวเตอร์ที่ใช้งาน.....	60

สารบัญรูป

เรื่อง	หน้า
รูปที่ 2.1 การต่อสายสัญญาณ RS232 เพื่อใช้แหล่งจ่ายไฟจากบอร์ดไมโครคอนโทรลเลอร์ของ อีทีที.....	7
รูปที่ 2.2 การต่อ แหล่งจ่ายไฟรุ่น “ACH-4E” จากภายนอกให้กับเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0.....	8
รูปที่ 2.3 การต่อสายสัญญาณ RS232 เพื่อใช้กับเครื่องส่งสัญญาณไร้สายในโหมดภาครับและภาคส่ง.....	11
รูปที่ 2.4 แสดง รูปโปรแกรมที่ใช้สำหรับกำหนดค่า Configuration ของ ET-RF24G V2.0.....	14
รูปที่ 2.5 การต่อสายส่งสัญญาณ RS232 เพื่อใช้งานกับบอร์ดไมโครคอนโทรลเลอร์ในโหมดการทำงานปกติ.....	19
รูปที่ 2.6 การใช้งาน โปรแกรม Hyper terminal.....	20
รูปที่ 2.7 การใช้งาน โปรแกรม Hyper terminal.....	21
รูปที่ 2.8 การใช้งาน โปรแกรม Hyper terminal.....	22
รูปที่ 2.9 การใช้งาน โปรแกรม Hyper terminal.....	23
รูปที่ 2.10 Fingerprint sweep 001.....	25
รูปที่ 2.11 โครงสร้างบอร์ดเครื่องสแกนลายนิ้วมือรุ่น FPS-001.....	26
รูปที่ 3.1 แผนภาพการทำงานของตัวสแกนลายนิ้วมือ.....	33
รูปที่ 3.2 แผนภาพ Flow chart แสดงการทำงานของระบบสแกนลายนิ้วมือ....	34
รูปที่ 3.3 สถานะพร้อมใช้งาน.....	36
รูปที่ 3.4 สถานะสแกนผิดพลาด.....	37
รูปที่ 3.5 สถานะสแกนลายนิ้วมือผ่าน.....	37
รูปที่ 3.6 การบันทึกเวลาเข้า-ออก ของผู้ใช้.....	38
รูปที่ 3.7 สถานะสแกนผิดพลาดแล้วหลุดออกจาก Loop.....	38
รูปที่ 3.8 สถานะสแกนลายนิ้วมือผิดพลาด (ลายนิ้วมือเสียหาย).....	38
รูปที่ 3.9 สถานะสแกนลายนิ้วมือผิดพลาด (ลายนิ้วมือเล็กลงไป).....	39
รูปที่ 3.10 การสั่งบันทึกชื่อและ ID.....	41
รูปที่ 3.11 สถานะการส่งคำสั่งลบค่าใน ID:6.....	41
รูปที่ 3.12 สถานะบอกว่าพร้อมลงทะเบียน.....	41

สารบัญรูป (ต่อ)

เรื่อง	หน้า
รูปที่ 3.13 สถานะการร้อบลายนิ้วมือจาก Finger print.....	42
รูปที่ 3.14 สถานะการเก็บลายนิ้วมือสำเร็จ.....	43
รูปที่ 3.15 สถานะหมดเวลาในการบันทึก.....	43
รูปที่ 3.16 สถานะบอกว่าลายนิ้วมือมีความผิดพลาด.....	43
รูปที่ 3.17 สถานะบอกว่าสแกนเร็วเกินไป.....	44
รูปที่ 3.18 สถานะบอกลายนิ้วมือเล็กเกินไป.....	44
รูปที่ 3.19 สถานะบอกว่าใส่ลายนิ้วมือที่ 2	44
รูปที่ 3.20 สถานะบอกว่าใส่ลายนิ้วมือที่ 3.....	45
รูปที่ 3.21 สถานะบอกว่า ลายนิ้วมือที่ 3 ผ่านแล้ว.....	45
รูปที่ 3.22 สถานะบอกว่าลายนิ้วมือซ้ำกับลายนิ้วมือที่มีอยู่แล้ว.....	45
รูปที่ ก.1 การใช้งาน โปรแกรม Code vision AVR.....	55
รูปที่ ก.2 การใช้งาน โปรแกรม Code vision AVR.....	55
รูปที่ ก.3 การใช้งาน โปรแกรม Code vision AVR.....	56
รูปที่ ก.4 การใช้งาน โปรแกรม Code vision AVR.....	56
รูปที่ ก.5 การใช้งาน โปรแกรม Code vision AVR.....	57
รูปที่ ก.6 การใช้งาน โปรแกรม Code vision AVR.....	58
รูปที่ ก.7 การใช้งาน โปรแกรม Code vision AVR.....	58
รูปที่ ก.8 การใช้งาน โปรแกรม Code vision AVR.....	58
รูปที่ ก.9 การใช้งาน โปรแกรม Code vision AVR.....	59
รูปที่ ก.10 การใช้งาน โปรแกรม Code vision AVR.....	59

บทที่ 1

บทนำ

1.1 บทนำ

ในปัจจุบันการระบุตัวบุคคลสามารถทำได้หลายวิธี เช่น การดูหมายเลขบัตรประจำตัวประชาชน การดูรอยตำหนิ การดูรอยแผลเป็นตามร่างกาย เป็นต้น ซึ่งเป็นวิธีที่ง่ายและสะดวก ทำให้สามารถทำงานได้อย่างรวดเร็ว แต่ก็สามารถเกิดข้อผิดพลาดได้เช่นเดียวกัน เพราะวิธีการดังกล่าวสามารถลอกเลียนแบบได้ง่ายและมีความเป็นเอกลักษณ์ต่ำ เพราะฉะนั้นจึงหาวิธีการพิสูจน์และระบุตัวบุคคลที่มีประสิทธิภาพและแม่นยำมากขึ้นเพื่อเลี่ยงการเกิดข้อผิดพลาดต่างๆ และปัจจุบันนี้ระบบรักษาความปลอดภัยของการเปิด-ปิดประตู เป็นสิ่งจำเป็นอย่างยิ่งที่จะนำมาใช้เพื่อป้องกันการโจรกรรมของผู้ที่ประสงค์ไม่คิดต่อองค์กรหรือหน่วยงานในด้านต่างๆ จากการศึกษาพบว่าวิธีการพิสูจน์และระบุตัวตนของบุคคลโดยการอาศัยข้อมูลพื้นฐานทางด้านร่างกายของมนุษย์หรือที่เรียกว่า ไบโอมेटริก (Biometrics) ซึ่งสามารถทำได้หลายวิธี เช่น การตรวจสอบม่านตา การตรวจสอบคิเอ็นเอ การตรวจสอบโครงสร้างใบหน้า และการตรวจสอบลายนิ้วมือ เป็นต้น ซึ่งวิธีไบโอมेटริก เป็นวิธีการระบุตัวบุคคลที่มีประสิทธิภาพมากที่สุดในปัจจุบัน เพราะเป็นวิธีที่ยากต่อการเลียนแบบ มีความคงสภาพสูงและมีเอกลักษณ์ของแต่ละบุคคล จึงเป็นวิธีที่แพร่หลายในปัจจุบันและมีแนวโน้มที่จะเพิ่มมากขึ้นในอนาคต

ในด้านระบบรักษาความปลอดภัยก็เช่นกัน มีการนำเอาเทคโนโลยีทางด้านไบโอมेटริก (Biometrics) มาพัฒนาและใช้งานจริงในชีวิตประจำวัน เช่น การตรวจสอบลายนิ้วมือ เป็นต้น มีการคิดค้นและพัฒนาระบบตรวจสอบลายนิ้วมือเพื่อให้ใช้งานได้อย่างเต็มประสิทธิภาพและสมบูรณ์ที่สุด

1.2 ความเป็นมาและความสำคัญของปัญหา

ระบบรักษาความปลอดภัยที่ใช้กันอย่างแพร่หลายในปัจจุบันมีให้เลือกใช้มากมาย แต่แบบไหนที่จะประหยัดต้นทุนและมีประสิทธิภาพในการทำงานที่สมบูรณ์ที่สุด ผู้ทำโครงการได้เห็นถึงปัญหาที่เกิดขึ้นกับผู้ทำโครงการคือ การเช็คชื่อเข้าเรียน ปัญหาคือ นักศึกษาบางคนเข้าห้องเรียนช้ากว่าเวลาที่กำหนด ผู้ทำโครงการจึงได้คิดค้นหาวิธีการเช็คชื่อและเวลาเข้าห้องเรียนของนักศึกษาว่า นักศึกษาคนใดเข้าห้องเรียนช้ากว่าเวลาที่กำหนด ผู้ทำโครงการจึงได้ทำโครงการนี้ขึ้นมาเพื่อนำไปใช้ได้จริงในชีวิตประจำวัน

1.3 วัตถุประสงค์ของโครงการ

1. เพื่อศึกษาการทำงานของระบบตัวสแกนลายนิ้วมือ
2. เพื่อรักษาความปลอดภัยของทรัพย์สินภายในอาคาร
3. เพื่อศึกษาโปรแกรมควบคุมและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์
4. เพื่อตรวจสอบข้อมูลการเข้าใช้งานในสถานที่ที่ติดตั้งอุปกรณ์

1.4 ขอบเขตการทำงาน

1. อุปกรณ์สามารถสแกนลายนิ้วมือได้ถูกต้องแม่นยำ
2. โปรแกรมสามารถวิเคราะห์ข้อมูลได้อย่างถูกต้อง
3. อุปกรณ์สามารถรับส่งข้อมูลได้
4. อุปกรณ์สามารถแสดงผลได้จากการวิเคราะห์ข้อมูลของโปรแกรมได้

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้เรียนรู้ระบบการทำงานของอุปกรณ์สแกนลายนิ้วมือ
2. ได้เรียนรู้การเขียนโปรแกรมควบคุมและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ (Microcontroller)
3. ได้เรียนรู้การทำงานของอุปกรณ์รับ-ส่งสัญญาณด้วยระบบไร้สาย
4. ได้เรียนรู้การทำงานเป็นทีมและได้นำความรู้ที่ได้จากการศึกษาภาคทฤษฎีมาประยุกต์ใช้งานจริง

1.6 ขั้นตอนการดำเนินงาน

1. ปรึกษาอาจารย์ที่ปรึกษาโครงการเกี่ยวกับขอบเขตของโครงการที่จะทำ
2. ศึกษาข้อมูลเกี่ยวกับอุปกรณ์แต่ละตัวที่ต้องใช้โครงการ ได้แก่ เครื่องพิมพ์ลายนิ้วมือ , ตัวล็อคประตูอิเล็กทรอนิกส์ และไมโครคอนโทรลเลอร์
3. สั่งซื้ออุปกรณ์ที่เกี่ยวข้อง
4. ฝึกการใช้โปรแกรมไมโครคอนโทรลเลอร์
5. ประกอบวงจรอิเล็กทรอนิกส์
6. เขียนโปรแกรมไมโครคอนโทรลเลอร์เพื่อสั่งการไมโครคอนโทรลเลอร์ให้ทำงานตามวัตถุประสงค์
7. ทดลองใช้งานและแก้ไขสิ่งที่ผิดพลาด
8. จัดทำรูปเล่มรายงานของโครงการเพื่อเสนออาจารย์ประจำสาขาวิชา

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 บทนำ

จากความก้าวหน้าของเทคโนโลยีจึงพบว่าอุปกรณ์บางส่วนที่นำมาควบคุมในเครื่องใช้ไฟฟ้าระบบวัดคุมทางอิเล็กทรอนิกส์ เป็นต้น จะใช้ไมโครคอนโทรลเลอร์ ดังนั้นด้านการศึกษาและการเรียนรู้จึงมีการพัฒนาทางด้านการเรียนการสอนวิชาไมโครคอนโทรลเลอร์ เราจึงได้ออกแบบชุดการทดลองเพื่อให้ผู้ที่สนใจได้ศึกษาและทำการทดลอง โดยใช้ภาษาแอสเซมบลีในการควบคุมการทำงานของบอร์ด ในการทดลองของเราจะทำให้ผู้ใช้ได้ฝึกต่ออุปกรณ์อิเล็กทรอนิกส์เอง ซึ่งทำให้เกิดความชำนาญในด้านฮาร์ดแวร์เพิ่มขึ้นด้วย เพราะบอร์ดของเรามี Photo Board ไว้สำหรับต่ออุปกรณ์อิเล็กทรอนิกส์เพิ่มเติม บอร์ดไมโครคอนโทรลเลอร์นี้ใช้อุปกรณ์ที่ต่อรวมในวงจรน้อยชิ้น ราคาถูก หาแหล่งข้อมูลได้ง่าย มีการพัฒนาประสิทธิภาพในการประมวลผลเทียบเท่าไมโครโพรเซสเซอร์ขนาด 8 บิต – 16 บิต และใช้ในการพัฒนากับงานที่ไม่ต้องการความซับซ้อนซึ่งจะทำให้มีความสะดวกมากขึ้นในงานออกแบบ และสำหรับในการเรียนการสอนและผู้ที่ยังสนใจสามารถนำไปทดลองที่บ้านได้

2.2 ไมโครคอมพิวเตอร์คืออะไร

เมื่อเราเริ่มต้นเรียนรู้เกี่ยวกับระบบคอมพิวเตอร์ เราจะพิจารณาได้อย่างไรว่าชิ้นงานที่เห็นอยู่นั้นเป็นระบบไมโครคอมพิวเตอร์หรือไม่ ให้เราพิจารณาได้จาก องค์ประกอบ ของชิ้นงาน ซึ่งมี ส่วนประกอบที่สำคัญดังนี้

2.2.1 หน่วยประมวลผลกลาง (Central processing unit, CPU)

มีคุณสมบัติหลัก คือการประมวลผลข้อมูลการคำนวณทางคณิตศาสตร์และลอจิก เรารู้จักกันดีในชื่อ ไมโครโพรเซสเซอร์ เช่น Intel pentium ฯลฯ ซึ่งเป็นของบริษัท Intel หรืออาจเป็น CPU รุ่นเก่าที่มีขนาด 8 บิต เช่น เบอร์ Z80 ที่เป็น CPUของบริษัท ZILOG เป็นต้น

2.2.2 หน่วยความจำ (Memory unit)

2.2.2.1 หน่วยความจำรอม (Read only memory, ROM)

เป็นหน่วยความจำแบบถาวรที่มีการบันทึกข้อมูลไว้ล่วงหน้าก่อนแล้วไม่สามารถเปลี่ยนแปลงข้อมูลหรือคำสั่งใด ๆ ได้อีก ตัวอย่างเช่น ไอซีที่เป็นไบออส (BIOS) ของคอมพิวเตอร์ ในขณะที่เริ่มเปิดเครื่องไมโครคอมพิวเตอร์ครั้งแรก สังเกตได้ว่าจะมีการแสดงชื่อผู้ผลิตของบริษัท หรือ คุณสมบัติของเครื่องบนหน้าจอคอมพิวเตอร์ ถ้าหากเป็นเครื่องเล่นวีดีโอ จะเป็นตัวอักษรที่ทำหน้าที่แสดงผลเพื่อ บอกให้ตั้งค่าข้อมูลต่างๆ ซึ่งไม่ว่าจะปิดแล้วเปิดก็ครั้งตัวอักษรเดิมนั้นจะยังคงอยู่

2.2.2.2 หน่วยความจำแรม (Random access memory, RAM)

คือ หน่วยความจำที่ใช้เก็บข้อมูลไว้เพียงชั่วคราวอาจเป็นข้อมูลที่ CPU ต้องการประมวลผลในขณะนั้น และเมื่อ CPU ประมวลผลเรียบร้อยแล้วอาจลบหรือเปลี่ยนข้อมูลได้ บางครั้งเมื่อหยุดการจ่ายไฟให้กับวงจรจะทำให้ข้อมูลสูญหายไปได้ในทันที ตัวอย่าง เช่น ขณะที่เรากำลังพิมพ์งานแต่ยังไม่ได้บันทึกข้อมูลไว้ในส่วนใด ข้อมูลนี้จะถูกเก็บไว้ที่หน่วยความจำแรมก่อนหากเกิดเหตุการณ์ไฟฟ้าดับ จะทำให้ข้อมูลสูญหายไป หรือการเก็บค่าของเวลาและอุณหภูมิของเครื่องไมโครเวฟ ที่สามารถเปลี่ยนแปลงค่าได้ตลอด หน่วยความจำแบบแรมนี้จะแตกต่างกับหน่วยความจำแบบรอม โดยหน่วยความจำแบบรอมจะไม่สามารถแก้ไข ข้อมูลได้ในขณะนั้น ในขณะที่หน่วยความจำแบบแรมไม่สามารถเก็บค่าข้อมูลไว้ได้ตลอดคั้งนั้นหากต้องการให้ข้อมูลคงอยู่ต้องใช้แบตเตอรี่สำรองไฟไว้

2.2.3 หน่วยอินพุต (Input unit)

เป็นหน่วยที่ใช้สำหรับการรับสัญญาณข้อมูลจากภายนอกเช่น คีย์บอร์ด สแกนเนอร์ หรือที่รับสัญญาณมาจากอุปกรณ์เซนเซอร์ (Sensor) ซึ่งอาจเป็นค่าแรงเสียดทานของล้อรถยนต์ขณะเบรก การกดปุ่มสวิทช์ช่วงเวลาของวีดีโอเทป ฯลฯ กล่าวได้ว่าส่วนที่เป็นอินพุต คือส่วนที่ทำหน้าที่ป้อนข้อมูล

2.2.4 หน่วยเอาต์พุต (Output unit)

เป็นหน่วยที่ใช้สำหรับการแสดงผลของข้อมูลเช่นจอคอมพิวเตอร์ เครื่องคิดเลข เครื่องตัดสติ๊กเกอร์หรืออุปกรณ์ประเภทแอลอีดี (LED) ลำโพง มอเตอร์ รีเลย์ หลอดไฟ ฯลฯ

2.3 ไมโครคอนโทรลเลอร์คืออะไร

ปัจจุบันการพัฒนาและการแข่งขันทางด้านเทคโนโลยีผลิตชิ้นส่วนสารกึ่งตัวนำ ที่นำไปสร้างเป็นไอซีมี ประสิทธิภาพสูงมากขึ้นและมีเทคโนโลยีที่เกิดจากการผลิตของบริษัทต่างๆซึ่งส่งผลให้การผลิตชิปไอซีมีขนาดที่เล็กลง แต่มีประสิทธิภาพและคุณสมบัติต่างๆมากขึ้น ไอซีที่ถูกสร้างเป็นแบบ LSI (Large scale integrate circuit) เป็นเทคโนโลยีการสร้างโดยการนำเอาทรานซิสเตอร์จำนวนมากมาสร้างเป็นไอซีดิจิทัลที่ซับซ้อน โดยทำขึ้นเพื่อหน้าที่เป็นหน่วยประมวลผลข้อมูล หรือเรียกว่าไมโครโพรเซสเซอร์ ที่มีคุณสมบัติหลัก คือการประมวลผลข้อมูล การคำนวณทางคณิตศาสตร์ และลอจิก ถ้าหากมีการติดต่อกับหน่วยความจำที่เป็นแบบแรมแบบรวม หรืออุปกรณ์ภายนอกที่เป็นอินพุต-เอาต์พุตต้องมีการต่ออุปกรณ์อื่นๆ ร่วมด้วย เพื่อทำหน้าที่เลือกอุปกรณ์ในการติดต่อหรือวงจรถอดรหัส (Decoder) ซึ่งสามารถทำงานได้ภายใต้การควบคุมของโปรแกรม และในการที่เรา นำ ไมโครโพรเซสเซอร์มาเป็นตัวประมวลผลกลางมีหน่วยความจำแบบแรมพอร์ตอินพุตและเอาต์พุตที่เราเรียกว่า ไมโครคอมพิวเตอร์ เป็นสิ่งไม่คุ้มกับการลงทุนหากนำมาใช้ในงานควบคุมขนาดเล็ก และอาจต้องใช้เนื้อที่มาก ในการออกแบบ ดังนั้นการพัฒนาทางด้านเทคโนโลยีในการสร้างชิป จึงมีการรวบรวมคุณสมบัติที่ต้องการใช้งานมาอยู่ในตัวเดียวกันคือมีองค์ประกอบเกือบทุกอย่างของคอมพิวเตอร์อยู่ในตัว ไอซี ที่เราเรียกว่า ไมโครคอมพิวเตอร์แบบชิปเดี่ยวประกอบด้วยอุปกรณ์พื้นฐานเหมือนไมโครคอมพิวเตอร์ เช่นหน่วยประมวลผลกลางขนาดเล็ก (8บิต -16 บิต) และหน่วยประมวลผล ที่สามารถเข้าข้อมูลแบบบิตหน่วยความจำข้อมูลพื้นฐานแบบแรมขนาด 128 ไบต์ และบรรจุหน่วยความจำโปรแกรมประเภทรอม (บางเบอร์) สามารถใช้งานให้เป็นได้ทั้งอินพุตและเอาต์พุตมีวงจรสื่อสารอนุกรมแบบพูลดิวลิตีซ์ วงจร Counter/Timer ที่อยู่ใน สามารถต่ออุปกรณ์ที่ใช้ในการสร้างวงจรถ่ายสัญญาณนาฬิกาเช่น คริสตอล (Crystal) และตัวเก็บประจุก็สามารถใช้งานได้เป็นต้น เราเรียกกันทั่วไปว่า ไมโครคอนโทรลเลอร์ ดังนั้นเมื่อเราต้องการใช้งานควบคุมขนาดเล็ก เช่น เตาไมโครเวฟ เครื่องซักผ้า เครื่องเล่นวีดีโอเทปและเครื่องใช้ไฟฟ้าอื่น ๆ เราจึงนิยมนำไมโครคอนโทรลเลอร์มาใช้งาน เพราะมีทุกอย่างพร้อมในตัวเดียวกันประกอบกับมีขนาดที่เล็กอุปกรณ์ที่จะนำมาต่อรวมมีน้อยและเหมาะสำหรับใช้งานในการคำนวณที่ไม่ซับซ้อนมากนัก

2.4 เครื่องรับ - ส่งสัญญาณไร้สาย

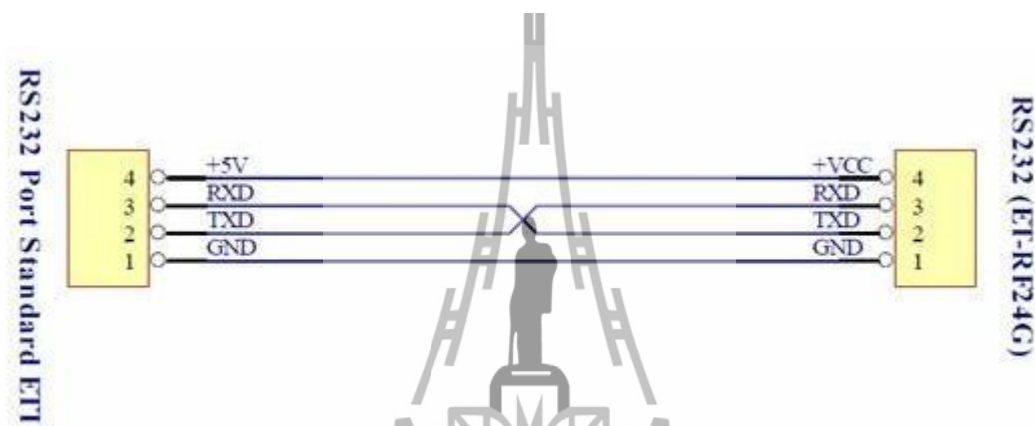
เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 เป็นชุด Signal converter สำหรับใช้แปลงสัญญาณระหว่าง RS232 และ RF-Wireless โดยในโหมดการทำงานของการส่งข้อมูล (Transmitter) จะทำหน้าที่ที่รับข้อมูลจากพอร์ตสื่อสารอนุกรม RS232 จากขา RX แล้วแปลงเป็นสัญญาณความถี่ (GFSK) ส่งออกไปในอากาศ และในทางกลับกันในโหมดการทำงานแบบรับ (Receiver) ชุด ET-

RF24G V2.0 ก็จะทำหน้าที่คอยตรวจจับข้อมูลที่อยู่ในรูปของสัญญาณความถี่ (GFSK) จากด้าน RF เพื่อแปลงกลับเป็นข้อมูลแบบ RS232 ส่งออกไปทางขา TX ได้ด้วย ซึ่งจะเห็นได้ว่าชุดแปลงสัญญาณนั้น สามารถนำไปต่อใช้งานร่วมกับพอร์ตสื่อสารอนุกรม แบบ RS232 เพื่อใช้งานในลักษณะของการสื่อสารอนุกรมแบบไร้สาย (Wireless transceiver) ได้โดยตรงโดยจะมีข้อดีกว่า คือสามารถรับส่งข้อมูลกันได้ในระยะทางที่ไกลกว่า RS232 หลายเท่าตัว และประการสำคัญ คือไม่จำเป็นต้องใช้สายสัญญาณที่เป็นตัวนำสัญญาณทางไฟฟ้าในการสื่อสารข้อมูลกัน ทำให้สามารถเปลี่ยนแปลง หรือเคลื่อนย้ายจุดรับส่งข้อมูลได้ตลอดเวลา ซึ่งถ้าเป็นการรับส่งข้อมูลด้วยระบบ RS232 แบบที่ใช้สายสัญญาณนั้น จะเกิดความยุ่งยากในการติดตั้งสายสัญญาณเป็นอย่างมากแต่อย่างไรก็ตามการรับส่งข้อมูลโดยใช้อากาศเป็นตัวกลางในการสื่อสารนั้น ก็มีข้อจำกัดบางประการเหมือนกัน โดยเฉพาะอย่างยิ่ง เรื่องความน่าเชื่อถือของข้อมูลที่รับส่งกัน ซึ่งมีโอกาสผิดพลาดหรือสูญหายได้เหมือนกัน เนื่องจากในการลำเลียงข้อมูลนั้นไม่ได้ใช้สายสัญญาณเป็นตัวกลางในการรับส่งข้อมูล แต่ใช้อากาศเป็นตัวกลางในการรับส่งข้อมูลแทน ซึ่งมีโอกาสที่ข้อมูลจะเกิดการรบกวนจากสัญญาณอื่นๆที่มีย่านความถี่ใกล้เคียงกันแล้วทำให้ข้อมูลผิดเพี้ยนไปได้บ้างเหมือนกัน ซึ่งระบบการจัดการข้อมูลของเครื่อง รับ - ส่งสัญญาณไร้สายนั้น มีระบบการเข้ารหัสและถอดรหัสข้อมูลที่มีความน่าเชื่อถืออยู่ในเกณฑ์ที่ดีว่าดี โดยข้อมูลแต่ละ Byte ที่มีการรับส่งกันนั้น จะมีการตรวจสอบความถูกต้องของข้อมูลให้ด้วยแล้ว โดยข้อมูลที่รับได้จากด้าน RF นั้นรับประกันได้ว่าเป็นข้อมูลที่มีความถูกต้องแน่นอน แต่อย่างไรก็ตามการรับส่งข้อมูลนั้นมีโอกาสผิดพลาดในเรื่องของการสูญหายของข้อมูลบ้างเหมือนกัน เนื่องจากกลไกในการรับส่งข้อมูลของ เครื่องรับ - ส่งสัญญาณไร้สายนั้น จะมีการตรวจสอบข้อมูลทุก Byte ที่รับได้จาก RF เสมอ ซึ่งถ้าพบว่ามีความผิดพลาดเกิดขึ้นจะทิ้งข้อมูล Byte นั้นไป ซึ่งผู้ใช้ควรมีกลไกในการตรวจสอบข้อมูลที่รับส่งกันว่าครบถ้วนหรือไม่ด้วย ซึ่งหากพบว่ามีการสูญหายของข้อมูลเกิดขึ้นก็ให้ร้องขอให้มีการส่งข้อมูลนั้นซ้ำนั้นๆใหม่อีกครั้งหนึ่ง ก็จะสามารแก้ไขปัญหาดังกล่าวได้

2.4.1 Power Supply

สำหรับการต่อแหล่งจ่ายไฟให้กับเครื่อง รับ - ส่งสัญญาณไร้สาย นั้น จะสามารถเลือกต่อแหล่งจ่ายไฟให้กับตัวเครื่องได้ 2 ทางด้วยกัน โดยเครื่อง รับ - ส่งสัญญาณไร้สาย ต้องการไฟเลี้ยงวงจร ซึ่งเป็นแหล่งจ่ายกระแสตรง ขนาดประมาณ +5VDC ถึง +9VDC โดยจุดเชื่อมต่อแหล่งจ่ายไฟของเครื่องรับ - ส่งสัญญาณไร้สายนี้ สามารถเชื่อมต่อได้ 2 จุดด้วยกัน โดยผู้ใช้งานสามารถ

เลือกต่อแหล่งจ่ายไฟให้กับเครื่อง ET-RF24G V2.0 จุดใดจุดหนึ่งก็ได้ในกรณีที่น่าเครื่อง รับ - ส่ง สัญญาณไร้สาย ไปเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์รุ่นต่างๆของ อีทีที นั้นสามารถใช้ แหล่งจ่ายไฟจากบอร์ดไมโครคอนโทรลเลอร์ เพื่อจ่ายให้กับตัวเครื่อง รับ - ส่งสัญญาณไร้สาย ได้ทันที โดยไม่ต้องใช้แหล่งจ่ายไฟจากภายนอก เนื่องจากขั้วต่อสัญญาณ RS232 ของบอร์ด ไมโครคอนโทรลเลอร์รุ่นต่างๆ ของบริษัท อีทีที นั้น ได้จัดเตรียมแหล่งจ่ายไฟตรง ขนาด +5V เตรียมไว้ให้ด้วยแล้ว โดยผู้ใช้เพียงแต่นำสายสัญญาณ RS232 ซึ่งทำการต่อสายสัญญาณครบทั้ง 4 เส้น ดังรูปมาเชื่อมต่อก็สามารถใช้งานได้แล้ว



รูปที่ 2.1 การต่อสายสัญญาณ RS232 เพื่อใช้แหล่งจ่ายไฟจากบอร์ดไมโครคอนโทรลเลอร์ของ อีทีที

แต่สำหรับกรณีที่น่าเครื่อง รับ - ส่งสัญญาณไร้สาย ไปต่อใช้งานกับอุปกรณ์อื่นๆที่ไม่ได้มีการจัดเตรียมจุดต่อไฟเลี้ยงไว้ให้ด้วย ผู้ใช้จำเป็นต้องจัดหา Adapter จ่ายไฟจากภายนอกมาต่อให้กับ เครื่องรับ - ส่งสัญญาณไร้สาย ต่างหากด้วย โดยให้เลือกแหล่งจ่ายไฟที่มีขนาดแรงดันไฟตรง ประมาณ +5VDC และสามารถจ่ายกระแสได้ประมาณ 300mAเป็นอย่างน้อย ซึ่งในกรณีนี้ขอ แนะนำให้เลือกใช้ Power supply รุ่น “ACH-4E” ซึ่งเป็นแหล่งจ่ายไฟแบบ Switching power ใช้กับ ไฟบ้าน 220VAC และให้เอาพุตเป็นไฟกระแสตรง ขนาดประมาณ 5VDC / 750mA เพราะPower supply รุ่นนี้ สามารถใช้งานร่วมกับเครื่อง รับ - ส่งสัญญาณไร้สาย ได้อย่างต่อเนื่องเป็นเวลานานๆ โดยไม่เกิดความร้อนสะสมที่วงจร Regulate ของบอร์ดมากนัก ซึ่งถ้าผู้ใช้เลือกแหล่งจ่ายไฟรุ่น อื่นๆ ที่มีขนาดแรงดันสูงกว่า +5V มากๆ ซึ่งถึงแม้ว่าจะสามารถใช้งานร่วมกันกับเครื่อง รับ - ส่ง สัญญาณไร้สาย ได้ แต่ถ้ามีการใช้งานอย่างต่อเนื่องเป็นเวลานานๆแล้ว อาจทำให้เกิดความร้อน

สะสมที่ตัวไอซี Regulate มากเกินไป จนอาจทำให้ภาค Power ของเครื่องรับ - ส่งสัญญาณไร้สายหยุดจ่ายไฟทำให้เครื่องหยุดทำงานได้



รูปที่ 2.2 การต่อ แหล่งจ่ายไฟรุ่น “ACH-4E” จากภายนอกให้กับเครื่องรับ - ส่งสัญญาณไร้สาย

ET-RF24G V2.0

2.4.2 โหมดการทำงาน

สำหรับโหมดการทำงานของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 นั้นจะแบ่งออกเป็น 2 โหมด ด้วยกัน โดยการกำหนดโหมดการทำงานของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 นั้นจะกระทำผ่าน Switch เล็กโหมด ซึ่งอยู่ด้านใต้กล่อง โดยการเลือกโหมดการทำงานนั้นจะต้องกระทำให้เสร็จเรียบร้อยก่อนการจ่ายไฟให้กับเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ด้วยเสมอ เนื่องจากการทำงานของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 นั้นจะทำการตรวจสอบโหมดการทำงานของเครื่องจาก Switch เล็กโหมด เฉพาะในช่วงของการจ่ายไฟเลี้ยงให้เครื่องเริ่มต้นทำงานครั้งแรก (Power-on) เท่านั้น ซึ่งการเปลี่ยนแปลงตำแหน่งการทำงานของ Switch เล็กโหมด หลังจากทำการจ่ายไฟให้กับเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ไปแล้ว จะไม่มีผลต่อการทำงานของเครื่องแต่อย่างใด โดยการทำงานของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 นั้นจะมี LED แสดงสถานะการทำงานของเครื่องจำนวน 2 หลอด คือ LED POWER ซึ่งเป็น LED สีแดง โดยที่ LED POWER นี้จะติดสว่างให้เห็นตลอดเวลาที่

มีการจ่ายไฟเลี้ยงให้เครื่องทำงานอยู่ ส่วน LED อีกดวงหนึ่งนั้นจะเป็น LED สีเขียว ใช้แสดงสถานะการทำงานของเครื่อง ซึ่งเรียกว่า LED STATUS โดย LED STATUS นี้จะเกิดการกระพริบตามจังหวะของการรับส่งข้อมูลกันในแต่ละครั้ง โดยในสภาวะปรกตินั้น ถ้าเครื่องทำงานอยู่ใน RUN MODE หลอด LED STATUS จะดับอยู่ตลอดเวลาถ้าไม่มีการรับส่งข้อมูล แต่ถ้าตัวเครื่องทำงานอยู่ใน SETUP MODE หลอด LED STATUS จะติดอยู่ตลอดเวลาถ้าไม่มีการรับส่งข้อมูล โดยโหมดการทำงานของ ET-RF24G V2.0 จะมีอยู่ด้วยกัน 2 โหมด คือ

2.4.2.1 การใช้งานเครื่องส่งสัญญาณไร้สายในโหมดปรกติ

การใช้งานใน Run mode ซึ่งเป็นโหมดของการใช้งานตามปรกติของเครื่อง โดยเมื่อเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24GV2.0 เข้าทำงานในโหมดนี้แล้ว จะสังเกตเห็นหลอดไฟแสดงสถานะการทำงานของเครื่อง หรือ LED STATUS ดับอยู่ แต่เมื่อมีการ รับ หรือ ส่ง ข้อมูล เกิดขึ้นสถานะการทำงานของ LED STATUS จึงจะกระพริบตามจังหวะของการรับส่งข้อมูลนั้นๆ แต่ถ้ายังไม่มีการรับส่งข้อมูลกัน LED STATUS จะดับอยู่ตลอดเวลาสำหรับการทำงานใน Run mode นั้น จะแบ่งลักษณะการทำงานออกเป็น 3 แบบด้วยกัน โดยลักษณะการทำงานนี้ จะถูกกำหนดไว้แล้วใน Configuration ของเครื่องใน Setup mode ดังนั้นก่อนการใช้งานเครื่อง ในครั้งแรกจะต้องทำการกำหนดค่า Configuration ต่างๆให้เรียบร้อยเสียก่อน โดยเมื่อเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 เริ่มต้นเข้าทำงานใน Run mode แล้วมันจะทำการอ่านค่า Configuration ที่เก็บไว้ออกมา เพื่อใช้เป็นเงื่อนไขในการทำงานตามค่าที่ได้กำหนดไว้ โดยลักษณะการทำงานใน Run mode แบ่งออกเป็นดังนี้

2.4.2.1.1 การทำงานแบบภาครับอย่างเดียว

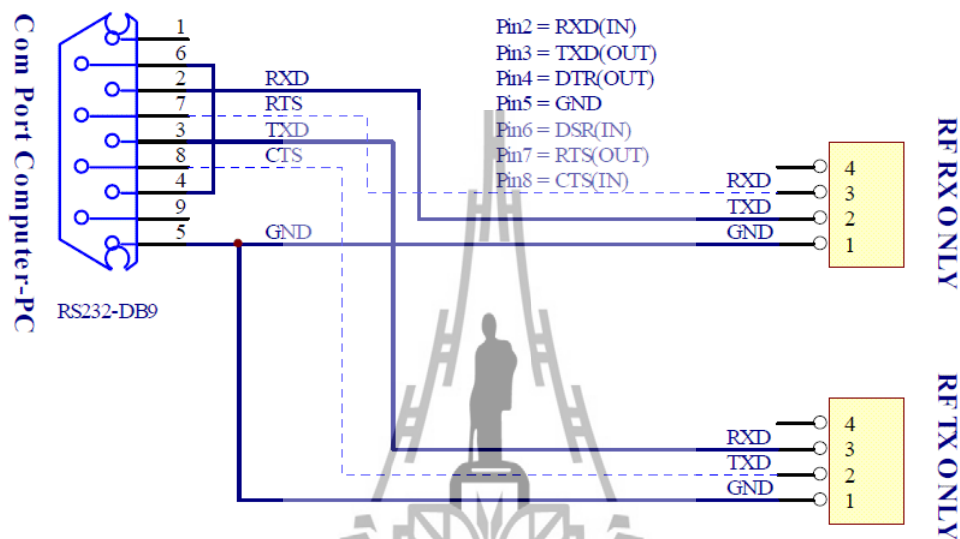
เป็นการทำงานแบบทิศทางเดียว โดยการทำงานในโหมดนี้ จะเป็นการรอรับข้อมูลความถี่แบบ GFSK จากด้าน RF แล้วเปลี่ยนเป็นข้อมูลอนุกรมส่งออกไปทางขา TX (Transmit) ของ RS232 โดยการทำงานจะวนรอบอยู่เช่นนี้ไปตลอด ซึ่งในการใช้งานเครื่อง รับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ในโหมดนี้จะต้องนำสัญญาณ TX(Transmit) ไปต่อกับขาสัญญาณ RX (Receive) ของอุปกรณ์ด้านตรงข้าม (RS232 ของคอมพิวเตอร์ PC) โดยในโหมดนี้ การทำงานของขาสัญญาณ RX ด้าน RS232 ของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะถูกเปลี่ยนหน้าที่เป็นสัญญาณ CTS (Clear To Send) สำหรับใช้ตรวจสอบความพร้อมในการส่งข้อมูลไปให้อุปกรณ์ด้าน

ตรงข้ามแทน ซึ่งในการใช้งานจะต้องนำสัญญาณนี้ไปต่อเข้ากับสัญญาณ RTS (Ready To Send) ของอุปกรณ์ด้านตรงข้าม โดยเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะทำการตรวจสอบสถานะของสัญญาณ RX ซึ่งในโหมดนี้เปรียบเสมือน CTS ว่ามีค่าเป็น “0” หรือไม่ โดยถ้าพบว่าเป็น “0” ก็จะส่งข้อมูลออกไปให้ทางขา TX แต่ถ้าพบว่ามีค่าเป็น “1” แสดงว่าอุปกรณ์ด้านตรงข้ามยังไม่พร้อมรับข้อมูลก็จะรอจนกว่าจะพบว่าสถานะของสัญญาณดังกล่าวมีค่าเป็น “0” จึงจะส่งข้อมูลออกไปให้ โดยเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะสามารถจัดเก็บข้อมูลไว้ใน Buffer เพื่อรอการส่งได้สูงสุด 64 Byte เท่านั้น ซึ่งถ้าในระหว่างที่รอความพร้อมอยู่นั้น มีข้อมูลด้าน RF ส่งเข้ามาเกินกว่า 64 Byte จะทำให้ข้อมูลที่เกินมานั้นสูญหายไป

2.4.2.1.2 การทำงานแบบภาคส่งอย่างเดียว

เป็นการทำงานแบบทิศทางเดียว โดยการทำงานในโหมดนี้จะมีลักษณะตรงกันข้ามกับ RF Receive Only กล่าวคือ เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะทำหน้าที่รอรับข้อมูลจากขา RX (Receive) ด้าน RS232 แล้วเปลี่ยนเป็นข้อมูลแบบ GFSK ส่งออกไปทางด้าน RF โดยการใช้งานเครื่องในโหมดนี้ จะต้องนำสัญญาณ TX (Transmit) ซึ่งเป็นขาส่งข้อมูลจาก RS232 ของอุปกรณ์ด้านตรงข้ามมาต่อเข้ากับขา RX (Receive) ของเครื่อง ET-RF24G V2.0 ส่วนขาสัญญาณ TX จะถูกเปลี่ยนหน้าที่เป็น RTS (Ready To Send) เพื่อใช้แสดงสถานะความพร้อมในการรับข้อมูลจากด้าน RS232 ซึ่งในการใช้งานจะต้องนำสัญญาณ TX ซึ่งในขณะนี้จะเปรียบเสมือนกับ RTS นำไปต่อเข้ากับสัญญาณ CTS (Clear to Send) ของอุปกรณ์ด้านตรงข้าม เพื่อใช้ในการตรวจสอบความพร้อมในการรับข้อมูล โดยอุปกรณ์ด้านตรงข้ามจะต้องทำการตรวจสอบสถานะของสัญญาณ RTS นี้เพื่อตรวจสอบความพร้อมในการรับข้อมูลของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ด้วย โดยถ้าเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 พร้อมรับข้อมูลจาก RS232 มันจะส่งสัญญาณ RTS ให้มีค่าเป็น “0” รอไว้ และเมื่อใดก็ตามที่การรับข้อมูลทางด้านของ RS232 มีจำนวนข้อมูลที่ยังไม่สามารถเปลี่ยนเป็น GFSK เพื่อส่งออกไปทางด้าน RF ได้ทันจนเกือบจะเต็ม Buffer แล้วเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะทำการส่งสัญญาณ RTS ให้มีค่าเป็น “1” ออกไปบอกให้อุปกรณ์ด้านตรงข้ามทราบเพื่อจะได้หยุดการส่งข้อมูลออกมา โดยอุปกรณ์ด้านตรงข้ามจะต้องหยุดการส่งข้อมูลและรอจนกว่าสถานะของสัญญาณ RTS จะกลับเป็น “0” จึงจะเริ่มส่งข้อมูลออกมาใหม่ ซึ่งหลังจากที่เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ส่งสัญญาณ RTS ด้วยค่า “1” ออกไปแล้ว จะยังคงสามารถรับข้อมูลได้เพิ่มเติมอีกไม่เกิน 16 Byte เท่านั้น ซึ่งถ้า

อุปกรณ์ด้านตรงข้ามยังส่งข้อมูลต่อเนื่องมาอีกจนเกินขนาดของ Buffer ที่เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะรับไว้ได้จะทำให้ข้อมูลที่เกินมานั้นเกิดการสูญหายได้โดยเราสามารถนำเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จำนวน 4 ชุด มาต่อใช้งานร่วมกัน เพื่อใช้งานในการรับส่งข้อมูลกันแบบ Full Duplex โดยแบ่งการใช้งานออกเป็น 2 ด้าน คือ ต้นทาง และ ปลายทาง ด้านละ 2 ชุด โดยแต่ละด้านให้กำหนดหน้าที่การทำงานเป็น RF Receive Only 1 ชุด และ RF Transmit Only อีก 1 ชุด



รูปที่ 2.3 การต่อสายสัญญาณ RS232 เพื่อใช้กับเครื่องส่งสัญญาณไร้สายในโหมดภาครับและภาคส่ง

2.4.2.1.3 การทำงานแบบภาครับและส่ง

เป็นการทำงานชนิด 2 ทิศทาง แบบ Half Duplex หรือ ผลัดกันรับผลัดกันส่ง ซึ่งสามารถใช้รับส่งข้อมูลระหว่างต้นทาง และ ปลายทาง ได้ โดยใช้เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ด้านละ 1 ชุด เท่านั้น เพียงแต่การรับส่งข้อมูลแบบนี้จะไม่สามารถส่งข้อมูลสวนทางกันได้ เหมือนกับแบบ Full Duplex แต่จะต้องใช้วิธีการผลัดกันรับข้อมูลและส่งข้อมูลแทน โดยเมื่อฝ่ายรับทำการรับข้อมูลได้จนครบแล้วจึงจะสลับหน้าที่เป็นฝ่ายส่งเพื่อส่งข้อมูลย้อนกลับไปโดยในโหมดนี้ เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะทำหน้าที่เป็นทั้ง ฝ่ายรับ และ ฝ่ายส่ง ข้อมูล แบบอัตโนมัติ โดยในสภาวะปรกติจะอยู่ในสภาวะของการรอรับข้อมูล ทั้งด้าน RF และ RS232 ซึ่งถ้าพบว่าข้อมูลส่งเข้ามาทางด้านของ RF ก็จะนำข้อมูลนั้นส่งออกไปทางด้านขา TX ของ RS232 ทันที และในทำนองเดียวกัน ถ้าพบว่าข้อมูลส่งเข้ามาทางด้าน RX ของ RS232 มันก็จะทำการรับ

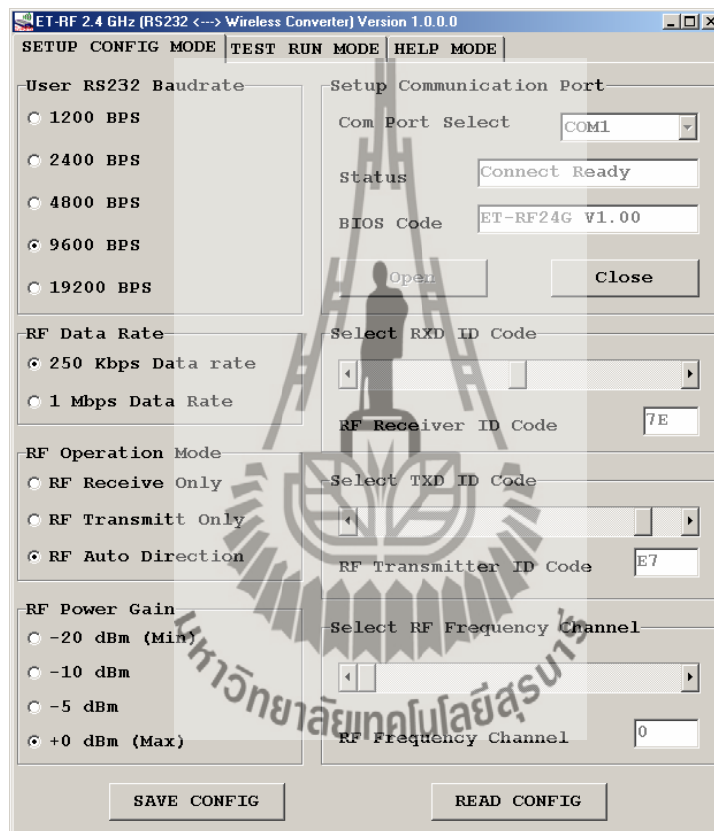
ข้อมูลนั้นจาก RS232 พร้อมกับเปลี่ยนทิศทางของอุปกรณ์ RF จากการรับข้อมูลให้ทำหน้าที่เป็นตัวส่งข้อมูลแทน เพื่อทำการส่งข้อมูลที่รับได้จาก RS232 ออกไปทาง RF ในทันที ซึ่งหลังจากที่เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ทำการสลับโหมดการทำงานของอุปกรณ์ด้าน RF จากการรับเป็นการส่งและทำการเริ่มต้นส่งข้อมูลออกไปทางด้าน RF เรียบร้อยแล้ว มันจะวนกลับไปตรวจสอบการรับข้อมูลจากด้าน RS232 อีกว่ายังมีข้อมูลส่งเข้ามาอีกหรือไม่ ถ้าพบว่ามีข้อมูลส่งเข้ามามีอีกก็จะทำการแปลงข้อมูลนั้นเพื่อส่งออกไปยังด้าน RF ต่อไปอีกจนกว่าการส่งข้อมูลด้าน RS232 จะสิ้นสุดลง ซึ่งข้อมูลด้าน RS232 ที่ส่งเข้ามานั้น ควรส่งอย่างต่อเนื่องโดยเมื่อเครื่อง ET-RF24G V2.0 ทำการส่งข้อมูลแต่ละ Byte ออกไปทางด้าน RF เรียบร้อยแล้วมันจะวนรอบรอรับข้อมูล Byte ถัดไปจาก RS232 ภายในเวลา 2.5 มิลลิวินาที ถ้าไม่พบข้อมูลส่งเข้ามาอีกภายในระยะเวลาดังกล่าวมันจึงจะทำการเปลี่ยนหน้าที่ของอุปกรณ์ด้าน RF ให้กลับมาทำหน้าที่เป็นการรอรับข้อมูลตามเดิม โดยในขณะที่อุปกรณ์ด้าน RF ถูกกำหนดให้เป็นฝ่ายส่งข้อมูลอยู่นั้น จะไม่สามารถทำการรับข้อมูลจาก RF ได้ ซึ่งถ้ามีการส่งข้อมูลเข้ามาในขณะนั้นก็จะไม่สามารถรับได้ โดยค่าเวลาที่จะใช้ในการสลับโหมดการทำงานของ RF จากฝ่ายส่งข้อมูลให้เป็นฝ่ายรับข้อมูลนั้น จะมีค่าเป็น 2.5 มิลลิวินาที ดังนั้นเมื่อฝ่ายรับสามารถรับข้อมูลได้ครบหมดแล้วก่อนที่จะทำการส่งข้อมูลเพื่อตอบกลับไปยังฝ่ายตรงข้ามนั้น ควรทำการหน่วงเวลาไว้ไม่น้อยกว่า 3 มิลลิวินาที นับจากรับข้อมูล Byte สุดท้ายได้เรียบร้อยแล้วจึงเริ่มต้นส่งข้อมูล Byte แรกย้อนกลับไป ซึ่งถ้าฝ่ายรับทำการส่งข้อมูลตอบกลับไปยังฝ่ายตรงข้ามเร็วกว่านี้อาจทำให้ฝ่ายตรงข้ามไม่สามารถรับข้อมูล Byte แรกได้ทันที สำหรับการใช้งานเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ในโหมด RF Auto Direction นี้ การ รับ และ ส่ง ข้อมูล ด้าน RS232 จะไม่มีการตรวจสอบความพร้อมของฝ่ายรับและ ส่ง ด้วยสัญญาณทางไฟฟ้า (CTS/RTS) เหมือนกับการใช้งานใน 2 โหมดที่ผ่านมาแล้ว โดยเมื่อมันสามารถรับข้อมูลจาก RF ได้ ก็จะทำการส่งข้อมูลนั้นออกไปทางขา TX(Transmit) ของ RS232 ในทันที โดยไม่สนใจว่า อุปกรณ์ที่ต่อไว้ด้าน RS232 จะพร้อมรับข้อมูลหรือไม่ ซึ่งถ้าด้าน RS232 ไม่พร้อมรับข้อมูลก็จะทำให้ข้อมูล Byte นั้นสูญหายไปทันที ซึ่งในการใช้งานนั้น ผู้ใช้ควรกำหนดค่าความเร็วในการรับส่งข้อมูลด้าน RS232 ที่จะใช้กับเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ทุกๆตัวด้วยค่าความเร็วที่เท่ากันด้วยเพื่อให้การรับและส่งข้อมูลเกิดความสัมพันธ์กันอย่างเหมาะสม

สำหรับความสามารถในการรองรับข้อมูลจาก RS232 ของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ในโหมดนี้ จะสามารถรับข้อมูลได้อย่างต่อเนื่องสูงสุด ไม่เกิน 64 Byte ดังนั้นในกรณีที่มีการส่งข้อมูลจากด้าน RS232 ด้วยข้อมูลจำนวนมากว่า 64 Byte ต่อเนื่องกันนั้น ควรทำการแบ่งข้อมูลออกเป็นชุดๆ โดยให้มีขนาดชุดละไม่เกิน 64 Byte ซึ่งหลังจากทำการส่งข้อมูลอย่างต่อเนื่องไปได้ 1 ชุด (64 Byte) แล้วควรทำการหน่วงเวลาไว้ชั่วขณะหนึ่งอย่างน้อย 1 มิลลิวินาที แล้วจึงเริ่มส่งข้อมูลชุดถัดไป สลับกับการหน่วงเวลา อย่างนี้เรื่อยๆ เพื่อให้เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 สามารถนำข้อมูลที่รับได้จากด้าน RS232 ส่งออกไปทางด้าน RF ได้ทัน ซึ่งถ้าทำการส่งข้อมูลอย่างต่อเนื่องโดยไม่มีการหน่วงเวลาเลยอาจทำให้ข้อมูลบาง Byte เกิดการสูญหายไปได้

2.4.2.2 การใช้งานเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ในโหมดการตั้งค่า

การใช้งานเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ใน Setup mode ซึ่งเป็นโหมดสำหรับใช้กำหนดค่า Configurationต่างๆ สำหรับควบคุมการทำงานของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ที่จะใช้ในขณะที่เครื่องทำงานอยู่ใน Run Mode โดยในการ Setup ค่า Configuration ต่างๆนั้นจะกระทำร่วมกับโปรแกรม “ET_RF24G_V1.EXE” ของ อีทีที ซึ่งเมื่อเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 เข้าทำงานในโหมด Setup แล้ว จะสังเกตเห็นหลอดไฟแสดงสถานะการทำงาน หรือ LED STATUS ติดสว่างค้างอยู่ตลอดเวลา แต่เมื่อมีการสั่งอ่านหรือเขียนข้อมูลกับบอร์ด สถานะการทำงานของ LED STATUS จึงจะกระพริบตามจังหวะของการรับส่งข้อมูล แต่ถ้ายังไม่มีการรับส่งข้อมูลกับ LED STATUS จะติดค้างอยู่ตลอดเวลา ซึ่งการกำหนดค่า Configuration ให้กับเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 นั้น จะต้องกระทำในขณะที่ตัวเครื่องทำงานอยู่ใน Setup mode เท่านั้น (เลือก Switch กำหนดโหมดไว้ทางด้าน Setup แล้วจ่ายไฟให้เครื่องเริ่มต้นทำงาน) โดยค่าของ Configuration ต่างๆนั้นจะถูกใช้สำหรับเป็นเงื่อนไขในการทำงานของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0ในขณะที่อยู่ใน Run mode ดังนั้นก่อนการเริ่มต้นใช้งานเครื่องในครั้งแรกนั้น จึงจำเป็นต้องทำอย่างที่จะต้องทำการกำหนดค่าของ Configuration ต่างๆให้ถูกต้องและตรงกับความต้องการที่จะใช้งานเสียก่อน โดยเมื่อทำการกำหนดค่าตัวเลือกต่างๆของ Configuration เรียบร้อยแล้ว ก็สามารถเปลี่ยนโหมดการทำงานของตัวเครื่องกลับเป็น Run Mode พร้อมกับการปิดไฟที่จ่ายให้กับตัวเครื่อง (Power-off) ชั่วขณะหนึ่ง จากนั้นจึงเริ่มต้นจ่ายไฟให้กับตัวเครื่องใหม่ (Power-on) ก็สามารถใช้งานเครื่องรับ - ส่งสัญญาณไร้

สาย ET-RF24G V2.0 ตามค่าของ Configuration ที่กำหนดไว้แล้วได้ทันทีโดยค่าตัวเลือกต่างๆของ Configuration ที่ได้กำหนดไว้แล้วจะถูกเก็บไว้ในตัวเครื่องอย่างถาวร ถึงแม้ว่าจะไม่ได้ทำการจ่ายไฟให้กับตัวเครื่องแล้วก็ตาม ดังนั้นเมื่อทำการกำหนดค่า Configuration ต่างๆเรียบร้อยแล้ว ถ้าไม่มีการเปลี่ยนแปลงเงื่อนไขการทำงานของตัวเครื่องต่างไปจากเงื่อนไขเดิมที่ได้กำหนดไว้แล้ว ก็ไม่จำเป็นต้องทำการกำหนดค่า Configuration ใหม่อีกแต่อย่างใด โดยทุกๆครั้งที่เริ่มต้นจ่ายไฟเข้าเครื่องในครั้งแรกนั้น การทำงานของเครื่องรับ – ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะเป็นไปตามเงื่อนไขที่กำหนดไว้ใน Configuration เสมอทุกๆครั้ง โดยคุณสมบัติของ Configuration ต่างๆนั้นมีดังนี้



รูปที่ 2.4 แสดง รูปโปรแกรมที่ใช้สำหรับกำหนดค่า Configuration ของ ET-RF24G V2.0

2.4.2.2.1 User RS232 Baud rate

ใช้สำหรับกำหนดค่าความเร็วในการรับส่งข้อมูลทางด้าน RS232 ของตัวเครื่อง ในขณะที่ทำงานอยู่ใน Run Mode ซึ่งสามารถกำหนดได้ 5 ค่าคือ 1200 BPS, 2400 BPS, 4800 BPS, 9600 BPS, 19200 BPS

2.4.2.2.2 RF Data Rate

ใช้สำหรับกำหนดความเร็วในการรับส่งข้อมูลทางด้าน RF ของ ET-RF24G V2.0 ซึ่งจะต้องกำหนดให้เครื่อง ET-RF24G V2.0 ทุกๆตัว ที่จะนำมาใช้ติดต่อกัน มีค่าอัตราความเร็วในการรับส่งข้อมูลด้าน RF หรือ RF Data Rate นี้มีค่าเท่ากันทั้งหมด ซึ่งถ้ากำหนดค่าความเร็วต่างกันจะไม่สามารถรับส่งข้อมูลกันได้ ซึ่งค่าอัตราความเร็วในการส่งข้อมูลนี้จะมีผลกระทบต่อระยะทางการรับส่งข้อมูลด้วย ซึ่งถ้าใช้ความเร็วในการส่งสูง (1Mbps) จะทำให้รัศมีการรับส่งข้อมูลได้ระยะทางสั้นลง แต่ถ้าใช้ความเร็วในการรับส่งข้อมูลที่ช้าลง (250Kbps) จะทำให้ได้รัศมีการรับส่งไกลขึ้น โดยค่า RF Data rate สามารถกำหนดได้ 2 ค่า คือ 250 Kbps, 1 Mbps

2.4.2.2.3 RF Operation mode

ใช้สำหรับกำหนดโหมดการทำงานของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ซึ่งสามารถกำหนดหน้าที่การทำงานได้ 3 แบบด้วยกันคือ

1. RF Receive only เป็นการกำหนดให้เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ทำหน้าที่เป็นฝ่ายรอรับข้อมูลทางด้าน RF เพื่อเปลี่ยนเป็นข้อมูลแบบ RS232 และส่งออกไปทางด้านขา TX ของ RS232 ตลอดเวลา

2. RF Transmit only เป็นการกำหนดให้เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ทำหน้าที่เป็นฝ่ายรอรับข้อมูลทางด้าน RS232 จากขา RX เพื่อเปลี่ยนเป็นข้อมูลแบบ GFSK และส่งออกไปทางด้าน RF ตลอดเวลา

3. RF Auto direction เป็นการกำหนดโหมดการทำงานแบบ Half Duplex 2 ทิศทาง ซึ่งสามารถสลับโหมดการทำงานระหว่างการรับและส่งข้อมูลได้เองโดยอัตโนมัติ โดยในโหมดการทำงานนี้ เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะรอตรวจสอบข้อมูลทั้งจากด้าน RS232 และด้าน RF อยู่ตลอดเวลา โดยถ้าได้รับข้อมูลจากด้าน RS232 ก็จะทำการแปลงแล้วส่งออกไปทางด้าน RF จากนั้นก็จะกำหนดให้ด้าน RF กลับมาเป็นฝ่ายรอรับข้อมูลตามเดิม และเมื่อได้รับข้อมูลจากด้าน RF ก็จะแปลงเป็นข้อมูลแล้วส่งออกไปทางด้าน RS232 โดยอัตโนมัติ

2.4.2.4 RF Power gain

เป็นการกำหนดกำลังส่งของวงจร RF Power ที่ใช้ในการส่งข้อมูล โดยค่า +0dBmเป็นค่ากำลังส่งสูงสุด ส่วน -20dBm เป็นค่ากำลังส่งต่ำสุด โดยสามารถกำหนดได้ 4 ระดับคือ -20dBm (กำลังส่งต่ำสุด), -10dBm, -5dBm, +0dBm (กำลังส่งสูงสุด)

2.4.2.5 RXD ID code

เป็นรหัส ID Code ของเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ในโหมดของการรับข้อมูลจาก RF โดยเมื่อเครื่อง ET-RF24G V2.0 ด้านส่งจะทำการส่งข้อมูลออกไปทาง RF นั้นจะมีการระบุหมายเลข ID Code ของด้านรับรวมไปกับชุดข้อมูลด้วยเสมอ โดยเมื่อเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ที่อยู่ทางด้านรับทำการรับข้อมูลจากด้าน RF ได้ อันดับแรกมันจะทำการเปรียบเทียบรหัส ID Code ที่รวมมากับข้อมูลที่รับมาได้ว่าตรงกับรหัสของ RXD ID Code ที่กำหนดไว้ในตัวมันหรือไม่ ซึ่งถ้าถูกต้องก็จะแยกเอาเฉพาะส่วนของข้อมูลที่รับเข้ามาได้เพื่อเปลี่ยนเป็นข้อมูลแบบ RS232 แล้วส่งออกไปทางด้าน TX ของ RS232 แต่ถ้ารหัส ID Code ที่รับมาได้ไม่ตรงกับรหัส RXD ID Code ที่กำหนดไว้ เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24GV2.0 จะทิ้งข้อมูลชุดนั้นไปทันที โดยค่า RXD ID Code นั้นสามารถกำหนดได้ 256 ค่าในรูปแบบของเลขฐานสิบหก (00H-FFH)

2.4.2.6 TXD ID code

เป็นรหัส ID Code ปลายทางที่จะส่งข้อมูลไปหา โดยที่เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ที่ถูกกำหนดให้ทำหน้าที่เป็นฝ่ายส่งข้อมูลนั้น เมื่อมันสามารถรับข้อมูลจาก RS232 ได้แล้ว มันจะทำการนำเอาข้อมูลนั้นไปเข้ารหัสรวมกับ TXD ID Code ที่กำหนดไว้ แล้วส่งออกไปทางด้าน RF โดยรหัสของ TXD ID Code นี้หมายถึง รหัส RXD ID Code ของฝ่ายรับที่ต้องการส่งข้อมูลไปหาตัวเอง โดยค่า TXD ID Code นั้นสามารถกำหนดได้ 256 ค่าในรูปแบบของเลขฐานสิบหก (00H-FFH)

2.4.2.7 RF Frequency channel

เป็นการกำหนดค่าของช่องความถี่ที่จะใช้ในการรับส่งข้อมูลกัน โดยสามารถเลือกกำหนดช่องความถี่ได้สูงสุดมากถึง 125 ช่อง (0-124) โดยการที่เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะทำการรับส่งข้อมูลกันได้นั้นจะต้องกำหนดช่องความถี่ที่ตรงกัน และ ใช้อัตราความเร็ว RF Data Rate ที่เท่ากันด้วย ซึ่งที่สามารถเลือกกำหนดช่องความถี่ RF Frequency Channel ได้นั้น จะมีประโยชน์เป็นอย่างมากในกรณีที่มีการใช้งานเครื่อง รับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จำนวนหลายๆกลุ่ม ในบริเวณพื้นที่ใกล้เคียงกันโดยให้กำหนดช่องความถี่ของเครื่อง รับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 กลุ่มที่จะสื่อสารข้อมูลร่วมกันไว้ที่ช่องความถี่เดียวกันส่วนกลุ่มอื่นๆก็ให้เลือกกำหนดช่องความถี่ที่แตกต่างกันออกไป เพื่อลดปัญหาการรบกวนกัน

2.4.3 ข้อเสนอแนะในการกำหนดค่าโครงร่าง

การกำหนดค่า Configuration ให้กับเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 นั้นสามารถเลือกกำหนดได้ตามความต้องการและจุดประสงค์ของการใช้งาน โดยแต่ละโหมดของการใช้งานนั้นจะมีค่า Configuration ที่เหมาะสมต่างกัน ซึ่งขอแนะนำวิธีการกำหนดค่า Configuration ดังแนวทางต่อไปนี้ - ความเร็วในการรับส่งข้อมูลด้าน RS232 หรือ User RS232 baud rate ที่ความเร็ว 19200 Bps นั้นเหมาะกับการใช้งานเครื่อง รับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 แบบ Receive Only หรือ Transmit Only ซึ่งมีการตรวจสอบความพร้อมของสัญญาณในการรับส่งข้อมูลกันด้วย แต่ถ้าต้องการใช้งานเครื่อง ET-RF24G V2.0 ในโหมด Auto Direction นั้น ควรกำหนดค่า User RS232 baud rate ไว้ที่ความเร็วไม่เกิน 9600 Bps จะดีที่สุด และควรกำหนดค่า baud rate ของทั้งสองฝ่ายให้มีค่าเท่ากันด้วย

ค่าความเร็วของการรับส่งข้อมูลด้าน RF หรือ RF Data rate ที่สามารถรับส่งข้อมูลกันได้ระยะทางไกลมากที่สุด และมีโอกาสผิดพลาดน้อยที่สุด คือ 250Kbps

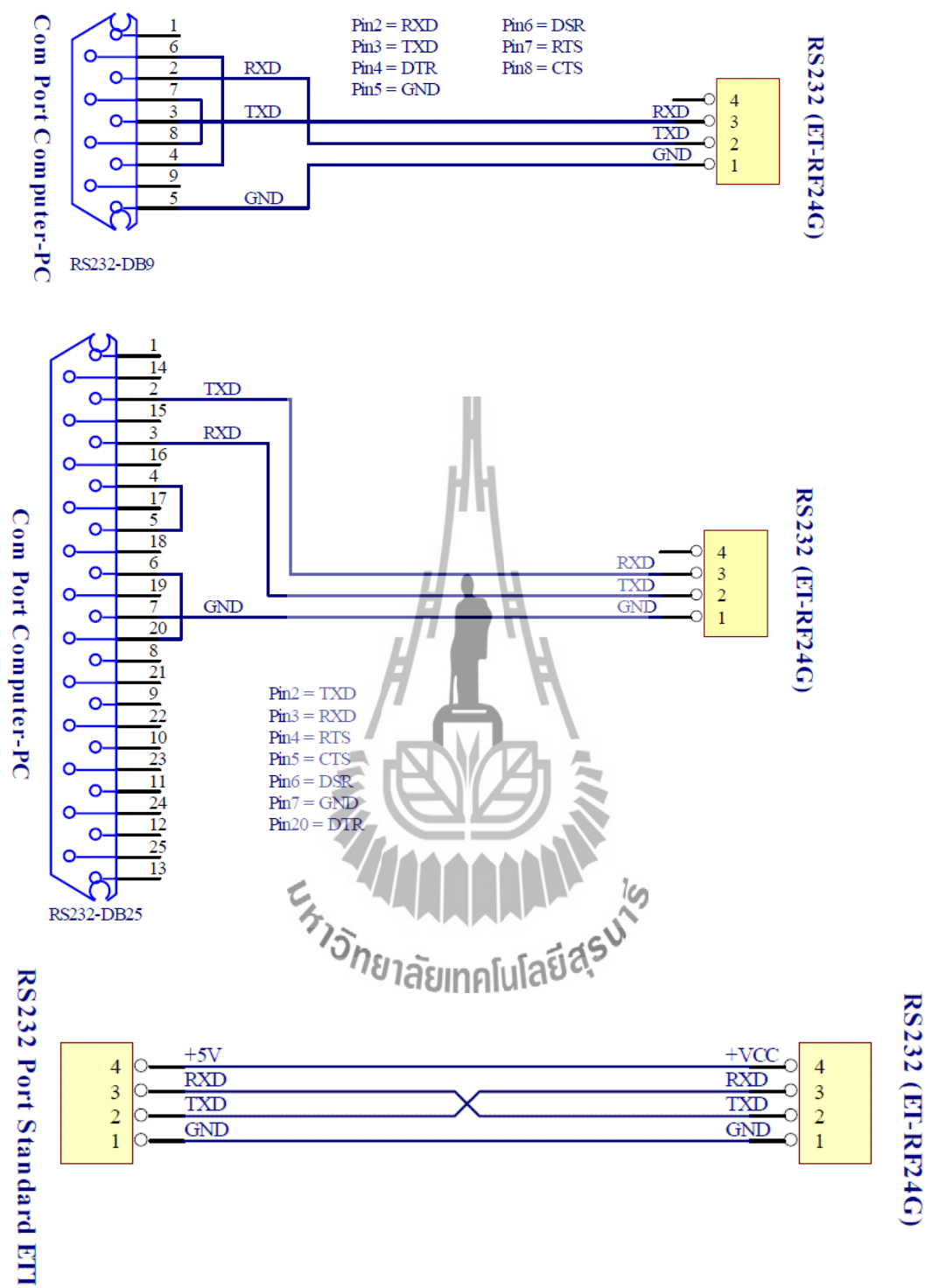
ค่า RF Power gain ที่ดีที่สุดคือ 0 dBm ซึ่งเป็นค่ากำลังส่งสูงสุด ซึ่งจะทำให้สามารถส่งข้อมูลได้ระยะทางไกลที่สุด แต่ถ้าระยะการรับส่งข้อมูลไม่ไกลกันมาก และมีการใช้งานเครื่อง รับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จำนวนหลายๆกลุ่มในพื้นที่ใกล้เคียงกัน ก็อาจทำการลดกำลังส่งให้ต่ำลงเพื่อลดปัญหาการรบกวนกันหรือกำหนดช่องความถี่ RF Frequency channel ให้ห่างกันมากๆ

ในกรณีที่มีการใช้เครื่อง รับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 หลายๆกลุ่มในพื้นที่ใกล้เคียงกัน ควรกำหนดช่องความถี่ในการใช้งาน หรือ RF Frequency channel ให้ห่างกันด้วยเพื่อป้องกันการรบกวนกัน

การใช้งานเครื่อง รับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 แบบ Auto Direction นั้น ถ้ามีการส่งข้อมูลจำนวนมากๆ ควรจัดแบ่งข้อมูลออกเป็นชุดๆ โดยให้มีขนาดข้อมูลชุดละไม่เกิน 64 Byte โดยในการส่งข้อมูลแต่ละชุดนั้นให้ทำการส่งข้อมูลอย่างต่อเนื่องโดยให้ข้อมูลแต่ละ Byte มีระยะเวลาห่างกันไม่เกิน 2.5mS เนื่องจากถ้าข้อมูลขาดหายไปนานกว่านี้ เครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 จะทำการเปลี่ยนโหมดของการส่งข้อมูลกลับเป็นโหมดของการรับข้อมูลแทน ซึ่งเมื่อมีการส่งข้อมูล Byte ถัดไปมาก็จะต้องเสียเวลาในการสลับโหมดจากฝ่ายรอรับข้อมูลให้เป็นฝ่ายส่งข้อมูลอีก ซึ่งจะทำให้ประสิทธิภาพในการจัดส่งข้อมูลลดลงเนื่องจากต้องเสียเวลาในการสลับโหมดการทำงานของวงจรภาค RF อยู่ตลอดเวลา โดยที่เมื่อทำการจัดส่งข้อมูลครบ 64 Byte แล้ว ให้ทำการหน่วงเวลาไว้ชั่วขณะหนึ่ง ประมาณ 1 – 2 มิลลิวินาที แล้วจึงส่งข้อมูลชุดถัดไปอีกอย่างนี้เรื่อยๆ จะทำให้การรับส่งข้อมูลมีประสิทธิภาพสูงสุด

การใช้งานเครื่อง รับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 แบบ Auto Direction นั้น ควรหน่วงเวลาในการสลับโหมดจากฝ่ายของการรอรับข้อมูลเป็นฝ่ายส่งข้อมูล อย่างน้อยที่สุด 3 – 5 มิลลิวินาที ซึ่งถ้าส่งข้อมูลย้อนกลับด้วยเวลาที่เร็วกว่านี้อาจทำให้ฝ่ายตรงข้ามไม่สามารถรับข้อมูล Byte แรกได้ทัน

2.4.4 การเชื่อมต่อสัญญาณ RS232



รูปที่ 2.5 การต่อสายส่งสัญญาณ RS232 เพื่อใช้งานกับบอร์ดไมโครคอนโทรลเลอร์ในโหมดการทำงานปกติ

2.4.5 ตัวอย่างการใช้งานโปรแกรม

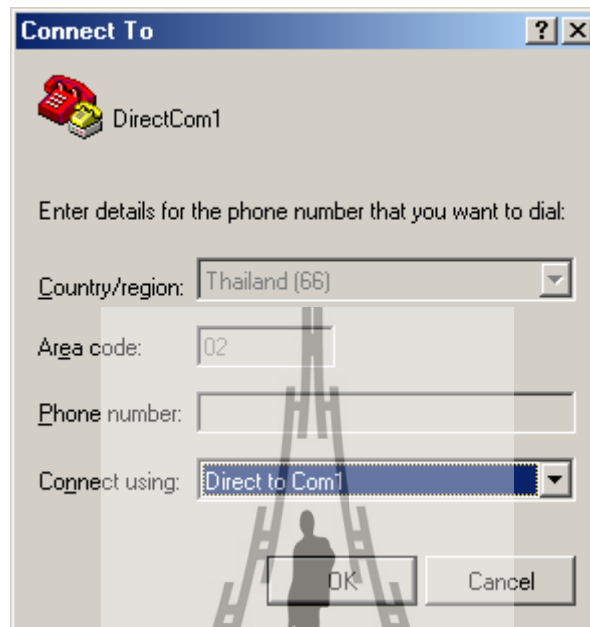
สำหรับตัวอย่างการใช้งานนั้น จะขอแสดงให้เห็นโดยใช้คอมพิวเตอร์ PC เป็นอุปกรณ์การทดลอง โดยในที่นี้จะขอเลือกใช้โปรแกรมสำเร็จรูปสำหรับการสื่อสารของ Windows ซึ่งก็คือ Hyper terminal โดยใน 2 ตัวอย่างแรกนั้นจะใช้งานกับเครื่อง รับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 ในโหมด Auto direction ซึ่งมีวิธีการใช้งานดังต่อไปนี้

1. เรียกใช้โปรแกรม Hyper terminal ของ Windows โดยเรียกจาก Start → Programs → Accessories → Communications → Hyper terminal ซึ่งจะได้ผลดังรูป



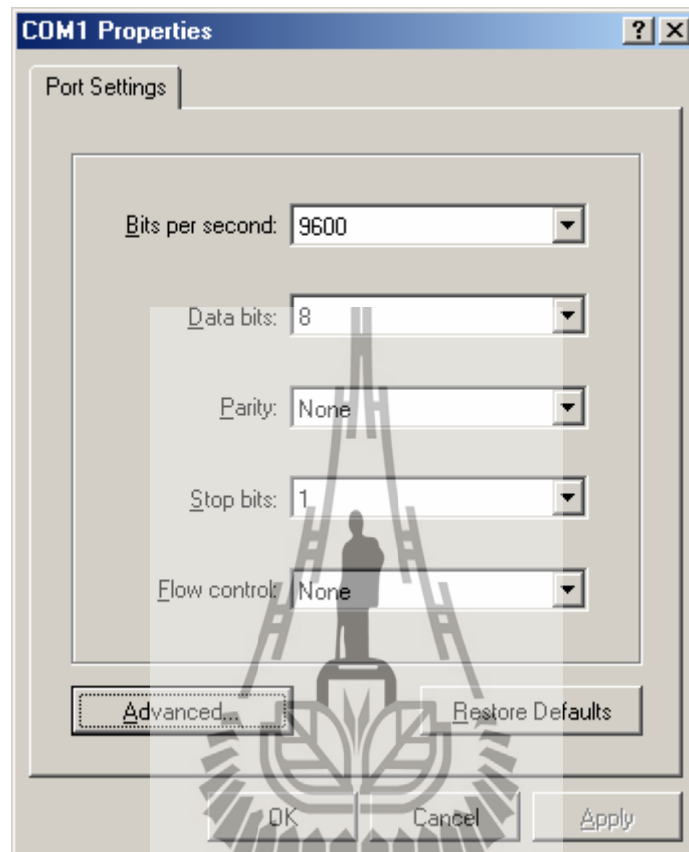
รูปที่ 2.6 การใช้งานโปรแกรม Hyper terminal

2. ให้เลือกกำหนดชื่อสำหรับใช้ในการเชื่อมต่อ ซึ่งสามารถกำหนดได้เองตามต้องการ โดยในตัวอย่างจะกำหนดเป็นDirectCom1จากนั้นให้เลือก OK เพื่อเข้าไปยังขั้นตอนถัดไป



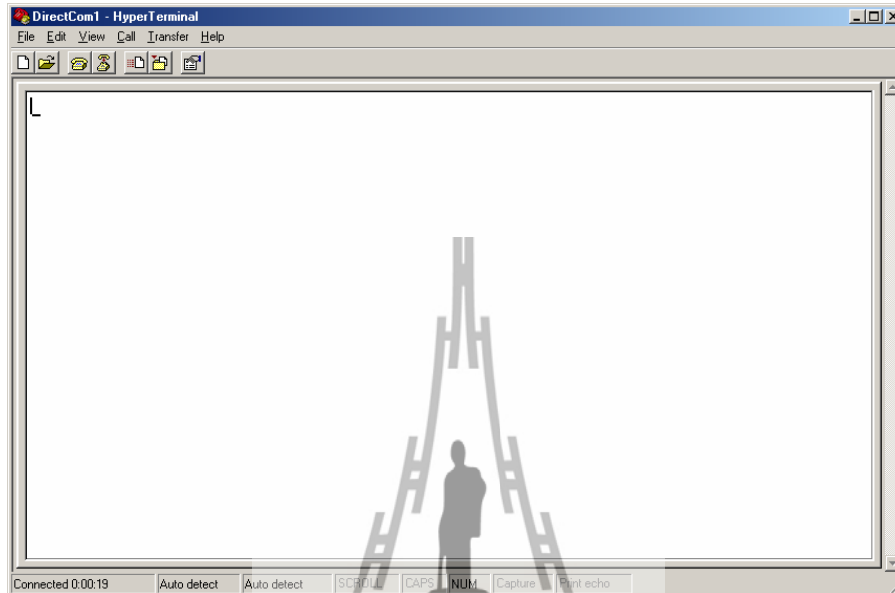
รูปที่ 2.7 การใช้งานโปรแกรม Hyper terminal

3. ให้เลือกกำหนดการเชื่อมต่อเป็น Direct to Com1 ซึ่งถ้าเครื่องคอมพิวเตอร์ที่ใช้เป็น Comport อื่นที่ไม่ใช่Com1 ก็ให้เลือกให้ตรงกับความเป็นจริง จากนั้นให้เลือก OK เพื่อเข้าไปยังขั้นตอนถัดไป



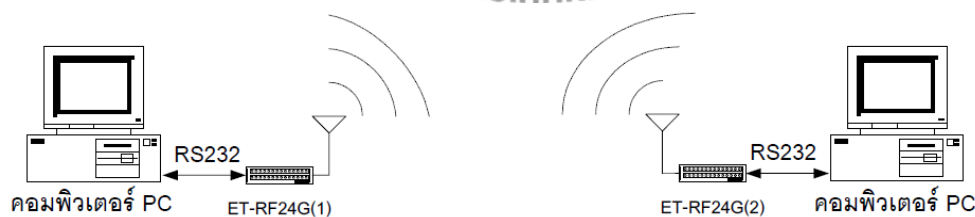
รูปที่ 2.8 การใช้งาน โปรแกรม Hyper terminal

4. ในขั้นตอนนี้ จะใช้สำหรับกำหนดคุณสมบัติของพอร์ตอนุกรม RS232 โดยให้เลือก Bit per second = 9600 ,Data Bit = 8 ,Parity = None ,Stop Bit=1 ส่วน Flow Control ให้เลือกเป็น None จากนั้นเลือก OK ซึ่งจะเข้าสู่หน้าต่างโปรแกรมหลักของ Hyper terminal ดังรูป



รูปที่ 2.9 การใช้งานโปรแกรม Hyper terminal

ตัวอย่างที่ 1 การรับส่งข้อมูล 2 ทิศทาง (Half Duplex) แบบ จุดต่อจุด (Point-to-Point)



สำหรับตัวอย่างนี้จะเป็นการรับส่งข้อมูลระหว่างอุปกรณ์ที่มีการสื่อสารอนุกรมแบบ RS232 จำนวน 2 ชุด โดยต้องใช้รูปแบบการสื่อสารแบบ Half Duplex หรือ ผลัดกันรับ ผลัดกันส่ง กล่าวคือ ด้านรับจะต้องทำการรอรับข้อมูลจากด้านส่งจนครบทั้งหมด แล้วจึงจะส่งข้อมูลตอบกลับไปได้ ซึ่งจะไม่สามารถส่งข้อมูลสวนทางกลับไปในขณะที่กำลังรับข้อมูลอยู่ได้ โดยการสื่อสารแบบนี้ฝ่ายรับข้อมูลจะต้องรอให้รับข้อมูลได้ครบทั้งหมดเสียก่อน จากนั้นจึงจะส่งข้อมูล

ตอบกลับไปได้ โดยให้กำหนดค่า Configuration ของตัวเครื่องรับ - ส่งสัญญาณไร้สาย ET-RF24G V2.0 เป็นดังนี้

ค่า Configuration	ET-RF24G V2.0 ตัวที่1	ET-RF24G V2.0 ตัวที่2
User RS232 Baudrate	9600 Bps	9600 Bps
RF Data Rate	250 Kbps	250 Kbps
RF Operation Mode	Auto Direction	Auto Direction
RF Power Gain	+0dBm	+0dBm
RXD ID Code	01	02
TXD ID Code	02	01
RF Frequency Channel	0	0

2.4.6 ข้อสังเกตในการกำหนดโครงสร้าง

1. ค่า RF Frequency channel ต้องกำหนดให้ตรงกันทั้ง 2 ตัว
2. ค่า RF Data rate ต้องกำหนดให้ตรงกันทั้ง 2 ตัว
3. ค่า RXD ID code ของตัวที่1 ต้องตรงกับ TXD ID code ของตัวที่2
4. ค่า TXD ID code ของตัวที่1 ต้องตรงกับ RXD ID code ของตัวที่2

สำหรับการทดสอบการทำงานด้วย Hyper terminal นั้นให้ทดลองกดคีย์ใดๆ ในขณะที่ Run โปรแกรม Hyper terminal อยู่ โดยจะสังเกตเห็นตัวอักษรจากแป้นพิมพ์ของฝ่ายที่เป็นฝ่ายส่งข้อมูล จะถูกส่งออกไปแสดงผลที่หน้าจอ โปรแกรม Hyper terminal ของอีกฝ่ายหนึ่งในทันที

2.5 อุปกรณ์ตรวจสอบภาพพิมพ์ลายนิ้วมือ



รูปที่ 2.10 Fingerprint sweep 001

2.5.1 อุปกรณ์เครื่องสแกนลายนิ้วมือรุ่น FPS-001

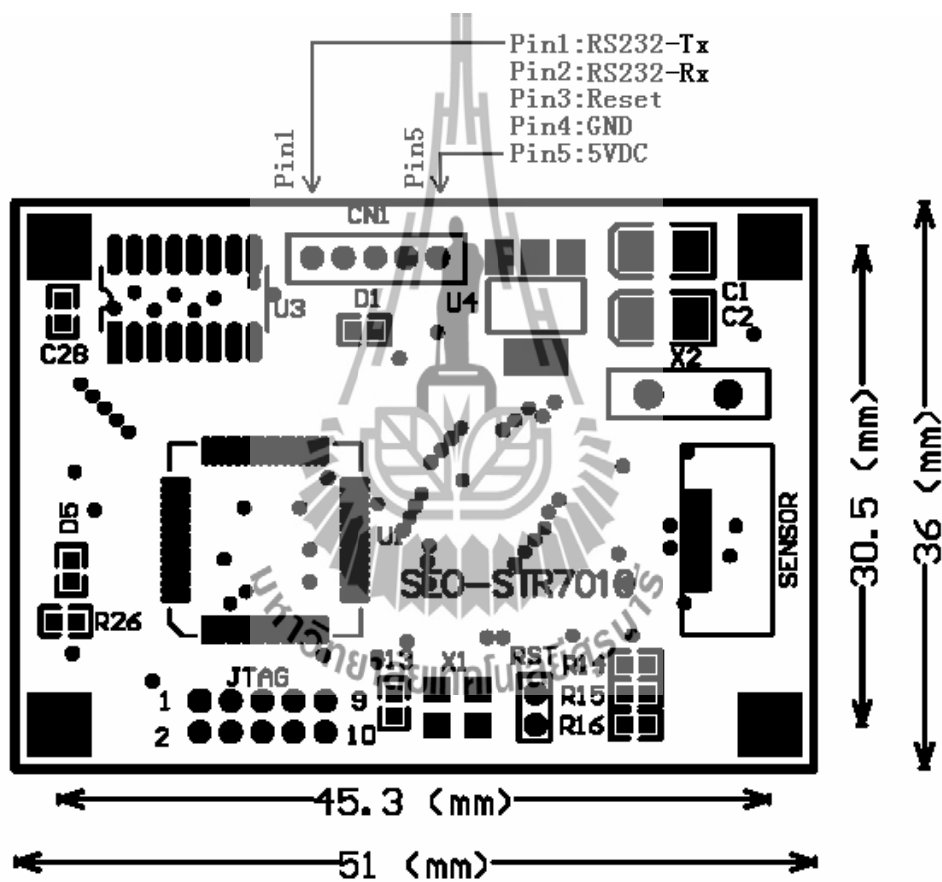
เครื่องสแกนลายนิ้วมือรุ่น FPS-001 เป็นโมดูลที่มี Fingerprint processor (MCU STR7010 ที่มี Firmware algorithm อยู่ภายใน) ซึ่งเป็นตัวประมวลผลสัญญาณข้อมูลที่ได้จาก Optical sweep sensor ซึ่งเป็นอุปกรณ์ สำหรับการอ่านลายนิ้วมือด้วยวิธี รูด (Sweep)

2.5.1.1 คุณสมบัติทั่วไปของเครื่องสแกนลายนิ้วมือรุ่น FPS-001

1. ออกแบบมาให้มีลักษณะเล็กกะทัดรัด
2. Firmware ที่พัฒนานั้นเป็น algorithm ที่ให้ความแม่นยำสูงและมีความรวดเร็วในการสแกนลายนิ้วมือ
3. การต่อใช้งานง่าย ผ่าน UART
4. การจับคู่แบบ 1:N & 1:1 matching
5. สามารถปรับ identification threshold ได้
6. ใช้พลังงานต่ำ
7. ไม่มีส่วนของความถี่สูงในวงจร

2.5.1.2 คุณสมบัติทางเทคนิค

1. ขนาดภาพ 232 x 324 พิกเซล
2. ความละเอียดภาพ 600 DPI
3. ขนาดของบอร์ด 51 x 36 x 10 มิลลิเมตร
4. ขนาดของเซนเซอร์ 18.2 x 4.8 x 2.75 มิลลิเมตร
5. ไฟเลี้ยง 5 โวลต์
6. กระแส 15 มิลลิแอมป์
7. สามารถใช้งานได้กว่า 1 ล้านครั้ง



รูปที่ 2.11 โครงสร้างบอร์ดเครื่องสแกนลายนิ้วมือรุ่น FPS-001

2.5.1.3 ขาอุปกรณ์ที่ใช้ในติดต่อใช้งาน

Port	Pin	คำนิยาม	หมายเหตุ
CN1	Pin1	RS232-Tx	Output ของอุปกรณ์ที่ใช้ในการส่งข้อมูล
	Pin2	RS232-Rx	Input ของอุปกรณ์ที่ใช้ในการรับข้อมูล
	Pin3	Reset	เมื่อต้องการ restart system ให้ส่ง input low-level pulse ไปที่ pin นี้
	Pin4	GND	กราวด์
	Pin5	5VDC	ไฟเลี้ยง 5 โวลต์

2.6 รูปแบบการสื่อสาร (Communication protocol)

2.6.1 หลักการสื่อสาร (Communication method)

โมดูลนี้ถูกตั้งให้เป็น Slave Device ดังนั้นอุปกรณ์ที่เป็น Master device จะต้องส่งคำสั่งที่เกี่ยวข้องต่างๆ เพื่อการควบคุมตัวโมดูลโดยที่ “CMD” เป็น Header ของเฟรมข้อมูลที่ Master device ใช้ในการส่งไปยัง FBS-001 เพื่อการควบคุมและ “RCM” เป็นสัญญาณตอบกลับที่จะได้รับ เมื่อฝั่ง Slave device ได้รับข้อมูลครบถ้วน

ก. รูปแบบของคำสั่ง (Command packet)

CMD	LEN	DAT	CHK
2 byte	2 byte	LEN byte	1 Byte

Offset	Item	Size (byte)	Description
0	CMD	2	Command Code
1	LEN	2	Length of Data
2	DAT	LEN	Command Process Result Data. LEN=0 no Data LEN=XX=>DATA
3	CHK	1	Chek Sum is the low byte of value listed below $offset[0]+offset[1]+offset[2]$

ข. การตอบกลับของโมดูล

RCM	LEN	DAT	CHK
2 byte	2 byte	LEN byte	1 Byte

Offset	Item	Size (byte)	Description
0	CMD	2	Response Code
1	LEN	2	Data Length
2	DAT	LEN	Command Process Result Data. LEN=0 no Data LEN=XX=>DATA
3	CHK	1	Chek Sum is the low byte of value listed below offset[0]+offset[1]+offset[2]

2.6.2 คำอธิบายคำสั่ง (Command description)

การคำสั่งสื่อสารระหว่าง Host และ โมดูล FPS-001 มีลักษณะตามด้านล่าง

Host (MCU or Computer) =====> Command =====> FPS-001 Module

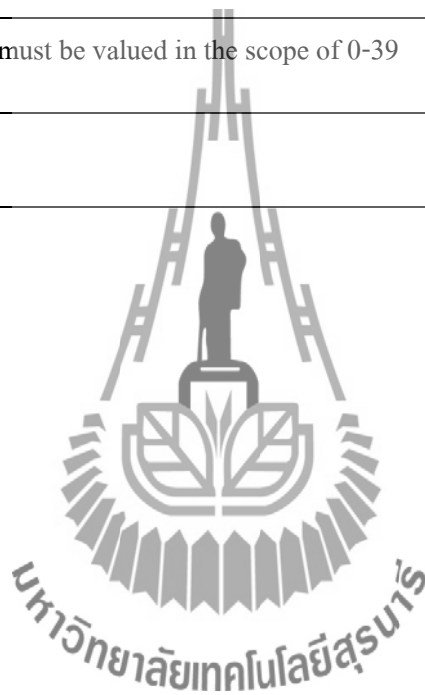
Host (MCU or Computer) <===== Response <===== FPS-001 Module

ก. การทำงานของโปรแกรม

[ฟังก์ชัน]: การจับคู่ระหว่างเครื่องต่อเครื่อง

[คำสั่ง]:

CMD	0x0101
LEN	0x0001
DAT	User ID, must be valued in the scope of 0-39
CHK	



[ผลของคำสั่ง]:

RCM	0101h			
LEN	0002h			
DAT	DAT1	User ID		
	DAT2	Result		Description
		0x00	Reject	Verification failure
		0x01	Accept	Successful verification
		0x02	Time Out	Image capturing time out
		0x03	Bad Quality Image	Bad quality image, no enough minutiae, cannot verify
		0x05	Too Small Lines	Image is too small, get no enough minutiae, cannot verify
		0x06	Empty Template	There is no fingerprint data with user ID
		0xFF	Invalid Template Number	ID is out of the range of enrollment
CHK				

บทที่ 3

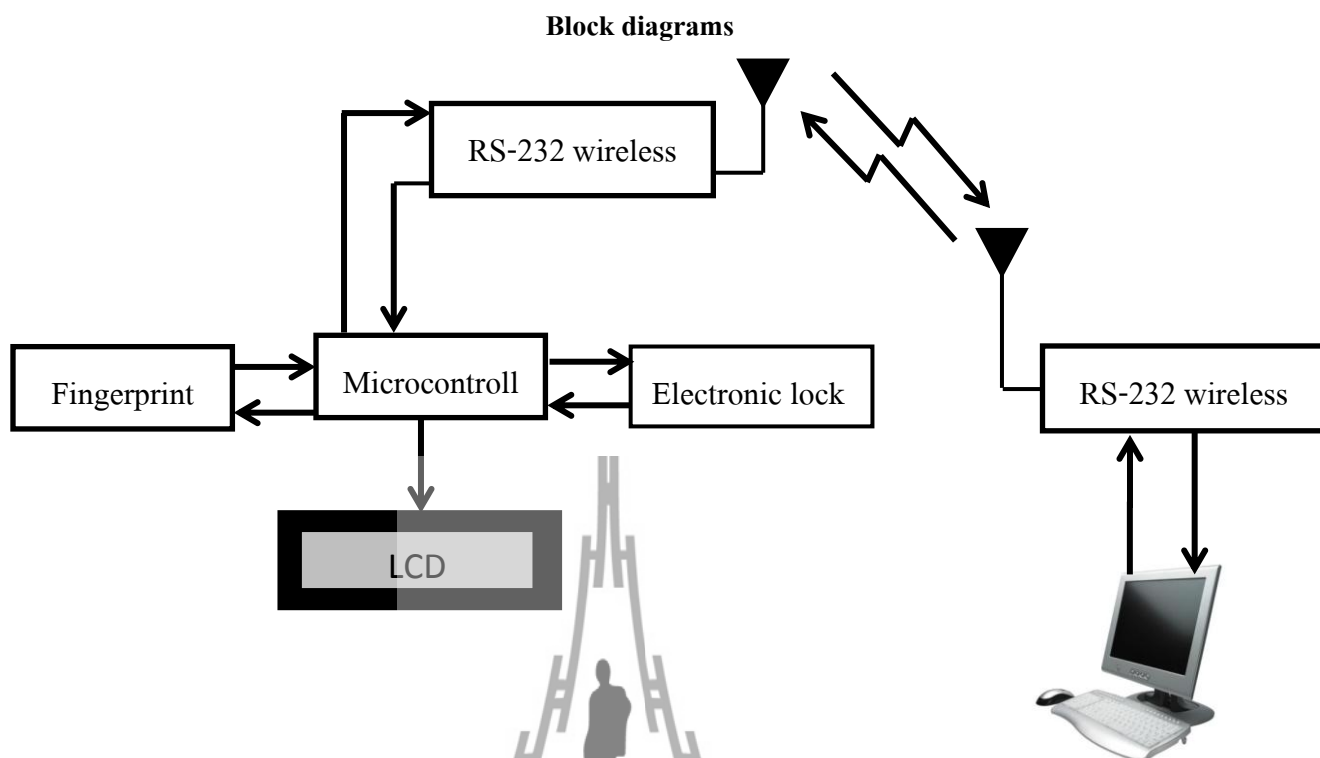
การออกแบบโครงงาน

3.1 บทนำ

ในบทนี้จะกล่าวถึงการออกแบบทางด้าน Hardware ของเครื่องสแกนลายนิ้วมือ ที่ทำการควบคุมการทำงานด้วยบอร์ด AVR และ COMPUTER ผ่านพอร์ต RS-232 แผนภาพของตัวสแกนลายนิ้วมือ พร้อมคำอธิบายแผนภาพการทำงานของตัวสแกนลายนิ้วมือแบบ และด้าน Software การทำงานของโปรแกรมที่แสดงอยู่ในรูปของแผนภาพ Flow chart พร้อมคำอธิบายการทำงานของโปรแกรมทั้งทางด้านการบันทึกข้อมูลและการสแกนลายนิ้วมือเพื่อเปิดประตู ขั้นตอนการใช้งานตัวสแกนลายนิ้วมือเพื่อเปิดประตูพร้อมสิ่งที่แสดงบนจอแสดงผล ขั้นตอนการบันทึกลายนิ้วมือเพื่อเก็บข้อมูลลงในตัวสแกนลายนิ้วมือพร้อมสิ่งที่แสดงบนจอแสดงผล เพื่อเป็นแนวทางในการใช้และการดำเนินงานของตัวสแกนลายนิ้วมือ และเพื่อให้ทราบขอบเขตการทำงานของตัวสแกนลายนิ้วมือชุดนี้



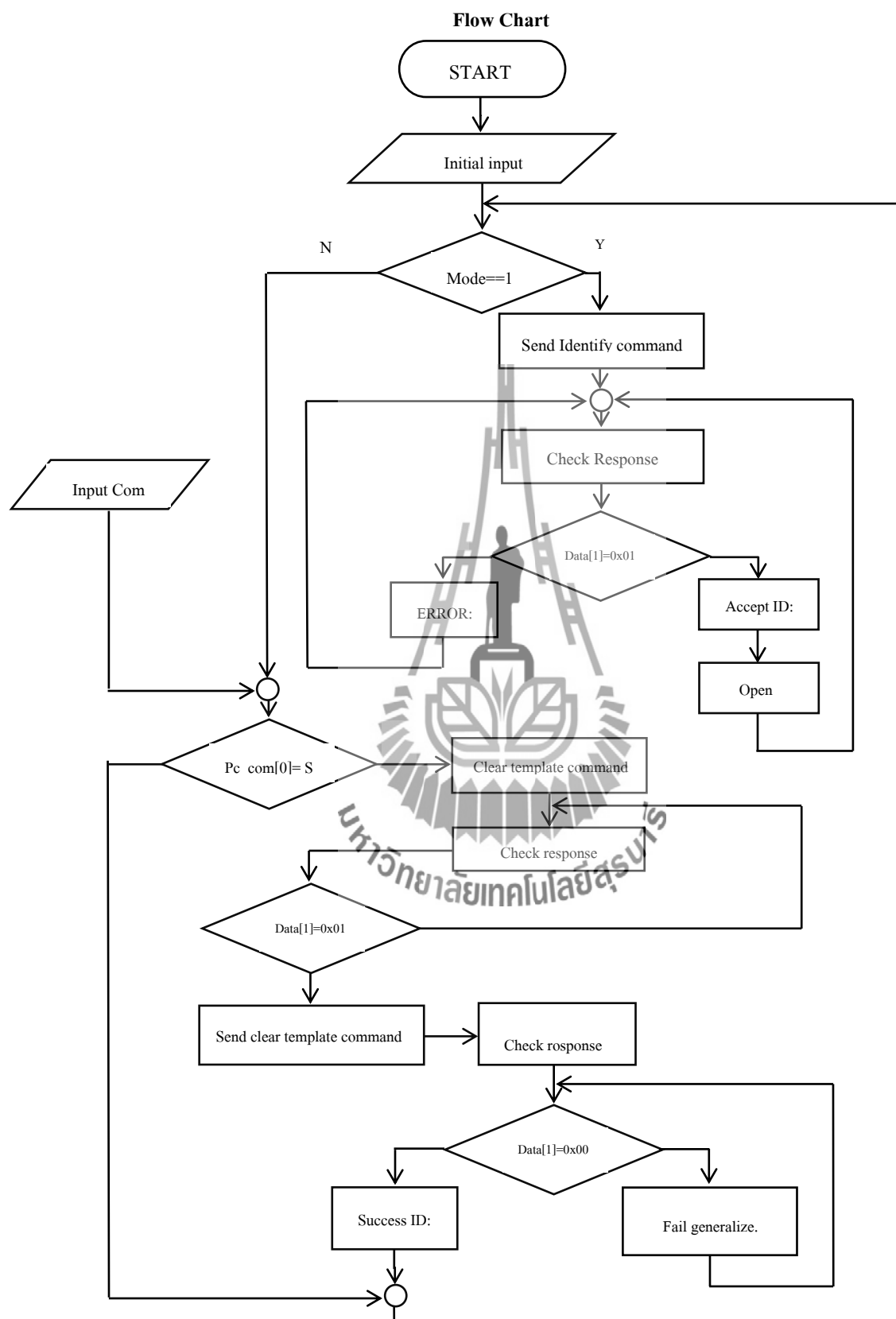
3.2 ออกแบบ Hardware



รูปที่ 3.1 แผนภาพการทำงานของตัวสแกนลายนิ้วมือ

รูปที่ 3.1 แสดงแผนภาพการทำงานของตัวสแกนลายนิ้วมือ โดยภาพนี้ประกอบไปด้วย 1. บอร์ด AVR 2. LCD 3. Fingerprint 4. Electronic lock 5. ชุดรับส่งข้อมูลผ่านพอร์ต RS-232 แบบไร้สาย 6. Computer ระบบนี้เป็นการควบคุมแบบอัตโนมัติ โดยใช้บอร์ด Microcontroller AVR เป็นตัวควบคุมหลักของระบบ โดยการส่งค่าไปหาตัวสแกนลายนิ้วมือแล้วเช็คค่าที่เครื่องสแกนลายนิ้วมือว่าได้ค่าออกมาเท่าไรแล้วจึงส่งค่าตอบสนองกลับมาที่บอร์ด microcontroller AVR แล้วแสดงข้อความออกที่จอแสดงผล (LCD) ถ้าเช็คค่าถูกบอร์ดจะดำเนินการให้ Relay ทำงานเพื่อ เปิดประตู (Electronic lock) แล้วส่งค่าไปบันทึกเวลาที่ Computer ผ่านชุดรับส่งข้อมูลผ่านพอร์ต RS-232 แบบไร้สาย

3.3 ออกแบบ Software



รูปที่ 3.2 แผนภาพ Flow chart แสดงการทำงานของระบบสแกนลายนิ้วมือ

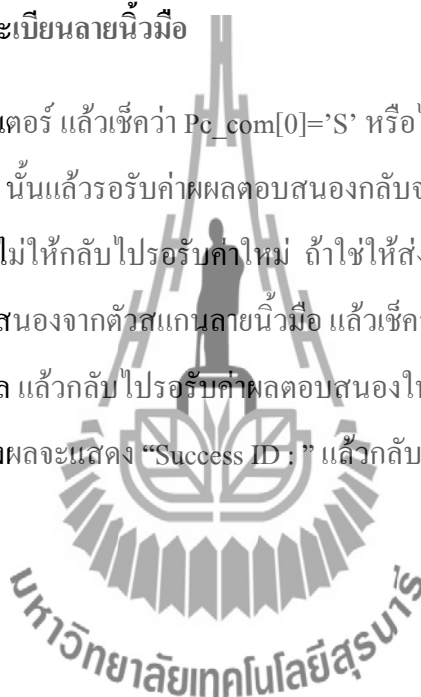
อธิบาย แผนภาพ Flow chart

ส่วนที่ 1 ตรวจสอบลายนิ้วมือ

เริ่มโปรแกรมเช็คความพร้อมใช้งาน โดยเช็คค่า Mode = 1 หรือไม่ ถ้าไม่ทำการเช็คค่าอย่างอื่นต่อในส่วนที่ 2 ถ้าใช่แสดงว่าพร้อมใช้งาน บอร์ด AVR จะส่ง Command code ไปที่ตัวสแกนลายนิ้วมือ แล้วรอรับผลตอบสนองกลับจากตัวสแกนลายนิ้วมือ เพื่อเช็คค่า Data[1] = 0x01 หรือไม่ ถ้าไม่แสดง “Error” บนจอแสดงผลแล้วกลับไปรอรับผลตอบสนองกลับจากตัวสแกนลายนิ้วมือ ถ้าใช่แสดง “Accept ID :” ที่จอแสดงผล จากนั้นทำการเปิดประตู แล้วกลับไปรอเช็ค Mode อีกครั้ง

ส่วนที่ 2 การลงทะเบียนลายนิ้วมือ

รับค่าจากคอมพิวเตอร์ แล้วเช็คค่า Pc_com[0]='S' หรือไม่ ถ้าไม่จะกลับไปรอเช็คค่าเริ่มต้น ถ้าใช่ให้ล้างข้อมูลใน ID นั้นแล้วรอรับค่าผลตอบสนองกลับจากตัวสแกนลายนิ้วมือ แล้วเช็คค่า Data[1]=0x01 หรือไม่ ถ้าไม่ให้กลับไปรอรับค่าใหม่ ถ้าใช่ให้ส่ง Command code ไปเพื่อล้างข้อมูลอีกครั้ง แล้วรอรับผลตอบสนองจากตัวสแกนลายนิ้วมือ แล้วเช็คค่า Data[1]=0x00 หรือไม่ ถ้าไม่แสดง “Fail” ที่จอแสดงผล แล้วกลับไปรอรับค่าผลตอบสนองใหม่อีกครั้ง ถ้าใช่แสดงว่าสแกนลายนิ้วมือสำเร็จที่จอแสดงผลจะแสดง “Success ID :” แล้วกลับไปรอเช็คค่าเริ่มต้นอีกครั้ง

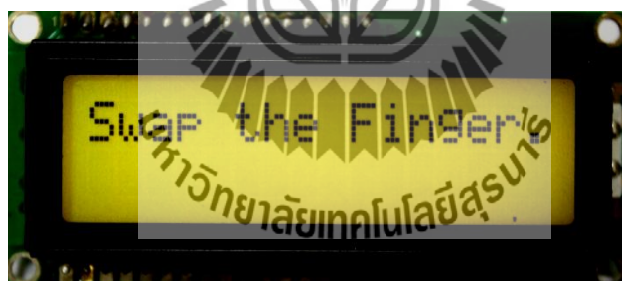


3.4 การทำงาน

3.4.1 ขั้นตอนการใช้งานการเปิดประตู

- 1.1. สังเกตสถานะพร้อมใช้งาน หน้าจอ LCD จะแสดง “Swap the Finger” ดังแสดงในรูปที่ 3.3
- 1.2. สแกนนิ้วมือลงบนแถบสแกนลายนิ้วมือ
- 1.3. จอแสดงผลแสดง “Accept ID:” แสดงว่าลายนิ้วมือถูกต้อง ดังแสดงในรูปที่ 3.5 และคอมพิวเตอร์จะทำการบันทึกเวลา ว่า ID ไหน ชื่ออะไร เปิดประตูเวลาเท่าไร ดังแสดงให้ดูในรูปที่ 3.6
- 1.4. จอแสดงผลแสดง “Error: Failure” แสดงว่าสแกนลายนิ้วมือผิดพลาด ดังแสดงในรูปที่ 3.4
- 1.5. จอแสดงผลแสดง “Error: Time out” แสดงว่าสแกนลายนิ้วมือผิดพลาดแล้วหลุดออกจาก Loop ดังแสดงในรูปที่ 3.7
- 1.6. จอแสดงผลแสดง “Error: Bad Quality” แสดงว่าสแกนลายนิ้วมือผิดพลาด (ลายนิ้วมือเสียหาย) ดังแสดงในรูปที่ 3.8
- 1.7. จอแสดงผลแสดง “Error: Too Small” แสดงว่าสแกนลายนิ้วมือผิดพลาด (ลายนิ้วมือเล็กเกินไป) ดังแสดงในรูปที่ 3.9

ถ้า mode = 1 แสดงสถานะพร้อมใช้งาน “Swap the Finger”



รูปที่ 3.3 สถานะพร้อมใช้งาน

SendIdentify Command ; CMD + LEN

```
ตัวอย่างเช่น  putchar(0x01);
               putchar(0x02);
               putchar(0x00);
               putchar(0x00);
               putchar(0x03);
```

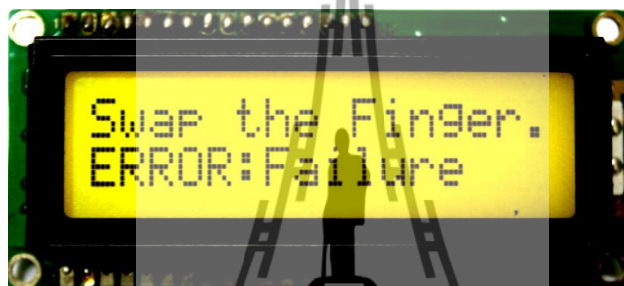
รับค่า Response

ตัวอย่างเช่น

```
rcm[0] = getchar();
rcm[1] = getchar();
len[0] = getchar();
len[1] = getchar();
data [0] = getchar();
data [1] = getchar();
chk = getchar();
```

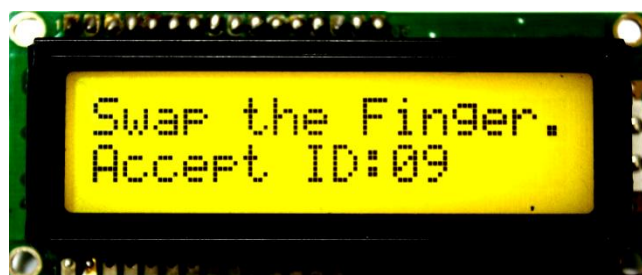
Check Response

data [1] == 0x00 ; LCD Show ***“ERROR : Failure”***
 “สแกนผิดพลาด”



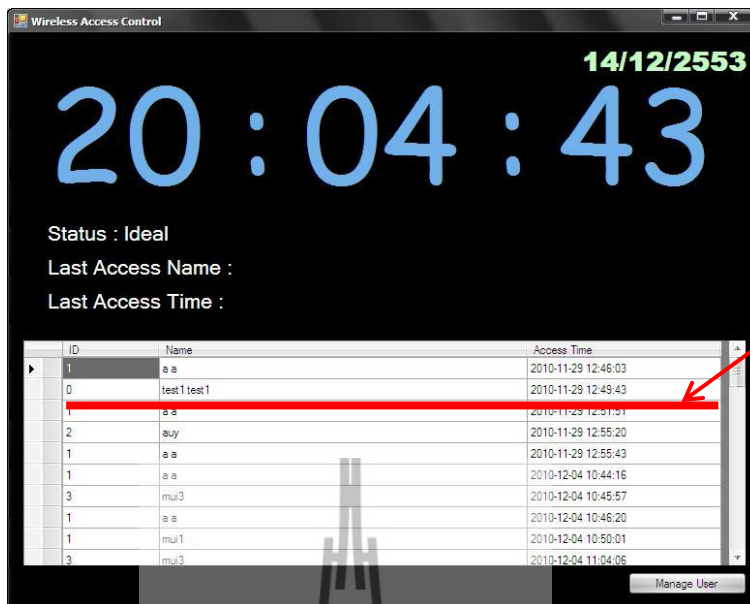
รูปที่ 3.4 สถานะสแกนผิดพลาด

data [1] == 0x01 ; LCD Show ***“Accept ID :***
 “สแกนผ่าน (แสดง ID ของผู้ใช้นี้ ID: 09)”



รูปที่ 3.5 สถานะสแกนลายนิ้วมือผ่าน

จากนั้น Software ที่อยู่ใน Com จะทำการบันทึกเวลาที่ทำการตรวจสอบแล้วผ่านทุกครั้ง (เปิดประตู)



เปิดประตู
บันทึกเวลา 12:49:43
วันที่ 29/11/2010
ชื่อ Test1Test1

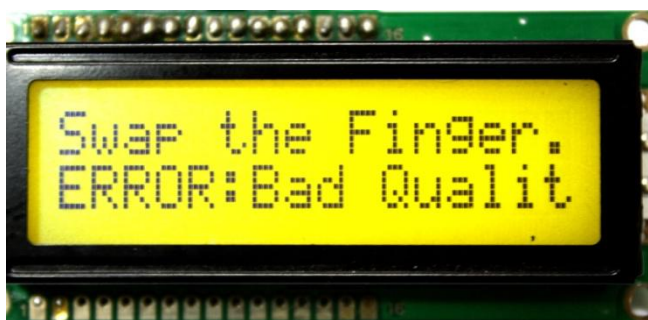
รูปที่ 3.6 การบันทึกเวลาเข้า-ออก ของผู้ใช้

data [1] == 0x02 ; LCD Show “*ERROR :Time out*”
“สแกนผิดพลาด”



รูปที่ 3.7 สถานะสแกนผิดพลาดแล้วหลุดออกจาก Loop

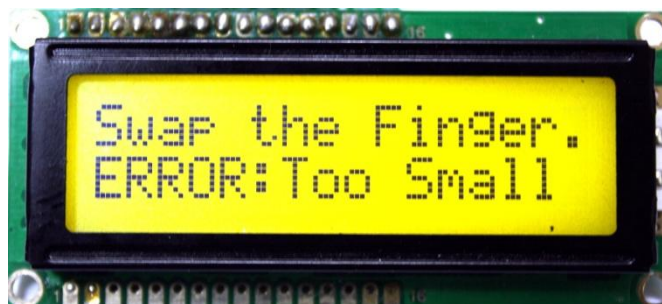
data [1] == 0x03 ; LCD Show “*ERROR : Bad Quality*”
“สแกนผิดพลาด (ลายมือเสียหาย)”



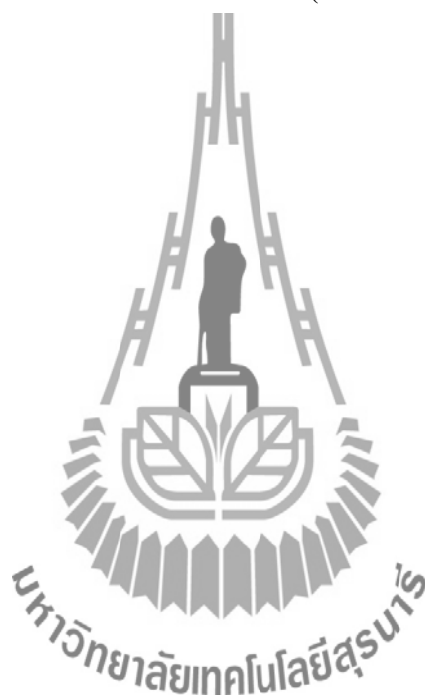
รูปที่ 3.8 สถานะสแกนลายนิ้วมือผิดพลาด (ลายนิ้วมือเสียหาย)

`data[1] == 0x00 ; LCD Show "ERROR : Too small"`

“สแกนผิดพลาด (ลายมือเล็กเกินไป)”

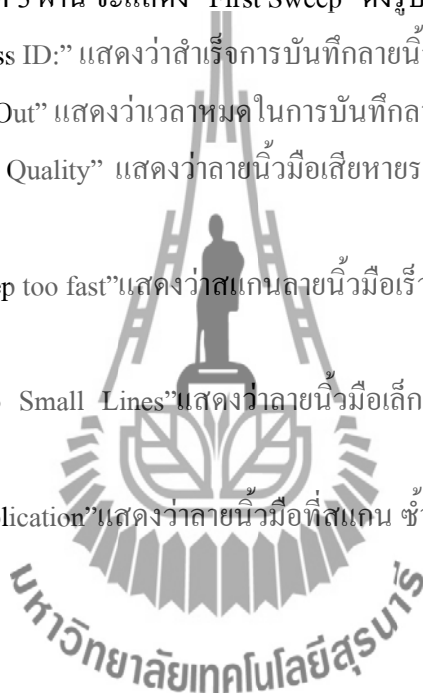


รูปที่ 3.9 สถานะสแกนลายนิ้วมือผิดพลาด (ลายนิ้วมือเล็กเกินไป)



3.4.2 ขั้นตอนการใช้งานการลงทะเบียนลายนิ้วมือ

- 2.1. กด Next or Last เพื่อเลื่อน ID
- 2.2. ตั้งชื่อของ ID นั้นๆ
- 2.3. กด Scan ดังรูปที่ 3.10 ที่จอ LCD จะแสดง “Clearing id:” (id ที่เลือก) ดังรูปที่ 3.11 และจะแสดง “Stesting Enroll” ดังรูปที่ 3.12
- 2.4. จอ LCD จะแสดง First Sweep เพื่อบอกว่า รอรับลายนิ้วมือที่จะบันทึก (จะทำการบันทึก 3 ครั้ง) ให้ทำการสแกนลายนิ้วมือที่ 1
- 2.5. บันทึกลายนิ้วมือครั้งที่ 1 ผ่าน จะแสดง “Two Sweep Left” ดังรูปที่ 3.19
- 2.6. บันทึกลายนิ้วมือครั้งที่ 2 ผ่าน จะแสดง “One Sweep Left” ดังรูปที่ 3.20
- 2.7. บันทึกลายนิ้วมือครั้งที่ 3 ผ่าน จะแสดง “First Sweep” ดังรูปที่ 3.21
- 2.8. หน้าจอแสดง “Success ID:” แสดงว่าสำเร็จการบันทึกลายนิ้วมือ ดังแสดงในรูปที่ 3.14
- 2.9. หน้าจอแสดง “Time Out” แสดงว่าเวลาหมดในการบันทึกลายนิ้วมือ ดังแสดงในรูปที่ 3.15
- 2.10. หน้าจอแสดง “Bad Quality” แสดงว่าลายนิ้วมือเสียหายระหว่างการลงทะเบียน ดังแสดงในรูปที่ 3.16
- 2.11. หน้าจอแสดง “Sweep too fast” แสดงว่าสแกนลายนิ้วมือเร็วเกินไประหว่างการลงทะเบียน ดังแสดงในรูปที่ 3.17
- 2.12. หน้าจอแสดง “Too Small Lines” แสดงว่าลายนิ้วมือเล็กเกินไประหว่างการลงทะเบียน ดังแสดงในรูปที่ 3.17
- 2.13. หน้าจอแสดง “Duplication” แสดงว่าลายนิ้วมือที่สแกน ซ้ำกับลายนิ้วมือที่มีอยู่แล้ว ดังแสดงในรูปที่ 3.17



จาก Software ที่เขียนขึ้น



รูปที่ 3.10 การสั่งบันทึกชื่อและ ID

กด Next or Last เพื่อเลื่อน ID

ตั้งชื่อของ ID นั้นๆ

กด Scan (เมื่อกดปุ่มนี้ Com จะส่งค่า 'S' ไปที่Finger print ผ่านบอร์ด AVR)

ถ้า pc_com = 'S' แสดงคำสั่ง "Clearing id:"

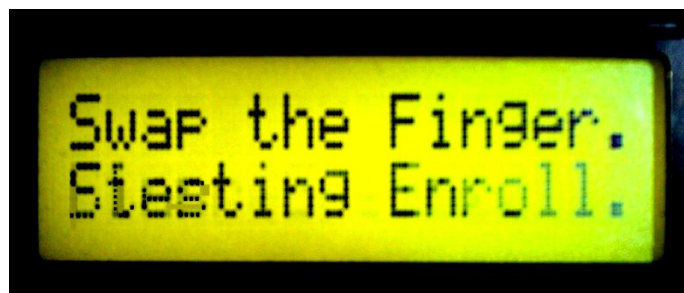
ถ้าได้รับคำสั่ง ID : 1-40 ; จะเคลียร์ค่าใน ID นั้นแล้วรับค่าใหม่



รูปที่ 3.11 สถานะการส่งคำสั่งลบค่าใน ID:6

LCD Show "Starting Enroll"

ส่งคำสั่งเพื่อลงทะเบียนเข้าใช้งาน แสดงผลดังนี้



รูปที่ 3.12 สถานะการบอกว่าพร้อมลงทะเบียน

Send clear template Command

```
ตัวอย่างเช่น    putchar(0x01);
                putchar(0x03);
                putchar(0x00);
                putchar(0x01);
                putchar(save_id);
                putchar(0x05+save_id);
```

LCD Show **“First Sweep”**เพื่อรอรับค่า ที่จะส่งมาจาก ตัวแสแกนลายนิ้วมือ



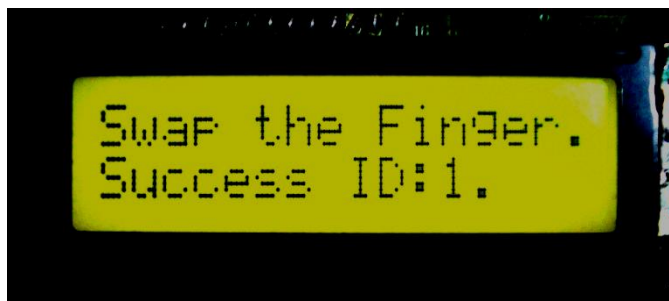
รูปที่ 3.13 สถานะการรอรับลายนิ้วมือจาก Finger print

รับค่า Response

```
ตัวอย่างเช่น    loop = 0;
                while (loop==0){
                rcm[0] = getchar();
                rcm[1] = getchar();
                len[0] = getchar();
                len[1] = getchar();
                data[0] = getchar();
                data[1] = getchar();
                chk = getchar();
```

Check Response

data [1] == 0x01; LCD Show “*Success ID:.....*”
 “สแกนเก็บค่าลายนิ้วมือสำเร็จที่ ID: 1 (ประตูเปิด)”



รูปที่ 3.14 สถานะการเก็บลายนิ้วมือสำเร็จ

data [1] == 0x02; LCD Show “*Time Out.*”
 “หมดเวลาในกาสแกนลายนิ้วมือ (หลุดออกไปป้อนค่า ‘S’ ใหม่เพื่อรับค่า ID: ตัวต่อไป)”



รูปที่ 3.15 สถานะหมดเวลาในกระบวนการ

data [1] == 0x03; LCD Show “*Bad Quality.*”
 “ลายนิ้วมือมีความผิดพลาด”



รูปที่ 3.16 สถานะบอกว่าลายนิ้วมือมีความผิดพลาด

data [1] == 0x04; LCD Show “*Sweep too fast*”

“สแกนเร็วเกินไป”



รูปที่ 3.17 สถานะบอกว่าสแกนเร็วเกินไป

data [1] == 0x05; LCD Show “*Too Small Lines*”

“ลายนิ้วมือเล็กเกินไป”



รูปที่

3.18 สถานะบอกลายนิ้วมือเล็กเกินไป

data [1] == 0x06; LCD Show “*Two Sweep Left*”

“ลายนิ้วมือแรกผ่านแล้วสแกนเพื่อเก็บลายนิ้วมืออื่นที่ 2 เพื่อนำมาเทียบกับลายมือแรก”



รูปที่ 3.19 สถานะบอกว่าใส่ลายนิ้วมือที่ 2

data [1] == 0x07; LCD Show *“One Sweep Left”*

“สแกนลายนิ้วมือสุดท้ายเพื่อเปรียบเทียบกับลายนิ้วมือทั้งสองที่ผ่านมา”



รูปที่ 3.20 สถานะบอกว่าใส่ลายนิ้วมือที่ 3

data [1] == 0xA1; LCD Show *“First Sweep”*

“สิ้นสุดการเก็บค่าลายนิ้วมือสุดท้าย”



รูปที่ 3.21 สถานะบอกว่าลายนิ้วมือที่ 3 ผ่านแล้ว

data [1] == 0xA2; LCD Show *“Duplication”*

“ลายนิ้วมือมีบันทึกอยู่แล้ว (ซ้ำกับลายนิ้วมือเดิม)”



รูปที่ 3.22 สถานะบอกว่าลายนิ้วมือซ้ำกับลายนิ้วมือที่มีอยู่แล้ว

บทที่ 4

การทดสอบเครื่องสแกนลายนิ้วมือเพื่อเปิด-ปิดประตู

4.1 บทนำ

ในบทนี้จะกล่าวถึงการทดสอบการทำงานของเครื่องสแกนลายนิ้วมือเพื่อเปิด - ปิดประตู โดยประกอบด้วยปฏิบัติการจำนวน 3 ปฏิบัติการ ดังนี้

- ปฏิบัติการที่ 1 การทดสอบการเก็บข้อมูลผู้ใช้งาน
 ปฏิบัติการที่ 2 การทดสอบการเก็บข้อมูลและส่งข้อมูลผ่านเครื่องส่งสัญญาณไร้สาย
 ปฏิบัติการที่ 3 การทดสอบการใช้งานเครื่องสแกนลายนิ้วมือเพื่อเปิด - ปิดประตู ในการใช้งานจริง

4.2 การทดสอบการเก็บข้อมูลผู้ใช้งาน

4.2.1 วัตถุประสงค์

1. เพื่อทดสอบว่าเครื่องสแกนลายนิ้วมือสามารถสแกนและรับข้อมูลได้
2. เพื่อทดสอบว่าโปรแกรมที่ใช้สแกนลายนิ้วมือสามารถทำงานได้
3. เพื่อทดสอบว่าโปรแกรมที่ใช้ในการแสดงผลใช้งานได้

4.2.2 อุปกรณ์การทดสอบ

1. ชุดการทดลองที่สร้างขึ้นประกอบด้วย
 - เครื่องสแกนลายนิ้วมือ
 - บอร์ด AVR-ATMEGA128
 - จอแสดงผล
 - ตัวล็อคประตูแบบแม่เหล็ก
 - เครื่องรับ – ส่งสัญญาณไร้สาย
2. เครื่องคอมพิวเตอร์ 1 เครื่อง

4.2.3 ทฤษฎีที่เกี่ยวข้อง

สำหรับการใช้งานเครื่องสแกนลายนิ้วมือและบอร์ด AVR-ATMEGA128 สามารถทบทวนได้จากหัวข้อบทที่ 2

4.2.4 ขั้นตอนการทดสอบ

1. การติดตั้งอุปกรณ์การทดสอบ

1.1 ชุดการทดสอบที่ได้สร้างขึ้นประกอบด้วย

- เครื่องสแกนลายนิ้วมือ
- บอร์ด AVR-ATMEGA128
- จอแสดงผล
- ตัวลิ้อคประตูแบบแม่เหล็ก
- เครื่องรับ – ส่งสัญญาณไร้สาย

2. ทดสอบการทำงานของเครื่องสแกนลายนิ้วมือ

3.1 สร้าง ID ผู้ใช้งานขึ้นจากโปรแกรม

3.2 สแกนลายนิ้วมือของผู้ใช้ ID นั้นๆ จำนวน 3 ครั้ง ต่อ 1 ผู้ใช้

3. ทำการเก็บข้อมูลจนครบ 10 ผู้ใช้งาน

4.2.5 ตารางบันทึกผลการทดสอบ

id	Name	Access times
7	Autthapol Sillapakitkosol	2010-11-29 12:51:51
5	Channarong Prakobdee	2010-11-29 13:50:13
1	Poorithat Sutamma	2010-11-29 13:58:25
11	Rattaphong Sutamma	2010-11-29 14:05:29
3	Jaroen Lokruam	2010-11-29 14:12:34
13	Sompong Buntaem	2010-12-04 12:11:12
15	Ladda Hitayaso	2010-12-04 12:17:53
9	Sutthikiat Poomidit	2010-12-05 11:34:31
17	Rugpong Wunnawat	2010-12-05 11:49:51
19	Aniwat Sucksamorn	2010-12-05 12:31:41

ตารางบันทึกผลการทดสอบ : การทดสอบเก็บข้อมูลผู้ใช้งานจำนวน 10 คน

4.3 การทดสอบการเก็บข้อมูลและส่งข้อมูลผ่านเครื่องส่งสัญญาณไร้สาย

4.3.1 วัตถุประสงค์

เพื่อทดสอบว่าเครื่องส่งสัญญาณไร้สาย สามารถรับ - ส่งสัญญาณได้ไกลทั้งภายในและภายนอกอาคารตามทฤษฎี

4.3.2 อุปกรณ์การทดสอบ

1. ชุดการทดสอบที่สร้างขึ้นประกอบด้วย

- เครื่องสแกนลายนิ้วมือ
- บอร์ด AVR-ATMEGA128
- จอแสดงผล
- ตัวล๊อคประตูแบบแม่เหล็ก
- เครื่องรับ - ส่งสัญญาณไร้สาย

2. เครื่องคอมพิวเตอร์ 1 เครื่อง

4.3.3 ทฤษฎีที่เกี่ยวข้อง

สำหรับการใช้งานเครื่องรับ

- ส่งสัญญาณไร้สายสามารถทบทวนได้จากหัวข้อบทที่ 2.5

4.3.4 ขั้นตอนการทดสอบ

4.3.4.1 ทดสอบภายนอกอาคาร

1. เริ่มที่ระยะ 10 เมตร และเพิ่มขึ้นทีละ 10 เมตร จนครบ 100 เมตร
2. สแกนลายนิ้วมือและดูผลว่าสามารถรับ - ส่งสัญญาณได้หรือไม่เมื่อระยะห่างตามที่กำหนด
3. บันทึกผลการทดลองลงตารางบันทึกผลการทดลอง

4.3.4.2 การทดสอบภายในอาคาร

1. เริ่มที่ระยะ 10 เมตร และเพิ่มขึ้นทีละ 10 เมตร จนครบ 100 เมตร
2. สแกนลายนิ้วมือและดูผลว่าสามารถรับ - ส่งสัญญาณได้หรือไม่เมื่อระยะห่างตามที่กำหนด
3. บันทึกผลการทดลองลงตารางบันทึกผลการทดลอง

4.3.5 ตารางบันทึกผลการทดสอบ

ตารางบันทึกผลการทดสอบ : การทดสอบการเก็บข้อมูลและส่งข้อมูลผ่านเครื่องส่งสัญญาณแบบไร้สายผ่านพอร์ต RS 232

ที่โล่ง			ในอาคาร		
ระยะทาง (m)	สำเร็จ	ไม่สำเร็จ	ระยะทาง (m)	สำเร็จ	ไม่สำเร็จ
10	✓		10	✓	
20	✓		20	✓	
30	✓		30	✓	
40	✓		40	✓	
50	✓		50	✓	
60	✓		60	✓	
70	✓		70	✓	
80	✓		80	✓	
90	✓		90	✓	
100	✓		100	✓	

4.4 การทดสอบการใช้งานเครื่องสแกนลายนิ้วมือเพื่อเปิด-ปิดประตู ในการใช้งานจริง

4.4.1 วัตถุประสงค์

เพื่อทดสอบความผิดพลาดของเครื่องสแกนลายนิ้วมือและการรับ - ส่งข้อมูลผ่านทางสัญญาณ wireless

4.4.2 อุปกรณ์การทดสอบ

1. ชุดการทดสอบที่สร้างขึ้นประกอบด้วย

- เครื่องสแกนลายนิ้วมือ
- บอร์ด AVR-ATMEGA128
- จอแสดงผล
- ตัวล๊อคประตูแบบแม่เหล็ก
- เครื่องรับ - ส่งสัญญาณไร้สาย

2. เครื่องคอมพิวเตอร์ 1 เครื่อง

4.4.3 ทฤษฎีที่เกี่ยวข้อง

สำหรับการใช้เครื่องสแกนลายนิ้วมือ สามารถทบทวนได้จากหัวข้อที่ 2.5.1

4.4.4 ขั้นตอนการทดสอบ

1. เลือกผู้ปฏิบัติการ 2 คน โดยทำการทดสอบทีละคน
2. สแกนลายนิ้วมือเพื่อเปิด – ปิดระบบล็อก จำนวน 10 ครั้ง
3. ภายในการทดสอบ 10 ครั้งควรมีการผิดพลาดทั้งหมดกี่ครั้งและบันทึกผลลงตารางบันทึก

ผลการทดสอบ

4. ผู้ปฏิบัติการคนที่ 2 เริ่มทำการทดสอบตามข้อที่ 2 – 3 และบันทึกผลการทดสอบ

4.4.5 ตารางบันทึกผลการทดสอบ

การทดสอบการใช้งานเครื่องสแกนลายนิ้วมือของผู้ทดลองที่ 1

จำนวนรอบการทดลอง	การแสดงผลที่หน้าจอ	ผลการทดลอง	
		สำเร็จ	ไม่สำเร็จ
1	Accept : ID07	✓	
2	Accept : ID07	✓	
3	Accept : ID07	✓	
4	Accept : ID07	✓	
5	Accept : ID07	✓	
6	Accept : ID07	✓	
7	Accept : ID07	✓	
8	Failure		✓
9	Failure		✓
10	Accept : ID07	✓	

*Failure: เกิดจากการความผิดพลาดของเครื่องสแกนลายนิ้วมือ ที่อ่านข้อมูลได้ไม่ตรงตามที่ได้บันทึกข้อมูลไว้

การทดสอบการใช้งานเครื่องสแกนลายนิ้วมือของผู้ทดลองที่ 2

จำนวนรอบการทดลอง	การแสดงผลที่หน้าจอ	ผลการทดลอง	
		สำเร็จ	ไม่สำเร็จ
1	Accept : ID02	✓	
2	Accept : ID02	✓	
3	Accept : ID02	✓	
4	Accept : ID02	✓	
5	Accept : ID02	✓	
6	Accept : ID02	✓	
7	Too Small		✓
8	Accept : ID02	✓	
9	Accept : ID02	✓	
10	Accept : ID02	✓	

*Too Small: ความเข้มของเส้นลายนิ้วมือน้อยเกินไป (น้ำหนักของนิ้วมือในการสแกนเบาเกินไป)



บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 บทนำ

เนื้อหาในบทนี้จะเป็นการสรุปผลที่ได้จากการทำการทดสอบเครื่องสแกนลายนิ้วมือเพื่อเปิด - ปิดประตูทั้งหมด รวมไปถึงข้อเสนอแนะในการทำปฏิบัติการ

5.2 สรุปผล

จากการออกแบบการทดสอบและการทดสอบระบบของเครื่องสแกนลายนิ้วมือเพื่อเปิด - ปิดประตูสรุปได้ดังนี้

1. การทดสอบการเก็บข้อมูลผู้ใช้งาน ทดสอบผ่านคือสามารถเก็บข้อมูลได้อย่างถูกต้อง
2. การทดสอบการส่งข้อมูลผ่านเครื่องส่งสัญญาณไร้สาย ทดสอบผ่าน โดยสามารถส่งข้อมูลได้ครบตาม ระยะทางที่ทำการทดสอบ ทั้งภายในตัวอาคารและภายนอกอาคาร โดยระยะทางไกลสุดที่ได้ประมาณ 100 เมตร
3. การทดสอบการใช้งานจริงทดสอบผ่านคือ สามารถเปิด - ปิดระบบล็อกของประตูได้
4. สามารถบันทึกเวลาการสแกนลายนิ้วมือได้ทุกครั้งที่มีการสแกนลายนิ้วมือเพื่อผ่านประตู

5.3 ข้อเสนอแนะ

1. การรูดนิ้วของผู้ทำปฏิบัติการ บางครั้งรูดนิ้วเร็วหรือช้าเกินไป ทำให้การอ่านข้อมูลของเครื่องสแกนลายนิ้วมือเกิดการอ่านผิดพลาด
2. โปรแกรมการทำงานอาจมีข้อผิดพลาดบ้าง อาจเกิดจากข้อผิดพลาดของตัวโปรแกรมเอง
3. ตัวอุปกรณ์เกิดความร้อนเมื่อใช้งานเป็นระยะเวลาานานขึ้นอยู่กับสถานที่การติดตั้งอุปกรณ์

ประวัติผู้เขียน



นายอรรถพล ศิลปกิจโกศลโกศล
เกิดเมื่อวันที่ 23 ธันวาคม พ.ศ.2530
ภูมิลำเนาอยู่ที่ ตำบลหน้าเมือง อำเภอเมือง จังหวัดปราจีนบุรี
สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลาย
จากโรงเรียนปราจีนราษฎร์บำรุง อำเภอเมือง จังหวัดปราจีนบุรี
เมื่อปี พ.ศ. 2548
ปัจจุบันเป็นนักศึกษาชั้นปีที่ 5 สาขาวิชาวิศวกรรมโทรคมนาคม



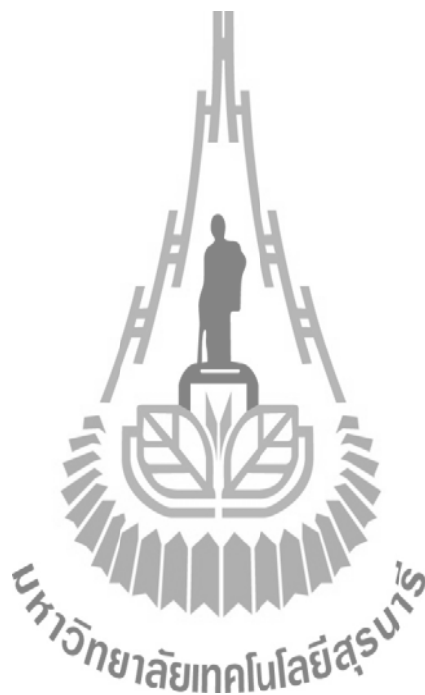
นายกฤษิต สุธรรมมา
เกิดเมื่อวันที่ 23 กุมภาพันธ์ พ.ศ.2532
ภูมิลำเนาอยู่ที่ ตำบลตลาด อำเภอเมือง จังหวัดมหาสารคาม
สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลาย
จากโรงเรียนดงใหญ่วิทยาคม รัชมังกลาภิเษก อำเภอลำปำ
จังหวัดมหาสารคาม เมื่อปี พ.ศ.2549
ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม



นายชาตณรงค์ ประกอบดี
เกิดเมื่อวันที่ 23 พฤษภาคม พ.ศ.2532
ภูมิลำเนาอยู่ที่ ตำบลไชยสอ อำเภอลำทะเมนชัย จังหวัดขอนแก่น
สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลาย
จากโรงเรียนชุมแพศึกษา อำเภอลำทะเมนชัย จังหวัดขอนแก่น
เมื่อปี พ.ศ.2549
ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม

บรรณานุกรม

- คู่มือการพัฒนาด้วย AVR Studio จาก <http://www.etteam.com/product/03A21.html>
- คู่มือการใช้งานวงจรมicro AVR-ATMEGA128 บริษัท อีทีที จำกัด
- คู่มือตัวรับ – ส่งสัญญาณไร้สาย จาก <http://www.etteam.com/product/intf/man-ET-RF24Gv2.pdf>
- คู่มือการใช้งานโปรแกรม Code Vision AVR V.2.03.4

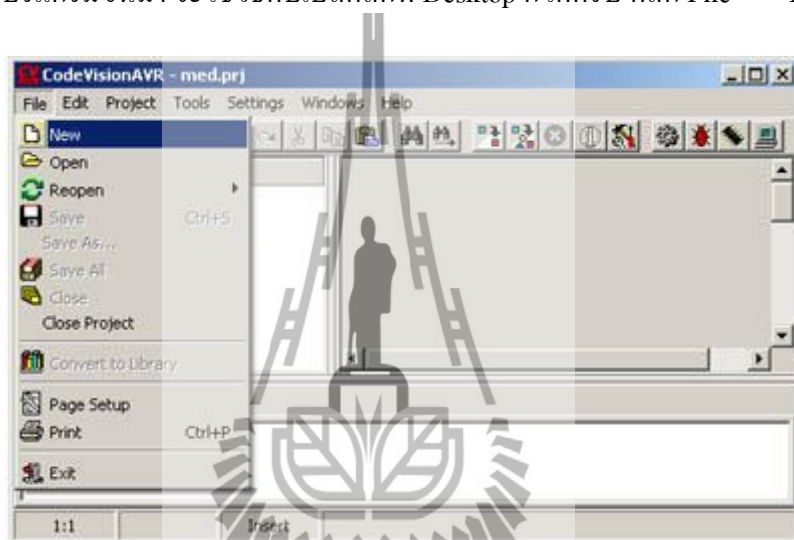


เริ่มต้นใช้งาน Code Vision AVR



เมื่อพูดถึงไมโครคอนโทรลเลอร์ในบ้านเรา AVR จะเป็นอีกหนึ่งตระกูลที่มีผู้ใช้เป็นจำนวนมาก ในการเขียนโปรแกรมสามารถเขียนได้หลายภาษาเช่นกัน ในวันนี้ผมจะแนะนำการเริ่มต้นใช้งาน Code Vision AVR ตัวนี้จะเป็น Software ลิขสิทธิ์นะครับ แต่สามารถทดลองใช้ได้โดยการ Download ตัว demo จากเว็บไซต์มาลองใช้

1. ทำการเปิดโปรแกรมขึ้นมา จะใช้วิธีดับเบิลคลิกที่ Desktop ก็ได้ครับ คลิก File → New



รูปที่ ก.1

2. เลือก Project แล้วกด OK



รูปที่ ก.2



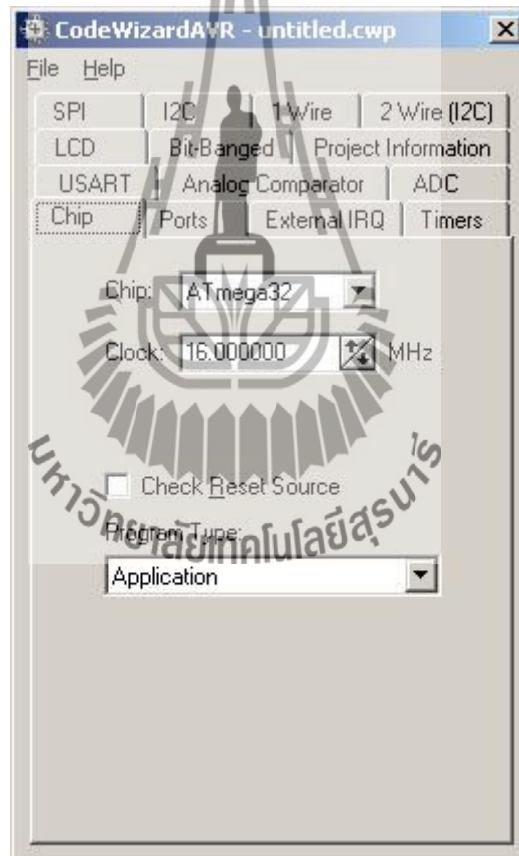


3. จะมีหน้าต่างมาถามเราว่าต้องจะสร้างโปรเจกต์ด้วย Code Wizard AVR หรือเปล่า ให้กด Yes ไปครับ ตรงนี้จะเป็น ลักษณะที่เด่นอย่างหนึ่งของ Code Vision AVR เราสามารถ initial & Config ได้ โดยการคลิกๆ



รูปที่ ก.3

4. เราจะเจอ Tab อยู่หลายอย่าง ให้ตั้งค่าตามที่ต้องการ



รูปที่ ก.4

- **Chip** เบอร์ AVR ที่ใช้งาน, ความถี่ที่ใช้ ชนิดของ โปรแกรม
- **Ports** ไว้กำหนดว่าเป็นอินพุต/เอาต์พุต Pull up
- **External IRQ** Enable ขาอินเทอร์รัปภายนอก

- **Timers** ตั้งรูปแบบการทำงาน Timer รวมทั้ง Watchdog
- **USART** การรับส่งข้อมูลผ่านพอร์ตอนุกรม
- **Analog Comparator** ใช้/ไม่ใช่ Analog Comparator
- **ADC** ตั้งค่า ADC ภายในของ AVR
- **LCD** กำหนดพอร์ตที่ใช้งานจอ LCD มีฟังก์ชัน LCD ให้ ไม่ต้องเขียนฟังก์ชันเอง
- **Bit-Banged** กำหนดพอร์ต ที่ใช้ติดต่อกับ DS1302
- **Project Information** ไว้เขียนรายละเอียดเกี่ยวกับโปรเจก
- **SPI** ตั้งค่าการใช้งาน SPI
- **I2C** ตั้งค่าการใช้งาน I2C เลือก I2C device
- **1 Wire** ตั้งค่าการใช้งาน 1 Wire (DS1820)
- **2 Wire (I2C)** ตั้งค่าเพิ่มเติม I2C

5. หลังจากที่ตั้งค่าเสร็จแล้ว ให้เลือก Generate, Save and Exit โปรแกรมจะทำการ Generate Code ตามที่เราตั้งค่าไว้ในขั้นตอนที่ 4



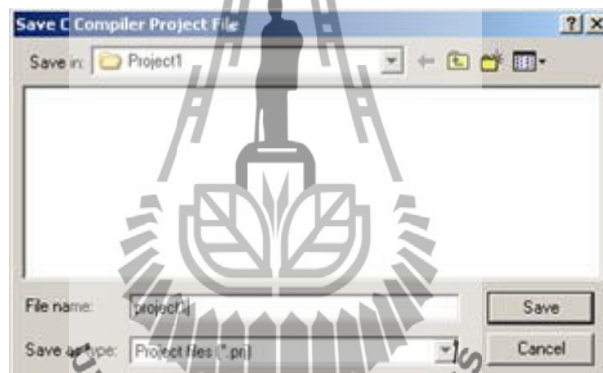
รูปที่ ก.5

6. ตั้งชื่อ File โดยปกติจะตั้งชื่อว่า main



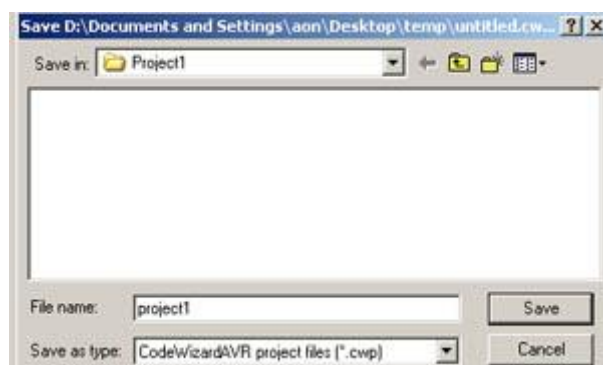
รูปที่ 6.6

7. ตั้งชื่อโปรเจก



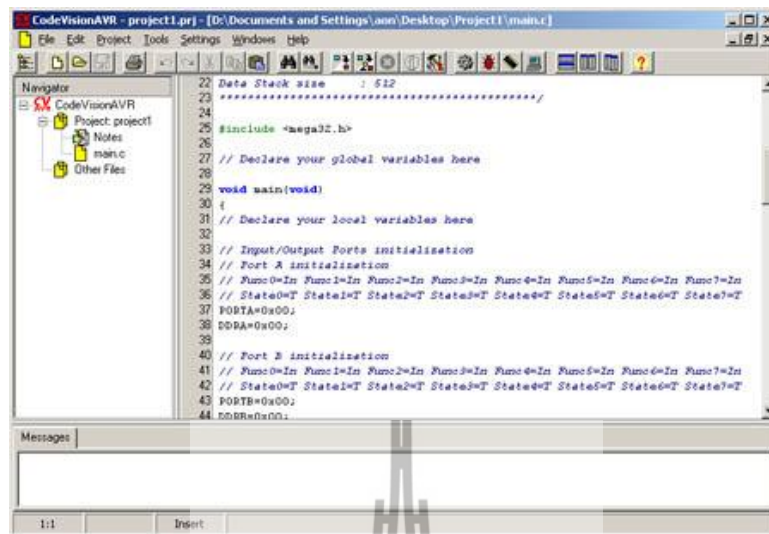
รูปที่ 6.7

8. ตั้งชื่อ Code Wizard AVR โปรเจก



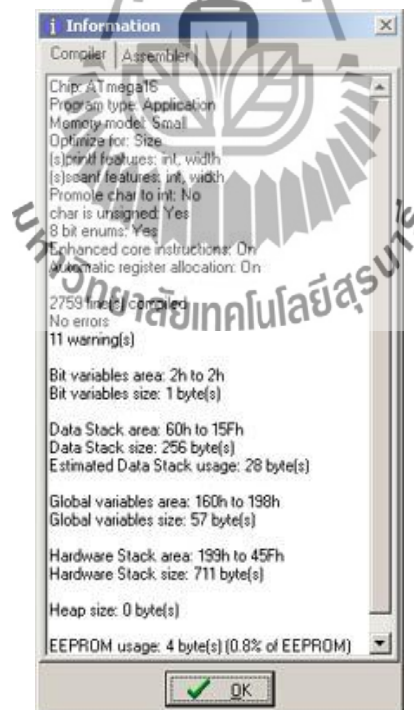
รูปที่ 6.8

9. หลังจากคลิก Save ในขั้นตอนที่ 8 แล้ว โปรแกรมจะ Generate โปรแกรมมาให้ เราสามารถเพิ่ม Code เพิ่ม File .h, .C เข้าไปในโปรเจกต์ได้เลย



รูปที่ ก.9

10. เมื่อเขียนโปรแกรมเสร็จแล้วให้กด Shift+F9 เพื่อ Compile โปรแกรมที่เราเขียนไป ถ้าไม่มีข้อผิดพลาดจะเห็น หน้าต่างขึ้นมาดังรูปที่ 10



รูปที่ ก.10

11. File .HEX ที่ได้มาจากการ Compile จะอยู่ในโฟลเดอร์โปรเจกต์ที่เราเซฟไว้ หลังจากนั้นเรานำ .HEX ไป burn ใส่ IC เป็นอันเสร็จครับ

Code โปรแกรม

```

1      #include <mega128.h>
2      #include <delay.h>
3      #include <string.h>
4      #include "lcd.h"
5
6      #define door_delay 5000
7
8      #define RXB8 1
9      #define TXB8 0
10     #define UPE 2
11     #define OVR 3
12     #define FE 4
13     #define UDRE 5
14     #define RXC 7
15
16     #define FRAMING_ERROR (1<<FE)
17     #define PARITY_ERROR (1<<UPE)
18     #define DATA_OVERRUN (1<<OVR)
19     #define DATA_REGISTER_EMPTY (1<<UDRE)
20     #define RX_COMPLETE (1<<RXC)
21
22     // USART0 Receiver buffer
23     #define RX_BUFFER_SIZE0 1024
24
25     void putchar1(char c);
26     void send_serial(char str[32]);
27     unsigned char mode=1,mode_p=1;
28     char rx_buffer0[RX_BUFFER_SIZE0];
29
30     #if RX_BUFFER_SIZE0<256
31     unsigned char rx_wr_index0,rx_rd_index0,rx_counter0;

```

```

32     #else
33     unsigned int rx_wr_index0,rx_rd_index0,rx_counter0;
34     #endif
35
36     // This flag is set on USART0 Receiver buffer overflow
37     bit rx_buffer_overflow0;
38
39     // USART0 Receiver interrupt service routine
40     interrupt [USART0_RXC] void usart0_rx_isr(void)
41     {
42     char status, data;
43     status=UCSR0A;
44     data=UDR0;
45     if ((status & (FRAMING_ERROR | PARITY_ERROR |
46     DATA_OVERRUN))==0)
47     {
48     rx_buffer0[rx_wr_index0]=data;
49     if (++rx_wr_index0 == RX_BUFFER_SIZE0) rx_wr_index0=0;
50     if (++rx_counter0 == RX_BUFFER_SIZE0)
51     {
52     rx_counter0=0;
53     rx_buffer_overflow0=1;
54     };
55     };
56     }
57
58     #ifndef _DEBUG_TERMINAL_IO_
59     // Get a character from the USART0 Receiver buffer
60     #define _ALTERNATE_GETCHAR_
61     #pragma used+
62     char getchar(void)
63     {
64     char data;

```

```

64     while (rx_counter0==0);
65     data=rx_buffer0[rx_rd_index0];
66     if (++rx_rd_index0 == RX_BUFFER_SIZE0) rx_rd_index0=0;
67     #asm("cli")
68     --rx_counter0;
69     #asm("sei")
70     return data;
71     }
72     #pragma used-
73     #endif
74
75     // USART1 Receiver buffer
76     #define RX_BUFFER_SIZE1 512
77     char rx_buffer1[RX_BUFFER_SIZE1];
78
79     #if RX_BUFFER_SIZE1<256
80     unsigned char rx_wr_index1,rx_rd_index1,rx_counter1;
81     #else
82     unsigned int rx_wr_index1,rx_rd_index1,rx_counter1;
83     #endif
84
85     // This flag is set on USART1 Receiver buffer overflow
86     bit rx_buffer_overflow1;
87
88     // USART1 Receiver interrupt service routine
89     interrupt [USART1_RXC] void usart1_rx_isr(void)
90     {
91     char status,data;
92     status=UCSR1A;
93     data=UDR1;
94     if((status & (FRAMING_ERROR | PARITY_ERROR |
95     DATA_OVERRUN))==0)

```



```

96     rx_buffer1[rx_wr_index1]=data;
97     if(++rx_wr_index1 == RX_BUFFER_SIZE1) rx_wr_index1=0;
98     if(++rx_counter1 == RX_BUFFER_SIZE1)
99         {
100         rx_counter1=0;
101         rx_buffer_overflow1=1;
102         };
103     };
104 }
105
106 // Get a character from the USART1 Receiver buffer
107 #pragma used+
108 char getchar1(void)
109 {
110     char data;
111     while (rx_counter1==0);
112     data=rx_buffer1[rx_rd_index1];
113     if(++rx_rd_index1 == RX_BUFFER_SIZE1) rx_rd_index1=0;
114     #asm("cli")
115     --rx_counter1;
116     #asm("sei")
117     return data;
118 }
119 #pragma used-
120 // Write a character to the USART1 Transmitter
121 #pragma used+
122 void putchar1(char c)
123 {
124     while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
125     UDR1=c;
126 }
127 #pragma used-
128

```

```

129 // Standard Input/Output functions
130 #include <stdio.h>
131
132 // Declare your global variables here
133
134 void main(void)
135 {
136 // Declare your local variables here
137 char lcdbuf[16+1]; // LCD Display Buffer
138 char strbuf[32];
139 unsigned char rcm[2],len[2],data[16],chk,save_id,loop;
140 char pc_com[3];
141
142 // Input/Output Ports initialization
143 // Port A initialization
144 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
145 Func0=In
146 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
147 State0=T
148 PORTA=0xFF;
149 DDRA=0xFF;
150
151 // Port B initialization
152 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
153 Func0=In
154 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
155 State0=T
156 PORTB=0xFF;
157 DDRB=0xFF;
158
159 // Port C initialization
160 // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
161 Func0=In
162 // State7=T State6=T State5=T State4=T State3=T State2=T State1=T

```

```

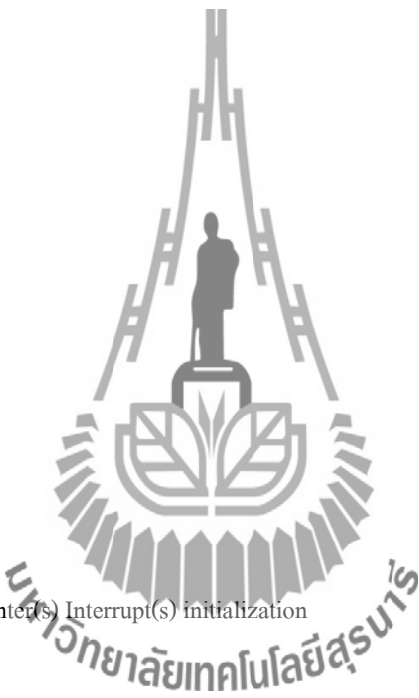
State0=T
158   PORTC=0x00;
159   DDRC=0x00;
160
161   // Port D initialization
      // Func7=Out Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
162   Func0=In
      // State7=0 State6=T State5=T State4=T State3=T State2=T State1=T
163   State0=T
164   PORTD=0x00;
165   DDRD=0x80;
166
167   // Port E initialization
      // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=Out Func1=In
168   Func0=In
      // State7=T State6=T State5=T State4=T State3=T State2=0 State1=T
169   State0=T
170   PORTE=0x00;
171   DDRE=0x04;
172
173   // Port F initialization
      // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
174   Func0=In
      // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
175   State0=T
176   PORTF=0x00;
177   DDRF=0x00;
178
179   // Port G initialization
180   // Func4=In Func3=In Func2=In Func1=In Func0=In
181   // State4=T State3=T State2=T State1=T State0=T
182   PORTG=0xFF;
183   DDRG=0xFF;
184

```

```
185 // Timer/Counter 0 initialization
186 // Clock source: System Clock
187 // Clock value: Timer 0 Stopped
188 // Mode: Normal top=FFh
189 // OC0 output: Disconnected
190 ASSR=0x00;
191 TCCR0=0x00;
192 TCNT0=0x00;
193 OCR0=0x00;
194
195 // Timer/Counter 1 initialization
196 // Clock source: System Clock
197 // Clock value: Timer 1 Stopped
198 // Mode: Normal top=FFFFh
199 // OC1A output: Discon.
200 // OC1B output: Discon.
201 // OC1C output: Discon.
202 // Noise Canceler: Off
203 // Input Capture on Falling Edge
204 // Timer 1 Overflow Interrupt: Off
205 // Input Capture Interrupt: Off
206 // Compare A Match Interrupt: Off
207 // Compare B Match Interrupt: Off
208 // Compare C Match Interrupt: Off
209 TCCR1A=0x00;
210 TCCR1B=0x00;
211 TCNT1H=0x00;
212 TCNT1L=0x00;
213 ICR1H=0x00;
214 ICR1L=0x00;
215 OCR1AH=0x00;
216 OCR1AL=0x00;
217 OCR1BH=0x00;
```

```
218     OCR1BL=0x00;
219     OCR1CH=0x00;
220     OCR1CL=0x00;
221
222     // Timer/Counter 2 initialization
223     // Clock source: System Clock
224     // Clock value: Timer 2 Stopped
225     // Mode: Normal top=FFh
226     // OC2 output: Disconnected
227     TCCR2=0x00;
228     TCNT2=0x00;
229     OCR2=0x00;
230
231     // Timer/Counter 3 initialization
232     // Clock source: System Clock
233     // Clock value: Timer 3 Stopped
234     // Mode: Normal top=FFFFh
235     // Noise Canceler: Off
236     // Input Capture on Falling Edge
237     // OC3A output: Discon.
238     // OC3B output: Discon.
239     // OC3C output: Discon.
240     // Timer 3 Overflow Interrupt: Off
241     // Input Capture Interrupt: Off
242     // Compare A Match Interrupt: Off
243     // Compare B Match Interrupt: Off
244     // Compare C Match Interrupt: Off
245     TCCR3A=0x00;
246     TCCR3B=0x00;
247     TCNT3H=0x00;
248     TCNT3L=0x00;
249     ICR3H=0x00;
250     ICR3L=0x00;
```

```
251     OCR3AH=0x00;
252     OCR3AL=0x00;
253     OCR3BH=0x00;
254     OCR3BL=0x00;
255     OCR3CH=0x00;
256     OCR3CL=0x00;
257
258     // External Interrupt(s) initialization
259     // INT0: Off
260     // INT1: Off
261     // INT2: Off
262     // INT3: Off
263     // INT4: Off
264     // INT5: Off
265     // INT6: Off
266     // INT7: Off
267     EICRA=0x00;
268     EICRB=0x00;
269     EIMSK=0x00;
270
271     // Timer(s)/Counter(s) Interrupt(s) initialization
272     TIMSK=0x00;
273     ETIMSK=0x00;
274
275     // USART0 initialization
276     // Communication Parameters: 8 Data, 1 Stop, No Parity
277     // USART0 Receiver: On
278     // USART0 Transmitter: On
279     // USART0 Mode: Asynchronous
280     // USART0 Baud Rate: 19200
281     UCSR0A=0x00;
282     UCSR0B=0x98;
283     UCSR0C=0x06;
```



```

284     UBRR0H=0x00;
285     UBRR0L=0x33;
286
287     // USART1 initialization
288     // Communication Parameters: 8 Data, 1 Stop, No Parity
289     // USART1 Receiver: On
290     // USART1 Transmitter: On
291     // USART1 Mode: Asynchronous
292     // USART1 Baud Rate: 9600
293     UCSR1A=0x00;
294     UCSR1B=0x98;
295     UCSR1C=0x06;
296     UBRR1H=0x00;
297     UBRR1L=0x67;
298
299     // Analog Comparator initialization
300     // Analog Comparator: Off
301     // Analog Comparator Input Capture by Timer/Counter 1: Off
302     ACSR=0x80;
303     SFIOR=0x00;
304
305     //for(delay=300;delay>0;delay--);
306     delay_ms(30); // Power-on Delay
307     init_lcd(); // Initial LCD
308
309     // Global enable interrupts
310     #asm("sei")
311
312
313     //gotolcd(0); // Set Cursor Line-1
314     //sprintf(lcdbuf,"Test2"); // Display Line-1
315     //printlcd(lcdbuf);
316     //gotolcd(0x40); // Set Cursor Line-2

```

```
317 //sprintf(lcdbuf,"LCD2"); // Display Line-2
318 //printf(lcdbuf);
319 PORTE.2 = 1;
320 delay_ms(500);
321
322 lcd_clear();
323 sprintf(lcdbuf,"Ready to Scan."); // Display Line-2
324 printf(lcdbuf);
325 gotolcd(0x40); // Set Cursor Line-2
326 sprintf(lcdbuf,"Push the Button."); // Display Line-2
327 printf(lcdbuf);
328
329 while (1)
330 {
331     if(mode!=0){
332         if(mode==1){
333             lcd_clear();
334             sprintf(lcdbuf,"Swap the Finger..."); // Display Line-2
335             printf(lcdbuf);
336             //Send Identify Command
337             putchar(0x01);
338             putchar(0x02);
339             putchar(0x00);
340             putchar(0x00);
341             putchar(0x03);
342             //Get Response
343             rcm[0] = getchar();
344             rcm[1] = getchar();
345             len[0] = getchar();
346             len[1] = getchar();
347             data[0] = getchar();
348             data[1] = getchar();
349             chk = getchar();
```



```

350         //Check Response
           if(rcm[0]==0x01 && rcm[1]==0x02 && len[0]==0x00 &&
351 len[1]==0x02){
352         if(data[1] == 0x00){//Reject
353             gotolcd(0x40); // Set Cursor Line-2
354             sprintf(lcdbuf,"ERROR:Failure"); // Display Line-2
355             printlcd(lcdbuf);
356             delay_ms(1000);
357         }else if(data[1] == 0x01){//Accept
358             gotolcd(0x40); // Set Cursor Line-2
359             sprintf(lcdbuf,"Accept ID:%02d",data[0]); // Display Line-2
360             printlcd(lcdbuf);
361             sprintf(strbuf,"A%02d",data[0]);
362             send_serial(strbuf);
363             PORTB = 0x00;
364             delay_ms(door_delay);
365             PORTB = 0xFF;
366             //putchar1('0');
367         }else if(data[1] == 0x02){//Time Out
368             gotolcd(0x40); // Set Cursor Line-2
369             sprintf(lcdbuf,"ERROR:Time out",mode); // Display Line-2
370             printlcd(lcdbuf);
371         }else if(data[1] == 0x03){//Bad Quality
372             gotolcd(0x40); // Set Cursor Line-2
373             sprintf(lcdbuf,"ERROR:Bad Quality",mode); // Display Line-2
374             printlcd(lcdbuf);
375             delay_ms(1000);
376         }else if(data[1] == 0x05){//Too Small
377             gotolcd(0x40); // Set Cursor Line-2
378             sprintf(lcdbuf,"ERROR:Too Small",mode); // Display Line-2
379             printlcd(lcdbuf);
380             delay_ms(1000);
381         }else if(data[1] == 0x06){//Empty

```

```

382         gotolcd(0x40); // Set Cursor Line-2
383         sprintf(lcdbuf,"ERROR:Empty",mode); // Display Line-2
384         printlcd(lcdbuf);
385         delay_ms(1000);
386     }
387 }
388 //Restore Mode
389 mode_p = mode;
390 mode = 1;
391 }
392 }
393
394 if(rx_counter1>0){
395     pc_com[0] = getchar();
396     pc_com[1] = getchar();
397     pc_com[2] = getchar();
398     if(pc_com[0] == 'S'){
399         save_id = ((pc_com[1]-0x30)*10)+(pc_com[2]-0x30);
400         gotolcd(0x40); // Set Cursor Line-2
401         sprintf(lcdbuf,"Clearing id:%d ",save_id); // Display Line-2
402         printlcd(lcdbuf);
403         //Send clear template Command
404         putchar(0x01);
405         putchar(0x05);
406         putchar(0x00);
407         putchar(0x01);
408         putchar(save_id);
409         putchar(0x07+save_id);
410         //Get Response
411         rcm[0] = getchar();
412         rcm[1] = getchar();
413         len[0] = getchar();
414         len[1] = getchar();

```

```
415     data[0] = getchar();
416     data[1] = getchar();
417     chk = getchar();
418     //Check Response
         if(rcm[0]==0x01 && rcm[1]==0x05 && len[0]==0x00 &&
419 len[1]==0x02){
420         if(data[1] == 0x00){//Already Empty
421             goto lcd(0x40); // Set Cursor Line-2
422             sprintf(lcdbuf,"Already Empty."); // Display Line-2
423             printlcd(lcdbuf);
424         } else if(data[1] == 0x01){//Success
425             goto lcd(0x40); // Set Cursor Line-2
426             sprintf(lcdbuf,"Cleared ID:%d.",data[0]); // Display Line-2
427             printlcd(lcdbuf);
428         } else if(data[1] == 0xFF){//Invalid
429             goto lcd(0x40); // Set Cursor Line-2
430             sprintf(lcdbuf,"Invalid Template.",mode); // Display Line-2
431             printlcd(lcdbuf);
432         } else if(data[1] == 0xA0){//Flash Error
433             goto lcd(0x40); // Set Cursor Line-2
434             sprintf(lcdbuf,"Writing Error",mode); // Display Line-2
435             printlcd(lcdbuf);
436         }
437     }
438     //End Clear Process
439     //Send Enroll command
440     goto lcd(0x40); // Set Cursor Line-2
441     sprintf(lcdbuf,"Starting Enroll."); // Display Line-2
442     printlcd(lcdbuf);
443     //Send clear template Command
444     putchar(0x01);
445     putchar(0x03);
446     putchar(0x00);
```

```

447     putchar(0x01);
448     putchar(save_id);
449     putchar(0x05+save_id);
450     //Get Response
451     loop = 0;
452     while(loop==0){
453         //delay_ms(4000);
454         rcm[0] = getchar();
455         rcm[1] = getchar();
456         len[0] = getchar();
457         len[1] = getchar();
458         data[0] = getchar();
459         data[1] = getchar();
460         chk = getchar();
461         //Check Response
462         if(rcm[0]==0x01 && rcm[1]==0x03 && len[0]==0x00 &&
len[1]==0x02){
463             if(data[1] == 0x00){//Fail
464                 loop = 1;
465                 gotolcd(0x40); // Set Cursor Line-2
466                 sprintf(lcdbuf,"Fail Generalize."); // Display Line-2
467                 printf(lcdbuf);
468                 putchar1('F');
469             }else if(data[1] == 0x01){//Success
470                 loop = 1;
471                 gotolcd(0x40); // Set Cursor Line-2
472                 sprintf(lcdbuf,"Success ID:%d. ",save_id); // Display Line-2
473                 printf(lcdbuf);
474                 putchar1('O');
475             }else if(data[1] == 0x02){//Timeout
476                 loop = 1;
477                 gotolcd(0x40); // Set Cursor Line-2
478                 sprintf(lcdbuf,"Time Out.    "); // Display Line-2

```

```
479         printlcd(lcdbuf);
480         putchar1('F');
481     }else if(data[1] == 0x03){//Bad Quality
482         gotolcd(0x40); // Set Cursor Line-2
483         sprintf(lcdbuf,"Bad Quality. "); // Display Line-2
484         printlcd(lcdbuf);
485     }else if(data[1] == 0x04){//Too Fast
486         gotolcd(0x40); // Set Cursor Line-2
487         sprintf(lcdbuf,"Sweep too fast. "); // Display Line-2
488         printlcd(lcdbuf);
489     }else if(data[1] == 0x05){//Small Lines
490         gotolcd(0x40); // Set Cursor Line-2
491         sprintf(lcdbuf,"Too Small Lines "); // Display Line-2
492         printlcd(lcdbuf);
493     }else if(data[1] == 0x06){//Second Sweep
494         gotolcd(0x40); // Set Cursor Line-2
495         sprintf(lcdbuf,"Two Sweep Left. "); // Display Line-2
496         printlcd(lcdbuf);
497     }else if(data[1] == 0x07){//Third Sweep
498         gotolcd(0x40); // Set Cursor Line-2
499         sprintf(lcdbuf,"One Sweep Left. "); // Display Line-2
500         printlcd(lcdbuf);
501     }else if(data[1] == 0xA1){//First Sweep
502         gotolcd(0x40); // Set Cursor Line-2
503         sprintf(lcdbuf,"First Sweep. "); // Display Line-2
504         printlcd(lcdbuf);
505     }else if(data[1] == 0xA2){//Duplication
506         loop = 1;
507         gotolcd(0x40); // Set Cursor Line-2
508         sprintf(lcdbuf,"Duplication. "); // Display Line-2
509         printlcd(lcdbuf);
510         putchar1('F');
511     }else if(data[1] == 0x08){//Not Empty
```

```
512     loop = 1;
513     gotolcd(0x40); // Set Cursor Line-2
514     sprintf(lcdbuf,"Not Empty.  "); // Display Line-2
515     printlcd(lcdbuf);
516     putchar1('F');
517 }else if(data[1] == 0xFF){//Invalid
518     loop = 1;
519     gotolcd(0x40); // Set Cursor Line-2
520     sprintf(lcdbuf,"Invalid.  "); // Display Line-2
521     printlcd(lcdbuf);
522     putchar1('F');
523 }else if(data[1] == 0xA0){//Writing Error.
524     loop = 1;
525     gotolcd(0x40); // Set Cursor Line-2
526     sprintf(lcdbuf,"Writing Error.  "); // Display Line-2
527     printlcd(lcdbuf);
528     putchar1('F');
529 }
530 }
531 //End Enroll command
532 //Restore Mode
533 delay_ms(500);
534 }
535 }
536 }
537 };
538 }
539
540 void send_serial(char str[32])
541 {
542     unsigned char count;
543     for(count=0;count<strlen(str);count++){
544         putchar1(str[count]);
```

545 }
546 }
547
548
549

