

ระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต
AIR CONDITIONING CONTROL SYSTEM VIA INTERNET



รัชพงษ์ พุดทรา
สุรัสวดี สุดประเสริฐ
อชิตา รัตนภรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมเอ็ดโมเนติกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2555

ระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต
AIR CONDITIONING CONTROL SYSTEM VIA INTERNET



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมอัตโนมัติ

คณะวิศวกรรมศาสตร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
ปีการศึกษา 2555

AIR CONDITIONING CONTROL SYSTEM VIA INTERNET



A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN AUTOMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2012

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้ภายในเท่านั้น ไม่ควรนำออกนอกห้องเรียนโดยไม่ได้รับอนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น

ใบรับรองปริญญาโท

หัวข้อปริญญาโท ระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต
AIR CONDITIONING CONTROL SYSTEM VIA INTERNET

นักศึกษาผู้จัดทำ นายรัชพงษ์ พุดทรา รหัสนักศึกษา 52010988
นางสาวสุรัสวดี สุดประเสริฐ รหัสนักศึกษา 52011345
นางสาวอชิตา รัตนภรณ์ รหัสนักศึกษา 52011366

ปริญญา วิศวกรรมศาสตรบัณฑิต
สาขาวิชา วิศวกรรมอัตโนมัติ
ปีการศึกษา 2555

อาจารย์ผู้ควบคุมปริญญาโท	ลายมือชื่อ
รองศาสตราจารย์ประภาส อุคคกิมพันธ์	
อาจารย์ฤกษ์ณี เสมอพิทักษ์	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	ระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต AIR CONDITIONING CONTROL SYSTEM VIA INTERNET		
นักศึกษาผู้จัดทำ	นายรัชพงษ์	พุดทรา	รหัสนักศึกษา 52010988
	นางสาวสุรัสวดี	สุดประเสริฐ	รหัสนักศึกษา 52011345
	นางสาวอชิตา	รัตนภรณ์	รหัสนักศึกษา 52011366
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ประภาช อุดคภิมาพันธ์ อาจารย์กฤษณ์ เสมอพิทักษ์		
ปีการศึกษา	2555		

บทคัดย่อ

โครงการนี้เป็นการศึกษาโปรโตคอลและชุดคำสั่งที่ใช้ในการสื่อสารระหว่างรีโมทคอนโทรลกับเครื่องใช้ไฟฟ้า และออกแบบ สร้างอุปกรณ์ควบคุมเครื่องใช้ไฟฟ้าที่ใช้รีโมทคอนโทรลแบบส่งสัญญาณอินฟราเรดในการควบคุม โดยเป็นการประยุกต์ใช้ไมโครคอนโทรลเลอร์ร่วมกับการประยุกต์ใช้เทคโนโลยีไร้สายเข้าด้วยกัน ทำให้เกิดการสร้างเครือข่ายของการควบคุมอุปกรณ์ไฟฟ้าขึ้น ซึ่งสามารถจัดการอุปกรณ์ไฟฟ้าผ่านศูนย์ควบคุม โดยใช้หลักการจำลองการทำงานของรีโมทที่ส่งสัญญาณอินฟราเรดในการสั่งงานควบคุมเครื่องใช้ไฟฟ้า และสามารถเชื่อมต่อกับรูปแบบการสื่อสารนี้ผ่านระบบอินเทอร์เน็ตได้อีกด้วย ซึ่งจะทำให้เกิดความคล่องตัวในการใช้งาน รวมถึงสามารถบริหารจัดการอุปกรณ์ไฟฟ้าภายในบ้านได้อย่างง่ายดาย โดยปริญญานิพนธ์นี้ได้ทำการทดลองกับเครื่องปรับอากาศเพื่อเป็นแนวทางที่สามารถประยุกต์ใช้ในการควบคุมเครื่องใช้ไฟฟ้าอื่นๆ ที่สามารถรับสัญญาณอินฟราเรดได้อีกด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title	AIR CONDITIONING CONTROL SYSTEM VIA INTERNET	
Authors	Mr. Ratchaphong	Pudtha
	Ms. Surassawadee	Sudprasert
	Ms. Achita	Rattanaorn
Thesis Advisor	Assoc. Prof. Prapart	Ukakiaparn
	Mr. Krit	Smerpitak
Year	2012	

ABSTRACT

This project is a study protocol and command set used to communicate between the remote control for home appliances and electrical equipment designed for remote control infrared transmitter to control. By the application of microcontrollers with application of wireless technologies together. Cause a build-up of electrical control network. Which can be managed through the control center equipment. Using the simulation of infrared remote control signal in order to control electrical appliances. Form of communication and can be connected through the Internet as well. This allows greater flexibility in use. Including home electronics can manage easily. The project was conducted with air conditioning to an approach that can be applied to control other appliances that can receive infrared signals as well.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี คณะผู้จัดทำขอกราบขอบพระคุณอาจารย์ที่ปรึกษา รองศาสตราจารย์ประภาช อุดคคิมาพันธ์ และอาจารย์กฤษณ์ เสมอพิทักษ์ ที่ได้ให้คำปรึกษา ชี้แนะแนวทางในการแก้ปัญหา ให้ความดูแลเอาใจใส่ ให้คำปรึกษา ความคิดริเริ่มสร้างสรรค์ ถ่ายทอดประสบการณ์การทำงาน ให้แนวคิดและทฤษฎีที่คณะผู้จัดทำไม่เคยได้รู้มาก่อน ตลอดจนสอบถามถึงความก้าวหน้าอย่างสม่ำเสมอ

ขอขอบพระคุณคณาจารย์ภาควิชาวิศวกรรมการวัดและควบคุม สาขาวิศวกรรมอัตโนมัติทุกท่าน ที่ได้ให้คำแนะนำ และการช่วยเหลืออันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

ขอขอบคุณเพื่อนๆ ทุกคนที่ให้กำลังใจ คอยกระตุ้นถามไถ่ความคืบหน้าของปริญญานิพนธ์ ซึ่งเป็นเหมือนแรงกระตุ้นเป็นอย่างดี จนสามารถทำให้โครงการนี้สำเร็จไปได้ด้วยดี

และสุดท้าย ขอกราบขอบพระคุณพ่อ และแม่อันเป็นที่รักยิ่ง ซึ่งคอยสนับสนุนและเป็นแรงบันดาลใจในการทำโครงการนี้ คุณค่าและประโยชน์ของโครงการนี้ คณะผู้จัดทำขอมอบให้แก่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

	หน้า
บทคัดย่อภาษาไทย.....	I
บทคัดย่อภาษาอังกฤษ.....	II
กิตติกรรมประกาศ.....	III
สารบัญ.....	IV
สารบัญตาราง.....	VIII
สารบัญรูป.....	IX

บทที่ 1 บทนำ.....	1
1.1 ความสำคัญของปริญญาโท.....	1
1.2 วัตถุประสงค์ของปริญญาโท.....	1
1.3 ขอบเขตของปริญญาโท.....	2
1.4 ขั้นตอนการศึกษา.....	2
1.5 ประโยชน์ที่คาดว่าจะได้รับของโครงการวิจัย.....	2
1.6 รายละเอียดของปริญญาโท.....	2

บทที่ 2 ทฤษฎีและหลักการ.....	4
2.1 กล่าวนำ.....	4
2.2 หลักการทำงานของรีโมตคอนโทรล.....	4
2.2.1 สัญญาณที่รีโมตคอนโทรลส่งออกมา.....	6
2.3 เทคโนโลยี E-House.....	7
2.3.1 โมดูล nRF24LE1.....	7
2.3.2 Enhanced ShockBurst™.....	7
2.3.3 โทโพลยีในการสื่อสารของระบบเครือข่าย.....	9
2.3.4 โครงสร้างการติดต่อสื่อสารกันภายในระบบเครือข่ายย่อย.....	10
2.4 การสื่อสารข้อมูลอนุกรม (UART).....	11
2.4.1 RS-232 (Recommended Standard 232).....	12
2.5 พัลส์วิตมอดูเลชัน (PWM).....	12
2.5.1 ประเภทของการมอดูเลต.....	13
2.5.2 คุณสมบัติที่สำคัญของสัญญาณเอพเอ็ม.....	14

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ การใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้เผยแพร่หรือใช้ซ้ำโดยไม่ได้รับอนุญาต
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปะลงในสื่ออื่น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มาเป็นใช้

สารบัญ (ต่อ)

หน้า

2.5.3	ข้อเสียของการมอดูเลตแบบสัญญาณเอฟเอ็ม	14
2.6	ภาษาคอมพิวเตอร์ในการเขียนสั่งงานไมโครคอนโทรลเลอร์	15
2.6.1	ระดับของภาษาคอมพิวเตอร์.....	16
2.6.1.1	ภาษาเครื่อง (Machine Language).....	16
2.6.1.2	ภาษาระดับต่ำ (Low Level Language).....	16
2.6.1.3	ภาษาระดับสูง (High Level Language).....	17
2.6.2	โครงสร้างของภาษา C51.....	20
2.6.2.1	พรีโพรเซสเซอร์ ไดเรกทีฟ (Preprocessor directives).....	20
2.6.2.2	การประกาศ (Declarations).....	20
2.6.2.3	การกำหนดค่า (Definition).....	20
2.6.2.4	นิพจน์ (Expressions).....	20
2.6.2.5	สเตตเมนต์ (Statements).....	21
2.6.2.6	ฟังก์ชัน (Function).....	21
2.6.2.7	ฟังก์ชัน main () (Main Function).....	21
2.6.3	การประกาศตัวแปรรีจิสเตอร์หน้าที่พิเศษ (SFR).....	21
2.6.3.1	การใช้งานรีจิสเตอร์อินเทอร์รัปต์จากสัญญาณภายนอก.....	21
2.6.3.1.1	รีจิสเตอร์และบิตที่เกี่ยวข้องกับการอินเทอร์รัปต์สัญญาณ ภายนอก.....	22
2.6.3.2	การใช้งานรีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ตอนุกรม	22
2.7	การใช้โปรแกรม KeilVision 3.....	23
2.7.1	ขั้นตอนที่ 1 การสร้างโปรเจค.....	24
2.7.2	ขั้นตอนที่ 2 เขียนโค้ดโปรแกรม	26
2.7.3	ขั้นตอนที่ 3 การเลือกตัวเลือกเครื่องมือสำหรับซีพียู.....	27
2.7.4	ขั้นตอนที่ 4 สร้างโปรเจคเพื่อให้ได้ไฟล์เฮกซ์	29
2.8	Appserv	30
2.8.1	ขั้นตอนการติดตั้งโปรแกรม Appserv	31
2.8.2	การถอนการติดตั้งระบบ	35

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์การใช้งานเพื่อการศึกษาค้นคว้าเท่านั้น ไม่อนุญาตให้พิมพ์หรือจำหน่าย การค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

	หน้า
บทที่ 3 การออกแบบและการสร้าง	36
3.1 แนวคิดในการออกแบบ	36
3.2 การออกแบบโครงสร้างของฮาร์ดแวร์	37
3.2.1 การออกแบบวงจรส่งสัญญาณอินฟราเรด	37
3.2.2 การออกแบบแผ่นลายวงจร	37
3.3 การออกแบบการทำงานของไมโครคอนโทรลเลอร์	39
3.3.1 ศึกษาการส่งสัญญาณอินฟราเรด	39
3.3.2 การเขียนโปรแกรมตรวจจับสัญญาณอินฟราเรด	40
3.3.3 การสร้างสัญญาณอินฟราเรดโดยใช้หลักการพัลส์วidthมอดูเลชั่น	42
3.3.4 การออกแบบการทำงานของไมโครคอนโทรลเลอร์	42
3.3.4.1 การทำงานของไมโครคอนโทรลเลอร์มาสเตอร์	43
3.3.4.2 การทำงานของไมโครคอนโทรลเลอร์สลาฟ	45
3.4 การออกแบบระบบการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต	46
3.4.1 การสร้างเว็บเพจด้วยโปรแกรม Dreamweaver	46
3.4.2 สร้างไฟล์คำสั่งภาษาพีเอชพี	47
3.4.3 การติดต่อกับเซิร์ฟเวอร์	48
บทที่ 4 ผลการทดสอบ	49
4.1 วิธีการสั่งงานเครื่องปรับอากาศ	49
4.2 ข้อจำกัดในการใช้งาน	50
4.2.1 ระยะเวลาของเครื่องปรับอากาศและอุปกรณ์สลาฟ	50
4.2.2 ในการสั่งงานเครื่องปรับอากาศยี่ห้ออื่น	50
บทที่ 5 สรุปผลการทดลองและข้อเสนอแนะ	51
5.1 สรุปผลการทดลอง	51
5.2 ปัญหาที่พบ	51
5.2 ข้อเสนอแนะ	51
บรรณานุกรม	53

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ไม่อนุญาตให้นำไปใช้หรือเผยแพร่โดยไม่ได้รับอนุญาต

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

สารบัญ (ต่อ)

หน้า

ภาคผนวก..... 54



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตารางที่

หน้า

2.1 แสดงความสัมพันธ์ของคำสั่งในภาษาระดับต่ำและภาษาเครื่อง..... 16



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

รูปที่	หน้า
2.1 โครงสร้างการทำงาน	4
2.2 พัลส์ที่รีโมทคอนโทรลส่งออกมา	5
2.3 แสดงบล็อกไดอะแกรมรีโมทคอนโทรล	5
2.4 ตัวอย่างการอ่านค่าโค้ดของรีโมทด้วยออสซิลโลสโคป	6
2.5 ตัวอย่างรูปแบบแพ็คเกจของ Enhanced ShockBurst™	8
2.6 รูปแบบ Packet Control Field	9
2.7 ลักษณะตัวอย่างในการรับส่งข้อมูลอย่างง่าย	9
2.8 ลักษณะตัวอย่างในการรับส่งข้อมูลในวงเครือข่ายย่อย	10
2.9 โครงสร้างการติดต่อสื่อสารกันภายในระบบเครือข่าย	10
2.10 การเชื่อมต่อการสื่อสารข้อมูลอนุกรม	12
2.11 การมอดูเลชันเชิงมุม	13
2.12 สัญญาณที่มอดูเลตด้วยความถี่	14
2.13 สัญญาณพัลส์ที่มีความกว้างคงที่	15
2.14 สัญญาณพัลส์ที่เปลี่ยนแปลงตลอดเวลา	15
2.15 หน้าต่างโปรแกรม KeilUvision 3	24
2.16 หน้าต่างการสร้างโปรเจคใหม่	24
2.17 หน้าต่างการสร้างไฟล์เดอริใหม่	25
2.18 กรอกชื่อโปรเจคและทำการบันทึก	25
2.19 เลือกซีพียูที่ต้องการ	25
2.20 หน้าต่าง Text 1*	26
2.21 หน้าต่างการบันทึก	26
2.22 ตัวอย่างการเพิ่มไฟล์เข้าไปในโปรเจค	27
2.23 หน้าต่างการเพิ่มไฟล์ไปยังกลุ่ม 'Source Group1'	27
2.24 ขั้นตอนการเปิดหน้าต่างตัวเลือกสำหรับทาร์เก็ต 'Target1'	28
2.25 หน้าต่างตัวเลือกสำหรับทาร์เก็ต 'Target 1'	28
2.26 การกำหนดการสร้างไฟล์เฮ็กซ์	29
2.27 การสร้างทาร์เก็ต	29
2.28 รูปตัวอย่างการแสดงข้อผิดพลาด	30
2.29 แสดงการติดตั้งโปรแกรม Appserv	31

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้เผยแพร่ไปยังเว็บไซต์อื่น การค้า
ไม่ว่ากรณีใดๆ กรุณาแจ้งให้ทราบก่อนการเผยแพร่ และต้องอ้างอิงถึงชื่อของเอกสารทุกครั้งที่มาเป็นใช้

สารบัญรูป (ต่อ)

รูปที่	หน้า
2.30 แสดงการยอมรับในการติดตั้งตัวโปรแกรม	31
2.31 เลือกโพลเดอร์ที่ต้องการในการติดตั้งตัวโปรแกรม Appserv	32
2.32 โปรแกรมทั้งหมดสำหรับการติดตั้ง	32
2.33 กำหนด Server Name , Email Address และ Port	33
2.34 กำหนด User , Password สำหรับการติดต่อตัวโปรแกรม.....	34
2.35 กระบวนการติดตั้งโปรแกรม Appserv	34
2.36 ติดตั้งโปรแกรมเสร็จสมบูรณ์.....	34
2.37 รูปแบบของตัวโปรแกรม ที่ติดตั้งเสร็จสมบูรณ์.....	35
3.1 ลักษณะการรับส่ง คำสั่งการทำงาน	36
3.2 วงจรส่งสัญญาณอินฟราเรด	37
3.3 ออกแบบแผ่นลายวงจรโดยใช้โปรแกรม Altium Designer Summer 09.....	38
3.4 อุปกรณ์ส่งสัญญาณอินฟราเรดควบคุมเครื่องใช้ไฟฟ้า	38
3.5 การเชื่อมต่ออุปกรณ์.....	39
3.6 ลักษณะของสัญญาณที่ส่งมาจากรีโมท	40
3.7 แสดงตำแหน่งขาของไมโครคอนโทรลเลอร์	40
3.8 หลักการทำงานของไมโครคอนโทรลเลอร์ในการตรวจจับสัญญาณอินฟราเรด	41
3.9 การแสดงผลรูปแบบของสัญญาณ.....	41
3.10 ลักษณะของสัญญาณที่ส่งมาจากอุปกรณ์ที่สร้างขึ้น.....	42
3.11 ไมโครคอนโทรลเลอร์มาสเตอร์	43
3.12 ขั้นตอนการทำงานของไมโครคอนโทรลเลอร์มาสเตอร์	44
3.13 ขั้นตอนการทำงานของไมโครคอนโทรลเลอร์สลาฟ	45
3.14 ไมโครคอนโทรลเลอร์สลาฟ	46
3.15 หน้าต่างการสร้างหน้าเว็บใหม่	46
3.16 หน้าต่างการออกแบบเว็บเพจสำหรับล็อกอินเข้าสู่ระบบ	47
3.17 หน้าต่างการออกแบบเว็บเพจสำหรับการส่งงานไมโครคอนโทรลเลอร์	47
3.18 หน้าต่างการเขียนโค้ดส่งงานไมโครคอนโทรลเลอร์	48
3.19 หน้าต่างโปรแกรม FileZilla	48
4.1 หน้าเว็บเพจการใส่รหัสก่อนการใช้งาน.....	49
4.2 หน้าเว็บเพจการส่งงาน	49

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้เผยแพร่ไปใช้ประโยชน์ทางการค้า
ไม่ว่ากรณีใดๆ กรุณาอย่าทำซ้ำโดยไม่ได้รับอนุญาต และต้องอ้างอิงถึงชื่อของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญญรูป (ต่อ)

รูปที่

หน้า

4.3 ลักษณะการส่งสัญญาณอินฟราเรดควบคุมเครื่องปรับอากาศ..... 50



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1

บทนำ

1.1 ความสำคัญของปัญญานิพนธ์

บ้านพักอาศัยในปัจจุบันเป็นบ้านที่ถูกออกแบบโครงสร้างการวางรูปแบบระบบไฟฟ้าและเครื่องใช้ไฟฟ้าไว้เป็นแบบแผนที่ตายตัว ซึ่งยากต่อการปรับเปลี่ยนรูปแบบการติดตั้ง ทำให้เกิดความไม่คล่องตัวในการเคลื่อนย้ายและการบริหารจัดการ นอกจากนี้การควบคุมเครื่องใช้ไฟฟ้าภายในบ้านแต่ละชนิดต่างมีข้อจำกัดที่แตกต่างกันตามลักษณะการใช้งาน ในบ้านหนึ่งหลังอาจมีห้องหลายห้อง แต่ละห้องต่างมีเครื่องใช้ไฟฟ้าหลายชนิด ทำให้การควบคุมการทำงานของเครื่องใช้ไฟฟ้าแต่ละชนิดนั้นมีความยุ่งยากและไม่ตอบสนองต่อยุคสมัยของสังคมที่มีความเร่งรีบ

เทคโนโลยี E-House เป็นเทคโนโลยีการควบคุมเครื่องใช้ไฟฟ้าด้วยระบบไร้สาย (Wireless) โดยใช้หลักการแทรกวงจรตัดต่อรีเลย์ไว้กับเครื่องใช้ไฟฟ้าหรือการตัดกำลังไฟฟ้าที่จ่ายให้กับเครื่องใช้ไฟฟ้าแต่ละตัวด้วยระบบไร้สายผ่านการควบคุมจากเครือข่ายอินเทอร์เน็ต ซึ่งเป็นการควบคุมที่สั่งงานให้เครื่องใช้ไฟฟ้าปิด-เปิดได้เท่านั้น แต่ยังมีเครื่องใช้ไฟฟ้าอีกหลายชนิดที่ไม่สามารถแทรกวงจรการตัดต่อกำลังไฟฟ้าได้และยังต้องการการควบคุมที่มากกว่าการปิด-เปิด เช่น เครื่องปรับอากาศ โทรทัศน์ เครื่องเล่นวีดีโอ เป็นต้น โดยเครื่องใช้ไฟฟ้าเหล่านี้สามารถควบคุมได้ด้วยรีโมทคอนโทรล ซึ่งใช้การรับส่งคำสั่งด้วยสัญญาณอินฟราเรด แต่เครื่องใช้ไฟฟ้าแต่ละชนิดมีรีโมทคอนโทรลเฉพาะเครื่องนั้น ๆ ไม่สามารถใช้ร่วมกับเครื่องใช้ไฟฟ้าชนิดอื่นได้ ปัญญานิพนธ์นี้จึงได้นำเทคโนโลยี E-House มาประยุกต์ใช้ในการพัฒนาอุปกรณ์ที่สามารถควบคุมเครื่องใช้ไฟฟ้าได้โดยใช้เทคนิคของรีโมทคอนโทรลส่งสัญญาณอินฟราเรดไปควบคุม ซึ่งสามารถสั่งงานการควบคุมผ่านเครือข่ายอินเทอร์เน็ตได้และใช้ระบบไร้สายในการสื่อสารข้อรับส่งคำสั่งจากเครือข่ายอินเทอร์เน็ตมายังอุปกรณ์ที่ใช้ในการควบคุม โดยไม่ต้องแทรกวงจรหรือปรับเปลี่ยนวงจรการทำงานของเครื่องใช้ไฟฟ้าที่มีอยู่เดิมและอุปกรณ์นี้ยังสามารถควบคุมเครื่องใช้ไฟฟ้าหลายชนิดได้ด้วยอุปกรณ์เพียงตัวเดียว โดยใช้หลักการปรับเปลี่ยนซอฟต์แวร์ที่ใช้ในการควบคุม ในปัญญานิพนธ์นี้ได้ทำการทดสอบการควบคุมกับเครื่องปรับอากาศยี่ห้อไดกิ้น ซึ่งจะเป็แนวทางในการนำไปประยุกต์ใช้กับเครื่องใช้ไฟฟ้าชนิดอื่น ๆ ต่อไป

1.2 วัตถุประสงค์ของปัญญานิพนธ์

เพื่อออกแบบการควบคุมเครื่องใช้ไฟฟ้าที่เพิ่มความสะดวกสบายให้แก่ผู้ใช้งานโดยอาศัยการทำงานรับส่งคำสั่งด้วยสัญญาณอินฟราเรดที่ควบคุมด้วยไมโครคอนโทรลเลอร์ ซึ่งผู้ใช้งานสามารถสั่งงานได้ผ่านเครือข่ายอินเทอร์เน็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับครู ใช้งานเพื่อการศึกษาดูงาน ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.3 ขอบเขตของปริญญาโท

1. ออกแบบโครงสร้างและการทำงานของอุปกรณ์ควบคุมเครื่องปรับอากาศ
2. การประยุกต์การทำงานของไมโครคอนโทรลเลอร์ในการควบคุมอุปกรณ์
3. การประยุกต์ระบบการสั่งงานผ่านเครือข่ายอินเทอร์เน็ต

1.4 ขั้นตอนการศึกษา

1. ศึกษาการประยุกต์การทำงานและการรับส่งสัญญาณของรีโมทคอนโทรล
2. ศึกษาการประยุกต์การสร้างสัญญาณอินฟราเรดและการรับส่งเพื่อการควบคุมเครื่องใช้ไฟฟ้า
3. ศึกษาโครงสร้างการทำงานของไมโครคอนโทรลเลอร์ที่เกี่ยวข้อง
4. ศึกษาการประยุกต์การส่งสัญญาณในระบบของเครือข่ายไร้สาย
5. ศึกษาการประยุกต์การสั่งงานจากระบบเครือข่ายอินเทอร์เน็ต

1.5 ประโยชน์ที่คาดว่าจะได้รับของโครงการวิจัย

1. เพื่อสร้างความสะดวกสบายในการควบคุมเครื่องใช้ไฟฟ้าภายในบ้าน
2. ทำให้ตระหนักถึงความสามารถในการพัฒนาของเทคโนโลยีต่าง ๆ ที่มีการพัฒนาไปอย่างรวดเร็ว
3. เกิดความรู้ความเข้าใจในการทำงานของเครื่องใช้ไฟฟ้าที่ใช้ในชีวิตประจำวันมากขึ้น
4. สามารถนำองค์ความรู้ที่ได้จากการศึกษาวิจัยนี้ มาเป็นแนวทางในการนำไปสู่การสร้างมาตรฐาน เพื่อรองรับเทคโนโลยีใหม่ ๆ ในอนาคต
5. สามารถนำอุปกรณ์ที่ได้ทดลองจากงานวิจัยนี้มาใช้ประโยชน์ได้จริงในชีวิตประจำวัน

1.6 รายละเอียดของปริญญาโท

ปริญญาโทฉบับนี้มีเนื้อหาทั้งหมด 5 บท โดยแต่ละบทมีรายละเอียดดังต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญ วัตถุประสงค์ และขอบเขตของปริญญาโท รวมไปถึงประโยชน์ที่คาดว่าจะได้รับ และรายละเอียดของปริญญาโท

บทที่ 2 กล่าวถึงทฤษฎีและหลักการที่ใช้ในปริญญาโท ซึ่งได้แก่ หลักการของรีโมทคอนโทรล อินฟราเรดแอลอีดี สัญญาณอินฟราเรด พัลส์วิดมอดูเลชัน ภาษาคอมไพเลอร์ โครงสร้างของภาษาซีและไมโครคอนโทรลเลอร์ตลอดจนหลักการที่เกี่ยวข้องในการแสดงผลผ่านเว็บเบราว์เซอร์

บทที่ 3 กล่าวถึงการออกแบบและการสร้างระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต ซึ่งประกอบไปด้วย การออกแบบโครงสร้างของฮาร์ดแวร์ การ

ออกแบบการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ และการออกแบบการประยุกต์
ระบบการสื่อสารผ่านอินเทอร์เน็ต

บทที่ 4 กล่าวถึงผลการทดสอบและวิธีการใช้งานระบบควบคุมเครื่องปรับอากาศผ่าน
อินเทอร์เน็ต

บทที่ 5 กล่าวถึงการสรุปผลการทดสอบการทำงาน และข้อเสนอแนะเกี่ยวกับปริญญาโท



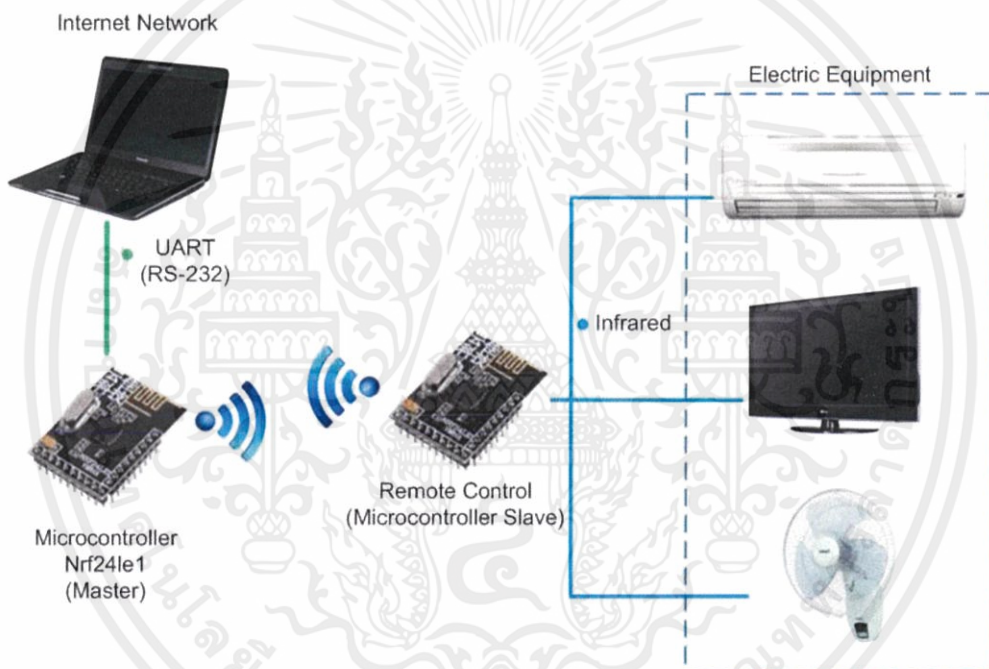
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

ทฤษฎีและหลักการ

2.1 กล่าวนำ

ในบทนี้จะกล่าวถึงทฤษฎีและหลักการของระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต (Air Conditioning Control System via Internet) โดยมีโครงสร้างการทำงานดังรูปที่ 2.1 ซึ่งมีส่วนประกอบหลัก คือ หลักการทำงานของรีโมตคอนโทรล การประยุกต์ใช้การสื่อสารด้วยระบบไร้สายจากเทคโนโลยี E-House ด้วยไมโครคอนโทรลเลอร์ nrf24le1 และการประยุกต์ใช้งานการสั่งงานจากเครือข่ายอินเทอร์เน็ต

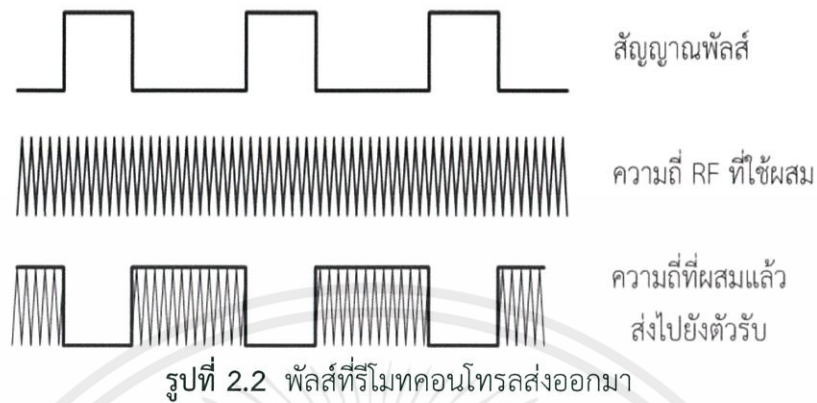


รูปที่ 2.1 โครงสร้างการทำงาน

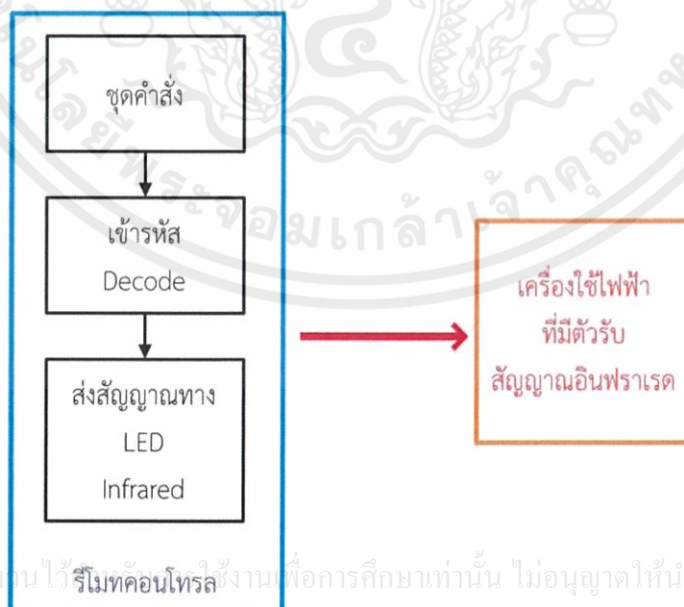
2.2 หลักการทำงานของรีโมตคอนโทรล

รีโมตคอนโทรล คือเครื่องมืออิเล็กทรอนิกส์ชนิดหนึ่งที่มีการส่งสัญญาณแบบดิจิทัล ใช้สำหรับควบคุมการดำเนินการของสิ่งประดิษฐ์หรือเครื่องจักรต่างๆ โดยเฉพาะเครื่องใช้ไฟฟ้าภายในบ้าน เช่น เครื่องปรับอากาศ โทรทัศน์ เครื่องเสียงจากระยะไกลด้วยแสงอินฟราเรดหรือรังสีใต้แดงซึ่งเป็นคลื่นแม่เหล็กไฟฟ้าที่มีความยาวคลื่น 700 นาโนเมตร - 1 มิลลิเมตร ถึงในช่วง 10^{11} - 10^{14} เฮิร์ตซ์ มีทิศทางในระดับสายตาไม่สามารถทะลุผ่านวัตถุทึบแสงได้ และเนื่องจากการเดินทางไกลของพัลส์ในรูปแบบดิจิทัล มีปัญหาในการทะลุวงอากาศได้ไม่ไกล เกิดการรบกวนได้ง่าย จึงมีการผสมความถี่วิทยุ

(Radio Frequency) เพิ่มเข้าไปเพื่อลดปัญหาที่เกิดขึ้น ความถี่วิทยุที่ผสมเข้าไปมีความถี่ในช่วง 30-40 กิโลเฮิร์ตซ์ โดยมีรูปแบบของพัลส์ที่ส่งออกมาดังรูปที่ 2.2



สัญญาณจากรีโมทเป็นสัญญาณแบบดิจิทัล ที่มีระดับสัญญาณ 0 และ 1 ปะปนกัน ตามโค้ดของชุดปุ่มคำสั่ง เช่น ปุ่มปิดกับปุ่มเปิด จะมีรูปแบบของระดับสัญญาณ 0 และ 1 ที่แตกต่างกัน หลักการที่ใช้ในปรีญญาณิพจน์นี้ คือ เมื่อมีการกดปุ่มคำสั่ง คำสั่งนั้นจะถูกทำการเข้ารหัสดีโค้ดด้วยไมโครคอนโทรลเลอร์และแปลงเป็นสัญญาณไฟฟ้าผ่าน อินฟราเรดแอลอีดี ซึ่งเป็นตัวส่งสัญญาณอินฟราเรดทำหน้าที่แปลงสัญญาณไฟฟ้าเป็นสัญญาณอินฟราเรดที่สว่างมืด ปะปนกัน ส่งไปยังเครื่องรับสัญญาณอินฟราเรดในเครื่องใช้ไฟฟ้าที่ต้องการควบคุม วงจรในเครื่องรับจะถอดโค้ดว่าเป็นปุ่มใด เช่น ปุ่มปิดหรือปุ่มเปิด ดังรูปที่ 2.3 แสดงบล็อกไดอะแกรมรีโมทคอนโทรล



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้งานเพื่อการศึกษเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะผิดใดๆทั้งสิ้น อีกทั้งห้ามมิให้นำไปเผยแพร่ในที่สาธารณะทุกครั้งที่มีการนำไปใช้

รูปที่ 2.3 แสดงบล็อกไดอะแกรมรีโมทคอนโทรล

2.2.1 สัญญาณที่รีโมตคอนโทรลส่งออกมาประกอบด้วย 5 ส่วนหลัก ๆ ดังนี้

1. ดาต้าโค้ด (Data Code) เป็นข้อมูลหลักที่จะส่งออกไปควบคุมวงจรในส่วนของภาครับ โดยชุดปุ่มคำสั่งจะเป็นตัวส่งงานเข้าสู่ระบบการเข้ารหัสข้อมูลกำหนดความเป็นไปของแต่ละฟังก์ชัน

2. อินเวิร์ตดาต้าโค้ด (Inverted Data Code) เป็นการกลับลอจิกของข้อมูลหลักเพื่อรักษาฐานเวลาให้คงที่ทุกข้อมูลซึ่งเป็นข้อมูลที่มีจำนวนบิตเท่ากับดาต้าโค้ด

3. ดีไวส์โค้ด (Device Code) หรือบางครั้งใช้ศัพท์ “คัสโตเมอร์โค้ด” เนื่องจากปัจจุบันเครื่องใช้ต่างๆล้วนเป็นระบบรีโมตคอนโทรลแบบอินฟราเรด การสั่งงานจากรีโมทอาจมีการควบคุมอุปกรณ์ไฟฟ้าอย่างอื่นได้ เช่น เครื่องรับโทรทัศน์ต่อพ่วงอยู่กับเครื่องเล่นวีดีโอ จูนเนอร์พ่วงอยู่กับเครื่องขยายเสียง เลเซอร์ดิสก์พ่วงอยู่กับเครื่องรับโทรทัศน์หรืออื่นๆ กรณีเช่นนี้หากสั่งเครื่องหนึ่งเครื่องใด เครื่องที่ต่อร่วมอยู่ด้วยสามารถรับเอาข้อมูลฟังก์ชันการทำงานเข้าไปได้ด้วย จึงมีการสร้างรหัสข้อมูลของเครื่องเล่นแต่ละอย่างให้แตกต่างกันออกไป เช่น เครื่องรับโทรทัศน์มีดีไวส์โค้ด ซึ่งเป็นโค้ดที่ใช้แยกประเภทของเครื่องใช้ด้วยข้อมูล 00000 (5 บิต) ในขณะที่เครื่องเล่นวีดีโอใช้ดีไวส์โค้ด 11111 (5 บิต)

4. อินเวิร์ต ดีไวส์โค้ด (Inverted Device Code) เป็นการกลับข้อมูลดีไวส์โค้ดเพื่อรักษา เช่นเดียวกันกับข้อมูลหลักข้อมูลดังกล่าวต้องมี 5 บิตเหมือนดีไวส์โค้ด

5. เฮดพัลส์ (Head Pulse) การกลับข้อมูลหรือการอินเวิร์ตเป็นเพียงการรักษาเวลาของข้อมูล แต่การใส่เฮดพัลส์เป็นวิธีที่สามารถตรวจเช็คข้อมูลเพิ่มความแน่นอนของข้อมูลขึ้นอีกเพราะในบางครั้งการกดคีย์แช่ไว้นานๆ ความต่อเนื่องของข้อมูลจะมีตลอด ทำให้แยกไม่ว่าคือข้อมูลหลักหรือข้อมูลรอง จึงต้องมีเฮดพัลส์เป็นสัญญาณนำร่องก่อนจะมีข้อมูลต่างๆส่งออกมา และในขณะที่มีการส่งข้อมูลอย่างต่อเนื่องจะมีเฮดพัลส์ส่งออกมาเป็นช่วงๆ ให้เครื่องรับสามารถแยกกลุ่มข้อมูลออกมาได้



รูปที่ 2.4 ตัวอย่างการอ่านค่าโค้ดของรีโมทด้วยออสซิลโลสโคป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับครู ใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น ถือว่าท่านมีข้อตกลงเนื้อหา และต้องอ้างอิงถึงแหล่งเอกสารทุกครั้งที่มีวางไปใช้
สำหรับค่าที่ต้องการใช้งานจริง คือ ค่าข้อมูลขนาด 8 บิต หรือ 1 ไบท์ และมีสัญญาณอินเวิร์ตของค่าข้อมูลเพื่อตรวจสอบข้อมูลว่าถูกต้องหรือไม่ และสุดท้ายคือ บิตสตอป ซึ่งเป็น การบอกให้ซีพียูรู้ว่ามันสิ้นสุดการส่งค่าแล้ว

หลักการของรีโมทคอนโทรลนั้นใช้ในการติดต่อสื่อสารระหว่างตัวอุปกรณ์ที่สร้างขึ้นเพื่อควบคุมเครื่องใช้ไฟฟ้ากับเครื่องใช้ไฟฟ้าที่ต้องการควบคุม โดยภายในตัวอุปกรณ์นั้นจะมีไมโครคอนโทรลเลอร์ซึ่งเป็นตัวสถาป คอยรับส่งข้อมูลคำสั่งกับไมโครคอนโทรลเลอร์ที่เป็นตัวมาสเตอร์ด้วยระบบไร้สาย โดยประยุกต์ใช้หลักการการสื่อสารจากเทคโนโลยี E-House

2.3 เทคโนโลยี E-House

2.3.1 โมดูล nRF24LE1

nRF24LE1 เป็นโมดูลที่มีความสามารถในการติดต่อสื่อสารแบบไร้สายด้วยความถี่ 2.4 GHz โดยมีการกระจายส่งสัญญาณให้กับอุปกรณ์อื่น ๆ ภายในบริเวณรอบ ๆ nRF24LE1 มีฟังก์ชันการทำงานของไมโครคอนโทรลเลอร์อย่างครบถ้วน เช่น โมดูล Serial Peripheral Interface (SPI), โมดูล 2-wire Universal Asynchronous Receiver Transmitter (UART), โมดูล Analog to Digital Converter(ADC), โมดูลPulse Width Modulation(PWM) และอื่น ๆ และมีความโดดเด่นในเรื่องของข้อมูลที่สามารถตรวจสอบการสูญหายของข้อมูลได้

การนำไปประยุกต์ใช้งาน

ไมโครคอนโทรลเลอร์ nRF24LE1 สามารถนำไปประยุกต์ใช้ในงานต่าง ๆ ได้อย่างหลากหลาย โดยจำแนกได้ดังนี้

- อุปกรณ์ต่อพ่วงเกี่ยวกับคอมพิวเตอร์ เช่น เม้าส์ คีย์บอร์ด หรือรีโมทคอนโทรล
- อุปกรณ์ในการควบคุมระยะในไกล เช่น เครื่องใช้ภายในบ้าน เป็นต้น
- อุปกรณ์ในการติดตามและตรวจสอบ เช่น อุปกรณ์เครือข่ายเซนเซอร์ อุปกรณ์ที่ใช้งานเกี่ยวข้องกับ RFID เป็นต้น
- อุปกรณ์ในระบบรักษาความปลอดภัย เช่น อุปกรณ์แจ้งเตือน เป็นต้น

2.3.2 Enhanced ShockBurst™

โดยในส่วนของการรับส่งข้อมูลนั้นได้มีการเลือกใช้ไมโครคอนโทรลเลอร์ที่มีโมดูลไร้สาย ในการรับส่งข้อมูล โดยได้มีการใช้โปรโตคอลที่มีชื่อว่า Enhanced ShockBurst[4] โดยในส่วน of โมดูลตัวนี้ได้มีการรับส่งข้อมูลที่ใช้พลังงานต่ำเป็นพิเศษแต่มีการสื่อสารที่มีประสิทธิภาพสูง นอกจากนี้ยังมีการตรวจสอบเช็คการสูญหายข้อมูลด้วย ซึ่งในส่วนนี้จะทำให้การรับ-ส่งข้อมูลนั้นมีประสิทธิภาพเพิ่มขึ้น

คุณสมบัติหลัก

- ขนาดข้อมูลมีขนาดตั้งแต่ 1 - 32 ไบต์ซึ่งสามารถเลือกกำหนดตามความต้องการได้
- สามารถตั้งค่าการจัดการในส่วน of แพ็คเกจข้อมูลได้โดยอัตโนมัติ
- ตอบสนองการรับ-ส่งข้อมูลแบบอัตโนมัติ และจัดส่งข้อมูลใหม่อีกครั้งโดยอัตโนมัติ

เมื่อมีการสูญหายของข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ติดต่อกันในลักษณะโทโพลยีแบบสตาร์ (Star Topology) แบบ1 : 6

หลักการจัดการรับ-ส่งข้อมูล

- การส่งแพ็คเกจข้อมูลจาก PTX เมื่อ PRX ได้รับข้อมูล จะกำหนดPTXให้อยู่ในโหมดรับเพื่อรอให้แพ็คเกจ ACK ส่งกลับ
- เมื่อได้รับแพ็คเกจข้อมูลแล้ว PRX จะประกอบแพ็คเกจและส่ง แพ็คเกจที่รับมา (ACK แพ็คเกจ)กลับไปที PTX ที่อยู่ในโหมดรับ
- ถ้า PTX ไม่ได้รับแพ็คเกจ ACK ในทันทีหรือภายในระยะเวลาที่กำหนด จะมีการจัดการส่งข้อมูลชุดเดิมส่งออกไปใหม่ เพื่อเป็นการตรวจสอบการสูญหายของข้อมูล
- รูปแบบแพ็คเกจของ Enhanced ShockBurst™ (ตามรูปที่ 2.5) ประกอบไปด้วย

Preamble 1 byte	Address 3-5 byte	Packet Control Field 9 bit	Payload 0 - 32 byte	CRC 1-2 byte
-----------------	------------------	----------------------------	---------------------	--------------

รูปที่ 2.5 ตัวอย่างรูปแบบแพ็คเกจของ Enhanced ShockBurst™

จากรูปที่ 2.5 เป็นตัวอย่างรูปแบบแพ็คเกจของ Enhanced ShockBurst™จะประกอบได้ด้วยส่วนต่าง ๆ ดังนี้

- Preamble มีขนาด 1 ไบต์ จะมีการเปลี่ยนแปลงอยู่ตลอดเวลา โดยขึ้นอยู่กับบิตแรก ถ้าเป็น 1 จะมีการตั้งค่า เริ่มต้นอัตรานอ้ติเป็น 10101010 แต่ถ้าบิตแรกเป็น 0 จะมีการตั้งค่าเริ่มต้นอัตรานอ้ติเป็น 01010101

- Address เปรียบเสมือนขอบเขตที่อยู่ของสัญญาณ เพื่อให้สามารถรับทราบได้ว่า จะมีการตรวจพบข้อมูลที่ถูกส่งออกมาจะมีการกำหนด Address ให้สามารถตอบสนองกันได้ โดยสามารถกำหนดได้เป็น 3 ,4 และ 5 ไบต์ ซึ่งถ้าตัวอุปกรณ์มี Address ที่ไม่เหมือนกัน ก็ไม่สามารถที่จะรับส่งข้อมูลกันได้

- Packet Control Field ดังรูปที่ 2.6 จะแบ่งออกเป็น 3 ส่วน ดังนี้

1. Payload length มีขนาด 6 บิต ทำหน้าที่กำหนดขนาดของข้อมูลได้ 0-32 ไบต์
2. PID (Packet Identification) มีขนาด 2บิต ทำหน้าที่เป็นรีจิสเตอร์ที่กำกับในโหมด PTX และ PRX ในการรับส่งข้อมูล และตรวจสอบข้อมูลสูญหาย
3. No Acknowledgment flag (NO_ACK) มีขนาด 1 บิต ทำหน้าที่เป็นรีจิสเตอร์ที่จะบอกถึงการตอบสนองของข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.6 รูปแบบ Packet Control Field

Payload เป็นส่วนของข้อมูลที่ต้องการจะสื่อสาร ซึ่งจะมีขนาดตามที่กำหนด โดยขนาดสูงสุดอยู่ที่ 32 ไบต์

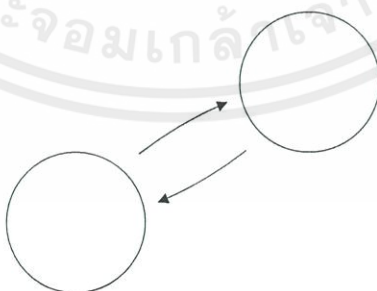
- CRC (Cyclic Redundancy Check) มีขนาด 1 – 2 ไบต์ จะทำหน้าที่ตรวจสอบข้อมูลที่รับเข้ามาว่าตรงกับที่ต้องการหรือไม่ โดยขึ้นอยู่กับความละเอียดที่ต้องการตรวจสอบ

2.3.3 โทโปโลยีในการสื่อสารของระบบเครือข่าย

รูปแบบในการสื่อสารจะเป็นลักษณะรูปแบบอุปกรณ์มาสเตอร์ (Master) และอุปกรณ์สลาฟ (Slave) โดยจะมีตัวอุปกรณ์สลาฟเพียงตัวเดียวเท่านั้น (point-to-point) ที่เป็นตัวอุปกรณ์ที่จะสามารถติดต่อสื่อสารกับตัวอุปกรณ์มาสเตอร์ โดยเลือกจากตัวอุปกรณ์สลาฟที่ใกล้ที่สุดเป็นตัวติดต่อสื่อสาร โดยลักษณะของข้อมูลที่ใช้ในการติดต่อสื่อสารจะมีการเข้ารหัส ตรวจสอบตำแหน่งที่อยู่ของอุปกรณ์สลาฟ เพื่อให้สามารถระบุตัวอุปกรณ์สลาฟได้ถูกต้อง

ส่วนต่อไปเป็นการกล่าวถึงการสื่อสารของอุปกรณ์สลาฟภายในเครือข่ายย่อย โดยจะแบ่งเป็น 2 ลักษณะ คือ แบบวงแหวน (Token ring) และ แบบต้นไม้ (Tree)

โทโปโลยีในส่วนของารรับส่งข้อมูลย่อยภายในเครือข่าวนั้น ได้มีการวางหลักการการรับส่งข้อมูลในวงเครือข่าวย่อยในลักษณะบรอดแคสโดยได้มีการจำกัดขนาดของจำนวนอุปกรณ์ให้อยู่ในจำนวนที่กำหนด จึงได้มีการจำกัดทิศทางการรับส่งข้อมูลแบบไร้สายนี้ ให้เป็นไปตามที่กำหนด โดยได้มีการอ้างอิงการเชื่อมต่อในขั้นตอนแรกในลักษณะแบบหนึ่งต่อหนึ่ง ดังรูปที่ 2.7 ซึ่งทำให้ง่ายและสะดวกต่อการจัดการของข้อมูล



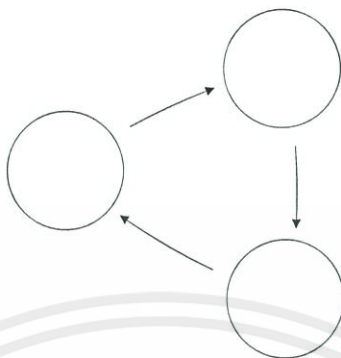
รูปที่ 2.7 ลักษณะตัวอย่างในการรับส่งข้อมูลอย่างง่าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไมออนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งยังมีลิขสิทธิ์ของเนื้อหาบางส่วนซึ่งจึงเป็นเอกสารของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

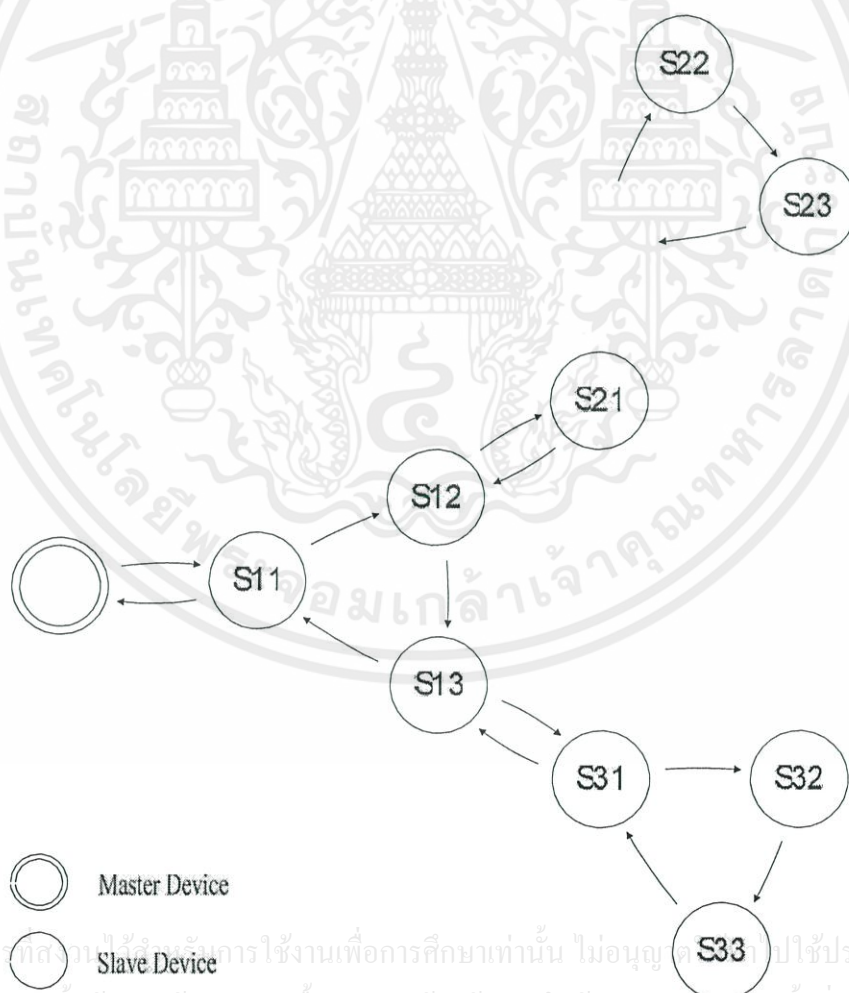
หลังจากนั้น จึงได้มีการวางหลักการคร่าว ๆ ในการรับส่งข้อมูลในวงเครือข่าวย่อยดังรูปที่ 2.8 เพื่อให้เกิดความต่อเนื่องและสะดวกในการนำไปใช้ จึงได้มีการจำกัดทิศทางการสื่อสาร

รับ-ส่งข้อมูลย่อยที่แตกต่างจากเดิม โดยได้กำหนดให้มีการอ้างอิงการเชื่อมต่อกันภายในวงย่อยของตนเองก่อน ก่อนที่จะรับ-ส่งไปยังวงย่อยในวงต่อ ๆ ไป โดยหลักการในลักษณะนี้ สามารถทำให้ขยายจำนวนเครือข่ายได้ง่ายและจำนวนมากขึ้น



รูปที่ 2.8 ลักษณะตัวอย่างในการรับส่งข้อมูลในวงเครือข่ายย่อย

2.3.4 โครงสร้างการติดต่อสื่อสารกันภายในระบบเครือข่ายย่อย



เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาติไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.9 โครงสร้างการติดต่อสื่อสารกันภายในระบบเครือข่าย

โดยการติดต่อสื่อสารสารกันภายในเครือข่ายย่อยนั้นได้มีการแบ่งวงในการสื่อสารกันเป็น 3 รูปแบบ คือ ในรูปแบบที่หนึ่งเครือข่ายย่อยจะติดต่อสื่อสารกันภายในวงของตัวเอง ในรูปแบบที่สอง จะเป็นการติดต่อสื่อสารกันโดยข้ามไปยังวงอื่น ๆ และในรูปแบบที่ 3 เป็นการติดต่อสื่อสารจากตัวมาสเตอร์กับวงสลาฟที่กำหนด โดยในรูปแบบที่ 1 และ 2 สามารถติดต่อสื่อสารกันได้สูงสุด 3 ตัว

จากรูปที่ 2.9 ตัวเลขในรูป เช่น s11 s12 จะเป็นตัวบ่งบอกถึงวงเครือข่ายย่อยในการติดต่อสื่อสาร โดยตัวเลขหลักที่ 1 จะเป็นตัวบอกถึงลำดับของวงเครือข่าย และลำดับที่ 2 หมายถึงลำดับอุปกรณ์ที่ติดต่อสื่อสาร ซึ่งมีได้สูงสุด 3 ตัว โดยโครงสร้างนั้นได้มีการวางไว้ให้เกิดความยืดหยุ่นมากขึ้น ซึ่งไม่จำเป็นต้องมีตัวอุปกรณ์สลาฟ ครบสามตามวงเครือข่ายย่อยก็สามารถที่จะติดต่อสื่อสารได้เช่นกัน แต่จะมีข้อจำกัดในเรื่องของความสามารถในการสื่อสารกันข้ามวงเครือข่ายย่อยได้ถ้าขาดอุปกรณ์สลาฟบางตัวไป

ไมโครคอนโทรลเลอร์มาสเตอร์นอกจากจะสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์สลาฟด้วยระบบไร้สายแล้ว ยังต้องเชื่อมต่อกับคอมพิวเตอร์เพื่อรองรับข้อมูลที่ส่งมาจากเครือข่ายอินเทอร์เน็ตอีกด้วย ไมโครคอนโทรลเลอร์มาสเตอร์จึงมีการเชื่อมต่อกับคอมพิวเตอร์ด้วยการสื่อสารข้อมูลอนุกรมหรือ UART

2.4 การสื่อสารข้อมูลอนุกรม (UART)

UART ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส สำหรับการสื่อสารข้อมูลอนุกรมบนคอมพิวเตอร์ หน้าที่หลักของ UART คือ ทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัสแล้วส่งออก และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าคอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังทำการแจ้งข้อมูลอื่นๆ ให้คอมพิวเตอร์ทราบด้วย เช่น อัตราความเร็วในการรับส่งข้อมูล (Baud rate), รูปแบบการส่งข้อมูล, ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล (ผิดพลาดจากพาริตี, เฟรมข้อมูล, โอเวอร์รัน) เป็นต้น

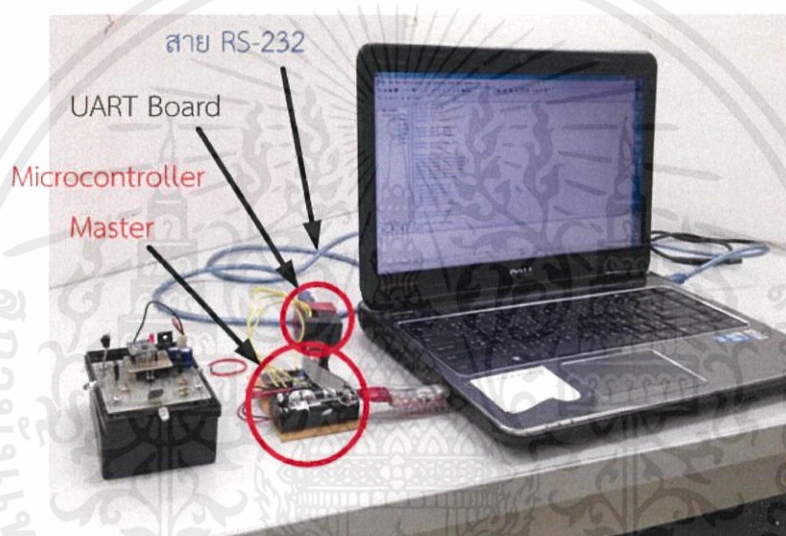
ภายใน UART จะมีส่วนของวงจรสร้างอัตราการถ่ายทอดข้อมูลแบบโปรแกรมได้ (Programmable Baudrate Generator) โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART โดยตัวหารนี้มีขนาด 16 บิตดังนั้นจึงกำหนดตัวหารให้อยู่ในช่วง 10 – 65,535

UART สามารถรับส่งข้อมูลได้ทั้งแบบฮาล์ฟดูเพล็กซ์ (Half Duplex) และฟูลดูเพล็กซ์ (Full Duplex) โดยการส่งแบบ ฮาล์ฟดูเพล็กซ์ เป็นการส่งแบบทิศทางเดียว และการส่งแบบฟูลดูเพล็กซ์นั้นสามารถรับและส่งข้อมูลได้ในคราวเดียวกัน โดยส่วนใหญ่ UART จะใช้มาตรฐานการสื่อสารแบบ RS-232 โดยถ้าติดตั้งบนคอมพิวเตอร์ทั่วไปเรียกว่า คอมพอร์ต (Comport)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.1 RS-232 (Recommended Standard 232)

RS-232 คือ มาตรฐานการเชื่อมต่อข้อมูลแบบอนุกรม ใช้เพื่อเพิ่มระยะทางในการส่งข้อมูลแบบอนุกรม ให้สามารถส่งได้ระยะทางที่มากขึ้น โดยมีการเปลี่ยนระดับแรงดัน ของลอจิก (Logic) จากเดิมที่จะอยู่ในช่วง 0-5 โวลต์ หรือ 0-3.3 โวลต์ เป็นช่วง -15 ถึง 15 โวลต์ ลักษณะสัญญาณอินพุตและเอาต์พุตของพอร์ต RS-232 สัญญาณเอาต์พุตที่ใช้ควบคุม (RTS และ DTR) และสัญญาณแสดงสถานะอินพุต (CTS, DSR & DCD) ของพอร์ตอนุกรม RS-232 จะถูกกลับสถานะภายในตัว UART ส่วนสัญญาณข้อมูลทั้งภาคส่งและภาครับจะไม่ถูกกลับสถานะUART จึงต้องส่งเข้าสู่วงจรขับเพื่อปรับระดับแรงดันให้ได้ระดับสัญญาณเป็นไปตามมาตรฐาน RS-232 ก่อนส่งออกไปจากคอมพิวเตอร์สำหรับอุปกรณ์ต่อเชื่อมปลายทางก็จะต้องมีวงจรขับในลักษณะนี้เช่นเดียวกัน เพื่อให้ได้ระดับสัญญาณในระดับเดียวกัน การเชื่อมต่ออุปกรณ์แต่ละชนิดแสดงดังรูปที่ 2.10



รูปที่ 2.10 การเชื่อมต่อการสื่อสารข้อมูลอนุกรม

2.5 พัลส์วิดมอดูเลชั่น (PWM)

เมื่อต้องการส่งสัญญาณหรือข้อมูลผ่านช่องทางการสื่อสารจำเป็นต้องอาศัยพลังงานไฟฟ้าในการพาสัญญาณเหล่านั้นให้เคลื่อนย้ายจากที่หนึ่งไปยังอีกที่หนึ่ง ขบวนการในการเพิ่มพลังงานไฟฟ้าดังกล่าวเรียกว่า “การมอดูเลต” อุปกรณ์สำหรับมอดูเลตสัญญาณ(Modulator)จะสร้างสัญญาณคลื่นพาหะและรวมเข้ากับสัญญาณข้อมูลเพื่อให้สัญญาณมีความแรงพอที่จะส่งผ่านสื่อกลางไปยังอีกจุดหนึ่งเมื่อถึงปลายทางจะมีอุปกรณ์ในการแยกสัญญาณคลื่นพาหะออกเรียกวิธีการแยกสัญญาณนี้ว่า “การดีมอดูเลต” (Demodulation)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5.1 ประเภทของการมอดูเลต แบ่งเป็น 3 ประเภท คือ

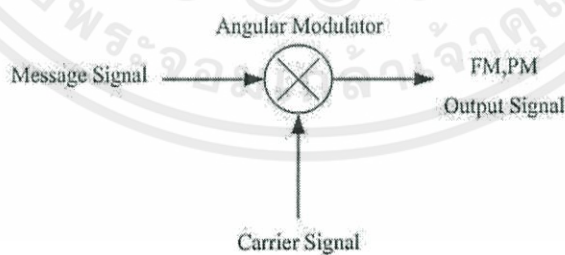
1. การมอดูเลตแอมพลิจูด (Amplitude Modulation หรือ AM) วิธีนี้แอมพลิจูด ของคลื่นพาห์จะเปลี่ยนแปลงตามสัญญาณของข้อมูลที่เข้ามา การมอดูเลตแบบ AM เป็นวิธีที่ง่ายที่สุดใน การมอดูเลต แต่คุณภาพของสัญญาณไม่ดี มีความต้านทานสัญญาณรบกวนต่ำ เหมาะกับข้อมูลที่ไม่ ต้องการคุณภาพมากนัก เช่น สัญญาณเสียง เป็นต้น

2. การมอดูเลตความถี่ (Frequency Modulation หรือ FM) วิธีการนี้เป็นการ เปลี่ยนแปลงความถี่ของคลื่นพาห์ตามสัญญาณของข้อมูลที่เข้ามา การมอดูเลตแบบความถี่ให้คุณภาพ ที่ดีกว่าการมอดูเลตแบบแอมพลิจูด แต่ระบบจะซับซ้อนกว่า

3. การมอดูเลตเฟส (Phase Modulation หรือ PM) เป็นการมอดูเลตที่ใช้ในการ เปลี่ยนแปลงเฟสของคลื่นพาห์ตามสัญญาณข้อมูลที่เข้ามา ทั้งคุณภาพของสัญญาณ และความซับซ้อน ไม่ค่อยแตกต่างจากการมอดูเลตแบบความถี่ ข้อแตกต่างระหว่างการมอดูเลตแบบความถี่ กับการมอดู เลตแบบเฟส คือการมอดูเลตแบบเฟสใช้คลื่นพาห์เพียงความถี่เดียว

แต่การส่งสัญญาณด้วยรีโมตนั้นเป็นการมอดูเลชันทางความถี่ (Frequency Modulation : FM) ซึ่งการมอดูเลตเป็นการผสมสัญญาณข้อมูลข่าวสารเข้าไปกับสัญญาณคลื่นพาห์ (Carrier) ซึ่งสัญญาณนี้มีความถี่ที่เหมาะสมกับช่องสัญญาณนั้นๆ เพื่อให้ข้อมูลข่าวสารที่ส่งเข้าไปใน ช่องสัญญาณเดินทางได้ไกลมากขึ้น การเลือกวิธีการมอดูเลชันขึ้นอยู่กับปัจจัยหลายประการ เช่น ชนิด ของสัญญาณ แบนด์วิดท์ ประสิทธิภาพของระบบที่ต้องการและความต้านทานต่อสัญญาณรบกวน ซึ่ง สามารถแบ่งการมอดูเลชันเชิงมุมออกเป็น 2 แบบด้วยกันคือ

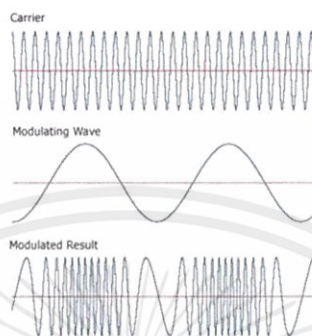
1. การมอดูเลชันทางความถี่ (Frequency Modulation: FM)
2. การมอดูเลชันทางเฟส (Phase Modulation: PM)



รูปที่ 2.11 การมอดูเลชันเชิงมุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การมอดูเลชันทางความถี่ (Frequency Modulation หรือ FM) ที่มอดูเลตแล้วจะมีแอมพลิจูดคงที่แต่ความถี่ของสัญญาณจะเปลี่ยนแปลงไปตามแอมพลิจูดของสัญญาณข้อมูลข่าวสาร ซึ่งการมอดูเลตทางความถี่จะให้คุณภาพที่ดีกว่าการมอดูเลตทางแอมพลิจูด (AM) แต่ระบบจะมีความซับซ้อนกว่า



รูปที่ 2.12 สัญญาณที่มอดูเลตด้วยความถี่

2.5.2 คุณสมบัติที่สำคัญของสัญญาณเอฟเอ็ม มีดังนี้

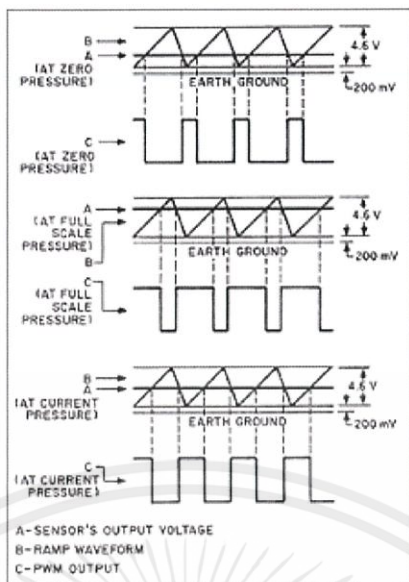
1. มีแอมพลิจูดคงที่ตลอด แต่ความถี่เปลี่ยนแปลงตามสัญญาณที่เข้ามามอดูเลต
2. อัตราการเบี่ยงเบนความถี่ของสัญญาณพาหะมีค่าเท่ากับความถี่ของสัญญาณที่เข้ามา
3. ช่วงความถี่เบี่ยงเบนเป็นสัดส่วนกับแอมพลิจูดของสัญญาณที่เข้ามามอดูเลต

2.5.3 ข้อเสียของการมอดูเลตแบบสัญญาณเอฟเอ็ม คือ

1. ต้องการแบนด์วิดท์ที่มีขนาดกว้าง เนื่องจากสัญญาณข้อมูลมีหลายความถี่
2. คุณภาพดีว่าการมอดูเลตแบบ AM แต่การทำงานจะซับซ้อนกว่า

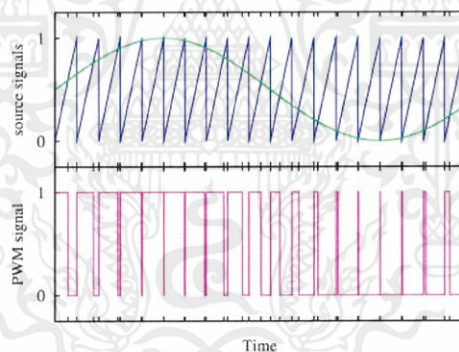
PWM ย่อมาจาก Pulse Width Modulation คือ การปรับความกว้างของพัลส์โดยการนำเอาสองสัญญาณมาเปรียบเทียบกัน สองสัญญาณนี้คือสัญญาณสามเหลี่ยม และสัญญาณที่ต้องการปรับความกว้างของพัลส์ ตัวอย่างเช่น การนำสัญญาณไฟฟ้ากระแสตรงมาเปรียบเทียบกับสัญญาณสามเหลี่ยม จะได้สัญญาณพัลส์ที่มีความกว้างคงที่ ดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.13 สัญญาณพัลส์ที่มีความกว้างคงที่

แต่ถ้านำสัญญาณที่มีการเปลี่ยนแปลงอยู่ตลอดเวลาความกว้างของพัลส์ก็จะเปลี่ยนแปลงตามไปด้วย ดังรูป



รูปที่ 2.14 สัญญาณพัลส์ที่เปลี่ยนแปลงตลอดเวลา

2.6 ภาษาคอมพิวเตอร์ในการเขียนสั่งงานไมโครคอนโทรลเลอร์

มนุษย์ใช้ภาษาในการสื่อสารมาตั้งแต่สมัยโบราณ การใช้ภาษาเป็นเรื่องที่มนุษย์พยายามถ่ายทอดความคิดและความรู้สึกต่างๆ เพื่อการโต้ตอบและสื่อความหมายภาษาที่ใช้ติดต่อสื่อสารในชีวิตประจำวัน เช่น ภาษาไทย ภาษาอังกฤษ หรือภาษาจีน ต่างเรียกว่า ภาษาธรรมชาติ (Natural Language) เพราะมีการศึกษา ได้ยิน ได้ฟัง กันมาตั้งแต่เกิดการใช้งานคอมพิวเตอร์ ซึ่งเป็นเครื่องมือทางอิเล็กทรอนิกส์ให้ทำงานตามที่ต้องการ จำเป็นต้องมีการกำหนดภาษา สำหรับใช้ติดต่อสั่งงานกับคอมพิวเตอร์ ภาษาคอมพิวเตอร์จะเป็น ภาษาประดิษฐ์ (Artificial Language) ที่มนุษย์คิดสร้างมาเอง เป็นภาษาที่มีจุดมุ่งหมายเฉพาะ มีกฎเกณฑ์ที่ตายตัวและจำกัด คืออยู่ในกรอบให้ใช้คำและไวยากรณ์ที่กำหนดและมีการตีความหมายที่ชัดเจนภาษาคอมพิวเตอร์จึงเป็นภาษาที่มีรูปแบบเป็นทาง

การ (Formal Language) ต่างกับภาษาธรรมชาติที่มีขอบเขตกว้างมาก ไม่มีรูปแบบตายตัวที่แน่นอน กฎเกณฑ์ของภาษาจะขึ้นกับหลักไวยากรณ์และการยอมรับของกลุ่มผู้ใช้นั้นๆ

2.6.1 ระดับของภาษาคอมพิวเตอร์

ภาษาคอมพิวเตอร์อาจแบ่งได้เป็น 3 ระดับ คือ ภาษาเครื่อง (Machine Language) ภาษาระดับต่ำ (Low Level Language) และภาษาระดับสูง (High Level Language)

2.6.1.1 ภาษาเครื่อง (Machine Language)

การเขียนโปรแกรมเพื่อสั่งให้คอมพิวเตอร์ทำงานในยุคแรก ๆ จะต้องเขียนด้วยภาษาซึ่งเป็นที่ยอมรับของเครื่องคอมพิวเตอร์ที่เรียกว่า “ภาษาเครื่อง” ภาษานี้ประกอบด้วยตัวเลขล้วน ทำให้เครื่องคอมพิวเตอร์สามารถทำงานได้ทันที ผู้ที่จะเขียนโปรแกรมภาษาเครื่องได้ ต้องสามารถจำรหัสแทนคำสั่งต่าง ๆ ได้ และในการคำนวณต้องสามารถจำได้ว่าจำนวนต่าง ๆ ที่ใช้ในการคำนวณนั้นถูกเก็บไว้ที่ตำแหน่งใด ดังนั้นโอกาสที่จะเกิดความผิดพลาดในการเขียนโปรแกรมจึงมีมาก นอกจากนี้เครื่องคอมพิวเตอร์แต่ละระบบมีภาษาเครื่องที่แตกต่างกันออก ทำให้เกิดความไม่สะดวกเมื่อมีการเปลี่ยนเครื่องคอมพิวเตอร์เพราะจะต้องเขียนโปรแกรมใหม่ทั้งหมด

2.6.1.2 ภาษาระดับต่ำ (Low Level Language)

เนื่องจากภาษาเครื่องเป็นภาษาที่มีความยุ่งยากในการเขียนดังได้กล่าวมาแล้ว จึงไม่มีผู้นิยมและมีการใช้น้อย ดังนั้นได้มีการพัฒนาภาษาคอมพิวเตอร์ขึ้นอีกระดับหนึ่ง โดยการใช้ตัวอักษรภาษาอังกฤษเป็นรหัสแทนการทำงาน การใช้และการตั้งชื่อตัวแปรแทนตำแหน่งที่ใช้เก็บจำนวนต่าง ๆ ซึ่งเป็นค่าของตัวแปรนั้นๆ การใช้สัญลักษณ์ช่วยให้การเขียนโปรแกรมนี้เรียกว่าภาษาระดับต่ำภาษาระดับต่ำเป็นภาษาที่มีความหมายใกล้เคียงกับภาษาเครื่องมากบางครั้งจึงเรียกภาษานี้ว่าภาษาอิงเครื่อง (Machine - Oriented Language) ตัวอย่างของภาษาระดับต่ำได้แก่ ภาษาแอสเซมบลี เป็นภาษาที่ใช้คำในอักษรภาษาอังกฤษเป็นคำสั่งให้เครื่องทำงานเช่น ADD หมายถึง บวกและ SUB หมายถึง ลบ เป็นต้น การใช้คำเหล่านี้ช่วยให้การเขียนโปรแกรมง่ายขึ้นกว่าการใช้ภาษาเครื่องซึ่งเป็นตัวเลขล้วน ดังตารางแสดงตัวอย่างของภาษาระดับต่ำและภาษาเครื่องที่สั่งให้มีการบวกจำนวนที่เก็บอยู่ในหน่วยความจำ

ตารางที่ 2.1 แสดงความสัมพันธ์ของคำสั่งในภาษาระดับต่ำและภาษาเครื่อง

ภาษาระดับต่ำ		ภาษาเครื่อง		รหัสเลขฐานสิบหก	
MOV	AL,05	10110000	00000101	B0	05
MOV	BL,08	10110011	00001000	B3	08
ADD	AL,BL	00000000	11011000	00	D8
MOV	CL,AL	10001000	11000001	88	C1

จากตารางบรรทัดแรก 10110000 00000101 เป็นคำสั่งให้นำจำนวน 5 (หรือเขียนในรูปของเลขฐานสองเป็น 00000101) ไปเก็บในรีจิสเตอร์ชื่อ AL โดยส่วนแรก 10110000 คือรหัสคำสั่ง MOV ซึ่งเป็นการเคลื่อนย้ายข้อมูลจำนวนมาเก็บไว้ในรีจิสเตอร์ AL

บรรทัดที่สอง 10110011 00001000 เป็นคำสั่งให้นำจำนวน 8 (หรือเขียนในรูปของเลขฐานสองเป็น 00001000) ไปเก็บในรีจิสเตอร์ชื่อ BL โดยส่วนแรก 10110011 คือรหัสคำสั่ง MOV ซึ่งเป็นการเคลื่อนย้ายข้อมูลจำนวนมาเก็บไว้ในรีจิสเตอร์ BL

บรรทัดที่สาม เป็นคำสั่งการบวกระหว่างรีจิสเตอร์ AL กับ BL หรือนำ 5 บวก 8 ผลลัพธ์เก็บในรีจิสเตอร์ AL

บรรทัดที่สี่เป็นการนำผลลัพธ์จากรีจิสเตอร์ชื่อALไปเก็บไว้ในรีจิสเตอร์ชื่อCLการใช้โปรแกรมที่เขียนด้วยภาษาแอสเซมบลีนั้น เครื่องคอมพิวเตอร์ไม่สามารถทำงานได้ทันทีจำเป็นต้องมีการแปลโปรแกรมในการแปลที่มีชื่อว่า แอสเซมเบลอร์ (Assembler) ซึ่งแตกต่างกันไปตามเครื่องคอมพิวเตอร์แต่ละชนิดดังนั้นแอสเซมเบลอร์ของเครื่องชนิดหนึ่งจะไม่สามารถใช้แปลโปรแกรมภาษาแอสเซมบลีของเครื่องชนิดอื่น ๆ ได้ภาษาแอสเซมบลีนี้ยังคงใช้ยาก เพราะผู้เขียนโปรแกรมจะต้องเข้าใจในการทำงานของเครื่องคอมพิวเตอร์อย่างละเอียด ต้องรู้ว่าจำนวนที่จะนำมาคำนวณนั้นอยู่ตำแหน่งใดในหน่วยความจำ เช่นกับการเขียนโปรแกรมเป็นภาษาเครื่อง ภาษาแอสเซมบลีจึงมีผู้ใช้บ่อยและมักจะใช้ในกรณีที่ต้องการควบคุมการทำงานภายในของตัวเครื่องคอมพิวเตอร์

2.6.1.3 ภาษาระดับสูง (High Level Language)

ภาษาระดับสูงเป็นภาษาที่สร้างขึ้นเพื่อช่วยอำนวยความสะดวกในการเขียนโปรแกรมกล่าวคือลักษณะของคำสั่งจะประกอบด้วยคำต่าง ๆ ในภาษาอังกฤษ ซึ่งสามารถเข้าใจความหมายได้ทันทีที่เขียนโปรแกรมจึงเขียนโปรแกรมด้วยภาษาระดับสูงได้ง่ายกว่าเขียนด้วยภาษาแอสเซมบลีหรือภาษาเครื่อง ภาษาระดับสูงมีมากมายหลายภาษา อาทิเช่น ภาษาฟอร์แทรน (FORTRAN) ภาษาโคบอล (COBOL) ภาษาปาสคาล (Pascal) ภาษาเบสิก(BASIC) ภาษาวิซวลเบสิก (Visual Basic) ภาษาซี (C) และภาษาจาวา (Java) เป็นต้น โปรแกรมที่เขียนด้วยภาษาระดับสูงแต่ละภาษาจะต้องมีโปรแกรมที่ทำหน้าที่แปลภาษาระดับสูงให้เป็นภาษาเครื่อง เช่น โปรแกรมแปลภาษาฟอร์แทรนเป็นภาษาเครื่อง โปรแกรมแปลภาษาปาสคาลเป็นภาษาเครื่อง คำสั่งหนึ่งคำสั่งในภาษาระดับสูงจะถูกแปลเป็นภาษาเครื่องหลายคำสั่ง

ภาษาระดับสูงที่จะกล่าวถึงในที่นี้ ได้แก่

1. ภาษาฟอร์แทรน (FORmulaTRANstation : FORTRAN)จัดเป็นภาษาระดับสูงที่เก่าแก่ที่สุด ได้รับการคิดค้นขึ้นเป็นครั้งแรก รว พ.ศ. 2497 โดยบริษัท ไอบีเอ็ม เป็นภาษาที่เหมาะสมสำหรับงานที่ต้องการการคำนวณ เช่น งานทางด้านวิทยาศาสตร์ วิศวกรรมศาสตร์ และงานวิจัยต่าง ๆ เนื่องจากแนวคิดในการเขียนโปรแกรมในระยะหลังนี้เปลี่ยนมานิยมการเขียนโปรแกรมแบบโครงสร้างมากขึ้น เนื่องจากลักษณะของคำสั่งภาษาฟอร์แทรนแบบเดิมไม่เอื้ออำนวยที่จะให้เขียน

ได้จึงมีการปรับปรุงโครงสร้างของภาษาฟอร์แทรนให้สามารถเขียนโปรแกรมแบบโครงสร้างขึ้นมาได้ ในปี พ.ศ. 2509 เรียกว่า FORTRAN 66 และในปี พ.ศ. 2520 สถาบันมาตรฐานแห่งชาติของสหรัฐอเมริกา (American National Standard Institute หรือ ANSI) ได้ปรับปรุง FORTRAN 66 และยอมรับให้เป็นภาษาฟอร์แทรนที่เป็นมาตรฐาน เรียกว่า FORTRAN 77 ใช้ได้กับเครื่องคอมพิวเตอร์ที่มีตัวแปลภาษานี้

2. ภาษาโคบอล (Common Business Oriented Language : COBOL) เป็นภาษาที่พัฒนาขึ้นในราวปี พ.ศ. 2502 ต่อมาได้รับการปรับปรุงจากคณะกรรมการซึ่งเป็นตัวแทนของหน่วยงานธุรกิจและรัฐบาลของสหรัฐอเมริกาเป็นภาษาโคบอลมาตรฐานในปี พ.ศ. 2517 เป็นภาษาที่เหมาะสมสำหรับงานด้านธุรกิจ เครื่องคอมพิวเตอร์ขนาดใหญ่ส่วนมากมีโปรแกรมแปลภาษาโคบอล

3. ภาษาเบสิก (Beginner's All - purpose Symbolic Instruction Code : BASIC) เป็นภาษาที่ได้รับการคิดขึ้นเป็นครั้งแรกที่วิทยาลัยดาร์มัทธ (Dartmouth College) และเผยแพร่เป็นทางการในปี พ.ศ. 2508 ภาษาเบสิกเป็นภาษาที่สร้างขึ้นโดยมีจุดประสงค์เพื่อใช้สอนเพื่อใช้สอนเขียนโปรแกรมแทนภาษาคอมพิวเตอร์ภาษาอื่น ซึ่งมีขนาดใหญ่และต้องใช้หน่วยความจำสูงในการทำงาน ซึ่งไม่เหมาะกับเครื่องคอมพิวเตอร์ในสมัยนั้น ภาษาเบสิกเป็นภาษาที่มีขนาดเล็กเป็นตัวแปลภาษานิดที่เรียกว่าอินเทอร์พรีเตอร์นอกจากนี้ภาษาเบสิกเป็นภาษาที่ง่ายต่อการเขียนซึ่งผู้เขียนจะสามารถนำไปประยุกต์กับการแก้ปัญหาต่างๆ ได้ทุกสาขาวิชาผู้ที่เพิ่งฝึกเขียนโปรแกรมใหม่ๆ หรือผู้ที่ไม่ใช่คนเขียนโปรแกรมมืออาชีพ แต่เป็นเพียงวิศวกรหรือนักวิจัย จะสามารถหัดเขียนโปรแกรมภาษาเบสิกได้ในเวลาไม่นานนัก ปกติภาษาเบสิกส่วนใหญ่ใช้กับไมโครคอมพิวเตอร์

4. ภาษาปาสคาล (Pascal) ตั้งชื่อตามนักคณิตศาสตร์ชาวฝรั่งเศส ชื่อเบลส ปาสคาล (Blaise Pascal) ซึ่งเป็นผู้ผลิตเครื่องคิดเลขโดยใช้เฟืองหมุน ภาษาปาสคาลคิดขึ้นในปี พ.ศ. 2514 โดยนิคคอสเวียช (Niklaus Wirth) ศาสตราจารย์วิชาคอมพิวเตอร์ชาวสวิส ภาษาปาสคาลได้รับการออกแบบให้ใช้งานง่ายและมีโครงสร้างที่ดี จึงเหมาะกับการใช้สอนหลักการเขียนโปรแกรม ปัจจุบันภาษาปาสคาลยังคงได้รับความนิยมใช้ในการเรียนเขียนโปรแกรมคอมพิวเตอร์

5. ภาษาซีและซีพลัสพลัส (C และ C++) ภาษาซีเป็นภาษาที่พัฒนาจากห้องปฏิบัติการเบลล์ของบริษัทเอทีแอนด์ทีในปี พ.ศ. 2515 หลังจากพัฒนาขึ้นได้ไม่นาน ภาษาซีก็กลายเป็นภาษาที่นิยมในหมู่นักเขียนโปรแกรมมาก และมีการใช้งานในเครื่องทุกระดับเนื่องจากภาษาซีได้รวมเอาข้อมูลของภาษาระดับสูงและภาษาระดับต่ำเข้าไว้ด้วยกัน กล่าวคือเป็นภาษาที่มีไวยากรณ์ที่เข้าใจง่าย ทำให้เขียนโปรแกรมได้ง่ายเช่นเดียวกับภาษาระดับสูงทั่วไป แต่ประสิทธิภาพและความเร็วในการทำงานดีกว่ามาก เนื่องจากมีการทำงานเหมือนภาษาระดับต่ำ สามารถทำงานได้ในระดับที่เป็นการควบคุมฮาร์ดแวร์ได้มากกว่าภาษาระดับสูงอื่น ๆ ดังจะเห็นว่าภาษาซีเป็นภาษาที่สามารถพัฒนาระบบปฏิบัติการได้

6. ภาษาวิซวลเบสิก (Visual Basic) เป็นภาษาที่พัฒนาต่อมาจากภาษาเบสิก ใช้ไวยากรณ์บางส่วนของภาษาเบสิกในการเขียนโปรแกรมแต่มีแนวคิดและวิธีการพัฒนาโปรแกรมที่แตกต่างจากภาษาเบสิกโดยสิ้นเชิง รวมทั้งการใช้เนื้อที่ในหน่วยความจำก็แตกต่างกันมาก ทั้งนี้เนื่องจากภาษาวิซวลเบสิกใช้แนวคิดที่ต่างออกไป

7. การเขียนโปรแกรมแบบจินตภาพ (Visual Programming) ภาษานี้พัฒนาขึ้นโดยบริษัทไมโครซอฟต์ออกแบบเพื่อเขียนโปรแกรมที่สามารถใช้งานไต่บนระบบปฏิบัติการแบบจ็อยโอ เช่น ระบบปฏิบัติการไมโครซอฟต์วินโดวส์ มีการติดต่อกับผู้ใช้โดยใช้รูปภาพ การเขียนโปรแกรมทำได้ง่ายกว่าการเขียนโปรแกรมแบบเก่ามาก

8. ภาษาจาวา (Java) พัฒนาขึ้นในปี พ.ศ. 2534 โดยบริษัทซันไมโครซิสเต็มส์ เป็นภาษาที่ได้รับความนิยมสูงมาโดยตลอด เนื่องจากเป็นภาษาที่มีความยืดหยุ่นสูง สามารถเขียนโปรแกรมและใช้งานไต่บนเครื่องคอมพิวเตอร์ทุกประเภทและระบบปฏิบัติการทุกรูปแบบ ในช่วงแรกๆที่เริ่มมีการนำภาษาจาวามาใช้งานจะเป็นการใช้งานบนเครือข่ายอินเทอร์เน็ต เป็นภาษาที่เน้นการทำงานบนเว็บ แต่ปัจจุบันสามารถสามารถนำมาประยุกต์สร้างโปรแกรมใช้งานทั่วไปได้นอกจากนี้ เมื่อเทคโนโลยีของการสื่อสารก้าวหน้าขึ้น จนกระทั่งเครื่องคอมพิวเตอร์ปาล์มที่ถือ หรือแม้แต่โทรศัพท์เคลื่อนที่ที่สามารถเชื่อมต่อเข้าสู่ระบบอินเทอร์เน็ตและใช้งานระบบเว็ลด์ไวด์เว็บได้ ภาษาจาวาก็สามารถสร้างส่วนที่เรียกว่า แอปเพล็ต (Applet) ให้อุปกรณ์อิเล็กทรอนิกส์ที่กล่าวข้างต้นเรียกใช้งานจากเครื่องที่เป็นแม่ข่าย (Server) ได้

9. ภาษาเดลฟาย (Delphi) เป็นภาษาที่ได้รับความนิยมภาษาหนึ่ง แนวคิดในการเขียนโปรแกรมภาษาเดลฟายเหมือนกับแนวคิดในการเขียนโปรแกรมภาษาวิซวลเบสิก คือเป็นการเขียนโปรแกรมเชิงจินตภาพ แต่ภาษาพื้นฐานที่ใช้ในการเขียนโปรแกรมจะเป็นภาษาปาสคาลการเขียนโปรแกรมเชิงจินตภาพนี้มีคอมโพเนนต์ (Component) ที่สามารถใช้เป็นส่วนประกอบเพื่อสร้างส่วนติดต่อผู้ใช้ที่เป็นแบบกราฟิก ทำให้ซอฟต์แวร์ที่พัฒนามีความน่าสนใจและใช้งานง่ายขึ้นการเขียนโปรแกรมด้วยภาษาเดลฟายจึงเป็นที่นิยมในการนำไปพัฒนาเป็นโปรแกรมใช้งานมารวมทั้งภาษาปาสคาลเป็นภาษาที่เข้าใจง่าย เหมาะแก่การนำมาใช้สอนเขียนโปรแกรม

ภาษาแอสเซมบลีจัดได้ว่าเป็นภาษาที่มีมาคู่กับไมโครคอนโทรลเลอร์ทุกตระกูล ไม่ว่าจะ MCS-52, PIC, 68HC11 หรือจะเป็นไมโครคอนโทรลเลอร์ ST ก็ตาม แต่ในปัจจุบันภาษา C กับได้รับความนิยมเพิ่มมากขึ้นในการนำมาใช้พัฒนาโปรแกรมสำหรับไมโครคอนโทรลเลอร์รวมทั้ง MCS-51 เนื่องจาก ไมโครคอนโทรลเลอร์ nRF24LE1 ที่ใช้ในปริญญาบัตรนี้ เป็นไมโครคอนโทรลเลอร์ตระกูลเดียวกันกับ MCS-51 มีสถาปัตยกรรมภายในที่คล้ายคลึงกัน การเขียนซอฟต์แวร์ควบคุมหรือการใช้งานฟังก์ชันต่าง ๆ จึงคล้ายคลึงกัน

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ที่มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้ง

1. ภาษา C เป็นภาษาโครงสร้าง งานต่อการทำงานความเข้าใจ ปรับปรุง และพัฒนาต่อ

2. ภาษา C ใช้หลังการเขียนโปรแกรมแบบสมัยใหม่มีทั้งที่เป็นแบบโครงสร้างและออบเจกต์
3. ภาษา C เป็นภาษามาตรฐานไม่ขึ้นกับฮาร์ดแวร์(ไมโครคอนโทรลเลอร์)มีความยืดหยุ่นในการโยกย้ายไปใช้งานกับไมโครคอนโทรลเลอร์ตระกูลอื่นได้ง่าย
4. ตัวคอมไพเลอร์และลิงเกอร์ของภาษา C มีการพัฒนาอยู่ตลอดเวลาจนมีความสามารถสูง
5. ภาษา C มีเครื่องมือในการพัฒนาโปรแกรมที่เรียกว่า Integrated Development environment (IDE) ก้าวหน้ามาก
6. ภาษา C มีความเข้ากันได้กับภาษาแอสเซมบลีในระดับซอร์สโค้ดทั้งแบบ Inline assembly และการลิงก์ (Link) โปรแกรมเข้ากันด้วย
ด้วยเหตุผลดังกล่าวจึงทำให้ภาษา C เป็นตัวเลือกระดับต้นๆในการพัฒนาโปรแกรมสำหรับการใช้งานไมโครคอนโทรลเลอร์ในปัจจุบัน และโปรแกรมที่เลือกใช้ในการเขียนภาษาซีในปริญญาโทครั้งนี้ คือ Keiluvision 3

2.6.2 โครงสร้างของภาษา C51

ภาษา C51 มีรูปแบบและโครงสร้างในการเขียนโปรแกรมแบบเดียวกับภาษา C มาตรฐานทั่วไปหรือที่เรียกกันว่า ANSI C (American National Standards Institute, ANSI) ที่ประกอบด้วยรายละเอียดดังนี้

2.6.2.1 พรีโพรเซสเซอร์ ไดเรกทีฟ (Preprocessor directives)

เป็นชุดคำสั่งที่ใช้ในการจัดการเตรียมข้อมูลไว้สำหรับการประมวลผลก่อนที่จะมีการคอมไพล์ สำหรับคอมไพเลอร์ C51 จะมีไดเรกทีฟทั้งที่เป็นมาตรฐานเดียวกับภาษา C และที่เพิ่มเติมเฉพาะสำหรับ C51

2.6.2.2 การประกาศ (Declarations)

ก่อนใช้งานตัวแปรหรือฟังก์ชัน ต้องมีการประกาศและสร้างตัวแปรหรือฟังก์ชัน ขึ้นมาก่อนการใช้งาน

2.6.2.3 การกำหนดค่า (Definition)

การประกาศและจองหน่วยความจำ หรือการกำหนดค่าให้กับตัวแปรหรือฟังก์ชัน

2.6.2.4 นิพจน์ (Expressions)

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สงวนสิทธิ์ทางการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงชื่อของลิขสิทธิ์ทุกครั้งที่มีการนำมาใช้
ค่าหนึ่ง

2.6.2.5 สเตตเมนต์ (Statements)

คำสั่งการทำงาน คือ คำสั่งที่ใช้ในการทำงานตามความต้องการของผู้เขียนโปรแกรม

2.6.2.6 ฟังก์ชัน (Function)

เป็นส่วนประกอบของโปรแกรมที่กำหนดให้ทำงานอย่างใดอย่างหนึ่งจนเสร็จสิ้น ภายในฟังก์ชันโดยภายในฟังก์ชันจะประกอบไปด้วย การประกาศใช้งานตัวแปร การกำหนดค่าให้กับตัวแปร นิพจน์ แะคำสั่งการทำงาน

2.6.2.7 ฟังก์ชัน main () (Main Function)

ฟังก์ชัน main () เป็นฟังก์ชันที่ต้องมีการประกาศทุกครั้งเมื่อมีการใช้โปรแกรมภาษา C เพราะการทำงานโปรแกรมจะเริ่มต้นที่ฟังก์ชันนี้ และเป็นฟังก์ชันที่ใช้ในการเรียกฟังก์ชันอื่นๆในการทำงาน

2.6.3 การประกาศตัวแปรรีจิสเตอร์หน้าที่พิเศษ (SFR)

ชนิดข้อมูลแบบ SFR

ชนิดข้อมูลและการประกาศใช้งานตัวแปรแบบ SFR เพื่อเป็นการเข้าถึงละใช้งานรีจิสเตอร์หน้าที่พิเศษการประกาศใช้งานตัวแปรชนิดข้อมูล SFR นั้นจะมีการกำหนดในลักษณะแบบเดียวกับตัวแปรภาษา C ทั่วไป แต่เมื่อมีการประกาศใช้งานแล้วจะต้องกำหนดค่าทันที (ตำแหน่งแอดเดรส)

การประกาศใช้งานตัวแปรรีจิสเตอร์หน้าที่พิเศษนั้นจะไม่ใช่ไปตาม ANSI C มีใช้งานเฉพาะใน C51 ที่ใช้มาตรฐานเดียวกับ Keil เท่านั้น เนื่องจากการประกาศตัวแปรดังกล่าวเกี่ยวข้องกับบิตข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 โดยใช้ชนิดข้อมูลแบบ bit, sbit, sfr และ sfr16

2.6.3.1 การใช้งานรีจิสเตอร์อินเตอร์รัปต์จากสัญญาณภายนอก

การอินเตอร์รัปต์ (Interrupt) คือ ขั้นตอนของการขัดจังหวะการทำงานของโปรแกรมที่กำลังทำงานอยู่เพื่อมาทำงานในส่วนองงานที่ได้กำหนดไว้สำหรับอินเตอร์รัปต์ ซึ่งจะช่วยให้ผู้พัฒนาประหยัดเวลาในการทำงานของโปรแกรม ที่ไม่ต้องไปคอยตรวจสอบเงื่อนไขหนึ่งตลอดเวลาของการทำงานองโปรแกรม โดยส่งหน้าที่การตรวจสอบนี้ให้กับการบริการอินเตอร์รัปต์แทน ประเภทของการอินเตอร์รัปต์ใน MCS -51 จะแบ่งออกเป็น 2 ประเภท คือ หนึ่งการอินเตอร์รัปต์จากสัญญาณภายนอก (External Interrupt) และสองการอินเตอร์รัปต์จากสัญญาณภายใน (Internal Interrupt)

การอินเตอร์รัปต์จากสัญญาณภายนอกจะเป็นการตรวจสอบสัญญาณที่รับจาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภายนอกตัว ขณะที่สัญญาณเริ่มมีการเปลี่ยนแปลงจากลอจิกสูงไปเป็นลอจิกต่ำ ลักษณะของการอินเทอร์รัปต์แบบนี้จะมี 2 ลักษณะคือ

1. การอินเทอร์รัปต์จากสัญญาณภายนอกที่ P3.2 หรืออินเทอร์รัปต์สัญญาณ INTO
2. การอินเทอร์รัปต์จากสัญญาณภายนอกที่ P3.3 หรืออินเทอร์รัปต์สัญญาณ INTI

การใช้งานอินเทอร์รัปต์จากสัญญาณภายนอก เป็นการนับสัญญาณจากภายนอกผ่านพอร์ค P3 บิตที่ 2 (P3.2) ซึ่งเป็นขาของสัญญาณอินเทอร์รัปต์ INTO และพอร์ค P3 (P3.3) ซึ่งเป็นขาของสัญญาณอินเทอร์รัปต์ทั้งสองสามารถที่จะกำหนดการเกิดอินเทอร์รัปต์ได้ 2 รูปแบบคือ พิจารณาการเปลี่ยนระดับสัญญาณสูงเป็นระดับสัญญาณลอจิกต่ำ และเปลี่ยนจากลอจิกสูงเป็นลอจิกต่ำ

2.6.3.1.1 รีจิสเตอร์และบิตที่เกี่ยวข้องกับการอินเทอร์รัปต์สัญญาณภายนอกสามารถสรุปได้ดังนี้

1. รีจิสเตอร์ TCON (Timer/Counter Control register) สามารถที่จะอ้างถึงแบบบิตได้ ซึ่งประกอบไปด้วยบิตที่เกี่ยวข้องกับการใช้วานอินเทอร์รัปต์สัญญาณภายนอกได้แก่

- บิต IE1 เป็นบิตแฟล็กแสดงการอินเทอร์รัปต์ของ INT1
- บิต IT1 เป็นบิตเลือกประเภทสัญญาณอินเทอร์รัปต์ของ INT1
- บิต IE0 เป็นบิตแฟล็กแสดงการอินเทอร์รัปต์ของ INTO
- บิต IT0 เป็นบิตเลือกประเภทสัญญาณอินเทอร์รัปต์ของ INTO

2. รีจิสเตอร์ IE (Interrupt enable register) สามารถที่จะอ้างถึงแบบบิตได้ ซึ่งประกอบไปด้วยบิตที่เกี่ยวข้องกับการใช้วานอินเทอร์รัปต์สัญญาณภายนอกได้แก่

- บิต EA บิตเอ็นเอเบิล/ดิสเอเบิลการเกิดอินเทอร์รัปต์โดยรวม
- บิต EX1 บิตเอ็นเอเบิล/ดิสเอเบิลการเกิดอินเทอร์รัปต์ INT1
- บิต EX0 บิตเอ็นเอเบิล/ดิสเอเบิลการเกิดอินเทอร์รัปต์ INTO

2.6.3.2 การใช้งานรีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ตอนุกรม

พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51สามารถกำหนดการทำงานได้ 4 โหมด หรือ 4 รูปแบบ ดังนี้

1. โหมด 0 เป็นโหมดสำหรับการขยายพอร์ตอินพุตเอาต์พุต ทำงานร่วมกับไอซีรีจิสเตอร์ภายนอก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. โหมด 1 เป็นโหมดการเชื่อมต่อแบบอนุกรม UART (Universal asynchronous receive/transmitter)

3. โหมด 2 เป็นโหมดการเชื่อมต่อแบบอนุกรม UART ใช้ข้อมูล 11 บิตและกำหนดอัตราความเร็วในการส่งข้อมูลคงที่

4. โหมด 3 เป็นโหมดการเชื่อมต่อแบบอนุกรม UART ใช้ข้อมูล 11 บิตและสามารถเปลี่ยนแปลงความเร็วในการส่งข้อมูลได้

สำหรับการสื่อสารข้อมูลแบบอนุกรมในบทนี้จะเน้นไปที่การทำงานในโหมด 1 เท่านั้น เนื่องจากเป็นโหมดที่นิยมนำมาใช้ในการติดต่อกับคอมพิวเตอร์ผ่านพอร์ตอนุกรม RS-232 โดยมีรูปแบบคือ เป็นข้อมูลรวมขนาด 10 บิต อันประกอบด้วยบิตเริ่มต้น 1 บิต, บิตข้อมูล 8 บิต และบิตหยุดอีก 1 บิต โดยมีรีจิสเตอร์และบิตที่เกี่ยวข้องดังนี้

1. รีจิสเตอร์ SCON (Serial Port Control Register) ใช้กำหนดโหมดการทำงานของพอร์ตอนุกรมโดยกำหนดค่าในบิต SM0 และ SM1 เพื่อเลือกโหมดการทำงาน
2. รีจิสเตอร์ PCON (Power Control Register) รีจิสเตอร์นี้จะกำหนดค่าในบิต SMOD เป็นบิตกำหนดการทวีคูณของอัตราบอด (Baud rate) ปกติ
3. รีจิสเตอร์ TMOD (Timer/Counter Mode Control Register) รีจิสเตอร์นี้จะกำหนดค่าในบิต M1 เป็นบิตบนสำหรับการกำหนดโหมดการทำงาน ไทเมอร์ 0
4. บิต TH1 ในรีจิสเตอร์ T1 ขนาด 16 บิต ประกอบด้วย TH1 ขนาด 8 บิต และ TL1 ขนาด 8 บิต
5. บิต TR1 ในรีจิสเตอร์ TCON บิต TR1 เป็นบิตที่ใช้ในการกำหนดการทำงาน/หยุดทำงานของไทเมอร์ 1
6. บิต T1 ในรีจิสเตอร์ SCON เป็นบิตแฟลกแสดงการอินเตอร์รัปต์ภายหลังการส่งข้อมูล
7. บิต RI ในรีจิสเตอร์ SCON เป็นบิตแฟลกแสดงการอินเตอร์รัปต์เมื่อมีข้อมูลเข้า
8. รีจิสเตอร์ SUBF เป็นรีจิสเตอร์ที่ใช้ในการรับส่งข้อมูลผ่านพอร์ตอนุกรม

2.7 การใช้โปรแกรม KeilVision 3

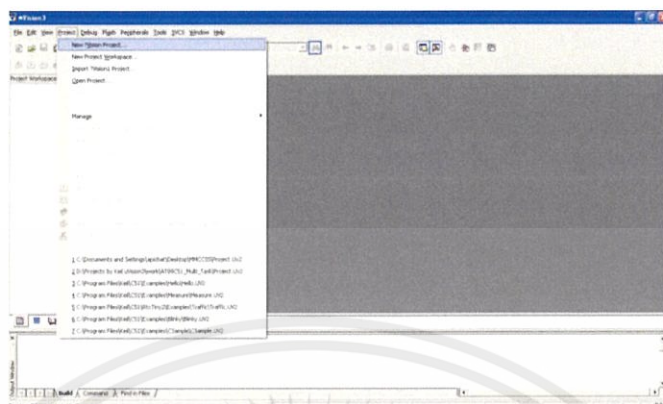
โปรแกรม KeilVision 3 หรือเรียกว่า Keil C51 เป็นโปรแกรมที่สร้างขึ้นโดยบริษัท Keil Software สามารถดาวน์โหลดได้ที่ <http://www.keil.com> ซึ่งเป็นเวอร์ชันทดลองใช้

โปรแกรม KeilVision 3 ช่วยในการเขียนโปรแกรม C51 ได้ง่ายและรวดเร็วยิ่งขึ้น โดยสามารถแปลงภาษา C51 เป็นโค้ดเฮกซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ในการเรียนการสอนเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

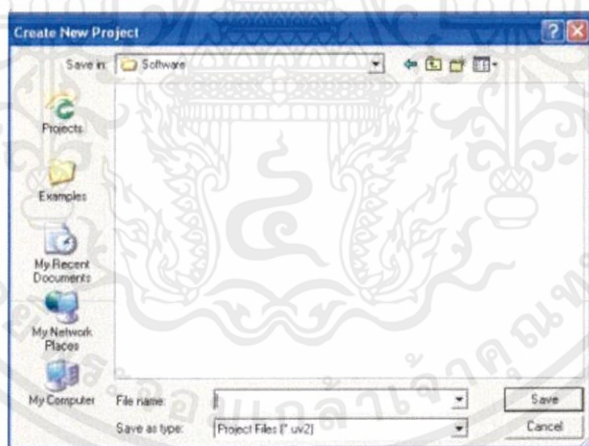
2.7.1 ขั้นตอนที่ 1 การสร้างโปรเจค

1. คลิกเมนู Project >New ?Vision Project



รูปที่ 2.15 หน้าต่างโปรแกรม Keil uVision 3

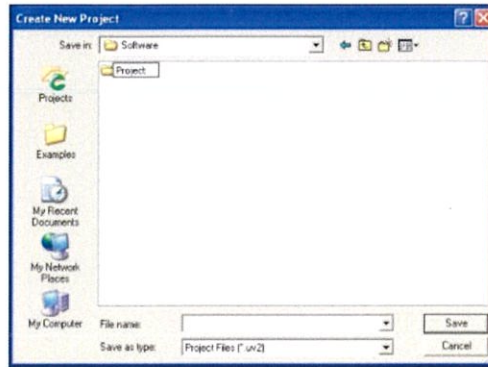
เมื่อเลือกคำสั่ง New ?Vision Project จะปรากฏหน้าต่าง Create New Project ขึ้นมา โดยหน้าต่างการสร้างโปรเจคใหม่จะกำหนดให้เราใส่ชื่อโปรเจค สำหรับใช้งาน ซึ่งในการทำงาน อาจมีหลายโปรเจค เพื่อความง่ายในการใช้งานควรสร้างโฟลเดอร์ (folder) ขึ้นมาใหม่แต่ละโปรเจค โดยเฉพาะ



รูปที่ 2.16 หน้าต่างการสร้างโปรเจคใหม่

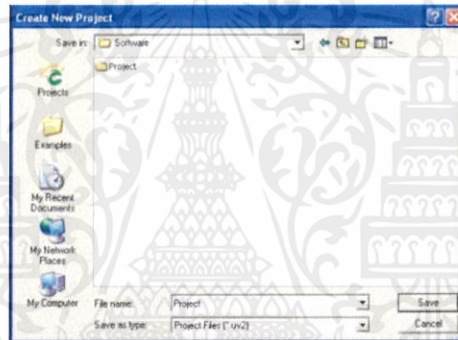
2. สร้างโฟลเดอร์ใหม่ ในตัวอย่างตั้งชื่อโฟลเดอร์ว่า Project

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.17 หน้าต่างการสร้างโพลเดอร์ใหม่

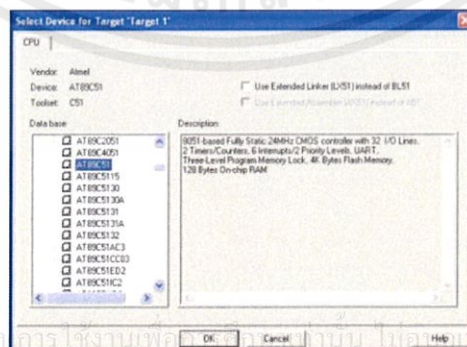
3. คลิกเข้าไปในโพลเดอร์ที่สร้างใหม่ เพื่อสร้างโปรเจกต์ในโพลเดอร์นั้น
4. กรอกชื่อโปรเจกต์ที่ช่อง File name:
5. คลิกปุ่ม Save เพื่อบันทึกไฟล์โปรเจกต์



รูปที่ 2.18 กรอกชื่อโปรเจกต์และทำการบันทึก

หลังจากที่บันทึกโปรเจกต์แล้ว โปรแกรม uVision 3 จะแสดงหน้าต่าง Select Device for Target 'Target1' ขึ้นมา เพื่อให้เราเลือกชิพที่จะใช้งานจากอุปกรณ์ฐานข้อมูล

6. เลือกชิพที่ต้องการ โดยในตัวอย่างเลือกชิพ AT89C51 ของบริษัท ATMEL

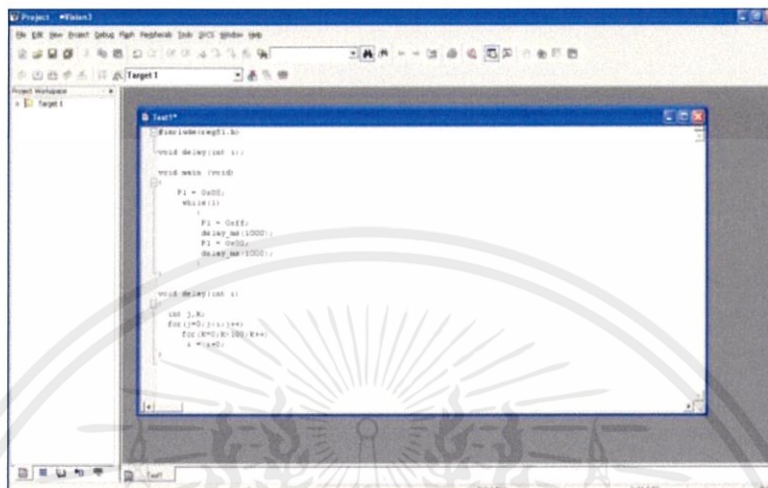


รูปที่ 2.19 เลือกชิพที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเท่านั้น ไม่ควรนำเอกสารนี้ไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงชื่อเอกสารทุกครั้งที่มีการนำไปใช้

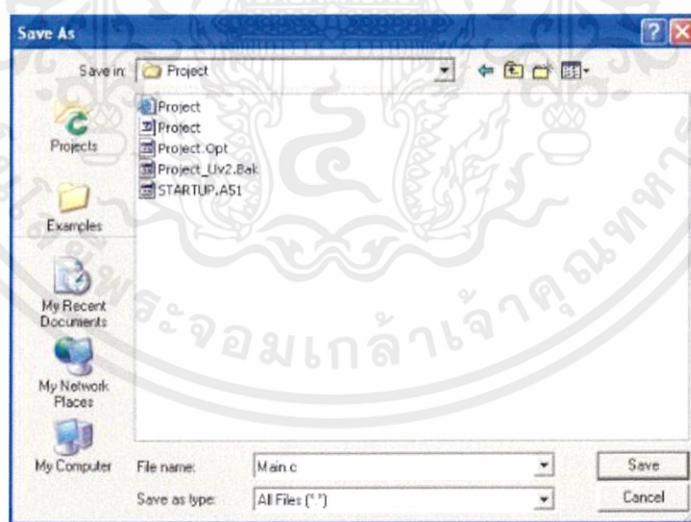
2.7.2 ขั้นตอนที่ 2 เขียนโค้ดโปรแกรม

1. คลิกเมนู File > New เพื่อสร้างไฟล์ที่จะใช้เขียนโค้ดโปรแกรม
2. เขียนโค้ดโปรแกรมที่หน้าต่างว่างตรงส่วนกลางของโปรแกรม uVision 3 (หน้าต่าง Text 1*)



รูปที่ 2.20 หน้าต่าง Text 1*

3. คลิกเมนู File > Save เพื่อบันทึกโค้ดโปรแกรม ถ้าเป็นการบันทึกครั้งแรกโปรแกรม uVision 3 จะโชว์หน้าต่าง Save As ขึ้นมา เพื่อให้ตั้งชื่อไฟล์โค้ดโปรแกรม

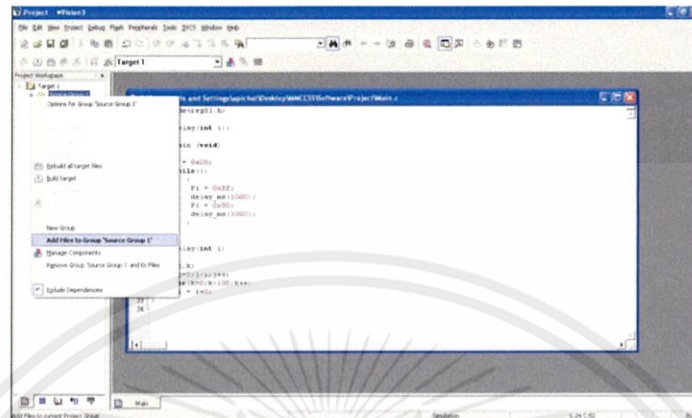


รูปที่ 2.21 หน้าต่างการบันทึก

4. กรอกชื่อไฟล์ในช่อง File Name: โดยในตัวอย่างตั้งชื่อไฟล์เป็น Main.c
5. คลิกปุ่ม Save เพื่อบันทึกโปรแกรมเมื่อสร้างโค้ดโปรแกรมเสร็จเรียบร้อยแล้ว จากนั้นให้เพิ่มไฟล์เข้าไปในโปรเจกของเรา

6. คลิกขวาที่ Source Group1 ตรงหน้าต่างพื้นที่การทำงานของโปรเจกต์ถ้าหา Source Group1 ไม่เจอก็ให้คลิกเครื่องหมายบวกข้างหน้า Target 1

7. คลิกเลือกเมนู Add Files to Group 'Source Group1'

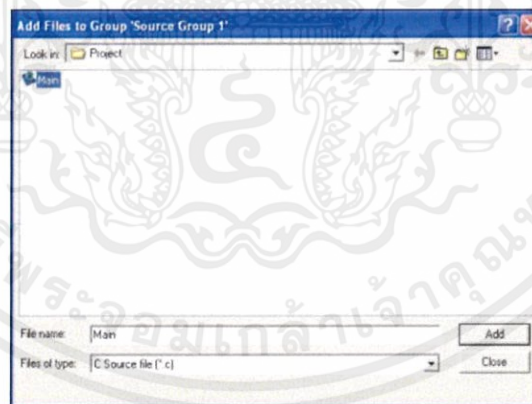


รูปที่ 2.22 ตัวอย่างการเพิ่มไฟล์เข้าไปในโปรเจก

จากนั้นหน้าต่างการเพิ่มไฟล์ไปยังกลุ่ม 'Source Group1' จะปรากฏขึ้นมาให้เราเลือกไฟล์ที่จะเพิ่มเข้าไปในโปรเจก

8. คลิกเลือกไฟล์ที่จะเพิ่มเข้าไปในโปรเจก

9. คลิกปุ่มAddเพื่อเพิ่มไฟล์



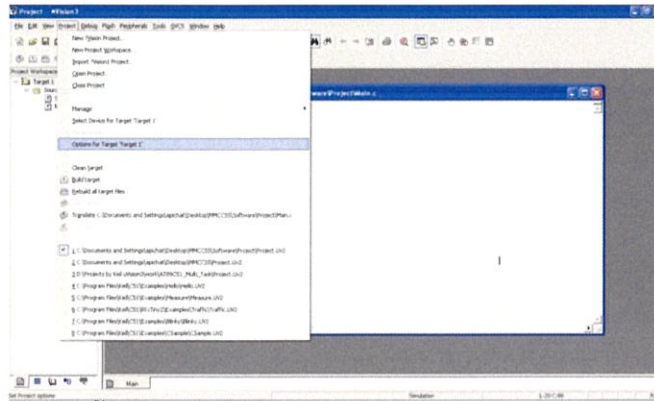
รูปที่ 2.23 หน้าต่างการเพิ่มไฟล์ไปยังกลุ่ม 'Source Group1'

2.7.3 ขั้นตอนที่ 3 การเลือกตัวเลือกเครื่องมือสำหรับซีพียู

โปรแกรม uVision 3 ให้เราสามารถกำหนดค่าพารามิเตอร์ต่างๆ ที่จำเป็นสำหรับซีพียูที่เราจะใช้งาน โดยผ่านทางหน้าต่างOption for Target ก่อนอื่นต้องเลือกซีพียูที่ต้องการใช้งานเสียก่อน ดังแสดงต่อไปนี้

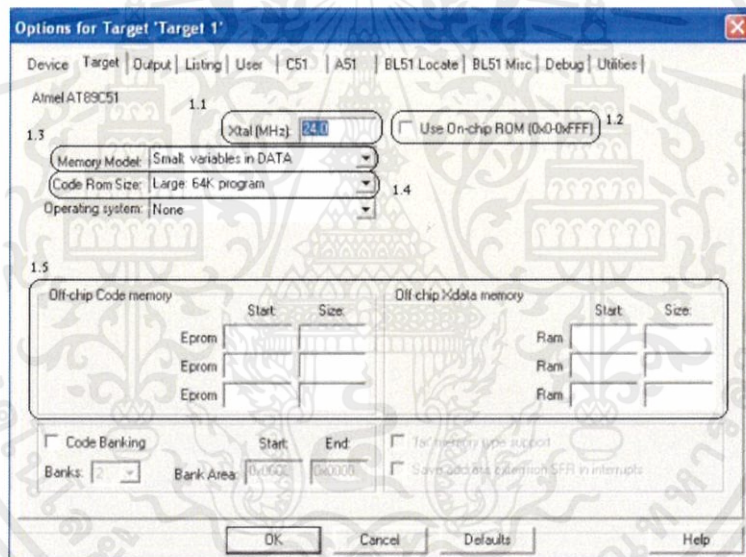
ไม่ว่ากรณีใดๆ ทางสน. อีกรังทงามมี ให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. คลิกเมนู Project > Option for Target 'Target 1' เพื่อเปิดหน้าต่าง Option for Target 'Target1'



รูปที่ 2.24 ขั้นตอนการเปิดหน้าต่างตัวเลือกสำหรับทาร์เก็ต 'Target1'

หน้าต่างตัวเลือกสำหรับทาร์เก็ต 'Target 1' จะปรากฏขึ้นมา ให้เรากำหนดพารามิเตอร์ต่างๆ ตามต้องการ (แถบเมนูในหน้าต่างจะต้องอยู่ที่ทาร์เก็ตถ้าไม่อยู่ให้คลิกแถบ Target)



รูปที่ 2.25 หน้าต่างตัวเลือกสำหรับทาร์เก็ต 'Target 1'

การกำหนดพารามิเตอร์สำหรับซีพียูในส่วนต่างๆ มีรายละเอียดดังแสดงต่อไปนี้

1.1 กำหนดสัญญาณนาฬิกาของซีพียู ซึ่งจะมีค่าเดียวกับค่าความถี่ Xtal ที่ใช้

1.2 กำหนดว่าจะให้ใช้พื้นที่หน่วยความจำเก็บโปรแกรมภายในชิปหรือไม่ ปกติจะ

กำหนดใช้ภายในชิปไว้อัตโนมัติอยู่แล้ว ซึ่งกำหนดภายในไฟล์ STARTUP.A51

1.3 กำหนดเมมโมรีโมเดลหรือโมเดลพื้นที่หน่วยความจำ ปกติจะกำหนดเป็น Small

1.4 กำหนดโค้ดรอมไซส์สำหรับขนาดพื้นที่หน่วยความจำที่จะเก็บโค้ดโปรแกรม ซึ่ง

ไม่ว่ากรณีใดๆ เราอาจจะเก็บไว้สูงสุด 64K

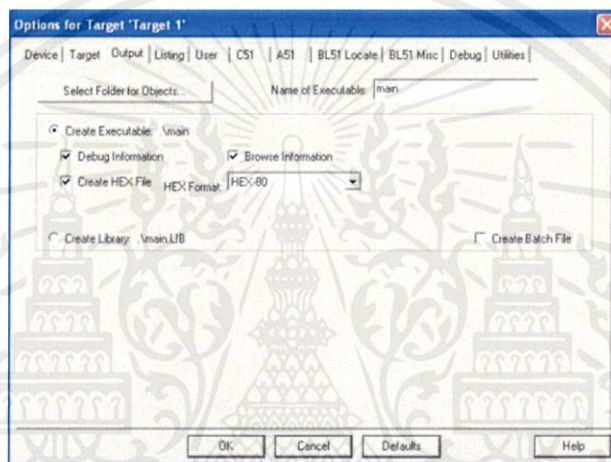
จัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.5 กำหนดพื้นที่ใช้งานหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลไว้ภายนอกชิปทั้งหมด โดยให้เราใส่แอดเดรสเริ่มต้นและขนาดพื้นที่หน่วยความจำที่ใช้ และใช้งานได้ถูกต้องควรทำการเปิดใช้งานการใช้หน่วยความจำภายนอกในไฟล์ STARTUP.A51 ด้วย

2. คลิกแถบ Output เพื่อกำหนดค่าเอาต์พุต

3. ใส่ชื่อไฟล์เอาต์พุต (ไฟล์นามสกุล.hex) ในช่อง Name of Executable เพื่อให้ง่ายควรใส่ชื่อให้เหมือนชื่อไฟล์โค้ดโปรแกรม ถ้าไม่กำหนด โปรแกรม uVision 3 จะตั้งชื่อเดียวกับชื่อโปรเจค

4. กำหนดว่าจะให้โปรแกรม uVision 3 สร้างไฟล์เฮกซ์หรือไม่ ถ้าต้องการให้คลิกที่ช่องว่างข้างหน้า

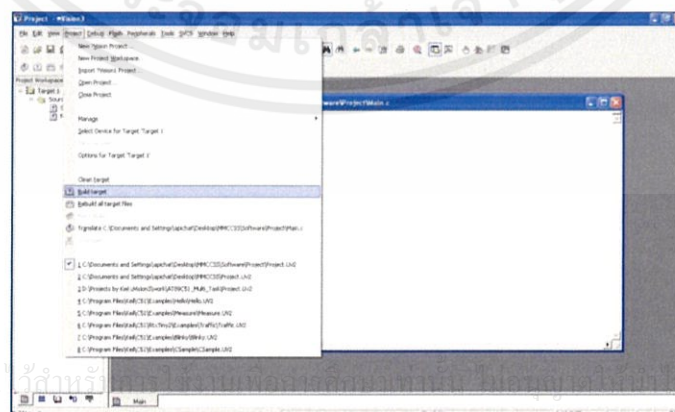


รูปที่ 2.26 การกำหนดการสร้างไฟล์เฮกซ์

2.7.4 ขั้นตอนที่ 4 สร้างโปรเจคเพื่อให้ได้ไฟล์เฮกซ์

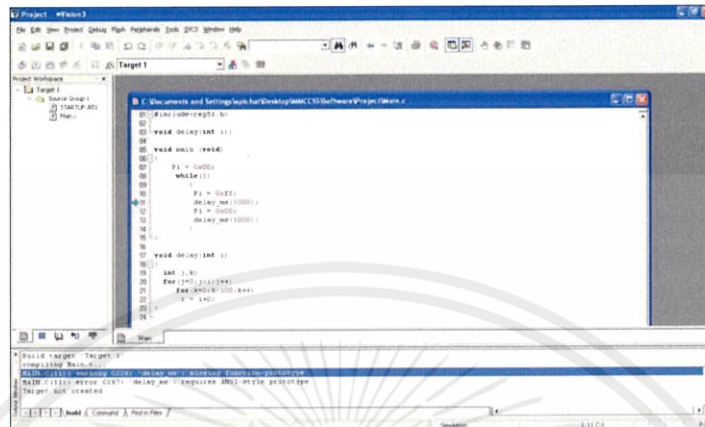
ในขั้นตอนนี้เป็นการแปลงโค้ดโปรแกรมไปเป็นไฟล์เฮกซ์ซึ่งมีขั้นตอนดังต่อไปนี้

1. คลิกเมนู Project > Build target



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงรูปที่ 2.27 การสร้างทาร์เก็ตของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้ามีข้อผิดพลาดเกิดขึ้นโปรแกรม uVision จะแสดงข้อผิดพลาดและข้อความเตือนให้รับทราบที่หน้า Build ของหน้าต่าง Output Window และถ้าเราดับเบิลคลิกบรรทัดที่แสดงข้อผิดพลาดก็จะกระโดดไปยังโค้ดโปรแกรมที่ผิดพลาดนั้นทันที



รูปที่ 2.28 รูปตัวอย่างการแสดงผลข้อผิดพลาด

จากตัวอย่างข้อผิดพลาดที่เกิดขึ้นก็คือ ฟังก์ชัน delay_ms(1000); ไม่มีในการประกาศไว้ เมื่อแก้เป็น delay(1000); ให้ถูกต้องแล้ว ให้ทำการสร้างทาร์เก็ตเหมือนขั้นตอนที่ 1 อีกครั้ง โปรแกรม uVision 3 จะรายงานว่ามีข้อผิดพลาดเกิดขึ้น

```

Build target 'Target 1'
compiling Main.c...
linking...
Program Size: data=9.0 xdata=0 code=78
creating hex file from "main"...
"main" - 0 Error(s), 0 Warning(s).

```

ผลลัพธ์ที่ได้ก็คือไฟล์เฮกซ์หรือไฟล์นามสกุล .Hex นั้นเอง โดยไฟล์เฮกซ์ที่ได้จะเก็บอยู่ในโฟลเดอร์โปรเจกต์ที่ใช้งานปัจจุบัน เราสามารถนำไฟล์นี้ไปโหลดเข้าชิปไมโครคอนโทรลเลอร์ได้เลย

2.8 Appserv

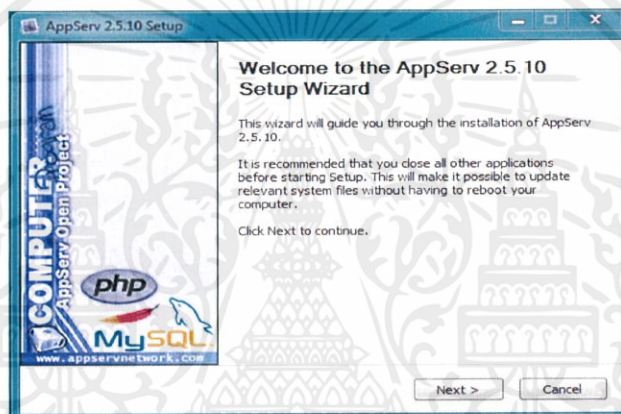
ในการสั่งงานจากอินเทอร์เน็ต ปริมาณนิพจน์นี้ได้เลือกใช้โปรแกรม AppServ มาประยุกต์ใช้ในการสั่งงานจากเครือข่ายอินเทอร์เน็ต เนื่องจากมีการใช้งานที่ง่าย หลักการใช้งานคร่าว ๆ คือ เมื่อได้รับคำสั่งจากหน้าเว็บเพจ โปรแกรม Appserv จะส่งข้อมูลจากอินเทอร์เน็ตมายังเครื่องคอมพิวเตอร์ที่เป็นเซิร์ฟเวอร์ และส่งข้อมูลออกทางพอร์ตที่ได้เชื่อมต่อกับไมโครคอนโทรลเลอร์มาสเตอร์ไว้ จากนั้นไมโครคอนโทรลเลอร์มาสเตอร์ จึงเริ่มทำการส่งข้อมูลไปยัง ตัวสลาฟ ให้ทำการควบคุมเครื่องใช้ไฟฟ้าที่ต้องการต่อไป

AppServ คือโปรแกรมที่รวบรวมเอา Open Source Software หลายๆ อย่างมารวมกัน โดยมี Package หลักดังนี้

- Apache
- PHP
- MySQL
- phpMyAdmin

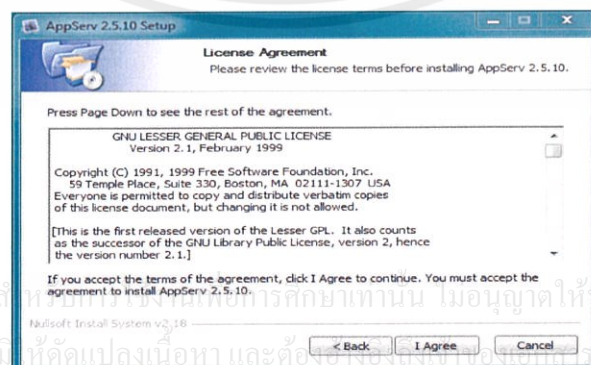
2.8.1 ขั้นตอนการติดตั้งโปรแกรม Appserv มีดังต่อไปนี้

1. คลิกที่ไฟล์ติดตั้ง  appserv-win32-x.x.x.exe เพื่อทำการติดตั้งจะปรากฏหน้าจอตามรูป



รูปที่ 2.29 แสดงการติดตั้งโปรแกรม Appserv

2. หลังจากนั้นเข้าสู่ขั้นตอนเงื่อนไขการใช้งานโปรแกรม โดยโปรแกรม AppServ ได้แจกจ่ายในรูปแบบ GNU License หากผู้ติดตั้ง อ่านเงื่อนไขต่างๆ เสร็จสิ้นแล้ว หากยอมรับเงื่อนไขให้กด I Agree เพื่อเข้าสู่การติดตั้ง ในขั้นต่อไป แต่หากว่าไม่ยอมรับเงื่อนไข ให้กด Cancel เพื่อออกจากกรติดตั้งโปรแกรม AppServ ดังในรูป



รูปที่ 2.30 แสดงการยอมรับในการติดตั้งโปรแกรม

3. ขั้นตอนการเลือกปลายทางที่ต้องการติดตั้ง โดยค่าเริ่มต้นปลายทางที่ติดตั้งจะเป็น C:\AppServ หากต้องการเปลี่ยนปลายทางที่ติดตั้ง ให้กด Browse แล้วเลือกปลายทางที่ต้องการตามรูปที่ 5 เมื่อเลือกปลายทางเสร็จสิ้น ให้กดปุ่ม Next เพื่อเข้าสู่ขั้นตอนการติดตั้งขั้นต่อไป



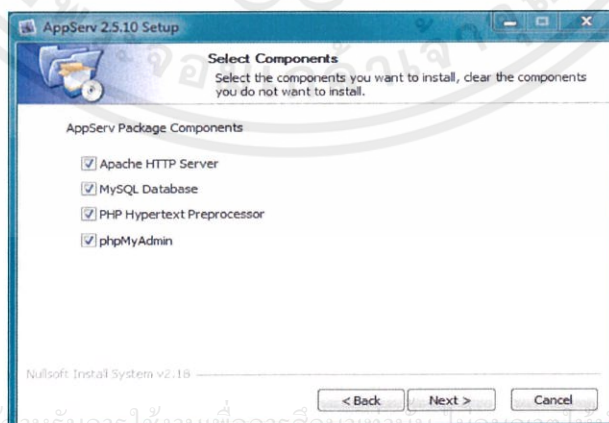
รูปที่ 2.31 เลือกโฟลเดอร์ที่ต้องการในการติดตั้งตัวโปรแกรม Appserv

4. เลือก Package Components ที่ต้องการติดตั้ง โดยค่าเริ่มต้นนั้นจะให้เลือกลงทุก Package แต่หากว่าผู้ใช้งาน ต้องการเลือกลงเฉพาะบาง Package ก็สามารเลือกตามข้อที่ต้องการออกแล้วกด Next โดยรายละเอียดแต่ละ Package มีดังนี้

- Apache HTTP Server คือ โปรแกรมที่ทำหน้าเป็น Web Server
- MySQL Database คือ โปรแกรมที่ทำหน้าเป็น Database Server
- PHP Hypertext Preprocessor คือ โปรแกรมที่ทำหน้าประมวลผลของภาษา
- phpMyAdmin คือ โปรแกรมที่ใช้ในการบริหารจัดการฐานข้อมูล MySQL ผ่าน

PHP

เว็บไซต์



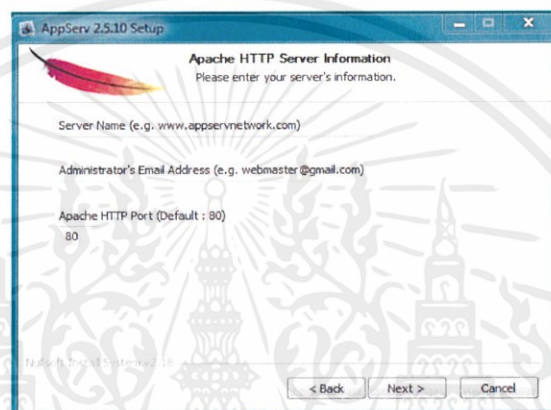
รูปที่ 2.32 โปรแกรมทั้งหมดสำหรับการติดตั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้คัดลอกเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5. กำหนดค่าคอนฟิกของ Apache Web Server มีอยู่ด้วยกันทั้งหมด 3 ส่วน คือ

- Server Name คือช่องสำหรับป้อนข้อมูลชื่อ Web Server
- Admin Email คือช่องสำหรับป้อนข้อมูล อีเมลล์ผู้ดูแลระบบ
- HTTP Port คือช่องสำหรับระบุ Port ที่จะเรียกใช้งาน Apache Web Server

โดยทั่วไปแล้ว Protocol HTTP นั้นจะมีค่าหลักคือ 80 หากว่าท่านต้องการหลีกเลี่ยงการใช้ Port 80 ก็สามารรถแก้ไขได้ หากมีการเปลี่ยนแปลง Port การเข้าใช้งาน Web Server แล้ว ทุกครั้งที่เรียกใช้งานเว็บไซต์ จำเป็นที่จะต้องระบุหมายเลข Port ด้วย เช่นหากเลือกใช้ Port 99 ในการเข้าเว็บไซต์ต้องใช้ <http://www.appservnetwork.com:99> จึงจะเข้าใช้งานได้



รูปที่ 2.33 กำหนด Server Name , Email Address และ Port

6. กำหนดค่าคอนฟิกของ MySQL Database มีอยู่ด้วยกันทั้งหมด 3 ส่วน ตามรูปคือ Root Password คือช่องสำหรับป้อน รหัสผ่านการใช้งานฐานข้อมูลของ Root หรือผู้ดูแลระบบ ทุกครั้งที่เข้าใช้งานฐานข้อมูลในลักษณะที่เป็นผู้ดูแลระบบ ให้ระบุ user คือ root

Character Sets ใช้ในการกำหนดค่าระบบภาษาที่ใช้ในการจัดเก็บฐานข้อมูล, เรียงลำดับฐานข้อมูล, Import ฐานข้อมูล, Export ฐานข้อมูล และ ติดต่อฐานข้อมูล

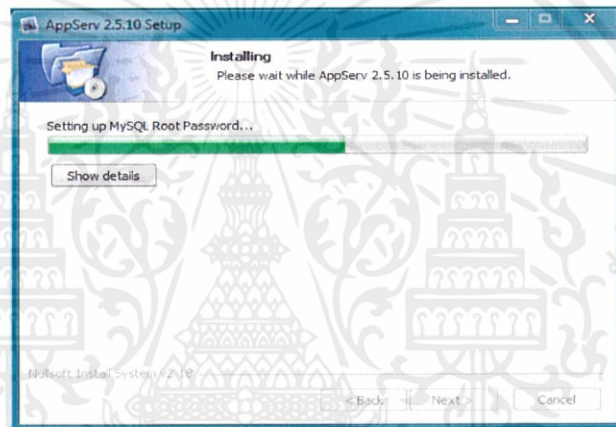
Old Password หากท่านมีปัญหาเกี่ยวกับการใช้งาน PHP กับ MySQL API เวอร์ชันเก่า โดยเจอ Error::Client does not support authentication protocol requested by server; consider upgrading MySQL client ให้เลือกในส่วนของ Old Password เพื่อหลีกเลี่ยงปัญหานี้ Enable InnoDB หากต้องการใช้งานฐานข้อมูลในรูปแบบ InnoDB ให้เลือกในส่วนนี้ด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



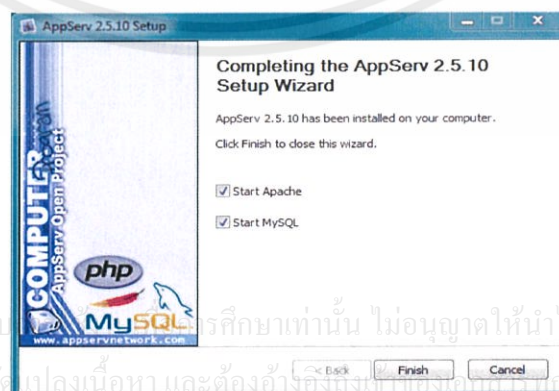
รูปที่ 2.34 กำหนด User , Password สำหรับการติดต่อตัวโปรแกรม

7. หลังจากนั้นรอให้ติดตั้งโปรแกรม จนเสร็จ



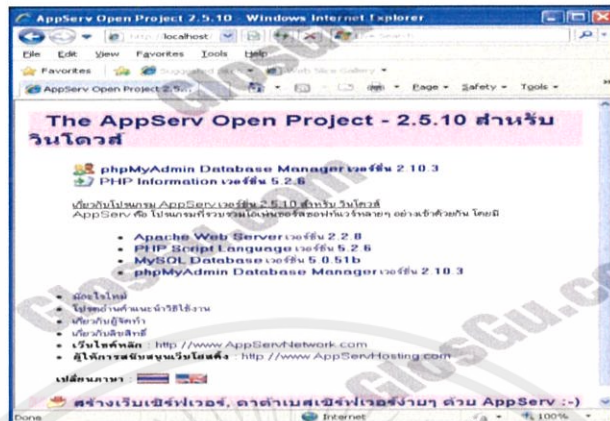
รูปที่ 2.35 กระบวนการติดตั้งโปรแกรม Appserv

8. สิ้นสุดขั้นตอนการติดตั้งโปรแกรม AppServ สำหรับขั้นตอนสุดท้ายนี้จะมีให้เลือกว่าต้องการสั่งให้มีการรัน Apache และ MySQL ทันทีหรือไม่ จากนั้นกดปุ่ม Finish เพื่อเสร็จสิ้นการติดตั้งโปรแกรม AppServ



รูปที่ 2.36 ติดตั้งโปรแกรมเสร็จสมบูรณ์

9. ทำการทดสอบโดยเปิด Web Browser แล้วพิมพ์ “http://localhost/” ที่ช่องกรอก URL ซึ่งจะปรากฏหน้าเว็บดังรูป



รูปที่ 2.37 รูปแบบของตัวโปรแกรม ที่ติดตั้งเสร็จสมบูรณ์

2.8.2 การถอนการติดตั้งระบบ

การลบ (Uninstall) โปรแกรม AppServ (PHP แอปพลิเคชันเซิร์ฟเวอร์) ออกจากเครื่องคอมพิวเตอร์ที่ติดตั้ง สามารถทำได้โดยวิธีการดังต่อไปนี้

1. หยุดการทำงานของ Apache เว็บเซิร์ฟเวอร์
2. หยุดการทำงานของ MySQL ดาต้าเบสเซิร์ฟเวอร์
3. ไปที่ Start→Programs→AppServ→Uninstall→AppServ2.5.10.exe

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

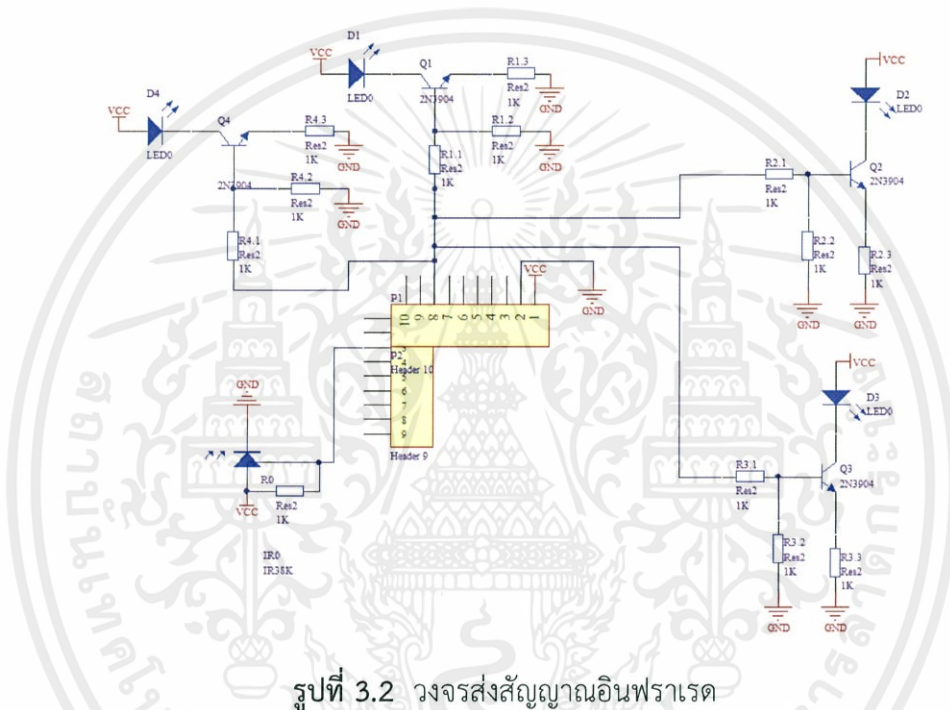
3.2 การออกแบบโครงสร้างของฮาร์ดแวร์

3.2.1 การออกแบบวงจรส่งสัญญาณอินฟราเรด

หลักการออกแบบอุปกรณ์นี้จะคล้ายคลึงกับลักษณะของรีโมททั่วไปแต่มีการปรับเปลี่ยนโครงสร้างเพื่อเพิ่มความสามารถให้มากขึ้น โครงสร้างหลักของอุปกรณ์มีดังนี้

1. ตัวส่งสัญญาณอินฟราเรด
2. วงจรขยายสัญญาณ
3. ไมโครคอนโทรลเลอร์

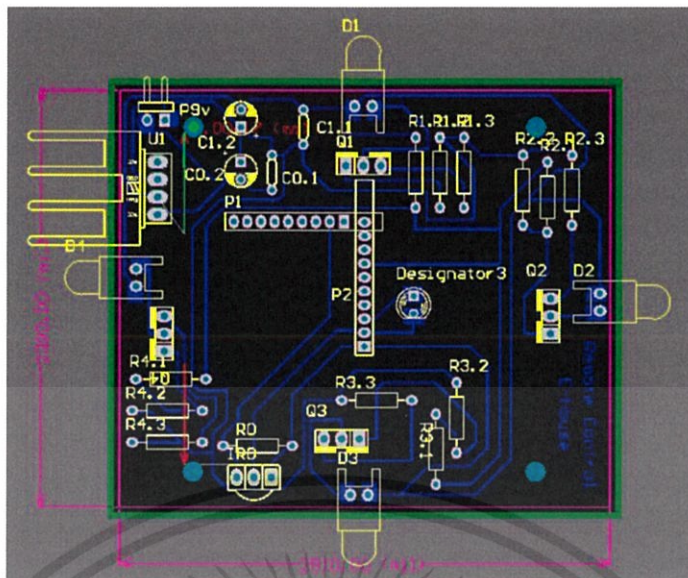
การต่อวงจรและเชื่อมต่อแต่ละองค์ประกอบเป็นดังรูป 3.2



นำวงจรที่ได้ออกแบบมาทดลองการทำงานโดยการเชื่อมต่อไมโครคอนโทรลเลอร์ด้วยพอร์ตยูอาร์ทีเข้ากับคอมพิวเตอร์และสั่งงานจากคอมพิวเตอร์ให้มีการส่งสัญญาณอินฟราเรดออกจากวงจรส่งสัญญาณ จากนั้นต่อวงจรรับสัญญาณอินฟราเรดเข้ากับออสซิลโลสโคป และสังเกตหน้าจอแสดงผลว่ามีการส่งสัญญาณออกมาจริงหรือไม่ เมื่อทำการทดลองและได้ผลแล้วว่าวงจรส่งสัญญาณอินฟราเรดสามารถทำงานได้จริง จึงได้นำวงจรมาสร้างเป็นแผ่นลายวงจรต่อไป

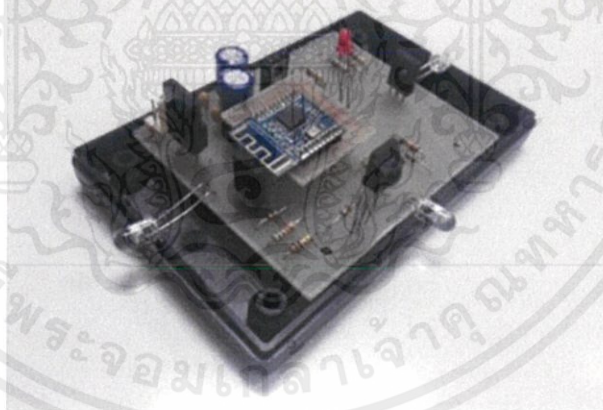
3.2.2 การออกแบบแผ่นลายวงจร

การออกแบบวงจร ออกแบบโดยใช้โปรแกรม Altium Designer Summer 09 เอกสารนี้เป็นกรณำเอาวงจรที่ได้ออกแบบไว้แล้ว มาออกแบบการวางลงในกล่องอุปกรณ์ที่มีขนาด 4 × 2.6 นิ้ว ไม่ว่าจะการออกแบบมีลักษณะดังรูปที่ 3.3 นี้เพื่อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.3 ออกแบบแผ่นลายวงจรโดยใช้โปรแกรม Altium Designer Summer 09

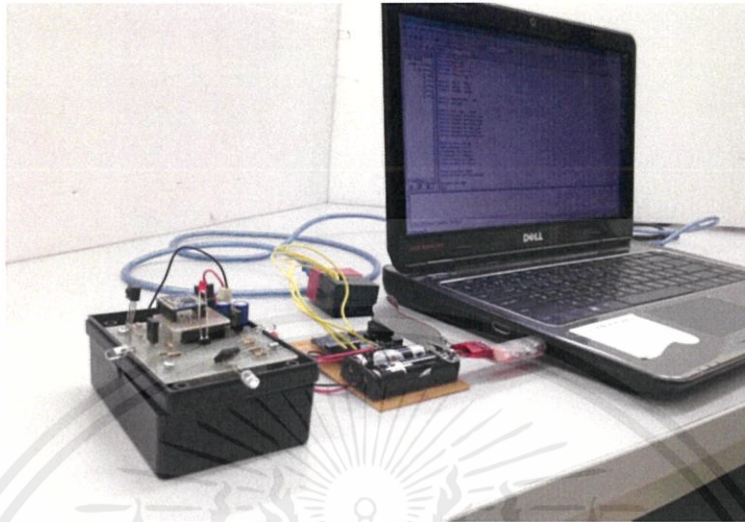
เมื่อได้ทำการสั่งกัดแผ่นลายวงจรออกมาเป็นที่เรียบร้อยแล้ว จึงประกอบอุปกรณ์ต่างๆลงบนแผ่นลายวงจรดังรูปที่ 3.4



รูปที่ 3.4 อุปกรณ์ส่งสัญญาณอินฟราเรดควบคุมเครื่องใช้ไฟฟ้า

เมื่อทำการประกอบอุปกรณ์เรียบร้อยแล้วเป็นไปดังรูปที่ 3.4 จากนั้นจึงทำการทดสอบการทำงานของอุปกรณ์โดยการ เสียบแบตเตอรี่ให้กับอุปกรณ์ นำอุปกรณ์ไปวางไว้ที่ด้านหน้าของเครื่องปรับอากาศ ทำการสั่งงานจากคอมพิวเตอร์ที่มีอุปกรณ์มาสเตอร์ที่ส่งสัญญาณไวเลสได้เชื่อมต่อไว้แล้ว ซึ่งก่อนการจะทดลองอุปกรณ์ที่ประกอบเรียบร้อยแล้วได้นั้น ต้องมีการเขียนซอฟต์แวร์และประกอบอุปกรณ์มาสเตอร์เป็นที่เรียบร้อยแล้วเท่านั้น หากสั่งงานจากคอมพิวเตอร์และเครื่องปรับ

อากาศสามารถปรับเปลี่ยนตามที่ตั้งได้ ถือว่าอุปกรณ์สามารถใช้งานได้ การเชื่อมต่อของอุปกรณ์เป็นดังรูปที่ 3.5



รูปที่ 3.5 การเชื่อมต่ออุปกรณ์

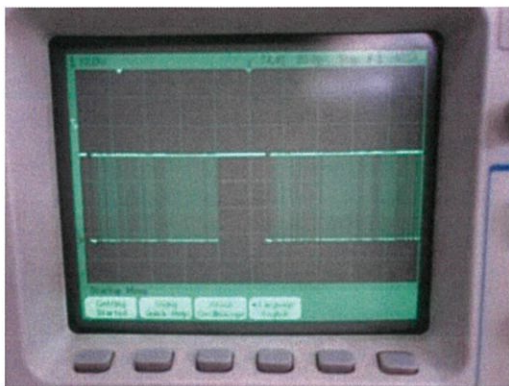
กล่องอุปกรณ์จะมีอินฟราเรดแอลอีดีทั้ง 4 ทิศทางเพื่อการครอบคลุมในการส่งสัญญาณ และเพื่อการประยุกต์ในการใช้งานควบคุมอุปกรณ์อื่นที่สามารถรับสัญญาณอินฟราเรดได้

3.3 การออกแบบการทำงานของไมโครคอนโทรลเลอร์

3.3.1 ศึกษาการส่งสัญญาณอินฟราเรด

ขั้นตอนการศึกษานี้ทำได้โดยการทดลองต่อตัวรับสัญญาณอินฟราเรด (Infrared Receiver) เข้ากับเครื่องออสซิลโลสโคปเพื่อสังเกตลักษณะของสัญญาณอินฟราเรดที่ส่งมาจากรีโมททดลองกดปุ่มรีโมทให้ส่งสัญญาณไปยังตัวรับสัญญาณอินฟราเรด และสังเกตลักษณะของสัญญาณอินฟราเรดที่ส่งมา จากนั้นทำการนับจำนวนบิตของสัญญาณ และวัดค่าความยาวคลื่นของแต่ละบิตทำการบันทึกผลเป็นต้นแบบของสัญญาณการส่งงานของรีโมทยี่ห้ออื่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

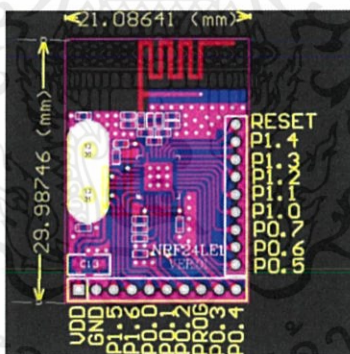


รูปที่ 3.6 ลักษณะของสัญญาณที่ส่งมาจากรีโมท

จากการบันทึกรูปแบบของสัญญาณที่ส่งออกมาพบว่าลักษณะของสัญญาณจะแบ่งเป็น 2 ชุด คือ ชุดแรกเป็นดีไวส์โค้ด และชุดที่ 2 เป็นข้อมูลคำสั่ง โดยแต่ละชุดจะมีเฮดพัลส์เป็นสัญญาณนำร่องก่อนจะมีข้อมูลต่างๆส่งออกมา

3.3.2 การเขียนโปรแกรมตรวจจับสัญญาณอินฟราเรด

ก่อนการตรวจจับสัญญาณต้องสร้างวงจรตัวรับสัญญาณอินฟราเรดต่อเข้ากับไมโครคอนโทรลเลอร์โดยต่อขาออกของสัญญาณเข้ากับ ขา P0.6 ของไมโครคอนโทรลเลอร์เนื่องจากเป็นขาที่ใช้งานการอินเตอร์รัปต์จากภายนอก

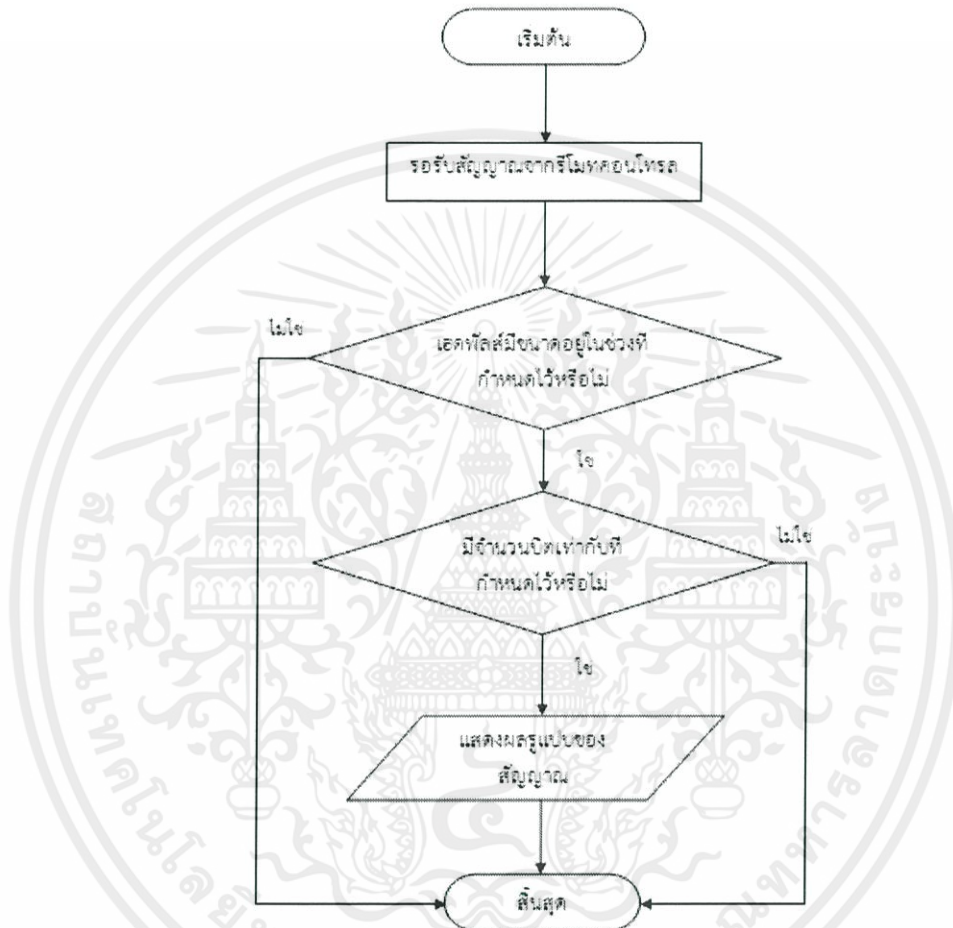


รูปที่ 3.7 แสดงตำแหน่งขาของไมโครคอนโทรลเลอร์

อินเทอร์รัปต์ (Interrupt) คือ การขัดจังหวะการทำงานของซีพียูหรือโปรแกรมให้หยุดพักจากงานที่กระทำอยู่ในปัจจุบัน แล้วกระโดดไปทำงานอีกงานหนึ่งจนเสร็จแล้ว จึงกระโดดกลับมาทำงานชิ้นเดิมที่หยุดพักไว้ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากการใช้งานอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์เมื่อตัวรับสัญญาณอินฟราเรดได้รับสัญญาณอินฟราเรด สัญญาณจะถูกส่งไปยังไมโครคอนโทรลเลอร์และใช้หลักการอินเตอร์รัปต์ในการตรวจจับขอบขาลงของสัญญาณเพื่อวัดขนาดความกว้างของพัลส์ที่เข้ามาว่าเป็นสัญญาณที่ต้องการหรือไม่ ถ้าหากเป็นสัญญาณที่ต้องการ จะมีการแสดงผลลักษณะของสัญญาณที่ละ 8 บิต ในรูปแบบของเลขฐานสิบ หลักการตรวจจับสัญญาณของไมโครคอนโทรลเลอร์เป็นดังรูปที่ 3.8



รูปที่ 3.8 หลักการทำงานของไมโครคอนโทรลเลอร์ในการตรวจจับสัญญาณอินฟราเรด

```
// Open //
unsigned char Temp_up1[9]={136,91,228,15,0,0,0,64,};
unsigned char Temp_up2[20]={136,91,228,0,0,140,60,0,245,0,0,0,0,0,3,0,0,119,};
```

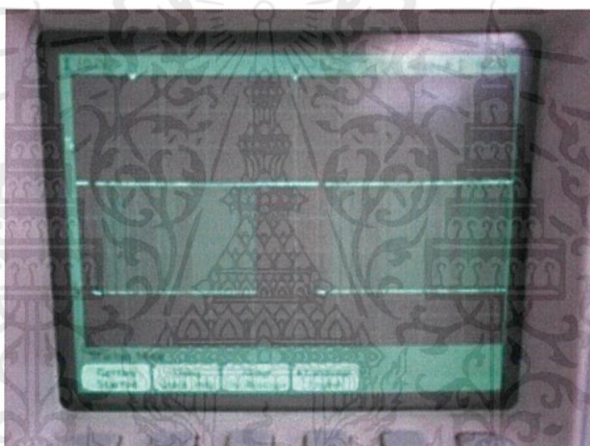
รูปที่ 3.9 การแสดงผลรูปแบบของสัญญาณ

เอกสารนี้เป็นเอกสารที่สง ทดลองการกดปุ่มของรีโมตทุกปุ่มที่ต้องการและบันทึกรูปแบบสัญญาณของแต่ละปุ่มไว้
ไม่ว่ากรกดปุ่มไว้ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.3 การสร้างสัญญาณอินฟราเรดโดยใช้หลักการพัลส์วิธมอดูเลชัน

โดยใช้ไมโครคอนโทรลเลอร์สร้างสัญญาณพัลส์วิธมอดูเลชัน หรือ PWM เพื่อควบคุมการส่งสัญญาณอินฟราเรดไปสั่งงานเครื่องปรับอากาศ สำหรับไมโครคอนโทรลเลอร์ nrf24le1 ใช้ขา P0.2 ในการสร้างสัญญาณพัลส์วิธมอดูเลชัน ซึ่งจะใช้การสร้างสัญญาณพัลส์วิธมอดูเลชันที่มีค่าดิวตี้ไซเคิล (duty cycle) ที่ 50 เปอร์เซ็นต์ และ 0 เปอร์เซ็นต์ ใช้ความถี่ในการสร้างที่ 38 kHz เนื่องจากแอลอีดีที่ใช้ในการส่งสัญญาณมีการส่งสัญญาณในย่านความถี่นี้ ในส่วนของข้อมูลหลักจะมีรูปแบบสัญญาณ 2 ลักษณะสลับ ปะปนกัน จึงได้กำหนดให้เป็นรูปแบบสัญญาณที่ลอจิก 0 และ 1

ในการเขียนโปรแกรมจะมีการเขียนข้อมูลของปุ่มสั่งงานที่ได้ทำการบันทึกไว้ก่อนหน้านี้อยู่แล้ว ว่าปุ่มใดมีลักษณะอย่างไร มีจำนวนกี่บิตและมีความกว้างพัลส์เท่าใด เมื่อต้องการสั่งงานคำสั่งไหน โปรแกรมจะเรียกข้อมูลที่ได้นั้นมาไว้ นำมาเป็นแม่แบบในการสร้างสัญญาณที่มีการเรียงลอจิก 0 และ 1 ที่แตกต่างกันในแต่ละปุ่มคำสั่ง



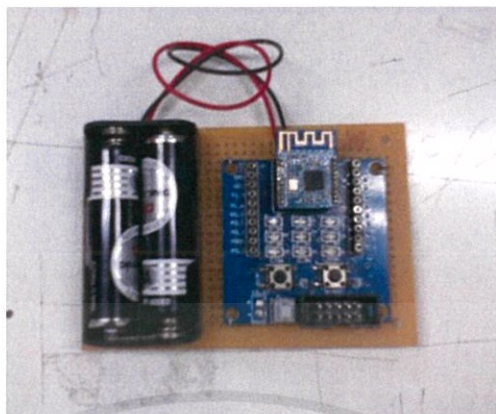
รูปที่ 3.10 ลักษณะของสัญญาณที่ส่งมาจากอุปกรณ์ที่สร้างขึ้น

3.3.4 การออกแบบการทำงานของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์จะแบ่งการทำงานออกเป็น 2 ส่วน คือ มาสเตอร์และสลาฟ ส่วนของไมโครคอนโทรลเลอร์มาสเตอร์จะเชื่อมต่อกับคอมพิวเตอร์ที่ติดต่อกับเซิร์ฟเวอร์อินเทอร์เน็ต และส่วนของไมโครคอนโทรลเลอร์สลาฟนั้นจะอยู่ในกล่องอุปกรณ์ส่งสัญญาณอินฟราเรดควบคุมเครื่องปรับอากาศ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

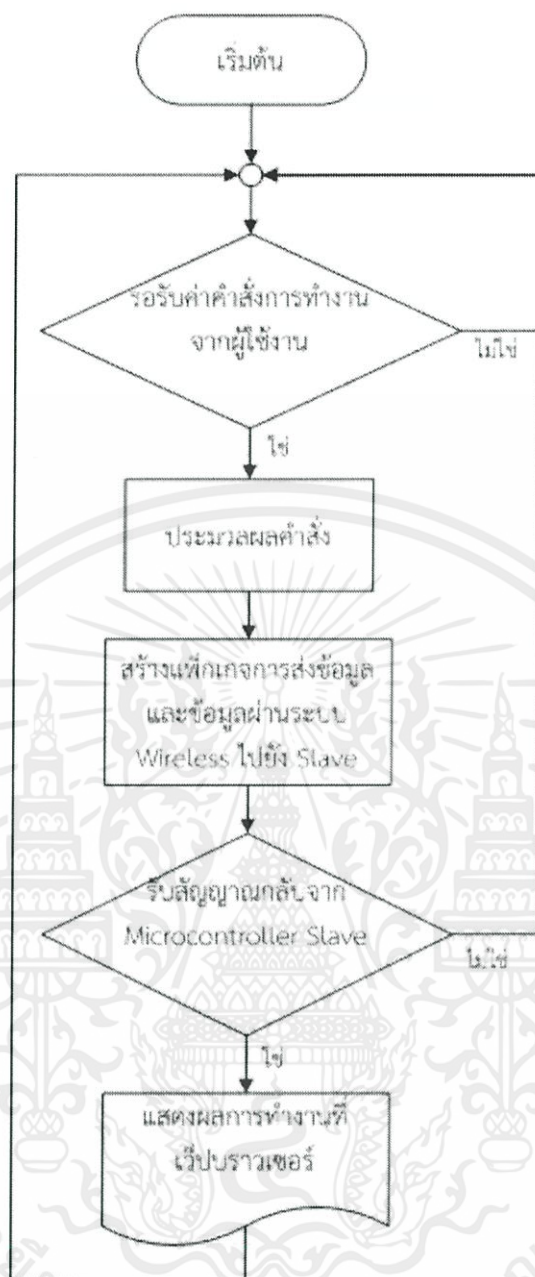
3.3.4.1 การทำงานของไมโครคอนโทรลเลอร์มาสเตอร์



รูปที่ 3.11 ไมโครคอนโทรลเลอร์มาสเตอร์

ไมโครคอนโทรลเลอร์มาสเตอร์เปรียบเสมือนสื่อกลางในการสื่อสารระหว่างผู้ใช้งานและอุปกรณ์โดยมีขั้นตอนการทำงานดังนี้ไมโครคอนโทรลเลอร์มาสเตอร์เชื่อมต่อกับคอมพิวเตอร์ด้วยซีเรียลพอร์ตผ่านทางบอร์ดยูอาร์ทีเมื่อมีการสั่งงานมาจากเว็บเบราว์เซอร์ ข้อมูลจะถูกส่งมายังไมโครคอนโทรลเลอร์และจะมีการประมวลผลคำสั่งพร้อมกับการสร้างแพ็คเกจข้อมูลเพื่อส่งข้อมูลนั้นไปยังไมโครคอนโทรลเลอร์โดยส่งผ่านระบบไวลเลสและสถาปนาส่งค่ากลับมายังมาสเตอร์เพื่อนำไปแสดงสถานะการทำงานของเครื่องปรับอากาศบนหน้าเว็บเบราว์เซอร์

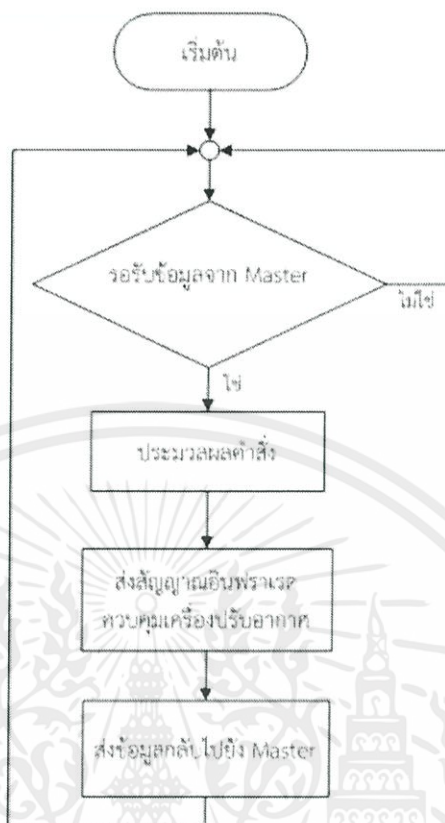
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.12 ขั้นตอนการทำงานของไมโครคอนโทรลเลอร์มาสเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

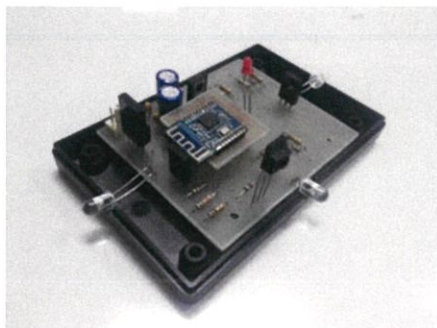
3.3.4.2 การทำงานของไมโครคอนโทรลเลอร์สลาฟ



รูปที่ 3.13 ขั้นตอนการทำงานของไมโครคอนโทรลเลอร์สลาฟ

ไมโครคอนโทรลเลอร์สลาฟจะเชื่อมต่อกับวงจรส่งสัญญาณอินฟราเรดเพื่อส่งสัญญาณไปควบคุมเครื่องปรับอากาศ โดยรับคำสั่งมาจากมาสเตอร์ในรูปแบบของแพ็คเกจข้อมูล จากนั้นไมโครคอนโทรลเลอร์จะทำการประมวลผล ถอดคำสั่งการทำงานและส่งสัญญาณอินฟราเรดไปสั่งงานเครื่องปรับอากาศตามคำสั่งของผู้ใช้งานพร้อมกับการส่งข้อมูลกลับไปยังมาสเตอร์เพื่อแสดงสถานะของคำสั่งที่ได้สั่งงานเครื่องปรับอากาศไปแล้ว และที่ตัวสลาฟนี้ยังมีอินฟราเรดรีซีฟเวอร์รับสัญญาณรีโมตปกติที่ใช้สั่งงานเครื่องปรับอากาศ เพื่อส่งคำสั่งนั้นไปแสดงสถานะการทำงานที่หน้าเว็บเบราว์เซอร์ได้เช่นเดียวกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



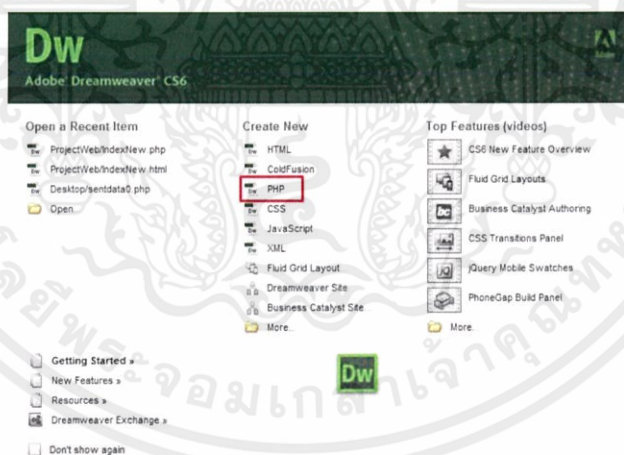
รูปที่ 3.14 ไมโครคอนโทรลเลอร์สถาฟ

3.4 การออกแบบระบบการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต

ออกแบบเว็บเพจสำหรับสั่งงานไมโครคอนโทรลเลอร์ด้วยชุดคำสั่งภาษาพีเอชพีให้ส่งสัญญาณอินฟราเรดเพื่อทำการควบคุมการทำงานของเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต

3.4.1 การสร้างเว็บเพจด้วยโปรแกรม Dreamweaver

1. หลังจากติดตั้งโปรแกรม Dreamweaver แล้ว ให้ทำการเปิดโปรแกรมขึ้นมาจากนั้นสร้างหน้าเว็บขึ้นมาใหม่ โดยคลิกที่ PHP



รูปที่ 3.15 หน้าต่างการสร้างหน้าเว็บใหม่

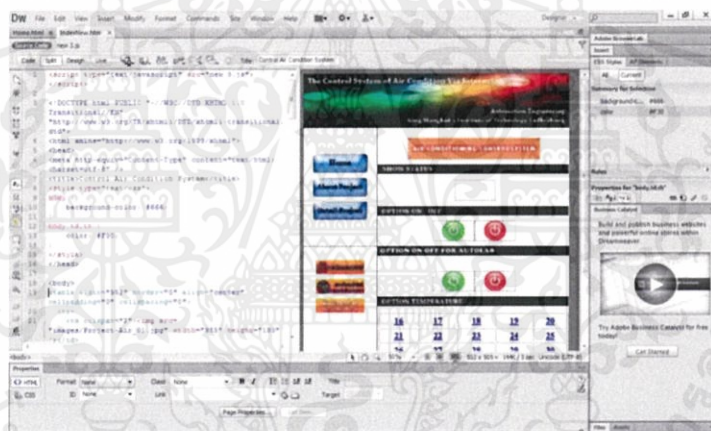
2. ออกแบบเว็บเพจที่จะนำมาเป็นหน้าแรกสำหรับล็อกอินเข้าสู่ระบบการสั่งงานไมโครคอนโทรลเลอร์เพื่อควบคุมการทำงานของเครื่องปรับอากาศ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.16 หน้าต่างการออกแบบเว็บเพจสำหรับล็อกอินเข้าสู่ระบบ

3. ออกแบบเว็บเพจที่ใช้สำหรับการสั่งงานไมโครคอนโทรลเลอร์เพื่อควบคุมการทำงานของเครื่องปรับอากาศ



รูปที่ 3.17 หน้าต่างการออกแบบเว็บเพจสำหรับการสั่งงานไมโครคอนโทรลเลอร์

3.4.2 สร้างไฟล์คำสั่งภาษาพีเอชพี

ทำการติดตั้งโปรแกรม Notepad++ และเขียนโค้ดภาษาพีเอชพีที่ใช้ในการสั่งงานไมโครคอนโทรลเลอร์ และบันทึกไปที่ C: >Appserv> www

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

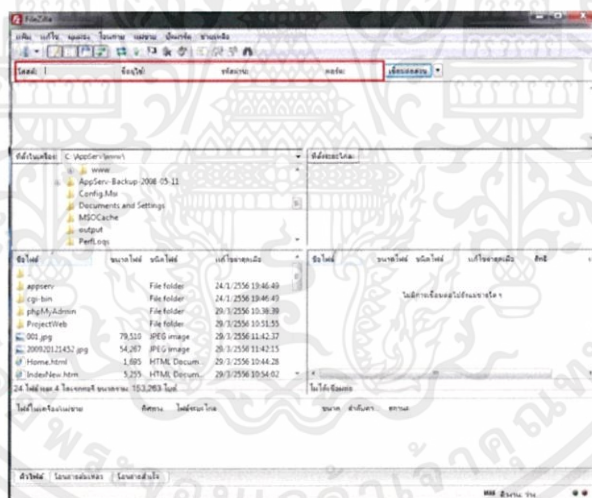
1 <?php
2 exec("mode com8: BAUD=38400 PARITY=N DATA=8 STOP=1 timeout 0x1000 0x1000");
3 $fp = fopen("com8", "w");
4 // $fp = fopen("dev/ttyUSB0", "w"); //use this for Linux
5 fwrite($fp, "0"); //write string to serial
6 fclose($fp);
7
8
9
10 <html>
11 <head><meta http-equiv="refresh" content="0;URL=http://localhost/ProjectWeb/indexOpen.php"></head>
12 <title>
13 </body>
14 </title>
15 </html>

```

รูปที่ 3.18 หน้าต่างการเขียนโค้ดสั่งงานไมโครคอนโทรลเลอร์

3.4.3 การติดต่อกับเซิร์ฟเวอร์

1. หลังจากที่ได้ทำการติดต่อขอพื้นที่เว็บไซต์บนเว็บเซิร์ฟเวอร์แล้ว ให้ทำการติดตั้งโปรแกรม FileZillaเพื่อใช้ในการอัปโหลดไฟล์ไปยังเว็บเซิร์ฟเวอร์
2. ในการอัปโหลดไฟล์ไปยังเว็บเซิร์ฟเวอร์นั้นจะต้องทำการกรอกโฮสต์, ชื่อผู้ใช้, รหัสผ่าน และพอร์ต ซึ่งข้อมูลเหล่านี้จะได้มาจากขั้นตอนการขอพื้นที่เว็บไซต์บนเว็บเซิร์ฟเวอร์



รูปที่ 3.19 หน้าต่างโปรแกรม FileZilla

3. ทำการอัปโหลดไฟล์ทั้งหมดของเว็บเพจขึ้นไปยังเว็บเซิร์ฟเวอร์ เพื่อใช้เป็นเว็บไซต์สำหรับระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

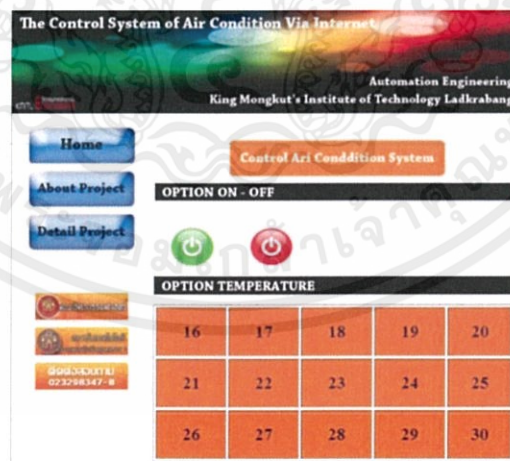
ผลการทดสอบ

4.1 วิธีการสั่งงานเครื่องปรับอากาศ

การสั่งงานสามารถสั่งงานได้จากหน้าเว็บเบราว์เซอร์โดยมีแอดเดรสคือ www.ehouse-air.com โดยจะมีการใส่รหัสก่อนการใช้งาน เพื่อป้องกันความปลอดภัยในการแทรกแซงการทำงานของผู้อื่น



รูปที่ 4.1 หน้าเว็บเพจการใส่รหัสก่อนการใช้งาน



รูปที่ 4.2 หน้าเว็บเพจการสั่งงาน

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อใช้ในการใช้งานเพื่อการศึกษาเท่านั้น ไม่สามารถนำไปใช้เพื่อประโยชน์ทางการค้า
ไม่ว่ากรณีใดๆก็ตาม ผู้ใช้สามารถใช้งานได้ฟรีโดยไม่เสียค่าใช้จ่าย และข้อมูลที่มีอยู่ในเอกสารนี้เป็นข้อมูลที่ไม่ได้
เมื่อใส่รหัสผ่านที่ถูกต้อง จะสามารถเข้าไปยังหน้าเว็บเพจการสั่งงานได้ ผู้ใช้งานสามารถสั่งงาน
ปิด-เปิด หรือปรับเปลี่ยนอุณหภูมิได้ตามต้องการ เมื่อมีการสั่งงานข้อมูลจะถูกส่งมายังอุปกรณ์
มาสเตอร์ที่เชื่อมต่อกับคอมพิวเตอร์ที่เป็นเซิร์ฟเวอร์

จากนั้นอุปกรณ์มาสเตอร์จะส่งสัญญาณด้วยระบบไร้สายไปยังอุปกรณ์สลาฟ อุปกรณ์สลาฟ จะทำการส่งสัญญาณอินฟราเรดไปควบคุมเครื่องปรับอากาศ ให้ทำการเปิด ปิด และปรับอุณหภูมิของ เครื่องปรับอากาศตามที่ต้องการ



รูปที่ 4.3 ลักษณะการส่งสัญญาณอินฟราเรดควบคุมเครื่องปรับอากาศ

4.2 ข้อจำกัดในการใช้งาน

4.2.1 ระยะห่างของเครื่องปรับอากาศและอุปกรณ์สลาฟ

เพื่อความแม่นยำในการสั่งงานเครื่องปรับอากาศและอุปกรณ์สลาฟควรมีระยะห่างไม่เกิน 4 เมตร และระยะห่างสูงสุดที่สามารถทำงานได้คือ ระยะห่างประมาณ 7 เมตร

4.2.1 ในการสั่งงานเครื่องปรับอากาศยี่ห้ออื่น

ต้องทำการแกะโค้ดใหม่ เนื่องจากโค้ดการสั่งงานเครื่องปรับอากาศแต่ละยี่ห้อ มีความแตกต่างกัน จึงเกิดข้อจำกัดในการสั่งงานของเครื่องปรับอากาศต่างยี่ห้อกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

สรุปผลการทดสอบและข้อเสนอแนะ

5.1 สรุปผลการทดสอบ

จากการทดลองเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์สร้างสัญญาณพัลส์วิทมอดูเลชันนั้น ได้ผลตามที่ต้องการ คือ สามารถส่งสัญญาณอินฟราเรดออกมาได้ตามสัญญาณรีโมทที่ใช้ในการควบคุมเครื่องปรับอากาศทั่วไป และจากการทดลองการส่งสัญญาณอินฟราเรดควบคุมเครื่องปรับอากาศก็ได้ผลตามที่ต้องการเช่นกัน คือ สามารถสั่งงานเครื่องปรับอากาศจากเว็บเบราว์เซอร์ได้ตามที่ตั้งเป้าหมายไว้ แต่การส่งสัญญาณจากรีโมทจริงนั้น มีคำสั่งที่ค่อนข้างซับซ้อน ในการถอดสัญญาณของรีโมทแล้วนำสัญญาณนั้นมาส่งค่าออกจากรูปลักษณ์ที่สร้างขึ้น จึงเป็นเพียงการส่งคำสั่งขั้นพื้นฐาน ในการควบคุมเครื่องปรับอากาศ ที่ใช้ในการเปิด-ปิด และปรับอุณหภูมิแบบปกติ

จากที่กล่าวมาข้างต้น การศึกษา ออกแบบ และสร้างอุปกรณ์ควบคุมเครื่องปรับอากาศที่สามารถสั่งงานผ่านระบบอินเทอร์เน็ต ได้ประสบความสำเร็จตามวัตถุประสงค์ที่ตั้งไว้ ซึ่งจะก่อให้เกิดความสะดวกสบายของผู้ใช้งาน และง่ายต่อการบริหารจัดการการใช้งานเครื่องปรับอากาศ เพื่อความประหยัดและปลอดภัย

5.2 ปัญหาที่พบ

การสั่งงานเครื่องปรับอากาศในยี่ห้อที่ต่างกัน โค้ดในการสั่งงานย่อมต่างกัน จึงต้องใช้เวลาในการถอดโค้ดของยี่ห้อที่ต้องการก่อนที่จะนำมาสั่งงาน ฉะนั้นโดยเริ่มต้นของอุปกรณ์ตัวนี้ยังไม่สามารถสั่งงานเครื่องปรับอากาศได้ทุกยี่ห้อ

5.2 ข้อเสนอแนะ

1. ไมโครคอนโทรลเลอร์ nrf24le1 เป็นไมโครคอนโทรลเลอร์ที่ค่อนข้างมีข้อจำกัดในหลายด้าน เช่น ระยะเวลาการส่งสัญญาณ ความจำ และฟังก์ชันการทำงานที่จำกัด ดังนั้น การจะนำแนวคิดจากโครงการนี้ไปใช้งานจึงอาจจะประยุกต์ใช้กับไมโครคอนโทรลเลอร์ที่มีความสามารถมากกว่าไมโครคอนโทรลเลอร์ชนิดนี้

2. การเพิ่มระยะห่างของการสั่งงานจากอุปกรณ์สลาฟไปยังเครื่องปรับอากาศ สามารถทำได้ โดยการปรับเปลี่ยนวงจรส่งสัญญาณอินฟราเรดให้มีการขยายสัญญาณเพิ่มขึ้นเพื่อเพิ่มความแม่นยำและความสะดวกในการใช้งาน

เอกสารนี้เป็นเอกสารทศงานไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. ในการใช้งานเครื่องใช้ไฟฟ้าแต่ละชนิดจะต้องใช้รีโมทหลายตัวตามจำนวนของเครื่องใช้ไฟฟ้า แต่เราสามารถนำแนวคิดจากโครงการนี้มาใช้ในการออกแบบรวบรวมสัญญาณรีโมทของเครื่องใช้ไฟฟ้าแต่ละชนิดไว้ในตัวเดียวกัน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

- [1] เขียนโปรแกรมไมโครคอนโทรลเลอร์ด้วยภาษาซี. (2551). 10 กุมภาพันธ์ 2556, จาก <http://www.thaiembedded.com/blog/?p=50>
- [2] การใช้โปรแกรม Keil uVision 3 เบื้องต้น. 10 กุมภาพันธ์ 2556, จาก <http://www.mcuthailand.com/articles/mcs/Keil.html>
- [3] นายจักรพล ชุณหพาณิชย์. (2553). ระบบเครือข่ายไร้สาย (Wireless LANs). 11 กุมภาพันธ์ 2556, จาก [http://www.docstoc.com/docs/22710035/\(Wireless-LANs\)](http://www.docstoc.com/docs/22710035/(Wireless-LANs))
- [4] nRF24LE1. 11 กุมภาพันธ์ 2556, จาก <http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24LE1>
- [5] รีโมทคอนโทรล.(2553). 15 กุมภาพันธ์ 2556, จาก <http://thai-maker.com/index.php?topic=22.0>
- [6] หลักการสร้างสัญญาณ PWM. (2550). 20 กุมภาพันธ์ 2556, จาก <http://introduction-pwm.blogspot.com/2007/09/pwm.html>
- [7] ภาษา C กับไมโครคอนโทรลเลอร์. 21 กุมภาพันธ์ 2556, จาก http://www.mwit.ac.th/~cs/download/tech30310/sheet_01_2555.pdf
- [8] นายปรัชญา ศิริภูรี. โครงสร้างของภาษา C.21 กุมภาพันธ์ 2556, จาก http://itd.htc.ac.th/st_it50/it5016/nidz/Web_C/unit2.html
- [9] สุริยา นิมตระกุล. ภาษาคอมพิวเตอร์. 21 กุมภาพันธ์ 2556, จาก http://www.chakkham.ac.th/krusuriya/index.php?option=com_content&view=article&id=97&Itemid=114
- [10] เซิร์ฟเวอร์ทรงพลัง.21 กุมภาพันธ์ 2556, จาก <http://www.scribd.com>
- [11] เทคโนโลยี E-House. 21 พฤษภาคม 2556, จาก ปรินูญานิพนธ์ การควบคุมไฟฟ้าไร้สายภายในบ้าน (Wireless electrical control in house)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก

โปรแกรมที่ใช้ในโครงการ

ในบทนี้จะอธิบายถึงส่วนของโปรแกรมต่างๆที่ใช้ในโครงการซึ่งประกอบด้วย 4 ส่วนคือส่วนโปรแกรมการทำงานของไมโครคอนโทรลเลอร์มาสเตอร์ ส่วนโปรแกรมการทำงานของไมโครคอนโทรลเลอร์สลาฟ ส่วนโปรแกรมการติดต่อระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ผ่านพอร์ตยูอาร์ทีและส่วนโปรแกรมการสร้างเว็บเพจ

ก.1 ส่วนโปรแกรมการทำงานของไมโครคอนโทรลเลอร์มาสเตอร์

ไมโครคอนโทรลเลอร์มาสเตอร์เชื่อมต่ออยู่กับคอมพิวเตอร์ด้วยซีเรียลพอร์ตผ่านทางบอร์ดยูอาร์ทีเมื่อมีการสั่งงานมาจากเว็บเบราว์เซอร์ ข้อมูลจะถูกส่งมายังไมโครคอนโทรลเลอร์และจะมีการประมวลผลคำสั่ง พร้อมกับการสร้างแพ็กเกจข้อมูลเพื่อส่งข้อมูลนั้นไปยังไมโครคอนโทรลเลอร์โดยส่งผ่านระบบไวเลส และสลาฟจะส่งค่ากลับมายังมาสเตอร์เพื่อนำไปแสดงสถานะการทำงานของเครื่องปรับอากาศบนหน้าเว็บเบราว์เซอร์สามารถแสดงได้ดังนี้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 #include<reg24le1.h>
2 #include<stdio.h>
3 #include "main.h"
4 #include "nrf_lob.h"
5 #include "nrf_defines.h"
6
7 #define nRF_CE RFCE
8 #define nRF_CS RFCSEN
9 #define SSPBUF SPIRDAT
10
11 #define Address_Route 0xFF
12
13 unsigned char LOOP, inputdata, Check, Reset_Loop ,Test ,Checksum_Tx = 0;
14 unsigned int Datanet,p;
15 unsigned char Temp,keep;
16 unsigned int Rx_Flag ,Uart_Flag , Decode_in, Decode_out = 0x0000, Send_Decode = 0,x;
17
18
19
20 void Delay_us(volatile unsigned int n){
21     unsigned char i;
22     while(n--){
23         for(i=0;i<3;i++);
24     }
25 }
26 void Delay_100us(volatile unsigned int n){
27     unsigned char i;
28     while(n--){
29         for(i=0;i<35;i++);
30     }
31 }
32 void Delay_ms(int i)
33 {
34     int j;
35     for(j=0;j<1--){
36         for(j=0;j<100;j++);
37     }
38 }
39
40 void UART_Mode() interrupt 4 using 3
41 {
42     if(RID)
43     {
44         keep = (unsigned int)SOBUF;
45         Datanet = keep;
46         RIO = 0;
47         LOOP = 1;
48     }
49     else if(TID)
50     {
51         TIO=0;
52     }
53 }
54 void CE_Pin_Active(unsigned char action) // CE pin high, low or pulse..
55 {
56     switch(action)
57     {
58         case CE_LOW: // action == 0, CE low
59             nRF_CE = 0;
60             break;
61         case CE_HIGH: // action == 1, CE high
62             nRF_CE = 1;
63             break;
64         case CE_PULSE: // action == 2, CE pulse (107s)
65             // THR2 = 0;
66             nRF_CE = 1; // Set CE pin high
67             Delay_us(100);
68             nRF_CE = 0;
69             //THR2CN_bit = 1; // Start Timer2, CE pulse timer
70             break;
71     }
72 }
73 unsigned char LO1_Clear_IRQ(unsigned char irq_flag)
74 {
75     return nRF_SPI_RW_Reg(WRITE_REG + STATUS, irq_flag);
76 }
77
78 unsigned char nRF_SPI_RW(unsigned char byte1)
79 {
80     unsigned char status;
81     //Delay_us(10);
82     SSPBUF = byte1;
83     Delay_us(1); // Delay 300s ok
84     status = SSPBUF;
85     return(status);
86 }
87
88 unsigned char nRF_SPI_RW_Reg(unsigned char reg, unsigned char value)
89 {
90     unsigned char status;
91     nRF_CS = 0; // CSN low, init SPI transaction
92     status = nRF_SPI_RW(reg); // select register
93     nRF_SPI_RW(value); // ..and write value to it..
94     nRF_CS = 1; // CSN high again
95     return(status); // return nRF24L01 status byte
96 }
97
98 unsigned char nRF_SPI_Write_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
99 {
100     unsigned char status, byte_ctr;
101     nRF_CS = 0; // Set CSN low, init SPI transaction
102     status = nRF_SPI_RW(reg); // Select register to write to and read status byte
103     for(byte_ctr=0; byte_ctr<bytes; byte_ctr++) // then write all byte in buffer(*pBuf)
104     {
105         nRF_SPI_RW(*pBuf++);
106     }
107     nRF_CS = 1; // Set CSN high again
108     return(status); // return nRF24L01 status byte
109 }
110
111 unsigned char LO1_RD_RX_PW_n(unsigned char pipe)
112 {
113     return nRF_SPI_Read(RX_PW_PO + pipe);
114 }
115 void LO1_Write_TX_Pload(unsigned char *pBuf, unsigned char plWidth)
116 {
117     nRF_SPI_Write_Buf(WR_TX_PLOAD, pBuf, plWidth);
118 }
119
120
121
122

```

เอกสารนี้เป็นทรัพย์สินทางปัญญาของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี การใช้งานเพื่อการศึกษานั้น ไม่อนุญาติให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ หากมีข้อผิดพลาดประการใดขออภัยและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

147 unsigned char Send_Packet() // Send one data packet, controlled by Button1(BM1)
148 {
149     LO1_Write_TX_Pload(<TX_pload, TX_PLOAD_WIDTH); // Write new TX payload
150     CE_Pin_Active(CE_PULSE); // Enable CE pulse, 12?µs
151     //ucTry_Ctr = 0; // Reset Try_Ctr before nex transmitt..
152     return 1;
153 }
154
155 void LO1_Flush_RX(void)
156 {
157     nRF_SPI_RW_Reg(FLUSH_RX,0);
158 }
159 void LO1_Flush_TX(void)
160 {
161     nRF_SPI_RW_Reg(FLUSH_TX,0);
162 }
163 unsigned char LO1_Get_FIFO(void)
164 {
165     return nRF_SPI_Read(FIFO_STATUS);
166 }
167
168 unsigned char nRF_SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
169 {
170     unsigned char status, byte_ctr;
171     nRF_CS = 0; // Set CSN low, init SPI transaction
172     status = nRF_SPI_RW(reg); // Select register to write to and read status byte
173     for(byte_ctr=0;byte_ctr<bytes;byte_ctr++) // Perform SPI_RW to read byte from nRF24LO1
174     {
175         pBuf[byte_ctr] = nRF_SPI_RW(0);
176     }
177     nRF_CS = 1; // Set CSN high again
178     return(status); // return nRF24LO1 status byte
179 }
180
181 unsigned char nRF_SPI_Read(unsigned char reg)
182 {
183     unsigned char reg_val;
184     nRF_CS = 0; // CSN low, initialize SPI communication...
185     nRF_SPI_RW(reg); // Select register to read from..
186     reg_val = nRF_SPI_RW(0); // ..then read register value
187     nRF_CS = 1; // CSN high, terminate SPI communication
188     return(reg_val); // return register value
189 }
190
191 unsigned char LO1_Get_Status(void)
192 {
193     return nRF_SPI_Read(STATUS);
194 }
195 unsigned char LO1_Get_Current_Pipenum(void)
196 {
197     return ((LO1_Get_Status() < RX_P_NO) >> 1);
198 }
199 unsigned int LO1_Read_RX_Pload(unsigned char *pBuf)
200 {
201     unsigned char pWidth, pipe;
202     pWidth = LO1_RD_RX_P_N_Pipe = LO1_Get_Current_Pipenum(); // Read current pipe's payload width
203     nRF_SPI_Read_Buf(RD_RX_PLOAD, pBuf, pWidth); // Then get RX data
204     return ((pipe << 0) + pWidth); // return pipe# < pipe#.pWidth
205 }
206
207 /*unsigned char nRF_SPI_RW_Reg_IRQ(unsigned char reg, unsigned char value)
208 {
209     unsigned char status;
210     nRF_CS = 0; // CSN low, init SPI transaction
211     SSPIBUF = reg;
212     //while (SSPSTAT.BF == 0)
213     Delay_us(10);
214     status = SSPIBUF; // free the register
215     SSPIBUF = value;
216     //while (SSPSTAT.BF == 0)
217     Delay_us(10);
218     reg = SSPIBUF; // free the register
219     nRF_CS = 1; // CSN high again
220     return(status); // return nRF24LO1 status byte
221 }*/
222
223 void Tx_Mode ()
224 {
225     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
226     Delay_ms(200);
227     LO1_Flush_TX();
228     LO1_Flush_RX();
229     //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
230     nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
231     //ucIRQ_Source = CLEAR;
232     //ucLastStat = ucLinkStat = LINK_ESTABLISH; // LINK_ESTABLISH = 0x02
233     nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack:Pipes 0-5
234     nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3F); // Enable Pipes 0-5
235     nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a); // 500?µs + 86?µs, 10 retrans...
236     nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
237     nRF_SPI_RW_Reg(WRITE_REG + SETUP_AM, 0x03); // Select Address
238     nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08); // TX_PWR:0dBm, DataRate:2Mbps, LNA:HCURR
239     nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
240     nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Writes TX_Address to nRF24LO1
241     nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // RX_Addr0 same as TX_Adr for Auto.Ack
242     CE_Pin_Active(CE_HIGH);
243 }
244
245 /*void Tx_Mode_runtime()
246 {
247     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
248     Delay_ms(200);
249     LO1_Flush_TX();
250     LO1_Flush_RX();
251     LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
252     //nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
253     //ucIRQ_Source = CLEAR;
254     //ucLastStat = ucLinkStat = LINK_ESTABLISH; // LINK_ESTABLISH = 0x02
255     //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack:Pipes 0-5
256     //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3F); // Enable Pipes 0-5
257     //nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a); // 500?µs + 86?µs, 10 retrans...
258     //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
259     //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08); // TX_PWR:0dBm, DataRate:2Mbps, LNA:HCURR
260     //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
261     //nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Writes TX_Address to nRF24LO1
262     //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // RX_Addr0 same as TX_Adr for Auto.Ack
263     CE_Pin_Active(CE_HIGH);
264     Delay_ms(50);
265     //CE_Pin_Active(CE_LOW);
266 }*/

```

เอกสารนี้เป็นทรัพย์สินทางปัญญาของบริษัทฯ ใช้งานเพื่อการศึกษาเท่านั้น ไม่ควรนำไปใช้ประโยชน์ด้านการค้า
 "ไม่ว่า" ลิขสิทธิ์ © 2013 บริษัทฯ

```

255 void Rx_Mode ()
256 {
257     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
258     Delay_ms(1); // 200ms
259     LO1_Flush_TX();
260     LO1_Flush_RX();
261
262     LO1_Clear_IRQ(HASK_IRQ_FLAGS); // Clear interrupts
263
264     nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x31); // Enable Auto.Ack:Pipes 0-5
265     nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x31); // Enable Pipes 0-5
266     nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
267     nRF_SPI_RW_Reg(WRITE_REG + SETUP_AV, 0x03); // Select Address
268     nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x02); // TX_PWR:0dBm, Datarate:2Mbps, LNA:NCURR
269     nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x02); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..
270
271     nRF_SPI_Write_Bur(WRITE_REG + RX_ADDR_P0, &ADDRESS_P0, sizeof(ADDRESS_P0)); // Use the same address on the RX device as the TX device
272     nRF_SPI_Write_Bur(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));
273
274     nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
275     nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
276     nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
277     nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);
278
279     nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P0, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
280     nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_PLOAD_WIDTH);
281     nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_PLOAD_WIDTH);
282     nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_PLOAD_WIDTH);
283     nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_PLOAD_WIDTH);
284     nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_PLOAD_WIDTH);
285
286     CE_Pin_Active(CE_HIGH);
287 }
288
289 void Rx_Mode_Runtime()
290 {
291     //CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
292     //Delay_ms(1); // 200ms
293     //LO1_Flush_TX();
294     //LO1_Flush_RX();
295
296     //LO1_Clear_IRQ(HASK_IRQ_FLAGS); // Clear interrupts
297
298     //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x31); // Enable Auto.Ack:Pipes 0-5
299     //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x31); // Enable Pipes 0-5
300     //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
301     //nRF_SPI_RW_Reg(WRITE_REG + SETUP_AV, 0x03); // Select Address
302     //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x02); // TX_PWR:0dBm, Datarate:2Mbps, LNA:NCURR
303     //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x02); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..
304
305     //nRF_SPI_Write_Bur(WRITE_REG + RX_ADDR_P0, &ADDRESS_P0, sizeof(ADDRESS_P0)); // Use the same address on the RX device as the TX device
306     //nRF_SPI_Write_Bur(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));
307
308     //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
309     //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
310     //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
311     //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);
312
313     //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P0, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
314     //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_PLOAD_WIDTH);
315     //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_PLOAD_WIDTH);
316     //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_PLOAD_WIDTH);
317     //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_PLOAD_WIDTH);
318     //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_PLOAD_WIDTH);
319
320     CE_Pin_Active(CE_HIGH);
321     Delay_ms(30);
322 }
323
324 void Rx_Real_Mode()
325 {
326     Rx_Mode_Runtime();
327     do
328     {
329         //PO0 = ~PO0;
330         uIRX_info = LO1_Read_RX_Pload(RX_pload); // Read current payload
331         // temp = LO1_Get_FIFO() & 0x02; temp >> 1;
332
333     } while(!(LO1_Get_FIFO() < RX_EMPTY)); // ..until FIFO empty
334     //while( temp == 0 );
335     temp = LO1_Get_FIFO() & 0x01; // temp >> 1;
336
337     //P14 = temp;
338
339     //uIRX_info = LO1_Read_RX_Pload(RX_pload);
340     LO1_Clear_IRQ(HASK_RX_DR_FLAG);
341
342     /*if (RX_pload[0] == 0x31) ( PO0 = ~PO0; RX_pload[0] = 0; )
343     if (RX_pload[1] == 0x32) ( PO1 = ~PO1; RX_pload[1] = 0; )
344     if (RX_pload[2] == 0x33) ( PO2 = ~PO2; RX_pload[2] = 0; )
345     if (RX_pload[3] == 0x34) ( PO3 = ~PO3; RX_pload[3] = 0; )
346     if (RX_pload[4] == 0x35) ( PO4 = ~PO4; RX_pload[4] = 0; )*/
347     CE_Pin_Active(CE_LOW);
348 }
349
350 void Tx_Real_Mode()
351 {
352     Send_Packet();
353     P13 = ~P13;
354     Delay_ms(1);
355     LO1_Clear_IRQ(HASK_TX_DS_FLAG);
356 }
357
358 /*void Decode_output ()
359 {
360     int DO;
361     unsigned char Data_All;
362     DO = 3*(((Address_Route>>4) & 9)+1);
363     if ( Send_Decode > Data_All )
364     {
365         Send_Decode = 0;
366     }
367     switch ( Send_Decode )
368     {
369         case 0:
370             break;
371     }
372     Send_Decode++;
373 }*/
374
375 void Decode_input ()
376 {
377     int DI;
378     DI = 3*(((Address_Route>>4) & 9)+1);
379     switch ( (RX_pload[DI] & 0x20) >> 4 )
380     {
381         case 0:
382             //SOBUR = RX_pload[2];
383             while(TIO)
384             {
385                 TIO = 0;
386             }
387             break;
388         case 1:
389             break;
390     }
391 }
392 }

```

เอกสารนี้เป็นเอกสารที่งานวิศวกรรมไฟฟ้าใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ได้รับผิดชอบหากมีการนำเอกสารนี้ไปใช้โดยไม่ได้รับอนุญาต และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

3593 void Check_Device()
3594 {
3595     /* int CD,set,pload;
3596     CD = ((Address_Route>>4) & 9);
3597     if(CD == 0)
3598     {
3599         Rx_Real_Mode();
3600         for(set=3;set<=31;set++)
3601             TX_pload[set] = RX_pload[set];
3602         for(pload=0;pload<=30;pload++)
3603         {
3604             if((pload & 3) == 0)
3605                 TX_pload[pload]= TX_pload[pload] & 0xF0;
3606         }
3607         TX_pload[0] = Address_Route;
3608     }
3609     else if(CD == 1)
3610     {
3611         Rx_Real_Mode();
3612         for(set=0;set<=2;set++)
3613             TX_pload[set] = RX_pload[set];
3614         for(set=6;set<=31;set++)
3615             TX_pload[set] = RX_pload[set];
3616         for(pload=0;pload<=30;pload++)
3617         {
3618             if((pload & 3) == 0)
3619                 TX_pload[pload]= TX_pload[pload] & 0xF0;
3620         }
3621         TX_pload[3] = Address_Route;
3622     }
3623     else if(CD == 2)
3624     {
3625         Rx_Real_Mode();
3626         for(set=0;set<=5;set++)
3627             TX_pload[set] = RX_pload[set];
3628         for(set=9;set<=31;set++)
3629             TX_pload[set] = RX_pload[set];
3630         for(pload=0;pload<=30;pload++)
3631         {
3632             if((pload & 3) == 0)
3633                 TX_pload[pload]= TX_pload[pload] & 0xF0;
3634         }
3635         TX_pload[6] = Address_Route;
3636     }
3637     else if(CD == 3)
3638     {
3639         Rx_Real_Mode();
3640         for(set=0;set<=8;set++)
3641             TX_pload[set] = RX_pload[set];
3642         for(set=12;set<=31;set++)
3643             TX_pload[set] = RX_pload[set];
3644         for(pload=0;pload<=30;pload++)
3645         {
3646             if((pload & 3) == 0)
3647                 TX_pload[pload]= TX_pload[pload] & 0xF0;
3648         }
3649         TX_pload[9] = Address_Route;
3650     }
3651     else if(CD == 4)
3652     {
3653         Rx_Real_Mode();
3654         for(set=0;set<=11;set++)
3655             TX_pload[set] = RX_pload[set];
3656         for(set=15;set<=31;set++)
3657             TX_pload[set] = RX_pload[set];
3658         for(pload=0;pload<=30;pload++)
3659         {
3660             if((pload & 3) == 0)
3661                 TX_pload[pload]= TX_pload[pload] & 0xF0;
3662         }
3663         TX_pload[12] = Address_Route;
3664     }
3665     else if(CD == 5)
3666     {
3667         Rx_Real_Mode();
3668         for(set=0;set<=14;set++)
3669             TX_pload[set] = RX_pload[set];
3670         for(set=18;set<=31;set++)
3671             TX_pload[set] = RX_pload[set];
3672         for(pload=0;pload<=30;pload++)
3673         {
3674             if((pload & 3) == 0)
3675                 TX_pload[pload]= TX_pload[pload] & 0xF0;
3676         }
3677         TX_pload[15] = Address_Route;
3678     }
3679     else if(CD == 6)
3680     {
3681         Rx_Real_Mode();
3682         for(set=0;set<=17;set++)
3683             TX_pload[set] = RX_pload[set];
3684         for(set=21;set<=31;set++)
3685             TX_pload[set] = RX_pload[set];
3686         for(pload=0;pload<=30;pload++)
3687         {
3688             if((pload & 3) == 0)
3689                 TX_pload[pload]= TX_pload[pload] & 0xF0;
3690         }
3691         TX_pload[18] = Address_Route;
3692     }
3693     else if(CD == 7)
3694     {
3695         Rx_Real_Mode();
3696         for(set=0;set<=20;set++)
3697             TX_pload[set] = RX_pload[set];
3698         for(set=24;set<=31;set++)
3699             TX_pload[set] = RX_pload[set];
3700         for(pload=0;pload<=30;pload++)
3701         {
3702             if((pload & 3) == 0)
3703                 TX_pload[pload]= TX_pload[pload] & 0xF0;
3704         }
3705         TX_pload[21] = Address_Route;
3706     }
3707     else if(CD == 8)
3708     {
3709         Rx_Real_Mode();
3710         for(set=0;set<=23;set++)
3711             TX_pload[set] = RX_pload[set];
3712         for(set=27;set<=31;set++)
3713             TX_pload[set] = RX_pload[set];
3714         for(pload=0;pload<=30;pload++)
3715         {
3716             if((pload & 3) == 0)
3717                 TX_pload[pload]= TX_pload[pload] & 0xF0;
3718         }
3719         TX_pload[24] = Address_Route;
3720     }
3721     /*
3722     Rx_Real_Mode();
3723     CE_Pin_Active(CE_LOU);
3724     }

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งการนำเอกสารนี้ไปเผยแพร่โดยไม่ได้รับอนุญาตจากทางมหาวิทยาลัยฯ อาจส่งผลถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

549 void Read_Data ()
550 {
551     int set;
552     /*int SD;
553     SD = ((Address_Route>>4) & 9);
554     if(SD == 0)
555     {
556         if(((Address_Route & 0x0f) == (RX_pload[0] & 0x0f)) && (((Address_Route & 0x0f)+2) == (RX_pload[6] & 0x0f)) || (((Address_Route & 0x0f)+15) == (RX_pload[30] & 0x0f)))
557         {
558             Decode_input();
559             /*if(RX_pload[2] == 0x01)
560             { Delay_ms(100); P00 = -P00; Check = 1; }
561             //P00 = -P00;
562             Check = 1;
563         }
564         if(RX_pload[6] == 0x00)
565         {
566             if(((Address_Route & 0x0f) == (RX_pload[0] & 0x0f)) && (((Address_Route & 0x0f)+1) == (RX_pload[3] & 0x0f)) || (((Address_Route & 0x0f)+15) == (RX_pload[30] & 0
567             //P02 = -P02;
568             Decode_input();
569             Check = 1;
570         }
571     }
572     else if(SD == 1)
573     {
574         if(((Address_Route & 0x0f) == (RX_pload[3] & 0x0f)) && (((Address_Route & 0x0f)-1) == (RX_pload[0] & 0x0f))
575         //if(((Address_Route>>4) == (RX_pload[3]>>4)) && (((Address_Route&0x0f)-1) && (RX_pload[0]&0x0f))
576         {
577             Decode_input();
578             /*if(RX_pload[5] == 0x01)
579             { Delay_ms(100); P00 = -P00; Check = 1; }
580             Check = 1;
581         }
582     }
583     else if(SD == 2)
584     {
585         if(((Address_Route & 0x0f) && (RX_pload[6] & 0x0f)) && (((Address_Route & 0x0f)-1) == (RX_pload[3] & 0x0f))
586         {
587             Decode_input();
588             /*if(RX_pload[8] == 0x01)
589             { Delay_ms(100); P00 = -P00; Check = 1; }
590             if (TX_pload[7] == 0x00) P00 = -P00;
591             if (TX_pload[8] == 0x01) P01 = -P01;
592             Check = 1;
593         }
594     }
595     else if(SD == 3)
596     {
597         if(((Address_Route & 0x0f) && (RX_pload[9] & 0x0f)) && (((Address_Route & 0x0f)+2) == (RX_pload[15] & 0x0f)) || (((Address_Route & 0x0f)-2) == (RX_pload[3] & 0x0f)
598         {
599             if(RX_pload[11] == 0x01)
600             { Delay_ms(100); P00 = -P00; Check = 1; }
601         }
602         if(RX_pload[15] == 0x00)
603         {
604             if(((Address_Route & 0x0f) && (RX_pload[9] & 0x0f)) && (((Address_Route & 0x0f)+1) == (RX_pload[12] & 0x0f)) || (((Address_Route & 0x0f)-2) == (RX_pload[3] & 0x
605             P02 = -P02;
606         }
607     }
608     else if(SD == 4)
609     {
610         if(((Address_Route & 0x0f) && (RX_pload[12] & 0x0f)) && (((Address_Route & 0x0f)-1) == (RX_pload[6] & 0x0f))
611         {
612             if(RX_pload[14] == 0x01)
613             { Delay_ms(100); P00 = -P00; Check = 1; }
614         }
615     }
616     else if(SD == 5)
617     {
618         if(((Address_Route & 0x0f) && (RX_pload[15] & 0x0f)) && (((Address_Route & 0x0f)-1) == (RX_pload[12] & 0x0f))
619         {
620             if(RX_pload[17] == 0x01)
621             { Delay_ms(100); P00 = -P00; Check = 1; }
622         }
623     }
624     else if(SD == 6)
625     {
626         if(((Address_Route & 0x0f) && (RX_pload[18] & 0x0f)) && (((Address_Route & 0x0f)+2) == (RX_pload[24] & 0x0f)) || (((Address_Route & 0x0f)-3) == (RX_pload[6] & 0x0
627         {
628             if(RX_pload[20] == 0x01)
629             { Delay_ms(100); P00 = -P00; Check = 1; }
630         }
631         if(RX_pload[24] == 0x00)
632         {
633             if(((Address_Route & 0x0f) && (RX_pload[9] & 0x0f)) && (((Address_Route & 0x0f)+1) == (RX_pload[21] & 0x0f)) || (((Address_Route & 0x0f)-3) == (RX_pload[2] & 0x
634             P02 = -P02;
635         }
636     }
637     else if(SD == 7)
638     {
639         if(((Address_Route & 0x0f) && (RX_pload[21] & 0x0f)) && (((Address_Route & 0x0f)-1) == (RX_pload[18] & 0x0f))
640         {
641             if(RX_pload[23] == 0x01)
642             { Delay_ms(100); P00 = -P00; Check = 1; }
643         }
644     }
645     else if(SD == 8)
646     {
647         if(((Address_Route & 0x0f) && (RX_pload[24] & 0x0f)) && (((Address_Route & 0x0f)-1) == (RX_pload[21] & 0x0f))
648         {
649             if(RX_pload[26] == 0x01)
650             { Delay_ms(100); P00 = -P00; Check = 1; }
651         }
652     }
653     //P02 = -P02;
654     if(((Address_Route&0x0f) && (RX_pload[30]&0x0f)) && (((Address_Route&0x0f)-15) == (RX_pload[0]&0x0f))
655     {
656         for(set=0;set<=3;set++)
657         {
658             x = RX_pload[set];
659         }
660     }
661 }
662 }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่มีการนำเอกสารนี้ไปเผยแพร่หรือทำซ้ำโดยไม่ได้รับอนุญาต และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

663 void RF_Mode() interrupt 9 using 1
664 {
665     RF = 0;
666     PO7 = ~PO7;
667     Rx_flag = 1;
668     RF = 1;
669 }
670 void main()
671 {
672     PODIR = 0x10;
673     PIDIR = 0x00;
674
675     SOCON = 0x50;           //Mode 1 8-bit UART.
676     SOBUF = 0x00;
677     SORELH = 0x03;
678     SORELL = 0xF3;
679     ADCON = 0x80;
680
681     PCON = PCON| 0x80;
682
683
684     nRF_CE = 0;
685     nRF_CS = 1;
686
687     RFCKEN = 1;
688     Delay_ms(500);
689
690     CE_Pin_Active(CE_LOW);           // Set CE pin low to enable standby mc
691     Delay_ms(200);
692     LOI_Flush_TX();
693     LOI_Flush_RX();
694
695     /*for (p=0;p<32;p++)
696     {
697         SOBUF = 'A';
698         Delay_ms(1);
699         while(TIO);
700         TIO = 0 ;
701     }*/
702     //printf("Hello");
703     Delay_ms(500);
704
705     Rx_Mode();
706     RF = 1;
707     ESO = 0;
708     RIO = 0;
709     TIO = 1;
710     EA = 1;
711
712
713
714
715
716 while(1)
717 {
718     if(RIO == 1)
719     {
720         P12 = ~P12;
721         keep = (unsigned int)SOBUF;
722         Datanet = keep;
723         RIO = 0;
724         LOOP = 1;
725         // printf("%c\r\n",keep);
726     }
727
728     if(LOOP == 1)
729     {
730         PO6 = ~PO6;
731
732         //Decode_output();
733         TX_pload[0] = 0x00;
734         TX_pload[1] = 0x00;
735         TX_pload[2] = Datanet;
736         for (p=3;p<32;p++)
737         {
738             TX_pload[p] = 0x00;
739         }
740         TX_pload[30] = 0xFF;
741
742         Tx_Mode();
743         Tx_Real_Mode();
744         PO1 = ~PO1;
745
746         //PO6 = ~PO6;
747         /* for (p=0;p<32;p++)
748         {
749             printf("%c",TX_pload[p]);
750         }*/
751         LOOP = 0;
752         Delay_ms(100);
753     }
754
755     if(Rx_flag == 1)
756     {
757         PO2 = ~PO2;
758         Check_Device();
759         Read_Data();
760         //CE_Pin_Active(CE_HIGH);
761         Rx_flag = 0;
762
763         /* for (p=0;p<32;p++)
764         {
765             printf("%c",RX_pload[p]);
766         }*/
767     }
768
769     Rx_Mode();
770     Delay_ms(10);
771 }
772 }

```

การนี้เป็นเอกสารที่จัดทำขึ้นเพื่อให้บริการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 การณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ก.2 ส่วนโปรแกรมการทำงานของไมโครคอนโทรลเลอร์สถาฟ

ไมโครคอนโทรลเลอร์สถาฟจะเชื่อมต่อกับวงจรส่งสัญญาณอินฟราเรดเพื่อส่งสัญญาณไปควบคุมเครื่องปรับอากาศ โดยรับคำสั่งมาจากมาสเตอร์ในรูปแบบของแพคเกจข้อมูล จากนั้นไมโครคอนโทรลเลอร์จะทำการประมวลผล ถอดคำสั่งการทำงานและส่งสัญญาณอินฟราเรดไปสั่งงานเครื่องปรับอากาศตามคำสั่งของผู้ใช้งานพร้อมกับการส่งข้อมูลกลับไปยังมาสเตอร์เพื่อแสดงสถานะของคำสั่งที่ได้สั่งงานเครื่องปรับอากาศไปแล้ว และที่ตัวสถาฟนี้ยังมีอินฟราเรดรีซีฟเวอร์รับสัญญาณรีโมตปกติที่ใช้สั่งงานเครื่องปรับอากาศ เพื่อส่งคำสั่งนั้นไปแสดงสถานะการทำงานที่หน้าเว็บเบราว์เซอร์ได้เช่นเดียวกัน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


```

1426 //printf(" Part2\r\n");
1427 out = 0; k=0; j=0;
1428 for( Loop_Print =1;Loop_Print<153;Loop_Print++){
1429     out = out << 1;
1430     if ( THH2[Loop_Print] == 0x4 ) ( data1 = 0;out1 = 0; )
1431     if ( THH3[Loop_Print] == 0x8 ) ( data1 = 1;out1 = 1; )
1432     out = out | data1; //printf("%u ",out1);
1433
1434     j++;
1435     if ( j== 8 ) (
1436         j =0;
1437         Byte_out2[k] = out;// printf(" \r\n");
1438         out = 0;
1439         k++;
1440     )
1441
1442 }
1443
1444 State = 0;State2 = 0;
1445
1446 )
1447
1448 void Delay_us(unsigned int i)
1449 {
1450     TH1 = (0xffff-1)>>8;
1451     TL1 = (0xffff-1);
1452     TR1 =1;
1453     timer_1time = 1;
1454     while(timer_1time);
1455 }
1456 void timer1() interrupt 3 using 2
1457 {
1458     TF1=0;
1459     TR1 = 0;
1460     timer_1time = 0;
1461 }
1462
1463 void Delay_us(unsigned int n){
1464     unsigned char i;
1465     while(n--);
1466     for(i=0;i<3;i++);
1467 }
1468 void Delay_100us(unsigned int n){
1469     unsigned char i;
1470     while(n--);
1471     for(i=0;i<35;i++);
1472 }
1473
1474 void Delay_ms(unsigned int n){
1475     while(n--);
1476     Delay_100us(50);
1477 }
1478
1479 void RF_Mode() interrupt 9 using 1
1480 {
1481     RF = 0;
1482     //PO7 = ~PO7;
1483     Rx_flag = 1;
1484     RF = 1;
1485 }
1486
1487 void Duty_50()
1488 {
1489     PWMDCO = 0x02;
1490 }
1491 void Duty_0()
1492 {
1493     PWMDCO = 0x00;
1494 }
1495 void IR_Pulse_Head()
1496 {
1497     Duty_50();
1498     Delay_us(Delayup);
1499     Duty_0();
1500     Delay_us(Delayup2);
1501 }
1502 void IR_Pulse_Tail()
1503 {
1504     Duty_50();
1505     Delay_us(Delayup4);
1506     Duty_0();
1507     Delay_us(Delayup6);
1508 }
1509
1510 void IR_Pulse_1()
1511 {
1512     Duty_50();
1513     Delay_us(Delayup4);
1514     Duty_0();
1515     Delay_us(Delayup3);
1516     Duty_50();
1517 }
1518
1519 void Ir_Pulse_0()
1520 {
1521     Duty_50();
1522     Delay_us(Delayup4);
1523     Duty_0();
1524     Delay_us(Delayup5);
1525     Duty_50();
1526 }
1527
1528 void Out_Ir_Remote(char data_Ir)
1529 {
1530     unsigned char Bit_data;
1531
1532     Bit_data = data_Ir & 0x80;
1533     if (Bit_data == 0x80) (Ir_Pulse_1(); )
1534     else ( Ir_Pulse_0(); )
1535
1536     Bit_data = data_Ir & 0x40;
1537     if (Bit_data == 0x40) (Ir_Pulse_1(); )
1538     else (Ir_Pulse_0(); )
1539
1540     Bit_data = data_Ir & 0x20;
1541     if (Bit_data == 0x20) (Ir_Pulse_1(); )
1542     else (Ir_Pulse_0(); )
1543
1544     Bit_data = data_Ir & 0x10;
1545     if (Bit_data == 0x10) (Ir_Pulse_1(); )
1546     else (Ir_Pulse_0(); )
1547
1548     Bit_data = data_Ir & 0x08;
1549     if (Bit_data == 0x08) (Ir_Pulse_1(); )
1550     else (Ir_Pulse_0(); )
1551
1552     Bit_data = data_Ir & 0x04;
1553     if (Bit_data == 0x04) (Ir_Pulse_1(); )
1554     else (Ir_Pulse_0(); )
1555
1556     Bit_data = data_Ir & 0x02;
1557     if (Bit_data == 0x02) (Ir_Pulse_1(); )
1558     else (Ir_Pulse_0(); )
1559
1560     Bit_data = data_Ir & 0x01;
1561     if (Bit_data == 0x01) (Ir_Pulse_1(); )
1562     else (Ir_Pulse_0(); )
1563 }
1564
1565 )

```

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น หากมีข้อสงสัยหรือต้องการข้อมูลเพิ่มเติม และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

266 void Sent_IR_Bf(unsigned char type1[], unsigned char type2[])
267 {
268     unsigned int ii;
269     P15 = 1;
270     Ir_Pulse_Head();
271
272     for (ii=0; ii<8; ii++) {
273         Out_Ir_Remote(type1[ii]);
274     }
275     Ir_Pulse_Tail();
276     Ir_Pulse_Head();
277     for (ii=0; ii<19; ii++) {
278         Out_Ir_Remote(type2[ii]);
279     }
280     Ir_Pulse_Tail();
281     Delay_ms(1000);
282     P15 = 0;
283 }
284
285
286 void CE_Pin_Active(unsigned char action) // CE pin high, low or pulse..
287 {
288     switch(action)
289     {
290         case CE_LOW: // action == 0, CE low
291             nRF_CE = 0;
292             break;
293         case CE_HIGH: // action == 1, CE high
294             nRF_CE = 1;
295             break;
296         case CE_PULSE: // action == 2, CE pulse (10?µs)
297             // TMR2 = 0;
298             nRF_CE = 1; // Set CE pin high
299             Delay_us(100);
300             nRF_CE = 0;
301             // TMR2ON_bit = 1; // Start Timer2, CE pulse timer
302             break;
303     }
304 }
305
306 unsigned char LO1_Clear_IRQ(unsigned char irq_flag)
307 {
308     return nRF_SPI_RW_Reg(WRITE_REG + STATUS, irq_flag);
309 }
310
311 unsigned char nRF_SPI_RW(unsigned char byte1)
312 {
313     unsigned char status;
314     // Delay_us(10);
315     SSP1BUF = byte1;
316     Delay_us(1); // Delay 300µs ok
317     status = SSP1BUF;
318     return(status);
319 }
320
321 unsigned char nRF_SPI_RW_Reg(unsigned char reg, unsigned char value)
322 {
323     unsigned char status;
324
325     nRF_CS = 0; // CSN low, init SPI transaction
326     status = nRF_SPI_RW(reg); // select register
327     nRF_SPI_RW(value); // ..and write value to it..
328     nRF_CS = 1; // CSN high again
329
330     return(status); // return nRF24L01 status byte
331 }
332
333 unsigned char nRF_SPI_Write_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
334 {
335     unsigned char status, byte_ctr;
336
337     nRF_CS = 0; // Set CSN low, init SPI transaction
338     status = nRF_SPI_RW(reg); // Select register to write to and read status byte
339
340     for(byte_ctr=0; byte_ctr<bytes; byte_ctr++) // then write all byte in buffer(*pBuf)
341     {
342         nRF_SPI_RW(*pBuf++);
343     }
344
345     nRF_CS = 1; // Set CSN high again
346
347     return(status); // return nRF24L01 status byte
348 }
349
350 unsigned char LO1_RD_RX_PW_n(unsigned char pipe)
351 {
352     return nRF_SPI_Read(RX_PW_PO + pipe);
353 }
354
355 void LO1_Write_TX_Pload(unsigned char *pBuf, unsigned char plWidth)
356 {
357     nRF_SPI_Write_Buf(WR_TX_FLOAD, pBuf, plWidth);
358 }
359
360 unsigned char Send_Packet() // Send one data packet, controlled by Button1(SW1)
361 {
362     LO1_Write_TX_Pload(&TX_pload, TX_FLOAD_WIDTH); // Write new TX payload
363     CE_Pin_Active(CE_PULSE); // Enable CE pulse, 12?µs
364     //ucTry_Ctr = 0; // Reset Try_Ctr before nex transmitt..
365     return 1;
366 }
367
368 void LO1_Flush_RX(void)
369 {
370     nRF_SPI_RW_Reg(FLUSH_RX, 0);
371 }
372
373 void LO1_Flush_TX(void)
374 {
375     nRF_SPI_RW_Reg(FLUSH_TX, 0);
376 }
377
378 unsigned char LO1_Get_FIFO(void)
379 {
380     return nRF_SPI_Read(FIFO_STATUS);
381 }
382
383 unsigned char nRF_SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
384 {
385     unsigned char status, byte_ctr;
386 }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่สามารถเข้าถึงอื่น ยกเว้นกรณีเกิดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

392 nRF_CS = 0; // Set CSN low, init SPI transaction
393 status = nRF_SPI_RW(reg); // Select register to write to and read status byte
394
395 for(byte_ctr=0;byte_ctr<bytes;byte_ctr++) // Perform SPI_RW to read byte from nRF24L01
396 {
397     pBuf[byte_ctr] = nRF_SPI_RW(0);
398 }
399 nRF_CS = 1; // Set CSN high again
400
401 return(status); // return nRF24L01 status byte
402
403 unsigned char nRF_SPI_Read(unsigned char reg)
404 {
405     unsigned char reg_val;
406
407     nRF_CS = 0; // CSN low, initialise SPI communication...
408     nRF_SPI_RW(reg); // Select register to read from..
409     reg_val = nRF_SPI_RW(0); // ..then read register value
410     nRF_CS = 1; // CSN high, terminate SPI communication
411
412     return(reg_val); // return register value
413 }
414
415 unsigned char L01_Get_Status(void)
416 {
417     return nRF_SPI_Read(STATUS);
418 }
419
420 unsigned char L01_Get_Current_Pipenum(void)
421 {
422     return ((L01_Get_Status() & RX_P_NO) >> 1);
423 }
424
425 unsigned int L01_Read_RX_Pload(unsigned char *pBuf)
426 {
427     unsigned char piWidth, pipe;
428     piWidth = L01_RD_RX_PU_n(pipe = L01_Get_Current_Pipenum()); // Read current pipe's payload width
429     nRF_SPI_Read_Buf(RD_RX_PLOAD, pBuf, piWidth); // Then get RX data
430
431     return ((pipe << 8) + piWidth); // return pipe# & pipe#.piWidth
432 }
433
434 /*unsigned char nRF_SPI_RW_Reg_IRQ(unsigned char reg, unsigned char value)
435 {
436     unsigned char status;
437
438     nRF_CS = 0; // CSN low, init SPI transaction
439
440     SSPIBUF = reg;
441     //while (SSP1STAT.BF == 0)
442     Delay_us(10);
443     status = SSPIBUF; // free the register
444
445     SSPIBUF = value;
446     //while (SSP1STAT.BF == 0)
447     Delay_us(10);
448     reg = SSPIBUF; // free the register
449
450     nRF_CS = 1; // CSN high again
451
452     return(status); // return nRF24L01 status byte
453 }
454 */
455 void Tx_Mode ()
456 {
457     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
458     Delay_ms(200);
459     L01_Flush_TX();
460     L01_Flush_RX();
461     //L01_Clear_IRQ(NASK_IRQ_FLAGS); // Clear interrupts
462     nRF_SPI_RW_Reg(WRITE_REG + STATUS, NASK_IRQ_FLAGS);
463     //ucIRQ_Src = CLEAR; // LINK_ESTABLISH = 0x02
464     //ucLscStat = ucLinkStat = LINK_ESTABLISH; // LINK_ESTABLISH = 0x02
465
466     //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x32); // Enable Auto.Ack/Pipes 0-5
467     nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x32); // Enable Pipes 0-5
468     nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a); // 500's + 867's, 10 retrans...
469     nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
470     nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03); // Select Address
471     nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08); // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
472     nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX, MAX_RT & TX_DS enabled..
473
474     nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Writes TX_Address to nRF24L01
475     nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, &ADDRESS_PO, sizeof(ADDRESS_PO)); // RX_Addr0 same as TX_Adr for Auto.Ack
476     CE_Pin_Active(CE_HIGH);
477 }
478
479 /*void Tx_Mode_runtime ()
480 {
481     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
482     //Delay_ms(200);
483     L01_Flush_TX();
484     L01_Flush_RX();
485     L01_Clear_IRQ(NASK_IRQ_FLAGS); // Clear interrupts
486     //nRF_SPI_RW_Reg(WRITE_REG + STATUS, NASK_IRQ_FLAGS);
487     //ucIRQ_Src = CLEAR; // LINK_ESTABLISH = 0x02
488     //ucLscStat = ucLinkStat = LINK_ESTABLISH; // LINK_ESTABLISH = 0x02
489
490     //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x32); // Enable Auto.Ack/Pipes 0-5
491     //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x32); // Enable Pipes 0-5
492     //nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a); // 500's + 867's, 10 retrans...
493     //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
494
495     //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08); // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
496     //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX, MAX_RT & TX_DS enabled..
497
498     //nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Writes TX_Address to nRF24L01
499     //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, &ADDRESS_PO, sizeof(ADDRESS_PO)); // RX_Addr0 same as TX_Adr for Auto.Ack
500     CE_Pin_Active(CE_HIGH);
501     Delay_ms(50); //max 1 ms
502 }
503 */
504 void Rx_Mode ()
505 {
506     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
507     //Delay_ms(1); // 100ms
508     L01_Flush_TX();
509     L01_Flush_RX();
510
511     L01_Clear_IRQ(NASK_IRQ_FLAGS); // Clear interrupts
512
513     nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x32); // Enable Auto.Ack/Pipes 0-5
514     nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x32); // Enable Pipes 0-5
515     nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
516     nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03); // Select Address
517     nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x02); // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
518     nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x02); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX, RX_DR enabled..
519
520     nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Use the same address on the RX device as the TX device
521     nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));
522
523     nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
524     nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
525     nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
526     nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);
527
528     nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P0, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
529     nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P1, RX_PLOAD_WIDTH);
530     nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P2, RX_PLOAD_WIDTH);
531     nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P3, RX_PLOAD_WIDTH);
532     nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P4, RX_PLOAD_WIDTH);
533     nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P5, RX_PLOAD_WIDTH);
534
535     CE_Pin_Active(CE_HIGH);
536 }

```

เอกสารนี้เป็นเอกสารเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆก็ตาม ต้องขออนุญาตเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

824 void Rx_Mode_Runtime()
825 {
826     //CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
827     //Delay_ms(1); // 200ms
828     //LO1_Flush_TX();
829     //LO1_Flush_RX();
830
831     //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
832
833     //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack/Pipes 0-5
834     //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3E); // Enable Pipes 0-5
835     //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
836     //nRF_SPI_RW_Reg(WRITE_REG + SETUP_AU, 0x03); // Select Address
837     //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x0F); // TX_PMR:0dBm, Datarate:2Mbps, LNA:HCURR
838     //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0E); // Set PMR_UP bit, enable CRC(2 bytes) & Prim:RX, RX_DR enabled..
839
840     //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, &ADDRESS_P0, sizeof(ADDRESS_P0)); // Use the same address on the RX device as the TX devic
841     //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));
842
843     //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
844     //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
845     //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
846     //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);
847
848     //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P0, RX_PAYLOAD_WIDTH); // Select same RX payload width as TX Payload width
849     //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P1, RX_PAYLOAD_WIDTH);
850     //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P2, RX_PAYLOAD_WIDTH);
851     //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P3, RX_PAYLOAD_WIDTH);
852     //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P4, RX_PAYLOAD_WIDTH);
853     //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P5, RX_PAYLOAD_WIDTH);
854
855     CE_Pin_Active(CE_HIGH);
856     Delay_ms(30);
857 }
858
859 void Rx_Real_Mode()
860 {
861     Rx_Mode_Runtime();
862     do
863     {
864         //POG = ~POG;
865         u1RX_info = LO1_Read_RX_Pload(RX_pload); // Read current payload
866         // temp = LO1_Get_FIFO() & 0x02; temp >> 1;
867
868     } while(!(LO1_Get_FIFO() & RX_EMPTY)); // ..until FIFO empty
869     //while( temp == 0 );
870     temp = LO1_Get_FIFO() & 0x01; // temp >> 1;
871
872     //P14 = temp;
873     //u1RX_info = LO1_Read_RX_Pload();
874     LO1_Clear_IRQ(MASK_RX_DR_FLAG);
875
876     //if (RX_pload[0] == 0x11) { P00 = ~P00; RX_pload[0] = 0; }
877     if (RX_pload[1] == 0x32) { P01 = ~P01; RX_pload[1] = 0; }
878     if (RX_pload[2] == 0x33) { P02 = ~P02; RX_pload[2] = 0; }
879     if (RX_pload[3] == 0x34) { P03 = ~P03; RX_pload[3] = 0; }
880     if (RX_pload[4] == 0x35) { P04 = ~P04; RX_pload[4] = 0; } //
881     CE_Pin_Active(CE_LOW);
882 }
883
884 void Tx_Real_Mode()
885 {
886     Send_Packet();
887     Delay_ms(1);
888     LO1_Clear_IRQ(MASK_TX_DS_FLAG);
889 }
890
891 void Decode_output()
892 {
893     int a;
894     for(a = 0;a<32;a++) {
895         TX_pload[a] = RX_pload[a];
896     }
897 }
898
899 void Decode_input()
900 {
901     int DI;
902     DI = (3*((Address_Route>>4) & 9))+1;
903
904     switch ( (RX_pload[DI] & 0x20) >> 4 )
905     {
906     case 0:
907         switch (RX_pload[3]) {
908         case 0x41:
909             Sent_IR_Bf(Close1,Close2);
910             break;
911         case 0x42:
912             Sent_IR_Bf(Open1,Open2);
913             break;
914         case 0x43:
915             Sent_IR_Bf(T30_1,T30_2);
916             break;
917         case 0x44:
918             Sent_IR_Bf(T29_1,T29_2);
919             break;
920         case 0x45:
921             Sent_IR_Bf(T28_1,T28_2);
922             break;
923         case 0x46:
924             Sent_IR_Bf(T27_1,T27_2);
925             break;
926         case 0x47:
927             Sent_IR_Bf(T26_1,T26_2);
928             break;
929         case 0x48:
930             Sent_IR_Bf(T25_1,T25_2);
931             break;
932         case 0x49:
933             Sent_IR_Bf(T24_1,T24_2);
934             break;
935         case 0x4a:
936             Sent_IR_Bf(T23_1,T23_2);
937             break;
938         case 0x4b:
939             Sent_IR_Bf(T22_1,T22_2);
940             break;
941         case 0x4c:
942             Sent_IR_Bf(T21_1,T21_2);
943             break;
944         case 0x4d:
945             Sent_IR_Bf(T20_1,T20_2);
946             break;
947         case 0x4e:
948             Sent_IR_Bf(T19_1,T19_2);
949             break;
950         case 0x4f:
951             Sent_IR_Bf(T18_1,T18_2);
952             break;
953         case 0x50:
954             Sent_IR_Bf(T17_1,T17_2);
955             break;
956         case 0x51:
957             Sent_IR_Bf(T16_1,T16_2);
958             break;
959         }
960     }
961     break;
962 }
963 }
964 }
965 }

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อการใช้งานเพื่อการศึกษเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่าในกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

664 void Check_Device()
665 {
666     int CD, set, pload;
667     CD = ((Address_Route >> 4) & 9);
668     if (CD == 0)
669     {
670         RK_Real_Mode();
671         for (set=3; set<=31; set++)
672             TX_pload[set] = RX_pload[set];
673         for (pload=0; pload<=30; pload++)
674         {
675             if ((pload & 3) == 0)
676                 TX_pload[pload] = TX_pload[pload] & 0xF0;
677             TX_pload[0] = Address_Route;
678         }
679     }
680     else if (CD == 1)
681     {
682         RK_Real_Mode();
683         for (set=0; set<=2; set++)
684             TX_pload[set] = RX_pload[set];
685         for (set=6; set<=31; set++)
686             TX_pload[set] = RX_pload[set];
687         for (pload=0; pload<=30; pload++)
688         {
689             if ((pload & 3) == 0)
690                 TX_pload[pload] = TX_pload[pload] & 0xF0;
691             TX_pload[3] = Address_Route;
692         }
693     }
694     else if (CD == 2)
695     {
696         RK_Real_Mode();
697         for (set=0; set<=5; set++)
698             TX_pload[set] = RX_pload[set];
699         for (set=9; set<=31; set++)
700             TX_pload[set] = RX_pload[set];
701         for (pload=0; pload<=30; pload++)
702         {
703             if ((pload & 3) == 0)
704                 TX_pload[pload] = TX_pload[pload] & 0xF0;
705             TX_pload[6] = Address_Route;
706         }
707     }
708     else if (CD == 3)
709     {
710         RK_Real_Mode();
711         for (set=0; set<=8; set++)
712             TX_pload[set] = RX_pload[set];
713         for (set=12; set<=31; set++)
714             TX_pload[set] = RX_pload[set];
715         for (pload=0; pload<=30; pload++)
716         {
717             if ((pload & 3) == 0)
718                 TX_pload[pload] = TX_pload[pload] & 0xF0;
719             TX_pload[9] = Address_Route;
720         }
721     }
722     else if (CD == 4)
723     {
724         RK_Real_Mode();
725         for (set=0; set<=11; set++)
726             TX_pload[set] = RX_pload[set];
727         for (set=15; set<=31; set++)
728             TX_pload[set] = RX_pload[set];
729         for (pload=0; pload<=30; pload++)
730         {
731             if ((pload & 3) == 0)
732                 TX_pload[pload] = TX_pload[pload] & 0xF0;
733             TX_pload[12] = Address_Route;
734         }
735     }
736     else if (CD == 5)
737     {
738         RK_Real_Mode();
739         for (set=0; set<=14; set++)
740             TX_pload[set] = RX_pload[set];
741         for (set=18; set<=31; set++)
742             TX_pload[set] = RX_pload[set];
743         for (pload=0; pload<=30; pload++)
744         {
745             if ((pload & 3) == 0)
746                 TX_pload[pload] = TX_pload[pload] & 0xF0;
747             TX_pload[15] = Address_Route;
748         }
749     }
750     else if (CD == 6)
751     {
752         RK_Real_Mode();
753         for (set=0; set<=17; set++)
754             TX_pload[set] = RX_pload[set];
755         for (set=21; set<=31; set++)
756             TX_pload[set] = RX_pload[set];
757         for (pload=0; pload<=30; pload++)
758         {
759             if ((pload & 3) == 0)
760                 TX_pload[pload] = TX_pload[pload] & 0xF0;
761             TX_pload[18] = Address_Route;
762         }
763     }
764     else if (CD == 7)
765     {
766         RK_Real_Mode();
767         for (set=0; set<=20; set++)
768             TX_pload[set] = RX_pload[set];
769         for (set=24; set<=31; set++)
770             TX_pload[set] = RX_pload[set];
771         for (pload=0; pload<=30; pload++)
772         {
773             if ((pload & 3) == 0)
774                 TX_pload[pload] = TX_pload[pload] & 0xF0;
775             TX_pload[21] = Address_Route;
776         }
777     }
778     else if (CD == 8)
779     {
780         RK_Real_Mode();
781         for (set=0; set<=23; set++)
782             TX_pload[set] = RX_pload[set];
783         for (set=27; set<=31; set++)
784             TX_pload[set] = RX_pload[set];
785         for (pload=0; pload<=30; pload++)
786         {
787             if ((pload & 3) == 0)
788                 TX_pload[pload] = TX_pload[pload] & 0xF0;
789             TX_pload[24] = Address_Route;
790         }
791     }
792     CE_Pin_Active(CE_LOU);
793 }

```

เอกสารนี้เป็นเอกสารที่... เอกสารที่... คือการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

820 void Read_Data ()
821 {
822     int SD;
823     SD = ((Address_Route>>4) & 9);
824     if(SD == 0)
825     {
826         if(((Address_Route & 0x20) == (RX_pload[0] & 0x20) && (((Address_Route & 0x02)+2) == (RX_pload[6] & 0x02)) || (((Address_Route & 0x02)+15) == (RX_pload[30] & 0x02)))
827         {
828             Decode_input();
829             /*if(RX_pload[2] == 0x01)
830             {
831                 Delay_ms(100); P00 = -P00; Check = 1; }*/
832             //P00 = -P00;
833             Check = 1;
834         }
835         if(RX_pload[6] == 0x00)
836         {
837             if(((Address_Route & 0x20) == (RX_pload[0] & 0x20) && (((Address_Route & 0x02)+1) == (RX_pload[3] & 0x02)) || (((Address_Route & 0x02)+15) == (RX_pload[30] & 0x02) & 0
838             ) || P02 = -P02;
839         }
840     }
841     else if(SD == 1)
842     {
843         if(((Address_Route & 0x20) == (RX_pload[3] & 0x20) && (((Address_Route & 0x02)-1) == (RX_pload[0] & 0x02)))
844         //if(((Address_Route>>4) == (RX_pload[3]>>4)) && (((Address_Route<0x02)-1) && (RX_pload[0]<0x02)))
845         {
846             Decode_input();
847             /*if(RX_pload[5] == 0x01)
848             {
849                 Delay_ms(100); P00 = -P00; Check = 1; }*/
850             Check = 1;
851         }
852     }
853     else if(SD == 2)
854     {
855         if(((Address_Route & 0x20) && (RX_pload[6] & 0x20) && (((Address_Route & 0x02)-1) == (RX_pload[3] & 0x02)))
856         {
857             Decode_input();
858             /*if(RX_pload[8] == 0x01)
859             {
860                 Delay_ms(100); P00 = -P00; Check = 1; }*/
861             //if (TX_pload[7] == 0x00) P00 = -P00;
862             //if (TX_pload[8] == 0x01) P01 = -P01;
863             Check = 1;
864         }
865     }
866     else if(SD == 3)
867     {
868         if(((Address_Route & 0x20) && (RX_pload[9] & 0x20) && (((Address_Route & 0x02)+2) == (RX_pload[15] & 0x02)) || (((Address_Route & 0x02)-2) == (RX_pload[3] & 0x02)
869         {
870             if(RX_pload[11] == 0x01)
871             {
872                 Delay_ms(100); P00 = -P00; Check = 1; }
873             }
874             if(RX_pload[15] == 0x00)
875             {
876                 if(((Address_Route & 0x20) && (RX_pload[9] & 0x20) && (((Address_Route & 0x02)+1) == (RX_pload[12] & 0x02)) || (((Address_Route & 0x02)-2) == (RX_pload[3] & 0x02)
877                 ) || P02 = -P02;
878             }
879         }
880     }
881     else if(SD == 4)
882     {
883         if(((Address_Route & 0x20) && (RX_pload[12] & 0x20) && (((Address_Route & 0x02)-1) == (RX_pload[6] & 0x02)))
884         {
885             if(RX_pload[14] == 0x01)
886             {
887                 Delay_ms(100); P00 = -P00; Check = 1; }
888         }
889     }
890     else if(SD == 5)
891     {
892         if(((Address_Route & 0x20) && (RX_pload[15] & 0x20) && (((Address_Route & 0x02)-1) == (RX_pload[12] & 0x02)))
893         {
894             if(RX_pload[17] == 0x01)
895             {
896                 Delay_ms(100); P00 = -P00; Check = 1; }
897         }
898     }
899     else if(SD == 6)
900     {
901         if(((Address_Route & 0x20) && (RX_pload[18] & 0x20) && (((Address_Route & 0x02)+2) == (RX_pload[24] & 0x02)) || (((Address_Route & 0x02)-3) == (RX_pload[6] & 0x02)
902         {
903             if(RX_pload[20] == 0x01)
904             {
905                 Delay_ms(100); P00 = -P00; Check = 1; }
906             }
907             if(RX_pload[24] == 0x00)
908             {
909                 if(((Address_Route & 0x20) && (RX_pload[9] & 0x20) && (((Address_Route & 0x02)+1) == (RX_pload[21] & 0x02)) || (((Address_Route & 0x02)-3) == (RX_pload[2] & 0x02)
910                 ) || P02 = -P02;
911             }
912         }
913     }
914     else if(SD == 7)
915     {
916         if(((Address_Route & 0x20) && (RX_pload[21] & 0x20) && (((Address_Route & 0x02)-1) == (RX_pload[18] & 0x02)))
917         {
918             if(RX_pload[23] == 0x01)
919             {
920                 Delay_ms(100); P00 = -P00; Check = 1; }
921         }
922     }
923     else if(SD == 8)
924     {
925         if(((Address_Route & 0x20) && (RX_pload[24] & 0x20) && (((Address_Route & 0x02)-1) == (RX_pload[21] & 0x02)))
926         {
927             if(RX_pload[26] == 0x01)
928             {
929                 Delay_ms(100); P00 = -P00; Check = 1; }
930             }
931         }
932     }
933 }

```

เอกสารนี้เป็นลิขสิทธิ์สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 หากกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

0133 void main()
0134 {
0135     MF = 1;
0136     EA = 1;
0137
0138     // Set FUSE INTO_INT1
0139     PORTB = 0x70;
0140     FIBIP = 0x00;
0141     P0B = 1;
0142     P0C = 1;
0143     INTRIP = 0x10; //set INTO_INT1
0144     IT1 = 1;
0145     ITU = 1;
0146     ///////////////////////////////////////////////////
0147     // Set TIMEL1 //
0148     TRCN = 0x10; // 01 Mode 11 1A bit counter/clock
0149     TH1 = 0;
0150     TL1 = 0;
0151     R11 = 1;
0152     ///////////////////////////////////////////////////
0153     // Set HART //
0154     R0CON = 0x80; //Mode 1 B bit HART.
0155     M0BIF = 0x00;
0156     R0RLL = 0x0A;
0157     M0RLL = 0x0F;
0158     R0CON = 0x80;
0159     P0CON = P0CON & 0x00;
0160
0161     // Interrupt //
0162     *EKU = 1;
0163     *EK1 = 1; //
0164     // EMN //
0165     *EMCON = 0x11;
0166
0167     M0FCE = 0;
0168     M0FCR = 1;
0169     M0FCKEN = 1;
0170     //Delay_ms(500);
0171
0172     CE_Pin_Available(CE_LOW); // Set CE pin low to enable standby mode
0173     //
0174     //
0175     //
0176     //
0177     //
0178     //
0179     //
0180     //
0181     //
0182     //
0183     //
0184     //
0185     //
0186     //
0187     //
0188     //
0189     //
0190     //
0191     //
0192     //
0193     //
0194     //
0195     //
0196     //
0197     //
0198     //
0199     //
0200     //
0201     //
0202     //
0203     //
0204     //
0205     //
0206     //
0207     //
0208     //
0209     //
0210     //
0211     //
0212     //
0213     //
0214     //
0215     //
0216     //
0217     //
0218     //
0219     //
0220     //
0221     //
0222     //
0223     //
0224     //
0225     //
0226     //
0227     //
0228     //
0229     //
0230     //
0231     //
0232     //
0233     //
0234     //
0235     //
0236     //
0237     //
0238     //
0239     //
0240     //
0241     //
0242     //
0243     //
0244     //
0245     //
0246     //
0247     //
0248     //
0249     //
0250     //
0251     //
0252     //
0253     //
0254     //
0255     //
0256     //
0257     //
0258     //
0259     //
0260     //
0261     //
0262     //
0263     //
0264     //
0265     //
0266     //
0267     //
0268     //
0269     //
0270     //
0271     //
0272     //
0273     //
0274     //
0275     //
0276     //
0277     //
0278     //
0279     //
0280     //
0281     //
0282     //
0283     //
0284     //
0285     //
0286     //
0287     //
0288     //
0289     //
0290     //
0291     //
0292     //
0293     //
0294     //
0295     //
0296     //
0297     //
0298     //
0299     //
0300     //
0301     //
0302     //
0303     //
0304     //
0305     //
0306     //
0307     //
0308     //
0309     //
0310     //
0311     //
0312     //
0313     //
0314     //
0315     //
0316     //
0317     //
0318     //
0319     //
0320     //
0321     //
0322     //
0323     //
0324     //
0325     //
0326     //
0327     //
0328     //
0329     //
0330     //
0331     //
0332     //
0333     //
0334     //
0335     //
0336     //
0337     //
0338     //
0339     //
0340     //
0341     //
0342     //
0343     //
0344     //
0345     //
0346     //
0347     //
0348     //
0349     //
0350     //
0351     //
0352     //
0353     //
0354     //
0355     //
0356     //
0357     //
0358     //
0359     //
0360     //
0361     //
0362     //
0363     //
0364     //
0365     //
0366     //
0367     //
0368     //
0369     //
0370     //
0371     //
0372     //
0373     //
0374     //
0375     //
0376     //
0377     //
0378     //
0379     //
0380     //
0381     //
0382     //
0383     //
0384     //
0385     //
0386     //
0387     //
0388     //
0389     //
0390     //
0391     //
0392     //
0393     //
0394     //
0395     //
0396     //
0397     //
0398     //
0399     //
0400     //
0401     //
0402     //
0403     //
0404     //
0405     //
0406     //
0407     //
0408     //
0409     //
0410     //
0411     //
0412     //
0413     //
0414     //
0415     //
0416     //
0417     //
0418     //
0419     //
0420     //
0421     //
0422     //
0423     //
0424     //
0425     //
0426     //
0427     //
0428     //
0429     //
0430     //
0431     //
0432     //
0433     //
0434     //
0435     //
0436     //
0437     //
0438     //
0439     //
0440     //
0441     //
0442     //
0443     //
0444     //
0445     //
0446     //
0447     //
0448     //
0449     //
0450     //
0451     //
0452     //
0453     //
0454     //
0455     //
0456     //
0457     //
0458     //
0459     //
0460     //
0461     //
0462     //
0463     //
0464     //
0465     //
0466     //
0467     //
0468     //
0469     //
0470     //
0471     //
0472     //
0473     //
0474     //
0475     //
0476     //
0477     //
0478     //
0479     //
0480     //
0481     //
0482     //
0483     //
0484     //
0485     //
0486     //
0487     //
0488     //
0489     //
0490     //
0491     //
0492     //
0493     //
0494     //
0495     //
0496     //
0497     //
0498     //
0499     //
0500     //
0501     //
0502     //
0503     //
0504     //
0505     //
0506     //
0507     //
0508     //
0509     //
0510     //
0511     //
0512     //
0513     //
0514     //
0515     //
0516     //
0517     //
0518     //
0519     //
0520     //
0521     //
0522     //
0523     //
0524     //
0525     //
0526     //
0527     //
0528     //
0529     //
0530     //
0531     //
0532     //
0533     //
0534     //
0535     //
0536     //
0537     //
0538     //
0539     //
0540     //
0541     //
0542     //
0543     //
0544     //
0545     //
0546     //
0547     //
0548     //
0549     //
0550     //
0551     //
0552     //
0553     //
0554     //
0555     //
0556     //
0557     //
0558     //
0559     //
0560     //
0561     //
0562     //
0563     //
0564     //
0565     //
0566     //
0567     //
0568     //
0569     //
0570     //
0571     //
0572     //
0573     //
0574     //
0575     //
0576     //
0577     //
0578     //
0579     //
0580     //
0581     //
0582     //
0583     //
0584     //
0585     //
0586     //
0587     //
0588     //
0589     //
0590     //
0591     //
0592     //
0593     //
0594     //
0595     //
0596     //
0597     //
0598     //
0599     //
0600     //
0601     //
0602     //
0603     //
0604     //
0605     //
0606     //
0607     //
0608     //
0609     //
0610     //
0611     //
0612     //
0613     //
0614     //
0615     //
0616     //
0617     //
0618     //
0619     //
0620     //
0621     //
0622     //
0623     //
0624     //
0625     //
0626     //
0627     //
0628     //
0629     //
0630     //
0631     //
0632     //
0633     //
0634     //
0635     //
0636     //
0637     //
0638     //
0639     //
0640     //
0641     //
0642     //
0643     //
0644     //
0645     //
0646     //
0647     //
0648     //
0649     //
0650     //
0651     //
0652     //
0653     //
0654     //
0655     //
0656     //
0657     //
0658     //
0659     //
0660     //
0661     //
0662     //
0663     //
0664     //
0665     //
0666     //
0667     //
0668     //
0669     //
0670     //
0671     //
0672     //
0673     //
0674     //
0675     //
0676     //
0677     //
0678     //
0679     //
0680     //
0681     //
0682     //
0683     //
0684     //
0685     //
0686     //
0687     //
0688     //
0689     //
0690     //
0691     //
0692     //
0693     //
0694     //
0695     //
0696     //
0697     //
0698     //
0699     //
0700     //
0701     //
0702     //
0703     //
0704     //
0705     //
0706     //
0707     //
0708     //
0709     //
0710     //
0711     //
0712     //
0713     //
0714     //
0715     //
0716     //
0717     //
0718     //
0719     //
0720     //
0721     //
0722     //
0723     //
0724     //
0725     //
0726     //
0727     //
0728     //
0729     //
0730     //
0731     //
0732     //
0733     //
0734     //
0735     //
0736     //
0737     //
0738     //
0739     //
0740     //
0741     //
0742     //
0743     //
0744     //
0745     //
0746     //
0747     //
0748     //
0749     //
0750     //
0751     //
0752     //
0753     //
0754     //
0755     //
0756     //
0757     //
0758     //
0759     //
0760     //
0761     //
0762     //
0763     //
0764     //
0765     //
0766     //
0767     //
0768     //
0769     //
0770     //
0771     //
0772     //
0773     //
0774     //
0775     //
0776     //
0777     //
0778     //
0779     //
0780     //
0781     //
0782     //
0783     //
0784     //
0785     //
0786     //
0787     //
0788     //
0789     //
0790     //
0791     //
0792     //
0793     //
0794     //
0795     //
0796     //
0797     //
0798     //
0799     //
0800     //
0801     //
0802     //
0803     //
0804     //
0805     //
0806     //
0807     //
0808     //
0809     //
0810     //
0811     //
0812     //
0813     //
0814     //
0815     //
0816     //
0817     //
0818     //
0819     //
0820     //
0821     //
0822     //
0823     //
0824     //
0825     //
0826     //
0827     //
0828     //
0829     //
0830     //
0831     //
0832     //
0833     //
0834     //
0835     //
0836     //
0837     //
0838     //
0839     //
0840     //
0841     //
0842     //
0843     //
0844     //
0845     //
0846     //
0847     //
0848     //
0849     //
0850     //
0851     //
0852     //
0853     //
0854     //
0855     //
0856     //
0857     //
0858     //
0859     //
0860     //
0861     //
0862     //
0863     //
0864     //
0865     //
0866     //
0867     //
0868     //
0869     //
0870     //
0871     //
0872     //
0873     //
0874     //
0875     //
0876     //
0877     //
0878     //
0879     //
0880     //
0881     //
0882     //
0883     //
0884     //
0885     //
0886     //
0887     //
0888     //
0889     //
0890     //
0891     //
0892     //
0893     //
0894     //
0895     //
0896     //
0897     //
0898     //
0899     //
0900     //
0901     //
0902     //
0903     //
0904     //
0905     //
0906     //
0907     //
0908     //
0909     //
0910     //
0911     //
0912     //
0913     //
0914     //
0915     //
0916     //
0917     //
0918     //
0919     //
0920     //
0921     //
0922     //
0923     //
0924     //
0925     //
0926     //
0927     //
0928     //
0929     //
0930     //
0931     //
0932     //
0933     //
0934     //
0935     //
0936     //
0937     //
0938     //
0939     //
0940     //
0941     //
0942     //
0943     //
0944     //
0945     //
0946     //
0947     //
0948     //
0949     //
0950     //
0951     //
0952     //
0953     //
0954     //
0955     //
0956     //
0957     //
0958     //
0959     //
0960     //
0961     //
0962     //
0963     //
0964     //
0965     //
0966     //
0967     //
0968     //
0969     //
0970     //
0971     //
0972     //
0973     //
0974     //
0975     //
0976     //
0977     //
0978     //
0979     //
0980     //
0981     //
0982     //
0983     //
0984     //
0985     //
0986     //
0987     //
0988     //
0989     //
0990     //
0991     //
0992     //
0993     //
0994     //
0995     //
0996     //
0997     //
0998     //
0999     //
1000     //

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อการเรียนการสอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่าในรูปแบบใดๆก็ตาม หากมีให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ก.3 ส่วนโปรแกรมการสร้างเว็บเพจ

โครงการนี้จะใช้โปรแกรม Dreamweaver ในการสร้างเว็บเพจ ซึ่งจะใช้ภาษา html ในการเขียนโค้ดการทำงาน เพื่อใช้ในการสั่งงานไม่โครคอนโทรลเลอร์ให้ควบคุมการทำงานของเครื่องปรับอากาศสามารถแสดงได้ดังนี้

```
<script type="text/javascript" src="new 3.js"></script>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Control Air Condition System</title>
<style type="text/css">
body {
    background-color: #666;
}
body,td,th {
    color: #F30;
}
</style>
</head>
<body>
<table width="952" border="0" align="center" cellpadding="0" cellspacing="0">
<tr>
<td colspan="2"></td>
</tr>
<tr>
<th width="266" bgcolor="#FFFFFF"><p><a href="ftp://www.ehouse-
air.com//ProjectWeb/IndexNew.htm"></a></p>
<p><a href="project-air.pptx"></a></p>
<p></p></th>
<th width="689" align="center" bgcolor="#FFFFFF"><p>&nbsp;</p>
<p></p>
<p></p>
```



```

<p>&nbsp;</p></th>
<th bgcolor="#FFFFFF"><p><br>
<br /></p>
<table width="300" border="0" align="center" cellpadding="0" cellspacing="0">
<tr>
<th width="150"><p><a href="http://localhost/sentdataOpen2.php"></a></p></th>
<th width="150">
<p><a href="http://localhost/sentdataClose2.php"></a></p></th>
</tr>
</table>
<p></p>
<table width="683" border="1" cellspacing="5" cellpadding="5">
<tr>
<th width="121" bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_16.php">16</a></h1></th>
<th width="121" bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_17.php">17</a></h1></th>
<th width="121" bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_18.php">18</a></h1></th>
<th width="121" bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_19.php">19</a></h1></th>
<th width="107" bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_20.php">20</a></h1></th>
</tr>
<tr>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_21.php">21</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_22.php">22</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_23.php">23</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_24.php">24</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_25.php">25</a></h1></th>
</tr>

```

```

<tr>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_26.php">26</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_27.php">27</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_28.php">28</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_29.php">29</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_30.php">30</a></h1></th>
</tr>
</table>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
</tr>
</table>
</body>
</html>

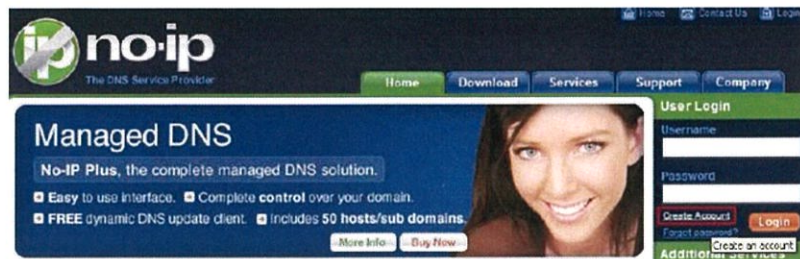
```

ก.4 ขั้นตอนการสมัครNo-ipและการติดตั้งNo-ip

No-ipคือการสร้างCreate Domainย่อยมาใช้งานหรือDDNSนั่นเองมีประโยชน์ในการ Remoteมาสั่งงานServer ในกรณีที่ไม่ได้อยู่ในวง Network เดียวกัน ทำให้การจัดการเกิดขึ้นที่ไหนบนโลกนี้ก็ได้ ขั้นตอนการสมัครมีดังนี้

1. เข้าไปที่เว็บไซต์ <http://www.no-ip.com>เพื่อสมัครสมาชิกเพื่อสมัครสมาชิก จากนั้นไปที่ Create Account

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



2. กรอกข้อมูลให้ครบแล้วเลือกที่ Accept

Create Your No-IP Account

If you already have an account then you can [login](#) [here](#).

Account Information:

Email:

Password:

Confirm Password:

About You:

First Name:

Last Name:

How did you hear about us?

Zip/Postal Code:

Account Access:

Security Questions:

Your Answer:

Birthdate:

Client Login

Home: [Client Login](#)

Email:

Password:

Forget your password? No problem. [Click Here](#)

If you are not currently a registered user of No-IP.com and would like to be for FREE, [register](#) as a new user.

5. เมื่อล็อกอินแล้วให้เลือก Add a Host

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Your No-IP

SeedSoftnoip welcome to your No-IP! Last Login: 2009-05-07 01:32:39 from IP [redacted]

You have successfully logged into No-IP's member section. To start using No-IP's services select an icon below or choose an item from the navigation above.

Manage Domains Add Domain Refer Friend **Add a Host** Manage Hosts

FROM \$15 Find a Domain Name [input] .com Search

6. ใส่ชื่อ Hostname ที่ต้องการ และกด Create Host

Hostname Information

Hostname: seedsoft no-ip.org

Host Type: DNS Host (A) DNS Host (Round Robin) DNS Alias (CNAME)
 Port 80 Redirect Web Redirect

IP Address: [redacted]

Assign to Group: No Group [Configure Groups](#)

Enable Wildcard: Wildcards are a Plus! Enhanced feature. [Upgrade Now!](#) [DNS Host \(A\) Options](#)

• Accept Mail for your Domain
Let No-IP do the dirty work. Setup POP or forwarding for your name.

Mail Options

MX Record MX Priority

Enter the name of your external mail exchangers (mx records) as hostnames not IP addresses

If you would like a more MX records, please upgrade to [Type Plus](#) or [Subhost](#)

[Revert](#) [Create host](#)

✔ Host seedsoft.no-ip.org created. Update will be applied within 1 minute.

Current Hosts: 2 of 5 Need More Hosts? Enhance Your Account! [Upgrade Now!](#)

Host	IP/URL	Action
Hosts By Domain		
no-ip.org		
seedsoft.no-ip.org	[redacted]	Modify Remove
seedsoft2.no-ip.org	[redacted]	Modify Remove

[Add a Host](#)



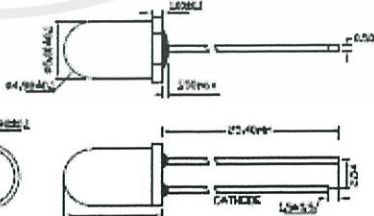




7. no-ip ที่สร้างขึ้นมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข
คู่มือการใช้งานอุปกรณ์ในโครงการ

ในส่วนของภาคผนวกข. จะอธิบายในรายละเอียดสำคัญของอุปกรณ์ที่ประกอบอยู่ภายในส่วนต่างๆของระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ท

ข.1 อินฟราเรดแอลอีดี

 TAIWAN OASIS LED DATA SHEET (FOR INFRARED)									
PART NO. : TOIR-50b94bCEa									
ABSOLUTE MAXIMUM RATINGS AT TA=25°C									
PARAMETER	SYMBOL	DATA	UNIT						
Forward Current	I_{FM}	100	mA						
Peak Forward Current (duty=1/100, f=100KHz)	I_{FM}	1000	mA						
Reverse Voltage	V_R	6	V						
Power Dissipation	P_D	150	mW						
Operating Temperature Range		-25 to +85	°C						
Storage Temperature Range		-30 to +85	°C						
Lead Solder Temperature (1/10 Inch Below Seating Plane)		260°C for 3 sec.							
ELECTRICAL/OPTICAL CHARACTERISTICS AT TA=25°C									
PARAMETER	SYMBOL	DATA	UNIT	TEST CONDITION					
Radiated Output Power	$P_{OUT(P)}$	12.0	mW	Distance: 10cm $I_F=50mA$ Detector Area: 1cm ²					
Forward Voltage	V_F	TYP: 1.25	V	$I_F=20mA$					
		MAX: 1.45							
Wavelength	λ_P	940	nm	$I_F=20mA$					
Spectrum Width of Half Value	$\Delta\lambda$	50	nm	$I_F=20mA$					
Reverse Current	I_R	10	uA	$V_R = -5V$					
Full Viewing Angle	$2\theta_{1/2}$	25	°	$I_F=20mA$					
Lens		Water Clear							
Radiation Material		GaAs/GaAs							
PACKAGE DIMENSIONS & INTERNAL CIRCUIT DIAGRAM									
 									
DATE	01/10/01	SCALE	2.5:1	TOLERANCE	±0.10 ±0.025	DRAWN		CHECKED	
UNIT	M/M	SHEET NO.	1/2	DRAWING NO.	S-50b94bCEa-1	CUSTOMER		APPROVED	

เอกสารนี้เป็นเอกสาร

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โครงการนี้เป็นเอกสาร

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข.2 ตัวรับสัญญาณอินฟราเรด (IR Receiver)



TSOP48..
Vishay Telefunken

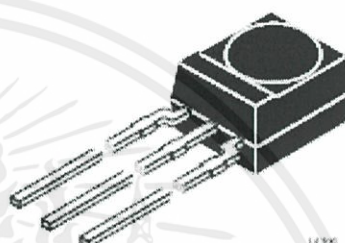
Photo Modules for PCM Remote Control Systems

Available types for different carrier frequencies

Type	fo	Type	fo
TSOP4830	30 kHz	TSOP4833	33 kHz
TSOP4836	36 kHz	TSOP4837	36.7 kHz
TSOP4838	38 kHz	TSOP4840	40 kHz
TSOP4856	56 kHz		

Description

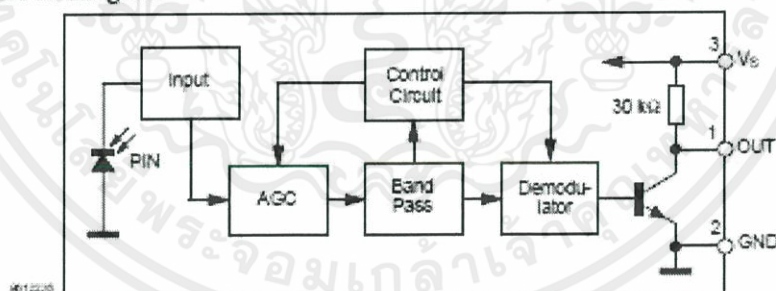
The TSOP48.. – series are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter. The demodulated output signal can directly be decoded by a microprocessor. TSOP48.. is the standard IR remote control receiver series, supporting all major transmission codes.



Features

- Photo detector and preamplifier in one package
- Internal filter for PCM frequency
- Improved shielding against electrical field disturbance
- TTL and CMOS compatibility
- Output active low
- Low power consumption
- High immunity against ambient light
- Continuous data transmission possible (800 bits)
- Suitable burst length ≥ 10 cycles/burst

Block Diagram



Document Number 82090
Rev. 0, 29-Mar-01

www.vishay.com
1 (7)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TSOP48..

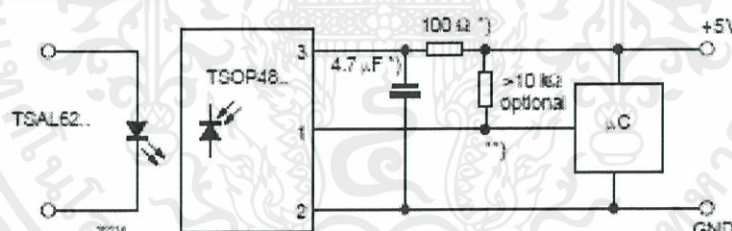
Vishay Telefunken

**Absolute Maximum Ratings** $T_{amb} = 25^{\circ}\text{C}$

Parameter	Test Conditions	Symbol	Value	Unit
Supply Voltage	(Pin 3)	V_{CC}	-0.3...6.0	V
Supply Current	(Pin 3)	I_{CC}	5	mA
Output Voltage	(Pin 1)	V_{OL}	-0.3...6.0	V
Output Current	(Pin 1)	I_{OL}	5	mA
Junction Temperature		T_j	100	$^{\circ}\text{C}$
Storage Temperature Range		T_{stg}	-25...+85	$^{\circ}\text{C}$
Operating Temperature Range		T_{opr}	-25...+85	$^{\circ}\text{C}$
Power Consumption	($T_{amb} \leq 85^{\circ}\text{C}$)	P_{tot}	50	mW
Soldering Temperature	$t \leq 10\text{ s}$, 1 mm from case	T_{solder}	260	$^{\circ}\text{C}$

Basic Characteristics $T_{amb} = 25^{\circ}\text{C}$

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Supply Current (Pin 3)	$V_{CC} = 5\text{ V}$, $E_{in} = 0$	I_{CC}	0.8	1.1	1.5	mA
	$V_{CC} = 5\text{ V}$, $E_{in} = 40\text{ klx}$, sunlight	I_{CC}		1.4		mA
Supply Voltage (Pin 3)		V_{CC}	4.5		5.5	V
Transmission Distance	$E_{in} = 0$, test signal see fig. 7, IR diode TSAL6200, $I_F = 250\text{ mA}$	d		35		m
Output Voltage Low (Pin 1)	$I_{OL} = 0.5\text{ mA}$, $E_{in} = 0.7\text{ mW/cm}^2$	V_{OL}			250	mV
Irradiance (30 – 40 kHz)	Pulse width tolerance: $t_{ON} - 5\tau_{ON} < t_{ON} < t_{ON} + 5\tau_{ON}$, test signal see fig. 7	$E_{in\ min}$		0.2	0.4	mW/cm^2
Irradiance (56 kHz)	Pulse width tolerance: $t_{ON} - 5\tau_{ON} < t_{ON} < t_{ON} + 5\tau_{ON}$, test signal see fig. 7	$E_{in\ min}$		0.3	0.6	mW/cm^2
Irradiance	$t_{ON} - 5\tau_{ON} < t_{ON} < t_{ON} + 5\tau_{ON}$	$E_{in\ max}$	30			W/cm^2
Directivity	Angle of half transmission distance	$\theta_{1/2}$		± 45		deg

Application Circuit

*) recommended to suppress power supply disturbances

**) The output voltage should not be hold continuously at a voltage below 3.3V by the external circuit.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Suitable Data Format

The circuit of the TSOP48.. is designed in that way that unexpected output pulses due to noise or disturbance signals are avoided. A bandpassfilter, an Integrator stage and an automatic gain control are used to suppress such disturbances.

The distinguishing mark between data signal and disturbance signal are carrier frequency, burst length and duty cycle.

The data signal should fulfill the following condition:

- Carrier frequency should be close to center frequency of the bandpass (e.g. 38kHz).
- Burst length should be 10 cycles/burst or longer.
- After each burst which is between 10 cycles and 70 cycles a gap time of at least 14 cycles is necessary.
- For each burst which is longer than 1.8ms a corresponding gap time is necessary at some time in the data stream. This gap time should be at least 4 times longer than the burst.
- Up to 800 short bursts per second can be received continuously.

Some examples for suitable data format are: NEC Code, Toshiba Miborn Format, Sharp Code, RC5 Code, RC6 Code, R-2000 Code.

When a disturbance signal is applied to the TSOP48.. It can still receive the data signal. However the sensitivity is reduced to that level that no unexpected pulses will occur.

Some examples for such disturbance signals which are suppressed by the TSOP48.. are:

- DC light (e.g. from tungsten bulb or sunlight)
- Continuous signal at 38kHz or at any other frequency
- Signals from fluorescent lamps with electronic ballast with high or low modulation (see Figure A or Figure B).

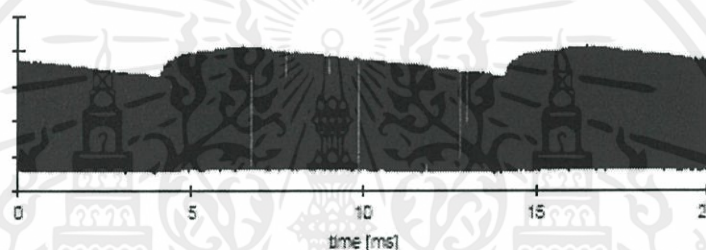


Figure A: IR Signal from Fluorescent Lamp with low Modulation

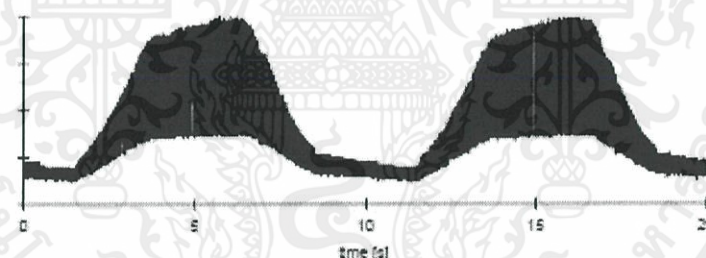


Figure B: IR Signal from Fluorescent Lamp with high Modulation

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TSOP48..

Vishay Telefunken

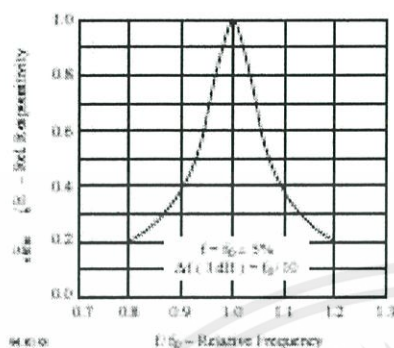
Typical Characteristics ($T_{amb} = 25^{\circ}\text{C}$ unless otherwise specified)

Figure 1. Frequency Dependence of Responsivity

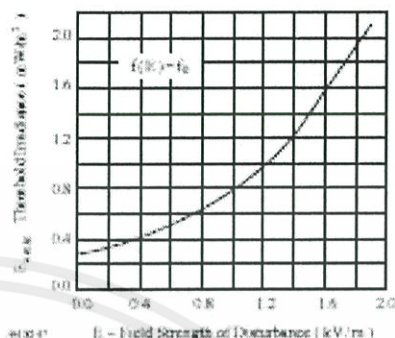


Figure 4. Sensitivity vs. Electric Field Disturbances

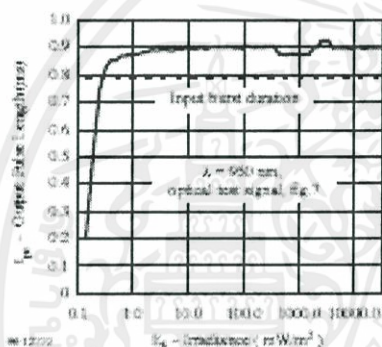


Figure 2. Sensitivity in Dark Ambient

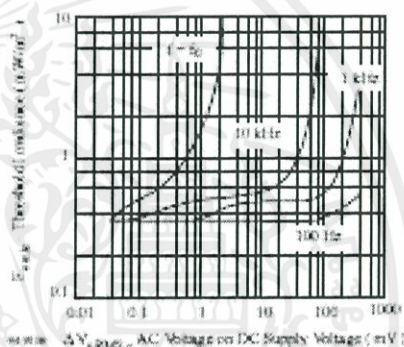


Figure 5. Sensitivity vs. Supply Voltage Disturbances

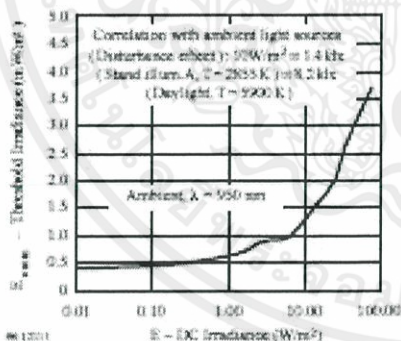


Figure 3. Sensitivity in Bright Ambient

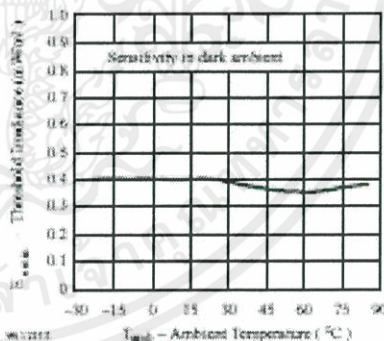


Figure 6. Sensitivity vs. Ambient Temperature

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



TSOP48..

Vishay Telefunken

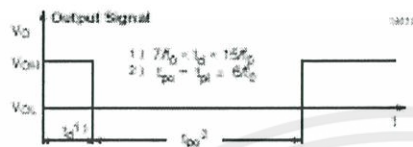
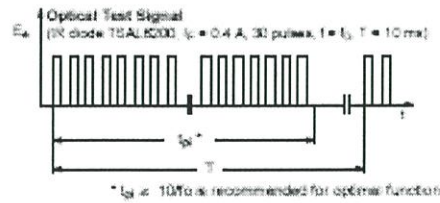


Figure 7.

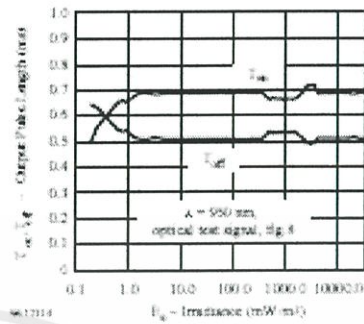


Figure 10. Output Pulse Diagram

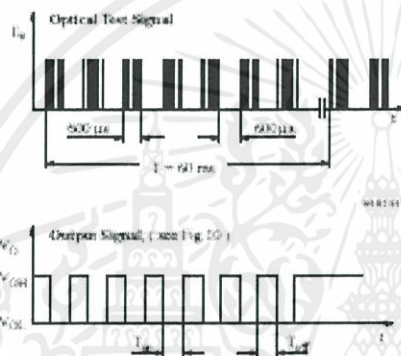


Figure 8. Output Function

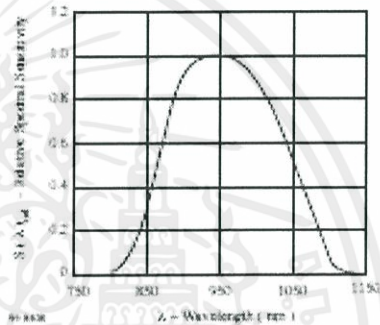


Figure 11. Relative Spectral Sensitivity vs. Wavelength

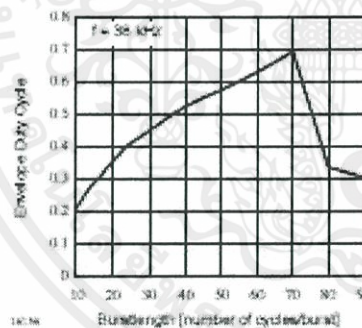


Figure 9. Max. Envelope Duty Cycle vs. Burstlength

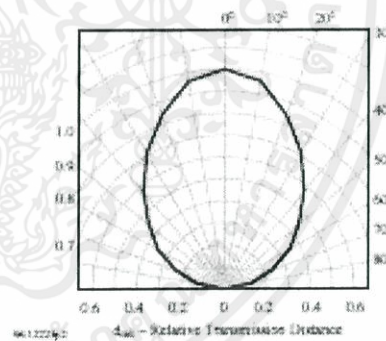


Figure 12. Directivity

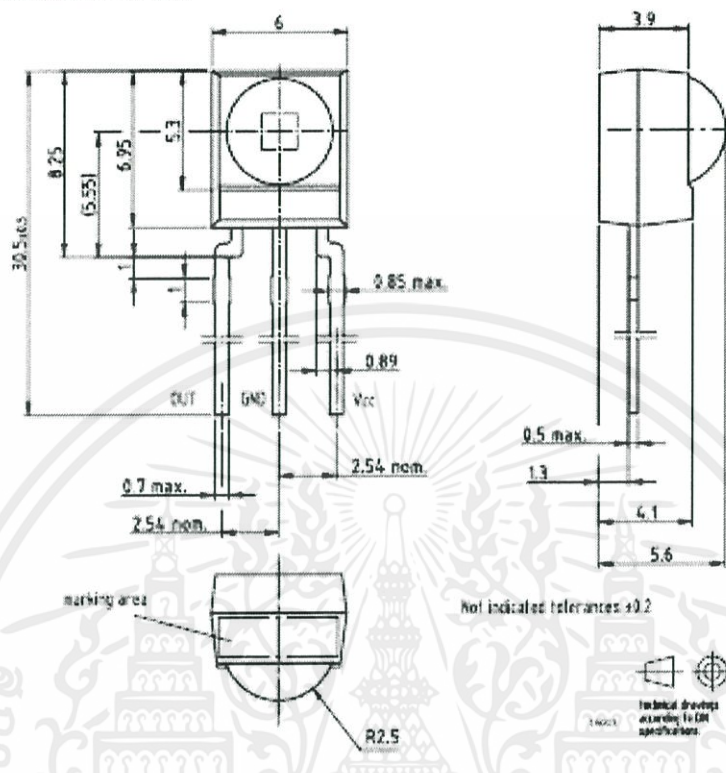
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TSOP48..

Vishay Telefunken




Dimensions in mm



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข.3 ตัวส่งสัญญาณอินฟราเรด

PARAMETER		SYMBOL	DATA	UNIT
Forward Current		I_{FM}	100	mA
Peak Forward Current (duty=1:100, $f=100\text{KHz}$)		I_{FP}	1000	mA
Reverse Voltage		V_R	6	V
Power Dissipation		P_D	150	mW
Operating Temperature Range			-25 to +85	°C
Storage Temperature Range			-30 to +85	°C
Lead Solder Temperature (1/10 Inch Below Soldering Plane)			260°C for 3 sec.	


TAIWAN OASIS LED DATA SHEET (FOR INFRARED)
 PART NO. : TOIR-50b94bCEa
 ABSOLUTE MAXIMUM RATINGS AT $T_A=25^\circ\text{C}$

ELECTRICAL/OPTICAL CHARACTERISTICS AT $T_A=25^\circ\text{C}$				
PARAMETER	SYMBOL	DATA	UNIT	TEST CONDITION
Radiated Output Power	$P_o(\text{typ.})$	12.0	mW	Distance: 10cm $I_F=50\text{mA}$ Detector Area: 1cm^2
Forward Voltage	V_f	TYP: 1.25 MAX: 1.45	V	$I_F=20\text{mA}$
Wavelength	λ_P	940	nm	$I_F=20\text{mA}$
Spectrum Width of Half Value	$\Delta\lambda$	50	nm	$I_F=20\text{mA}$
Reverse Current	I_R	10	μA	$V_R=-5\text{V}$
Full Viewing Angle	$2\theta_{1/2}$	25	°	$I_F=20\text{mA}$
Lens		Water Clear		
Radiation Material		GaAs/GaAs		

PACKAGE DIMENSIONS & INTERNAL CIRCUIT DIAGRAM

DATE	01/10/01	SCALE	2.5:1	TOLERANCE	±0.10 ±0.05 20P	DRAWN	พ.พ.จ	CHECKED	
UNIT	M/M	SHEET NO.	1/2	DRAWING NO.	S-50b94bCEa-1	CUSTOMER		APPROVED	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้