ระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต
AIR CONDITIONING CONTROL SYSTEM VIA INTERNET

รัชพงษ์ พุดทรา
สุรัสวดี สุดประเสริฐ
อชิตา รัตนภรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมอัตโนมัติ
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2555

ระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต

AIR CONDITIONING CONTROL SYSTEM VIA INTERNET

รัชพงษ์ พุดทรา
สุรัสวดี สุดประเสริฐ
อชิตา รัตนภรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมอัตโนมัติ
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2555

AIR CONDITIONING CONTROL SYSTEM VIA INTERNET

RATCHAPHONG PUDTHA
SURASSAWADEE SUDPRASERT
ACHITA RATTANAPORN

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

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์ 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

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รองศาสตราจารย์ประภาษ อุคคกิมาพันธุ์	
อาจารย์กฤษณ์ เสมอพิทักษ์	

หัวข้อปริญญานิพนธ์	ระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเตอร์เน็ต AIR CONDITIONING CONTROL SYSTEM VIA INTERNET		
นักศึกษาผู้จัดทำ	นายรัชพงษ์	พุดทรา	**รหัสนักศึกษา** 52010988
	นางสาวสุรัสวดี	สุดประเสริฐ	**รหัสนักศึกษา** 52011345
	นางสาวอชิตา	รัตนภรณ์	**รหัสนักศึกษา** 52011366
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ประภาษ อุคคกิมาพันธุ์		
	อาจารย์กฤษณ์	เสมอพิทักษ์	
ปีการศึกษา	2555		

บทคัดย่อ

โครงงานนี้เป็นการศึกษาโปรโตคอลและชุดคำสั่งที่ใช้ในการสื่อสารระหว่างรีโมทคอนโทรลกับเครื่องใช้ไฟฟ้า และออกแบบ สร้างอุปกรณ์ควบคุมเครื่องใช้ไฟฟ้าที่ใช้รีโมทคอนโทรลแบบส่งสัญญาณอินฟราเรดในการควบคุม โดยเป็นการประยุกต์ใช้ไมโครคอนโทรลเลอร์ร่วมกับการประยุกต์ใช้เทคโนโลยีไร้สายเข้าด้วยกัน ทำให้เกิดการสร้างเครือข่ายของการควบคุมอุปกรณ์ไฟฟ้าขึ้น ซึ่งสามารถจัดการอุปกรณ์ไฟฟ้าผ่านศูนย์ควบคุม โดยใช้หลักการจำลองการทำงานของรีโมทที่ส่งสัญญาณอินฟราเรดในการสั่งงานควบคุมเครื่องใช้ไฟฟ้า และสามารถเชื่อมต่อรูปแบบการสื่อสารนี้ผ่านระบบอินเตอร์เน็ตได้อีกด้วย ซึ่งจะทำให้เกิดความคล่องตัวในการใช้งาน รวมถึงสามารถบริหารจัดการอุปกรณ์ไฟฟ้าภายในบ้านได้อย่างง่ายดาย โดยปริญญานิพนธ์นี้ได้ทำการทดลองกับเครื่องปรับอากาศเพื่อเป็นแนวทางที่สามารถประยุกต์ใช้ในการควบคุมเครื่องใช้ไฟฟ้าอื่นๆ ที่สามารถรับสัญญาณอินฟราเรดได้อีกด้วย

Thesis Title	AIR CONDITIONING CONTROL SYSTEM VIA INTERNET	
Authors	Mr. Ratchaphong	Pudtha
	Ms. Surassawadee	Sudprasert
	Ms. Achita	Rattanaporn
Thesis Advisor	Assoc. Prof. Prapart	Ukakipaparn
	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.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี คณะผู้จัดทำขอกราบขอบพระคุณอาจารย์ที่ปรึกษา รองศาสตราจารย์ประภาษ อุคคกิมาพันธุ์ และอาจารย์กฤษณ์ เสมอพิทักษ์ ที่ได้ให้คำปรึกษา ชี้แนะแนวทางในการแก้ปัญหา ให้ความดูแลเอาใจใส่ ให้คำปรึกษา ความคิดริเริ่มสร้างสรรค์ ถ่ายทอดประสบการณ์การทำงาน ให้แนวคิดและทฤษฎีที่คณะผู้จัดทำไม่เคยได้รู้มาก่อน ตลอดจนสอบถามถึงความก้าวหน้าอย่างสม่ำเสมอ

ขอขอบพระคุณคณาจารย์ภาควิชาวิศวกรรมการวัดและควบคุม สาขาวิศวกรรมอัตโนมัติทุกท่าน ที่ได้ให้คำแนะนำ และการช่วยเหลืออันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

ขอขอบคุณเพื่อนๆ ทุกคนที่ให้กำลังใจ คอยกระตุ้นถามไถ่ความคืบหน้าของปริญญานิพนธ์ ซึ่งเป็นเหมือนแรงกระตุ้นเป็นอย่างดี จนสามารถทำให้โครงงานนี้สำเร็จไปได้ด้วยดี

และสุดท้าย ขอกราบขอบพระคุณพ่อ และแม่อันเป็นที่รักยิ่ง ซึ่งคอยสนับสนุนและเป็นแรงบัลดาลใจในการทำโครงงานนี้ คุณค่าและประโยชน์ของโครงงานนี้ คณะผู้จัดทำขอมอบให้แด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

บ้านพักอาศัยในปัจจุบันเป็นบ้านที่ถูกออกแบบโครงสร้างการวางรูปแบบระบบไฟฟ้าและเครื่องใช้ไฟฟ้าไว้เป็นแบบแผนที่ตายตัว ซึ่งยากต่อการปรับเปลี่ยนรูปแบบการติดตั้ง ทำให้เกิดความไม่คล่องตัวในการเคลื่อนย้ายและการบริหารจัดการ นอกจากนั้นการควบคุมเครื่องใช้ไฟฟ้าภายในบ้านแต่ละชนิดต่างมีข้อจำกัดที่แตกต่างกันตามลักษณะการใช้งาน ในบ้านหนึ่งหลังอาจมีห้องหลายห้องแต่ละห้องต่างมีเครื่องใช้ไฟฟ้าหลายชนิด ทำให้การควบคุมการทำงานของเครื่องใช้ไฟฟ้าแต่ละชนิดนั้นมีความยุ่งยากและไม่ตอบสนองต่อยุคสมัยของสังคมที่มีความเร่งรีบ

เทคโนโลยี E-House เป็นเทคโนโลยีการควบคุมเครื่องใช้ไฟฟ้าด้วยระบบไร้สาย (Wireless) โดยใช้หลักการแทรกวงจรตัดต่อรีเลย์ไว้กับเครื่องใช้ไฟฟ้าหรือการตัดกำลังไฟฟ้าที่จ่ายให้กับเครื่องใช้ไฟฟ้าแต่ละตัวด้วยระบบไร้สายผ่านการควบคุมจากเครือข่ายอินเตอร์เน็ต ซึ่งเป็นการควบคุมที่สั่งงานให้เครื่องใช้ไฟฟ้าปิด-เปิดได้เท่านั้น แต่ยังมีเครื่องใช้ไฟฟ้าอีกหลายชนิดที่ไม่สามารถแทรกวงจรการตัดต่อกำลังไฟฟ้าได้และยังต้องการการควบคุมที่มากกว่าการปิด-เปิด เช่น เครื่องปรับอากาศ โทรทัศน์ เครื่องเล่นวีดีโอ เป็นต้น โดยเครื่องใช้ไฟฟ้าเหล่านี้สามารถควบคุมได้ด้วยรีโมทคอนโทรล ซึ่งใช้การรับส่งคำสั่งด้วยสัญญาณอินฟราเรด แต่เครื่องใช้ไฟฟ้าแต่ละชนิดมีรีโมทคอนโทรลเฉพาะเครื่องนั้น ๆ ไม่สามารถใช้ร่วมกับเครื่องใช้ไฟฟ้าชนิดอื่นได้ ปริญญานิพนธ์นี้จึงได้นำเทคโนโลยี E-House มาประยุกต์ใช้ในการพัฒนาอุปกรณ์ที่สามารถควบคุมเครื่องใช้ไฟฟ้าได้โดยใช้เทคนิคของรีโมทคอนโทรลส่งสัญญาณอินฟราเรดไปควบคุม ซึ่งสามารถสั่งงานการควบคุมผ่านเครือข่ายอินเทอร์เน็ตได้และใช้ระบบไร้สายในการสื่อสารข้อรับส่งคำสั่งจากเครือข่ายอินเทอร์เน็ตมายังอุปกรณ์ที่ใช้ในการควบคุมโดยไม่ต้องแทรกวงจรหรือปรับเปลี่ยนวงจรการทำงานของเครื่องใช้ไฟฟ้าที่มีอยู่เดิมและอุปกรณ์นี้ยังสามารถควบคุมเครื่องใช้ไฟฟ้าหลายชนิดได้ด้วยอุปกรณ์เพียงตัวเดียว โดยใช้หลักการปรับเปลี่ยนซอฟแวร์ที่ใช้ในการควบคุม ในปริญญานิพนธ์นี้ได้ทำการทดสอบการควบคุมกับเครื่องปรับอากาศยี่ห้อไดกิ้น ซึ่งจะเป็นแนวทางในการนำไปประยุกต์ใช้กับเครื่องใช้ไฟฟ้าชนิดอื่น ๆ ต่อไป

1.2 วัตถุประสงค์ของปริญญานิพนธ์

เพื่อออกแบบการควบคุมเครื่องใช้ไฟฟ้าที่เพิ่มความสะดวกสบายให้แก่ผู้ใช้งานโดยอาศัยการทำงานรับส่งคำสั่งด้วยสัญญาณอินฟราเรดที่ควบคุมด้วยไมโครคอนโทรลเลอร์ ซึ่งผู้ใช้งานสามารถสั่งงานได้ผ่านเครือข่ายอินเทอร์เน็ต

1.3 ขอบเขตของปริญญานิพนธ์

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 รายละเอียดของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้มีเนื้อหาทั้งหมด 5 บท โดยแต่ละบทมีรายละเอียดดังต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญ วัตถุประสงค์ และขอบเขตของปริญญานิพนธ์ รวมไปถึงประโยชน์ที่คาดว่าจะได้รับ และรายละเอียดของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีและหลักการที่ใช้ในปริญญานิพนธ์ ซึ่งได้แก่ หลักการของรีโมทคอนโทรล อินฟราเรดแอลอีดี สัญญาณอินฟราเรด พัลส์วิดมอดูเลชั่น ภาษาคอมพิวเตอร์ โครงสร้างของภาษาซีและไมโครคอนโทรลเลอร์ตลอดจนหลักการที่เกี่ยวข้องในการแสดงผลผ่านเว็บบราวเซอร์

บทที่ 3 กล่าวถึงการออกแบบและการสร้างระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต ซึ่งประกอบไปด้วย การออกแบบโครงสร้างของฮาร์ดแวร์ การ

ออกแบบการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ และการออกแบบการประยุกต์ระบบการสื่อสารผ่านอินเทอร์เน็ต

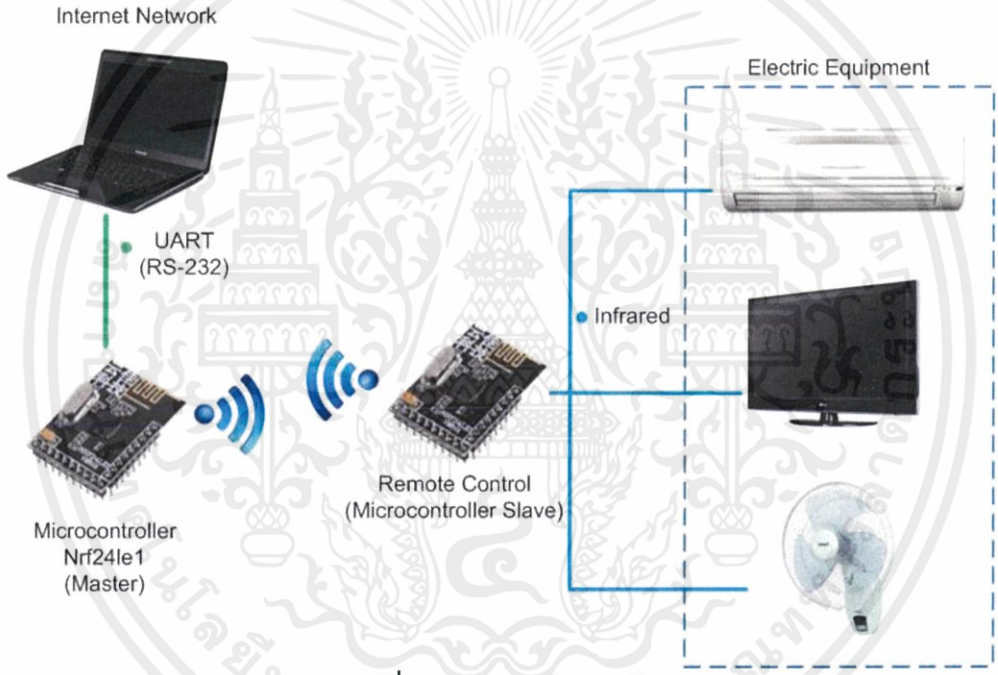
บทที่ 4 กล่าวถึงผลการทดสอบและวิธีการใช้งานระบบควบคุมเครื่องปรับอากาศผ่านอินเตอร์เน็ต

บทที่ 5 กล่าวถึงการสรุปผลการทดสอบการทำงาน และข้อเสนอแนะเกี่ยวกับปริญญานิพนธ์

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

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

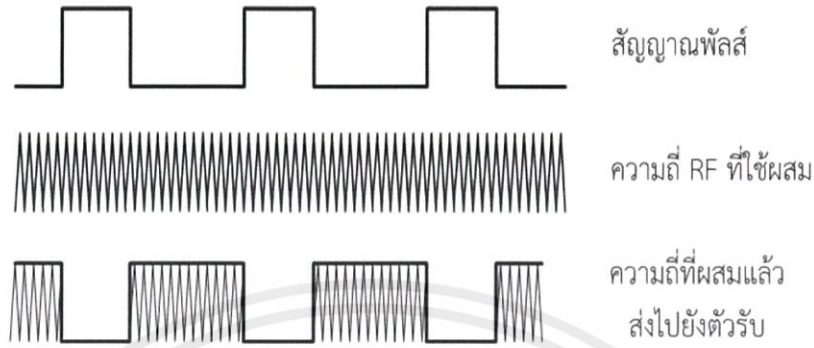


รูปที่ 2.1 โครงสร้างการทำงาน

2.2 หลักการทำงานของรีโมทคอนโทรล

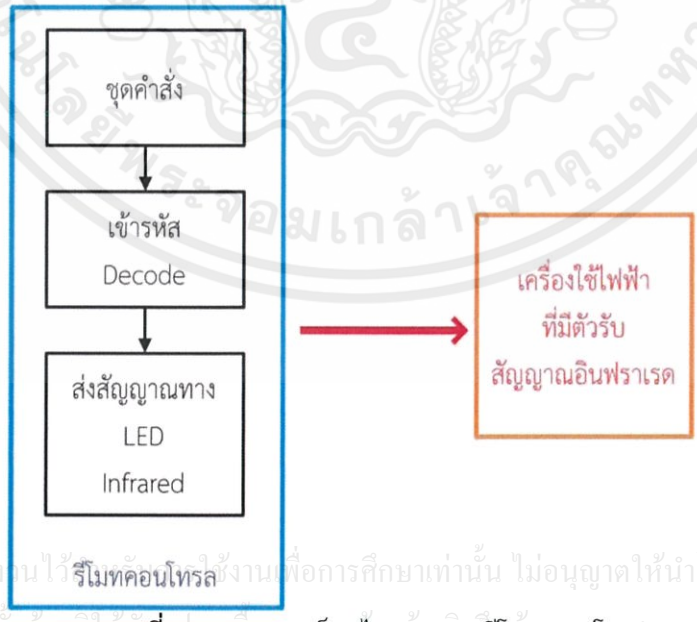
รีโมทคอนโทรล คือเครื่องมืออิเล็กทรอนิกส์ชนิดหนึ่งมีการส่งสัญญาณแบบดิจิตอล ใช้สำหรับควบคุมการดำเนินการของสิ่งประดิษฐ์หรือเครื่องจักรต่างๆโดยเฉพาะเครื่องใช้ไฟฟ้าภายในบ้าน เช่น เครื่องปรับอากาศ โทรทัศน์ เครื่องเสียงจากระยะไกลด้วยแสงอินฟราเรดหรือรังสีใต้แดงซึ่งเป็นคลื่นแม่เหล็กไฟฟ้าที่มีความยาวคลื่น 700 นาโนเมตร – 1 มิลลิเมตร ถี่ในช่วง 10^{11} - 10^{14} เฮิร์ตซ์ มีทิศทางในระดับสายตา ไม่สามารถทะลุผ่านวัตถุทึบแสงได้ และเนื่องจากการเดินทางไกลของพัลส์ในรูปแบบดิจิตอล มีปัญหาในการทะลวงอากาศได้ไม่ไกล เกิดการรบกวนได้ง่าย จึงมีการผสมความถี่วิทยุ

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



รูปที่ 2.2 พัลส์ที่รีโมทคอนโทรลส่งออกมา

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



รูปที่ 2.3 แสดงบล็อกไดอะแกรมรีโมทคอนโทรล

2.2.1 สัญญาณที่รีโมตคอนโทรลส่งออกมาประกอบด้วย 5 ส่วนหลัก ๆ ดังนี้

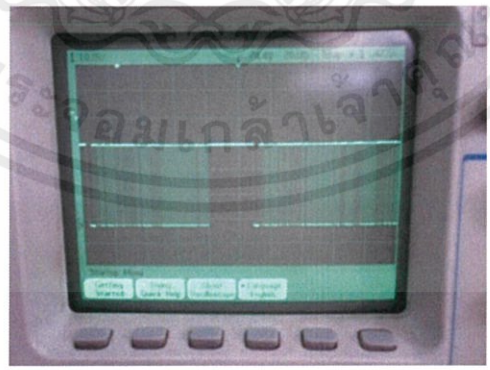
1. ดาต้าโค้ด (Data Code) เป็นข้อมูลหลักที่จะส่งออกไปควบคุมวงจรในส่วนของภาครับ โดยชุดปุ่มคำสั่งจะเป็นตัวสั่งงานเข้าสู่ระบบการเข้ารหัสข้อมูลกำหนดความเป็นไปของแต่ละฟังก์ชัน

2. อินเวิเวิร์ตดาต้าโค้ด (Inverted Data Code) เป็นการกลับลอจิกของข้อมูลหลักเพื่อรักษาฐานเวลาให้คงที่ทุกข้อมูลซึ่งเป็นข้อมูลที่มีจำนวนบิตเท่ากับดาต้าโค้ด

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

4. อินเวิร์ต ดีไวส์โค้ด (Inverted Device Code) เป็นการกลับข้อมูลดีไวส์โค้ดเพื่อรักษาเช่นเดียวกันกับข้อมูลหลักข้อมูลดังกล่าวต้องมี 5 บิตเหมือนดีไวส์โค้ด

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



รูปที่ 2.4 ตัวอย่างการอ่านค่าโค้ดของรีโมทด้วยออสซิลโลสโคป

สำหรับค่าที่ต้องการใช้งานจริง คือ ค่าข้อมูลขนาด 8 บิต หรือ 1 ไบท์ และมีสัญญาณอินเวิร์ตของค่าข้อมูลเพื่อตรวจสอบข้อมูลว่าถูกต้องหรือไม่ และสุดท้ายคือ บิตสต๊อป ซึ่งเป็นการบอกให้ซีพียูรู้ว่ามีสิ้นสุดการส่งค่าแล้ว

หลักการของรีโมทคอนโทรลนั้นใช้ในการติดต่อสื่อสารระหว่างตัวอุปกรณ์ที่สร้างขึ้นเพื่อควบคุมเครื่องใช้ไฟฟ้ากับเครื่องใช้ไฟฟ้าที่ต้องการควบคุม โดยภายในตัวอุปกรณ์นั้นจะมีไมโครคอนโทรลเลอร์ซึ่งเป็นตัวสลาฟ คอยรับส่งข้อมูลคำสั่งกับไมโครคอนโทรลเลอร์ที่เป็นตัวมาสเตอร์ด้วยระบบไร้สาย โดยประยุกต์ใช้หลักการการสื่อสารจากเทคโนโลยี E-House

2.3 เทคโนโลยี E-House

2.3.1 โมดูล nRF24LE1

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

การนำไปประยุกต์ใช้งาน

ไมโครคอนโทรลเลอร์ nRF24LE1 สามารถนำไปประยุกต์ใช้ในงานต่าง ๆ ได้อย่างหลากหลาย โดยจำแนกได้ดังนี้

- อุปกรณ์ต่อพ่วงเกี่ยวกับคอมพิวเตอร์ เช่น เมาส์ 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] โดยในส่วนของโมดูลตัวนี้ได้มีการรับส่งข้อมูลที่ใช้พลังงานต่ำเป็นพิเศษแต่มีการสื่อสารที่มีประสิทธิสูง นอกจากนี้ยังมีการตรวจเช็คการสูญหายข้อมูลด้วย ซึ่งในส่วนนี้จะทำให้การรับ–ส่งข้อมูลนั้นมีประสิทธิภาพเพิ่มขึ้น

คุณสมบัติหลัก

- ขนาดข้อมูลมีขนาดตั้งแต่ 1 – 32 ไบต์ซึ่งสามารถเลือกกำหนดตามความต้องการได้
- สามารถตั้งค่าการจัดการในส่วนของแพ็คเกจข้อมูลได้โดยอัตโนมัติ
- ตอบสนองการรับ-ส่งข้อมูลแบบอัตโนมัติ และจัดส่งข้อมูลใหม่อีกครั้งโดยอัตโนมัติเมื่อมีการสูญหายของข้อมูล

- ติดต่อกันในลักษณะโทโปโลยีแบบสตาร์ (Star Topology) แบบ1 : 6

หลักการจัดการรับ-ส่งข้อมูล

- การส่งแพ็คเกจข้อมูลจาก PTX เมื่อ PRX ได้รับข้อมูล จะกำหนดPTXให้อยู่ในโหมดรับเพื่อรอให้แพ็คเกจ ACK ส่งกลับ

- เมื่อได้รับแพ็คเกจข้อมูลแล้ว PRX จะประกอบแพ็คเกจและส่ง แพ็คเกจที่รับมา (ACK แพ็คเกจ)กลับไปที่ 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 เริ่มต้นอัตโนมัติเป็น 10101010 แต่ถ้าบิตแรกเป็น 0 จะมีการตั้งค่าเริ่มต้นอัตโนมัติเป็น 01010101

- Address เปรียบเสมือนขอบเขตที่อยู่ของสัญญาณ เพื่อให้สามารถรับทราบได้ว่าจะมีการตรวจพบข้อมูลที่ถูกส่งออกมาจะมีการกำหนด 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

รูปที่ 2.6 รูปแบบ Packet Control Field

Payload เป็นส่วนของข้อมูลที่ต้องการจะสื่อสาร ซึ่งจะมีขนาดตามที่กำหนด โดยขนาดสูงสุดอยู่ที่ 32 ไบต์

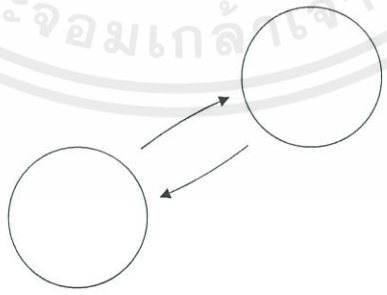
- CRC (Cyclic Redundancy Check) มีขนาด 1 – 2 ไบต์ จะทำหน้าที่ตรวจสอบข้อมูลที่รับเข้ามาว่าตรงกับที่ต้องการหรือไม่ โดยขึ้นอยู่กับความละเอียดที่ต้องการตรวจสอบ

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)

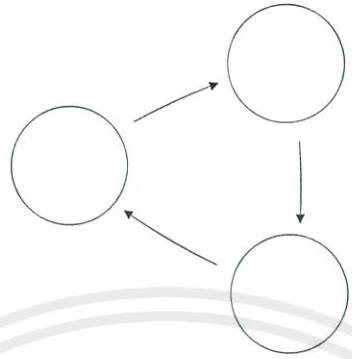
โทโปโลยีในส่วนของการรับส่งข้อมูลย่อยภายในเครือข่ายนั้น ได้มีการวางหลักการการรับส่งข้อมูลในวงเครือข่ายย่อยในลักษณะบรอดแคสโดยได้มีการจำกัดขนาดของจำนวนอุปกรณ์ให้อยู่ในจำนวนที่กำหนด จึงได้มีการจำกัดทิศทางในการรับส่งข้อมูลแบบไร้สายนี้ ให้เป็นไปตามที่กำหนด โดยได้มีการอ้างอิงการเชื่อมต่อในขั้นตอนแรกในลักษณะแบบหนึ่งต่อหนึ่ง ดังรูปที่ 2.7 ซึ่งทำให้ง่ายและสะดวกต่อการจัดการของข้อมูล



รูปที่ 2.7 ลักษณะตัวอย่างในการรับส่งข้อมูลอย่างง่าย

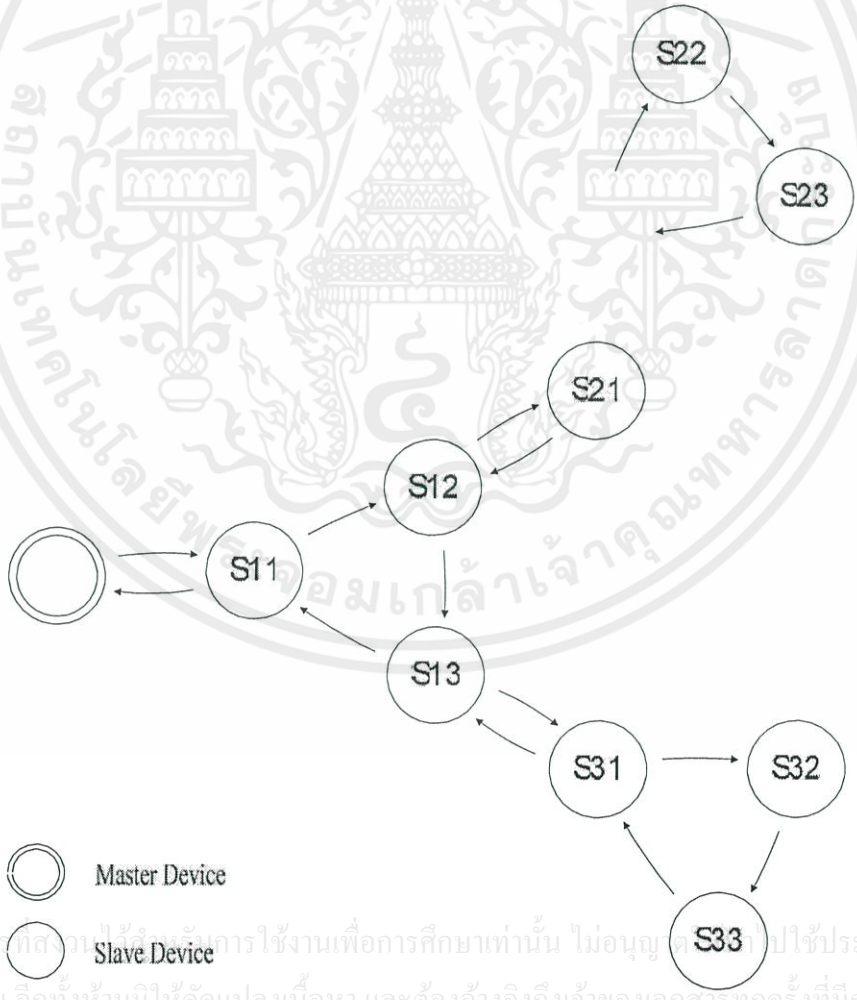
หลังจากนั้น จึงได้มีการวางหลักการคร่าว ๆ ในการรับส่งข้อมูลในวงเครือข่ายย่อยดังรูปที่ 2.8 เพื่อให้เกิดความต่อเนื่องและสะดวกในการนำไปใช้ จึงได้มีการจำกัดทิศทางลักษณะในการ

รับ–ส่งข้อมูลย่อยที่แตกต่างจากเดิม โดยได้กำหนดให้มีการอ้างอิงการเชื่อมต่อกันภายในวงย่อยของตนเองก่อน ก่อนที่จะรับ–ส่งไปยังวงย่อยในวงต่อ ๆ ไปโดยหลักการในลักษณะนี้ สามารถทำให้ขยายจำนวนเครือข่ายได้ง่ายและจำนวนมากขึ้น



รูปที่ 2.8 ลักษณะตัวอย่างในการรับส่งข้อมูลในวงเครือข่ายย่อย

2.3.4 โครงสร้างการติดต่อสื่อสารกันภายในระบบเครือข่ายย่อย



รูปที่ 2.9 โครงสร้างการติดตอสื่อสารกันภายในระบบเครือขาย

โดยการติดต่อสื่อสารสารกันภายในเครือข่ายย่อยนั้นได้มีการแบ่งวงในการสื่อสารกันเป็น 3 รูปแบบ คือ ในรูปแบบที่หนึ่งเครือข่ายย่อยจะติดต่อสื่อสารสารกันภายในวงของตัวเอง ในรูปแบบที่สอง จะเป็นการติดต่อสื่อสารกันโดยข้ามไปยังวงอื่น ๆ และในรูปแบบที่ 3 เป็นการติดต่อสื่อสารจากตัวมาสเตอร์กับวงสลาฟที่กำหนด โดยในรูปแบบที่ 1 และ2 สามารถติดต่อสื่อสารกันไดสูงสุด 3 ตัว

จากรูปที่ 2.9 ตัวเลขในรูป เช่น s11 s12 จะเป็นตัวบ่งบอกถึงวงเครือข่ายย่อยใน การติดต่อสื่อสาร โดยตัวเลขหลักที่ 1 จะเป็นตัวบอกถึงลำดับของวงเครือข่าย และลำดับที่ 2 หมายถึงลำดับอุปกรณ์ที่ติดต่อสื่อสาร ซึ่งมีได้สูงสุด 3 ตัว โดยโครงสร้างนั้นได้มีการวางไว้ให้เกิดความยืดหยุ่นมากขึ้น ซึ่งไม่จำเป็นต้องมีตัวอุปกรณ์สลาฟ ครบสามตามวงเครือข่ายย่อยก็สามารถที่จะติดต่อสื่อสารได้เช่นกัน แต่จะมีข้อจำกัดในเรื่องของความสามารถในการสื่อสารกันข้ามวงเครือ ข่ายย่อยได้ถ้าขาดอุปกรณ์สลาฟบางตัวไป

ไมโครคอนโทรลเลอร์มาสเตอร์นอกจากจะสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์สลาฟด้วยระบบไร้สายแล้ว ยังต้องเชื่อมต่อกับคอมพิวเตอร์เพื่อรอรับข้อมูลที่ส่งมาจากเครือข่ายอินเตอร์เน็ตอีกด้วย ไมโครคอนโทรลเลอร์มาสเตอร์จึงมีการเชื่อมต่อกับคอมพิวเตอร์ด้วยการสื่อสารข้อมูลอนุกรมหรือ UART

2.4 การสื่อสารข้อมูลอนุกรม (UART)

UART ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส สำหรับการสื่อสารข้อมูลอนุกรมบนคอมพิวเตอร์

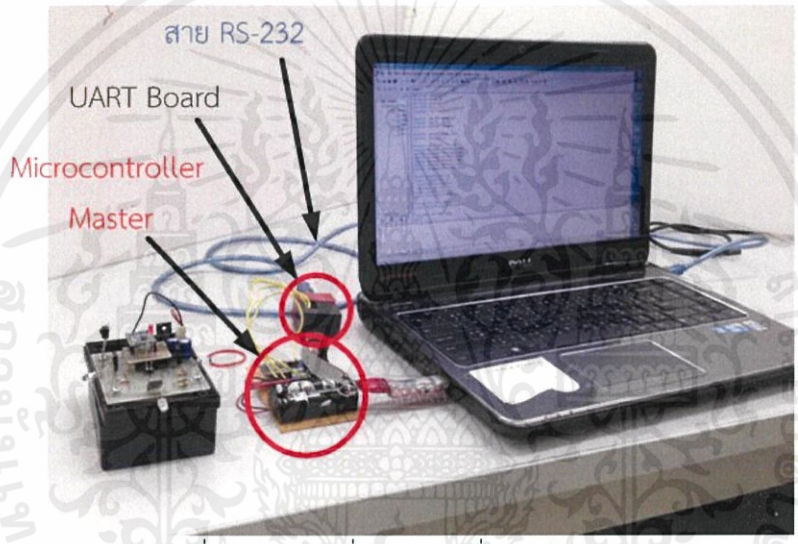
หน้าที่หลักของ UART คือ ทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัสแล้วส่งออก และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าคอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังทำการแจ้งข้อมูลอื่นๆ ให้คอมพิวเตอร์ทราบด้วย เช่น อัตราความเร็วในการรับส่งข้อมูล (Buad rate), รูปแบบการส่งข้อมูล, ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล (ผิดพลาดจากพาริตี้, เฟรมข้อมูล, โอเวอร์รัน) เป็นต้น

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

UART สามารถรับส่งข้อมูลได้ทั้งแบบฮาล์ฟดูเพล็กซ์ (Half Duplex) และฟูลดูเพล็ก (Full Doplex) โดยการส่งแบบ ฮาล์ฟดูเพล็ก เป็นการส่งแบบทิศทางเดียว และการส่งแบบฟูลดูเพล็กนั้นสามารถรับและส่งข้อมูลได้ในคราวเดียวกัน โดยส่วนใหญ่ UART จะใช้มาตรฐานการสื่อสารแบบ RS-232 โดยถ้าติดตั้งบนคอมพิวเตอร์ทั่วไปเรียกว่า คอมพอร์ต (Comport)

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)จะสร้างสัญญาณคลื่นพาหะและรวมเข้ากับสัญญาณข้อมูลเพื่อให้สัญญาณมีความแรงพอที่จะส่งผ่านสื่อกลางไปยังอีกจุดหนึ่งเมื่อถึงปลายทางจะมีอุปกรณ์ในการแยกสัญญาณคลื่นพาห์ออกเรียกวิธีการแยกสัญญาณนี้ว่า “การดีมอดูเลต” (Demodulation)

2.5.1 ประเภทของการมอดูเลต แบ่งเป็น 3 ประเภท คือ

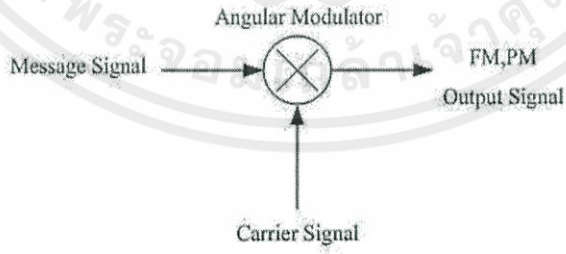
1. การมอดูเลตแอมพลิจูด (Amplitude Modulation หรือ AM)วิธีนี้แอมพลิจูด ของคลื่นพาห์จะเปลี่ยนแปลงตามสัญญาณของข้อมูลที่เข้ามา การมอดูเลตแบบ AM เป็นวิธีที่ง่ายที่สุดในการมอดูเลต แต่คุณภาพของสัญญาณไม่ดี มีความต้านทานสัญญาณรบกวนต่ำ เหมาะกับข้อมูลที่ไม่ต้องการคุณภาพมากนัก เช่น สัญญาณเสียง เป็นต้น

2. การมอดูเลตความถี่ (Frequency Modulation หรือ FM) วิธีการนี้เป็นการเปลี่ยนแปลงความถี่ของคลื่นพาห์ตามสัญญาณของข้อมูลที่เข้ามา การมอดูเลตแบบความถี่ให้คุณภาพที่ดีกว่าการมอดูเลตแบบแอมพลิจูด แต่ระบบจะซับซ้อนกว่า

3. การมอดูเลตเฟส (Phase Modulation หรือ PM) เป็นการมอดูเลตที่ใช้ในการเปลี่ยนแปลงเฟสของคลื่นพาห์ตามสัญญาณข้อมูลที่เข้ามา ทั้งคุณภาพของสัญญาณ และความซับซ้อนไม่ค่อยแตกต่างจากการมอดูเลตแบบความถี่ ข้อแตกต่างระหว่างการมอดูเลตแบบความถี่ กับการมอดูเลตแบบเฟส คือการมอดูเลตแบบเฟสใช้คลื่นพาห์เพียงความถี่เดียว

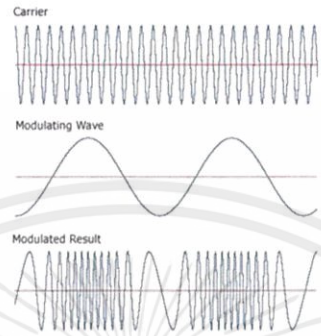
แต่การส่งสัญญาณด้วยรีโมตนั้นเป็นการมอดูเลชั่นทางความถี่ (Frequency Modulation : FM) ซึ่งการมอดูเลตเป็นการผสมสัญญาณข้อมูลข่าวสารเข้าไปกับสัญญาณคลื่นพาห์ (Carrier) ซึ่งสัญญาณนี้มีความถี่ที่เหมาะกับช่องสัญญาณนั้นๆ เพื่อให้ข้อมูลข่าวสารที่ส่งเข้าไปในช่องสัญญาณเดินทางได้ไกลมากขึ้น การเลือกวิธีการมอดูเลตขึ้นอยู่กับปัจจัยหลายประการ เช่น ชนิดของสัญญาณ แบนด์วิดท์ ประสิทธิภาพของระบบที่ต้องการและความต้านทานต่อสัญญาณรบกวน ซึ่งสามารถแบ่งการมอดูเลชั่นเชิงมุมออกเป็น 2 แบบด้วยกันคือ

1. การมอดูเลชั่นทางความถี่ (Frequency Modulation: FM)
2. การมอดูเลชั่นทางเฟส (Phase Modulation: PM)



รูปที่ 2.11 การมอดูเลชั่นเชิงมุม

การมอดูเลชั่นทางความถี่ (Frequency Modulation หรือ FM) ที่มอดูเลตแล้วจะมีแอมปลิจูดคงที่แต่ความถี่ของสัญญาณจะเปลี่ยนแปลงไปตามแอมปลิจูดของสัญญาณข้อมูลข่าวสาร ซึ่งการมอดูเลตทางความถี่จะให้คุณภาพที่ดีกว่าการมอดูเลตทางแอมปลิจูด (AM) แต่ระบบจะมีความซับซ้อนกว่า



รูปที่ 2.12 สัญญาณที่มอดูเลตด้วยความถี่

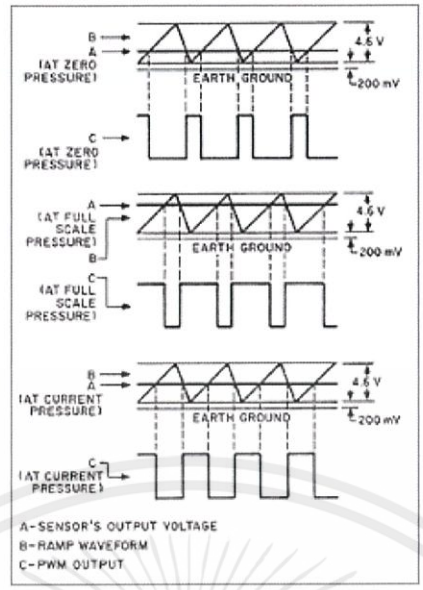
2.5.2 คุณสมบัติที่สำคัญของสัญญาณเอฟเอ็ม มีดังนี้

1. มีแอมปลิจูดคงที่ตลอด แต่ความถี่เปลี่ยนตามสัญญาณที่เข้ามอดูเลต
2. อัตราการเบี่ยงเบนความถี่ของสัญญาณพาหะมีค่าเท่ากับความถี่ของสัญญาณที่เข้ามา
3. ช่วงความถี่เบี่ยงเบนเป็นสัดส่วนกับแอมปลิจูดของสัญญาณที่เข้ามอดูเลต

2.5.3 ข้อเสียของการมอดูเลตแบบสัญญาณเอฟเอ็ม คือ

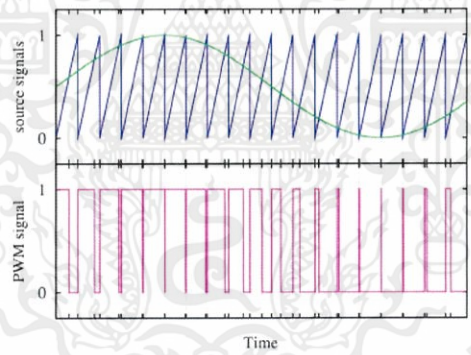
1. ต้องการแบนด์วิดท์ที่มีขนาดกว้าง เนื่องจากสัญญาณข้อมูลมีหลายความถี่
2. คุณภาพดีกว่าการมอดูเลตแบบ AM แต่การทำงานจะซับซ้อนกว่า

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



รูปที่ 2.13 สัญญาณพัลส์ที่มีความกว้างคงที่

แต่ถ้านำสัญญาณที่มีการเปลี่ยนแปลงอยู่ตลอดเวลาความกว้างของพัลส์ก็จะเปลี่ยนแปลงตามไปด้วย ดังรูป



รูปที่ 2.14 สัญญาณพัลส์ที่เปลี่ยนแปลงตลอดเวลา

2.6 ภาษาคอมพิวเตอร์ในการเขียนสั่งงานไมโครคอนโทรลเลอร์

มนุษย์ใช้ภาษาในการสื่อสารมาตั้งแต่สมัยโบราณ การใช้ภาษาเป็นเรื่องที่มนุษย์พยายามถ่ายทอดความคิดและความรู้สึกต่างๆเพื่อการโต้ตอบและสื่อความหมายภาษาที่ใช้ติดต่อสื่อสารในชีวิตประจำวัน 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 หมายถึงบวกและ 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) ซึ่งแตกต่างไปตามเครื่องคอมพิวเตอร์แต่ละชนิดดังนั้นแอสเซมเบลอร์ของเครื่องชนิดหนึ่งจะไม่สามารถใช้แปลโปรแกรมภาษาแอสเซมบลีของเครื่องชนิดอื่น ๆ ได้ภาษาแอสเซมบลีนี้ยังคงใช้ยาก เพราะผู้เขียนโปรแกรมจะต้องเข้าใจในการทำงานของเครื่องคอมพิวเตอร์อย่างละเอียด ต้องรู้ว่าจำนวนที่จะนำมาคำนวณนั้นอยู่ตำแหน่งใดในหน่วยความจำ เช่นกับการเขียนโปรแกรมเป็นภาษาเครื่อง ภาษาแอสเซมบลีจึงมีผู้ใช้น้อยและมักจะใช้ในกรณีที่ต้องการควบคุมการทำงานภายในของตัวเครื่องคอมพิวเตอร์

2.6.1.3 ภาษาระดับสูง (High Level Language)

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

ภาษาระดับสูงที่จะกล่าวถึงในที่นี้ ได้แก่

1. **ภาษาฟอร์แทรน (FORmulaTRANstation : FORTRAN)**จัดเป็นภาษาระดับสูงที่เก่าแก่ที่สุด ได้รับการคิดค้นขึ้นเป็นครั้งแรก ราว พ.ศ. 2497 โดยบริษัท ไอบีเอ็ม เป็นภาษาที่เหมาะสำหรับงานที่ต้องการการคำนวณ 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. 2502ต่อมาได้รับการปรับปรุงจากคณะกรรมการซึ่งเป็นตัวแทนของหน่วยงานธุรกิจและรัฐบาลของสหรัฐอเมริกาเป็นภาษาโคบอลมาตรฐานในปี พ.ศ.2517 เป็นภาษาที่เหมาะสมสำหรับงานด้านธุรกิจ เครื่องคอมพิวเตอร์ขนาดใหญ่ส่วนมากมีโปรแกรมแปลภาษาโคบอล

3. ภาษาเบสิก (Beginner's All – purpose Symbolic Instruction Code : BASIC) เป็นภาษาที่ได้รับการคิดขึ้นเป็นครั้งแรกที่วิทยาลัยดาร์ทมัธ (Dartmouth College) และเผยแพร่เป็นทางการในปี พ.ศ.2508ภาษาเบสิกเป็นภาษาที่สร้างขึ้นโดยมีจุดประสงค์เพื่อใช้สอนเพื่อใช้สอนเขียนโปรแกรมแทนภาษาคอมพิวเตอร์ภาษาอื่น ซึ่งมีขนาดใหญ่และต้องใช้หน่วยความจำสูงในการทำงาน ซึ่งไม่เหมาะกับเครื่องคอมพิวเตอร์ในสมัยนั้น ภาษาเบสิกเป็นภาษาที่มีขนาดเล็กเป็นตัวแปลภาษาชนิดที่เรียกว่าอินเทอร์พรีเตอร์นอกจากนี้ภาษาเบสิกเป็นภาษาที่ง่ายต่อการเขียนซึ่งผู้เขียนจะสามารถนำไปประยุกต์กับการแก้ปัญหาต่างๆ ได้ทุกสาขาวิชาผู้ที่เพิ่งฝึกเขียนโปรแกรมใหม่ๆ หรือผู้ที่ไม่ใช่นักเขียนโปรแกรมมืออาชีพ แต่เป็นเพียงวิศวกรหรือนักวิจัย จะสามารถหัดเขียนโปรแกรมภาษาเบสิกได้ในเวลาไม่นานนัก ปกติภาษาเบสิกส่วนใหญ่ใช้กับไมโครคอมพิวเตอร์

4. ภาษาปาสคาล (Pascal) ตั้งชื่อตามนักคณิตศาสตร์ชาวฝรั่งเศส ชื่อเบลส ปาสคาล (Blaise Pascal) ซึ่งเป็นผู้ผลิตเครื่องคิดเลขโดยใช้เฟืองหมุน ภาษาปาสคาลคิดขึ้นในปี พ.ศ. 2514 โดยนิคลอสเวียซ (Niklaus Wirth) ศาสตราจารย์วิชาคอมพิวเตอร์ชาวสวิต ภาษาปาสคาลได้รับการออกแบบให้ใช้ง่ายและมีโครงสร้างที่ดี จึงเหมาะกับการใช้สอนหลักการเขียนโปรแกรม ปัจจุบันภาษาปาสคาลยังคงได้รับความนิยมใช้ในการเรียนเขียนโปรแกรมคอมพิวเตอร์

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

6. ภาษาวิชวลเบสิก (Visual Basic)เป็นภาษาที่พัฒนาต่อมาจากภาษาเบสิก ใช้ไวยากรณ์บางส่วนของภาษาเบสิกในการเขียนโปรแกรมแต่มีแนวคิดและวิธีการพัฒนาโปรแกรมที่แตกต่างจากภาษาเบสิกโดยสิ้นเชิง รวมทั้งการใช้เนื้อที่ในหน่วยความจำก็แตกต่างกันมาก ทั้งนี้เนื่องจากภาษาวิชวลเบสิกใช้แนวคิดที่ต่างออกไป

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

8. ภาษาจาวา (Java) พัฒนาขึ้นในปี พ.ศ. 2534 โดยบริษัทซันไมโครซิสเตมส์ เป็นภาษาที่ได้รับความนิยมสูงมาโดยตลอด เนื่องจากเป็นภาษาที่มีความยืดหยุ่นสูง สามารถเขียนโปรแกรมและใช้งานได้บนเครื่องคอมพิวเตอร์ทุกประเภทและระบบปฏิบัติการทุกรูปแบบ ในช่วงแรกที่เริ่มมีการนำภาษาจาวามาใช้งานจะเป็นการใช้งานบนเครือข่ายอินเทอร์เน็ต เป็นภาษาที่เน้นการทำงานบนเว็บ แต่ปัจจุบันสามารถสามารถนำมาประยุกต์สร้างโปรแกรมใช้งานทั่วไปได้ นอกจากนี้ เมื่อเทคโนโลยีของการสื่อสารก้าวหน้าขึ้น จนกระทั่งเครื่องคอมพิวเตอร์ปาล์มท็อป หรือแม้แต่โทรศัพท์เคลื่อนที่สามารถเชื่อมต่อเข้าสู่ระบบอินเทอร์เน็ตและใช้งานระบบเวิลด์ไวด์เว็บได้ ภาษาจาวาก็สามารถสร้างส่วนที่เรียกว่า แอปเพล็ต (Applet) ให้อุปกรณ์อิเล็กทรอนิกส์ที่กล่าวข้างต้นเรียกใช้งานจากเครื่องที่เป็นแม่ข่าย (Server) ได้

9. ภาษาเดลฟาย (Delphi) เป็นภาษาที่ได้รับความนิยมภาษาหนึ่ง แนวคิดในการเขียนโปรแกรมภาษาเดลฟายเหมือนกับแนวคิดในการเขียนโปรแกรมภาษาวิชวลเบสิก คือเป็นการเขียนโปรแกรมเชิงจินตภาพ แต่ภาษาพื้นฐานที่ใช้ในการเขียนโปรแกรมจะเป็นภาษาปาสคาลการเขียนโปรแกรมเชิงจินตภาพนี้มีคอมโพเนนต์ (Component) ที่สามารถใช้เป็นส่วนประกอบเพื่อสร้างส่วนติดต่อผู้ใช้ที่เป็นแบบกราฟิก ทำให้ซอฟต์แวร์ที่พัฒนามีความน่าสนใจและใช้งานง่ายขึ้นการเขียนโปรแกรมด้วยภาษาเดลฟายจึงเป็นที่นิยมในการนำไปพัฒนาเป็นโปรแกรมใช้งานมากรวมทั้งภาษาปาสคาลเป็นภาษาที่เข้าใจง่าย เหมาะแก่การนำมาใช้สอนเขียนโปรแกรม

ภาษาแอสเซมบลีจัดได้ว่าเป็นภาษาที่มีมาคู่กับไมโครคอนโทรลเลอร์ทุกตระกูล ไม่ว่าจะเป็น MCS-52, PIC, 68HC11 หรือจะเป็นไมโครคอนโทรลเลอร์ ST ก็ตาม แต่ในปัจจุบันภาษา C กับได้รับความนิยมเพิ่มมากขึ้นในการนำมาใช้พัฒนาโปรแกรมสำหรับไมโครคอนโทรลเลอร์รวมทั้ง MCS-51 เนื่องจาก ไมโครคอนโทรลเลอร์ nRF24LE1 ที่ใช้ในปริญญานิพนธ์นี้ เป็นไมโครคอนโทรลเลอร์ตระกูลเดียวกันกับ MCS-51 มีสถาปัตยกรรมภายในที่คล้ายคลึงกัน การเขียนซอฟแวร์ควบคุมหรือการใช้งานฟังก์ชั่นต่าง ๆ จึงคล้ายคลึงกัน

สาเหตุที่ภาษาซีได้รับความนิยม คือ

1. ภาษา C เป็นภาษาโครงสร้าง ง่านต่อการทำความเข้าใจ ปรับปรุง และพัฒนาต่อ

2. ภาษา C ใช้หลังการเขียนโปรแกรมแบบสมัยใหม่มีทั้งที่เป็นแบบโครงสร้างและออบเจ็กต์

3. ภาษา C เป็นภาษามาตรฐานไม่ขึ้นกับฮาร์ดแวร์(ไมโครคอนโทรลเลอร์)มีความยืดหยุ่นในการโยกย้ายไปใช้งานกับไมโครคอนโทรลเลอร์ตระกูลอื่นได้ง่าย

4. ตัวคอมไพเลอร์และลิงเกอร์ของภาษา C มีการพัฒนาอยู่ตลอดเวลาจนมีความสามารถสูง

5. ภาษา C มีเครื่องมือในการพัฒนาโปรแกรมที่เรียกว่า Integrated Development environment (IDE) ก้าวหน้ามาก

6. ภาษา C มีความเข้าใจกันได้กับภาษาแอสเซมบลีในระดับซอร์สโค้ดทั้งแบบ Inline assembly และการลิงก์ (Link) โปรแกรมเข้ากันด้วย

ด้วยเหตุผลดังกล่าวจึงทำให้ภาษา C เป็นตัวเลือกระดับต้นๆในการพัฒนาโปรแกรมสำหรับการใช้งานไมโครคอนโทรลเลอร์ในปัจจุบัน และโปรแกรมที่เลือกใช้ในการเขียนภาษาซีในปริญญานิพนธ์นี้ 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 ไดเร็กตีฟ (Preprocessor directives)

เป็นชุดคำสั่งที่ใช้ในการจัดการเตรียมข้อมูลไว้สำหรับการประมวลผลก่อนที่จะมีการคอมไพล์ สำหรับคอมไพเลอร์ C51 จะมีไดเร็กตีฟทั้งที่เป็นมาตรฐานเดียวกับภาษา C และที่เพิ่มเติมเฉพาะสำหรับ C51

2.6.2.2 การประกาศ (Declarations)

ก่อนใช้งานตัวแปรหรือฟังก์ชั่น ต้องมีการประกาศและสร้างตัวแปลหรือฟังก์ชั่น ขึ้นมาก่อนการใช้งาน

2.6.2.3 การกำหนดค่า (Definition)

การประกาศและจองหน่วยความจำ หรือการกำหนดค่าให้กับตัวแปรหรือฟังก์ชั่น

2.6.2.4 นิพจน์ (Expressions)

เป็นการกระทำระหว่างตัวดำเนินการ กับตัวถูกดำเนินการ เพื่อให้เกิดค่าใดค่าหนึ่ง

2.6.2.5 สเตตเมนต์ (Statements)

คำสั่งการทำงาน คือ คำสั่งที่ใช้ในการทำงานตามความต้องการของผู้เขียนโปรแกรม

2.6.2.6 ฟังก์ชั่น (Function)

เป็นส่วนประกอบของโปรแกรมที่กำหนดให้ทำงานอย่างใดอย่างหนึ่งจนเสร็จสิ้น ภายในฟังก์ชั่นโดยภายในฟังก์ชั่นจะประกอบไปด้วย การประกาศใช้งานตัวแปร การกำหนดค่าให้กับตัวแปร นิพจน์ แบะคำสั่งการทำงาน

2.6.2.7 ฟังก์ชั่น main ( ) (Main Function)

ฟังก์ชั่น main ( ) เป็นฟังก์ชั่นที่ต้องมีการประกาศทุกครั้งเมื่อมีการใช้โปรแกรมภาษา C เพราะการทำงานโปรแกรมจะเริ่มต้นที่ฟังก์ชั่นนี้ และเป็นฟังก์ชั่นที่ใช้ในการเรียกฟังก์ชั่นอื่นๆในการทำงาน

2.6.3 การประกาศตัวแปรรีจิสเตอร์หน้าที่พิเศษ (SFR)

ชนิดข้อมูลแบบ SFR

ชนิดข้อมูลและการประกาศใช้งานตัวแปรแบบ SFR เพื่อเป็นการเข้าถึงละใช้งานรีจิสเตอร์หน้าที่พิเศษการประกาศใช้งานตัวแปรชนิดข้อมูล SFR นั้นจะมีการกำหนดในลักษณะแบบเดียวกับตัวแปรภาษา C ทั่วไป แต่เมื่อมีการประกาศใช้งานแล้วจะต้องกำหนดค่าทันที (ตำแหน่งแอดเดรส)

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

2.6.3.1 การใช้งานรีจิสเตอร์อินเตอร์รัปต์จากสัญญาณภายนอก

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

การอินเตอร์รัปต์จากสัญญาณภายนอกจะเป็นการตรวจสอบสัญญาณที่รับจาก

ภายนอกตัว ขณะที่สัญญาณเริ่มมีการเปลี่ยนแปลงจากลอจิกสูงไปเป็นลอจิกต่ำ ลักษณะของการอินเตอร์รัปต์แบบนี้จะมี 2 ลักษณะคือ

1. การอินเตอร์รัปต์จากสัญญาณภายนอกที่ P3.2 หรืออินเตอร์รัปต์สัญญาณ INTO
2. การอินเตอร์รัปต์จากสัญญาณภายนอกที่ P3.3 หรืออินเตอร์รัปต์สัญญาณ INTI

การใช้งานอินเตอร์รัปต์จากสัญญาณภายนอก เป็นการนับสัญญาณจากภายนอกผ่านพอร์ค P3 บิดที่ 2 (P3.2) ซึ่งเป็นขาของสัญญาณอินเตอร์รัปต์ INTO และพอร์ค P3 (P3.3) ซึ่งเป็นขาของสัญญาณอินเตอร์รัปต์ทั้งสองสามารถที่จะกำหนดการเกิดอินเตอร์รัปต์ได้ 2 รูปแบบคือ พิจารณาการเปลี่ยนระดับสัญญาณสูงเป็นระดับสัญญาณลอจิกต่ำ และเปลี่ยนจากลอจิกสูงเป็นลอจิกต่ำ

2.6.3.1.1 รีจิสเตอร์และบิตที่เกี่ยวข้องกับการอินเตอร์รัปต์สัญญาณภายนอก

สามารถสรุปได้ดังนี้

1. รีจิสเตอร์ TCON (Timer/Countert Control register) สามารถที่จะอ้างถึงแบบบิดได้ ซึ่งประกอบไปด้วยบิตที่เกี่ยวข้องกับการใช้วานอินเตอร์รัปต์สัญญาณภายนอกได้แก่

- บิต IE1 เป็นบิตแฟล็กแสดงการอินเตอร์รัปต์ของ INT1
- บิต IT1 เป็นบิตเลือกประเภทสัญญาณอินเตอร์รัปต์ของ INT1
- บิต IE0 เป็นบิตแฟล็กแสดงการอินเตอร์รัปต์ของ INT0
- บิต IT0 เป็นบิตเลือกประเภทสัญญาณอินเตอร์รัปต์ของ INT0

2. รีจิสเตอร์ IE (**Interrupt enable register**) สามารถที่จะอ้างถึงแบบบิดได้ ซึ่งประกอบไปด้วยบิตที่เกี่ยวข้องกับการใช้วานอินเตอร์รัปต์สัญญาณภายนอกได้แก่

- บิต EA บิตเอ็นเอเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์โดยรวม
- บิต EX1 บิตเอ็นเอเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ INT1
- บิต EX0 บิตเอ็นเอเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ INT0

2.6.3.2 การใช้งานรีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานพอร์ตอนุกรม

พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51สามารถกำหนดการทำงานได้ 4 โหมด หรือ 4 รูปแบบ ดังนี้

1. โหมด 0 เป็นโหมดสำหรับการขยายพอร์ตอินพุตเอาต์พุต ทำงานร่วมกับไอซีรีจิสเตอร์ภายนอก

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

สำหรับการสื่อสารข้อมูลแบบอนุกรมในบทนี้จะเน้นไปที่การทำงานในโหมด1 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) ใช้กำหนดโหมดการทำงานของพอร์ตอนุกรมโดยกำหมดค่าในบิต SM0 แบะ 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

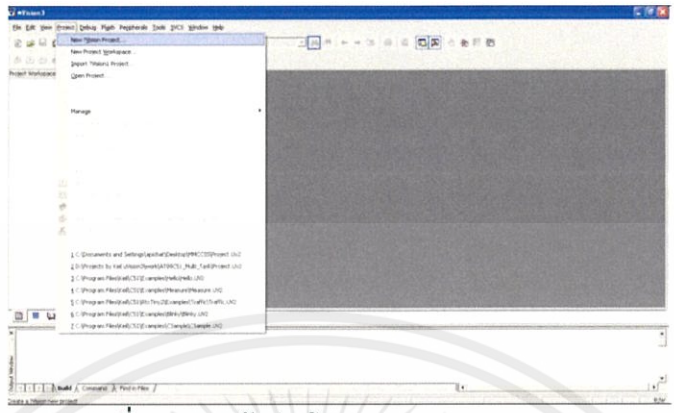
2.7 การใช้โปรแกรม KeiluVision 3

โปรแกรม KeiluVision 3 หรือเรียกว่า Keil C51 เป็นโปรแกรมที่สร้างขึ้นโดยบริษัท Keil Software สามารถดาวน์โหลดได้ที่ http://www.keil.com ซึ่งเป็นเวอร์ชั่นทดลองใช้

โปรแกรม KeiluVision 3 ช่วยในการเขียนโปรแกรม C51 ได้ง่ายและรวดเร็วยิ่งขึ้น โดยสามารถแปลงภาษา C51 เป็นโค้ดเฮ็กซ์

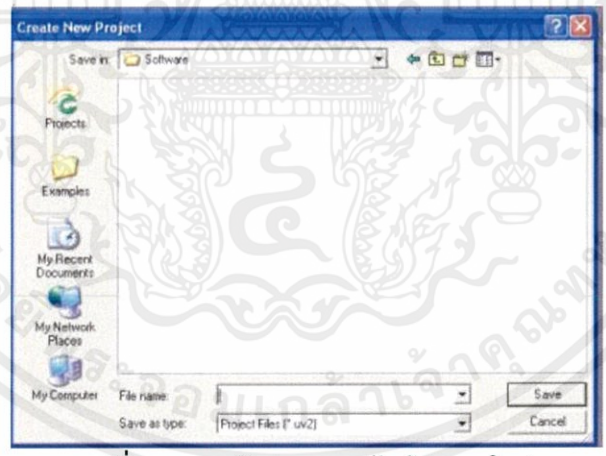
2.7.1 ขั้นตอนที่ 1 การสร้างโปรเจค

1. คลิกเมนู Project >New ?Vision Project



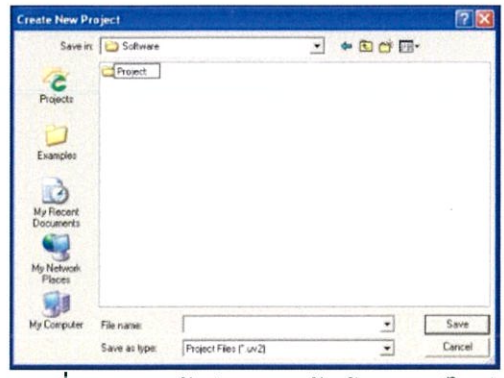
รูปที่ 2.15 หน้าต่างโปรแกรม KeiluVision 3

เมื่อเลือกคำสั่ง New ?Vision Project จะปรากฏหน้าต่าง Create New Project ขึ้นมา โดยหน้าต่างการสร้างโปรเจคใหม่จะกำหนดให้เราใส่ชื่อโปรเจค สำหรับใช้งาน ซึ่งในการทำงานอาจมีหลายโปรเจค เพื่อความง่ายในการใช้งานควรสร้างโฟลเดอร์ (folder) ขึ้นมาใหม่แต่ละโปรเจคโดยเฉพาะ



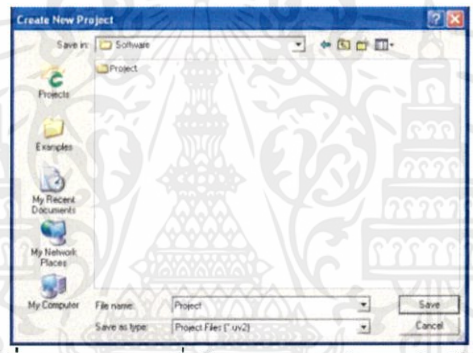
รูปที่ 2.16 หน้าต่างการสร้างโปรเจคใหม่

2. สร้างโฟลเดอร์ใหม่ ในตัวอย่างตั้งชื่อโฟลเดอร์ว่า Project



รูปที่ 2.17 หน้าต่างการสร้างโฟลเดอร์ใหม่

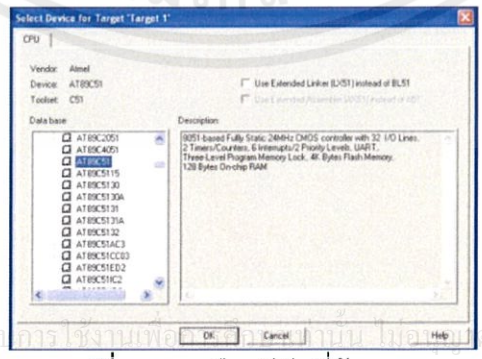
3. คลิกเข้าในโฟลเดอร์ที่สร้างใหม่ เพื่อสร้างโปรเจคในโฟลเดอร์นั้น
4. กรอกชื่อโปรเจคที่ช่อง File name:
5. คลิกปุ่มSaveเพื่อบันทึกไฟล์โปรเจค



รูปที่ 2.18 กรอกชื่อโปรเจคและทำการบันทึก

หลังจากที่บันทึกโปรเจคแล้ว โปรแกรม uVision 3 จะแสดงหน้าต่าง Select Device for Target 'Target1' ขึ้นมา เพื่อให้เราเลือกซีพียูที่จะใช้งานจากอุปกรณ์ฐานข้อมูล

6. เลือกซีพียูที่ต้องการ โดยในตัวอย่างเลือกซีพียู AT89C51 ของบริษัท ATMEL

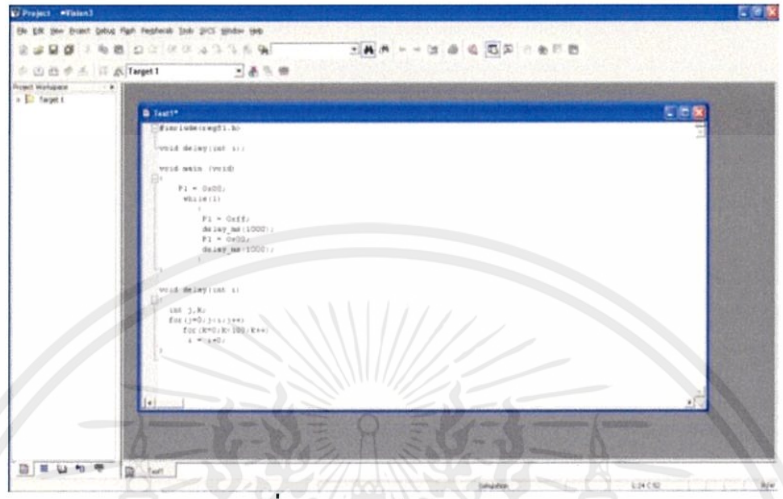


รูปที่ 2.19 เลือกซีพียูที่ต้องการ

2.7.2 ขั้นตอนที่ 2 เขียนโค้ดโปรแกรม

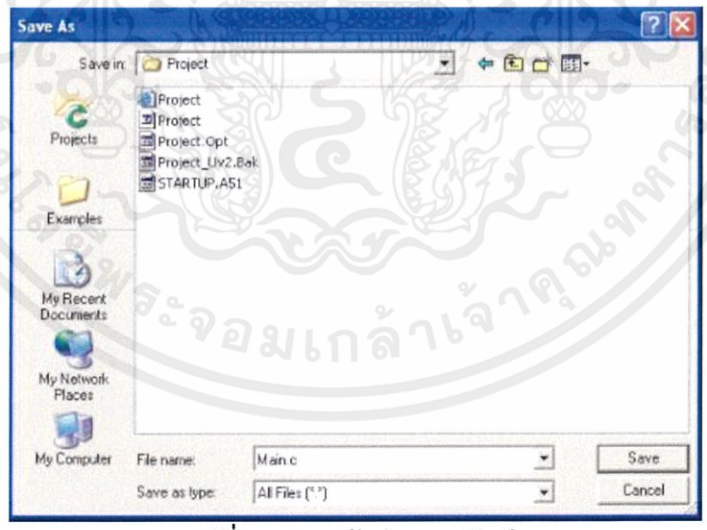
1. คลิกเมนู File > New เพื่อสร้างไฟล์ที่จะใช้เขียนโค้ดโปรแกรม

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



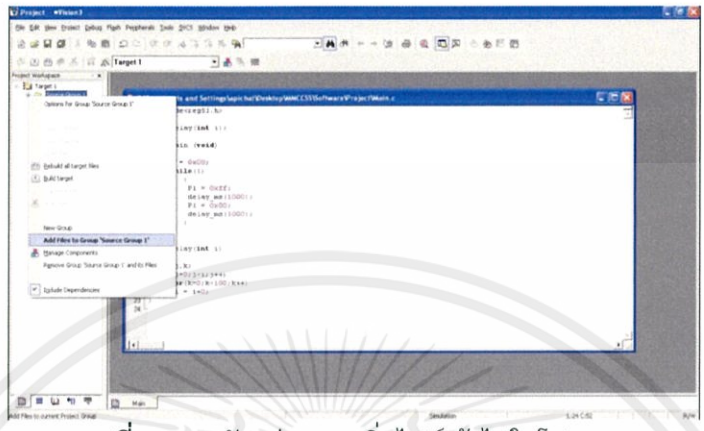
รูปที่ 2.21 หน้าต่างการบันทึก

4. กรอกชื่อไฟล์ในช่อง File Name: โดยในตัวอย่างตั้งชื่อไฟล์เป็น Main.c

5. คลิกปุ่มSave เพื่อบันทึกโปรแกรมเมื่อสร้างโค้ดโปรแกรมเสร็จเรียบร้อยแล้ว จากนั้นให้เพิ่มไฟล์เข้าไปในโปรเจคของเรา

6. คลิกขวาที่ Source Group1 ตรงหน้าต่างพื้นที่การทำงานของโปรเจคถ้าหา Source Group1 ไม่เจอก็ให้คลิกเครื่องหมายบวกข้างหน้า Target 1

7. คลิกเลือกเมนู Add Files to Group 'Source Group1'

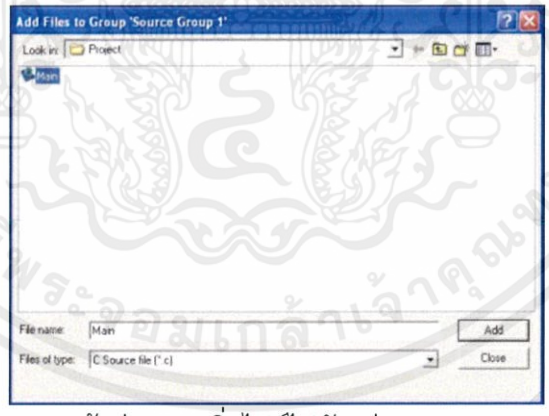


รูปที่ 2.22 ตัวอย่างการเพิ่มไฟล์เข้าไปในโปรเจค

จากนั้นหน้าต่างการเพิ่มไฟล์ไปยังกลุ่ม 'Source Group1' จะปรากฏขึ้นมาให้เราเลือกไฟล์ที่จะเพิ่มเข้าไปในโปรเจค

8. คลิกเลือกไฟล์ที่จะเพิ่มเข้าไปในโปรเจค

9. คลิกปุ่มAddเพื่อเพิ่มไฟล์

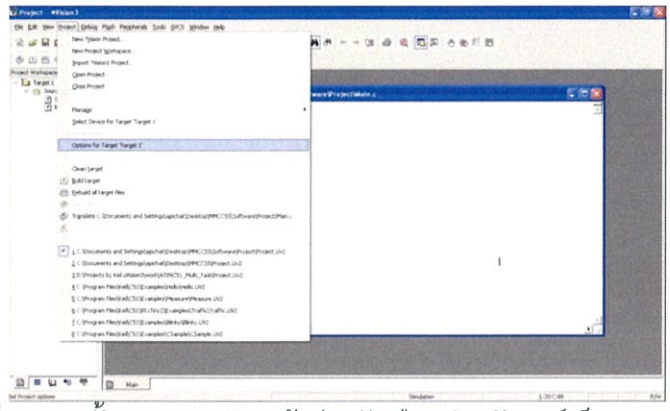


รูปที่ 2.23 หน้าต่างการเพิ่มไฟล์ไปยังกลุ่ม 'Source Group1'

2.7.3 ขั้นตอนที่ 3 การเลือกตัวเลือกเครื่องมือสำหรับซีพียู

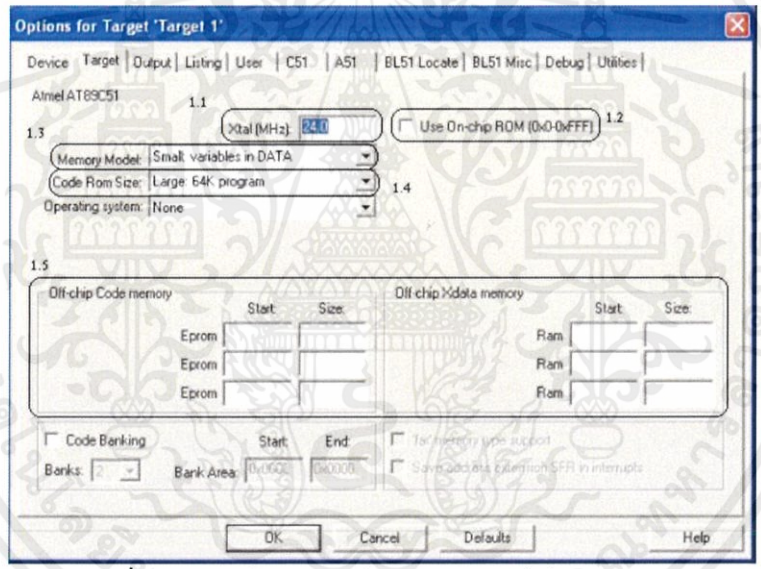
โปรแกรม uVision 3 ให้เราสามารถกำหนดค่าพารามิเตอร์ต่างๆ ที่จำเป็นสำหรับซีพียูที่เราจะใช้งาน โดยผ่านทางหน้าต่างOption for Target ก่อนอื่นต้องเลือกซีพียูที่ต้องการใช้งานเสียก่อน ดังแสดงต่อไปนี้

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 Xtalที่ใช้

1.2 กำหนดว่าจะให้ใช้พื้นที่หน่วยความจำเก็บโปรแกรมภายในชิปหรือไม่ ปกติจะกำหนดใช้ภายในชิปไว้อัตโนมัติอยู่แล้ว ซึ่งกำหนดภายในไฟล์ STARTUP.A51

1.3 กำหนดเมมโมรี่โมเดลหรือโมเดลพื้นที่หน่วยความจำ ปกติจะกำหนดเป็นSmall

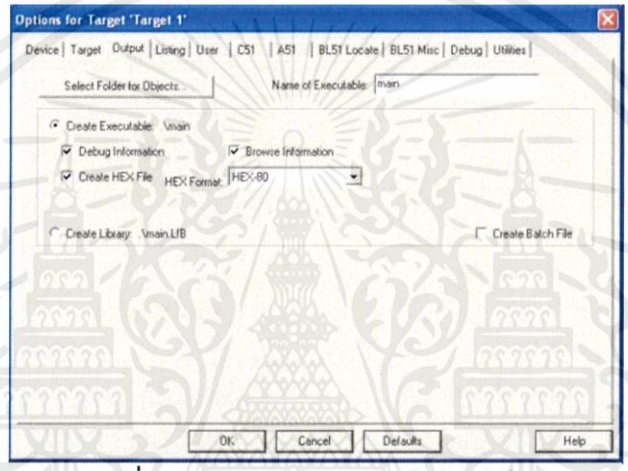
1.4 กำหนดโค้ดรอมไซส์สำหรับขนาดพื้นที่หน่วยความจำที่จะเก็บโค้ดโปรแกรม ซึ่งเราอาจจะเก็บไว้สูงสุด 64K

1.5 กำหนดพื้นที่ใช้งานหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลไว้ภายนอกชิป ทั้งหมด โดยให้เราใส่แอดเดรสเริ่มต้นและขนาดพื้นที่หน่วยความจำที่ใช้ และใช้งานได้ถูกต้องควรทำการเปิดใช้งานการใช้หน่วยความจำภายนอกในไฟล์ STARTUP.A51 ด้วย

2. คลิกแถบ Output เพื่อกำหนดค่าเอาต์พุต

3. ใส่ชื่อไฟล์เอาต์พุต (ไฟล์นามสกุล.hex) ในช่อง Nane of Executable เพื่อให้ง่าย ควรใส่ชื่อให้เหมือนชื่อไฟล์โค้ดโปรแกรม ถ้าไม่กำหนด โปรแกรม uVision 3 จะตั้งชื่อเดียวกับชื่อโปรเจค

4. กำหนดว่าจะให้โปรแกรม uVision 3 สร้างไฟล์เฮ็กซ์หรือไม่ ถ้าต้องการให้คลิกที่ช่องว่างข้างหน้า

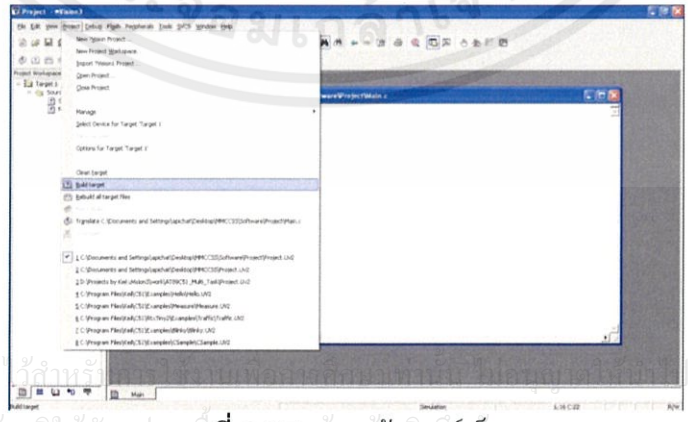


รูปที่ 2.26 การกำหนดการสร้างไฟล์เฮ็กซ์

2.7.4 ขั้นตอนที่ 4 สร้างโปรเจคเพื่อให้ได้ไฟล์เฮ็กซ์

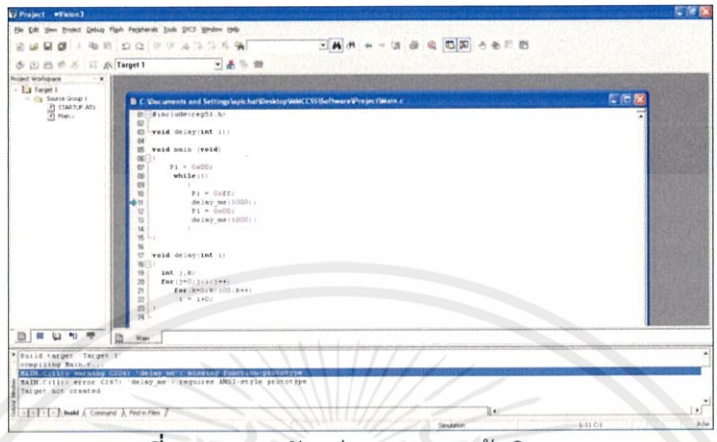
ในขั้นตอนนี้เป็นการแปลงโค้ดโปรแกรมไปเป็นไฟล์เฮ็กซ์ซึ่งมีขั้นตอนดังต่อไปนี้

1. คลิกเมนู Project > Build target



รูปที่ 2.27 การสร้างทาร์เก็ต

ถ้ามีข้อผิดพลาดเกิดขึ้นโปรแกรม uVision จะแสดงข้อผิดพลาดและข้อความเตือนให้รับทราบที่หน้า Build ของหน้าต่าง Output Windowและถ้าเราดับเบิลคลิกบรรทัดที่แสดงข้อผิดพลาดก็จะกระโดดไปยังโค้ดโปรแกรมที่ผิดพลาดนั้นทันที



รูปที่ 2.28 รูปตัวอย่างการแสดงข้อผิดพลาด

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

```
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).
```

ผลลัพธ์ที่ได้ก็คือไฟล์เฮ็กซ์หรือไฟล์นามสกุล .Hex นั้นเอง โดยไฟล์เฮ็กซ์ที่ได้จะเก็บอยู่ในโฟลเดอร์โปรเจคที่ใช้งานปัจจุบัน เราสามารถนำไฟล์นี้ไปโหลดเข้าชิปไมโครคอนโทรลเลอร์ได้เลย

2.8 Appserv

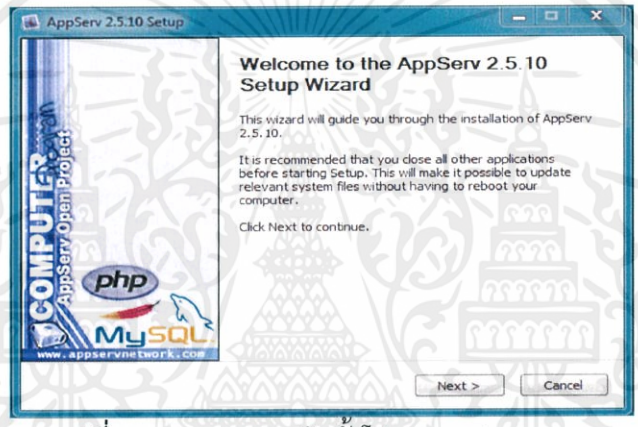
ในการสั่งงานจากอินเตอร์เน็ต ปริญญานิพนธ์นี้ได้เลือกใช้โปรแกรม AppServ มาประยุกต์ใช้ในการสั่งงานจากเครือข่ายอินเตอร์เน็ต เนื่องจากมีการใช้งานที่ง่าย หลักการใช้งานคร่าว ๆ คือ เมื่อได้รับคำสั่งจากหน้าเว็ปเพจ โปรแกรม Appserv จะส่งข้อมูลจากอินเตอร์เน็ตมายังเครื่องคอมพิวเตอร์ที่เป็นเซิพเวอร์ และส่งข้อมูลออกทางพอร์ตที่ได้เชื่อมต่อกับไมโครคอนโทรลเลอร์มาสเตอร์ไว้ จากนั้นไมโครคอนโทรลเลอร์มาสเตอร์ จึงเริ่มทำการส่งข้อมูลไปยัง ตัวสลาฟ ให้ทำการควบคุมเครื่องใช้ไฟฟ้าที่ต้องการต่อไป

AppServ คือโปรแกรมที่รวบรวมเอา Open Source Software หลายๆ อย่างมารวมกัน โดยมี Package หลักดังนี้

- Apache
- PHP
- MySQL
- phpMyAdmin

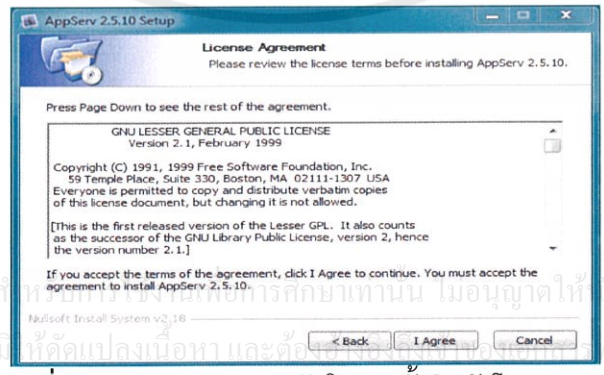
2.8.1 ขั้นตอนการติดตั้งโปรแกรม Appserv มีดังต่อไปนี้

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



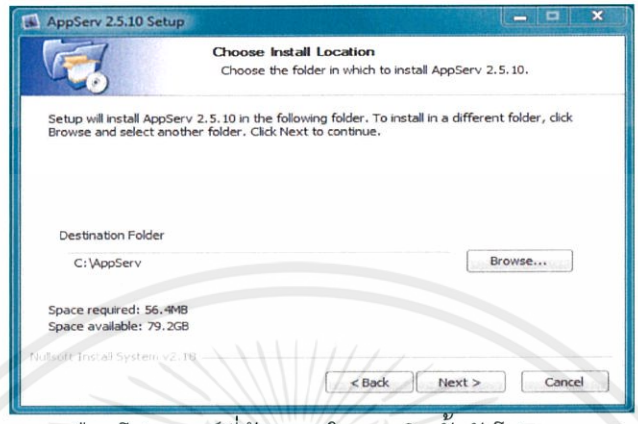
รูปที่ 2.29 แสดงการติดตั้งโปรแกรม Appserv

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



รูปที่ 2.30 แสดงการยอมรับในการตั้งติดตัวโปรแกรม

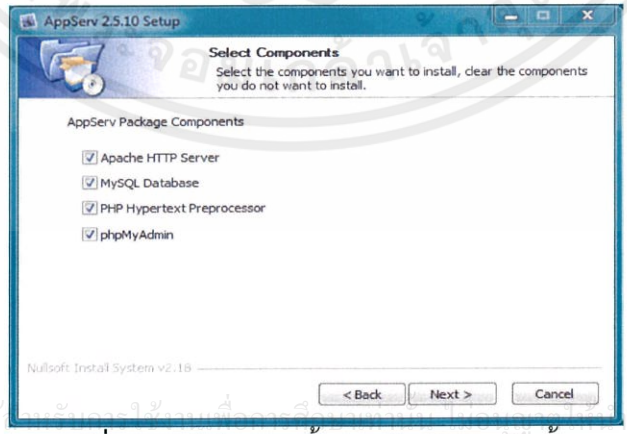
3. ขั้นตอนการเลือกปลายทางที่ต้องการติดตั้ง โดยค่าเริ่มต้นปลายทางที่ติดตั้งจะเป็น C:\AppServ หากต้องการเปลี่ยนปลายทางที่ติดตั้ง ให้กด Browse แล้วเลือกปลายทางที่ต้องการ ตามรูปที่ 5 เมื่อเลือกปลายทางเสร็จสิ้น ให้กดปุ่ม Next เพื่อเข้าสู่ขั้นตอนการติดตั้งขั้นต่อไป



รูปที่ 2.31 เลือกโฟลเดอร์ที่ต้องการในการติดตั้งตัวโปรแกรม Appserv

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

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

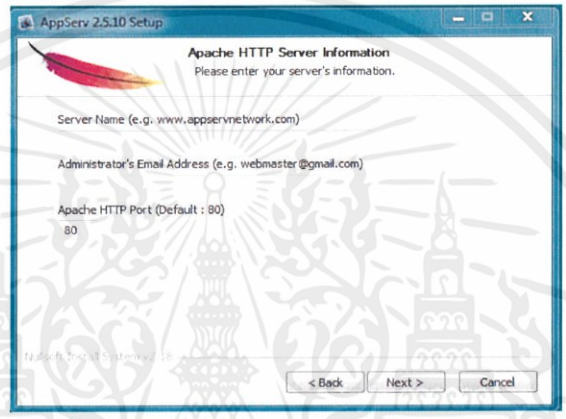


รูปที่ 2.32 โปรแกรมทั้งหมดสำหรับการติดตั้ง

5. กำหนดค่าคอนฟิกของ Apache Web Server มีอยู่ด้วยกันทั้งหมด 3 ส่วน คือ

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

โดยทั่วไปแล้ว Protocol HTTP นั้นจะมีค่าหลักคือ 80 หากว่าท่านต้องการหลีกเลี่ยงการใช้ Port80 ก็สามารถแก้ไขได้ หากมีการเปลี่ยนแปลง Port การเข้าใช้งาน Web Server แล้ว ทุกครั้งที่เรียกใช้งานเว็บไซต์ จำเป็นที่ต้องระบุหมายเลข Port ด้วย เช่นหากเลือกใช้ Port 99 ในการเข้าเว็บไซต์ต้องใช้ http://www.appservnetwork.com:99 จึงจะเข้าใช้งานได้

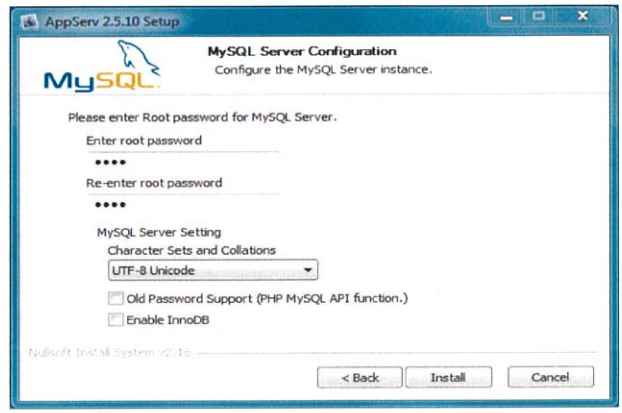


รูปที่ 2.33 กำหนด Server Name , Email Address และ Port

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

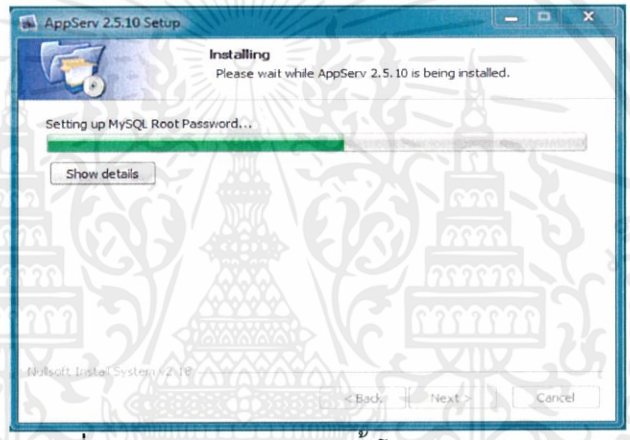
Character Sets ใช้ในการกำหนดค่าระบบภาษาที่ใช้ในการจัดเก็บฐานข้อมูล, เรียงลำดับฐานข้อมูล,Import ฐานข้อมูล, Export ฐานข้อมูล และ ติดต่อฐานข้อมูล

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



รูปที่ 2.34 กำหนด User , Password สำหรับการติดต่อตัวโปรแกรม

7. หลังจากนั้นรอให้ติดตั้งโปรแกรม จนเสร็จ

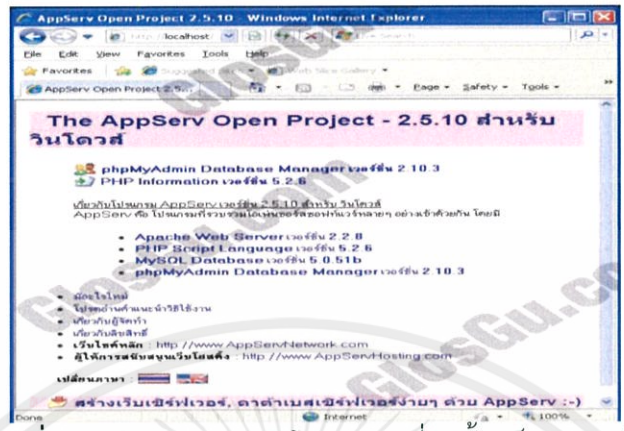


รูปที่ 2.35 กระบวณการติดตั้งโปรแกรม Appserv

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 แอพพลิเคชั่นเซิร์ฟเวอร์) ออกจากเครื่องคอมพิวเตอร์ที่ติดตั้ง สามารถทำได้โดยวิธีการดังต่อไปนี้

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

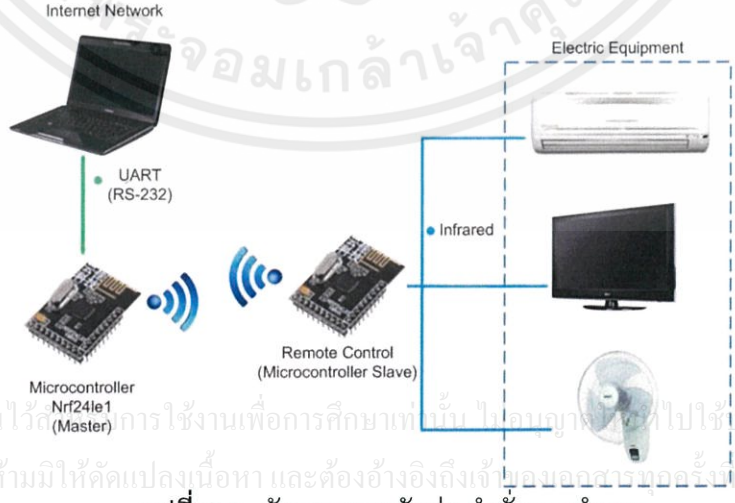
บทที่ 3
การออกแบบการควบคุมเครื่องปรับอากาศ

ในบทนี้จะกล่าวถึงการออกแบบ และการสร้างระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต ซึ่งสามารถแบ่งได้ดังนี้

- การออกแบบโครงสร้างของฮาร์ดแวร์
- การออกแบบการประยุกต์ใช้งานไมโครคอนโทรลเลอร์
- การออกแบบการประยุกต์ระบบการสื่อสารผ่านอินเทอร์เน็ต

3.1 แนวคิดในการออกแบบ

ในการใช้งานกล่องอุปกรณ์ควบคุมเครื่องใช้ไฟฟ้าภายในบ้าน จะต้องนำกล่องอุปกรณ์ไปวางไว้ในตำแหน่งที่อยู่ในแนวระดับเดียวกันหรือใกล้เคียงระหว่างตัวส่งสัญญาณจากกล่องอุปกรณ์กับช่องรับสัญญาณของเครื่องใช้ไฟฟ้าที่ต้องการควบคุมเนื่องจากสัญญาณอินฟราเรดจะถูกส่งออกมาในแนวระดับ การติดตั้งกล่องอุปกรณ์สามารถติดตั้งได้โดยไม่ต้องต่อสายไฟใดๆให้กับกล่องอุปกรณ์ เนื่องจากใช้พลังงานจากแบตเตอรี่ขนาดเล็ก โดยกล่องอุปกรณ์ควบคุมนั้นจะเป็นตัวอุปกรณ์สลาฟ คอยรับคำสั่งด้วยระบบไร้สายจากอุปกรณ์มาสเตอร์ซึ่งเชื่อมต่อกับเครื่องคอมพิวเตอร์และเครือข่ายอินเตอร์เน็ตรอรับคำสั่งจากผู้ใช้งาน เมื่อมีการสั่งงานมายังอุปกรณ์สลาฟ อุปกรณ์จะส่งสัญญาณสัญญาณอินฟราเรดผ่านทาง แอลอีดีอินฟราเรด ไปควบคุมเครื่องใช้ไฟฟ้าตามต้องการ โดยปริญญานิพนธ์นี้ได้ยกตัวอย่างการควบคุมเครื่องปรับอากาศยี่ห้อ ไดกิ้น เป็นกรณีศึกษา โดยมีลักษณะการรับส่งคำสั่ง ดังรูป



รูปที่ 3.1 ลักษณะการรับส่ง คำสั่งการทำงาน

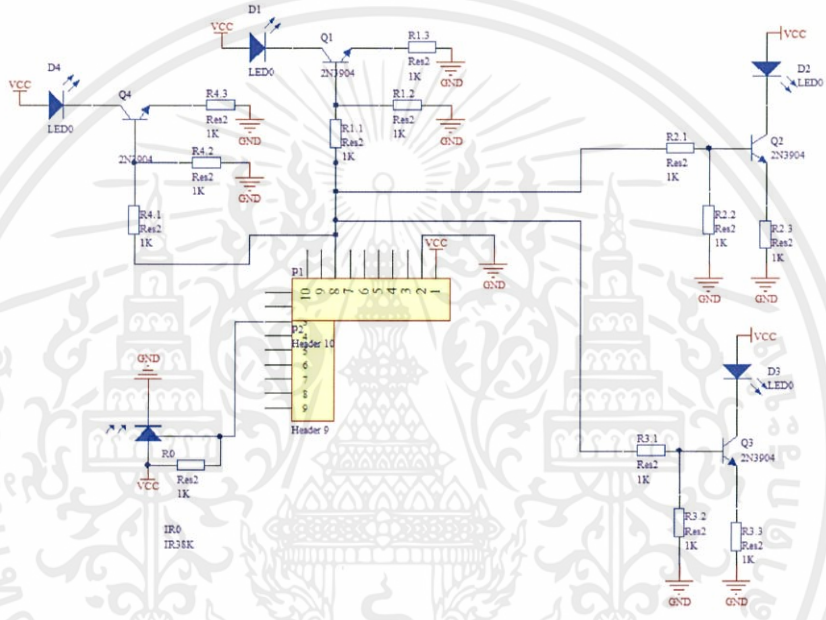
3.2 การออกแบบโครงสร้างของฮาร์ดแวร์

3.2.1 การออกแบบวงจรส่งสัญญาณอินฟราเรด

หลักการออกแบบอุปกรณ์นี้จะคล้ายคลึงกับลักษณะของรีโมททั่วไปแต่มีการปรับเปลี่ยนโครงสร้างเพื่อเพิ่มความสามารถให้มากขึ้น โครงสร้างหลักของอุปกรณ์มีดังนี้

1. ตัวส่งสัญญาณอินฟราเรด
2. วงจรขยายสัญญาณ
3. ไมโครคอนโทรลเลอร์

การต่อวงจรและเชื่อมต่อแต่ละองค์ประกอบเป็นดังรูป 3.2

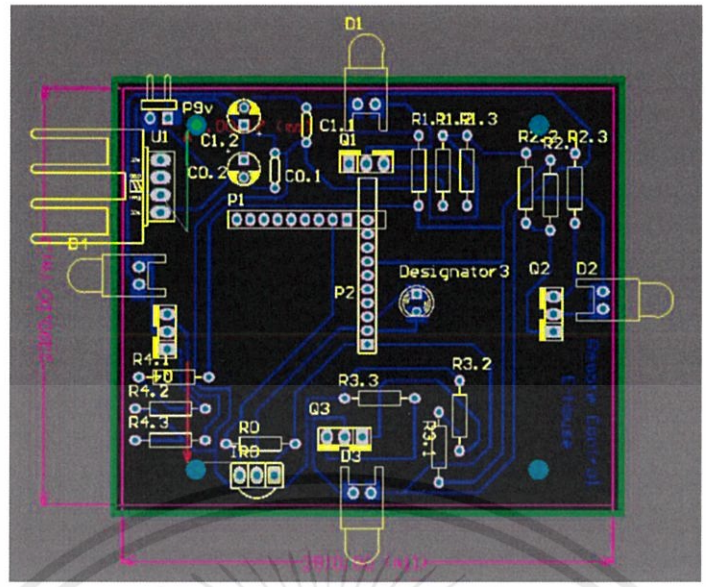


รูปที่ 3.2 วงจรส่งสัญญาณอินฟราเรด

นำวงจรที่ได้ออกแบบมาทดลองการทำงานโดยการเชื่อมต่อไมโครคอนโทรลเลอร์ด้วยพอร์ตยูอาร์ตเข้ากับคอมพิวเตอร์และสั่งงานจากคอมพิวเตอร์ให้มีการส่งสัญญาณอินฟราเรดออกจากวงจรส่งสัญญาณ จากนั้นต่อวงจรรับสัญญาณอินฟราเรดเข้ากับออสซิลโลสโคป และสังเกตหน้าจอแสดงผลว่ามีการส่งสัญญาณออกมาจริงหรือไม่ เมื่อทำการทดลองและได้ผลแล้วว่าวงจรส่งสัญญาณอินฟราเรดสามารถทำงานได้จริง จึงได้นำวงจรมาสร้างเป็นแผ่นลายวงจรต่อไป

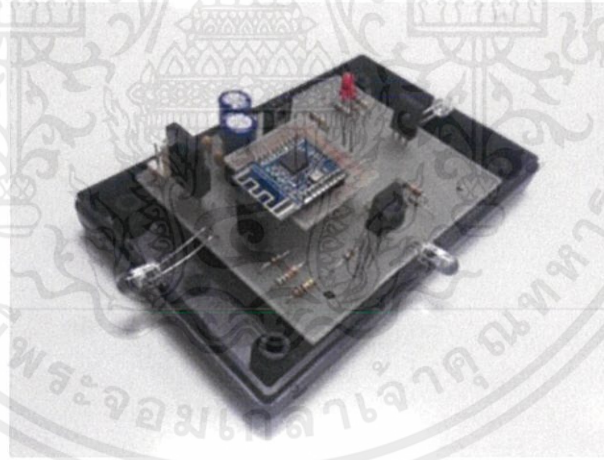
3.2.2 การออกแบบแผ่นลายวงจร

การออกแบบวงจร ออกแบบโดยการใช้โปรแกรม Altium Designer Summer 09 เป็นการนำเอาวงจรที่ได้ออกแบบไว้แล้ว มาออกแบบการวางลงในกล่องอุปกรณ์ที่มีขนาด 4 × 2.6 นิ้ว การออกแบบมีลักษณะดังรูปที่ 3.3



รูปที่ 3.3 ออกแบบแผ่นลายวงจรโดยใช้โปรแกรม Altium Designer Summer 09

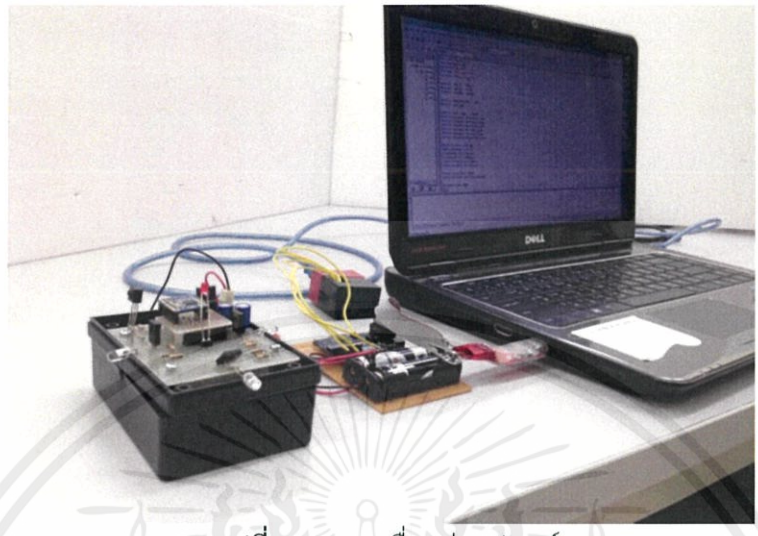
เมื่อได้ทำการสั่งกัดแผ่นลายวงจรออกมาเป็นที่เรียบร้อยแล้ว จึงประกอบอุปกรณ์ต่างๆลงบนแผ่นลายวงจรดังรูปที่ 3.4



รูปที่ 3.4 อุปกรณ์ส่งสัญญาณอินฟราเรดควบคุมเครื่องใช้ไฟฟ้า

เมื่อทำการประกอบอุปกรณ์เรียบร้อยเป็นไปดังรูปที่ 3.4 จากนั้นจึงทำการทดสอบการทำงานของอุปกรณ์โดยการ เสียบแบตเตอรี่ให้กับอุปกรณ์ นำอุปกรณ์ไปวางไว้ที่ด้านหน้าของเครื่องปรับอากาศ ทำการสั่งงานจากคอมพิวเตอร์ที่มีอุปกรณ์มาสเตอร์ที่ส่งสัญญาณไวเลสได้เชื่อมต่อไว้อยู่แล้ว ซึ่งก่อนการจะทดลองอุปกรณ์ที่ประกอบเรียบร้อยแล้วได้นั้น ต้องมีการเขียนซอฟแวร์และประกอบอุปกรณ์มาสเตอร์เป็นที่เรียบร้อยแล้วเท่านั้น หากสั่งงานจากคอมพิวเตอร์และเครื่องปรับ

อากาศสามารถปรับเปลี่ยนตามที่สั่งได้ ถือว่าอุปกรณ์สามารถใช้งานได้ การเชื่อมต่อของอุปกรณ์เป็นดังรูปที่ 3.5



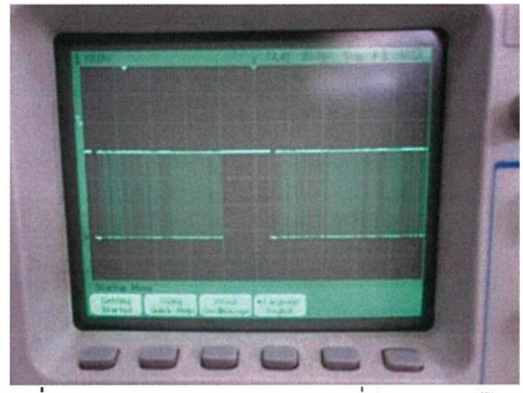
รูปที่ 3.5 การเชื่อมต่ออุปกรณ์

กล่องอุปกรณ์จะมีอินฟราเรดแอลอีดีทั้ง 4 ทิศทางเพื่อการครอบคลุมในการส่งสัญญาณ และเพื่อการประยุกต์ในการใช้งานควบคุมอุปกรณ์อื่นที่สามารถรับสัญญาณอินฟราเรดได้

3.3 การออกแบบการทำงานของไมโครคอนโทรลเลอร์

3.3.1 ศึกษาการส่งสัญญาณอินฟราเรด

ขั้นตอนการศึกษานี้ทำได้โดยการทดลองต่อตัวรับสัญญาณอินฟราเรด(Infrared Receiver) เข้ากับเครื่องออสซิลโลสโคปเพื่อสังเกตลักษณะของสัญญาณอินฟราเรดที่ส่งมาจากรีโมท ทดลองกดปุ่มรีโมทให้ส่งสัญญาณไปยังตัวรับสัญญาณอินฟราเรด และสังเกตลักษณะของสัญญาณอินฟราเรดที่ส่งมา จากนั้นทำการนับจำนวนบิตของสัญญาณ และวัดค่าความยาวคลื่นของแต่ละบิต ทำการบันทึกผลเป็นต้นแบบของสัญญาณการสั่งงานของรีโมทยี่ห้อนี้

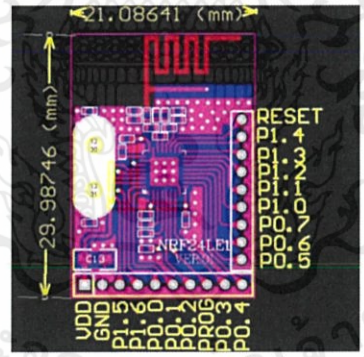


รูปที่ 3.6 ลักษณะของสัญญาณที่ส่งมาจากรีโมท

จากการบันทึกรูปแบบของสัญญาณที่ส่งออกมาพบว่าลักษณะของสัญญาณจะแบ่งเป็น 2 ชุด คือ ชุดแรกเป็นดีไวส์โค้ด และชุดที่ 2 เป็นข้อมูลคำสั่ง โดยแต่ละชุดจะมีเฮดพัลส์เป็นสัญญาณนำร่องก่อนจะมีข้อมูลต่างๆส่งออกมา

3.3.2 การเขียนโปรแกรมตรวจจับสัญญาณอินฟราเรด

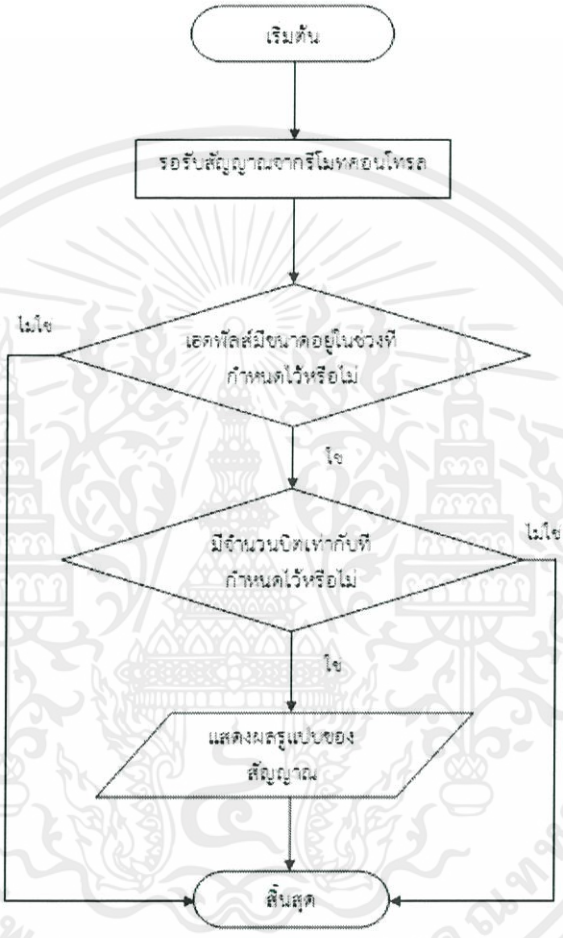
ก่อนการตรวจจับสัญญาณต้องสร้างวงจรตัวรับสัญญาณอินฟราเรดต่อเข้ากับไมโครคอนโทรลเลอร์โดยต่อขาออกของสัญญาณเข้ากับ ขา P0.6 ของไมโครคอนโทรลเลอร์เนื่องจากเป็นขาที่ใช้งานการอินเตอร์รัปต์จากภายนอก



รูปที่ 3.7 แสดงตำแหน่งขาของไมโครคอนโทรลเลอร์

อินเตอร์รัปต์ (Interrupt) คือ การขัดจังหวะการทำงานของซีพียูหรือโปรแกรมให้หยุดพักจากงานที่กระทำอยู่ในปัจจุบัน แล้วกระโดดไปทำงานอีกงานหนึ่งจนเสร็จแล้ว จึงกระโดดกลับมาทำงานชิ้นเดิมที่หยุดพักไว้ต่อไป

จากการใช้งานอินเตอร์รัปต์ของไมโครคอนโทรลเลอร์เมื่อตัวรับสัญญาณอินฟราเรดได้รับสัญญาณอินฟราเรด สัญญาณจะถูกส่งไปยังไมโครคอนโทรลเลอร์และใช้หลักการอินเตอร์รัปต์ในการตรวจจับขอบขาลงของสัญญาณเพื่อวัดขนาดความกว้างของพัลส์ที่เข้ามาว่าเป็นสัญญาณที่ต้องการหรือไม่ ถ้าหากเป็นสัญญาณที่ต้องการ จะมีการแสดงผลลักษณะของสัญญาณทีละ 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]={136,91,228,0,0,140,60,0,245,0,0,0,0,0,0,3,0,0,119,};
```

รูปที่ 3.9 การแสดงผลรูปแบบของสัญญาณ

ทดลองการกดปุ่มของรีโมตทุกปุ่มที่ต้องการและบันทึกรูปแบบสัญญาณของแต่ละปุ่มไว้

3.3.3 การสร้างสัญญาณอินฟราเรดโดยใช้หลักการพัลส์วิทมอทดูเลชั่น

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

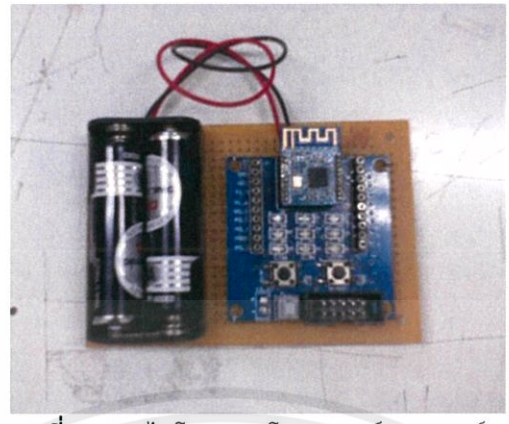
ในการเขียนโปรแกรมจะมีการเขียนข้อมูลของปุ่มสั่งงานที่ได้ทำการบันทึกไว้ก่อนหน้านี้แล้ว ว่าปุ่มใดมีลักษณะอย่างไร มีจำนวนกี่บิตและมีความกว้างพัลส์เท่าใด เมื่อต้องการสั่งงานคำสั่งไหน โปรแกรมจะเรียกข้อมูลที่ได้บันทึกไว้ นำมาเป็นแม่แบบในการสร้างสัญญาณที่มีการเรียงลอจิก 0 และ 1 ที่แตกต่างกันในแต่ละปุ่มคำสั่ง

รูปที่ 3.10 ลักษณะของสัญญาณที่ส่งมาจากอุปกรณ์ที่สร้างขึ้น

3.3.4 การออกแบบการทำงานของไมโครคอนโทรลเลอร์

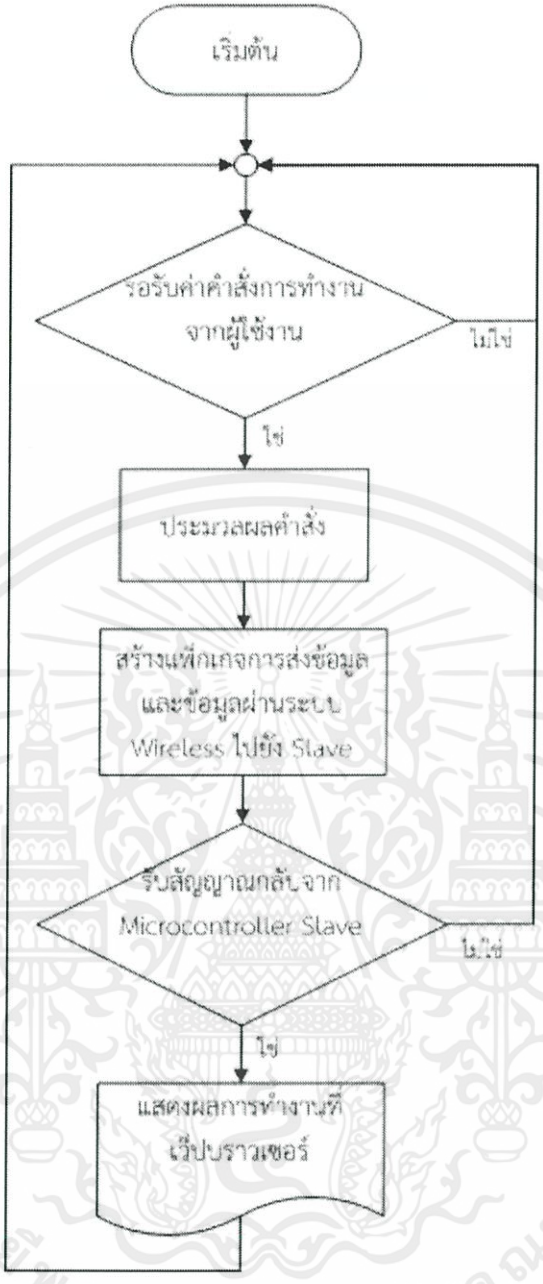
ไมโครคอนโทรลเลอร์จะแบ่งการทำงานออกเป็น 2 ส่วน คือ มาสเตอร์และสลาฟ ส่วนของไมโครคอนโทรลเลอร์มาสเตอร์จะเชื่อมต่ออยู่กับคอมพิวเตอร์ที่ติดต่อกับเซิร์ฟเวอร์อินเทอร์เน็ต และส่วนของไมโครคอนโทรลเลอร์สลาฟนั้นจะอยู่ในกล่องอุปกรณ์ส่งสัญญาณอินฟราเรดควบคุมเครื่องปรับอากาศ

3.3.4.1 การทำงานของไมโครคอนโทรลเลอร์มาสเตอร์



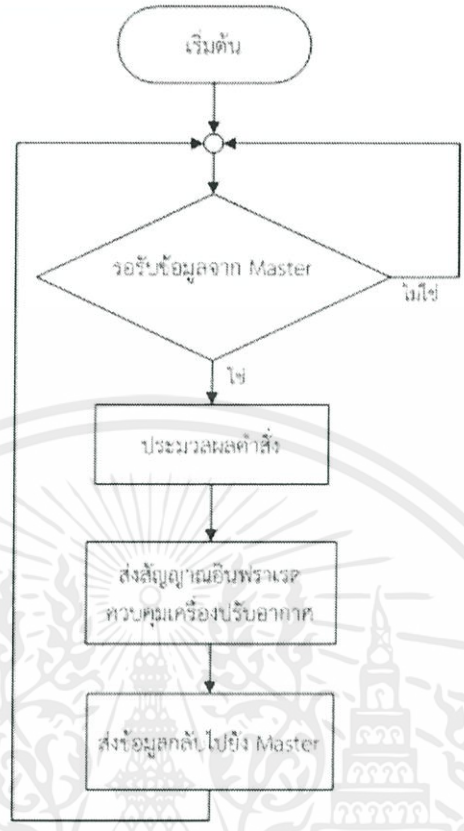
รูปที่ 3.11 ไมโครคอนโทรลเลอร์มาสเตอร์

ไมโครคอนโทรลเลอร์มาสเตอร์เปรียบเสมือนสื่อกลางในการสื่อสารระหว่างผู้ใช้งานและอุปกรณ์โดยมีขั้นตอนการทำงานดังนี้ไมโครคอนโทรลเลอร์มาสเตอร์เชื่อมต่ออยู่กับคอมพิวเตอร์ด้วยซีเรียลพอร์ตผ่านทางบอร์ดยูอาร์ทเมื่อมีการสั่งงานมาจากเว็บบราวเซอร์ ข้อมูลจะถูกส่งมายังไมโครคอนโทรลเลอร์และจะมีการประมวลผลคำสั่งพร้อมกับการสร้างแพ็กเกจข้อมูลเพื่อส่งข้อมูลนั้นไปยังไมโครคอนโทรลเลอร์โดยส่งผ่านระบบไวเลสและสลาฟจะส่งค่ากลับมายังมาสเตอร์เพื่อนำไปแสดงสถานะการทำงานของเครื่องปรับอากาศบนหน้าเว็บบราวเซอร์



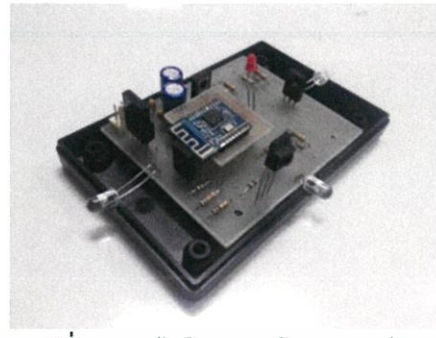
รูปที่ 3.12 ขั้นตอนการทำงานของไมโครคอนโทรลเลอร์มาสเตอร์

3.3.4.2 การทำงานของไมโครคอนโทรลเลอร์สลาฟ



รูปที่ 3.13 ขั้นตอนการทำงานของไมโครคอนโทรลเลอร์สลาฟ

ไมโครคอนโทรลเลอร์สลาฟจะเชื่อมต่อกับวงจรส่งสัญญาณอินฟราเรดเพื่อส่งสัญญาณไปควบคุมเครื่องปรับอากาศ โดยรับคำสั่งมาจากมาสเตอร์ในรูปของแพคเกจข้อมูล จากนั้นไมโครคอนโทรลเลอร์จะทำการประมวลผล ถอดคำสั่งการทำงานและส่งสัญญาณอินฟราเรดไปสั่งงานเครื่องปรับอากาศตามคำสั่งของผู้ใช้งานพร้อมกับการส่งข้อมูลกลับไปยังมาสเตอร์เพื่อแสดงสถานะของคำสั่งที่ได้สั่งงานเครื่องปรับอากาศไปแล้ว และที่ตัวสลาฟนี้ยังมีอินฟราเรดรีซีฟเวอร์รับสัญญาณรีโมตปกติที่ใช้สั่งงานเครื่องปรับอากาศ เพื่อส่งคำสั่งนั้นไปแสดงสถานะการทำงานที่หน้าเว็บบราวเซอร์ได้เช่นเดียวกัน



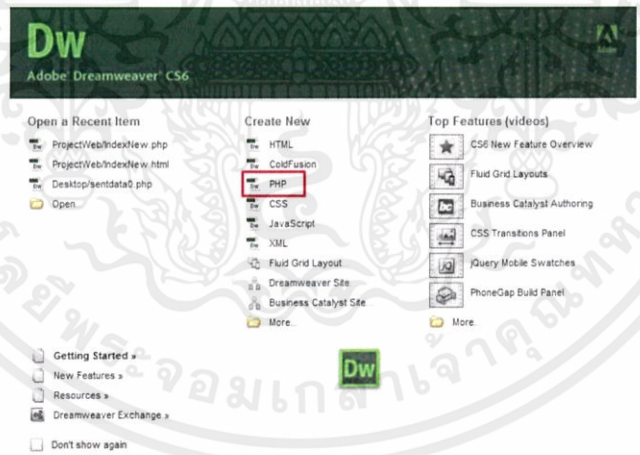
รูปที่ 3.14 ไมโครคอนโทรลเลอร์สลาฟ

3.4 การออกแบบระบบการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต

ออกแบบเว็บเพจสำหรับสั่งงานไมโครคอนโทรลเลอร์ด้วยชุดคำสั่งภาษาพีเอชพีให้ส่งสัญญาณอินฟราเรดเพื่อทำการควบคุมการทำงานของเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต

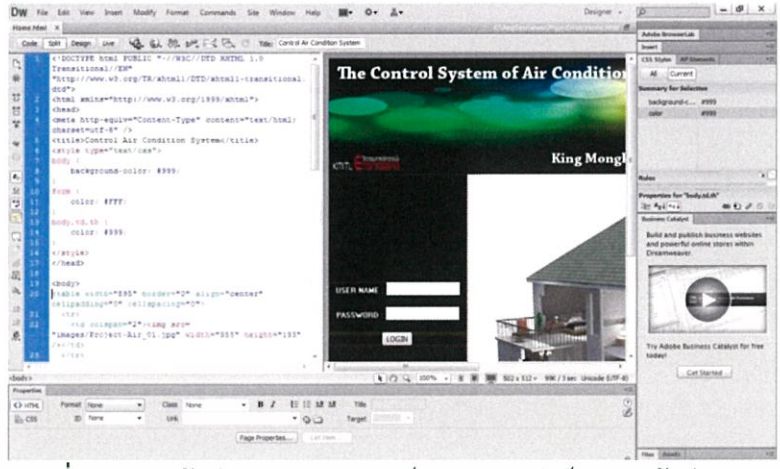
3.4.1 การสร้างเว็บเพจด้วยโปรแกรม Dreamweaver

1. หลังจากติดตั้งโปรแกรม Dreamweaver แล้ว ให้ทำการเปิดโปรแกรมขึ้นมาจากนั้นสร้างหน้าเว็บขึ้นมาใหม่ โดยคลิกที่ PHP



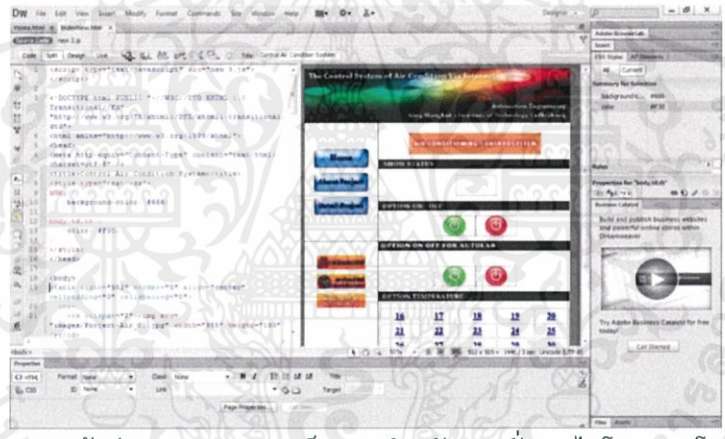
รูปที่ 3.15 หน้าต่างการสร้างหน้าเว็บใหม่

2. ออกแบบเว็บเพจที่จะนำมาเป็นหน้าแรกสำหรับล็อกอินเข้าสู่ระบบการสั่งงานไมโครคอนโทรลเลอร์เพื่อควบคุมการทำงานของเครื่องปรับอากาศ



รูปที่ 3.16 หน้าต่างการออกแบบเว็บเพจสำหรับล็อกอินเข้าสู่ระบบ

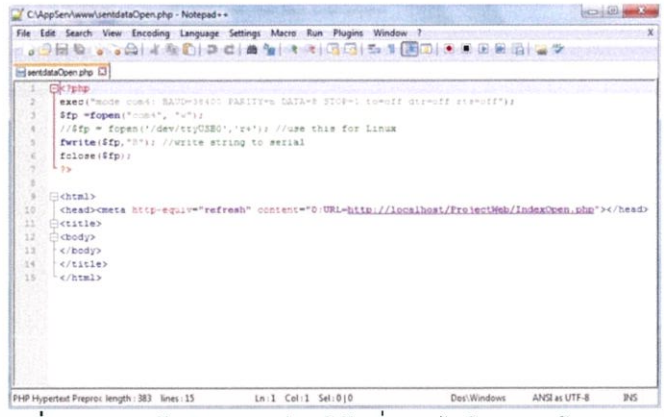
3. ออกแบบเว็บเพจที่ใช้สำหรับการสั่งงานไมโครคอนโทรลเลอร์เพื่อควบคุมการทำงานของเครื่องปรับอากาศ



รูปที่ 3.17 หน้าต่างการออกแบบเว็บเพจสำหรับการสั่งงานไมโครคอนโทรลเลอร์

3.4.2 สร้างไฟล์คำสั่งภาษาพีเอชพี

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

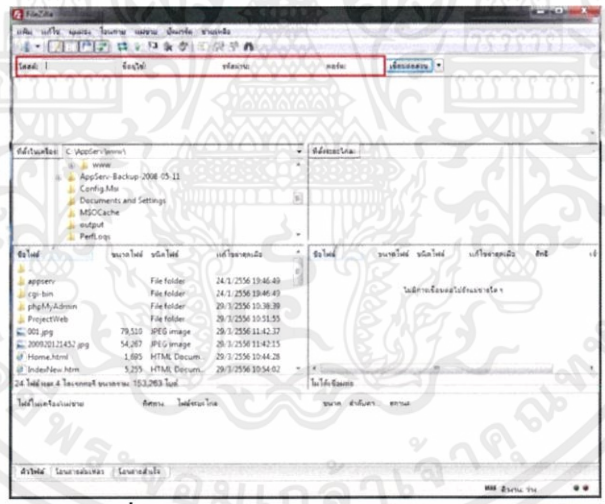


รูปที่ 3.18 หน้าต่างการเขียนโค้ดสั่งงานไมโครคอนโทรลเลอร์

3.4.3 การติดต่อกับเซิร์ฟเวอร์

1. หลังจากที่ได้ทำการติดต่อขอพื้นที่เว็บไซต์บนเว็บเซิร์ฟเวอร์แล้ว ให้ทำการติดตั้งโปรแกรม FileZillaเพื่อใช้ในการอัพโหลดไฟล์ไปยังเว็บเซิร์ฟเวอร์

2. ในการอัพโหลดไฟล์ไปยังเว็บเซิร์ฟเวอร์นั้นจะต้องทำการกรอกโฮสต์, ชื่อผู้ใช้, รหัสผ่าน และพอร์ต ซึ่งข้อมูลเหล่านี้จะได้มาจากขั้นตอนการขอพื้นที่เว็บไซต์บนเว็บเซิร์ฟเวอร์



รูปที่ 3.19 หน้าต่างโปรแกรม FileZilla

3. ทำการอัพโหลดไฟล์ทั้งหมดของเว็บเพจขึ้นไปยังเว็บเซิร์ฟเวอร์ เพื่อใช้เป็นเว็บไซต์สำหรับระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต

บทที่ 4
ผลการทดสอบ

4.1วิธีการสั่งงานเครื่องปรับอากาศ

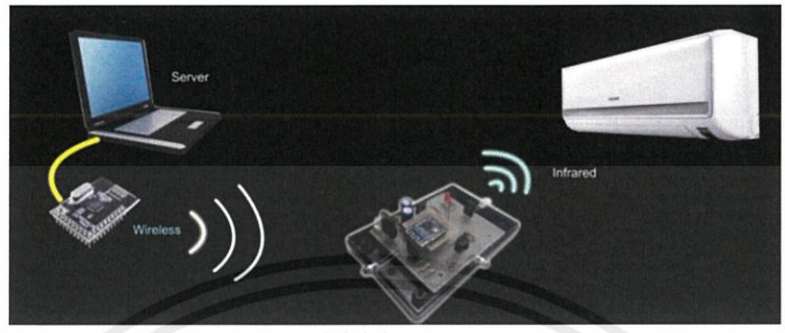
การสั่งงานสามารถสั่งงานได้จากหน้าเว็บบราวเซอร์โดยมีแอดเดรสคือ www.ehouse-air.com โดยจะมีการใส่รหัสก่อนการใช้งาน เพื่อป้องกันความปลอดภัยในการแทรกแซงการทำงานของผู้อื่น

รูปที่ 4.1 หน้าเว็บเพจการใส่รหัสก่อนการใช้งาน

รูปที่ 4.2 หน้าเว็บเพจการสั่งงาน

เมื่อใส่รหัสผ่านที่ถูกต้อง จะสามารถเข้าไปยังหน้าเว็บเพจการสั่งงานได้ ผู้ใช้งานสามารสั่งงานปิด-เปิด หรือปรับเปลี่ยนอุณหภูมิได้ตามต้องการ เมื่อมีการสั่งงานข้อมูลจะถูกส่งมายังอุปกรณ์มาสเตอร์ที่เชื่อมต่อกับคอมพิวเตอร์ที่เป็นเซิร์ฟเวอร์

จากนั้นอุปกรณ์มาสเตอร์จะส่งสัญญาณด้วยระบบไร้สายไปยังอุปกรณ์สลาฟ อุปกรณ์สลาฟจะทำการส่งสัญญาณอินฟราเรดไปควบคุมเครื่องปรับอากาศ ให้ทำการเปิด ปิด และปรับอุณหภูมิของเครื่องปรับอากาศตามที่ต้องการ



รูปที่ 4.3 ลักษณะการส่งสัญญาณอินฟราเรดควบคุมเครื่องปรับอากาศ

4.2 ข้อจำกัดในการใช้งาน

4.2.1 ระยะห่างของเครื่องปรับอากาศและอุปกรณ์สลาฟ

เพื่อความแม่นยำในการสั่งงานเครื่องปรับอากาศและอุปกรณ์สลาฟควรมีระยะห่างไม่เกิน 4 เมตร และระยะห่างสูงสุดที่สามารถทำงานได้คือ ระยะห่างประมาณ 7 เมตร

4.2.1 ในการสั่งงานเครื่องปรับอากาศยี่ห้ออื่น

ต้องทำการแกะโค้ดใหม่ เนื่องจากโค้ดการสั่งงานเครื่องปรับอากาศแต่ละยี่ห้อมีความแตกต่างกัน จึงเกิดข้อจำกัดในการสั่งงานของเครื่องปรับอากาศต่างยี่ห้อกัน

บทที่ 5
สรุปผลการทดสอบและข้อเสนอแนะ

5.1 สรุปผลการทดสอบ

จากการทดลองเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์สร้างสัญญาณพัลส์วิทมอทดูเลชั่นนั้น ได้ผลตามที่ต้องการ คือ สามารถส่งสัญญาณอินฟราเรดออกมาได้ตามสัญญาณรีโมทที่ใช้ในการควบคุมเครื่องปรับอากาศทั่วไป และจากการทดลองการส่งสัญญาณอินฟราเรดควบคุมเครื่องปรับอากาศก็ได้ผลตามที่ต้องการเช่นกัน คือ สามารถสั่งงานเครื่องปรับอากาศจากเว็บบราวเซอร์ได้ตามที่ตั้งเป้าหมายไว้ แต่การส่งสัญญาณจากรีโมทจริงนั้น มีคำสั่งที่ค่อนข้างซับซ้อน ในการถอดสัญญาณของรีโมทแล้วนำสัญญาณนั้นมาสั่งค่าออกจากอุปกรณ์ที่สร้างขึ้น จึงเป็นเพียงการสั่งคำสั่งขั้นพื้นฐาน ในการควบคุมเครื่องปรับอากาศ ที่ใช้ในการเปิด-ปิด และปรับอุณหภูมิแบบปกติ

จากที่กล่าวมาข้างต้น การศึกษา ออกแบบ และสร้างอุปกรณ์ควบคุมเครื่องปรับอากาศที่สามารถสั่งงานผ่านระบบอินเทอร์เน็ต ได้ประสบความสำเร็จตามวัตถุประสงค์ตามที่ตั้งไว้ ซึ่งจะก่อให้เกิดความสะดวกสบายของผู้ใช้งาน และง่ายต่อการบริหารจัดการการใช้งานเครื่องปรับอากาศเพื่อความประหยัดและปลอดภัย

5.2 ปัญหาที่พบ

การสั่งงานเครื่องปรับอากาศในยี่ห้อที่ต่างกัน โค้ดในการสั่งงานย่อมต่างกัน จึงต้องใช้เวลาในการถอดโค้ดของยี่ห้อที่ต้องการก่อนที่จะนำมาสั่งงาน ฉะนั้นโดยเริ่มต้นของอุปกรณ์ตัวนี้ยังไม่สามารถสั่งงานเครื่องปรับอากาศได้ทุกยี่ห้อ

5.2 ข้อเสนอแนะ

1. ไมโครคอนโทรลเลอร์ nrf24le1 เป็นไมโครคอนโทรลเลอร์ที่ค่อนข้างมีข้อจำกัดในหลายด้าน เช่น ระยะทางการส่งสัญญาณ ความจำ และฟังก์ชั่นการทำงานที่จำกัด ดังนั้น การจะนำแนวคิดจากโครงงานนี้ไปใช้งานจึงอาจจะประยุกต์ใช้กับไมโครคอนโทรลเลอร์ที่มีความสามารถมากกว่าไมโครคอนโทรลเลอร์ชนิดนี้

2. การเพิ่มระยะห่างของการสั่งงานจากอุปกรณ์สลาฟไปยังเครื่องปรับอากาศ สามารถทำได้โดยการปรับเปลี่ยนวงจรส่งสัญญาณอินฟราเรดให้มีการขยายสัญญาณเพิ่มขึ้นเพื่อเพิ่มความแม่นยำและความสะดวกในการใช้งาน

3. ในการสั่งงานเครื่องใช้ไฟฟ้าแต่ละชนิดจะต้องใช้รีโมทหลายตัวตามจำนวนของเครื่องใช้ไฟฟ้า แต่เราสามารถนำแนวคิดจากโครงงานนี้มาใช้ในการออกแบบรวบรวมสัญญาณรีโมทของเครื่องใช้ไฟฟ้าแต่ละชนิดไว้ในตัวเดียวกัน

บรรณานุกรม

[1] **เขียนโปรแกรมไมโครคอนโทรลเลอร์ด้วยภาษาซี**. (2551). 10 กุมภาพันธ์ 2556, จาก http://www.thaiembedded.com/blog/?p=50

[2] **การใช้โปรแกรม KeiluVision 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)

[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 นิ่มตระกูล. **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 ปริญญานิพนธ์ การควบคุมไฟฟ้าไร้สายภายในบ้าน (Wireless electrical control in house)

ภาคผนวก

ภาคผนวก ก
โปรแกรมที่ใช้ในโครงงาน

ในบทนี้จะอธิบายถึงส่วนของโปรแกรมต่างๆที่ใช้ในโครงงานซึ่งประกอบด้วย 4ส่วนคือส่วนโปรแกรมการทำงานของไมโครคอนโทรลเลอร์มาสเตอร์ส่วนโปรแกรมการทำงานของไมโครคอนโทรลเลอร์สลาฟ ส่วนโปรแกรมการติดต่อระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ผ่านพอร์ตยูอาร์ทและส่วนโปรแกรมการสร้างเว็บเพจ

ก.1 ส่วนโปรแกรมการทำงานของไมโครคอนโทรลเลอร์มาสเตอร์

ไมโครคอนโทรลเลอร์มาสเตอร์เชื่อมต่ออยู่กับคอมพิวเตอร์ด้วยซีเรียลพอร์ตผ่านทางบอร์ดยูอาร์ทเมื่อมีการสั่งงานมาจากเว็บบราวเซอร์ ข้อมูลจะถูกส่งมายังไมโครคอนโทรลเลอร์และจะมีการประมวลผลคำสั่ง พร้อมกับการสร้างแพ็กเกจข้อมูลเพื่อส่งข้อมูลนั้นไปยังไมโครคอนโทรลเลอร์โดยส่งผ่านระบบไวเลส และสลาฟจะส่งค่ากลับมายังมาสเตอร์เพื่อนำไปแสดงสถานะการทำงานของเครื่องปรับอากาศบนหน้าเว็บบราวเซอร์สามารถแสดงได้ดังนี้

```
#include<reg24le1.H>
#include<stdio.H>
#include "main.h"
#include "nrf_le0.h"
#include "nRF_defines.h"

#define  nRF_CE   RFCE
#define  nRF_CS   RFCSN
#define  SSP1BUF  SPIRDAT

#define  Address_Route  0xff

unsigned char LOOP, inputdata, Check, Reset_Loop ,Test ,Checksum_Tx = 0;
unsigned int Datanet,p;
unsigned char temp,keep;
unsigned int Rx_flag ,Uart_flag , Decode_in, Decode_out = 0x0000, Send_Decode = 0,x;


void Delay_1us(volatile unsigned int n){
   unsigned char i;
   while(n--)
     for(i=0;i<3;i++);
 }
 void Delay_100us(volatile unsigned int n){
   unsigned char i;
   while(n--)
     for(i=0;i<35;i++);
 }

 void Delay_ms(int i)
{
    int j;
    for(; i>0; i--)
    for(j=0; j<100; j++);
}

void UART_Mode() interrupt 4 using 3
{
    if(RIO)
    {

        keep = (unsigned int)SOBUF;
        Datanet = keep;
        RIO = 0;
        LOOP = 1;
    }
    else if(TIO)
    {
        TIO=0;
    }
}
void CE_Pin_Active(unsigned char action)    // CE pin high, low or pulse..
{
  switch(action)
  {
    case CE_LOW:                        // action == 0, CE low
      nRF_CE = 0;
    break;

    case CE_HIGH:                       // action == 1, CE high
      nRF_CE = 1;
    break;

    case CE_PULSE:                      // action == 2, CE pulse (10?s)
     // TMR2   = 0;
      nRF_CE = 1;                       // Set CE pin high
       Delay_1us(100);
       nRF_CE = 0;
      //TMR2ON_bit = 1;                  // Start Timer2, CE pulse timer
    break;
  }
}
unsigned char LO1_Clear_IRQ(unsigned char irq_flag)
{
  return nRF_SPI_RW_Reg(WRITE_REG + STATUS, irq_flag);
}

unsigned char nRF_SPI_RW(unsigned char byte1)
{
  unsigned char status;
  //Delay_1us(10);
  SSP1BUF = byte1;
  Delay_1us(1);          // Delay 300s ok
  status  = SSP1BUF;
  return(status);
}
unsigned char nRF_SPI_RW_Reg(unsigned char reg, unsigned char value)
{
  unsigned char status;

  nRF_CS = 0;                                   // CSN low, init SPI transaction
  status = nRF_SPI_RW(reg);                     // select register
  nRF_SPI_RW(value);                            // ..and write value to it..
  nRF_CS = 1;                                   // CSN high again

  return(status);                               // return nRF24LO1 status byte
}
unsigned char nRF_SPI_Write_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
{
  unsigned char status, byte_ctr;

  nRF_CS = 0;                                   // Set CSN low, init SPI transaction
  status = nRF_SPI_RW(reg);                     // Select register to write to and read status byte

  for(byte_ctr=0; byte_ctr<bytes; byte_ctr++)     // then write all byte in buffer(*pBuf)
  {
    nRF_SPI_RW(*pBuf++);
  }


  nRF_CS = 1;                                   // Set CSN high again

  return(status);                               // return nRF24LO1 status byte
}

unsigned char LO1_RD_RX_PW_n(unsigned char pipe)
{
  return nRF_SPI_Read(RX_PW_P0 + pipe);
}
void LO1_Write_TX_Pload(unsigned char *pBuf, unsigned char plWidth)
{
  nRF_SPI_Write_Buf(WR_TX_PLOAD, pBuf, plWidth);
}
```

```
unsigned char Send_Packet()                          // Send one data packet, controlled by Button1(SW1)
{
    LO1_Write_TX_Pload(&TX_pload, TX_PLOAD_WIDTH);     // Write new TX payload
    CE_Pin_Active(CE_PULSE);                             // Enable CE pulse, 12?s
  //ucTry_Ctr = 0;                                     // Reset Try_Ctr before nex transmitt..
  return 1;
}
void LO1_Flush_RX(void)
{
  nRF_SPI_RW_Reg(FLUSH_RX,0);
}
void LO1_Flush_TX(void)
{
  nRF_SPI_RW_Reg(FLUSH_TX,0);
}
unsigned char LO1_Get_FIFO(void)
{
  return nRF_SPI_Read(FIFO_STATUS);
}
unsigned char nRF_SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
{
  unsigned char status, byte_ctr;

  nRF_CS = 0;                                        // Set CSN low, init SPI tranaction
  status = nRF_SPI_RW(reg);                          // Select register to write to and read status byte

  for(byte_ctr=0;byte_ctr<bytes;byte_ctr++)          // Perform SPI_RW to read byte from nRF24LO1
  {
    pBuf[byte_ctr] = nRF_SPI_RW(0);
  }

  nRF_CS = 1;                                        // Set CSN high again

  return(status);                                    // return nRF24LO1 status byte
}
unsigned char nRF_SPI_Read(unsigned char reg)
{
  unsigned char reg_val;

  nRF_CS = 0;                                        // CSN low, initialize SPI communication...
  nRF_SPI_RW(reg);                                   // Select register to read from..
  reg_val = nRF_SPI_RW(0);                           // ..then read registervalue
  nRF_CS = 1;                                        // CSN high, terminate SPI communication

  return(reg_val);                                   // return register value
}
unsigned char LO1_Get_Status(void)
{
  return nRF_SPI_Read(STATUS);
}
unsigned char LO1_Get_Current_Pipenum(void)
{
  return ((LO1_Get_Status() & RX_P_NO) >> 1);
}
unsigned int LO1_Read_RX_Pload(unsigned char *pBuf)
{
  unsigned char plWidth, pipe;
  plWidth = LO1_RD_RX_PW_n(pipe = LO1_Get_Current_Pipenum());   // Read current pipe's payload width
  nRF_SPI_Read_Buf(RD_RX_PLOAD, pBuf, plWidth);                 // Then get RX data

  return ((pipe << 8) + plWidth);                               // return pipe# & pipe#.plWidth
}
 /*unsigned char nRF_SPI_RW_Reg_IRQ(unsigned char reg, unsigned char value)
{
  unsigned char status;

  nRF_CS = 0;                                        // CSN low, init SPI transaction

  SSP1BUF = reg;
  //while (SSP1STAT.BF == 0)
  Delay_1us(10);
  status = SSP1BUF;  // free the register

  SSP1BUF = value;
  //while (SSP1STAT.BF == 0)
    Delay_1us(10);
  reg = SSP1BUF;  // free the register

  nRF_CS = 1;                                        // CSN high again

  return(status);                                    // return nRF24LO1 status byte
}*/
void Tx_Mode ()
{
      CE_Pin_Active(CE_LOW);                           // Set CE pin low to enable stanby mode
      Delay_ms(200);
      LO1_Flush_TX();
      LO1_Flush_RX();
      //LO1_Clear_IRQ(MASK_IRQ_FLAGS);                 // Clear interrupts
      nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
      //ucIRQ_Source = CLEAR;
      //ucLastStat = ucLinkStat = LINK_ESTABLISH;          // LINK_ESTABLISH = 0x02

      nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f);           // Enable Auto.Ack:Pipes 0-5
      nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f);       // Enable Pipes 0-5
      nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a);      // 500?s + 86?s, 10 retrans...
      nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40);           // Select RF channel 40
      nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03);        // Select Addres
      nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08);        // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
      nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e);          // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..

      nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_P0, sizeof(ADDRESS_P0));    // Writes TX_Address to nRF24LO1
      nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, &ADDRESS_P0, sizeof(ADDRESS_P0)); // RX_Addr0 same as TX_Adr for Auto.Ack
      CE_Pin_Active(CE_HIGH);
}
/*void Tx_Mode_runtime()
{
      CE_Pin_Active(CE_LOW);                           // Set CE pin low to enable stanby mode
      //Delay_ms(200);
      LO1_Flush_TX();
      LO1_Flush_RX();
      LO1_Clear_IRQ(MASK_IRQ_FLAGS);                 // Clear interrupts
      //nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
      //ucIRQ_Source = CLEAR;
      //ucLastStat = ucLinkStat = LINK_ESTABLISH;          // LINK_ESTABLISH = 0x02

      //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f);           // Enable Auto.Ack:Pipes 0-5
      //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f);       // Enable Pipes 0-5
      //nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a);      // 500?s + 86?s, 10 retrans...
      //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40);           // Select RF channel 40
      //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08);        // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
      //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e);          // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..

      //nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_P0, sizeof(ADDRESS_P0));    // Writes TX_Address to nRF24LO1
      //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, &ADDRESS_P0, sizeof(ADDRESS_P0)); // RX_Addr0 same as TX_Adr for Auto.Ack
      CE_Pin_Active(CE_HIGH);
      Delay_ms(50);                                  //max 1 ms
      //CE_Pin_Active(CE_LOW);
}*/
```

```
void Rx_Mode ()
{

      CE_Pin_Active(CE_LOW);                                  // Set CE pin low to enable stanby mode
      Delay_ms(1);                                            // 200ms
      LO1_Flush_TX();
      LO1_Flush_RX();

      LO1_Clear_IRQ(MASK_IRQ_FLAGS);                          // Clear interrupts

      nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f);                // Enable Auto.Ack:Pipes 0-5
      nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f);            // Enable Pipes 0-5
      nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40);                  // Select RF channel 40
      nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03);             // Select Address
      nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x0f);             // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
      nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f);               // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..

      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 device
      nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));

      nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
      nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
      nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
      nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);

      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P0, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_PLOAD_WIDTH);
      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_PLOAD_WIDTH);
      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_PLOAD_WIDTH);
      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_PLOAD_WIDTH);
      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_PLOAD_WIDTH);

      CE_Pin_Active(CE_HIGH);

}
void Rx_Mode_Runtime()
{
      //CE_Pin_Active(CE_LOW);                                  // Set CE pin low to enable stanby mode
      //Delay_ms(1);                                              // 200ms
      //LO1_Flush_TX();
      //LO1_Flush_RX();

      //LO1_Clear_IRQ(MASK_IRQ_FLAGS);                          // Clear interrupts

      //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f);                // Enable Auto.Ack:Pipes 0-5
      //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f);            // Enable Pipes 0-5
      //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40);                  // Select RF channel 40
      //nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03);             // Select Address
      //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x0f);             // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
      //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f);               // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..

      //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 device
      //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));

      //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);

      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P0, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_PLOAD_WIDTH);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_PLOAD_WIDTH);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_PLOAD_WIDTH);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_PLOAD_WIDTH);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_PLOAD_WIDTH);

      CE_Pin_Active(CE_HIGH);
      Delay_ms(30);
}
void Rx_Real_Mode()
{
    Rx_Mode_Runtime();
          do
          {
            //P00 = ~P00;
            uiRX_info = LO1_Read_RX_Pload(RX_pload);        // Read current payload
            // temp  = LO1_Get_FIFO() & 0x02;    temp  >> 1;

          }
          while(!(LO1_Get_FIFO() & RX_EMPTY));              // ..until FIFO empty
           //while( temp == 0 );
          temp  = LO1_Get_FIFO() & 0x01;    // temp  >> 1;

          //P14 = temp;

          //uiRX_info = LO1_Read_RX_Pload(RX_pload);
          LO1_Clear_IRQ(MASK_RX_DR_FLAG);

          /*if (RX_pload[0] == 0x31){ P00 = ~P00; RX_pload[0] = 0; }
          if (RX_pload[1] == 0x32){ P01 = ~P01; RX_pload[1] = 0; }
          if (RX_pload[2] == 0x33){ P02 = ~P02; RX_pload[2] = 0; }
          if (RX_pload[3] == 0x34){ P03 = ~P03; RX_pload[3] = 0; }
          if (RX_pload[4] == 0x35){ P04 = ~P04; RX_pload[4] = 0; }*/
          CE_Pin_Active(CE_LOW);
}
void Tx_Real_Mode()
{
      Send_Packet();
      P13 = ~P13;
      Delay_ms(1);
      LO1_Clear_IRQ(MASK_TX_DS_FLAG);
}
/*void Decode_output()
{
    int DO;
    unsigned char Data_All;
    DO = (3*(((Address_Route>>4) & 9))+1);
    if ( Send_Decode > Data_All )
    {
        Send_Decode = 0;
    }
    switch ( Send_Decode )
    {
        case 0:
        break;
    }
    Send_Decode++;
}*/
void Decode_input()
{
    int DI;
    DI = (3*(((Address_Route>>4) & 9))+1);
    switch ( (RX_pload[DI] & 0xf0) >> 4 )
    {
        case 0:
        /*SOBUF = RX_pload[2];
        while(TI0)
          {
              TI0 = 0;
          }*/
        break;
        case 1:
        break;
    }
}
```

```
void Check_Device()
{
    /*    int CD,set,pload;
     CD = ((Address_Route>>4) & 9);
     if(CD == 0)
     {
          Rx_Real_Mode();

          for(set=3;set<=31;set++)
          TX_pload[set] = RX_pload[set];

          for(pload=0;pload<=30;pload++)
          {
               if((pload % 3) == 0)
               TX_pload[pload]= TX_pload[pload] & 0xf0;
          }
          TX_pload[0] = Address_Route;
     }
     else if(CD == 1)
     {
          Rx_Real_Mode();

          for(set=0;set<=2;set++)
          TX_pload[set] = RX_pload[set];

          for(set=6;set<=31;set++)
          TX_pload[set] = RX_pload[set];

          for(pload=0;pload<=30;pload++)
          {
               if((pload % 3) == 0)
               TX_pload[pload]= TX_pload[pload] & 0xf0;
          }
          TX_pload[3] = Address_Route;
     }
     else if(CD == 2)
     {
          Rx_Real_Mode();

          for(set=0;set<5;set++)
          TX_pload[set] = RX_pload[set];

          for(set=9;set<=31;set++)
          TX_pload[set] = RX_pload[set];

          for(pload=0;pload<=30;pload++)
          {
               if((pload % 3) == 0)
               TX_pload[pload]= TX_pload[pload] & 0xf0;
          }
          TX_pload[6] = Address_Route;
     }
     else if(CD == 3)
     {
          Rx_Real_Mode();

          for(set=0;set<=8;set++)
          TX_pload[set] = RX_pload[set];

          for(set=12;set<=31;set++)
          TX_pload[set] = RX_pload[set];

          for(pload=0;pload<=30;pload++)
          {
               if((pload % 3) == 0)
               TX_pload[pload]= TX_pload[pload] & 0xf0;
          }
          TX_pload[9] = Address_Route;
     }
     else if(CD == 4)
     {
          Rx_Real_Mode();

          for(set=0;set<=11;set++)
          TX_pload[set] = RX_pload[set];

          for(set=15;set<=31;set++)
          TX_pload[set] = RX_pload[set];

          for(pload=0;pload<=30;pload++)
          {
               if((pload % 3) == 0)
               TX_pload[pload]= TX_pload[pload] & 0xf0;
          }
          TX_pload[12] = Address_Route;
     }
     else if(CD == 5)
     {
          Rx_Real_Mode();

          for(set=0;set<=14;set++)
          TX_pload[set] = RX_pload[set];

          for(set=18;set<=31;set++)
          TX_pload[set] = RX_pload[set];

          for(pload=0;pload<=30;pload++)
          {
               if((pload % 3) == 0)
               TX_pload[pload]= TX_pload[pload] & 0xf0;
          }
          TX_pload[15] = Address_Route;
     }
     else if(CD == 6)
     {
          Rx_Real_Mode();

          for(set=0;set<=17;set++)
          TX_pload[set] = RX_pload[set];

          for(set=21;set<=31;set++)
          TX_pload[set] = RX_pload[set];

          for(pload=0;pload<=30;pload++)
          {
               if((pload % 3) == 0)
               TX_pload[pload]= TX_pload[pload] & 0xf0;
          }
          TX_pload[18] = Address_Route;
     }
     else if(CD == 7)
     {
          Rx_Real_Mode();

          for(set=0;set<=20;set++)
          TX_pload[set] = RX_pload[set];

          for(set=24;set<=31;set++)
          TX_pload[set] = RX_pload[set];

          for(pload=0;pload<=30;pload++)
          {
               if((pload % 3) == 0)
               TX_pload[pload]= TX_pload[pload] & 0xf0;
          }
          TX_pload[21] = Address_Route;
     }
     else if(CD == 8)
     {
          Rx_Real_Mode();

          for(set=0;set<=23;set++)
          TX_pload[set] = RX_pload[set];

          for(set=27;set<=31;set++)
          TX_pload[set] = RX_pload[set];

          for(pload=0;pload<=30;pload++)
          {
               if((pload % 3) == 0)
               TX_pload[pload]= TX_pload[pload] & 0xf0;
          }
          TX_pload[24] = Address_Route;
     }*/
     Rx_Real_Mode();
     CE_Pin_Active(CE_LOW);
}
```

```
void Read_Data ()
{
    int set;
    /*int SD;
    SD = ((Address_Route>>4) & 9);
    if(SD == 0)
    {
    if(((Address_Route & 0xf0) == (RX_pload[0] & 0xf0)) && ((((Address_Route & 0x0f)+2) == (RX_pload[6] & 0x0f)) || (((Address_Route & 0x0f)+15) == (RX_pload[30] & 0x0f)))
        {
          Decode_input();
          /*if(RX_pload[2] == 0x01)
          { Delay_ms(100); P00 = ~P00; Check = 1; }
          //P00 = ~P00;
          Check = 1;
        }
    if(RX_pload[6] == 0x00)
        {
        if(((Address_Route & 0xf0) == (RX_pload[0] & 0xf0)) && ((((Address_Route & 0x0f)+1) == (RX_pload[3] & 0x0f)) || (((Address_Route & 0x0f)+15) == (RX_pload[30] & 0
          //P02 = ~P02;
          Decode_input();
          Check = 1;
        }
    }
    else if(SD == 1)
    {
        if(((Address_Route & 0xf0) == (RX_pload[3] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[0] & 0x0f)))
        //if(((Address_Route>>4) == (RX_pload[3]>>4)) && (((Address_Route&0x0f)-1) && (RX_pload[0]&0x0f)))
          {
            Decode_input();
            /*if(RX_pload[5] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
            Check = 1;
          }
    }
    else if(SD == 2)
    {
        if(((Address_Route & 0xf0) && (RX_pload[6] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[3] & 0x0f)))
          {
            Decode_input();
            /*if(RX_pload[8] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
            if (TX_pload[7] == 0x00) P00 = ~P00;
            if (TX_pload[6] == 0x01) P01 = ~P01;
            Check = 1;
          }
    }
    else if(SD == 3)
    {
        if(((Address_Route & 0xf0) && (RX_pload[9] & 0xf0)) && ((((Address_Route & 0x0f)+2) == (RX_pload[15] & 0x0f)) || (((Address_Route & 0x0f)-2) == (RX_pload[3] & 0x0f
          {
            if(RX_pload[11] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
        if(RX_pload[15] == 0x00)
          {
          if(((Address_Route & 0xf0) && (RX_pload[9] & 0xf0)) && ((((Address_Route & 0x0f)+1) == (RX_pload[12] & 0x0f)) || (((Address_Route & 0x0f)-2) == (RX_pload[3] & 0x
            P02 = ~P02;
          }
    }
    else if(SD == 4)
    {
        if(((Address_Route & 0xf0) && (RX_pload[12] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[6] & 0x0f)))
          {
            if(RX_pload[14] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
    }
    else if(SD == 5)
    {
        if(((Address_Route & 0xf0) && (RX_pload[15] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[12] & 0x0f)))
          {
            if(RX_pload[17] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
    }
    else if(SD == 6)
    {
        if(((Address_Route & 0xf0) && (RX_pload[18] & 0xf0)) && ((((Address_Route & 0x0f)+2) == (RX_pload[24] & 0x0f)) || (((Address_Route & 0x0f)-3) == (RX_pload[6] & 0x0
          {
            if(RX_pload[20] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
        if(RX_pload[24] == 0x00)
          {
          if(((Address_Route & 0xf0) && (RX_pload[9] & 0xf0)) && ((((Address_Route & 0x0f)+1) == (RX_pload[21] & 0x0f)) || (((Address_Route & 0x0f)-3) == (RX_pload[2] & 0x
            P02 = ~P02;
          }
    }
    else if(SD == 7)
    {
        if(((Address_Route & 0xf0) && (RX_pload[21] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[18] & 0x0f)))
          {
            if(RX_pload[23] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
    }
    else if(SD == 8)
    {
        if(((Address_Route & 0xf0) && (RX_pload[24] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[21] & 0x0f)))
          {
            if(RX_pload[26] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
    }*/
    //P02 = ~P02;
    if(((Address_Route&0xf0)&&(RX_pload[30]&0xf0))&&(((Address_Route&0x0f)-15)==(RX_pload[0]&0x0f)))
    {
        for(set=0;set<=31;set++)
        {
            x = RX_pload[set];

        }
    }
}
```

```
void RF_Mode() interrupt 9 using 1
{
    RF = 0;
    P07 = ~P07;
    Rx_flag = 1;
    RF = 1;
}
void main()
{
 P0DIR = 0x10;
 P1DIR = 0x00;

 SOCON = 0x50;            //Mode 1 8-bit UART.
 SOBUF = 0x00;
 SORELH = 0x03;
 SORELL = 0xF3;
 ADCON = 0x80;

 PCON = PCON| 0x80;


 nRF_CE = 0;
 nRF_CS = 1;

 RFCKEN = 1;
 Delay_ms(500);

    CE_Pin_Active(CE_LOW);                                    // Set CE pin low to enable stanby mo
    Delay_ms(200);
    LO1_Flush_TX();
    LO1_Flush_RX();

    /*for(p=0;p<32;p++)
                {
                SOBUF = 'A';
                Delay_ms(1);
                while(TIO);
                TIO = 0 ;

                )*/
    //printf("Heeello");
    Delay_ms(500);

    Rx_Mode();
    RF = 1;
 ESO = 0;
 RIO = 0;
 TIO = 1;
 EA = 1;



 while(1)
 {
    if(RIO == 1)
    {
    P12 = ~P12;
    keep = (unsigned int)SOBUF;
        Datanet = keep;
        RIO = 0;
        LOOP = 1;
    //  printf("%c\r\n",keep);

    }

    if(LOOP == 1)
        {
            P06 = ~P06;


            //Decode_output();
            TX_pload[0] = 0x00;
            TX_pload[1] = 0x00;
            TX_pload[2] = Datanet;
            for (p=3;p<32;p++)
                {
                TX_pload[p] = 0x00;
                }
            TX_pload[30] = 0xff;

            Tx_Mode();
            Tx_Real_Mode();
            P01 = ~P01;

            //P06 = ~P06;
        /*  for (p=0;p<32;p++)
                {
                printf("%c",TX_pload[p]);
                )*/
            LOOP = 0;
            Delay_ms(100);

        }
            if(Rx_flag == 1)
            {
                P02 = ~P02;
                Check_Device();
                Read_Data();
                //CE_Pin_Active(CE_HIGH);
                Rx_flag = 0;


            /*  for(p=0;p<32;p++)
                {
                printf("%c",RX_pload[p]);
                )*/


            }
        Rx_Mode();

        Delay_ms(10);
   }
}
```

ก.2 ส่วนโปรแกรมการทำงานของไมโครคอนโทรลเลอร์สลาฟ

ไมโครคอนโทรลเลอร์สลาฟจะเชื่อมต่อกับวงจรส่งสัญญาณอินฟราเรดเพื่อส่งสัญญาณไปควบคุมเครื่องปรับอากาศ โดยรับคำสั่งมาจากมาสเตอร์ในรูปของแพคเกจข้อมูล จากนั้นไมโครคอนโทรลเลอร์จะทำการประมวลผล ถอดคำสั่งการทำงานและส่งสัญญาณอินฟราเรดไปสั่งงานเครื่องปรับอากาศตามคำสั่งของผู้ใช้งานพร้อมกับการส่งข้อมูลกลับไปยังมาสเตอร์เพื่อแสดงสถานะของคำสั่งที่ได้สั่งงานเครื่องปรับอากาศไปแล้ว และที่ตัวสลาฟนี้ยังมีอินฟราเรดรีซีฟเวอร์รับสัญญาณรีโมตปกติที่ใช้สั่งงานเครื่องปรับอากาศ เพื่อส่งคำสั่งนั้นไปแสดงสถานะการทำงานที่หน้าเว็บบราวเซอร์ได้เช่นเดียวกัน

```
#include <reg24le1.H>
#include <stdio.H>
#include "main.h"
#include "nrf_lab.h"
#include "nRF_defines.h"

#define  nRF_CE    RFCE
#define  nRF_CS    RFCSN
#define  SSPiBUF   SPIRDAT

#define  Address_Route  0x00
#define  Data_All  0

#define Scal_Timer  1/0.745
#define Delayup  3560*Scal_Timer
#define Delayup2 1646*Scal_Timer
#define Delayup3 1210*Scal_Timer
#define Delayup4 470*Scal_Timer
#define Delayup5 430*Scal_Timer
#define Delayup6 29104*Scal_Timer

idata unsigned char TH;
idata unsigned char TL;
//unsigned int timehead;
//unsigned int timehead1;
idata unsigned char State;
idata unsigned char State2;

idata int a,test1,Kapor;
idata unsigned char Byte_out1[9];

unsigned char temp;

unsigned int Rx_flag, Check, Decode_in, Decode_out = 0x0000, Send_Decode = 0 ,Loop_Print;

code unsigned char Close1[9]={136,91,228,15,0,0,0,64};
code unsigned char Close2[20]={136,91,228,0,0,12,60,0,245,0,0,0,0,0,0,3,0,0,183};
code unsigned char Open1[9]={136,91,228,15,0,0,0,64};
code unsigned char Open2[20]={136,91,228,0,0,140,60,0,245,0,0,0,0,0,0,3,0,0,119};
code unsigned char T30_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T30_2[20]={136,91,228,0,0,140,60,0,245,0,0,0,0,0,0,3,0,0,119};
code unsigned char T29_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T29_2[20]={136,91,228,0,0,140,92,0,245,0,0,0,0,0,0,3,0,0,55};
code unsigned char T28_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T28_2[20]={136,91,228,0,0,140,28,0,245,0,0,0,0,0,0,3,0,0,87};
code unsigned char T27_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T27_2[20]={136,91,228,0,0,140,108,0,245,0,0,0,0,0,0,3,0,0,23};
code unsigned char T26_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T26_2[20]={136,91,228,0,0,140,44,0,245,0,0,0,0,0,0,3,0,0,103};
code unsigned char T25_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T25_2[20]={136,91,228,0,0,140,76,0,245,0,0,0,0,0,0,3,0,0,39};
code unsigned char T24_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T24_2[20]={136,91,228,0,0,140,12,0,245,0,0,0,0,0,0,3,0,0,71};
code unsigned char T23_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T23_2[20]={136,91,228,0,0,140,116,0,245,0,0,0,0,0,0,3,0,0,7};
code unsigned char T22_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T22_2[20]={136,91,228,0,0,140,52,0,245,0,0,0,0,0,0,3,0,0,123};
code unsigned char T21_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T21_2[20]={136,91,228,0,0,140,84,0,245,0,0,0,0,0,0,3,0,0,59};
code unsigned char T20_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T20_2[20]={136,91,228,0,0,140,20,0,245,0,0,0,0,0,0,3,0,0,91};
code unsigned char T19_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T19_2[20]={136,91,228,0,0,140,100,0,245,0,0,0,0,0,0,3,0,0,27};
code unsigned char T18_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T18_2[20]={136,91,228,0,0,140,36,0,245,0,0,0,0,0,0,3,0,0,107};
code unsigned char T17_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T17_2[20]={136,91,228,0,0,140,68,0,245,0,0,0,0,0,0,3,0,0,43};
code unsigned char T16_1[9]={136,91,228,15,0,0,0,64};
code unsigned char T16_2[20]={136,91,228,0,0,140,4,0,245,0,0,0,0,0,0,3,0,0,75};
unsigned char timer_1time  ;

unsigned char THH[65];
//unsigned char TLL[65];

idata unsigned char Byte_out2[20];

unsigned char THH2[154];
//unsigned char TLL2[154];


void interruptup() interrupt  INTERRUPT_IPF using 2
{
  if(P06 == 0){
    P04 = ~P04;
    TR1 = 0;
    TH = TH1;
    TL = TL1;
    TH1 = 0;
    TL1 = 0;
    TR1 = 1;
//  HOK = 1;
    if (State !=0 && State2 == 0)
    {
      THH[State] = TH ;  //TLL[State] = TL ;
      State ++;
    }
    if (State2 !=0)
    {
      THH2[State2] = TH ;  //TLL2[State2] = TL ;
      State2 ++;
    }
    }
    Kapor = 1;
}
void Out_IR(){
 unsigned int  Loop_Print,out;
 unsigned int out1;
 bit  data1;
 unsigned char j,k;
 j = 0 ; k = 0;out = 0;
 // printf(" Part1\r\n");
    for( Loop_Print =1;Loop_Print<65;Loop_Print++){
                out =  out << 1;
               if ( THH[Loop_Print] == 0x4 ) { data1 = 0;out1 = 0;}
               if ( THH[Loop_Print] == 0x8 ) { data1 = 1;out1 = 1;}
               out = out | out1; //printf("%u ",out1);
                j++ ;
               if ( j== 8 ) {
                   j =0;
                   Byte_out1[k] = out;// printf(" \r\n");
                   out = 0;
                   k++;
                }

    }
```

```
      //printf(" Part2\r\n");
      out = 0; k=0; j=0;
      for( Loop_Print =1;Loop_Print<153;Loop_Print++){
                  out =  out << 1;
                if ( THH2[Loop_Print] == 0x4 ) { data1 = 0;out1 = 0;}
                if ( THH2[Loop_Print] == 0x8 ) { data1 = 1;out1 = 1;}
                out = out | data1; //printf("%u ",out1);

                j++ ;
                if ( j== 8 ) {

                    j =0;
                    Byte_out2[k] = out;// printf(" \r\n");
                    out = 0;
                    k++;
                 }


      }

      State = 0;State2 = 0;

}
void Delay_us(unsigned int i)
  {
     TH1 = (0xffff-i)>>8;
     TL1 = (0xffff-i);
     TR1 =1;
     timer_1time = 1;
     while(timer_1time);
  }
void timer1() interrupt 3 using 2
  {
     TF1=0;
     TR1 = 0;
     timer_1time = 0;
  }

 void Delay_1us(volatile unsigned int n){
   unsigned char i;
   while(n--)
     for(i=0;i<3;i++);
 }
 void Delay_100us(volatile unsigned int n){
   unsigned char i;
   while(n--)
     for(i=0;i<35;i++);
 }

 void Delay_ms(volatile unsigned int n){
   while(n--)
     Delay_100us(50);
 }

 void RF_Mode() interrupt 9 using 1
{
     RF = 0;
     //P07 = ~P07;
     Rx_flag = 1;
     RF = 1;
}

void Duty_50()
  {
     PWMDC0 = 0x0f;
  }
void Duty_0()
  {
     PWMDC0 = 0x00;
  }
 void Ir_Pulse_Head()
  {
      Duty_50();
      Delay_us(Delayup);
      Duty_0();
      Delay_us(Delayup2);
  }

void Ir_Pulse_Tail()
  {
      Duty_50();
      Delay_us(Delayup4);
      Duty_0();
      Delay_us(Delayup6);
  }

 void Ir_Pulse_1()
  {
      Duty_50();
      Delay_us(Delayup4);
      Duty_0();
      Delay_us(Delayup3);
      Duty_50();
  }

 void Ir_Pulse_0()
  {
      Duty_50();
      Delay_us(Delayup4);
      Duty_0();
      Delay_us(Delayup5);
      Duty_50();
  }
void Out_Ir_Remote(char data_Ir)
  {
      unsigned char Bit_data;

      Bit_data =  data_Ir & 0x80;
      if (Bit_data == 0x80)(Ir_Pulse_1();)
        else( Ir_Pulse_0(); )

      Bit_data =  data_Ir & 0x40;
      if (Bit_data == 0x40)(Ir_Pulse_1();)
        else(Ir_Pulse_0(); )

      Bit_data =  data_Ir & 0x20;
      if (Bit_data == 0x20)(Ir_Pulse_1();)
        else(Ir_Pulse_0(); )

      Bit_data =  data_Ir & 0x10;
      if (Bit_data == 0x10)(Ir_Pulse_1();)
        else(Ir_Pulse_0(); )

     Bit_data =  data_Ir & 0x08;
      if (Bit_data == 0x08)(Ir_Pulse_1();)
        else(Ir_Pulse_0(); )

     Bit_data =  data_Ir & 0x04;
      if (Bit_data == 0x04)(Ir_Pulse_1();)
        else(Ir_Pulse_0(); )

     Bit_data =  data_Ir & 0x02;
      if (Bit_data == 0x02)(Ir_Pulse_1();)
        else(Ir_Pulse_0(); )

     Bit_data =  data_Ir & 0x01;
      if (Bit_data == 0x01)(Ir_Pulse_1();)
        else(Ir_Pulse_0(); )
  }
```

```
void Sent_IR_Bf(unsigned char type1[],unsigned char type2[])
{
    unsigned int ii;
    P15 = 1;
    Ir_Pulse_Head();

    for (ii=0;ii<8;ii++){
         Out_Ir_Remote(type1[ii]);
        }
    Ir_Pulse_Tail();
    Ir_Pulse_Head();
    for (ii=0;ii<19;ii++){
         Out_Ir_Remote(type2[ii]);
        }
    Ir_Pulse_Tail();
    Delay_ms(1000);
    P15 = 0;
}


void CE_Pin_Active(unsigned char action)     // CE pin high, low or pulse..
{
  switch(action)
  {
    case CE_LOW:                        // action == 0, CE low
      nRF_CE = 0;
    break;

    case CE_HIGH:                       // action == 1, CE high
      nRF_CE = 1;
    break;

    case CE_PULSE:                      // action == 2, CE pulse (10?s)
     // TMR2    = 0;
      nRF_CE = 1;                       // Set CE pin high
       Delay_1us(100);
       nRF_CE = 0;
      //TMR2ON_bit = 1;                   // Start Timer2, CE pulse timer
    break;
  }
}
unsigned char LO1_Clear_IRQ(unsigned char irq_flag)
{
  return nRF_SPI_RW_Reg(WRITE_REG + STATUS, irq_flag);
}

unsigned char nRF_SPI_RW(unsigned char byte1)
{
  unsigned char status;
  //Delay_1us(10);
  SSP1BUF = byte1;
  Delay_1us(1);          // Delay 300s ok
  status  = SSP1BUF;
  return(status);
}
unsigned char nRF_SPI_RW_Reg(unsigned char reg, unsigned char value)
{
  unsigned char status;

  nRF_CS = 0;                                // CSN low, init SPI transaction
  status = nRF_SPI_RW(reg);                  // select register
  nRF_SPI_RW(value);                         // ..and write value to it..
  nRF_CS = 1;                                // CSN high again

  return(status);                            // return nRF24LO1 status byte
}
unsigned char nRF_SPI_Write_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
{
  unsigned char status, byte_ctr;

  nRF_CS = 0;                                // Set CSN low, init SPI tranaction
  status = nRF_SPI_RW(reg);                  // Select register to write to and read status byte

  for(byte_ctr=0; byte_ctr<bytes; byte_ctr++)     // then write all byte in buffer(*pBuf)
  {
    nRF_SPI_RW(*pBuf++);
  }


  nRF_CS = 1;                                // Set CSN high again

  return(status);                            // return nRF24LO1 status byte
}

unsigned char LO1_RD_RX_PW_n(unsigned char pipe)
{
  return nRF_SPI_Read(RX_PW_P0 + pipe);
}
void LO1_Write_TX_Pload(unsigned char *pBuf, unsigned char plWidth)
{
  nRF_SPI_Write_Buf(WR_TX_PLOAD, pBuf, plWidth);
}

unsigned char Send_Packet()                          // Send one data packet, controlled by Button1(SW1)
{
    LO1_Write_TX_Pload(&TX_pload, TX_PLOAD_WIDTH);     // Write new TX payload
    CE_Pin_Active(CE_PULSE);                               // Enable CE pulse, 12?s
  //ucTry_Ctr = 0;                                     // Reset Try_Ctr before nex transmitt..
  return 1;
}
void LO1_Flush_RX(void)
{
  nRF_SPI_RW_Reg(FLUSH_RX,0);
}
void LO1_Flush_TX(void)
{
  nRF_SPI_RW_Reg(FLUSH_TX,0);
}
unsigned char LO1_Get_FIFO(void)
{
  return nRF_SPI_Read(FIFO_STATUS);
}
unsigned char nRF_SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
{
  unsigned char status, byte_ctr;
```

```
  nRF_CS = 0;                                     // Set CSN low, init SPI transaction
  status = nRF_SPI_RW(reg);                       // Select register to write to and read status byte

  for(byte_ctr=0;byte_ctr<bytes;byte_ctr++)       // Perform SPI_RW to read byte from nRF24L01
  {
    pBuf[byte_ctr] = nRF_SPI_RW(0);
  }

  nRF_CS = 1;                                     // Set CSN high again

  return(status);                                 // return nRF24L01 status byte
}
unsigned char nRF_SPI_Read(unsigned char reg)
{
  unsigned char reg_val;

  nRF_CS = 0;                                     // CSN low, initialize SPI communication...
  nRF_SPI_RW(reg);                                // Select register to read from..
  reg_val = nRF_SPI_RW(0);                        // ..then read registervalue
  nRF_CS = 1;                                     // CSN high, terminate SPI communication

  return(reg_val);                                // return register value
}
unsigned char LO1_Get_Status(void)
{
  return nRF_SPI_Read(STATUS);
}
unsigned char LO1_Get_Current_Pipenum(void)
{
  return ((LO1_Get_Status() & RX_P_NO) >> 1);
}
unsigned int LO1_Read_RX_Pload(unsigned char *pBuf)
{
  unsigned char plWidth, pipe;
  plWidth = LO1_RD_RX_PW_n(pipe = LO1_Get_Current_Pipenum());   // Read current pipe's payload width
  nRF_SPI_Read_Buf(RD_RX_PLOAD, pBuf, plWidth);                 // Then get RX data

  return ((pipe << 8) + plWidth);                               // return pipe# & pipe#.plWidth
}
 /*unsigned char nRF_SPI_RW_Reg_IRQ(unsigned char reg, unsigned char value)
{
  unsigned char status;

  nRF_CS = 0;                                   // CSN low, init SPI transaction

  SSP1BUF = reg;
  //while (SSP1STAT.BF == 0)
  Delay_1us(10);
  status = SSP1BUF;  // free the register

  SSP1BUF = value;
  //while (SSP1STAT.BF == 0)
    Delay_1us(10);
  reg = SSP1BUF;  // free the register

  nRF_CS = 1;                                   // CSN high again

  return(status);                               // return nRF24L01 status byte
}*/
void Tx_Mode ()
{
      CE_Pin_Active(CE_LOW);                                // Set CE pin low to enable stanby mode
      Delay_ms(200);
      LO1_Flush_TX();
      LO1_Flush_RX();
      //LO1_Clear_IRQ(MASK_IRQ_FLAGS);                      // Clear interrupts
      nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
      //ucIRQ_Source = CLEAR;
      //ucLastStat = ucLinkStat = LINK_ESTABLISH;           // LINK_ESTABLISH = 0x02

      nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f);              // Enable Auto.Ack:Pipes 0-5
      nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f);          // Enable Pipes 0-5
      nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a);         // 500?s + 86?s, 10 retrans...
      nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40);              // Select RF channel 40
      nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03);           // Select Addres
      nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08);           // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
      nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e);             // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..

      nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_P0, sizeof(ADDRESS_P0));    // Writes TX_Address to nRF24L01
      nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, &ADDRESS_P0, sizeof(ADDRESS_P0)); // RX_Addr0 same as TX_Adr for Auto.Ack
      CE_Pin_Active(CE_HIGH);
}
/*void Tx_Mode_runtime()
{
      CE_Pin_Active(CE_LOW);                                // Set CE pin low to enable stanby mode
      //Delay_ms(200);
      LO1_Flush_TX();
      LO1_Flush_RX();
      LO1_Clear_IRQ(MASK_IRQ_FLAGS);                      // Clear interrupts
      //nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
      //ucIRQ_Source = CLEAR;
      //ucLastStat = ucLinkStat = LINK_ESTABLISH;           // LINK_ESTABLISH = 0x02

      //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f);              // Enable Auto.Ack:Pipes 0-5
      //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f);          // Enable Pipes 0-5
      //nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a);         // 500?s + 86?s, 10 retrans...
      //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40);              // Select RF channel 40

      //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08);           // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
      //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e);             // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..

      //nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_P0, sizeof(ADDRESS_P0));    // Writes TX_Address to nRF24L01
      //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, &ADDRESS_P0, sizeof(ADDRESS_P0)); // RX_Addr0 same as TX_Adr for Auto.Ack
      CE_Pin_Active(CE_HIGH);
      Delay_ms(50);                                       //max 1 ms
      //CE_Pin_Active(CE_LOW);
}*/
void Rx_Mode ()
{

      CE_Pin_Active(CE_LOW);                                // Set CE pin low to enable stanby mode
      Delay_ms(1);                                          // 200ms
      LO1_Flush_TX();
      LO1_Flush_RX();

      LO1_Clear_IRQ(MASK_IRQ_FLAGS);                        // Clear interrupts

      nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f);              // Enable Auto.Ack:Pipes 0-5
      nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f);          // Enable Pipes 0-5
      nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40);              // Select RF channel 40
      nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03);           // Select Addres
      nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x0f);           // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
      nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f);             // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..

      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 device
      nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));

      nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
      nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
      nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
      nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);

      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P0, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_PLOAD_WIDTH);
      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_PLOAD_WIDTH);
      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_PLOAD_WIDTH);
      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_PLOAD_WIDTH);
      nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_PLOAD_WIDTH);

      CE_Pin_Active(CE_HIGH);
}
```

```
void Rx_Mode_Runtime()
{
      //CE_Pin_Active(CE_LOW);                                  // Set CE pin low to enable standby mode
      //Delay_ms(1);                                              // 200ms
      //LO1_Flush_TX();
      //LO1_Flush_RX();

      //LO1_Clear_IRQ(MASK_IRQ_FLAGS);                          // Clear interrupts

      //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f);                // Enable Auto.Ack:Pipes 0-5
      //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f);            // Enable Pipes 0-5
      //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40);                  // Select RF channel 40
      //nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03);             // Select Address
      //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x0f);             // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
      //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f);               // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..

      //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 devi
      //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));

      //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);

      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P0, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_PLOAD_WIDTH);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_PLOAD_WIDTH);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_PLOAD_WIDTH);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_PLOAD_WIDTH);
      //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_PLOAD_WIDTH);

      CE_Pin_Active(CE_HIGH);
      Delay_ms(30);
}
void Rx_Real_Mode()
{
    Rx_Mode_Runtime();
          do
          {
            //P00 = ~P00;
            uiRX_info = LO1_Read_RX_Pload(RX_pload);        // Read current payload
            // temp  = LO1_Get_FIFO() & 0x02;     temp  >> 1;

          }
          while(!(LO1_Get_FIFO() & RX_EMPTY));              // ..until FIFO empty
           //while( temp == 0 );
          temp  = LO1_Get_FIFO() & 0x01;     // temp  >> 1;

          //P14 = temp;

          //uiRX_info = LO1_Read_RX_Pload(RX_pload);
          LO1_Clear_IRQ(MASK_RX_DR_FLAG);

          /*if (RX_pload[0] == 0x31){ P00 = ~P00; RX_pload[0] = 0; }
          if (RX_pload[1] == 0x32){ P01 = ~P01; RX_pload[1] = 0; }
          if (RX_pload[2] == 0x33){ P02 = ~P02; RX_pload[2] = 0; }
          if (RX_pload[3] == 0x34){ P03 = ~P03; RX_pload[3] = 0; }
          if (RX_pload[4] == 0x35){ P04 = ~P04; RX_pload[4] = 0; }*/
          CE_Pin_Active(CE_LOW);
}
void Tx_Real_Mode()
{
      Send_Packet();
      Delay_ms(1);
      LO1_Clear_IRQ(MASK_TX_DS_FLAG);
}
void Decode_output()
{
    int a;
    for(a = 0;a<32;a++ ){
        TX_pload[a] = RX_pload[a];
    }

}
void Decode_input()
{
    int DI;
    DI = (3*(((Address_Route>>4) & 9))+1);

    switch ( (RX_pload[DI] & 0xF0) >> 4 )
    {
        case 0:
            switch (RX_pload[2]){
              case 0x41:
                 Sent_IR_Bf(Close1,Close2);
              break;
              case 0x42:
                 Sent_IR_Bf(Open1,Open2);
              break;
              case 0x43:
                 Sent_IR_Bf(T30_1,T30_2);
              break;
              case 0x44:
                 Sent_IR_Bf(T29_1,T29_2);
              break;
              case 0x45:
                 Sent_IR_Bf(T28_1,T28_2);
              break;
              case 0x46:
                 Sent_IR_Bf(T27_1,T27_2);
              break;
              case 0x47:
                 Sent_IR_Bf(T26_1,T26_2);
              break;
              case 0x48:
                 Sent_IR_Bf(T25_1,T25_2);
              break;
              case 0x49:
                 Sent_IR_Bf(T24_1,T24_2);
              break;
              case 0x4a:
                 Sent_IR_Bf(T23_1,T23_2);
              break;
              case 0x4b:
                 Sent_IR_Bf(T22_1,T22_2);
              break;
              case 0x4c:
                 Sent_IR_Bf(T21_1,T21_2);
              break;
              case 0x4d:
                 Sent_IR_Bf(T20_1,T20_2);
              break;
              case 0x4e:
                 Sent_IR_Bf(T19_1,T19_2);
              break;
              case 0x4f:
                 Sent_IR_Bf(T18_1,T18_2);
              break;
              case 0x50:
                 Sent_IR_Bf(T17_1,T17_2);
              break;
              case 0x51:
                 Sent_IR_Bf(T16_1,T16_2);
              break;
            }
        break;

    }
}
```

```
void Check_Device()
{
    int CD,set,pload;
    CD = ((Address_Route>>4) % 9);
    if(CD == 0)
    {
        Rx_Real_Mode();

        for(set=3;set<=31;set++)
        TX_pload[set] = RX_pload[set];

        for(pload=0;pload<=30;pload++)
        {
            if((pload % 3) == 0)
            TX_pload[pload]= TX_pload[pload] & 0xf0;
        }
        TX_pload[0] = Address_Route;
    }
    else if(CD == 1)
    {
        Rx_Real_Mode();

        for(set=0;set<=2;set++)
        TX_pload[set] = RX_pload[set];

        for(set=6;set<=31;set++)
        TX_pload[set] = RX_pload[set];

        for(pload=0;pload<=30;pload++)
        {
            if((pload % 3) == 0)
            TX_pload[pload]= TX_pload[pload] & 0xf0;
        }
        TX_pload[3] = Address_Route;
    }
    else if(CD == 2)
    {
        Rx_Real_Mode();

        for(set=0;set<5;set++)
        TX_pload[set] = RX_pload[set];

        for(set=9;set<=31;set++)
        TX_pload[set] = RX_pload[set];

        for(pload=0;pload<=30;pload++)
        {
            if((pload % 3) == 0)
            TX_pload[pload]= TX_pload[pload] & 0xf0;
        }
        TX_pload[6] = Address_Route;
    }
    else if(CD == 3)
    {
        Rx_Real_Mode();

        for(set=0;set<=8;set++)
        TX_pload[set] = RX_pload[set];

        for(set=12;set<=31;set++)
        TX_pload[set] = RX_pload[set];

        for(pload=0;pload<=30;pload++)
        {
            if((pload % 3) == 0)
            TX_pload[pload]= TX_pload[pload] & 0xf0;
        }
        TX_pload[9] = Address_Route;
    }
    else if(CD == 4)
    {
        Rx_Real_Mode();

        for(set=0;set<=11;set++)
        TX_pload[set] = RX_pload[set];

        for(set=15;set<=31;set++)
        TX_pload[set] = RX_pload[set];

        for(pload=0;pload<=30;pload++)
        {
            if((pload % 3) == 0)
            TX_pload[pload]= TX_pload[pload] & 0xf0;
        }
        TX_pload[12] = Address_Route;
    }
    else if(CD == 5)
    {
        Rx_Real_Mode();

        for(set=0;set<=14;set++)
        TX_pload[set] = RX_pload[set];

        for(set=18;set<=31;set++)
        TX_pload[set] = RX_pload[set];

        for(pload=0;pload<=30;pload++)
        {
            if((pload % 3) == 0)
            TX_pload[pload]= TX_pload[pload] & 0xf0;
        }
        TX_pload[15] = Address_Route;
    }
    else if(CD == 6)
    {
        Rx_Real_Mode();

        for(set=0;set<=17;set++)
        TX_pload[set] = RX_pload[set];

        for(set=21;set<=31;set++)
        TX_pload[set] = RX_pload[set];

        for(pload=0;pload<=30;pload++)
        {
            if((pload % 3) == 0)
            TX_pload[pload]= TX_pload[pload] & 0xf0;
        }
        TX_pload[18] = Address_Route;
    }
    else if(CD == 7)
    {
        Rx_Real_Mode();

        for(set=0;set<=20;set++)
        TX_pload[set] = RX_pload[set];

        for(set=24;set<=31;set++)
        TX_pload[set] = RX_pload[set];

        for(pload=0;pload<=30;pload++)
        {
            if((pload % 3) == 0)
            TX_pload[pload]= TX_pload[pload] & 0xf0;
        }
        TX_pload[21] = Address_Route;
    }
    else if(CD == 8)
    {
        Rx_Real_Mode();

        for(set=0;set<=23;set++)
        TX_pload[set] = RX_pload[set];

        for(set=27;set<=31;set++)
        TX_pload[set] = RX_pload[set];

        for(pload=0;pload<=30;pload++)
        {
            if((pload % 3) == 0)
            TX_pload[pload]= TX_pload[pload] & 0xf0;
        }
        TX_pload[24] = Address_Route;
    }
    CE_Pin_Active(CE_LOW);
}
```

```
void Read_Data ()
{
    int SD;
    SD = ((Address_Route>>4) % 9);
    if(SD == 0)
    {
    if(((Address_Route & 0xf0) == (RX_pload[0] & 0xf0)) && ((((Address_Route & 0x0f)+2) == (RX_pload[6] & 0x0f)) || (((Address_Route & 0x0f)+15) == (RX_pload[30] & 0x0f)))
          {
            Decode_input();
            /*if(RX_pload[2] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }*/
            //P00 = ~P00;
            Check = 1;
          }
    if(RX_pload[6] == 0x00)
          {
          if(((Address_Route & 0xf0) == (RX_pload[0] & 0xf0)) && ((((Address_Route & 0x0f)+1) == (RX_pload[3] & 0x0f)) || (((Address_Route & 0x0f)+15) == (RX_pload[30] & 0
            ()//P02 = ~P02;
          }
    }
    else if(SD == 1)
    {
        if(((Address_Route & 0xf0) == (RX_pload[3] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[0] & 0x0f)))
        //if(((Address_Route>>4) == (RX_pload[3]>>4)) && (((Address_Route&0x0f)-1) && (RX_pload[0]&0x0f)))
          {
            Decode_input();
            /*if(RX_pload[5] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }*/
            Check = 1;
          }
    }
    else if(SD == 2)
    {
        if(((Address_Route & 0xf0) && (RX_pload[6] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[3] & 0x0f)))
          {
            Decode_input();
            /*if(RX_pload[8] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }*/
            //if (TX_pload[7] == 0x00) P00 = ~P00;
            //if (TX_pload[8] == 0x01) P01 = ~P01;
            Check = 1;
          }
    }
    else if(SD == 3)
    {
        if(((Address_Route & 0xf0) && (RX_pload[9] & 0xf0)) && ((((Address_Route & 0x0f)+2) == (RX_pload[15] & 0x0f)) || (((Address_Route & 0x0f)-2) == (RX_pload[3] & 0x0f
          {
            if(RX_pload[11] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
        if(RX_pload[15] == 0x00)
          {
          if(((Address_Route & 0xf0) && (RX_pload[9] & 0xf0)) && ((((Address_Route & 0x0f)+1) == (RX_pload[12] & 0x0f)) || (((Address_Route & 0x0f)-2) == (RX_pload[3] & 0x
            ()//P02 = ~P02;
          }
    }
    else if(SD == 4)
    {
        if(((Address_Route & 0xf0) && (RX_pload[12] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[6] & 0x0f)))
          {
            if(RX_pload[14] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
    }
    else if(SD == 5)
    {
        if(((Address_Route & 0xf0) && (RX_pload[15] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[12] & 0x0f)))
          {
            if(RX_pload[17] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
    }
    else if(SD == 6)
    {
        if(((Address_Route & 0xf0) && (RX_pload[18] & 0xf0)) && ((((Address_Route & 0x0f)+2) == (RX_pload[24] & 0x0f)) || (((Address_Route & 0x0f)-3) == (RX_pload[6] & 0x0
          {
            if(RX_pload[20] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
        if(RX_pload[24] == 0x00)
          {
          if(((Address_Route & 0xf0) && (RX_pload[9] & 0xf0)) && ((((Address_Route & 0x0f)+1) == (RX_pload[21] & 0x0f)) || (((Address_Route & 0x0f)-3) == (RX_pload[2] & 0x
            ()//P02 = ~P02;
          }
    }
    else if(SD == 7)
    {
        if(((Address_Route & 0xf0) && (RX_pload[21] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[18] & 0x0f)))
          {
            if(RX_pload[23] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
    }
    else if(SD == 8)
    {
        if(((Address_Route & 0xf0) && (RX_pload[24] & 0xf0)) && (((Address_Route & 0x0f)-1) == (RX_pload[21] & 0x0f)))
          {
            if(RX_pload[26] == 0x01)
            { Delay_ms(100); P00 = ~P00; Check = 1; }
          }
    }
}
```

```
void main()
{
  EF = 1;
  EA = 1;

    // Set Port INT0,INT1
    P0DIR = 0x70;
    P1DIR = 0x00;
    P04 = 1;
    P05 = 1;
    INTEXP = 0x1A;          //set INT0 INT1
    IT1 = 1;
    IT0 = 1;
    ////////////////////////
    //  Set Timer1 //
    TMOD = 0x10;   //  01    Mode 1: 16 bit counter/timer
    TH1 = 0 ;
    TL1 = 0 ;
    ET1 = 1;
    //////UART//////////
     SCON = 0x50;            //Mode 1 8 bit UART.
     SUBUF = 0x00;
     P0PELM = 0x03;
     SUBELL = 0xF2;
     ADCON = 0x80;
     PCON = PCON| 0x80;

    //    Interrupt //
/*  EXU = 1;
    EXI = 1; */
    // PWM  //
     PWMCON = 0x31;


  nRF_CE = 0;
  nRF_CS = 1;

  RFCKEN = 1;
  //Delay_ms(500);

  CE_Pin_Active(CE_LOW);                                    // Set CE pin low to enable standby mode
//  Delay_ms(200);
  LO1_Flush_TX();
  LO1_Flush_RX();

//  P0 = 0x00;
//  Delay_ms(500);
  Rx_Mode();

    while(1)
  {
        P01 = 0;
        if(Rx_flag == 1)
        {
            Check_Device();
   //Delay_ms(500);

     CE_Pin_Active(CE_LOW);                                 // Set CE pin low to enable standby mode
 //   Delay_ms(200);
     LO1_Flush_TX();
     LO1_Flush_RX();

 //   P0 = 0x00;
 //   Delay_ms(500);
     Rx_Mode();

     while(1)
   {
         P01 = 0;
         if(Rx_flag == 1)
         {
             Check_Device();
             Read_Data();
             P00 = ~P00;

             if(Check == 1)
             {
                 EF = 0;
                 P01 = ~P01;
                 Decode_Output();
                 Tx_Mode();
                 Tx_Real_Mode();
                 Delay_ms(10);
                 Check = 0;
                 EF = 1;
             }

             Rx_Mode();
             Rx_Flag = 0;
         }
     }
     //  P04 = EFF;
      Delay_ms(10);
     if (Kapor == 1)
     {
     EF = 0;
      if ( TH <= 0x1b && TH >= 0x1a  && State==0 )
      {
        THH[0] = TH ;  //TLL[0] = TL ;
        State=1;
      }
      if ( TH <= 0x1b && TH >= 0x1a  && State>=65 && State2==0 )
      {
       THH2[0] = TH ;  //TLL2[0] = TL ;
       State2=1;
      }
      if ( State >=65 && State2 >= 15)
      {
        EA=0;
        Out_LK();
                Loop_Print = Byte_out2[18];
                printf("Loopback \r\n",Loop_Print );
        switch (Byte_out2[18]){
            case 103:
                printf("Close");
                TX_pload[2] = 0x41;
            break;
            case 110:
                printf("Open");
                TX_pload[2] = 0x42;
            break;
            case 55:
                printf("TT_29");
                TX_pload[2] = 0x44;
            break;
            case 87:
                printf("TT_28");
                TX_pload[2] = 0x45;
            break;
            case 23:
                printf("TT_27");
                TX_pload[2] = 0x46;
            break;
            case 103:
                printf("TT_26");
                TX_pload[2] = 0x47;
            break;
            case 39:
                printf("TT_25");
                TX_pload[2] = 0x48;
            break;
            case 71:
                printf("TT_24");
                TX_pload[2] = 0x49;
            break;
            case 7:
                printf("TT_23");
                TX_pload[2] = 0x4A;
            break;
            case 123:
                printf("TT_22");
                TX_pload[2] = 0x50;
            break;
            case 59:
                printf("TT_21");
                TX_pload[2] = 0x51;
            break;
            case 91:
                printf("TT_20");
                TX_pload[2] = 0x52;
            break;
            /*case 27:
                printf("TT_19");
            break;
            case 107:
                printf("TT_18");
            break;
            case 43:
                printf("TT 17");
            break;
            case 75:
                printf("TT 16");
            break;*/
        }
        /*if(Lock1 == 0)
        {
          printf("NO");
        }*/
        EA = 1;
        Tx_Mode();
        Tx_Real_Mode();
        Delay_ms(10);
        Kapor = 0;
        EF = 1;
        }
    }
  }
}
```

ก.3 ส่วนโปรแกรมการสร้างเว็บเพจ

โครงงานนี้จะใช้โปรแกรม Dreamweaver ในการสร้างเว็บเพจ ซึ่งจะใช้ภาษา html ในการเขียนโค้ดการทำงาน เพื่อใช้ในการสั่งงานไมโครคอนโทรลเลอร์ให้ควบคุมการทำงานของเครื่องปรับอากาศสามารถแสดงได้ดังนี้

```
<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"><img src="images/Project-Air_01.jpg" width="955" height="193"
/></td>
</tr>
<tr>
<th width="266" bgcolor="#FFFFFF"><p><a href="ftp://www.ehouse-
air.com//ProjectWeb/IndexNew.htm"><img src="images/Project-Air_04.jpg"
width="197" height="66" /></a></p>
<p><a href="project-air.pptx"><img src="images/Project-Air_10.jpg" width="197"
height="67" /></a></p>
<p><img src="images/Project-Air_21.jpg" width="197" height="65" /></p></th>
<th width="689" align="center" bgcolor="#FFFFFF"><p> </p>
<p><img src="img_air.jpg" width="474" height="76" /></p>
<p><img src="New folder/images/Project-Air_34.gif" width="687" height="32" /></p>
```

```
<table width="150" border="1" cellspacing="5" cellpadding="15">
<tr>
<th width="121" bgcolor="#FFFFFF"><h1></h1></th>
</tr>
</table>
<p><img src="images/Project-Air_12.gif" width="686" height="33" /></p>
<table width="300" border="0" align="center" cellpadding="0" cellspacing="0">
<tr>
<th width="150"><p><a href="http://localhost/sentdataOpen.php"><img
src="images/Project-Air_17.jpg" width="82" height="72" /></a></p></th>
<th width="150">
<p><a href="http://localhost/sentdataClose.php"><img src="images/Project-
Air_19.jpg" width="82" height="72" /></a></p></th>
</tr>
</table><br>
</th>
</tr>
<tr>
<th nowrap="nowrap" bgcolor="#FFFFFF"><p> </p>
<p><a href="http://www.kmitl.ac.th/engineer/index.php"><img src="Pic/Bachelor of
Engineering (Automation Engineering) of Technology Ladkrabang_files/logo002.jpg"
width="175" height="50" /></a></p>
<p><a href="http://www.kmitl.ac.th/main.php"><img src="Pic/Bachelor of Engineering
(Automation Engineering) of Technology Ladkrabang_files/logo001.jpg" width="175"
height="50" /></a></p>
<p><img src="Pic/Bachelor of Engineering (Automation Engineering) of Technology
Ladkrabang_files/logo003.jpg" width="175" height="50" /></p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
```

```
<p> </p></th>
<th bgcolor="#FFFFFF"><p><img src="auto.jpg" width="686" height="33" /><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"><img
src="images/Project-Air_17.jpg" width="82" height="72" /></a></p></th>
<th width="150">
<p><a href="http://localhost/sentdataClose2.php"><img src="images/Project-
Air_19.jpg" width="82" height="72" /></a></p></th>
</tr>
</table>
<p><img src="images/Project-Air_25.gif" width="687" height="37" /></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>
```

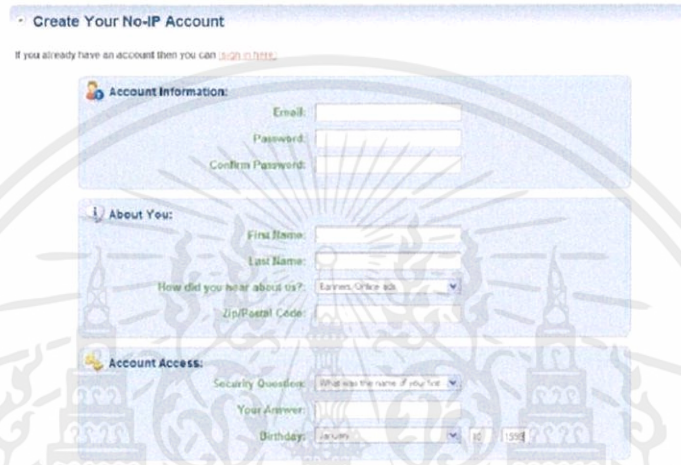
```
<tr>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_26.php">26</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_27.php">27</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_28.php">28</a></h1></th>
<th bgcolor="#FFFFFF"><h1><a
href="http://localhost/sentdataT_29.php">29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

ก.4 ขั้นตอนการสมัครNo-ipและการติดตั้งNo-ip

No-ipคือการสร้างCreate Domainย่อยมาใช้งานหรือDDNSนั้นเองมีประโยชน์ในการRemoteมาสั่งงานServer ในกรณีที่ไม่ได้อยู่ในวง Network เดียวกัน ทำให้การจัดการเกิดขึ้นที่ไหนบนโลกนี้ก็ได้ ขั้นตอนการสมัครมีดังนี้

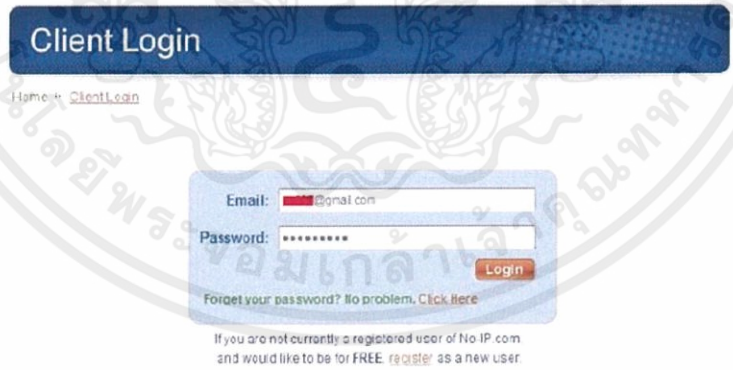
1. เข้าไปที่เว็บไซต์ http://www.no-ip.comเพื่อสมัครสมาชิกเพื่อสมัครสมาชิก จากนั้นไปที่Create Account

2. กรอกข้อมูลให้ครบแล้วเลือกที่ Accept



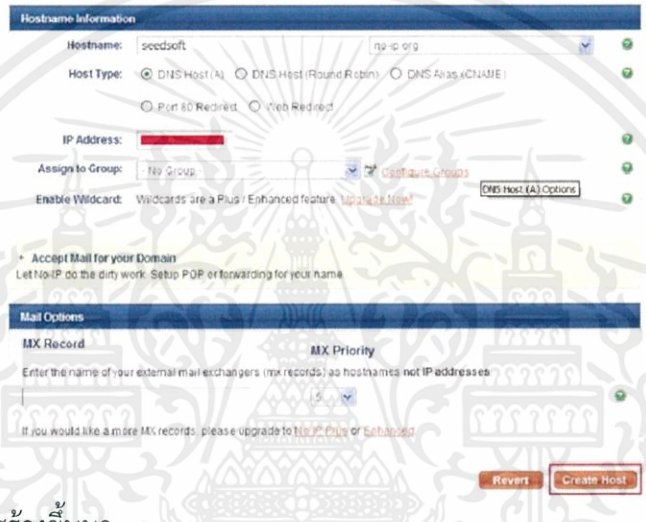
3. ทำการยืนยันทางอีเมลล์

4. ล๊อกอินโดยใช้อีเมลล์และพาสเวิร์ด

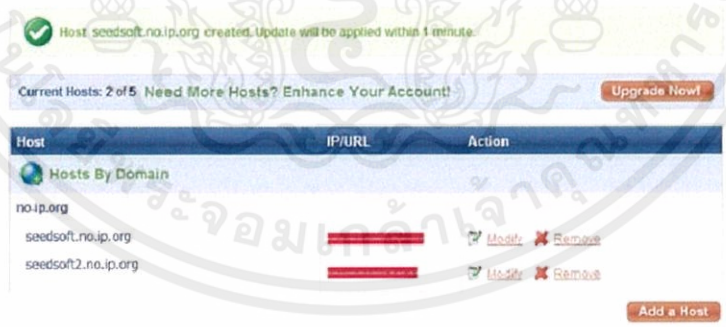


5. เมื่อล็อกอินแล้วให้เลือก Add a Host

6. ใส่ชื่อ Hostname ที่ต้องการ และกด Create Host



7. no-ip ที่สร้างขึ้นมา



ภาคผนวก ข
คู่มือการใช้งานอุปกรณ์ในโครงงาน

ในส่วนของภาคผนวกข. จะอธิบายในรายละเอียดสำคัญของอุปกรณ์ที่ประกอบอยู่ภายในส่วนต่างๆของระบบควบคุมเครื่องปรับอากาศผ่านเครือข่ายอินเทอร์เน็ต

ข.1 อินฟราเรดแอลอีดี

OASIS

TAIWAN OASIS LED DATA SHEET (FOR INFRARED)

PART NO. : TOIR-50b94bCEa

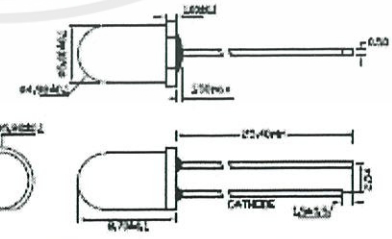
ABSOLUTE MAXIMUM RATINGS AT TA=25°C

PARAMETER	SYMBOL	DATA	UNIT
Forward Current	I_{FM}	100	mA
Peak Forward Current (duty=1/100, f=100HZ)	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 Sold 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	$P_{O(TYP.)}$	12.0	mW	Distance: 10cm I_F=50mA Detector Area: 1cm²
Forward Voltage	V_F	TYP: 1.25 MAX: 1.45	V	I_F=20mA
Wavelength	λP	940	nm	I_F=20mA
Spectrum Width of Half Value	$\triangle\lambda$	50	nm	I_F=20mA
Reverse Current	I_R	10	uA	V_R=5V
Full Viewing Angle	$2\times\frac{1}{2}\theta$	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	±0.25 UNLESS ANY	DRAWN	[illegible]	CHECKED	
UNIT	M/M	SHEET NO.	1/2	DRAWING NO.	S-50b94bCEa-1	CUSTOMER		APPROVED	

ข.2 ตัวรับสัญญาณอินฟราเรด (IR Receiver)

VISHAY

TSOP48..

Vishay Telefunken

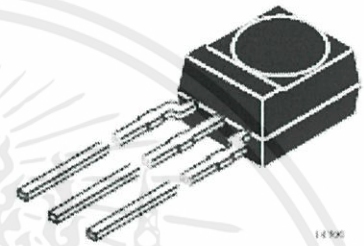
Photo Modules for PCM Remote Control Systems

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

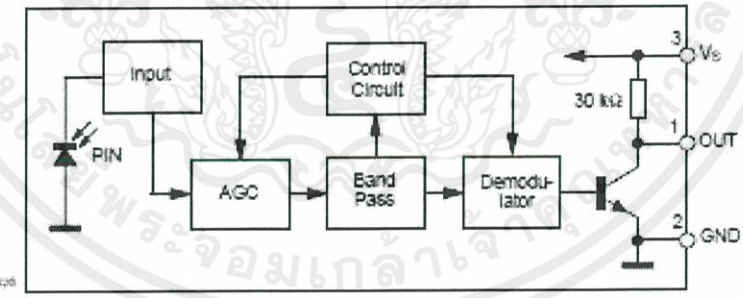
The TSOP48.. – series are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter.
The demodulated output signal can directly be decoded by a microprocessor. TSOP48.. is the standard IR remote control 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 bit/s)
- Suitable burst length ≥10 cycles/burst

Block Diagram



Absolute Maximum Ratings

T_{amb} = 25° C

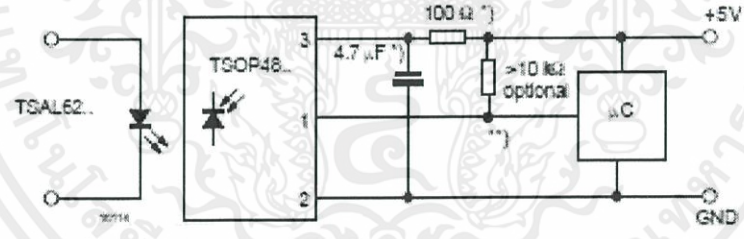
Parameter	Test Conditions	Symbol	Value	Unit
Supply Voltage	(Pin 3)	V_S	–0.3...6.0	V
Supply Current	(Pin 3)	I_S	5	mA
Output Voltage	(Pin 1)	V_O	–0.3...6.0	V
Output Current	(Pin 1)	I_O	5	mA
Junction Temperature		T_j	100	°C
Storage Temperature Range		T_{stg}	–25...+85	°C
Operating Temperature Range		T_{amb}	–25...+85	°C
Power Consumption	(T_{amb} ≤ 85 °C)	P_{tot}	50	mW
Soldering Temperature	t ≤ 10 s, 1 mm from case	T_{sd}	260	°C

Basic Characteristics

T_{amb} = 25° C

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Supply Current (Pin 3)	V_S = 5 V, E_v = 0	I_{SD}	0.6	1.1	1.5	mA
	V_S = 5 V, E_v = 40 klx, sunlight	I_{SH}		1.4		mA
Supply Voltage (Pin 3)		V_S	4.5		5.5	V
Transmission Distance	E_v = 0, test signal see fig.7, IR diode TSAL6200, I_F = 250 mA	d		35		m
Output Voltage Low (Pin 1)	I_{OSL} = 0.5 mA,E_e = 0.7 mW/m²	V_{OSL}			250	mV
Irradiance (30 – 40 kHz)	Pulse width tolerance: t_{po} – 5/f_o < t_{po} < t_{pi} + 6/f_o, test signal see fig.7	$E_{e\,min}$		0.2	0.4	mW/m²
Irradiance (56 kHz)	Pulse width tolerance: t_{pi} – 5/f_o < t_{po} < t_{pi} + 6/f_o, test signal see fig.7	$E_{e\,min}$		0.3	0.6	mW/m²
Irradiance	t_{pi} – 5/f_o < t_{po} < t_{pi} + 6/f_o	$E_{e\,max}$	30			W/m²
Directivity	Angle of half transmission distance	$\varphi_{1/2}$		±45		deg

Application Circuit



*) recommended to suppress power supply disturbances
**) The output voltage should not be held continuously at a voltage below 3.3V by the external circuit.

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 Micom 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 occure.

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).

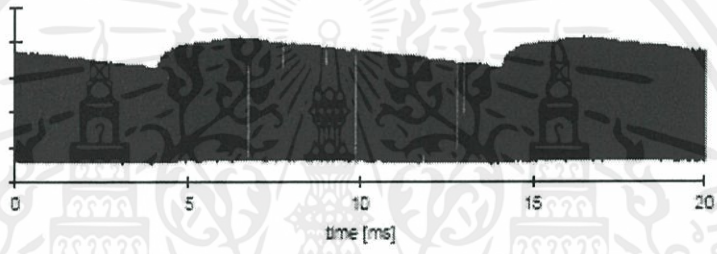


Figure A: IR Signal from Fluorescent Lamp with low Modulation

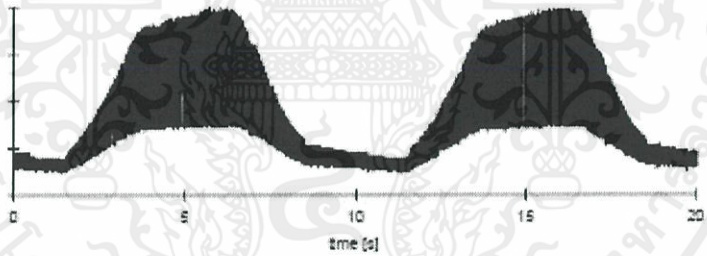


Figure B: IR Signal from Fluorescent Lamp with high Modulation

Typical Characteristics (T_{amb} = 25°C unless otherwise specified)

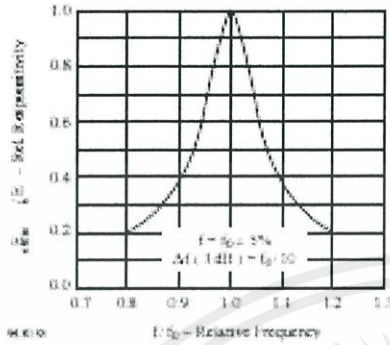


Figure 1. Frequency Dependence of Responsivity

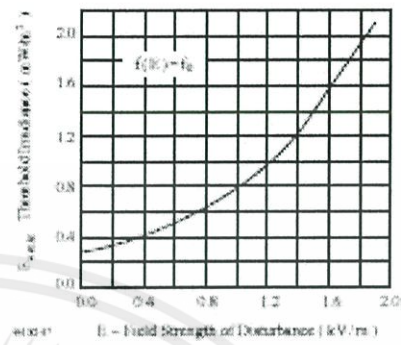


Figure 4. Sensitivity vs. Electric Field Disturbances

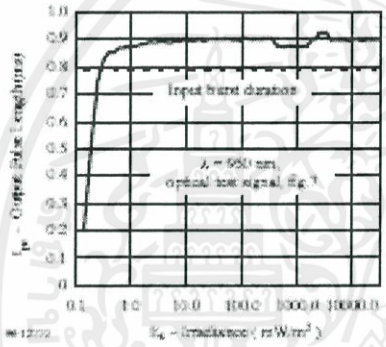


Figure 2. Sensitivity in Dark Ambient

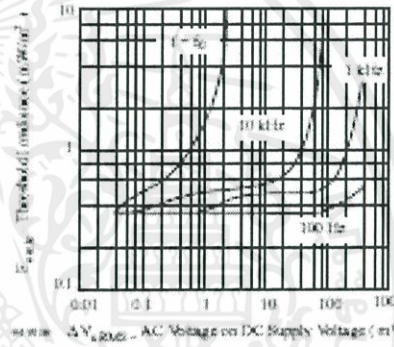


Figure 5. Sensitivity vs. Supply Voltage Disturbances

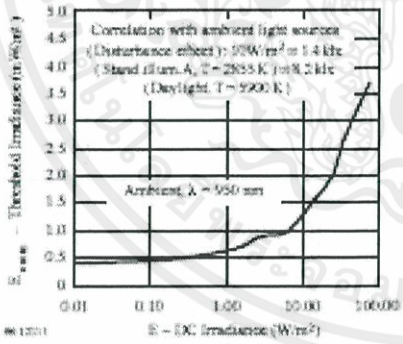


Figure 3. Sensitivity in Bright Ambient

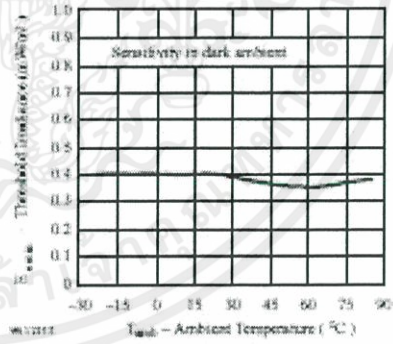
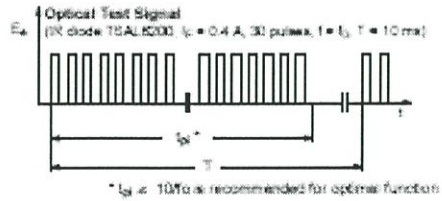


Figure 6. Sensitivity vs. Ambient Temperature



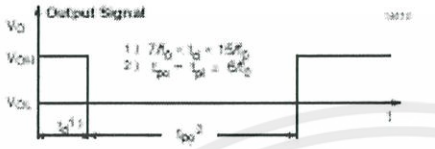


Figure 7.

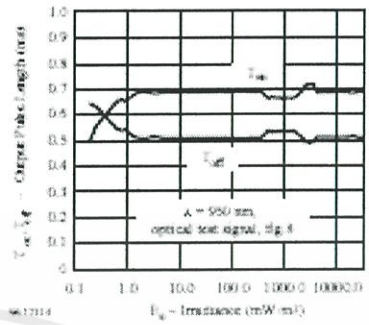


Figure 10. Output Pulse Diagram

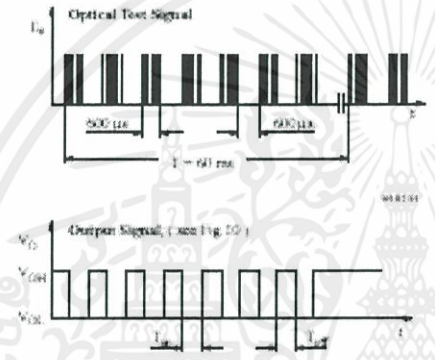


Figure 8. Output Function

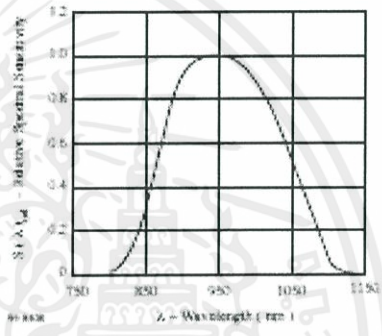


Figure 11. Relative Spectral Sensitivity vs. Wavelength

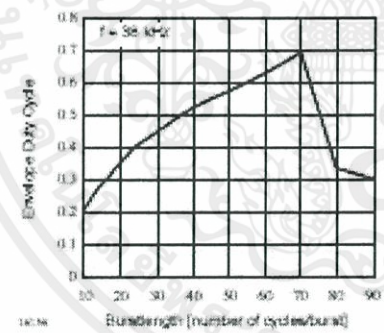


Figure 9. Max. Envelope Duty Cycle vs. Burstlength

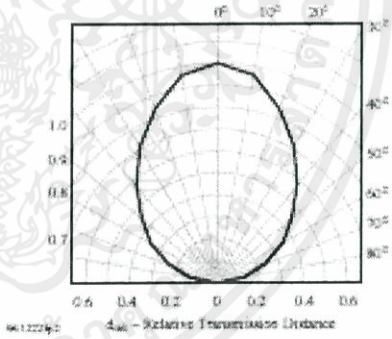
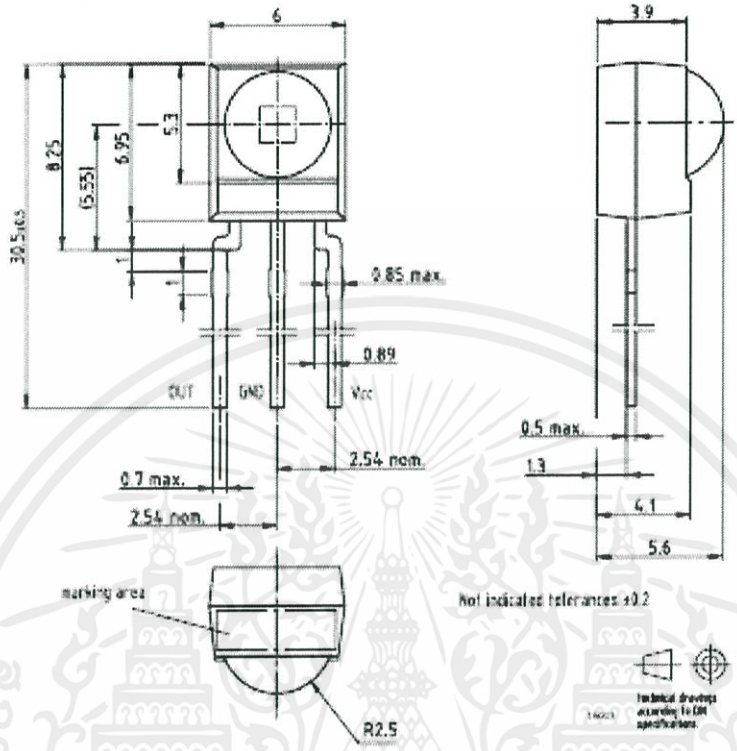


Figure 12. Directivity

Dimensions in mm



ข.3 ตัวส่งสัญญาณอินฟราเรด

TAIWAN OASIS LED DATA SHEET (FOR INFRARED)

PART NO. : TOIR-50b94bCEa

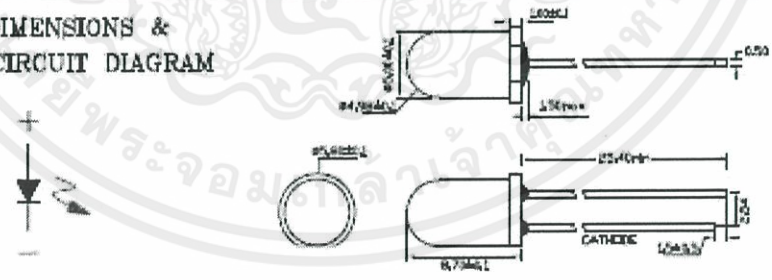
ABSOLUTE MAXIMUM RATINGS AT TA=25°C

PARAMETER	SYMBOL	DATA	UNIT
Forward Current	I_{FM}	100	mA
Peak Forward Current (duty=1:100, f=100KHz)	I_{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 Sold 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	$P_{O(typ.)}$	12.0	mW	Distance: 10cm I_F=50mA Detector Area: 1cm²
Forward Voltage	V_F	TYP: 1.25 MAX: 1.45	V	I_F=20mA
Wavelength	λP	940	nm	I_F=20mA
Spectrum Width of Half Value	$\triangle\lambda$	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	±0.25 ANGLE ±1°	DRAWN		CHECKED	
UNIT	M/M	SHEET NO.	1/2	DRAWING NO.	S-50b94bCEa-A	CUSTOMER		APPROVED	