


การควบคุมอุปกรณ์ส่องสว่างผ่านเทคโนโลยีแบบไร้สาย  
LIGHT BULB CONTROL VIA WIRELESS TECHNOLOGY



กฤติกา ชินรัตน์  
ณัฐพงษ์ สุขประเสริฐ  
สถิตย์พงศ์ ชนะโรจน์เจริญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาดตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต  
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์  
คณะวิศวกรรมศาสตร์  
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง  
ปีการศึกษา 2555

การควบคุมอุปกรณ์ส่องสว่างผ่านเทคโนโลยีแบบไร้สาย  
LIGHT BULB CONTROL VIA WIRELESS TECHNOLOGY



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต  
สาขาวิศวกรรมอัตโนมัติ  
คณะวิศวกรรมศาสตร์

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาระหว่างปีการศึกษา 2555 ถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

# LIGHT BULB CONTROL VIA WIRELESS TECHNOLOGY



KRITTIKA CHINRUT  
NATTAPONG SUKPRASERT  
SATIDPONG CHANAROTCHAROEN

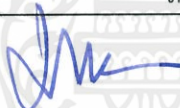

A THESIS SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENT FOR THE DEGREE OF  
BECHELOR OF ENGINEERING IN AUTOMATION ENGINEERING  
FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG  
ACADEMIC YEAR 2012

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญาานิพนธ์ปีการศึกษา 2555  
คณะวิศวกรรมศาสตร์  
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง  
ใบรับรองปริญญาานิพนธ์

หัวข้อปริญญาานิพนธ์ การควบคุมอุปกรณ์ส่องสว่างผ่านเทคโนโลยีแบบไร้สาย  
LIGHT BULB CONTROL VIA WIRELESS TECHNOLOGY  
นักศึกษาผู้จัดทำ นางสาวกฤติกา ชินรัตน์ รหัสนักศึกษา 52010024  
นายณัฐพงษ์ สุขประเสริฐ รหัสนักศึกษา 52010336  
นายสถิตพงศ์ ชนะโรจน์เจริญ รหัสนักศึกษา 52011236  
ปริญญา วิศวกรรมศาสตรบัณฑิต  
สาขาวิชา วิศวกรรมอัตโนมัติ  
ปีการศึกษา 2555

อาจารย์ผู้ควบคุมปริญญาานิพนธ์	ลายมือชื่อ
รศ.ประภาฯ อุดคภูมิพันธุ์	
อาจารย์กฤษณ์ เสมอพิทักษ์	

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์      การควบคุมอุปกรณ์ส่องสว่างผ่านเทคโนโลยีแบบไร้สาย  
LIGHT BULB CONTROL VIA WIRELESS TECHNOLOGY  
นักศึกษาผู้จัดทำ      นางสาวกฤติกา ชินรัตน์      รหัสนักศึกษา 52010024  
                                 นายณัฐพงษ์ สุขประเสริฐ      รหัสนักศึกษา 52010336  
                                 นายสถิตย์พงศ์ ชนะโรจน์เจริญ      รหัสนักศึกษา 52011236  
อาจารย์ที่ปรึกษา      รศ.ประภาฯ อุดคคิมาพันธุ์  
                                 อาจารย์กฤษณ์ เสมอพิทักษ์  
ปีการศึกษา      2555

### บทคัดย่อ

ในโครงการนี้เป็นการออกแบบและสร้างอุปกรณ์ซึ่งอำนวยความสะดวกในการจัดการระบบแสงสว่างสำหรับที่พักอาศัยนอกจากนั้นยังสามารถประยุกต์ใช้ในโรงงานอุตสาหกรรม รวมไปถึงสถานที่ต้องการจัดการระบบแสงสว่างผ่านระบบสารสนเทศ โดยใช้เทคโนโลยีแบบไร้สายทำงานร่วมกับไมโครคอนโทรลเลอร์ซึ่งอยู่ภายในชิปเดียวกัน ทำให้สามารถสร้างเครือข่ายของระบบประมวลผล ซึ่งอุปกรณ์ได้ออกแบบมาให้ติดตั้งเข้ากับระบบแสงสว่างแบบเดิม และลดค่าใช้จ่ายในส่วนฮาร์ดแวร์ที่มีความเป็นไปได้เมื่อนำมาประยุกต์ใช้จริง และยังเป็นแนวทางสำหรับผู้สนใจ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title	LIGHT BULB CONTROL VIA WIRELESS TECHNOLOGY	
Authors	Ms. Krittika	Chinrut
	Mr. Nattapong	Sukprasert
	Mr. Satidpong	Chanarotcharoen
Thesis Advisor	Assoc.Prof. Prapart	Ukakitpaparn
	Mr. Krit	Smerpitak
Year	2012	

## ABSTRACT

This project concerns about the education of equipment decoration. It is able to be convenience tools and lighting management system for the users. Although the equipment are able to apply with the indoor lighting system such as home, warehouse, industrial factory along with the place where would like to use tool effectively. According to the users needs and achieve the user goals too. Using wireless technology to work. This series can communicate via wireless technology. Besides the equipment designed for work together with normally lighting system. Final this project could be approach for interest to interpolating people.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี คณะผู้จัดทำขอกราบขอบพระคุณอาจารย์ที่ปรึกษา รศ.ประภาช อุคคกิมพันธ์ และ อาจารย์ฤกษ์ณี เสมอพิทักษ์ ที่ได้ให้คำปรึกษา ชี้แนะแนวทางในการ แก้ปัญหาและให้ความดูแลเอาใจใส่ตลอดการทำงานทั้งหมด

ขอขอบพระคุณ รศ.ประภาช อุคคกิมพันธ์ ที่ได้ให้คำปรึกษา วิธีแก้ไขปัญหา ถ่ายทอด ประสบการณ์ในการทำงาน ให้แนวทางความคิดริเริ่มสร้างสรรค์ และทฤษฎีต่างๆที่คณะผู้จัดทำไม่เคยได้ เรียนรู้และไม่เคยปฏิบัติจริงมาก่อน ตลอดจนให้การดูแลเอาใจใส่แก่คณะผู้จัดทำและยังสอบถาม ความก้าวหน้าของโครงการอย่างสม่ำเสมอ

ขอขอบพระคุณ อาจารย์ฤกษ์ณี เสมอพิทักษ์ ที่ได้เอื้อเฟื้ออุปกรณ์และสถานที่ตลอดการทำ โครงการนี้ที่เอื้อเฟื้ออุปกรณ์ในการทดลอง ให้คำแนะนำและช่วยแก้ไขปัญหิต่าง ๆ ที่เกิดขึ้นระหว่างการทำโครงการรวมถึงให้คำแนะนำในการทำปริญญานิพนธ์ฉบับนี้

ขอขอบพระคุณคณาจารย์สาขาวิชาวิศวกรรมอัตโนมัติและวิศวกรรมการวัดและควบคุมทุก ท่าน ที่ได้ให้คำแนะนำและการช่วยเหลืออันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

ขอขอบคุณเพื่อน ๆ ทุกคนที่ให้กำลังใจ สอบถามความคืบหน้าของโครงการรวมถึงปริญญานิพนธ์ ฉบับนี้ และให้คำปรึกษาต่าง ๆ ซึ่งเปรียบเสมือนแรงกระตุ้นเป็นอย่างดี จนสามารถทำให้โครงการและ ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบคุณบุคคลที่คอยให้คำปรึกษาและคำแนะนำถึงสิ่งที่ไม่ถนัดและยังไม่เข้าใจ ที่เกี่ยวข้องกับ โครงการ และช่วยแก้ไขปัญหิต่าง ๆ ที่เกิดขึ้นระหว่างการทำโครงการด้วย

และสุดท้าย ขอกราบขอบพระคุณ คุณพ่อและคุณแม่อันเป็นที่รักยิ่ง ซึ่งคอยเป็นแรงสนับสนุน เป็นแรงบันดาลใจรวมถึงแรงกระตุ้นในความสำเร็จและความก้าวหน้าของบุตร ทั้งคุณค่าและประโยชน์ ของโครงการในครั้งนี้ คณะผู้จัดทำขอมอบให้แด่ผู้มีพระคุณทุกท่านเป็นอย่างยิ่งที่ทำให้โครงการนี้สำเร็จ

คณะผู้จัดทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

# สารบัญ

หน้า

บทคัดย่อภาษาไทย.....	I
บทคัดย่อภาษาอังกฤษ.....	II
กิตติกรรมประกาศ.....	III
สารบัญ.....	IV
สารบัญรูป.....	VII

บทที่ 1 บทนำ.....	1
1.1 ที่มาและความสำคัญของโครงงาน.....	1
1.2 วัตถุประสงค์ของปริญญานิพนธ์.....	1
1.3 ขอบเขตของโครงงาน.....	2
1.4 ขั้นตอนการดำเนินงาน.....	2
1.5 ประโยชน์ที่คาดว่าจะได้รับ.....	2
1.6 รายละเอียดปริญญานิพนธ์.....	2
บทที่ 2 ทฤษฎีและหลักการ.....	4
2.1 กล่าวนำ.....	4
2.2 ไมโครคอนโทรลเลอร์ nRF24le1.....	4
2.2.1 คุณสมบัติไมโครคอนโทรลเลอร์.....	4
2.2.2 โครงร่างของพอร์ต nRF24le1.....	5
2.2.3 ระบบเครือข่ายไร้สาย (Wireless LAN).....	6
2.2.4 มาตรฐาน Shock Burst Protocol.....	7
2.3 แลตซิ่งรีเลย์.....	8
2.4 ทฤษฎีกำลังไฟฟ้า.....	9
2.5 หลักการวัดพลังงาน.....	10
2.5.1 Hall effect current sensor.....	10
2.5.2 Power IC.....	12
2.5.3 โวลต์เตจดีไวเดอร์.....	14
2.6 หลักการที่ใช้ในแหล่งจ่ายพลังงาน.....	15
2.6.1 หม้อแปลง.....	15
2.6.2 วงจรเรียงกระแสเต็มคลื่นแบบบริดจ์ (Bridge Rectifier).....	16

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อใช้ในการเรียนการสอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ในการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น

## สารบัญ (ต่อ)

2.6.3	โวลท์เรกกูเลเตอร์ไอซี .....	17
2.7	การสื่อสาร RS-232.....	17
2.8	หลักการออกแบบเว็บไซต์ .....	19
2.8.1	การออกแบบ .....	20
2.8.2	ระบบฐานข้อมูล.....	20
2.8.3	ระบบการจัดการฐานข้อมูล.....	22
บทที่ 3 การออกแบบและการทำงาน .....		23
3.1	กล่าวนำ.....	23
3.2	แผนภาพการวางแผนโครงสร้างของโครงการ .....	23
3.3	การออกแบบโมดูลควบคุมอุปกรณ์ส่องสว่าง .....	23
3.3.1	โมดูลส่วนสั่งการเปิด- ปิดหลอดไฟ.....	23
3.3.1.1	โวลท์เตจดีไวเดอร์ .....	23
3.3.1.2	เคอร์เรนท์เซนเซอร์.....	24
3.3.1.3	พาวเวอร์ไอซี.....	25
3.3.1.4	วงจรถับรีเลย์ .....	25
3.3.1.5	ไมโครคอนโทรลเลอร์ nRF24le1.....	26
3.3.1.6	วงจรรวม.....	26
3.3.1.7	การออกแบบแผ่นลายวงจร.....	28
3.3.1.8	สวิตช์ไมโครคอนโทรลเลอร์ nRF24le1.....	29
3.3.2	ส่วนการสื่อสารผ่านระบบอินเทอร์เน็ต.....	31
3.3.2.1	มาสเตอร์ไมโครคอนโทรลเลอร์ nRF24le1 .....	31
3.3.2.2	บอร์ดเชื่อมต่อ UART.....	32
3.4	การออกแบบการทำงานของไมโครคอนโทรลเลอร์ .....	32
3.4.1	การทำงานของมาสเตอร์ไมโครคอนโทรลเลอร์.....	32
3.4.2	การทำงานของสเลฟไมโครคอนโทรลเลอร์ .....	33
3.5	การออกแบบเว็บเพจและโปรแกรม.....	34
3.5.1	การออกแบบหน้าเว็บเพจ .....	35
3.5.2	การออกแบบโปรแกรม .....	36

เอกสารนี้เป็นเอกสารที่ 3.5.1 การออกแบบหน้าเว็บเพจ .....

ไม่ว่ากรณีใดๆทั้งสิ้น 3.5.2 การออกแบบโปรแกรม .....

## สารบัญ (ต่อ)

บทที่ 4 การทดสอบการทำงาน .....	38
4.1 กล่าวนำ .....	38
4.2 การทดลองการควบคุมหลอดไฟผ่านการสั่งงานจากคอมพิวเตอร์ .....	38
4.3 การทดลองการควบคุมหลอดไฟผ่านการสั่งงานบอร์ดสวีตช์ .....	40
4.4 การทดลองการควบคุมหลอดไฟผ่านการสั่งงานจากเว็บเบราว์เซอร์.....	41
บทที่ 5 สรุปผลและข้อเสนอแนะ.....	42
5.1 สรุปผล .....	42
5.2 ปัญหาและอุปสรรค .....	42
5.3 ข้อเสนอแนะ .....	42
บรรณานุกรม.....	43
ภาคผนวก.....	44

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

# สารบัญรูป

รูปที่	หน้า
2.1 โครงร่างของไมโครคอนโทรลเลอร์ nRF24le1.....	5
2.2 วงจรการเชื่อมต่ออุปกรณ์ของไมโครคอนโทรลเลอร์ nRF24le1 .....	6
2.3 บอร์ดไมโครคอนโทรลเลอร์ nRF24le1.....	6
2.4 ตัวอย่างรูปแบบแพ็คเกจของ Shock Burst Protocol .....	8
2.5 รูปแบบ Packet Control Field .....	8
2.6 โครงร่างของแลตซิงรเลย์.....	9
2.7 ความสัมพันธ์ระหว่างกำลังไฟฟ้า กระแสไฟฟ้าและแรงดันไฟฟ้า.....	9
2.8 ความสัมพันธ์ระหว่างกำลังไฟฟ้าจริง และกำลังไฟฟ้ารีแอกทีฟ.....	10
2.9 เคอร์เรนท์เซนเซอร์แสดง Hall Effect .....	11
2.10 ACS712.....	12
2.11 ตัวอย่างการต่อใช้งานไอซี ACS712 .....	12
2.12 ไอซี ADE7763 .....	12
2.13 โครงร่าง ADE7763.....	13
2.14 ไตอะแกรมของไอซี ADE7763.....	14
2.15 โวลท์เตจดีไวเดอร์ .....	15
2.16 หม้อแปลงแบบ 2 ขดลวด .....	15
2.17 วงจรเรียงกระแสเต็มคลื่นแบบบริดจ์.....	16
2.18 รูปคลื่นแรงดันขาออกเปรียบเทียบกับแรงดันขาเข้า .....	16
2.19 แสดงค่าแรงดันไฟตรงเฉลี่ยกับค่าแรงดันไฟสูงสุด .....	16
2.20 โวลท์เตจเรกกูเลเตอร์ไอซีเบอร์ LM7805 .....	17
2.21 วงจรโวลท์เตจเรกกูเลเตอร์ไอซีเบอร์ LM7805 .....	17
2.22 สายในการสื่อสารแบบอนุกรม .....	18
2.23 ระดับสัญญาณ RS-232.....	18
2.24 สัญญาณการสื่อสารแบบซิงโครนัส .....	19
2.25 สัญญาณการสื่อสารแบบอะซิงโครนัส.....	19
2.26 โปรแกรม Dreamweaver CS4 .....	20
3.1 แผนภาพรวมโครงงาน.....	23
3.2 วงจรโวลท์เตจดีไวเดอร์ .....	24
3.3 เคอร์เรนท์เซนเซอร์ในวงจร .....	24

## สารบัญญรูป (ต่อ)

รูปที่	หน้า
3.4 พาวเวอร์ไอซีในวงจร.....	25
3.5 วงจรขับรีเลย์ที่รับคำสั่งจากไมโครคอนโทรลเลอร์.....	25
3.6 ไมโครคอนโทรลเลอร์ nRF1e1 ในวงจร.....	26
3.7 วงจรโดยรวม.....	27
3.8 วงจรที่ออกแบบ.....	28
3.9 วงจรที่ใช้งาน.....	29
3.10 วงจรโดยรวมของสวิตช์ไมโครคอนโทรลเลอร์.....	30
3.11 ลายวงจรบอร์ดสวิตช์ไมโครคอนโทรลเลอร์.....	31
3.12 วงจรบอร์ดสวิตช์ไมโครคอนโทรลเลอร์ที่ใช้งาน.....	31
3.13 มาสเตอร์ไมโครคอนโทรลเลอร์ nRF24le1.....	32
3.14 บอร์ด UART.....	32
3.15 ขั้นตอนการทำงานของมาสเตอร์ไมโครคอนโทรลเลอร์.....	33
3.16 ขั้นตอนการทำงานของสเลฟไมโครคอนโทรลเลอร์.....	34
3.17 แสดงภาพรวมของหน้าเว็บไซต์และข้อมูลที่วัดค่าออกมา.....	36
3.18 แผนภาพขั้นตอนการทำงานของโปรแกรม.....	37
4.1 การสั่งงานไมโครคอนโทรลเลอร์จากคอมพิวเตอร์.....	38
4.2 บอร์ดวงที่ใช้ในการควบคุมการเปิด-ปิดหลอดไฟ.....	39
4.3 หน้าต่างโปรแกรม Hyper Terminal.....	39
4.4 บอร์ดวงจรสวิตช์ที่ใช้สั่งการหลอดไฟ แบบไร้สาย.....	40
4.5 อุปกรณ์ที่รับคำสั่งจากคอมพิวเตอร์และเชื่อมต่อกะหลอดไฟ.....	40
4.6 แสดงภาพรวมของหน้าเว็บไซต์และข้อมูลที่วัดค่าออกมา.....	41

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

# บทที่ 1

## บทนำ

### 1.1 ที่มาและความสำคัญของโครงการ

ถ้าพูดถึงอุปกรณ์ Electronics คงไม่มีใครที่จะปฏิเสธว่าไม่รู้จักอุปกรณ์ Electronics เพราะในทุกวันนี้เทคโนโลยีได้เจริญก้าวหน้าไกลและเป็นที่แพร่หลายมาก ซึ่งในชีวิตประจำวันนั้นเต็มไปด้วยเครื่องใช้ Electronics มากมาย ที่เป็นตัวช่วยในการใช้ชีวิตของผู้คนนั้นให้มีความคล่องตัวหรือลดความยุ่งยากให้ และเพราะสิ่งเหล่านี้ที่อุปกรณ์ Electronics มอบให้ จึงทำให้ผู้ใช้ส่วนใหญ่เกิดความเคยชินจนกลายเป็นความละเลยการใช้อุปกรณ์อย่างพอดี และไม่เล็งเห็นถึงคุณค่าของพลังงาน E-house หรือ Electronics House มีวัตถุประสงค์ในการจัดทำ โดยเป็นการนำอุปกรณ์ Electronics มาประยุกต์ใช้งานเพิ่มเติม เพื่อการจัดการด้านพลังงานแสงสว่าง การลดภาระของผู้ใช้งาน และการใช้พลังงานอย่างรู้คุณค่า

ในทุกวันของการใช้ชีวิต เราทุกคนจำเป็นต้องพึ่งแสงสว่าง ไม่ว่าจะเป็นจากธรรมชาติหรือจากการสร้างขึ้นโดยมนุษย์ จึงเป็นเหตุผลที่ไม่สามารถปฏิเสธได้ว่า ระบบแสงสว่างมีความสำคัญต่อการดำรงชีวิตเป็นอย่างมาก และระบบแสงสว่างที่สามารถตอบสนองความต้องการของตัวบุคคลได้นั้น ก็คงเป็นระบบแสงสว่างที่มนุษย์สร้างขึ้นนั่นเอง จึงได้พิจารณาและเห็นว่าถ้าพัฒนาหรือปรับปรุงระบบแสงสว่างให้มีความก้าวหน้ามากกว่าแค่การที่เป็นอยู่ในปัจจุบันนั้น จะเป็นการพัฒนาชีวิตและสิ่งแวดล้อมได้เป็นอย่างดี โดยการเพิ่มคุณสมบัติในการสั่งการ ซึ่งได้นำเทคโนโลยีแบบไร้สายเข้ามาช่วยในการทำงาน ซึ่งทำให้เกิดเป็นระบบเครือข่ายที่มีความสะดวกในการตรวจสอบการทำงาน คุณสมบัติในด้านการแสดงผลการใช้งาน คือมีการตรวจวัดค่าทางไฟฟ้าต่างๆ เช่น ค่าแรงดันไฟฟ้า ค่ากระแสไฟฟ้า เพื่อนำมาเป็นข้อมูลในการตรวจสอบปริมาณการใช้ไฟฟ้าหรือการใช้งานของหลอดไฟ ในด้านคุณสมบัติของระบบสำรองเมื่อเกิดการขัดข้อง จะมีการใช้ในส่วนของแม่คานิคเข้ามาช่วยเหลือผู้ใช้งานในกรณีที่โมดูลเกิดการขัดข้องหรือเสียหายทำให้ระบบแสงสว่างยังคงทำงานได้ตามปกติ หรือคุณสมบัติของด้านระบบความปลอดภัยต่อผู้ใช้งาน ก็จะช่วยเพิ่มความปลอดภัยของชีวิตของผู้ใช้งานหากเกิดการรั่วไหลของกระแสไฟฟ้า โดยในที่สุดแล้วผลที่ได้คาดว่าจะได้รับหลังจากการติดตั้งโมดูลที่มีคุณสมบัติดังกล่าวนี้ จะช่วยให้การใช้ชีวิตในแต่ละวันที่ดีขึ้นในด้านใดด้านหนึ่ง ทั้งการจัดการพลังงานและค่าใช้จ่ายภายในบ้าน ซึ่งจะส่งผลต่อสภาพสิ่งแวดล้อมในทางที่ดีขึ้นได้

### 1.2 วัตถุประสงค์ของปริญญานิพนธ์

เพื่อออกแบบตัวควบคุมการเปิด-ปิด อุปกรณ์ส่องสว่างภายในอาคารสำหรับ E-house ซึ่งตามแนวคิด จะประกอบด้วย 2 ส่วนหลักคือประกอบไปด้วยวงจรที่เรียกว่า วงจรโหลด และวงจรควบคุม ซึ่งวงจรโหลดนั้นหมายถึง วงจรควบคุมการจ่ายกำลังงานไฟฟ้าเปรียบเสมือนสวิตซ์ที่ต่ออนุกรมกับโหลดที่ต่ออนุกรมกับโหลดที่เชื่อมต่อกับแหล่งจ่ายกำลังไฟฟ้า ส่วนวงจรควบคุมนั้นเปรียบเสมือนตัวกำหนดสถานะของสวิตซ์ของวงจรโหลดว่าอยู่ในสถานะปิดวงจรหรือเปิดวงจร และการนำเทคโนโลยีสื่อสารแบบไร้สาย ( wireless ) มาใช้งานร่วมกับอุปกรณ์ส่องสว่าง และใช้ไมโครคอนโทรลเลอร์ในการควบคุมการทำงาน

### 1.3 ขอบเขตของโครงการ

1. สร้างโมดูลสำหรับควบคุมอุปกรณ์ส่องสว่างแบบไร้สาย
2. สร้างโมดูลสำหรับการสั่งการ ( Remote )
- 3 .แสดงผลผ่านเว็บเบราว์เซอร์

### 1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาการทำงานของเทคโนโลยีแบบไร้สาย
2. ศึกษารูปแบบการทำงานของไมโครคอนโทรลเลอร์
3. ศึกษาการออกแบบอุปกรณ์อิเล็กทรอนิกส์
4. รวบรวมข้อมูลทั้งหมด พร้อมศึกษาแนวทางการเป็นไปได้
5. ออกแบบและสร้างโมดูล
6. เขียนโปรแกรม
7. ทดสอบการทำงานร่วมกับโปรแกรม
8. สรุป วิเคราะห์ผลและเสนอแนะ

### 1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประยุกต์ใช้ไมโครคอนโทรลเลอร์ให้สั่งงานอุปกรณ์ส่องสว่างและสามารถลดภาระของผู้ใช้งานในส่วนของการควบคุมระบบไฟฟ้าและเป็นสิ่งอำนวยความสะดวกในชีวิตประจำวัน

### 1.6 รายละเอียดของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ แบ่งออกเป็น 5 บท ซึ่งครอบคลุมเนื้อหาดังนี้

บทที่ 1 บทนำ กล่าวถึงที่มาและความสำคัญ วัตถุประสงค์ ขั้นตอนการศึกษา ประโยชน์ที่ได้รับ และรายละเอียดโดยย่อของปริญญานิพนธ์แต่ละบท

บทที่ 2 หลักการที่เกี่ยวข้องและอุปกรณ์ที่เลือกใช้ กล่าวถึงไมโครคอนโทรลเลอร์ nRF24le1 แลชซีรี่เลย์ ทฤษฎีกำลังไฟฟ้า หลักการวัดพลังงาน หลักการที่ใช้ในแหล่งจ่ายพลังงาน การสื่อสาร RS-232 ตลอดจนหลักการที่เกี่ยวข้องในออกแบบเว็บไซต์ ที่นำมาใช้ในการทำโครงการ

บทที่ 3 การออกแบบการควบคุมอุปกรณ์ส่องสว่าง กล่าวถึงการออกแบบโครงสร้างของการควบคุมอุปกรณ์ส่องสว่าง การออกแบบวงจร การออกแบบการควบคุม และการออกแบบส่วนเชื่อมต่อเว็บเบราว์เซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4 การทดสอบการทำงาน กล่าวถึงการทดสอบการทำงานของการควบคุมอุปกรณ์ส่องสว่างแบบไร้สายในส่วนต่าง ๆ เช่น การทดลองการควบคุมผ่านการสั่งงานจากคอมพิวเตอร์ การทดลองการควบคุมผ่านการสั่งงานจากบอร์ดสวิตช์ไมโครคอนโทรลเลอร์ เป็นต้น

บทที่ 5 บทสรุปและข้อเสนอแนะ กล่าวถึง การสรุปผลของการทำงานปัญหาและอุปสรรค และ ข้อเสนอแนะของปริญญานิพนธ์



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## บทที่ 2

# หลักการที่เกี่ยวข้องและอุปกรณ์ที่เลือกใช้

### 2.1 กล่าวนำ

ทฤษฎีและหลักการที่นำมาใช้เพื่อไปประยุกต์ใช้ในการออกแบบตัวควบคุมการเปิด-ปิด อุปกรณ์ส่องสว่างภายในอาคารแบบไร้สาย ซึ่งเป็นแนวทางในการออกแบบอุปกรณ์ทั้งที่ใช้ในวงจรโหลดและวงจรควบคุม โดยการประยุกต์ใช้ไมโครคอนโทรลเลอร์ nRF24le1 ที่รวมเทคโนโลยีรับส่งสัญญาณคลื่นวิทยุที่รวมอยู่กับไมโครคอนโทรลเลอร์ ในการควบคุมการเปิด-ปิดอุปกรณ์ส่องสว่างผ่านรีเลย์ที่เป็นแบบแลชซึ่งเพื่อประหยัดพลังงาน นอกจากนี้ยังสามารถประมาณการใช้พลังงานส่องสว่างผ่านพาวเวอร์ไอซี โดยข้อมูลการใช้พลังงานสามารถเชื่อมต่อผ่านทาง RS-232 เพื่อใช้ในการจัดเก็บข้อมูล รวมถึงการวิเคราะห์การใช้พลังงานด้วยโปรแกรมฐานข้อมูลแสดงผลผ่านทางเว็บไซต์

### 2.2 ไมโครคอนโทรลเลอร์ nRF24le1

ไมโครคอนโทรลเลอร์ nRF24le1 เป็นผลิตภัณฑ์ของทางบริษัท Nordic semiconductor โดยตัวของผลิตภัณฑ์นั้นจะมีราคาค่อนข้างต่ำแต่มีประสิทธิภาพสูงในตระกูลที่รองรับเทคโนโลยีรับส่งสัญญาณคลื่นวิทยุช่วงความถี่ 2.4GHz (2.4GHz RF transceiver) ร่วมกับไมโครคอนโทรลเลอร์แบบฝังอยู่ในตัว บอร์ด nRF24le1 นั้นเหมาะสำหรับผู้ที่ต้องการใช้งานในลักษณะ Ultra Low Power แบบไร้สายโดยใช้ชิปเพียงตัวเดียว รวมถึงการใช้งานในลักษณะอื่น ๆ เช่น การประมวลผลพลังงาน, หน่วยความจำ, Oscillators พลังงานต่ำ, Real time, ตัวจับเวลา และอื่น ๆ สำหรับรูปแบบการใช้งาน nRF24le1 สามารถใช้งานร่วมกับชุดอุปกรณ์ต่าง ๆ ได้แก่ SPI (Serial Peripheral Interface), 2 wire UART, 6-12 bit ADC และการทำงานแบบ PWM (Pulse width modulation) ได้

#### 2.2.1 คุณสมบัติไมโครคอนโทรลเลอร์

- Fast 8-bit microcontroller:
- หน่วยประมวลผล MCS 51
- หน่วยความจำแฟลช สำหรับเขียนโปรแกรม 16 kB และ RAM 1 kB ในตัวชิป
- สามารถรองรับ Input/ Output ได้หลากหลาย
- GPIO (General Purpose Input/ Output)
- SPI master/SPI slave
- 2-Wire master/ slave
- Full duplex serial port
- PWM
- ADC

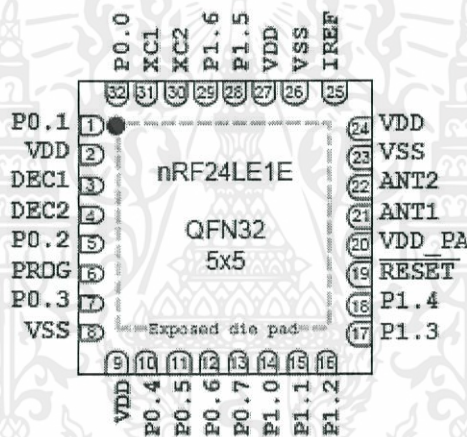
เอกสารนี้เป็นเอกสารที่ส่งมอบในการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งยังต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Analog comparator
- External interrupts
- Timer inputs

- 32.768 kHz crystal oscillator
- Debug interface
- High performance 2.4 GHz RF-transceiver
- GFSK transceiver
- Auto ACK and retransmit
- Data rate 250 kbps, 1 Mbps or 2 Mbps บนช่องทางอากาศ
- Digital interface (SPI) speed 0-8 Mbps
- ตอบสนองความถี่เปลี่ยนแปลงในระยะเวลาสั้น
- A/D converter: 6, 8, 10 or 12 bit resolution
- ตั้งค่าได้เต็มพิสัยโดย internal reference, external reference or VDD
- ใช้กระแสต่ำเพียงแค่ 0.1mA ที่ 2 kbps

### 2.2.2 โครงร่างของพอร์ต nRF24le1



รูปที่ 2.1 โครงร่างของ nRF24le1 ชนิด 32 Pin

ขา VDD หมายถึง Power Supply (+1.9V to +3.6V DC)

ขา VSS หมายถึง กราวด์(0V)

ขา DEC1,DEC2 หมายถึง Power Supply เอาต์พุต

ขา P0.0-P3.6 หมายถึง ขาที่สามารถใช้งานได้

ขา PROG หมายถึง ขาที่ใช้สำหรับโปรแกรม

ขา RESET หมายถึง ขารีเซตของไมโครคอนโทรลเลอร์ ซึ่งเป็นแบบ Active Low

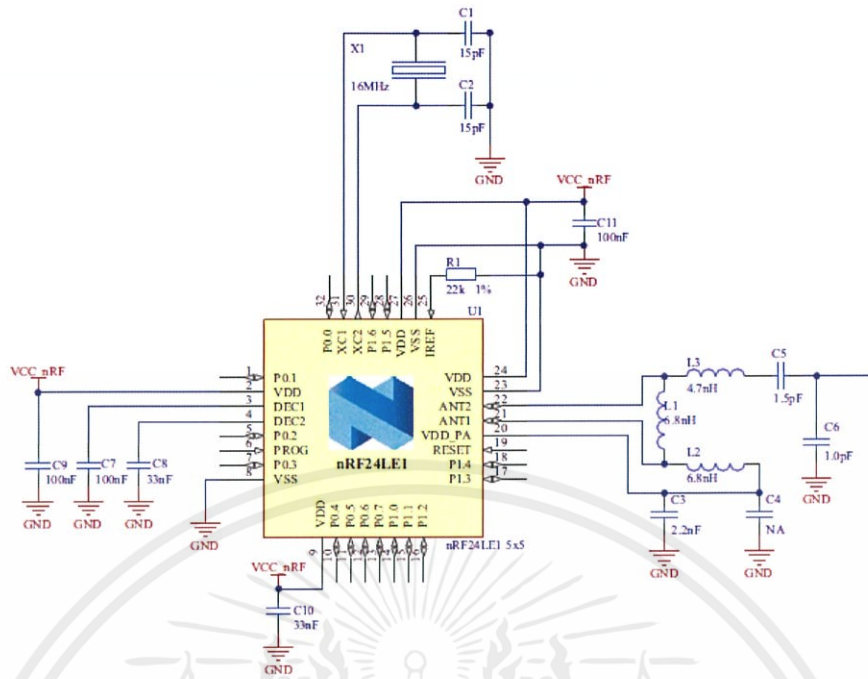
ขา IREF หมายถึง กระแสขาออกที่เป็นตัวอ้างอิง

ขา VDD\_PA หมายถึง Power Supply ขาออก (+1.8V)

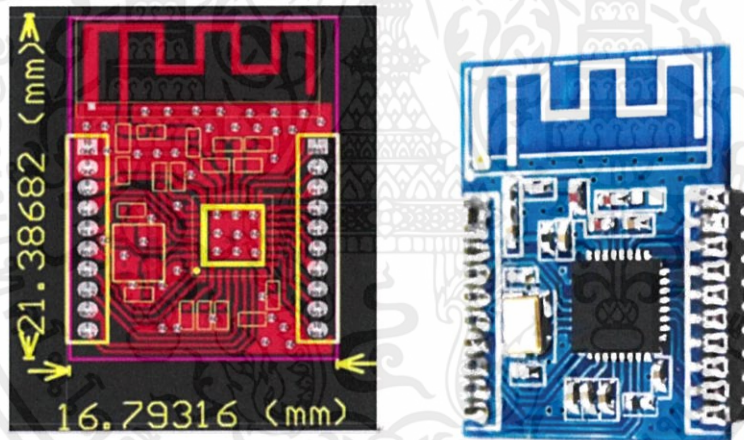
ขา ANT1,ANT2 หมายถึง เชื่อมต่อเสาอากาศอื่นๆ

เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ของ Nordic Semiconductor A/S. การนำเอกสารนี้ไปใช้โดยไม่ได้รับอนุญาตจาก Nordic Semiconductor A/S อาจทำให้เกิดข้อผิดพลาดได้

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.2 วงจรการเชื่อมต่ออุปกรณ์ของไมโครคอนโทรลเลอร์ nRF24le1



รูปที่ 2.3 บอร์ด nRF24LE1

### 2.2.3 ระบบเครือข่ายไร้สาย (Wireless LAN)

ระบบเครือข่ายไร้สาย (WLAN = Wireless Local Area Network) คือ ระบบการสื่อสารข้อมูลที่มีรูปแบบในการสื่อสารแบบไม่ใช้สาย โดยการใช้การส่งคลื่นความถี่วิทยุในย่านวิทยุ RF และ คลื่นอินฟราเรด ในการรับและส่งข้อมูลระหว่างคอมพิวเตอร์แต่ละเครื่อง ผ่านอากาศ, ทะลุ กำแพง, เพดานหรือสิ่งก่อสร้างอื่นๆ โดยปราศจากความต้องการของการเดินสาย นอกจากนี้ระบบเครือข่ายไร้สายก็ยังมีคุณสมบัติครอบคลุมทุกอย่างเหมือนกับระบบ LAN แบบใช้สาย

ที่สำคัญของระบบเครือข่ายไร้สายก็คือ การที่ไม่ต้องใช้สายทำให้การเคลื่อนย้ายการใช้งานทำได้โดยสะดวก ไม่เหมือนระบบ LAN แบบใช้สายที่ต้องใช้เวลาและการลงทุนในการปรับเปลี่ยนตำแหน่งการใช้งานเครื่องคอมพิวเตอร์

## ประโยชน์ของระบบเครือข่ายไร้สาย

1. mobility improves productivity & service มีความคล่องตัวสูง ดังนั้นไม่ว่าจะเคลื่อนที่ไปที่ไหน หรือเคลื่อนย้ายคอมพิวเตอร์ไปตำแหน่งใด ก็ยังมีการเชื่อมต่อกับเครือข่ายตลอดเวลาตราบใดที่ยังอยู่ในระยะการส่งข้อมูล
2. installation speed and simplicity สามารถติดตั้งได้ง่ายและรวดเร็ว เพราะไม่ต้องเสียเวลาติดตั้งสายเคเบิล และไม่รกรุงรัง
3. installation flexibility สามารถขยายระบบเครือข่ายได้ง่าย เพราะเพียงแค่มีพีซีการ์ดมาต่อเข้ากับโน้ตบุ๊ก หรือพีซี ก็เข้าสู่เครือข่ายได้ทันที
4. reduced cost- of-ownership ลดค่าใช้จ่ายโดยรวมที่ผู้ลงทุนต้องลงทุน ซึ่งมีราคาสูง เพราะในระยะยาวแล้วระบบเครือข่ายไร้สายไม่จำเป็นต้องเสียค่าบำรุงรักษาและการขยายเครือข่ายก็ลงทุนน้อยกว่าเดิมหลายเท่า เนื่องด้วยความง่ายในการติดตั้ง
5. scalability เครือข่ายไร้สายทำให้องค์กรสามารถปรับขนาดและความเหมาะสมได้ง่ายไม่ยุ่งยาก เพราะสามารถโยกย้ายตำแหน่งการใช้งานโดยเฉพาะระบบที่มีการเชื่อมระหว่างจุดต่อจุด เช่น ระหว่างตึก

### 2.2.4 มาตรฐาน Shock Burst Protocol

Shock burst Protocol เป็นแพ็คเก็ตข้อมูลในชั้นเชื่อมโยงข้อมูล โดยคุณสมบัติประกอบด้วย แพ็คเก็ตแบบอัตโนมัติ, เวลา, การรับรู้โดยอัตโนมัติ, การส่งสัญญาณใหม่ของแพ็คเก็ต Shock burst Protocol ใช้พลังงานในการใช้งานน้อยแต่มีประสิทธิภาพสูงในการทำงานเชื่อมต่อกับไมโครคอนโทรลเลอร์ และมีคุณสมบัติที่สำคัญช่วยในการปรับปรุงการใช้พลังงานของระบบที่มีสองทิศทางและทิศทางเดียว โดยไม่ต้องเพิ่มความซับซ้อนให้กับด้านตัวควบคุม

#### คุณสมบัติหลัก

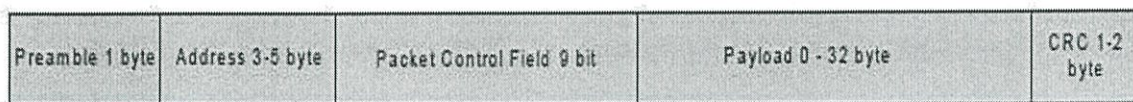
- ขนาดข้อมูลมีขนาดตั้งแต่ 1 – 32 ไบต์ซึ่งสามารถเลือกกำหนดตามความต้องการได้
- สามารถตั้งค่าการจัดการในส่วนของแพ็คเก็ตข้อมูลโดยอัตโนมัติ
- ตอบสนองการรับ-ส่งข้อมูลแบบอัตโนมัติ และจัดส่งข้อมูลใหม่อีกครั้งโดยอัตโนมัติเมื่อมีการสูญหายของข้อมูล
- ติดต่อกันในลักษณะโทโปโลยีแบบสตาร์ (Star Topology) แบบ 1 : 6

#### หลักการจัดการรับ-ส่งข้อมูล

- การส่งแพ็คเก็ตข้อมูลจาก PTX เมื่อ PRX ได้รับข้อมูล จะกำหนด PTX ให้อยู่ในโหมดรับ เพื่อรอให้แพ็คเก็ต ACK ส่งกลับ
- เมื่อได้รับแพ็คเก็ตข้อมูลแล้ว PRX จะประกอบแพ็คเก็ตและส่ง แพ็คเก็ตที่รับมา (ACK แพ็คเก็ต) กลับไปที่ PTX ในโหมดรับ
- ถ้า PTX ไม่ได้รับแพ็คเก็ต ACK ในทันทีหรือภายในระยะเวลาที่กำหนด จะ มีการจัดการส่งข้อมูลชุดเดิมส่งออกไปใหม่ เพื่อเป็นการตรวจสอบการสูญหายของข้อมูล

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี (ตามรูปที่ 2.4) ประกอบไปด้วย โยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.4 ตัวอย่างรูปแบบแพ็คเกจของ Shock Burst Protocol

จากรูปที่ 2.15 เป็นตัวอย่างรูปแบบแพ็คเกจของ Shock Burst Protocol จะประกอบด้วยส่วนต่าง ๆ ดังนี้

- Preamble มีขนาด 1 ไบต์ จะมีการเปลี่ยนแปลงอยู่ตลอดเวลา โดย ขึ้นอยู่กับบิตแรก ถ้าเป็น 1 จะมีการตั้งค่า เริ่มต้นอัตราโน้มนัดเป็น 10101010 แต่ถ้าบิตแรกเป็น 0 จะมีการตั้งค่าเริ่มต้นเป็น 01010101
- Address เปรียบเสมือนขอบเขตที่อยู่ของสัญญาณ เพื่อให้สามารถ รับทราบได้ว่าจะมีการตรวจพบข้อมูลที่ถูส่งออกมาจะมีการกำหนด Address ให้สามารถตอบสนองกันได้ โดยสามารถกำหนดได้เป็น 3 ,4 และ 5 ไบต์ ซึ่งถ้าตัวอุปกรณ์มี Address ที่ไม่เหมือนกัน ก็ไม่สามารถที่ จะรับ - ส่งข้อมูลกันได้
- Packet Control Field ดังรูปที่ 2.16 จะแบ่งออกเป็น 3 ส่วน ดังนี้
  1. Payload length มีขนาด 6 บิต ทำหน้าที่กำหนดขนาดของข้อมูล ได้ 0-32 ไบต์
  2. PID (Packet Identification) มีขนาด 2 บิต ทำหน้าที่ เป็นรีจิสเตอร์ที่กำกับในโหมด PTX และ PRX ในการรับ - ส่งข้อมูล และตรวจสอบข้อมูลสูญหาย
  3. No Acknowledgment flag (NO\_ACK) มีขนาด 1 บิต ทำหน้าที่เป็นรีจิสเตอร์ที่จะบอกถึงการตอบสนองของข้อมูล



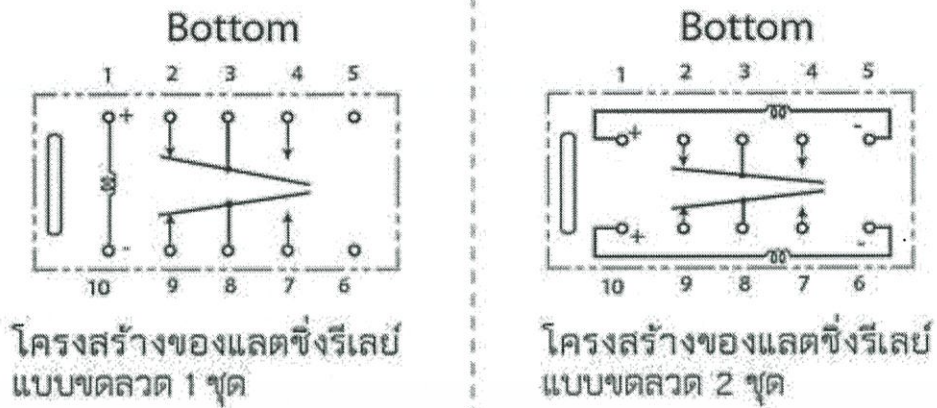
รูปที่ 2.5 รูปแบบ Packet Control Field

### 2.3 แลตชิ่งรีเลย์

แลตชิ่งรีเลย์ (Latching relay) ประกอบด้วยหน้าสัมผัส 2 ชุด และมีขดลวดเหนี่ยวนำหรือโซลินอยด์ (solenoid) 2 แบบคือ แบบขดลวดเหนี่ยวนำ 1 ชุด และขดลวดเหนี่ยวนำ 2 ชุด สำหรับควบคุมหน้าสัมผัสให้ต่อหรือจากกัน

แลตชิ่งรีเลย์แบบขดลวดเหนี่ยวนำ 2 ชุด ควบคุมโดยการจ่ายกระแสไฟฟ้ากระตุ้นที่ขั้วของขดลวดเหมือนกับแบบขดลวด 1 ชุด แต่จะต่างกันตรงที่หากต้องการกลับสถานะของหน้าสัมผัสจะไม่ใช้การสลับขั้วไฟ แต่จะต้องจ่ายกระแสไฟฟ้ากระตุ้นให้กับขดลวดอีกหนึ่งชุดดังรูปที่ 2.6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.6 โครงร่างของแลตซ์รีเลย์

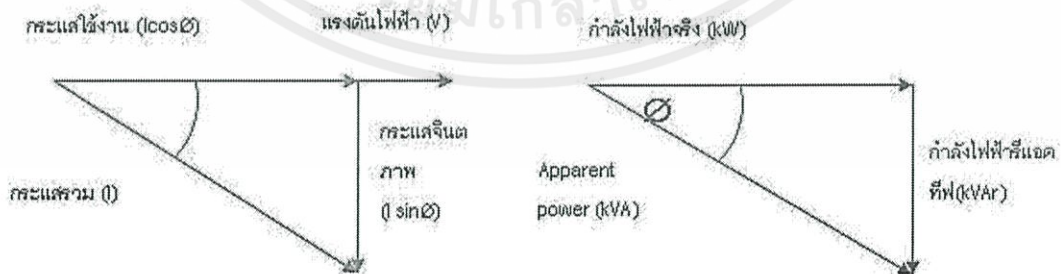
## 2.4 ทฤษฎีกำลังไฟฟ้า

โดยทั่วไปอุปกรณ์ไฟฟ้า ในอาคารหรือโรงงานนั้นต้องอาศัยทั้งกำลังไฟฟ้าจริง (real power) และกำลังไฟฟ้รีแอกทีฟ (reactive power) เพื่อใช้ในการทำงาน ค่าสัดส่วนของกำลังไฟฟ้าทั้งสองชนิดดังกล่าวบ่งบอกถึงค่าตัวประกอบกำลังไฟฟ้า (Power Factor :PF) ของอุปกรณ์ไฟฟ้าแต่ละชนิดหรือของอาคารหรือโรงงานโดยรวม

การจ่ายกำลังไฟฟ้าตามปกติประกอบด้วย 2 ส่วนหลัก คือ

1. กำลังไฟฟ้าจริง (real power) มีหน่วยเป็น (กิโลวัตต์) หรืออาจเรียกเป็นกำลังไฟหรือพลังไฟฟ้าใช้งานก็ได้ เป็นกำลังไฟฟ้าที่ใช้ประโยชน์ในการทำงานของเครื่องจักร เช่น งานที่ได้จากมอเตอร์หรือจากแสงสว่าง เป็นต้น
2. กำลังไฟฟ้รีแอกทีฟ (reactive power) มีหน่วยเป็น กิโลวา (kVAR) เป็นกำลังไฟฟ้าที่จ่ายเพื่อสร้างสนามแม่เหล็กให้แก่อุปกรณ์และเครื่องใช้ทางไฟฟ้า เช่น สนามแม่เหล็กในมอเตอร์ บัลลาสต์ของหลอดไฟแสงสว่าง เป็นต้น

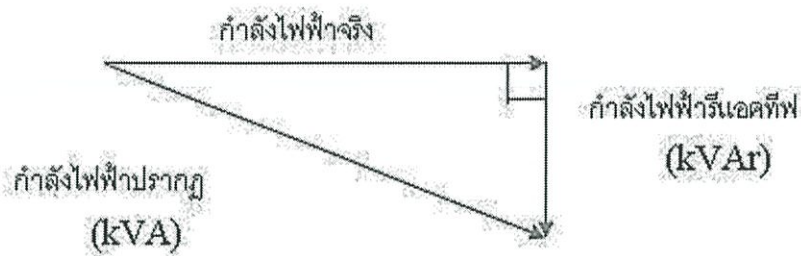
ทั้งนี้ค่ากำลังไฟฟ้าทั้งสองส่วนดังกล่าวจะมีทิศทางตั้งฉากซึ่งกันและกัน สามารถนำมาเขียนแสดงความสัมพันธ์ได้เป็นรูปสามเหลี่ยมมุมฉาก หรือ phasor diagram ดังแสดงในรูปที่ 2.7 ส่วนความสัมพันธ์ระหว่างกำลังไฟฟ้า กระแสไฟฟ้าและแรงดันไฟฟ้าสามารถแสดงได้ ดังรูปที่ 2.8



∠ คือ มุมระหว่างแรงดันและกระแสไฟฟ้า ส่วน I คือ ขนาดกระแสรวม

รูปที่ 2.7 ความสัมพันธ์ระหว่าง กำลังไฟฟ้า กระแสไฟฟ้า และแรงดันไฟฟ้า

เอกสารนี้เป็นเอกสารที่... ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.8 ความสัมพันธ์ระหว่างกำลังไฟฟ้าจริง และกำลังไฟฟ้ารีแอกทีฟ

## 2.5 หลักการวัดพลังงาน

### 2.5.1 Hall effect current sensor

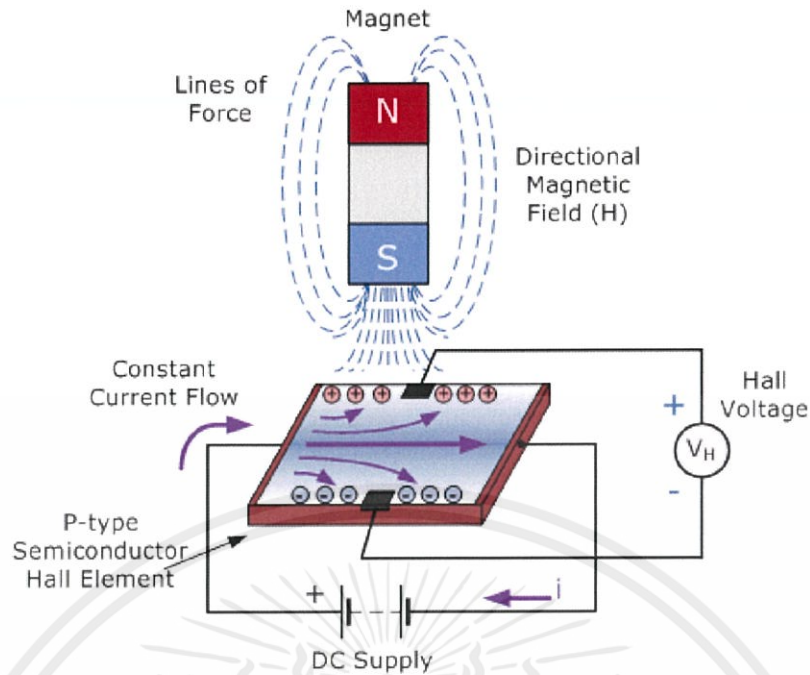
ฮอลล์เอฟเฟกต์เคอร์เรนท์เซนเซอร์ (Hall effect current sensor) สามารถวัดค่ากระแสโดยการวัดคλώงสายของกระแสที่ไหล และให้อาต์พุตออกมาเป็นแรงดัน ฮอลล์เอฟเฟกต์เคอร์เรนท์เซนเซอร์ มี 2 รุ่นคือรุ่นที่ให้อาต์พุตเป็นแบบดิจิตอลคือ 0 กับ 1 รุ่นที่ให้อาต์พุตเป็นแบบอนาล็อก แต่ ฮอลล์เอฟเฟกต์เคอร์เรนท์เซนเซอร์ ส่วนใหญ่จะใช้เป็นเครื่องมือวัดสำหรับความแม่นยำสูง สามารถวัดสัญญาณที่มี DC และ Harmonics ปะปนมากได้ หรือกระแสมีความซับซ้อนของสัญญาณปะปนสูง เหมาะสำหรับการต้องวัดเพื่อวิเคราะห์หา Harmonics ต่าง ๆ

#### ปรากฏการณ์ฮอลล์ (Hall Effect) หรือฮอลล์เอฟเฟกต์

ปรากฏการณ์ฮอลล์ (Hall Effect) หรือฮอลล์เอฟเฟกต์ เป็นปรากฏการณ์ทางไฟฟ้าที่ค้นพบโดย เอ็ดวิน ฮอลล์ (Edwin Hall) ในปี ค.ศ. 1879 สิ่งที่ได้ค้นพบมีหลักการโดยสรุปดังนี้

แผ่นตัวนำที่มีกระแสไหลผ่านเมื่อมีฟลักซ์แม่เหล็ก (Magnetic Flux) มากกระทำในทิศทางตั้งฉากกับแผ่นตัวนำ จะทำให้เกิดสนามไฟฟ้าหรือแรงดันเรียกว่าแรงดันฮอลล์ (Hall Voltage) ขึ้นที่ตัวนำในทิศทางตั้งฉากกับกระแสและฟลักซ์แม่เหล็ก เมื่อจ่ายกระแสคงที่ให้แผ่นตัวนำจะทำให้กระแสไหลผ่านแผ่นตัวนำอย่างคงที่ โดยอิเล็กตรอนจะเคลื่อนที่จากขั้วลบไปขั้วบวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.9 เคอร์เรนท์เซนเซอร์แสดง Hall Effect

เมื่อมีฟลักซ์แม่เหล็กมากระทำกับแผ่นตัวนำในทิศทางตั้งฉากจะทำให้ประจุพาหะ (Charge Carrier) ของตัวนำเบี่ยงเบนไปด้านบนของตัวนำ จากรูปที่ 2.9 ประจุพาหะเป็นอิเล็กตรอนมีประจุเป็นประจุลบทำให้ด้านบนของแผ่นตัวนำมีขั้วไฟฟ้าเป็นลบ ส่วนด้านล่างของแผ่นตัวนำจะมีขั้วตรงข้ามกับด้านบนนั่นคือมีประจุบวก เมื่อวัดความต่างศักย์ระหว่างด้านบนกับด้านล่างทำให้ได้แรงดันไฟฟ้าออกมาเป็นแรงดันลบ โดยขนาดของแรงดันที่วัดได้จะขึ้นอยู่กับความหนาแน่นของฟลักซ์แม่เหล็กที่มากระทำ หากความเข้มสนามแม่เหล็กมากก็จะทำให้เกิดแรงดันมาก และถ้าความเข้มสนามแม่เหล็กน้อย แรงดันก็จะน้อยตามไปด้วย

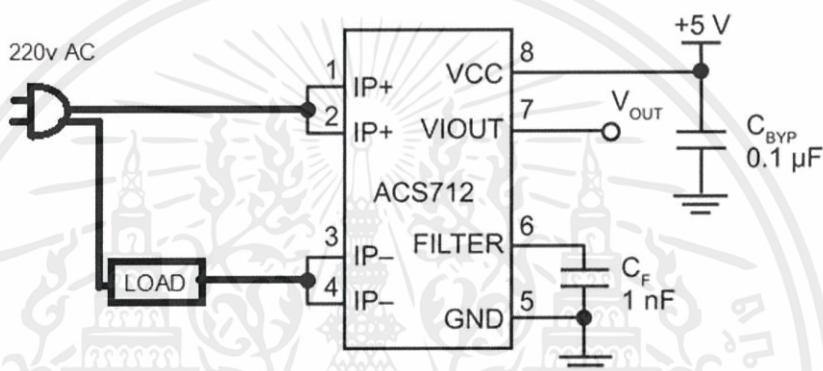
ส่วนกรณีที่มีการกลับขั้วแม่เหล็กจะทำให้แรงดันเอาต์พุตกลับขั้วกับกรณีทีกล่าวมา ตัวนำที่มีประจุพาหะเป็นอิเล็กตรอนได้แก่ ตัวนำไฟฟ้าทั่วไป สารกึ่งตัวนำชนิดเอ็น (N-Type) ส่วนตัวนำที่มีประจุพาหะเป็นประจุบวกได้แก่ สารกึ่งตัวนำชนิดพี (P-Type) ปัจจุบันฮอลล์เอฟเฟกต์จะอยู่ในรูปของวงจรรวมหรือ IC (Integrated Circuit) ที่ทำมาจากสารกึ่งตัวนำ เนื่องจากสารกึ่งตัวนำจะให้แรงดันเอาต์พุตสูงกว่าตัวนำไฟฟ้าทั่วไป

โดยในปริยญาณิพนธ์ฉบับนี้เลือกใช้ ไอซี ACS712 ดังรูปที่ 2.10 โดยมีตัวอย่างต่อการใช้งานดัง รูปที่ 2.11 โดยที่ค่าแรงดันเอาต์พุต  $V_{out}$  อยู่ภายในช่วง  $0.5\text{ V} - 4.5\text{ V}$  สามารถต่อใช้งาน ไอซีโดยอ้างอิงการเชื่อมต่อพอร์ตใช้งานจากรูปที่ 2.9

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.10 ACS712



รูปที่ 2.11 ตัวอย่างการต่อใช้งานไอซี ACS712

## 2.5.2 Power IC

ในปริณิธานฉบับนี้เลือกใช้ ไอซี ADE7763 รูปที่ 2.12 โดยไอซี ADE7763 เป็นไอซี วัดพลังงาน หรือ Energy Metering IC ซึ่ง IC ตัวนี้มีความสามารถในการวัดพลัง ที่มีความถูกต้อง และความแม่นยำสูง สามารถวัดแรงดันทางไฟฟ้า กระแสทางไฟฟ้า และค่ากำลังทางไฟฟ้าต่างๆ สามารถวัดค่าทางไฟฟ้าออกมาได้หลายค่า เช่น ค่ากำลังไฟฟ้าจริง ค่ากำลังไฟฟ้าที่ปรากฏ

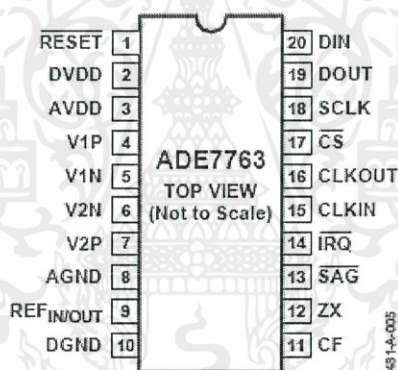


รูปที่ 2.12 ไอซี ADE7763

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

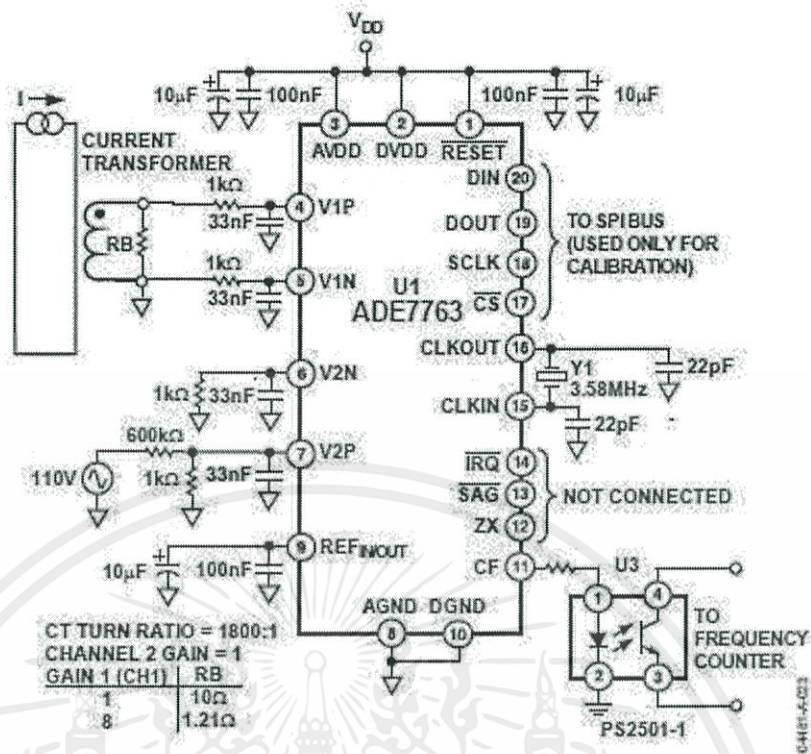
ADE7763 มีส่วนที่ใช้ติดต่อกับอุปกรณ์ภายนอก แบบอนุกรม (SPI: Serial Peripheral Interface) เพื่อใช้ในการติดต่อระหว่าง Power IC กับไมโครคอนโทรลเลอร์ได้อีกด้วย

ADE7763 มีอินพุตออกสองทาง คือ ทางที่หนึ่งเป็นอินพุตสำหรับวัดค่ากระแสไฟฟ้า (Current Channel) และอีกทาง คือ อินพุตของแรงดันไฟฟ้า (Voltage Channel) ซึ่งทั้งสอง Channel นี้ รับค่าแรงดันไฟฟ้าได้มากที่สุดไม่เกิน  $\pm 0.5$  Vrms หลังจากนั้นสัญญาณจะผ่านเข้ามาที่ตัวขยายสัญญาณ (Amplifier) ต่อมา สัญญาณที่วัดได้ ก็จะผ่านตัวแปลงสัญญาณจากอนาล็อกเป็นดิจิทัล (A to D) จากนั้นผ่านเข้าสู่ตัวกรองความถี่สูง (High Pass Filter) เพื่อตัดสัญญาณรบกวนต่าง ๆ ออก และเมื่อสัญญาณผ่านวงจรรวมอินทิเกรเตอร์ (Integrator) ทั้งสอง Channel สัญญาณจะแบ่งออกเป็นสองส่วน โดยส่วนที่หนึ่ง จะนำไปคูณกับแรงดันไฟฟ้า ผลที่ได้จะเป็นกำลังไฟฟ้าจริง (Active Power) มีหน่วยเป็นวัตต์ (Watt) และนำค่าที่ได้เก็บไว้ในรีจิสเตอร์ ซึ่งจะสะสมค่าเพิ่มขึ้นเรื่อยๆ ดังนั้นค่าในรีจิสเตอร์ จึงเป็นค่าพลังงานไฟฟ้าจริง (Active Energy) ส่วนสัญญาณที่สอง นั้นจะนำมาคำนวณ เป็นค่ารากของกำลังสองเฉลี่ย (rms) ก่อนที่จะนำมาคูณกับแรงดันไฟฟ้า (rms) เพื่อให้ได้กำลังไฟฟ้าปรากฏ (Apparent Power) มีหน่วยเป็น VA (Volt-Amp) และนำค่าที่ได้เก็บไว้ในรีจิสเตอร์ ซึ่งจะสะสมค่าเพิ่มขึ้นเรื่อยๆ ดังนั้นค่าในรีจิสเตอร์ จึงเป็นค่าพลังงานไฟฟ้าปรากฏ (Apparent Energy)



รูปที่ 2.13 โครงร่าง ADE7763

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.14 ไดอะแกรมของไอซี ADE7763

Voltage Channel เป็นอีก Input ซึ่งสัญญาณแรงดันไฟฟ้าผ่านเข้ามาแล้ว จะต้องผ่านตัวแปลงสัญญาณจากอนาล็อกเป็นดิจิตอล จากนั้นต้องผ่านส่วน การคำนวณ Phase ซึ่งเมื่อสัญญาณผ่านออกมาแล้ว จะแบ่งออกเป็นสามส่วน ส่วนที่หนึ่งจะไปคูณกับสัญญาณกระแส ซึ่งจะได้ผลลัพธ์ออกมาเป็นกำลังไฟฟ้าจริง ดังที่กล่าวไว้ ส่วนที่สอง จะนำไปคำนวณเป็นค่ารากที่สองของกำลังสองเฉลี่ย (rms) เพื่อคูณกับกระแส จากช่องทางที่หนึ่ง ที่เป็นค่ารากที่สองของกำลังสองเฉลี่ย (rms) เช่นกัน เพื่อให้ได้เป็นกำลังไฟฟ้าปรากฏ (Apparent Power) และส่วนที่สาม จะถูกนำค่าไปเก็บไว้ในรีจิสเตอร์ เพื่อส่งข้อมูลออกไป

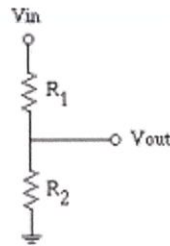
### 2.5.3 โวลต์เตจ ดีไวเดอร์

เป็นวงจรสำหรับแบ่งโวลต์เตจ ให้กับวงจรต่าง ๆ ตามต้องการ ซึ่งจำเป็นมากสำหรับแหล่งจ่ายพลังงานของเครื่องอิเล็กทรอนิกส์ ต่าง ๆ แทนที่จะสร้างแหล่งจ่ายพลังงานหลาย ๆ ชุดสำหรับค่าโวลต์เตจต่าง ๆ กัน ก็สามารถสร้างแหล่งจ่ายพลังงานที่มีโวลต์เตจ สูง เพียงชุดเดียวแล้วจึงใช้โวลต์เตจ ดีไวเดอร์ช่วยแบ่งโวลต์เตจ ลงมาตามค่าที่ต้องการ

คุณสมบัติในการหารแรงดันลงมานี้ ทำให้เป็นวิธีที่ง่ายที่สุดที่จำทำให้วัดค่าแรงดันออกมาได้ และแรงดันทางด้านเอาต์พุตจะเป็นไปตามดังรูปที่ 2.15

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## Voltage Divider



$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

รูปที่ 2.15 โวลต์เตจดีไวเดอร์

## 2.6 หลักการที่ใช้ในแหล่งจ่ายพลังงาน

## 2.6.1 หม้อแปลง

หม้อแปลงไฟฟ้า คืออุปกรณ์ที่ใช้แปลงแรงดันไฟฟ้าสลับ ให้มีขนาดแรงดันตามที่ต้องการ มีการนำหม้อแปลงไฟฟ้าไปใช้ในงานหลายด้าน ทั้งในระบบการจ่ายไฟฟ้า หรือเป็นอุปกรณ์ประกอบในเครื่องใช้ไฟฟ้าที่ใช้กันตามบ้านเรือน ไม่ว่าจะเป็น โทรทัศน์ เครื่องขยายเสียง วิทยุเทป หรือ อะแดปเตอร์แปลงไฟเพื่อใช้ในงาน จึงนับว่ามีความสำคัญและเกี่ยวข้องกับงานทางไฟฟ้าและอิเล็กทรอนิกส์อย่างมาก

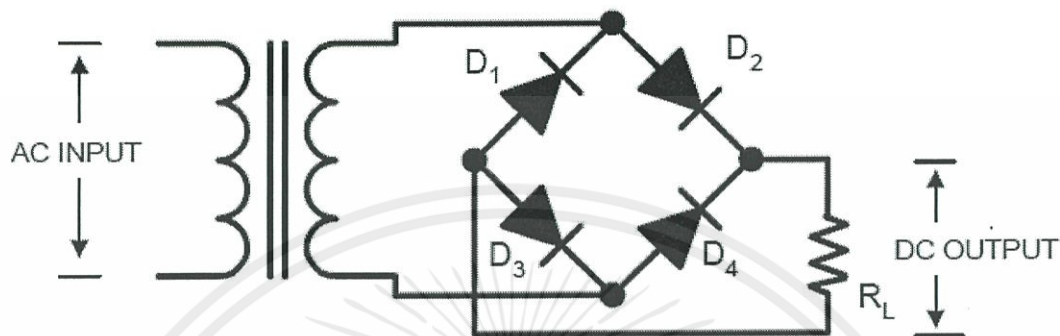


รูปที่ 2.16 หม้อแปลงแบบ 2 ขดลวด

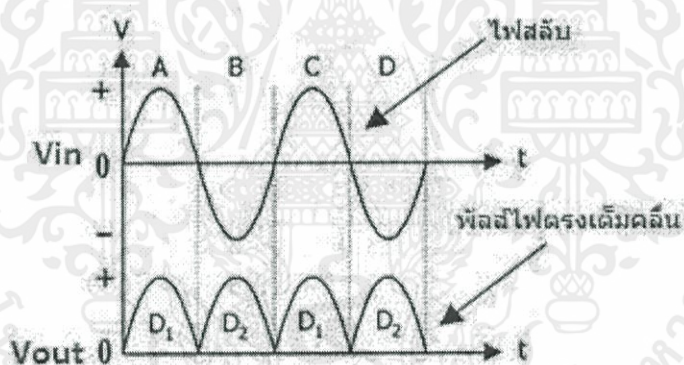
หลักการทำงานหม้อแปลง คือ ในระบบจ่ายไฟฟ้าจะมีการแปลงแรงดันไฟฟ้าสลับให้มีขนาดสูงมากๆ เช่นให้มีขนาดเป็น 48 kV หรือ 24 kV เพื่อลดขนาดของลวดตัวนำ ที่ต้องใช้ในการจ่ายไฟฟ้าเป็นระยะทางไกลๆ เมื่อถึงปลายทางก่อนที่จะจ่ายไฟฟ้าไปให้แก่บ้านเรือนต่างๆ ก็จะแปลงระดับแรงดันไฟฟ้าให้ลดลงเป็น 220 Vac เพื่อลดอันตรายที่จะเกิดแก่ผู้ใช้ไฟฟ้า และเมื่อต้องการใช้กับอุปกรณ์ไฟฟ้าที่ใช้ระดับแรงดันต่ำๆ เช่น 6 V หรือ 9 V ก็จะต้องมีการแปลงดันไฟฟ้า ตามบ้านจาก 220 Vac เป็นระดับแรงดันไฟฟ้าตามที่ต้องการ อุปกรณ์ที่ทำหน้าที่ดังกล่าว เราเรียกว่า หม้อแปลงเอกสารนี้ไฟฟ้า (Transformer) สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## 2.6.2 วงจรเรียงกระแสเต็มคลื่นแบบบริดจ์ (Bridge Rectifier)

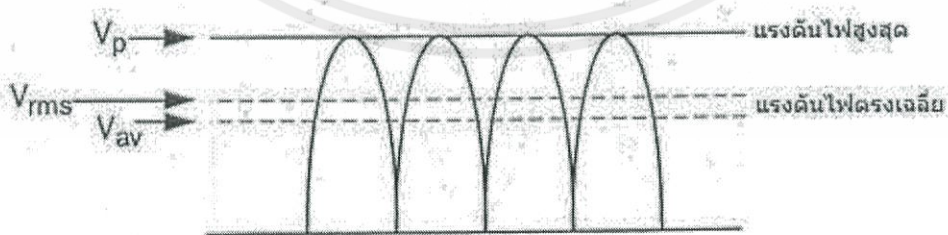
วงจรเรียงกระแสเต็มคลื่นแบบบริดจ์จะใช้ไดโอด 4 ตัวในการทำงานและสามารถใช้กับหม้อแปลงแบบ 2 หรือ 3 ขั้วก็ได้ โดยไดโอดจะผลัดกันนำกระแสครั้งละ 2 ตัว ซึ่งคลื่นเอาต์พุตที่ออกมาจะมีลักษณะดังรูปที่ 2.17 ซึ่งค่าแรงดันเอาต์พุตที่ออกมานั้นจะมีค่าเป็น 0.636 เท่าของแรงดันไฟสูงสุด



รูปที่ 2.17 วงจรเรียงกระแสเต็มคลื่นแบบบริดจ์



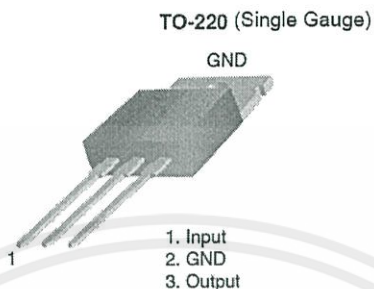
รูปที่ 2.18 รูปคลื่นแรงดันขาออกเปรียบเทียบกับแรงดันขาเข้า



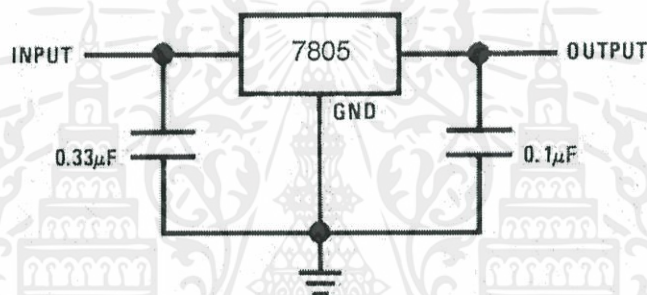
เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ โดยรูปที่ 2.19 แสดงค่าแรงดันไฟตรงเฉลี่ยกับค่าแรงดันไฟสูงสุด  $V_p$  ใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

### 2.6.3 โวลต์เตจเรกกูเลเตอร์ไอซี

ในวงจรจ่ายไฟที่ต้องมีการควบคุมระดับแรงดันและจ่ายกระแสให้คงที่ นิยมใช้ไอซีสำเร็จรูปเป็นอย่างมาก อย่างเบอร์ที่ใช้งานในโครงการอิเล็กทรอนิกส์ทั่วไปมักอยู่ในตระกูล 78XX และ 79XX เป็นไอซีเรกกูเลเตอร์แรงดันคงที่ที่ใช้งานสำหรับแรงดันไฟบวกและลบ จ่ายกระแสสูงสุด 1 แอมแปร์ ใช้งานได้ง่าย ต่ออุปกรณ์ภายนอกน้อยชิ้น



รูปที่ 2.20 โวลต์เตจเรกกูเลเตอร์ไอซีเบอร์ LM7805

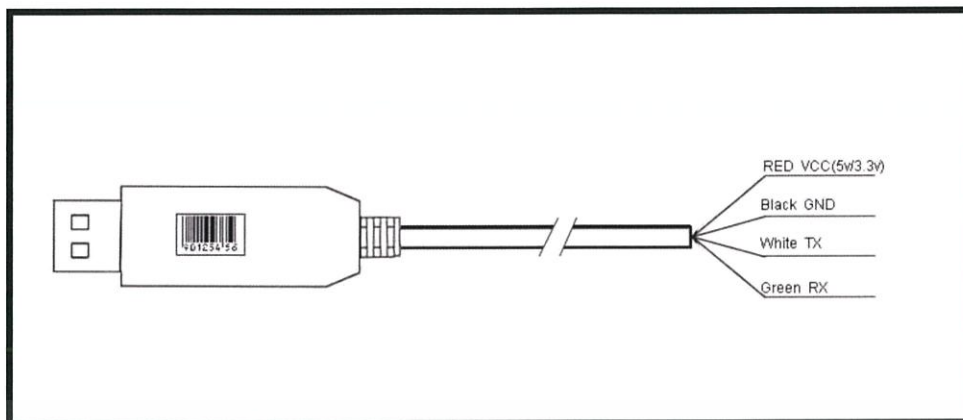


รูปที่ 2.21 วงจรโวลต์เตจเรกกูเลเตอร์ไอซีเบอร์ LM7805

## 2.7 การสื่อสาร RS-232

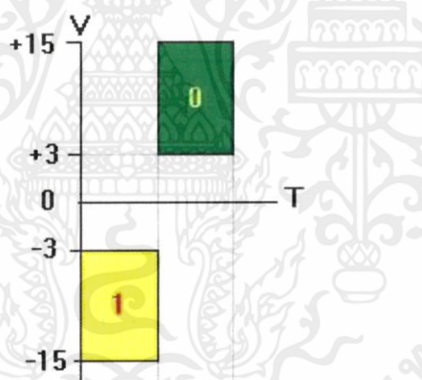
การสื่อสาร RS-232 มาจากคำว่า Recommended Standard-232 (มาตรฐานแนะนำรุ่น 232) เป็นมาตรฐานการเชื่อมต่อข้อมูล แบบอนุกรม (Serial Port) การสื่อสารแบบอนุกรม นับว่ามีความสำคัญต่อการใช้งาน ไมโครคอนโทรลเลอร์มาก เพราะสามารถใช้เป็นพิมพ์ และจอภาพของคอมพิวเตอร์ส่วนบุคคลเป็น อินพุต และ เอาต์พุต ในการติดต่อ หรือ ควบคุมไมโครคอนโทรลเลอร์ ด้วยสัญญาณอย่างน้อย เพียง 3 เส้นเท่านั้น คือ สายส่งสัญญาณ TX, สายรับสัญญาณ RX, และสายกราวด์ GND สำหรับในโครงการนี้ใช้สายในการสื่อสารแบบอนุกรมทั้งหมด 4 เส้นคือ สายส่งสัญญาณ TX, สายรับสัญญาณ RX, สายกราวด์ GND, และสาย VCC ที่ใช้เป็นแหล่งจ่ายพลังงานให้กับไมโครคอนโทรลเลอร์ ดังรูปที่ 2.22

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.22 สายในการสื่อสารแบบอนุกรม

สัญญาณรบกวนที่เกิดขึ้น ในสายนำสัญญาณ มักจะมีแรงดันเป็นบวก เมื่อเทียบกับกราวด์ เพื่อป้องกันสัญญาณรบกวนนี้ จึงออกแบบแรงดัน ของโลจิก 1 เป็นลบ คืออยู่ในช่วง  $-3\text{ V}$  ถึง  $-15\text{ V}$  ส่วนแรงดัน ของโลจิก 0 อยู่ในช่วง  $+3\text{ V}$  ถึง  $+15\text{ V}$  และเหตุที่ ระดับสัญญาณ ของ RS232 อยู่ในช่วง  $+15\text{ V}$  ถึง  $-15\text{ V}$  ก็เพื่อให้ต่อสายสัญญาณไปได้ไกลขึ้นและมีอัตราการส่งข้อมูล (Baud rate) หรือความเร็วของการรับ-ส่งข้อมูล เป็นจำนวนบิตต่อวินาทีเช่น 300, 1,200, 2,400, 4,800, 9,600, 14,400, 19,200, 38,400, 56,000 เป็นต้น



รูปที่ 2.23 ระดับสัญญาณ RS-232

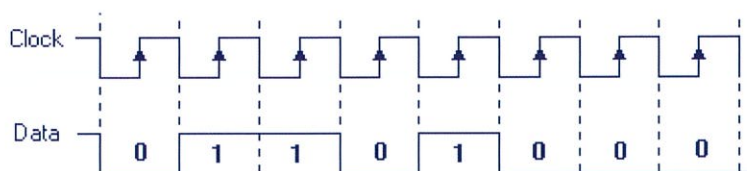
### รูปแบบการสื่อสารแบบอนุกรม

มีด้วยกันอยู่ 2 แบบ คือแบบซิงโครนัส (Synchronous) และแบบอะซิงโครนัส (Asynchronous)

#### 1. การสื่อสารแบบซิงโครนัส (Synchronous)

การรับส่งข้อมูล จะมีสัญญาณนาฬิกา ซึ่งเป็นตัวกำหนด จังหวะเวลา การส่งข้อมูล ร่วมอยู่ด้วยอีกเส้นหนึ่ง ใช้คู่กับสัญญาณข้อมูล ตัวอย่างเช่น การส่งสัญญาณจากคีย์บอร์ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

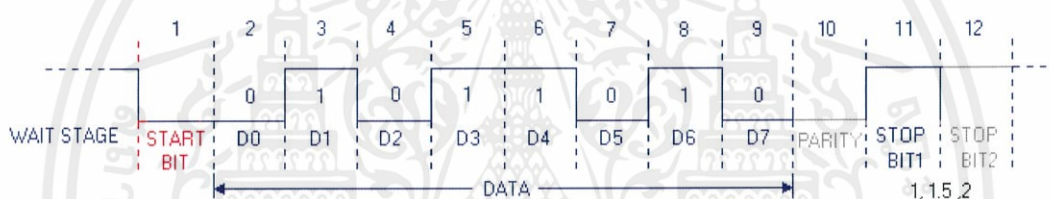


รูปที่ 2.24 สัญญาณการสื่อสารแบบซิงโครนัส

## 2. การสื่อสารแบบอะซิงโครนัส (Asynchronous)

การรับส่งข้อมูล โดยที่ไม่จำเป็นต้อง มีสัญญาณนาฬิกา ร่วมด้วย แต่จะใช้ให้ตัวส่ง และตัวรับ มีอัตราส่งข้อมูล ที่เท่ากันรูปแบบข้อมูลแบบอะซิงโครนัส ประกอบด้วย 4 ส่วนคือ

1. บิตเริ่มต้น (Start bit) มีขนาด 1 บิต
2. บิตข้อมูล (Data) มีขนาด 5, 6, 7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี (Parity bit) มีขนาด 1 บิตหรือไม่มี
4. บิตหยุด (Stop bit) มีขนาด 1, 1.5, 2 บิต



รูปที่ 2.25 สัญญาณการสื่อสารแบบอะซิงโครนัส

- เมื่อไม่มีการส่งข้อมูล ขา data จะมีสถานะเป็นลอจิก 1 หรือ สถานะหยุดรอ (Waiting stage)
- เมื่อเริ่มต้นส่งข้อมูลจะให้ขา data เป็นลอจิก 0 เป็นจำนวน 1 บิต เรียกว่าบิตเริ่มต้น (Start bit)
- จากนั้นก็จะเริ่มต้นส่งข้อมูล โดยส่งบิตต่ำไปก่อน (LSB)
- แล้วตามด้วยพาริตีบิต (จะมีหรือไม่มีก็ได้ ขึ้นอยู่กับการติดตั้งค่า ของทั้งสองฝ่าย)
- สุดท้ายตามด้วยลอจิก 1 อย่างน้อย 1 บิต ( มีขนาด 1, 1.5, หรือ 2 บิต) เพื่อแสดงว่าสิ้นสุดข้อมูล

## 2.8 หลักออกแบบเว็บไซต์

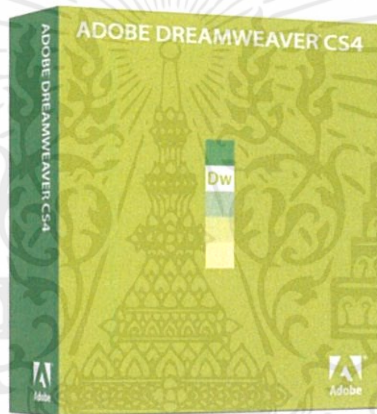
การสร้างเว็บไซต์ต้องมีการจัดเตรียมข้อมูล ออกแบบหน้าเว็บเพจ ให้มีรูปแบบที่สวยงาม สามารถใช้งานได้ง่าย และมีความสะดวกต่อการเข้าถึงข้อมูล การออกแบบเว็บไซต์จึงมีความสำคัญในการสร้างความประทับใจให้กับผู้ที่เข้ามาใช้บริการ ทำให้ผู้ใช้งานเข้าใจและเข้าถึงสิ่งที่ต้องการนำเสนอ และต้องการสื่อออกมาด้วย

เอกสารนี้เป็นเอกสารลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

### 2.8.1 การออกแบบ

1. ควรมีจุดเด่นหรือเอกลักษณ์ เนื่องจากเว็บไซต์แต่ละเว็บไซต์ก็มีเป้าหมายที่แตกต่าง ต้องหาจุดเด่นหรือเอกลักษณ์ เพื่อให้ผู้ใช้งานจดจำและเข้าใจง่าย
2. ควรมีการจัดวางข้อมูลที่เรียบง่าย เพื่อให้ผู้ที่เข้ามาใช้บริการสามารถอ่านเนื้อหาที่อยู่ภายในแต่ละเว็บเพจได้ง่าย สบายตา การจัดวางข้อมูลภายในเว็บไซต์ต้องไม่ซับซ้อน มีการจัดการที่ดีเป็นระบบ หมวดยุทธ์ให้กับข้อมูล ภาพกราฟิก ตัวอักษรอย่างเหมาะสม เพื่อให้ผู้เข้ามาใช้บริการสามารถเข้าถึงเนื้อหาที่ใช้งานได้อย่างสะดวกและรวดเร็ว

โปรแกรมในการสร้างเว็บไซต์มีด้วยกันหลายโปรแกรม ยกตัวอย่างเช่น Joomla , Drupal Opensource , NetBeans , Dreamweaver เป็นต้น โดยในที่นี้ ได้เลือกโปรแกรม Dreamweaver มาในการสร้างตัวเว็บไซต์นี้ โดยมีขั้นตอนในการดาวน์โหลดและติดตั้งโปรแกรม Dreamweaver ดังนี้ โดยในที่นี้ได้เลือก Version CS4



รูปที่ 2.22 โปรแกรม Dreamweaver CS4

### 2.8.2 ระบบฐานข้อมูล

ฐานข้อมูล (database) หมายถึง กลุ่มของข้อมูลที่ถูกเก็บรวบรวมไว้ โดยมีความสัมพันธ์ซึ่งกันและกัน โดยไม่ได้บังคับว่าข้อมูลทั้งหมดนี้จะต้องเก็บไว้ในแฟ้มข้อมูลเดียวกันหรือแยกเก็บหลาย ๆ แฟ้มข้อมูล นั่นก็คือการเก็บข้อมูลในฐานข้อมูลนั้นเราอาจจะเก็บทั้งฐานข้อมูล โดยใช้แฟ้มข้อมูลเพียงแฟ้มข้อมูลเดียวกันได้ หรือจะเก็บไว้ในหลาย ๆ แฟ้มข้อมูล ที่สำคัญคือจะต้องสร้างความสัมพันธ์ระหว่างระเบียบและเรียกใช้ความสัมพันธ์นั้นได้ มีการกำจัดความซ้ำซ้อนของข้อมูลออก และเก็บแฟ้มข้อมูลเหล่านี้ไว้ที่ศูนย์กลาง เพื่อที่จะนำข้อมูลเหล่านี้มาใช้ร่วมกัน ควบคุมดูแลรักษาเมื่อผู้ต้องการใช้งานและผู้มีสิทธิ์จะใช้ข้อมูลนั้นสามารถดึงข้อมูลที่ต้องการออกไปใช้ได้ ข้อมูลบางส่วนอาจใช้ร่วมกับผู้อื่นได้ แต่บางส่วนผู้มีสิทธิ์เท่านั้นจึงจะสามารถใช้ได้ โดยทั่วไปองค์กรต่าง ๆ จะสร้างฐานข้อมูลไว้ เพื่อเก็บข้อมูลต่าง ๆ ของตัวองค์กร โดยเฉพาะอย่างยิ่งข้อมูลในเชิงธุรกิจ เช่น ข้อมูลของลูกค้า ข้อมูลของสินค้า ข้อมูลของลูกค้า และการจ้างงาน เป็นต้น การควบคุมดูแลการใช้ฐานข้อมูลนั้น เป็นเรื่องที่ยุทธ์ยากกว่าการใช้แฟ้มข้อมูลมาก เพราะเราจะต้องตัดสินใจว่าโครงสร้างในการจัดเก็บข้อมูลควรจะเป็นเช่นไร การเขียนโปรแกรมเพื่อสร้างและเรียกใช้ข้อมูลจากโครงสร้างเหล่านี้ ถ้า

โปรแกรมเหล่านี้เกิดทำงานผิดพลาดขึ้นมา ก็จะทำให้เกิดความเสียหายต่อโครงสร้างของข้อมูลทั้งหมดได้ เพื่อเป็นการลดภาระการทำงานของผู้ใช้ จึงได้มีส่วนของฮาร์ดแวร์และโปรแกรมต่าง ๆ ที่สามารถเข้าถึงและจัดการข้อมูลในฐานข้อมูลนั้น เรียกว่า ระบบจัดการฐานข้อมูล หรือ DBMS (data base management system) ระบบจัดการฐานข้อมูล คือ ซอฟต์แวร์ที่เปรียบเสมือนสื่อกลางระหว่างผู้ใช้และโปรแกรมต่าง ๆ ที่เกี่ยวข้องกับการใช้ฐานข้อมูล ซึ่งมีหน้าที่ช่วยให้ผู้ใช้เข้าถึงข้อมูลได้ง่ายสะดวก และมีประสิทธิภาพ การเข้าถึงข้อมูลของผู้ใช้อาจเป็นการสร้างฐานข้อมูล การแก้ไขฐานข้อมูล หรือการตั้งคำถามเพื่อให้ข้อมูลมา โดยผู้ใช้ไม่จำเป็นต้องรับรู้เกี่ยวกับรายละเอียดภายในโครงสร้างของฐานข้อมูล เปรียบเสมือนเป็นสื่อกลางระหว่างผู้ใช้และโปรแกรมต่าง ๆ ที่เกี่ยวข้องกับการใช้ฐานข้อมูล

### ความสำคัญของระบบฐานข้อมูล

การจัดข้อมูลให้เป็นระบบฐานข้อมูลทำให้ข้อมูลมีส่วนดีกว่าการเก็บข้อมูลในรูปของแฟ้มข้อมูล เพราะการจัดเก็บข้อมูลในระบบฐานข้อมูล จะมีส่วนที่สำคัญกว่าการจัดเก็บข้อมูลในรูปของแฟ้มข้อมูลดังนี้

1. ลดการเก็บข้อมูลที่ซ้ำซ้อน ข้อมูลบางชุดที่อยู่ในรูปของแฟ้มข้อมูลอาจมีปรากฏอยู่หลาย ๆ แห่ง เพราะมีผู้ใช้ข้อมูลชุดนี้หลายคน เมื่อใช้ระบบฐานข้อมูลแล้วจะช่วยให้ความซ้ำซ้อนของข้อมูลลดน้อยลง เช่น ข้อมูลอยู่ในแฟ้มข้อมูลของผู้ใช้หลายคน ผู้ใช้แต่ละคนจะมีแฟ้มข้อมูลเป็นของตนเอง ระบบฐานข้อมูลจะลดการซ้ำซ้อนของข้อมูลเหล่านี้ให้มากที่สุด โดยจัดเก็บในฐานข้อมูลไว้ที่เดียวกัน ผู้ใช้ทุกคนที่ต้องการใช้ข้อมูลชุดนี้จะใช้โดยผ่านระบบฐานข้อมูล ทำให้ไม่เปลืองเนื้อที่ในการเก็บข้อมูลและลดความซ้ำซ้อนลงได้

2. รักษาความถูกต้องของข้อมูล เนื่องจากฐานข้อมูลมีเพียงฐานข้อมูลเดียว ในกรณีที่มีข้อมูลชุดเดียวกันปรากฏอยู่หลายแห่งในฐานข้อมูล ข้อมูลเหล่านี้จะต้องตรงกัน ถ้ามีการแก้ไขข้อมูลนี้ทุก ๆ แห่งที่ข้อมูลปรากฏอยู่จะแก้ไขให้ถูกต้องตามกันหมดโดยอัตโนมัติด้วยระบบจัดการฐานข้อมูล

3. การป้องกันและรักษาความปลอดภัยให้กับข้อมูลทำได้อย่างสะดวก การป้องกันและรักษาความปลอดภัยกับข้อมูลระบบฐานข้อมูลจะให้เฉพาะผู้ที่เกี่ยวข้องเท่านั้นจึงจะมีสิทธิ์เข้าไปใช้ฐานข้อมูลได้เรียกว่ามีสิทธิ์ส่วนบุคคล (privacy) ซึ่งก่อให้เกิดความปลอดภัย (security) ของข้อมูลด้วย ฉะนั้นผู้ใดจะมีสิทธิ์ที่จะเข้าถึงข้อมูลได้จะต้องมีการกำหนดสิทธิ์กันไว้ก่อนและเมื่อเข้าไปใช้ข้อมูลนั้น ๆ ผู้ใช้จะเห็นข้อมูลที่ถูกเก็บไว้ในฐานข้อมูลในรูปแบบที่ผู้ใช้ออกแบบไว้

ตัวอย่างเช่น ผู้ใช้สร้างตารางข้อมูลขึ้นมาและเก็บลงในระบบฐานข้อมูล ระบบจัดการฐานข้อมูลจะเก็บข้อมูลเหล่านี้ลงในอุปกรณ์เก็บข้อมูลในรูปแบบของระบบจัดการฐานข้อมูลซึ่งอาจเก็บข้อมูลเหล่านี้ลงในแผ่นจานบันทึกแม่เหล็กเป็นระเบียบ บล็อกหรืออื่น ๆ ผู้ใช้ไม่จำเป็นต้องรับรู้โครงสร้างของแฟ้มข้อมูลนั้นเป็นอย่างไร ปล่อยให้ทำหน้าที่ของระบบจัดการฐานข้อมูล

ไม่ว่าการฝึกๆ ทั้งสิ้น อีก 4. สามารถใช้ข้อมูลร่วมกันได้ เนื่องจากในระบบฐานข้อมูลจะเป็นที่เก็บรวบรวม

ข้อมูลทุกอย่างไว้ ผู้ใช้แต่ละคนจึงสามารถที่จะใช้ข้อมูลในระบบได้ทุกข้อมูล ซึ่งถ้าข้อมูลไม่ได้ถูกจัดให้

เป็นระบบฐานข้อมูลแล้ว ผู้ใช้ก็จะใช้ได้เพียงข้อมูลของตนเองเท่านั้น เช่น ดังภาพที่ 4.9 ข้อมูลของระบบเงินเดือน ข้อมูลของระบบงานบุคคลถูกจัดไว้ในระบบเพิ่มข้อมูลผู้ใช้ที่ใช้ข้อมูลระบบเงินเดือน จะใช้ข้อมูลได้ระบบเดียว แต่ถ้าข้อมูลทั้ง 2 ถูกเก็บไว้เป็นฐานข้อมูลซึ่งถูกเก็บไว้ในที่เดียวกัน ผู้ใช้ทั้ง 2 ระบบก็จะสามารถเรียกใช้ฐานข้อมูลเดียวกันได้ ไม่เพียงแต่ข้อมูลเท่านั้นสำหรับโปรแกรมต่าง ๆ ถ้าเก็บไว้ในฐานข้อมูลก็จะสามารถใช้ร่วมกันได้

5. มีความเป็นอิสระของข้อมูล เมื่อผู้ใช้ต้องการเปลี่ยนแปลงข้อมูลหรือนำข้อมูลมาประยุกต์ใช้ให้เหมาะสมกับโปรแกรมที่เขียนขึ้นมา จะสามารถสร้างข้อมูลนั้นขึ้นมาใช้ใหม่ได้ โดยไม่มีผลกระทบต่อระบบฐานข้อมูล เพราะข้อมูลที่ผู้ใช้นำมาประยุกต์ใช้ใหม่นั้นจะไม่กระทบต่อโครงสร้างที่แท้จริงของการจัดเก็บข้อมูล นั่นคือ การใช้ระบบฐานข้อมูลจะทำให้เกิดความเป็นอิสระระหว่างการจัดเก็บข้อมูลและการประยุกต์ใช้

6. สามารถขยายงานได้ง่าย เมื่อต้องการจัดเพิ่มเติมข้อมูลที่เกี่ยวข้องจะสามารถเพิ่มได้อย่างง่ายไม่ซับซ้อน เนื่องจากมีความเป็นอิสระของข้อมูล จึงไม่มีผลกระทบต่อข้อมูลเดิมที่มีอยู่

7. ทำให้ข้อมูลบูรณะกลับสู่สภาพปกติได้เร็วและมีมาตรฐาน เนื่องจากการจัดพิมพ์ข้อมูลในระบบที่ไม่ได้ใช้ฐานข้อมูล ผู้เขียนโปรแกรมแต่ละคนมีเพิ่มข้อมูลของตนเองเฉพาะ ฉะนั้นแต่ละคนจึงต่างก็สร้างระบบการบูรณะข้อมูลให้กลับสู่สภาพปกติในกรณีข้อมูลเสียหายด้วยตนเองและด้วยวิธีการของตนเอง จึงขาดประสิทธิภาพและมาตรฐาน แต่เมื่อมาเป็นระบบฐานข้อมูลแล้ว การบูรณะข้อมูลให้กลับคืนสู่สภาพปกติจะมีโปรแกรมชุดเดียวและมีผู้ดูแลเพียงคนเดียวที่ดูแลทั้งระบบ ซึ่งย่อมต้องมีประสิทธิภาพและเป็นมาตรฐานเดียวกันแน่นอน

### 2.8.3 ระบบการจัดการฐานข้อมูล

หน้าที่ของระบบการจัดการฐานข้อมูล

1. ระบบจัดการฐานข้อมูลเป็นซอฟต์แวร์ที่ทำหน้าที่ดังต่อไปนี้ ดูแลการใช้งานให้กับผู้ใช้ ในการติดต่อกับตัวจัดการระบบเพิ่มข้อมูลได้ ในระบบฐานข้อมูลนี้ข้อมูลจะมีขนาดใหญ่ ซึ่งจะถูกจัดเก็บไว้ในหน่วยความจำสำรองเมื่อผู้ใช้ต้องการจะใช้ฐานข้อมูล ระบบการจัดการฐานข้อมูลจะทำหน้าที่ติดต่อกับระบบเพิ่มข้อมูลซึ่งเสมือนเป็นผู้จัดการเพิ่มข้อมูล (file manager) นำข้อมูลจากหน่วยความจำสำรองเข้าสู่หน่วยความจำหลักเฉพาะส่วนที่ต้องการใช้งาน และทำหน้าที่ประสานกับตัวจัดการระบบเพิ่มข้อมูลในการจัดเก็บ เรียกใช้ และแก้ไขข้อมูล

2. ควบคุมระบบความปลอดภัยของข้อมูลโดยป้องกันไม่ให้ผู้ใช้ที่ไม่ได้รับอนุญาตเข้ามาเรียกใช้หรือแก้ไขข้อมูลในส่วนป้องกันเอาไว้ พร้อมทั้งสร้างฟังก์ชันในการจัดทำข้อมูลสำรอง โดยเมื่อเกิดความขัดข้องของระบบเพิ่มข้อมูลหรือของเครื่องคอมพิวเตอร์เกิดการเสียหายนั้น ฟังก์ชันนี้จะสามารถทำการฟื้นฟูสภาพของระบบข้อมูลกลับเข้าสู่สภาพที่ถูกต้องสมบูรณ์ได้

3. ควบคุมการใช้ข้อมูลในสภาพที่มีผู้ใช้พร้อม ๆ กันหลายคน โดยจัดการเมื่อมีข้อผิดพลาดของข้อมูลเกิดขึ้น

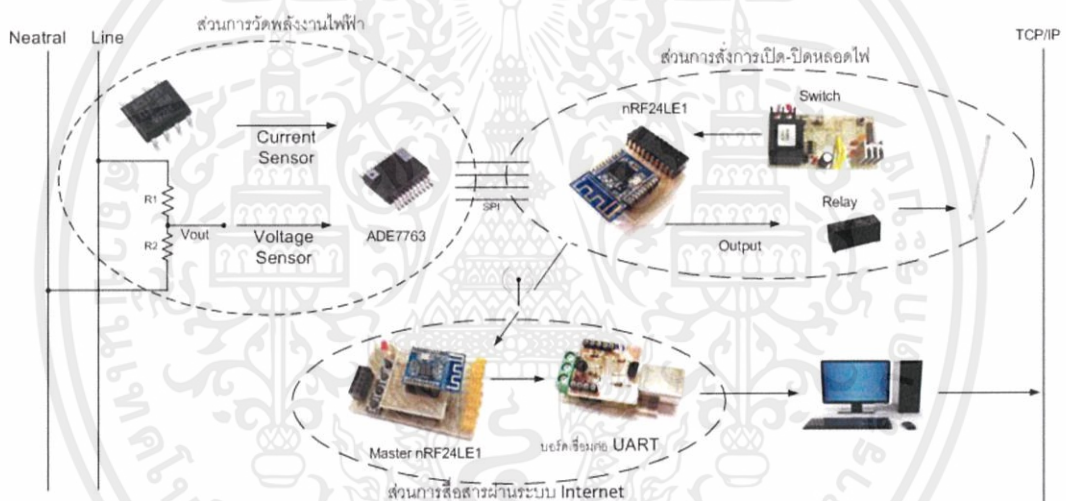
## บทที่ 3

# การออกแบบการควบคุมอุปกรณ์ส่องสว่าง

### 3.1 กล่าวนำ

ในปัจจุบันระบบไฟฟ้ายังคงต้องใช้สายไฟส่งแรงดันไฟฟ้ายังตัวอุปกรณ์ส่องสว่างและใช้สวิตซ์ในการเปิด- ปิด เพื่อให้อุปกรณ์ส่องสว่างสามารถประยุกต์ใช้งานร่วมกับเทคโนโลยีแบบไร้สายได้นั้น ได้ออกแบบโมดูลควบคุมอุปกรณ์เพื่อให้ผู้ใช้สามารถนำไปติดตั้งเข้ากับระบบในปัจจุบันได้ง่าย และยัง สามารถวัดปริมาณทางไฟฟ้า โดยโมดูลควบคุมมีช่องทางการสื่อสารผ่าน RS-232 ใช้เชื่อมต่อกับคอมพิวเตอร์ส่วนบุคคลที่มีโปรแกรมควบคุมและแสดงผลผ่านทางเว็บเบราว์เซอร์

### 3.2 แผนภาพการวางแผนโครงสร้างของโครงการ



รูปที่ 3.1 แผนภาพรวมโครงการ

### 3.3 การออกแบบโมดูลควบคุมอุปกรณ์ส่องสว่าง

โมดูลควบคุมอุปกรณ์ส่องสว่างนั้นแบ่งออกเป็น 2 ส่วนคือ โมดูลส่วนสั่งการเปิด- ปิดหลอดไฟ และโมดูลส่วนการสื่อสารผ่านทางระบบอินเทอร์เน็ต

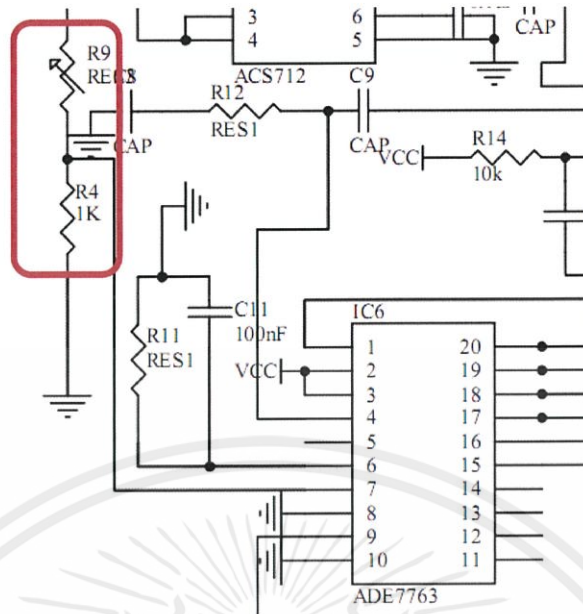
#### 3.3.1 โมดูลส่วนสั่งการเปิด- ปิดหลอดไฟ

โดยโมดูลส่วนสั่งการเปิด- ปิดหลอดไฟนั้นประกอบด้วยวงจรดังนี้

##### 3.3.1.1 โวลท์เตจดีไวเดอร์

พาวเวอร์ไอซีชิป ADE7763 ที่ได้เลือกมานั้นสามารถรับโวลท์เตจอินพุต ได้

ไม่เกิน  $\pm 0.5$  V จึงได้ใช้วงจรโวลท์เตจดีไวเดอร์ (Voltage Divider) ทำหน้าที่ในส่วนการหารแรงดันลงมา เนื่องจากแรงดันที่ทำการวัดนั้นเป็นแรงดันไฟฟ้าในบ้าน หรือ 220 Vrms และจากสมการการคำนวณวงจรโวลท์เตจดีไวเดอร์ เมื่อกำหนดค่า  $R1 = 1$  M $\Omega$  และ  $R2 = 1$  k $\Omega$  จะทำให้ได้ค่า  $V_{out} = 0.28$  Vrms ซึ่งจะทำให้ พาวเวอร์ไอซีสามารถทำงานได้ปกติ

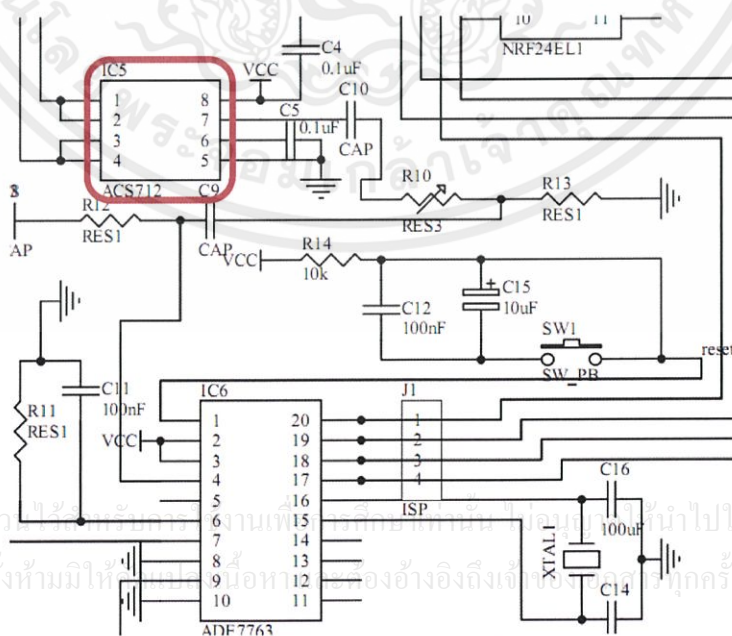


รูปที่ 3.2 วงจรโวลต์เตจดีไวเดอร์

### 3.3.1.2 เคอร์เรนทเซนเซอร์

ในส่วนของ เคอร์เรนทเซนเซอร์ (Current Sensor) นั้นจะทำการวัดค่ากระแสที่ไหลดใช้ และเนื่องจาก พาวเวอร์ไอซี ADE7763 ที่ได้เลือกมานั้นไม่สามารถรับค่ากระแสเข้ามาที่ตัวของพาวเวอร์ไอซีได้เลย ดังนั้นจึงต้องทำการแปลงกระแสที่ต้องการวัดมาเป็นแรงดันทางไฟฟ้า และแรงดันทางไฟฟ้าที่แปลงมานั้นก็ต้องไม่เกิน  $\pm 0.5$  V จึงต้องมีวงจรหารแรงดันทางไฟฟ้าจากตัว ACS712 ซึ่งเป็นเคอร์เรนทเซนเซอร์ลงมาอีกครั้ง

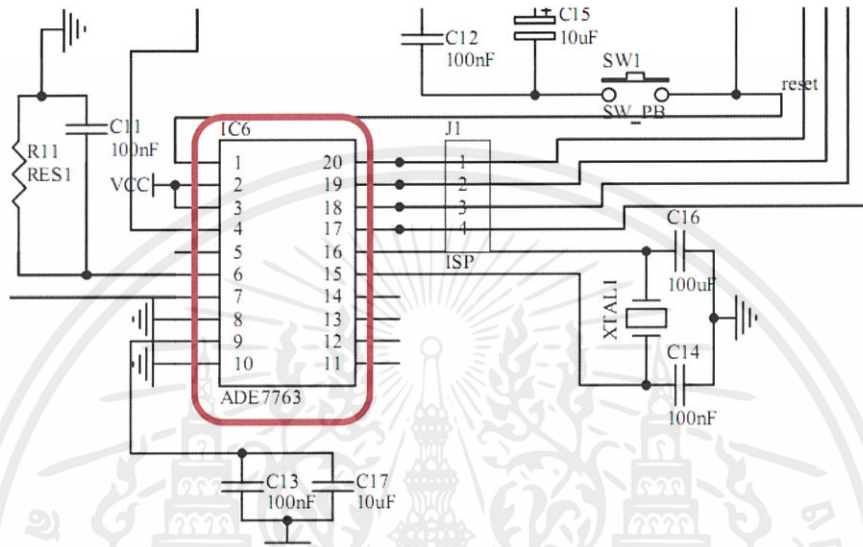
เนื่องจาก เคอร์เรนทเซนเซอร์ ACS712 มีค่า Sensitivity = 66 mV/A ดังนั้นสามารถส่งค่ากระแสไปให้พาวเวอร์ไอซี มากที่สุดและยังสามารถทำงานได้ปกติ คือ 7.57 A



รูปที่ 3.3 เคอร์เรนทเซนเซอร์ในวงจร

### 3.3.1.3 พาวเวอร์ไอซี

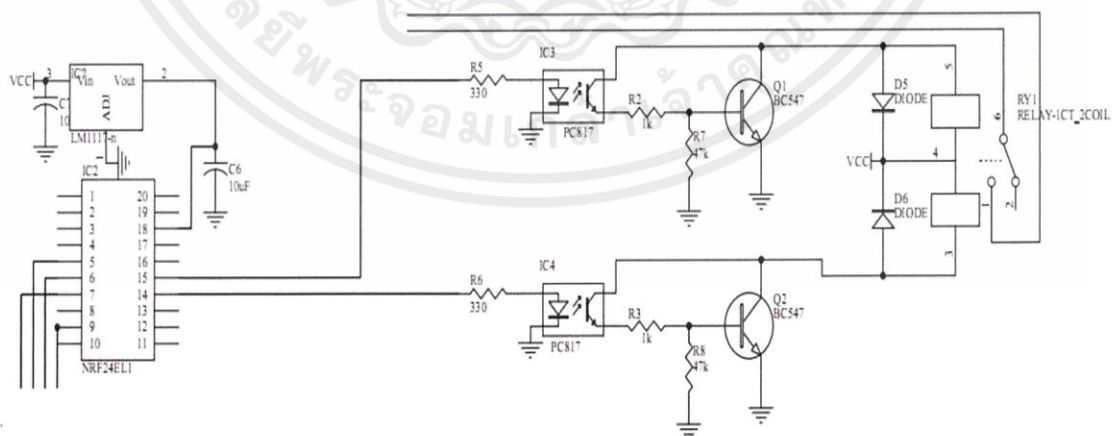
พาวเวอร์ไอซี (Power IC) ถือเป็นอุปกรณ์สำคัญในการรับค่าจากการวัดและนำมาคำนวณค่าพลังงานทางไฟฟ้าเพื่อให้ได้ค่าพลังงานทางไฟฟ้าในรูปแบบต่างๆ เช่น รับค่าจากเคอร์เรนท์เซนเซอร์ และคำนวณค่ากระแส หรือ Irms รับค่าจากวงจรโวลต์เตจดีไวเดอร์ และคำนวณค่าแรงดัน หรือ Vrms ค่าวนค่างำลังไฟฟ้าได้ทั้งในหน่วยของ Watt และ VA สามารถหาค่าพลังงานไฟฟ้าในหน่วยของ kWhr เป็นต้น



รูปที่ 3.4 พาวเวอร์ไอซีในวงจร

### 3.3.1.4 วงจรขับรีเลย์

วงจรขับรีเลย์มีหน้าที่ตัดต่อไฟฟ้าในส่วนที่ต้องการควบคุม โดยรับคำสั่งจากไมโครคอนโทรลเลอร์ โดยรีเลย์ที่ใช้เป็นแบบแลตซ์รีเลย์ ซึ่งหน้าสัมผัสจะเปลี่ยนแปลงเมื่อมีไฟฟ้ามาระดับ

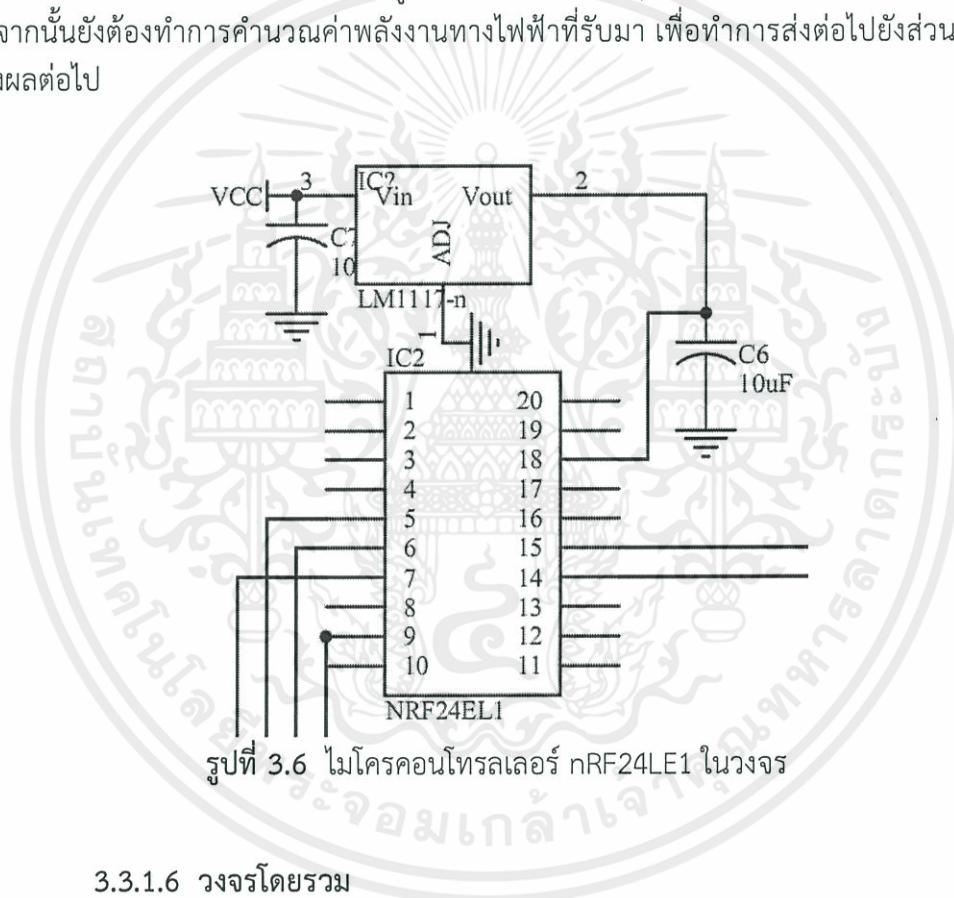


เอกสารนี้เป็นเอกสารที่สงวนไว้รูปที่ 3.5 วงจรขับรีเลย์ที่รับคำสั่งจากไมโครคอนโทรลเลอร์ ซึ่งประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

### 3.3.1.5 ไมโครคอนโทรลเลอร์ (nRF24le1)

หน้าที่ของไมโครคอนโทรลเลอร์ในส่วนการสั่งการเปิด-ปิดหลอดไฟนั้นจะทำหน้าที่ควบคุมการทำงานของรีเลย์ การควบคุมการทำงานนั้นไมโครคอนโทรลเลอร์จะรับคำสั่งมาจากอุปกรณ์สั่งการอื่น ๆ ได้แก่ สวิตช์เวอร์เลส หรือ รีโมทสั่งการ เมื่อได้รับคำสั่งมาแล้วนั้นจะทำการประมวลผลว่าคำสั่งที่ได้รับนั้นคือคำสั่งเปิดหลอดไฟหรือปิดหลอดไฟ เมื่อได้ผลลัพธ์จากการประมวลผลแล้วนั้นก็จะส่งคำสั่งไปสั่งการยังแลตซ์รีเลย์ต่อไป

ไมโครคอนโทรลเลอร์ (nRF24le1) ยังมีหน้าที่ส่วนของการวัดค่าพลังงานทางไฟฟ้า จะทำหน้าที่รับค่าพลังงานทางไฟฟ้าที่สามารถวัดมาได้โดยค่าพลังงานทางไฟฟ้าที่วัดได้นั้นจะเข้ามายัง พาวเวอร์ไอซี ADE7763 เพื่อทำการคำนวณค่าในรูปแบบอื่นๆ และค่าที่ได้จากพาวเวอร์ไอซี ADE7763 จะส่งต่อเข้ามายังไมโครคอนโทรลเลอร์ (nRF24le1) ซึ่งการส่งค่าจากพาวเวอร์ไอซีมายังไมโครคอนโทรลเลอร์นั้นจะใช้การส่งในรูปแบบ Serial Peripheral Interface หรือ SPI นั้นเอง นอกจากนี้ยังต้องทำการคำนวณค่าพลังงานทางไฟฟ้าที่รับมา เพื่อทำการส่งต่อไปยังส่วนของการแสดงผลต่อไป



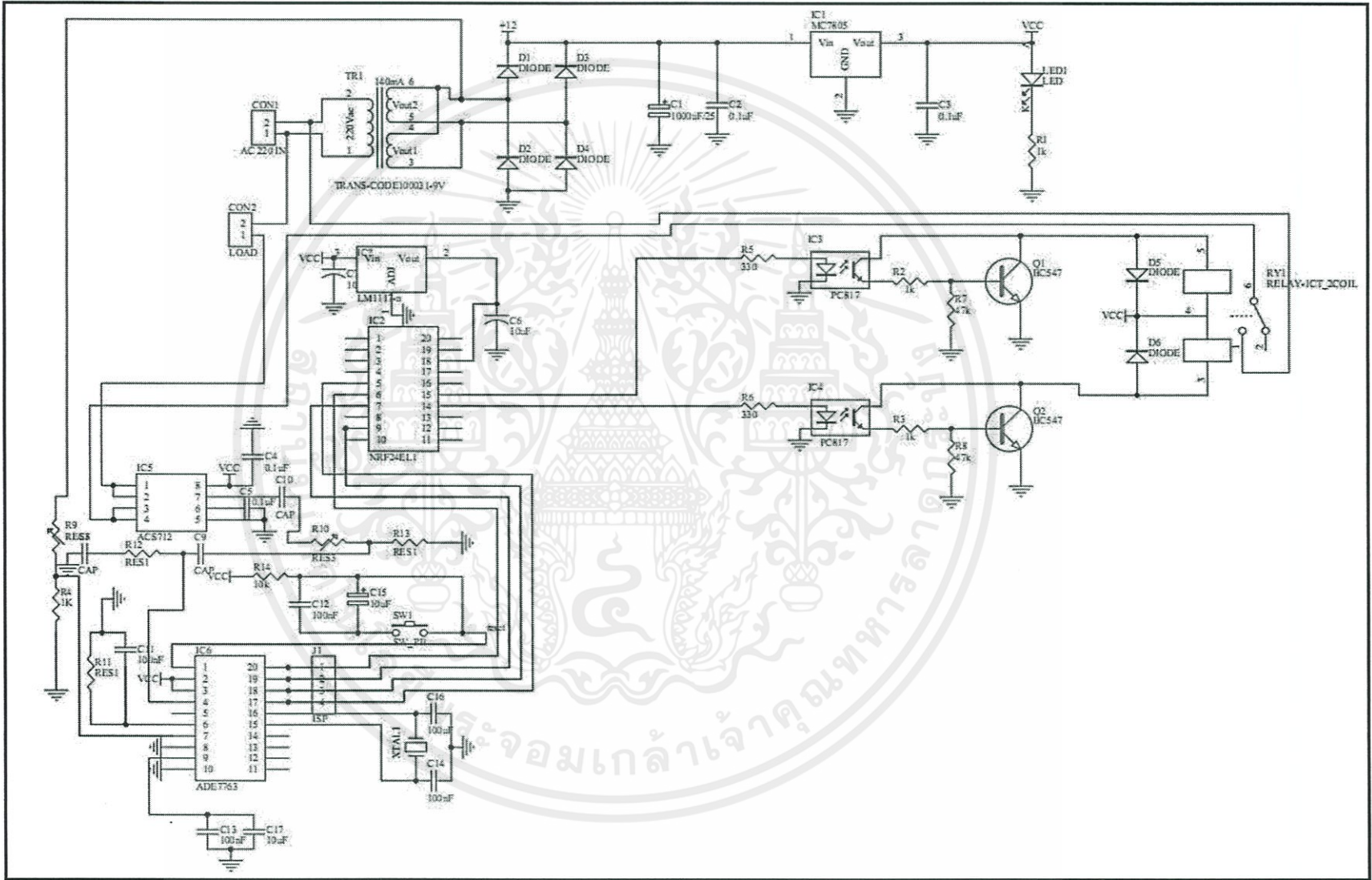
รูปที่ 3.6 ไมโครคอนโทรลเลอร์ nRF24LE1 ในวงจร

### 3.3.1.6 วงจรโดยรวม

เมื่อออกแบบวงจรการใช้งานในไมโครสั่งการเปิด-ปิดในแต่ละส่วนครบแล้ว จึงนำวงจรทั้งหมดมาเขียนแบบวงจรการทำงานของทุกส่วนร่วมกัน ดังรูปที่ 3.6

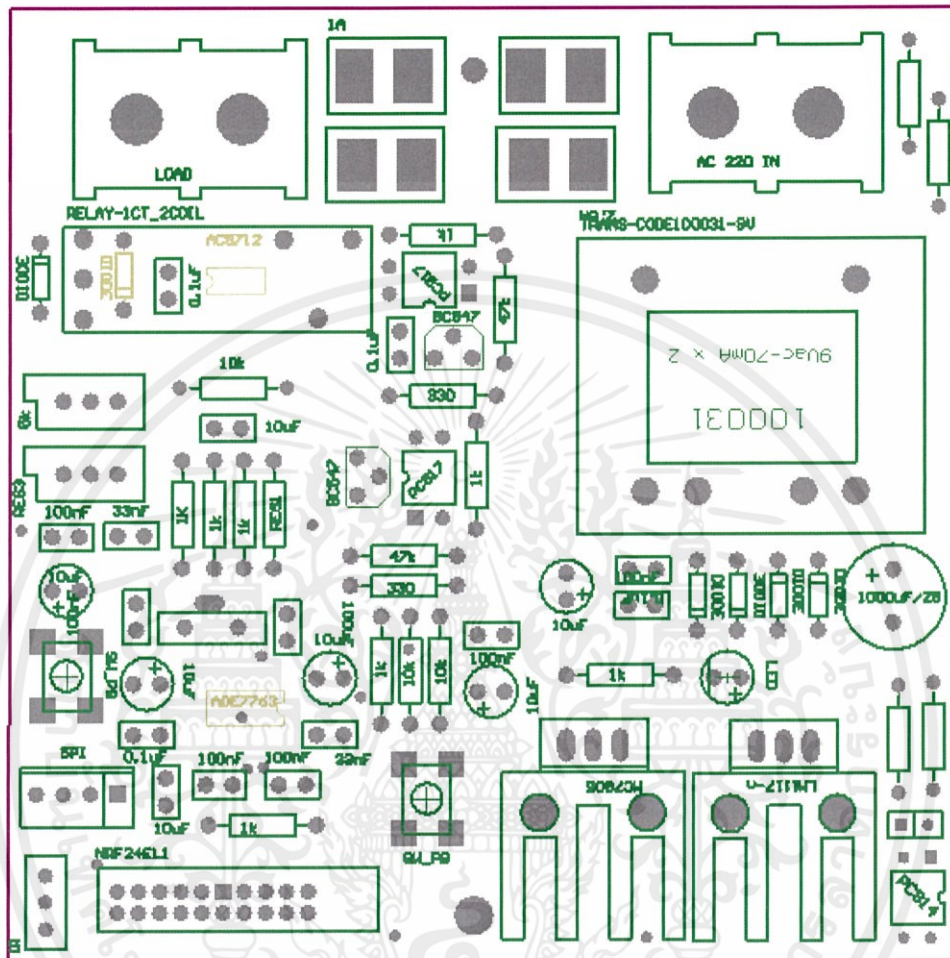
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3.7 วงจรโดยรวม



### 3.3.1.7 การออกแบบแผ่นลายวงจร

จากวงจรโดยรวมตามรูปที่ 3.6 ได้นำมาออกแบบลายวงจรที่ใช้งานด้วยโปรแกรม Protel



รูปที่ 3.8 ลายวงจรที่ออกแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



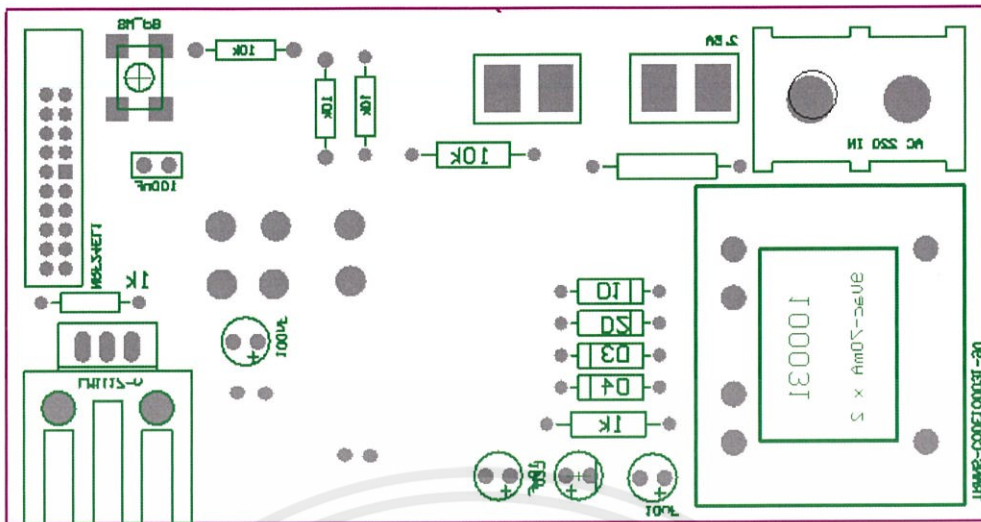
รูปที่ 3.9 วงจรที่ใช้งาน

### 3.3.1.8 สวิตช์ไมโครคอนโทรลเลอร์ (nRF24le1)

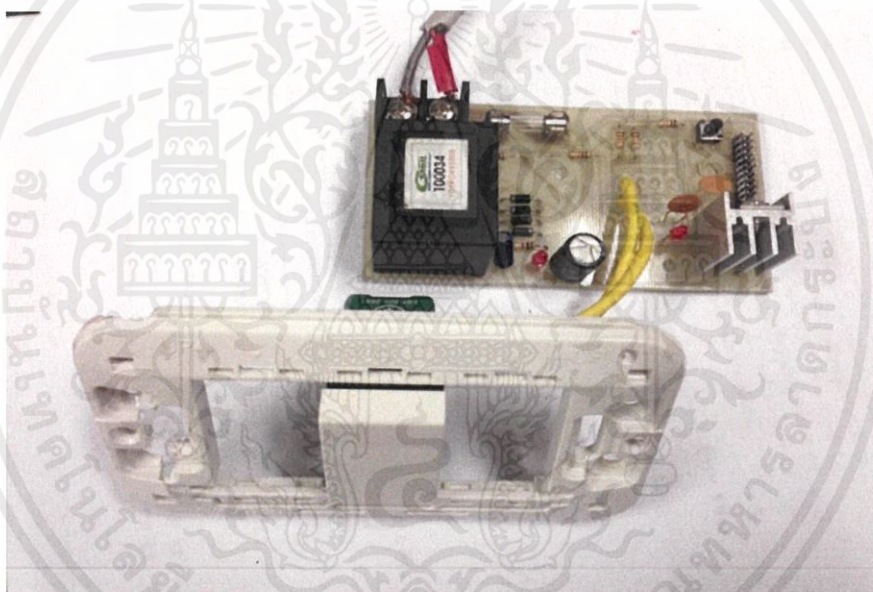
บอร์ดสวิตช์ไมโครคอนโทรลเลอร์จะมีลักษณะเหมือนบอร์ดสวิตช์ไฟบ้านทั่วไป และก็ยังทำหน้าที่เหมือนสวิตช์ไฟบ้านทั่วไปคือสามารถทำการสั่งการเปิดหรือปิดหลอดไฟภายในบ้านได้ปกติด้วย แต่ส่วนที่แตกต่างกันคือบอร์ดสวิตช์ไมโครคอนโทรลเลอร์นี้จะใช้ระบบการส่งคำสั่งที่ได้จากผู้นั้นก็คือเมื่อผู้ใช้กดเปิดหรือปิดสวิตช์สัญญาณจะเข้าไปที่ไมโครคอนโทรลเลอร์ เมื่อไมโครคอนโทรลเลอร์ประมวลผลเสร็จก็จะส่งคำสั่งไปยังโมดูลหรือไมโครคอนโทรลเลอร์อีกตัวที่ติดอยู่กับหลอดไฟแบบไร้สาย ด้วยความแตกต่างจากการส่งคำสั่งนั้นทำให้บอร์ดสวิตช์ไมโครคอนโทรลเลอร์สามารถเปลี่ยนตำแหน่งการติดตั้งได้อีกด้วย โดยในลายวงจรของบอร์ดสวิตช์ไมโครคอนโทรลเลอร์นั้นจะประกอบไปด้วยวงจรเรียงกระแสเต็มคลื่นแบบบริดจ์ จากนั้นจะใช้ไอซีเรกูเลเตอร์ในการควบคุมระดับแรงดันและจ่ายกระแสให้คงที่ดังรูปที่ 3.10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้





รูปที่ 3.11 ลายวงจรบอร์ดสวิตช์ไมโครคอนโทรลเลอร์



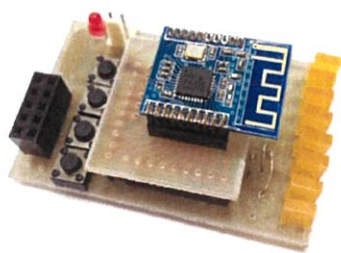
รูปที่ 3.12 วงจรบอร์ดสวิตช์ไมโครคอนโทรลเลอร์ที่ใช้งาน

### 3.3.2 ส่วนการสื่อสารผ่านระบบอินเทอร์เน็ต

#### 3.3.2.1 มาสเตอร์ไมโครคอนโทรลเลอร์ (nRF24le1)

บอร์ดมาสเตอร์ไมโครคอนโทรลเลอร์ก็ได้ใช้ไมโครคอนโทรลเลอร์ nRF24LE1 เหมือนกับในส่วนของการส่งเปิด-ปิดหลอดไฟ แต่ไมโครคอนโทรลเลอร์ในบอร์ดนี้จะทำหน้าที่เป็นตัวมาสเตอร์เพื่อรับค่าจากไมโครคอนโทรลเลอร์ตัวอื่นๆ แล้วทำการส่งข้อมูลต่อไปยังบอร์ดเชื่อมต่อ UART

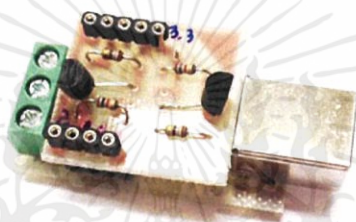
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.13 มาสเตอร์ ไมโครคอนโทรลเลอร์ nRF24LE1

### 3.3.2.2 บอร์ดเชื่อมต่อ UART

อุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัสนั่นเอง สำหรับการสื่อสารอนุกรมบนคอมพิวเตอร์แล้ว UART ถือว่าเป็นหัวใจสำคัญของการสื่อสารแบบอนุกรม



รูปที่ 3.14 บอร์ด UART

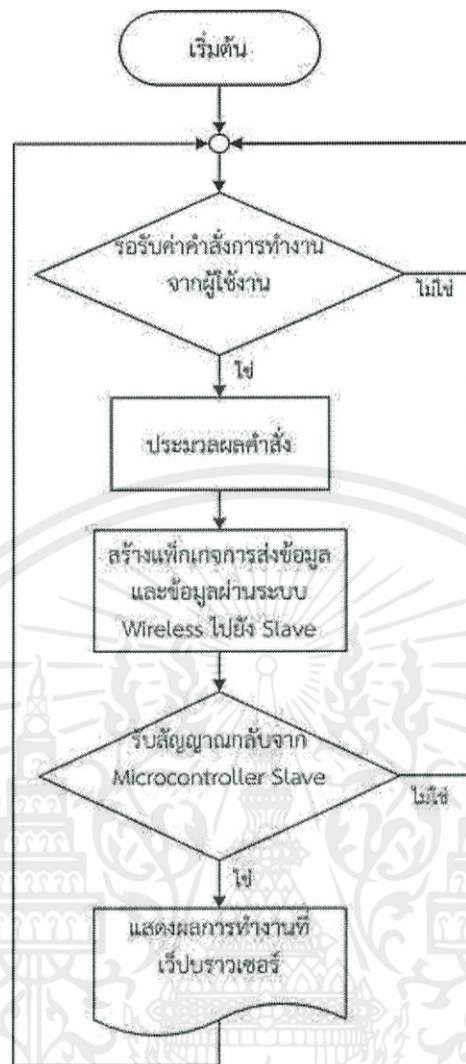
## 3.4 การออกแบบการทำงานของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์จะแบ่งการทำงานออกเป็น 2 ส่วน คือ มาสเตอร์ และ สเลฟ ส่วนของมาสเตอร์ไมโครคอนโทรลเลอร์ จะเชื่อมต่ออยู่กับคอมพิวเตอร์ที่ติดต่อกับเซิร์ฟเวอร์อินเทอร์เน็ต และ ส่วนของสเลฟไมโครคอนโทรลเลอร์ นั้นจะอยู่ในกล่องอุปกรณ์ส่งสัญญาณอินฟราเรด ควบคุมเครื่องปรับอากาศ

### 3.4.1 การทำงานของมาสเตอร์ไมโครคอนโทรลเลอร์

มาสเตอร์ไมโครคอนโทรลเลอร์ เปรียบเสมือนอุปกรณ์สื่อกลางในการสื่อสารระหว่างผู้ใช้งานและอุปกรณ์โดยมีขั้นตอนการทำงานดังรูปที่ 3.14

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



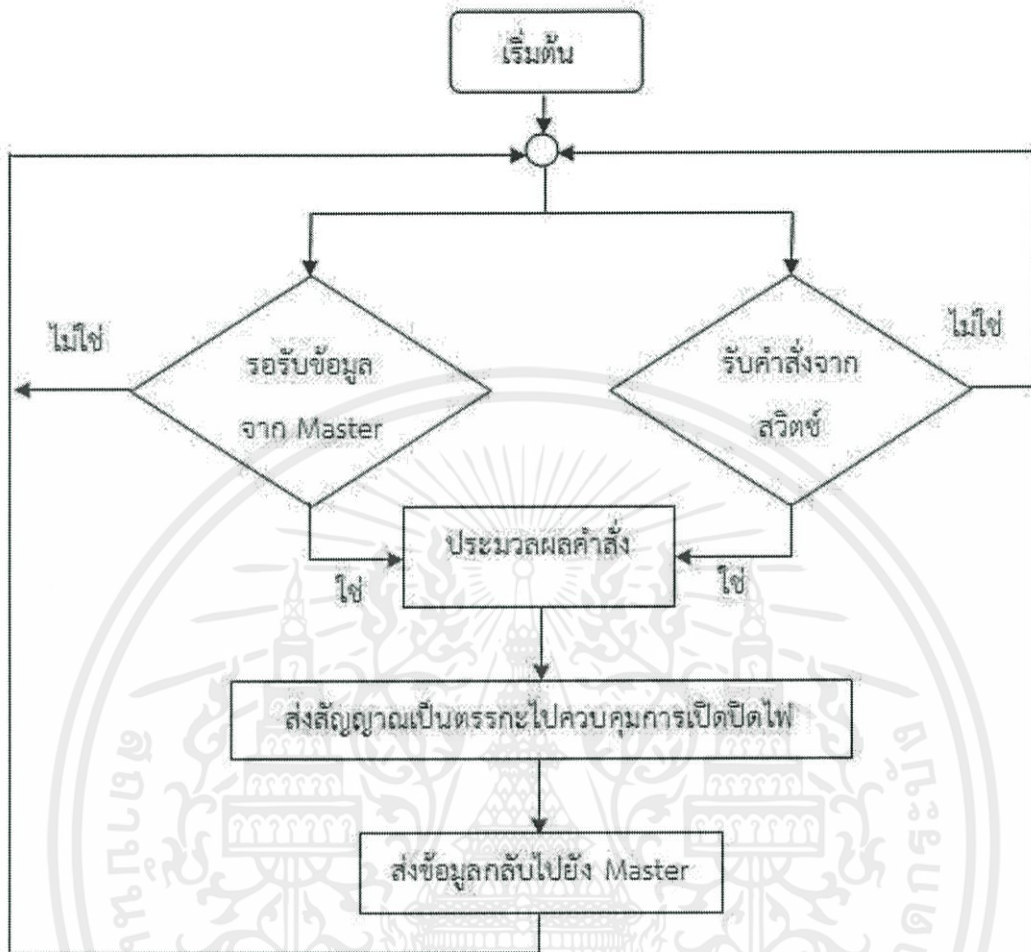
รูปที่ 3.15 ขั้นตอนการทำงานของมาสเตอร์ไมโครคอนโทรลเลอร์

มาสเตอร์ไมโครคอนโทรลเลอร์เชื่อมต่ออยู่กับคอมพิวเตอร์ด้วยพอร์ตอนุกรม ผ่านทางบอร์ด Uart เมื่อมีการสั่งงานมาจากเว็บเบราว์เซอร์ ข้อมูลจะถูกส่งมายังไมโครคอนโทรลเลอร์จะมีการประมวลผลคำสั่ง พร้อมกับการสร้างแพ็คเกจข้อมูลเพื่อส่งข้อมูลนั้นไปยังเสลฟไมโครคอนโทรลเลอร์ โดยส่งผ่านระบบไร้สายจากนั้นเสลฟไมโครคอนโทรลเลอร์จะส่งค่ากลับมายังมาสเตอร์ไมโครคอนโทรลเลอร์ เพื่อนำไปแสดงสถานะทำงานของอุปกรณ์ส่องสว่างบนหน้าเว็บเบราว์เซอร์

#### 3.4.2 การทำงานของเสลฟไมโครคอนโทรลเลอร์

เสลฟไมโครคอนโทรลเลอร์ จะเชื่อมต่อกับวงจรเพื่อส่งสัญญาณไปควบคุมหลอดไฟ โดยรับคำสั่งมาจากมาสเตอร์ไมโครคอนโทรลเลอร์ในรูปของแพ็คเกจข้อมูล จากนั้นหน่วยประมวลผลกลาง จะทำการประมวลผล และถอดคำสั่งการทำงานจากนั้นจะส่งสัญญาณไปสั่งงานรีเลย์เพื่อเปิด-ปิดหลอดไฟ ตามคำสั่งของผู้ใช้งานพร้อมกับการส่งข้อมูลกลับไปยังมาสเตอร์ไมโครคอนโทรลเลอร์ เพื่อ

แสดงสถานะของคำสั่งที่ได้ส่งงานไปแล้ว เพื่อส่งคำสั่งนั้นไปแสดงสถานะการทำงานที่หน้าเว็บเบราว์เซอร์ได้เช่นเดียวกัน



รูปที่ 3.16 ขั้นตอนการทำงานของสเลฟไมโครคอนโทรลเลอร์

### 3.5 การออกแบบและการทำงานของโปรแกรม

การออกแบบหน้าเว็บ (Application) และการออกแบบโปรแกรมสำหรับรับข้อมูลเว็บไซต์ (Application) เป็นสิ่งที่ได้รับความนิยมอย่างมากบนอินเทอร์เน็ต ซึ่งเว็บไซต์หรือแอปพลิเคชันเป็นสื่อที่อยู่ในความควบคุมของผู้ใช้โดยสมบูรณ์ กล่าวคือ ผู้ใช้สามารถตัดสินใจเลือกได้ว่า จะดูเว็บไซต์ใด ไม่ดูเว็บไซต์ใด โดยเว็บไซต์ที่ได้รับการออกแบบอย่างสวยงาม มีการใช้งานที่สะดวก ย่อมได้รับความสนใจจากผู้ชมมาก ดังนั้น การออกแบบเว็บไซต์จึงเป็นกระบวนการที่สำคัญในการสร้างเว็บไซต์ ให้ประทับใจแก่ผู้ใช้งานและมีประโยชน์อีกด้วย

การออกแบบเว็บไซต์ที่มีประสิทธิภาพนั้นต้องคำนึงถึง องค์ประกอบสำคัญดังต่อไปนี้

- ความเรียบง่าย (Simplicity)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

- ความสม่ำเสมอ (Consistency)

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ความเป็นเอกลักษณ์ (Identity)

- คุณภาพของสิ่งที่ปรากฏให้เห็นในเว็บไซต์ (Visual Appeal)
- ความสะดวกของการใช้ในสภาพต่าง ๆ (Compatibility)

### 3.5.1 การออกแบบหน้าเว็บเพจ

การออกแบบเว็บไซต์/แอปพลิเคชัน ในการแสดงผลของโครงการนี้มีดังนี้



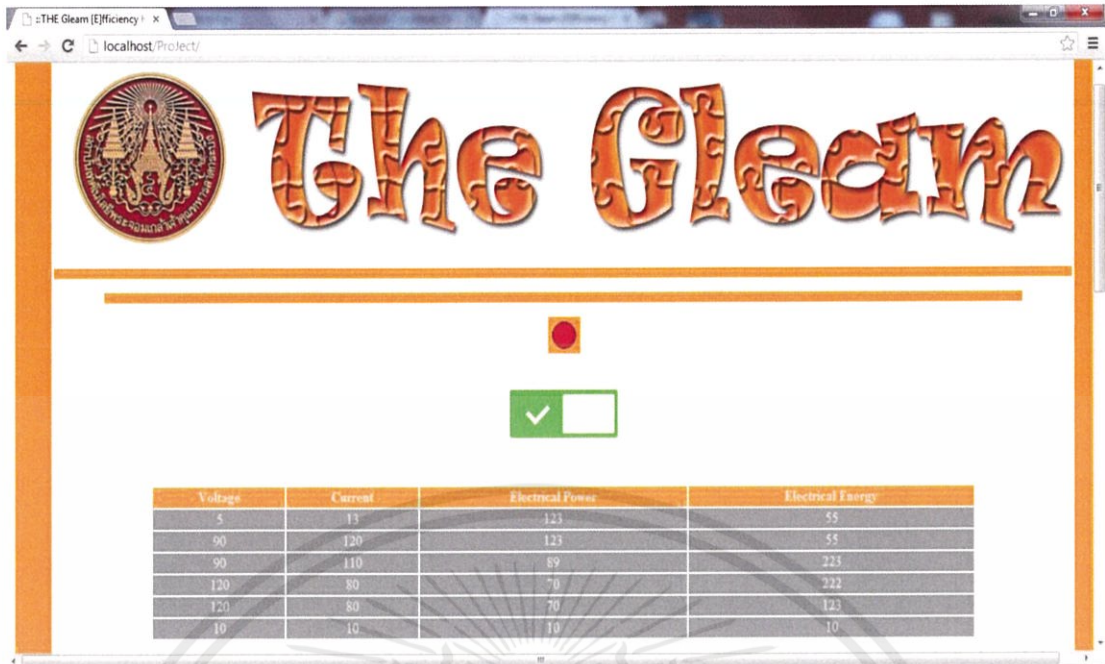
Voltage	Current	Electrical Power	Electrical Energy
5	73	121	55
90	120	121	55
90	110	59	223
120	90	75	223
120	80	70	123
10	10	10	10

โดยภาพรวมทั้งหมดจะประกอบไปด้วย

- สัญลักษณ์ของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
- ชื่อโครงการโดยเป็นการออกแบบร่วมกันของสมาชิกภายในกลุ่ม
- กราฟฟิคแสดงสถานะของหลอดไฟ
- วัน / เดือน / ปี
- ตารางแสดงผลของข้อมูลที่วัดได้

โดยพื้นฐานในการออกแบบหน้าจอในการแสดงผลหรือแอปพลิเคชันในครั้งนี้ เป็นการปรึกษาและรวบรวมความคิดเห็นจากสมาชิกภายในกลุ่ม โดยเป็นการลงความเห็นว่าจะออกแบบหน้าจอแสดงผลให้ออกมา ง่ายต่อการเข้าใจของผู้ใช้งาน , ง่ายต่อการอ่านค่าที่แสดงผลออกมา , สามารถตรวจสอบผลของการวัดย้อนหลังได้ , สามารถรับรู้สถานะของไฟที่ทำงานได้ ซึ่งจากที่กล่าวมาทั้งหมดนี้ เป็นจุดประสงค์หลักของงาน ส่วนจุดประสงค์รองลงมา คุณจะสามารรับรู้ถึง ความสวยงามของหน้าเว็บเพจ , ความทันสมัยและอื่น ๆ อีกมากมาย

โดยพื้นที่ ที่ไว้จัดเก็บหน้าเว็บไซต์นี้ อยู่ในหน่วยความจำของคอมพิวเตอร์ โดยได้ทำการจำลองว่า คอมพิวเตอร์เป็นตัวเซิร์ฟเวอร์จัดเก็บทั้งหน้าเว็บไซต์และข้อมูลที่วัดค่า ซึ่งผลที่ได้ไม่ว่าจะการแสดงผลออกมาจะปรากฏดังรูปที่ 3.16



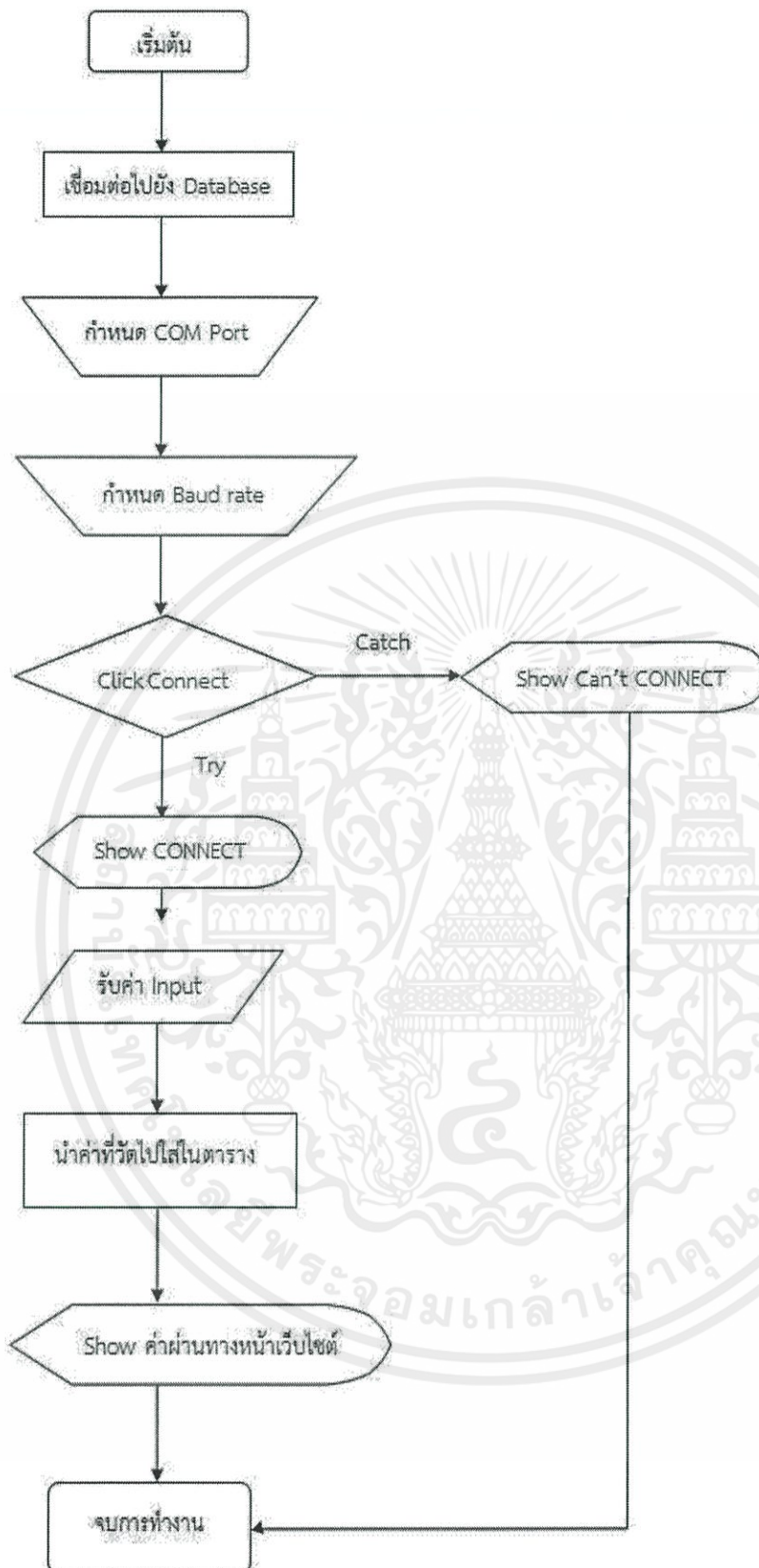
Voltage	Current	Electrical Power	Electrical Energy
5	13	123	55
90	120	123	55
90	110	89	223
120	80	70	222
120	80	70	123
10	10	10	10

รูปที่ 3.17 แสดงภาพรวมของหน้าเว็บไซต์และข้อมูลที่วัดค่าออกมา

### 3.5.2 การออกแบบโปรแกรม

โปรแกรมจะทำหน้าที่ในการรับค่าจากตัวไมโครคอนโทรลเลอร์ที่อยู่ในบอร์ด โดยเป็นการเชื่อมต่อโดยตรงผ่าน Port USB(COM Port 9) และ อัตราความเร็วในการรับส่งข้อมูล (Baudrate) เท่ากับ 38400 โดยค่าที่ส่งมามีลักษณะเป็นเฮกซ์ไฟล์ นำค่าที่ส่งมาเชื่อมต่อไปยังตารางภายในฐานข้อมูลที่ถูกสร้างเพื่อรองรับค่าที่ถูกส่งเข้ามา โดยหน้าเว็บไซต์ได้ทำการเขียนโปรแกรมให้แสดงข้อมูลที่อยู่ภายในตารางฐานข้อมูล เรียบร้อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.18 แผนภาพขั้นตอนการทำงานของโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น จากรูปแสดงถึงแผนภาพของขั้นตอนการทำงานของโปรแกรมเพื่อการรับค่าและส่งค่าไป  
แสดงผลหน้าเว็บไซต์

## บทที่ 4

### การทดสอบการทำงาน

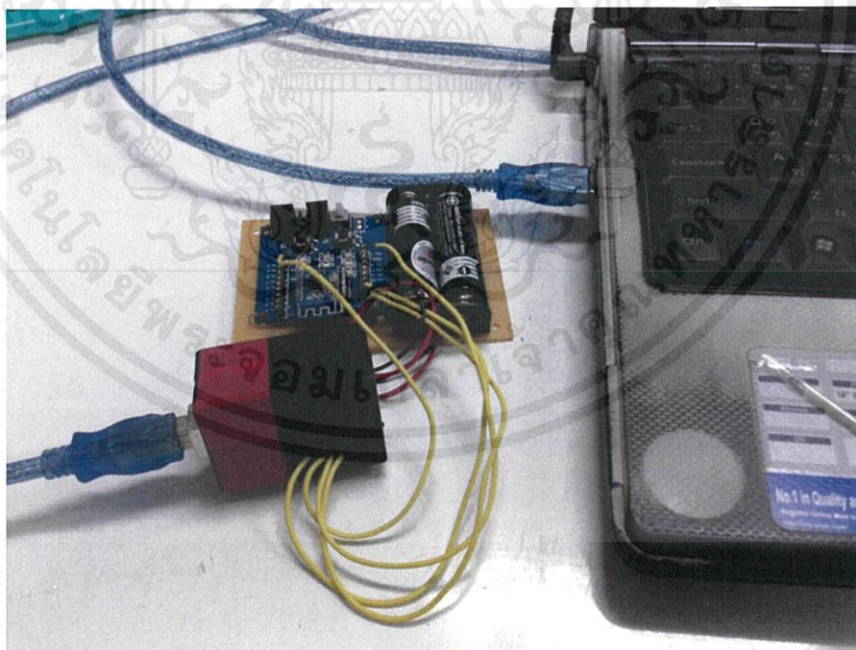
#### 4.1 กล่าวนำ

การทดลอง ซึ่งทดสอบการควบคุมหลอดไฟผ่านไมโครคอนโทรลเลอร์ในลักษณะต่าง ๆ ตามขอบเขตของโครงการที่ได้กำหนดไว้ซึ่งก่อนทำการทดลองจะต้องตรวจเช็คบอร์ดวงจร ทุก ๆ บอร์ดเสียก่อน เพื่อให้สามารถทดลองและได้ผลการทดลองตามที่คาดคิดไว้

#### 4.2 การทดลองการควบคุมหลอดไฟผ่านการสั่งงานจากคอมพิวเตอร์

ทำการทดลองการควบคุมหลอดไฟโดยการสั่งงานจากคอมพิวเตอร์ อุปกรณ์จะทำการส่งสัญญาณไปควบคุมหลอดไฟตามที่ต้องการ โดยการสั่งงานควบคุมการเปิด - ปิด ของหลอดไฟ

โดยการสั่งงานจะเริ่มจากผู้ใช้งาน จากนั้นคำสั่งจะถูกส่งมายังไมโครคอนโทรลเลอร์ที่เป็นตัวมาสเตอร์ ซึ่งเชื่อมตัวผ่านช่องทางของ USB ของคอมพิวเตอร์ไปยังไมโครคอนโทรลเลอร์ โดยใช้ฟังก์ชันคำสั่ง UART ในการรับคำสั่งจากคอมพิวเตอร์ จากนั้นจะส่งคำสั่งต่อไปยังตัวสเลฟ โดยผ่านการส่งข้อมูลด้วยระบบไร้สาย จากนั้นไมโครคอนโทรลเลอร์ที่เป็นตัวสเลฟจะทำการประมวลผลและส่งสัญญาณไปสั่งงานรีเลย์และเปิด - ปิดหลอดไฟตามคำสั่งที่ผู้ใช้งานต้องการ



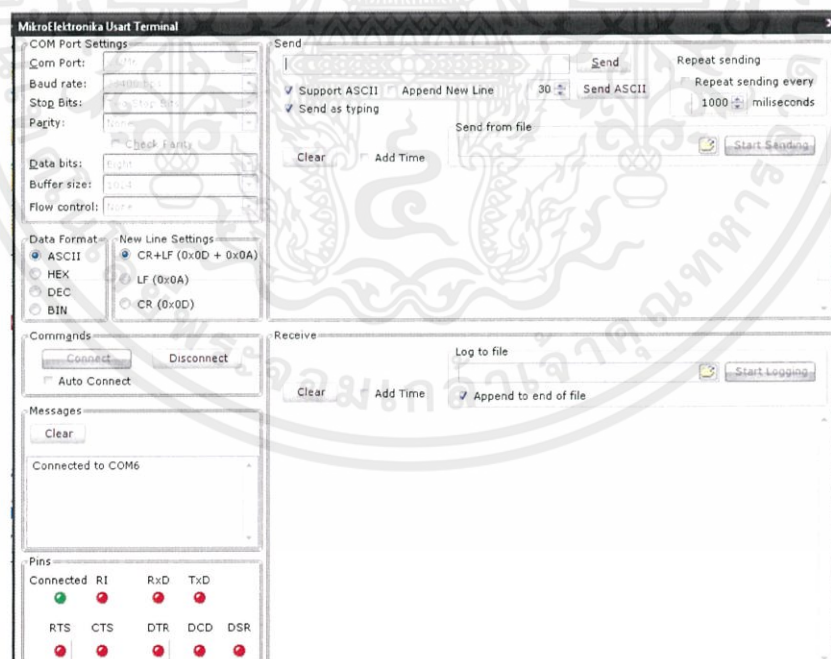
รูปที่ 4.1.1 การสั่งงานไมโครคอนโทรลเลอร์จากคอมพิวเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.1.2 บอร์ดวงจรที่ใช้ในการควบคุมการเปิด - ปิด หลอดไฟ

โดยการสั่งงานผ่านคอมพิวเตอร์โดยตรงนั้นจะต้องอาศัยการทำงานของโปรแกรมที่เรียกว่า Hyper terminal ซึ่งก่อนที่จะสั่งงานผ่าน Hyper Terminal ได้นั้นจะต้องทำการเชื่อมต่อกันก่อนจึงจะสามารถทำงานร่วมกันได้



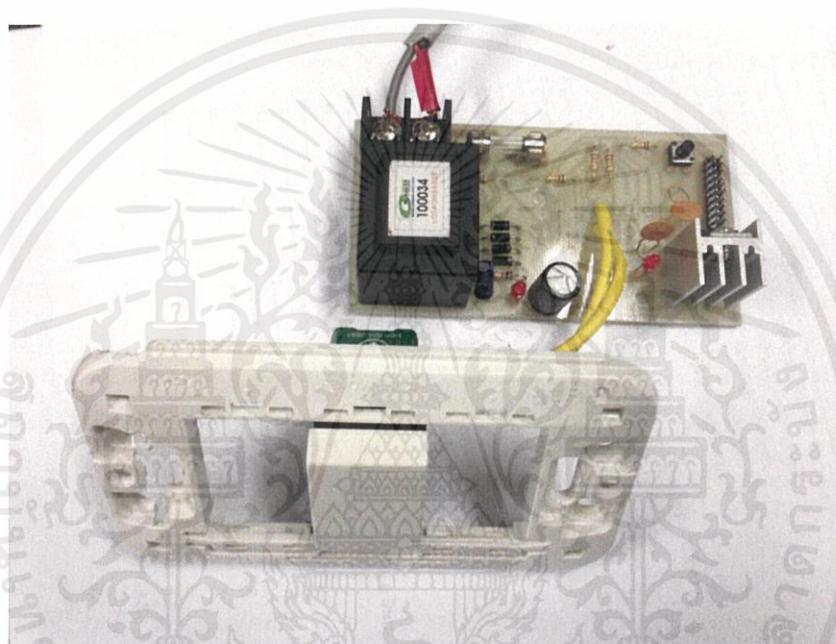
รูปที่ 4.1.3 หน้าต่างโปรแกรม Hyper Terminal

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

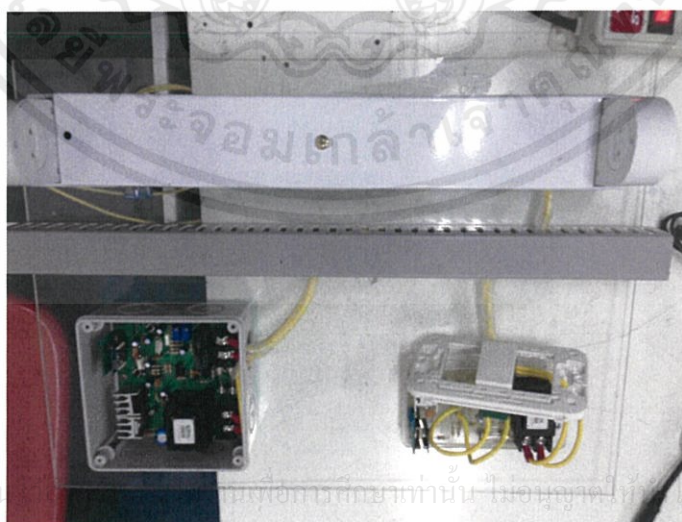
### 4.3 การทดลองการควบคุมหลอดไฟผ่านการสั่งงานบอร์ดสวิตช์

ทำการทดลองการควบคุมหลอดไฟโดยการสั่งงานจากบอร์ดสวิตช์ อุปกรณ์จะทำการส่งสัญญาณไปควบคุมหลอดไฟตามที่ต้องการ โดยการสั่งงานควบคุมการเปิด - ปิด ของหลอดไฟ

ในการทดลองนี้ อุปกรณ์ที่ใช้สั่งงานหลักคือตัวบอร์ดสวิตช์ โดยบอร์ดสวิตช์จะมีไมโครคอนโทรลเลอร์ติดอยู่ ซึ่งจะคอยรับสถานะจากสวิตช์เมื่อทำการเปิด - ปิด ที่สวิตช์ เมื่อมีกดปุ่มที่บอร์ดสวิตช์นั้นจะมีการส่งสัญญาณไปที่ไมโครคอนโทรลเลอร์ตามที่ได้เขียนโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ จากนั้นไมโครคอนโทรลเลอร์จะส่งสัญญาณผ่านเทคโนโลยีแบบไร้สาย เพื่อไปสั่งงานไมโครคอนโทรลเลอร์อีกตัวหนึ่งที่ติดตั้งไว้กับโมดูลที่เชื่อมต่อกับหลอดไฟ เพื่อทำการประมวลผลและส่งสัญญาณไปสั่งงานรีเลย์และเปิด - ปิดหลอดไฟตามคำสั่งที่ผู้ใช้งานต้องการ



รูปที่ 4.2.1 บอร์ดวงจรสวิตช์ที่ใช้สั่งการหลอดไฟ แบบไร้สาย



รูปที่ 4.2.2 อุปกรณ์ที่รับคำสั่งจากคอมพิวเตอร์และเชื่อมต่อกะหลอดไฟ

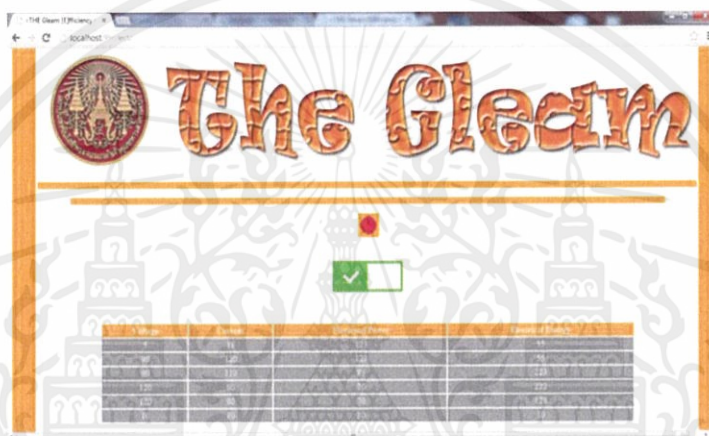
เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ การนำเอกสารนี้ไปใช้โดยไม่ได้รับอนุญาตถือว่าผิดกฎหมาย

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งยังมีให้ตัดแปะคู่มือและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

#### 4.4 การทดลองการควบคุมหลอดไฟผ่านการสั่งงานจากเว็บเบราว์เซอร์

ทำการทดลองการควบคุมหลอดไฟโดยการสั่งงานจากเว็บเบราว์เซอร์ อุปกรณ์จะทำการส่งสัญญาณไปควบคุมหลอดไฟตามที่ต้องการ โดยการสั่งงานควบคุมการเปิด - ปิด ของหลอดไฟ

ในการทดลองนี้จะมีหลักการคล้ายกับการทดลองการควบคุมหลอดไฟผ่านการสั่งงานจากคอมพิวเตอร์ คือ การสั่งงานจะเริ่มจากผู้ใช้งาน จากนั้นคำสั่งจะถูกส่งมายังไมโครคอนโทรลเลอร์ที่เป็นตัวมาสเตอร์ ซึ่งเชื่อมตัวผ่านช่องทางของ USB ของคอมพิวเตอร์ไปยังไมโครคอนโทรลเลอร์ โดยใช้ฟังก์ชันคำสั่ง UART ในการรับคำสั่งจากคอมพิวเตอร์ จากนั้นจะส่งคำสั่งต่อไปยังตัวเสลฟ โดยผ่านการส่งข้อมูลด้วยระบบไร้สาย โดยการสั่งงานนั้นจะเป็นการสั่งงานจากเว็บเบราว์เซอร์แทน โดยชุดข้อมูลคำสั่งที่ส่งลงมาจากเว็บเบราว์เซอร์นั้นจะถูกเขียนโดยใช้ภาษาซีชาร์ป (C#) และใช้โปรแกรม Dreamweaver มาในการสร้างตัวเว็บไซต์นี้



รูปที่ 4.3 แสดงภาพรวมของหน้าเว็บไซต์และข้อมูลที่วัดค่าออกมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## บทที่ 5

# สรุปผลและข้อเสนอแนะ

### 5.1 สรุปผล

ปริญญาานิพนธ์ฉบับนี้ได้นำเสนอแนวคิดในการจัดการและการควบคุมการเปิด - ปิดหลอดไฟแบบไร้สายภายในบ้าน โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมกระบวนการทั้งหมดและใช้งานร่วมกับเทคโนโลยีแบบไร้สาย ให้มีการรับส่งข้อมูล โดยการรับส่งข้อมูลนั้นเป็นรูปแบบมาสเตอร์-สเลฟ อีกทั้งยังสามารถคู่ค่าสถานการณ์ทำงาน ควบคุมการทำงานผ่านหน้าเว็บเบราว์เซอร์ได้

### 5.2 ปัญหาและอุปสรรค

ไมโครคอนโทรลเลอร์ที่ใช้ทำงานของระบบควบคุมนี้คือ nRF24le1 ซึ่งไมโครคอนโทรลเลอร์ตัวนี้ได้จัดอยู่ในตระกูล C-51 และตัวที่ได้เลือกมานั้นมีขนาดเล็ก และตัวอุปกรณ์มีตัวรับสัญญาณในการรับ-ส่งไวร์เลสที่ค่อนข้างเล็ก จึงทำให้เกิดปัญหาและอุปสรรคต่างๆดังนี้

1) เสาสัญญาณนั้นได้มีการออกแบบมาตั้งแต่แรกแล้ว ซึ่งได้มีขนาดเล็ก จึงทำให้เกิดปัญหาและอุปสรรคในการรับ - ส่งสัญญาณ ทำให้มีระยะในการรับ - ส่งสัญญาณที่ค่อนข้างใกล้ประมาณ 3 -5 เมตร จึงทำให้เกิดข้อจำกัดในด้านการรับ- ส่งข้อมูลในระยะไกล

2) ในด้านการประมวลผลของข้อมูล ซึ่งไมโครคอนโทรลเลอร์ที่ได้ทำการเลือกมาใช้เป็นเบอร์ที่มีขนาดเล็ก และการประมวลผลของซีพียูที่ช้า จึงทำให้เกิดการประมวลผลที่ช้าตามไปด้วย และส่งผลในการรับ - ส่งข้อมูลที่ช้าลงอีกด้วย ทั้งสองส่วนนี้จึงทำให้เกิดความล่าช้าในการรับ - ส่งข้อมูลทั้งในตัวอุปกรณ์ไมโครคอนโทรลเลอร์ด้วยกัน และยังส่งผลไปถึงการรับ - ส่งข้อมูลขึ้นไปยังเซิร์ฟเวอร์อีกด้วย

### 5.3 ข้อเสนอแนะ

1) การรับส่งข้อมูลระหว่างตัวอุปกรณ์มาสเตอร์กับอุปกรณ์สเลฟ ไม่สามารถรับส่งได้ไกลเท่าที่ควร ควรจัดการเรื่องเสาสัญญาณไวร์เลสให้มีการรับ - ส่งที่ไกลขึ้น เพื่อความสามารถในการทำงานที่มีประสิทธิภาพมากกว่าเดิม

2) ควรจะทำการเปลี่ยนขนาดเบอร์ของไมโครคอนโทรลเลอร์ให้มีขนาดใหญ่ขึ้น เพื่อให้มีการประมวลผลที่เร็วขึ้น พร้อมทั้งมีพอร์ตให้ใช้ได้มากขึ้น เพื่อความสะดวกและมีความยืดหยุ่นในการทำงานที่มากขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## บรรณานุกรม

- [1] สุธี พงศาสกุลชัย.คัมภีร์ Visual C# 2005. กรุงเทพฯ : เคทีพี คอมพ์ แอนด์ คอนซัลท์, 2549.
- [2] สุรพรรณ เพ็ญจำรัส.เรียนลัด C# และการเขียนโปรแกรม.NET. กรุงเทพฯ : โปรวิชั่น, 2546.
- [3] ชีรวัฒน์ ประกอบผล.คู่มือการเขียนโปรแกรม ภาษา C. กรุงเทพฯ : ชิมพลิฟาย, 2553.
- [4] ศุภชัย สมพานิช.คู่มือการเขียนโปรแกรม Visual C#.NET. นนทบุรี : อินโฟเพรส, 2546.
- [5] ประจัน พลั่งสันติกุล, และชัยวัฒน์ ลิ้มพรจิตร์วิไล. ปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 กับ Keil C51 คอมไพเลอร์. กรุงเทพฯ : อินโนเวทีฟ เอ็กเพอริเมนต์, 2521
- [6] การใช้โปรแกรม Keil uVision 3 เบื้องต้น. 10 กุมภาพันธ์ 2556, จาก <http://www.mcuthailand.com/articles/mcs/Keil.html>
- [7] General-purpose Latching Relay MYK, 2002 : 1-4
- [8] ACS712 Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS
- [9] ADE7763, 2004 : 13, 48-55
- [10] nRF24LE1, 2010 : 1-196

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



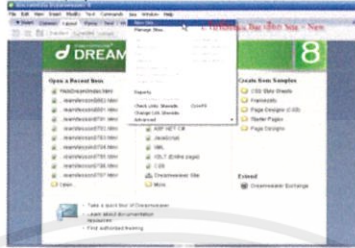
## ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## ขั้นตอนการสร้างไซต์

การสร้างไซต์ ก็เพื่อที่เวลาสร้างเว็บไซต์ ข้อมูลจะได้ถูกจัดอย่างเป็นระเบียบ ไม่กระจัดกระจาย ง่ายต่อการตรวจสอบ โดยมีขั้นตอนดังนี้

1. Click ที่ Tool Bar



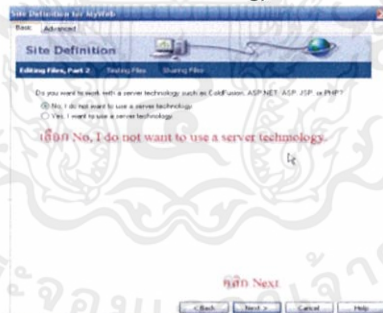
รูปที่ 1 กดแถบ Tool Bar และจะพบคำสั่ง New Site

2. พิมพ์ชื่อ Site ที่ต้องการ โดยเป็นชื่อของ file เว็บไซต์



รูปที่ 2 กรอกชื่อที่ต้องการให้เป็นชื่อของ Site งาน

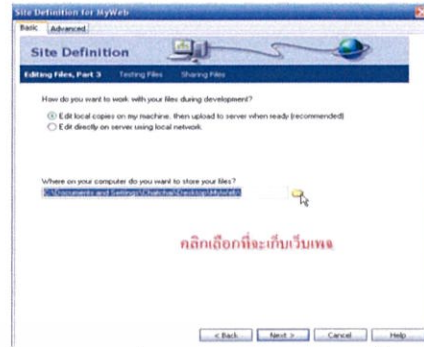
3. เลือกช่องที่ต้องการ “ต้องการใช้ Technology ร่วมกับพวก ASP NET, ASP, PHP”



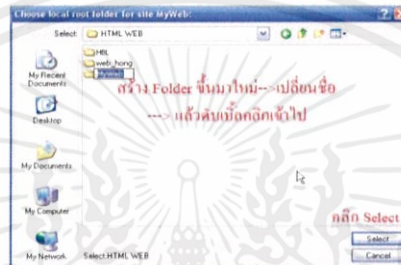
รูปที่ 3 เลือกช่องทางที่ต้องการและกด Next เพื่อไปยังขั้นตอนต่อไป

4. เลือก Folder ที่ต้องการเก็บ File ที่ได้สร้างขึ้น

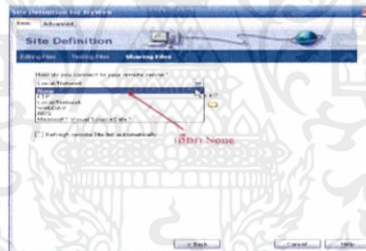
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



- รูปที่ 4 สร้างหรือเลือก Folder ที่ต้องการเก็บไฟล์งานทั้งหมดและทำขั้นต่อไป
5. สร้าง Folder ใหม่เพื่อความสะดวกและลดความสับสน



- รูปที่ 5 กรณีสร้างหรือเลือก Folder ขึ้นมาใหม่และคลิก Select เพื่อดำเนินการต่อ
6. หัวข้อ “ ต้องการเชื่อมต่อกับ Remote Server ” ตอบ None ก่อน

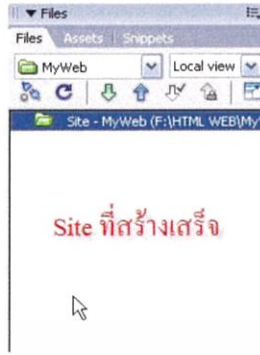


- รูปที่ 6 ตัวเลือกสำหรับการเชื่อมต่อกับ Remote Server
7. เมื่อตั้งค่าเรียบร้อยแล้วจะปรากฏภาพดังรูป (แสดงผลตามที่ตั้งค่า)



- รูปที่ 7 แสดงรายละเอียดในการตั้งค่า
8. เมื่อติดตั้งโปรแกรมเสร็จสมบูรณ์ จะปรากฏดังรูปภาพ

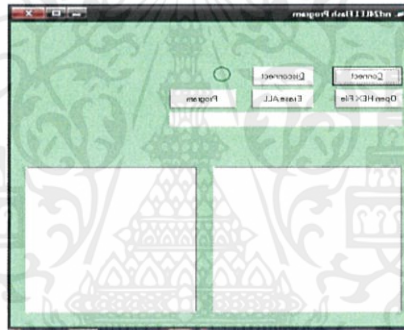
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8 โฟลเดอร์ (Folder) จัดเก็บหน้าเว็บไซต์ที่ได้สร้างไว้

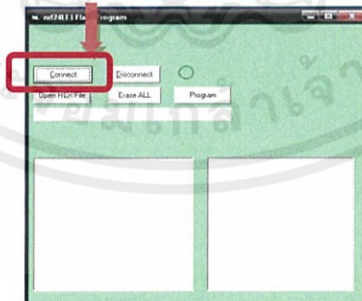
### การดาวน์โหลด Hex File ให้กับ MCU

โปรแกรม nRF24LE1 เป็นโปรแกรมที่ใช้ Download ข้อมูลแบบ Hex File ให้กับ CPU MCS-51 โดยใช้วิธีการแบบ Serial Programming ซึ่งวิธีการใช้งานโปรแกรมนั้นมีขั้นตอนที่ไม่ยากซับซ้อนมากนัก สามารถทำความเข้าใจได้ง่าย โดยในที่นี่จะขอแนะนำให้ทราบถึงวิธีการใช้โปรแกรม nRF24LE1 Flash Program เพื่อใช้งานกับบอร์ด nRF24LE1



รูปที่ 1 โปรแกรม nRF24LE1 Flash Program

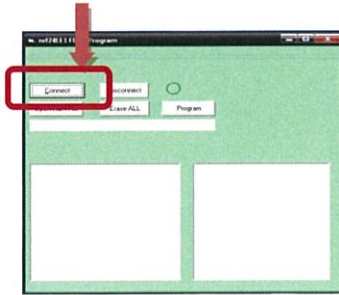
1. เมื่อทำการเชื่อมต่อบอร์ด nRF24LE1 เข้ากับเครื่องคอมพิวเตอร์ จากนั้นเมื่อเปิดโปรแกรม nRF24LE1 เพื่อดาวน์โหลดโปรแกรมให้กับบอร์ด nRF24LE1 จำเป็นที่จะต้องเชื่อมต่อโปรแกรมเข้าบอร์ด nRF24LE1 ทุกครั้ง โดยการกดที่ปุ่ม Connect ที่โปรแกรม nRF24LE1



รูปที่ 2 การเชื่อมต่อโปรแกรม nRF24LE1 Flash Program เข้ากับบอร์ด

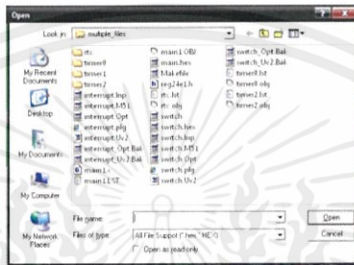
2. เลือก HEX File ที่ต้องการดาวน์โหลดลงในบอร์ด nRF24LE1 โดยไปที่คำสั่ง Open HEX File

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3 ปุ่มคำสั่ง Open HEX File

3. เมื่อกดปุ่ม Open HEX File แล้ว จะขึ้นหน้าต่างให้เลือกที่อยู่ที่ใช้บันทึก HEX File ไว้ จากนั้นเลือก HEX File ที่ต้องการ



รูปที่ 4 หน้าต่างแสดง ที่อยู่ของ HEX File

4. เมื่อเลือก HEX File ที่ต้องการดาวน์โหลดลงในบอร์ดแล้ว จากนั้นสามารถดาวน์โหลด HEX File ลงในบอร์ด โดยกดที่ปุ่ม Program



รูปที่ 5 คำสั่งโปรแกรมเพื่อใช้ในการดาวน์โหลดโปรแกรม

5. โปรแกรม จะแสดงว่า ดาวน์โหลด HEX File ลงในบอร์ดเรียบร้อยแล้ว



รูปที่ 6 ดาวน์โหลด HEX File ลงในบอร์ด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้ภายในห้องปฏิบัติการเท่านั้น มิใช่ให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งหลังจากการโปรแกรมเสร็จเรียบร้อยแล้ว CPU จะเริ่มต้นทำงานตามข้อมูลในโปรแกรมที่ส่ง

Download ให้ทันที

## โปรแกรมมาสเตอร์ไมโครคอนโทรลเลอร์

```

1 #include<reg241e1.h>
2 #include<stdio.h>
3 #include "main.h"
4 #include "nrf_1mb.h"
5 #include "nrf_definmm.h"
6
7 #define NRF_CE RFCE
8 #define NRF_CS RFCSN
9 #define SSPBUF SP1RDAT
10
11 #define Address_Write 0x0E
12
13 unsigned char LOOP, InputData, Check, Reset_Loop, Test, Checksum_TX = 0;
14 unsigned int Database, p;
15 unsigned char Temp, Keep;
16 unsigned int Bx_Flag, Uart_Flag, Decode_in, Decode_out = 0x0000, Send_Decode = 0, x;
17
18 void Delay_us(volatile unsigned int n){
19     unsigned char i;
20     while(n--){
21         for(i=0;i<3;i++){
22             ;
23         }
24     }
25 }
26 void Delay_100us(volatile unsigned int n){
27     unsigned char i;
28     while(n--){
29         for(i=0;i<3;i++){
30             ;
31         }
32     }
33 }
34 void Delay_ms(int i)
35 {
36     int j;
37     for(j=0;j<=i;j++){
38         for(k=0;k<100;k++){
39             ;
40         }
41     }
42 }
43 void Uart_Mode() interrupt 4 using 3
44 {
45     if(RID)
46     {
47         keep = (unsigned int)SOBUF;
48         Database = keep;
49         RIO = 0;
50         LOOP = 1;
51     }
52     else if(TIO)
53     {
54         TIO=0;
55     }
56 }
57 void CE_Pin_Active(unsigned char action) // CE pin high, low or pulse..
58 {
59     switch(action)
60     {
61         case CE_LOW: // action == 0, CE low
62             nRF_CE = 0;
63             break;
64         case CE_HIGH: // action == 1, CE high
65             nRF_CE = 1;
66             break;
67         case CE_PULSE: // action == 2, CE pulse (10?us)
68             // TRIG = 0;
69             nRF_CE = 1; // Set CE pin high
70             Delay_us(100);
71             nRF_CE = 0; // Set CE pin low
72             // TRIGON_bit = 1; // Start Times2, CE pulse timer
73             break;
74     }
75 }
76 unsigned char L01_Clear_IRQ(unsigned char irq_Flag)
77 {
78     return nRF_SPI_RW_Reg(WRITE_REG + STATUS, irq_Flag);
79 }
80 unsigned char nRF_SPI_RW(unsigned char byte1)
81 {
82     unsigned char status;
83     //Delay_us(10);
84     SSPBUF = byte1;
85     Delay_us(1); // Delay 300ns ok
86     status = SSPBUF;
87     return(status);
88 }
89 unsigned char nRF_SPI_RW_Reg(unsigned char reg, unsigned char value)
90 {
91     unsigned char status;
92     nRF_CS = 0; // CSN low, init SPI transaction
93     status = nRF_SPI_RW(reg); // select register
94     nRF_SPI_RW(value); // ..and write value to it..
95     nRF_CS = 1; // CSN high again
96     return(status); // return NRF24L01 status byte
97 }
98 unsigned char nRF_SPI_Write_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
99 {
100     unsigned char status, byte_ctr;
101     nRF_CS = 0; // Set CSN low, init SPI transaction
102     status = nRF_SPI_RW(reg); // select register to write to and read status byte
103     for(byte_ctr=0; byte_ctr<bytes; byte_ctr++){ // then write all byte in buffer(*pBuf)
104         nRF_SPI_RW(*pBuf++);
105     }
106     nRF_CS = 1; // Set CSN high again
107     return(status); // return NRF24L01 status byte
108 }
109 unsigned char L01_RD_RX_PU_n(unsigned char pipe)
110 {
111     return nRF_SPI_Read(RX_PU_PD + pipe);
112 }
113 void L01_Write_TX_Pload(unsigned char *pBuf, unsigned char piWidth)
114 {
115     nRF_SPI_Write_Buf(WR_TX_LOAD, pBuf, piWidth);
116 }
117 unsigned char Send_Packet() // Send one data packet, controlled by Button1(SW1)
118 {
119     L01_Write_TX_Pload(&TX_Pload, TX_LOAD_WIDTH); // Write new TX payload
120     CE_Pin_Active(CE_PULSE); // Enable CE pulse, 12?us
121     //wTey_Ctr = 0; // Reset Tey_Ctr before nex transmitt..
122     return 1;
123 }

```

เอกสารนี้เป็นลิขสิทธิ์งานของฉัน อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

132 void LO1_Flush_RX(void)
133 {
134     nRF_SPI_RW_Reg(FLUSH_RX, 0);
135 }
136 void LO1_Flush_TX(void)
137 {
138     nRF_SPI_RW_Reg(FLUSH_TX, 0);
139 }
140 unsigned char LO1_Get_FIFO(void)
141 {
142     return nRF_SPI_Read(FIFO_STATUS);
143 }
144 unsigned char nRF_SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
145 {
146     unsigned char status, byte_ctr;
147     nRF_CS = 0; // Set CSN low, init SPI transaction
148     status = nRF_SPI_RW_Reg(reg); // Select register to read from.
149     for(byte_ctr=0;byte_ctr<bytes;byte_ctr++) // Perform SPI_RW to read byte from nRF24L01
150     {
151         pBuf(byte_ctr) = nRF_SPI_RW(0);
152     }
153     nRF_CS = 1; // Set CSN high again
154     return(status); // return nRF24L01 status byte
155 }
156 unsigned char nRF_SPI_Read(unsigned char reg)
157 {
158     unsigned char reg_val;
159     nRF_CS = 0; // CSN low, initialize SPI communication...
160     nRF_SPI_RW_Reg(reg); // Select register to read from.
161     reg_val = nRF_SPI_RW(0); // ..then read register value
162     nRF_CS = 1; // CSN high, terminate SPI communication
163     return(reg_val); // return register value
164 }
165 unsigned char LO1_Get_Status(void)
166 {
167     return nRF_SPI_Read(STATUS);
168 }
169 unsigned char LO1_Get_Current_Pipenum(void)
170 {
171     return ((LO1_Get_Status() & RX_P_NO) >> 1);
172 }
173 unsigned int LO1_Read_RX_Pload(unsigned char *pBuf)
174 {
175     unsigned char pWidth, pipe;
176     pWidth = LO1_RD_RX_PW_n(pipe = LO1_Get_Current_Pipenum()); // Read current pipe's payload width
177     nRF_SPI_Read_Buf(RD_RX_PLOAD, pBuf, pWidth); // Then get RX data
178     return ((pipe << 8) + pWidth); // return pipe# & pipe#.pWidth
179 }
180 /* unsigned char nRF_SPI_RW_Reg_IRQ(unsigned char reg, unsigned char value)
181 {
182     unsigned char status;
183     nRF_CS = 0; // CSN low, init SPI transaction
184     SSPIBUF = reg; // Enable Auto.Ack:Pipes 0-5
185     //while (SSPISTAT.BF == 0) // Enable Pipes 0-5
186     Delay_us(10); // 500µs + 867µs, 10 retrans...
187     status = SSPIBUF; // free the register // Select RF channel 40
188     SSPIBUF = value; // Select Address
189     //while (SSPISTAT.BF == 0) // TX_PUR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
190     Delay_us(10); // Set PUR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
191     reg = SSPIBUF; // free the register
192     nRF_CS = 1; // CSN high again
193     return(status); // return nRF24L01 status byte
194 } */
195 void Tx_Mode ()
196 {
197     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
198     Delay_ms(200);
199     LO1_Flush_TX();
200     LO1_Flush_RX();
201     //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
202     nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS); // nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
203     //ucIRQ_Source = CLEAR; // ucIRQ_Source = CLEAR;
204     //ucLastStat = ucLinkStat = LINK_ESTABLISH; // LINK_ESTABLISH = 0x02
205     nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f); // Enable Auto.Ack:Pipes 0-5
206     nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f); // Enable Pipes 0-5
207     nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a); // 500µs + 867µs, 10 retrans...
208     nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
209     nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03); // Select Address
210     nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08); // TX_PUR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
211     nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PUR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
212     nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Writes TX_Address to nRF24L01
213     nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // RX_Addr0 same as TX_Adr for Auto.Ack
214     CE_Pin_Active(CE_HIGH);
215 }
216 /* void Tx_Mode_runtime ()
217 {
218     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
219     //Delay_ms(200);
220     LO1_Flush_TX();
221     LO1_Flush_RX();
222     //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
223     //nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS); // nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
224     //ucIRQ_Source = CLEAR; // ucIRQ_Source = CLEAR;
225     //ucLastStat = ucLinkStat = LINK_ESTABLISH; // LINK_ESTABLISH = 0x02
226     //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f); // Enable Auto.Ack:Pipes 0-5
227     //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f); // Enable Pipes 0-5
228     //nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a); // 500µs + 867µs, 10 retrans...
229     //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
230     //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08); // TX_PUR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
231     //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PUR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
232     //nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Writes TX_Address to nRF24L01
233     //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // RX_Addr0 same as TX_Adr for Auto.Ack
234     CE_Pin_Active(CE_HIGH);
235     Delay_ms(50); //max 1 ms
236     //CE_Pin_Active(CE_LOW);
237 } */
238 void Rx_Mode ()
239 {
240     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
241     Delay_ms(1); // 200ms
242     LO1_Flush_TX();
243     LO1_Flush_RX();
244     //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษานานับ ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
 ไม่ว่ากล่าวถึงผู้เขียนหรือผู้จัดทำแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

264 nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack!Pipes 0-5
265 nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3F); // Enable Pipes 0-5
266 nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
267 nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03); // Select Address
268 nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x02); // TX_PWR:0dBm, DataRate:2Mbps, LNA:HCURR
269 nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0F); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX, RX_DR enabled..
270
271 nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Use the same address on the RX device as the TX devi
272 nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));
273
274 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
275 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
276 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
277 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);
278
279
280 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_PO, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
281 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_PLOAD_WIDTH);
282 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_PLOAD_WIDTH);
283 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_PLOAD_WIDTH);
284 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_PLOAD_WIDTH);
285 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_PLOAD_WIDTH);
286
287 CE_Pin_Active(CE_HIGH);
288
289
290 void Rx_Mode_Runtime()
291 {
292 //CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
293 //Delay_ms(1); // 200ms
294 //LO1_Flush_TX();
295 //LO1_Flush_RX();
296
297 //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
298
299 //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack!Pipes 0-5
300 //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3F); // Enable Pipes 0-5
301 //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
302 //nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03); // Select Address
303 //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x02); // TX_PWR:0dBm, DataRate:2Mbps, LNA:HCURR
304 //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x02); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX, RX_DR enabled..
305
306 //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Use the same address on the RX device as the TX de
307 //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));
308
309 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
310 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
311 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
312 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);
313
314 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_PO, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
315 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_PLOAD_WIDTH);
316 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_PLOAD_WIDTH);
317 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_PLOAD_WIDTH);
318 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_PLOAD_WIDTH);
319 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_PLOAD_WIDTH);
320
321 CE_Pin_Active(CE_HIGH);
322 Delay_ms(30);
323
324 void Rx_Real_Mode()
325 {
326 Rx_Mode_Runtime();
327 do
328 {
329 //POD = ~POD;
330 u1RX_info = LO1_Read_RX_Pload(RX_pload); // Read current payload
331 // temp = LO1_Get_FIFO() & 0x02; temp >> 1;
332
333 }
334 while(! (LO1_Get_FIFO() & RX_EMPTY)); // ..until FIFO empty
335 //while( temp == 0 );
336 temp = LO1_Get_FIFO() & 0x01; // temp >> 1;
337
338 //P14 = temp;
339
340 //u1RX_info = LO1_Read_RX_Pload(RX_pload);
341 LO1_Clear_IRQ(MASK_RX_DR_FLAG);
342
343 /*if (RX_pload[0] == 0x31) ( POD = ~POD; RX_pload[0] = 0; )
344 if (RX_pload[1] == 0x32) ( PO1 = ~PO1; RX_pload[1] = 0; )
345 if (RX_pload[2] == 0x33) ( PO2 = ~PO2; RX_pload[2] = 0; )
346 if (RX_pload[3] == 0x34) ( PO3 = ~PO3; RX_pload[3] = 0; )
347 if (RX_pload[4] == 0x35) ( PO4 = ~PO4; RX_pload[4] = 0; ) */
348 CE_Pin_Active(CE_LOW);
349
350 void Tx_Real_Mode()
351 {
352 Send_Packet();
353 F13 = ~F13;
354 Delay_ms(1);
355 LO1_Clear_IRQ(MASK_TX_DS_FLAG);
356
357 /*void Decode_output()
358 {
359 int DO;
360 unsigned char Data_All;
361 DO = (3*(((Address_Route>>4) & 9)+1));
362 if ( Send_Decode > Data_All )
363 {
364 Send_Decode = 0;
365
366 switch ( Send_Decode )
367 {
368 case 0:
369 break;
370 }
371 Send_Decode++;
372 */
373 void Decode_input()
374 {
375 int DI;
376 DI = (3*(((Address_Route>>4) & 9)+1));
377
378 switch ( (RX_pload[DI] & 0x20) >> 4 )
379 {
380 case 0:
381 /*SOBUF = RX_pload[2];
382 while(TIO)
383 {
384 TIO = 0;
385 }
386 break;
387 case 1:
388 break;
389 }
390
391 */

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าในรูปแบบใดก็ตาม หากมีให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

391 void Check_Device()
392 {
393     /* int CD, set, pload;
394     CD = ((Address_Route)>>4) & 9;
395     if(CD == 0)
396     {
397         Rx_Real_Mode();
398         for(set=3;set<=31;set++)
399             TX_pload[set] = RX_pload[set];
400
401         for(pload=0;pload<=30;pload++)
402         {
403             if((pload & 3) == 0)
404                 TX_pload[pload] = TX_pload[pload] & 0xF0;
405         }
406         TX_pload[0] = Address_Route;
407     }
408     else if(CD == 1)
409     {
410         Rx_Real_Mode();
411         for(set=0;set<=2;set++)
412             TX_pload[set] = RX_pload[set];
413
414         for(set=6;set<=31;set++)
415             TX_pload[set] = RX_pload[set];
416
417         for(pload=0;pload<=30;pload++)
418         {
419             if((pload & 3) == 0)
420                 TX_pload[pload] = TX_pload[pload] & 0xF0;
421         }
422         TX_pload[3] = Address_Route;
423     }
424     else if(CD == 2)
425     {
426         Rx_Real_Mode();
427
428         for(set=0;set<=5;set++)
429             TX_pload[set] = RX_pload[set];
430
431         for(set=9;set<=31;set++)
432             TX_pload[set] = RX_pload[set];
433
434         for(pload=0;pload<=30;pload++)
435         {
436             if((pload & 3) == 0)
437                 TX_pload[pload] = TX_pload[pload] & 0xF0;
438         }
439         TX_pload[6] = Address_Route;
440     }
441     else if(CD == 3)
442     {
443         Rx_Real_Mode();
444
445         for(set=0;set<=8;set++)
446             TX_pload[set] = RX_pload[set];
447
448         for(set=12;set<=31;set++)
449             TX_pload[set] = RX_pload[set];
450
451         for(pload=0;pload<=30;pload++)
452         {
453             if((pload & 3) == 0)
454                 TX_pload[pload] = TX_pload[pload] & 0xF0;
455         }
456         TX_pload[9] = Address_Route;
457     }
458     else if(CD == 4)
459     {
460         Rx_Real_Mode();
461
462         for(set=0;set<=11;set++)
463             TX_pload[set] = RX_pload[set];
464
465         for(set=15;set<=31;set++)
466             TX_pload[set] = RX_pload[set];
467
468         for(pload=0;pload<=30;pload++)
469         {
470             if((pload & 3) == 0)
471                 TX_pload[pload] = TX_pload[pload] & 0xF0;
472         }
473         TX_pload[12] = Address_Route;
474     }
475     else if(CD == 5)
476     {
477         Rx_Real_Mode();
478
479         for(set=0;set<=14;set++)
480             TX_pload[set] = RX_pload[set];
481
482         for(set=18;set<=31;set++)
483             TX_pload[set] = RX_pload[set];
484
485         for(pload=0;pload<=30;pload++)
486         {
487             if((pload & 3) == 0)
488                 TX_pload[pload] = TX_pload[pload] & 0xF0;
489         }
490         TX_pload[15] = Address_Route;
491     }
492     else if(CD == 6)
493     {
494         Rx_Real_Mode();
495
496         for(set=0;set<=17;set++)
497             TX_pload[set] = RX_pload[set];
498
499         for(set=21;set<=31;set++)
500             TX_pload[set] = RX_pload[set];
501
502         for(pload=0;pload<=30;pload++)
503         {
504             if((pload & 3) == 0)
505                 TX_pload[pload] = TX_pload[pload] & 0xF0;
506         }
507         TX_pload[18] = Address_Route;
508     }
509     else if(CD == 7)
510     {
511         Rx_Real_Mode();
512
513         for(set=0;set<=20;set++)
514             TX_pload[set] = RX_pload[set];
515
516         for(set=24;set<=31;set++)
517             TX_pload[set] = RX_pload[set];
518
519         for(pload=0;pload<=30;pload++)
520         {
521             if((pload & 3) == 0)
522                 TX_pload[pload] = TX_pload[pload] & 0xF0;
523         }
524         TX_pload[21] = Address_Route;
525     }
526 }

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อการเรียนการสอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าในรูปแบบใดก็ตาม หากมีข้อผิดพลาดหรือต้องการแจ้งถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



```

663 void RF_Mode() interrupt 9 using 1
664 {
665     RF = 0;
666     PO7 = ~PO7;
667     Rx_Flag = 1;
668     RF = 1;
669 }
670 void main()
671 {
672     PDIR = 0x10;
673     PIDIR = 0x00;
674
675     SOCON = 0x50;           //Mode 1 8-bit UART.
676     SOBUF = 0x00;
677     SORELH = 0x03;
678     SORELL = 0xF3;
679     ADCON = 0x80;
680
681     PCON = PCON| 0x80;
682
683
684     nRF_CE = 0;
685     nRF_CS = 1;
686
687     RPKEN = 1;
688     Delay_ms(500);
689
690     CE_Pin_Active(CE_LOW);           // Set CE pin low to enable standby mode
691     Delay_ms(200);
692     LOI_Flush_TX();
693     LOI_Flush_RX();
694
695     /*for(p=0;p<32;p++)
696     {
697         SOBUF = 'A';
698         Delay_ms(1);
699         while(TIO);
700         TIO = 0;
701     }
702     */
703     //printf("Heello");
704     Delay_ms(500);
705
706     Rx_Mode();
707     RF = 1;
708     ESO = 0;
709     RIO = 0;
710     TIO = 1;
711     EA = 1;
712
713
714
715
716 while(1)
717 {
718     if(RIO == 1)
719     {
720         P12 = ~P12;
721         keep = (unsigned int)SOBUF;
722         Datanet = keep;
723         RIO = 0;
724         LOOP = 1;
725         // printf("%c\r\n",keep);
726     }
727
728     if(LOOP == 1)
729     {
730         PO6 = ~PO6;
731
732
733         //Decode_output();
734         TX_pload[0] = 0x00;
735         TX_pload[1] = 0x00;
736         TX_pload[2] = Datanet;
737         TX_pload[3] = 0x10;
738         for (p=4;p<32;p++)
739         {
740             TX_pload[p] = 0x00;
741         }
742
743         TX_pload[30] = 0xff;
744
745         Tx_Mode();
746         Tx_Real_Mode();
747         PO1 = ~PO1;
748
749         //PO6 = ~PO6;
750         /* for (p=0;p<32;p++)
751         {
752             printf("%c",TX_pload[p]);
753         }
754         */
755         LOOP = 0;
756         Delay_ms(100);
757     }
758
759     if(Rx_Flag == 1)
760     {
761         PO2 = ~PO2;
762         Check_Device();
763         Read_Data();
764         //CE_Pin_Active(CE_HIGH);
765         Rx_Flag = 0;
766
767         //for(p=0;p<32;p++)
768         //(
769         printf("%c",RX_pload[2]);
770         //)
771
772         Rx_Mode();
773
774         Delay_ms(10);
775     }
776 }
777
778
779

```

เอกสารนี้เป็นเอกสารสงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าในรูปแบบใดก็ตาม ยกเว้นแต่กรณีที่มีการขออนุญาตและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## โปรแกรมสวิตช์ไมโครคอนโทรลเลอร์

```

1 #include<reg51.h>
2 #include<stdio.h>
3 #include "main.h"
4 #include "nrf_lab.h"
5 #include "nrf_defines.h"
6
7 #define NRF_CE RFCE
8 #define NRF_CS RFCSN
9 #define SSPBUF SPIDAT
10
11 #define Address_Route 0x10
12
13 unsigned char LOOP, inputdata, Check, Reset_Loop, Test ,Checksum_Tx = 0,Do,onetime;
14 unsigned int Databaset,P;
15 unsigned char temp;
16 unsigned int Rx_flag ,Wart_flag , Decode_in, Decode_out = 0x0000, Send_Decode = 0,SW;
17
18 void Delay_us(volatile unsigned int n){
19     unsigned char i;
20     while(n--){
21         for(i=0;i<3;i++){
22             ;
23         }
24     }
25 }
26 void Delay_100us(volatile unsigned int n){
27     unsigned char i;
28     while(n--){
29         for(i=0;i<33;i++){
30             ;
31         }
32     }
33 }
34 void Delay_ms(int i)
35 {
36     int j;
37     for(j =0; j<100; j++){
38         ;
39     }
40 }
41 void Int() interrupt 0 using 2
42 {
43     if (P12 == 1 && SW == 0x62)
44     {
45         SW = 0x61;
46         Do = 1;
47         onetime = 1;
48     }
49     if (P12 == 0 && SW == 0x61)
50     {
51         SW = 0x62;
52         Do = 1;
53         onetime = 1;
54     }
55 }
56 void CE_Pin_Active(unsigned char action) // CE pin high, low or pulse..
57 {
58     switch(action)
59     {
60         case CE_LOW: // action == 0, CE low
61             nRF_CE = 0;
62             break;
63         case CE_HIGH: // action == 1, CE high
64             nRF_CE = 1;
65             break;
66         case CE_PULSE: // action == 2, CE pulse (107s)
67             // TRIP = 0;
68             nRF_CE = 1; // Set CE pin high
69             Delay_us(100);
70             nRF_CE = 0;
71             //TRIPON_bit = 1; // Start Timer, CE pulse timer
72             break;
73     }
74 }
75 unsigned char LO1_Clear_IRQ(unsigned char irq_flag)
76 {
77     return nRF_SPI_RW_Reg(WRITE_REG + STATUS, irq_flag);
78 }
79 unsigned char nRF_SPI_RW(unsigned char byte1)
80 {
81     unsigned char status;
82     //Delay_us(10);
83     SSPBUF = byte1;
84     Delay_us(1); // Delay 300s ok
85     status = SSPBUF;
86     return(status);
87 }
88 unsigned char nRF_SPI_RW_Reg(unsigned char Reg, unsigned char value)
89 {
90     unsigned char status;
91     nRF_CS = 0; // CSN low, init SPI transaction
92     status = nRF_SPI_RW(Reg); // select Register
93     nRF_SPI_RW(value); // ..and write value to it..
94     nRF_CS = 1; // CSN high again
95     return(status); // return nRF24L01 status byte
96 }
97 unsigned char nRF_SPI_Write_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
98 {
99     unsigned char status, byte_ctr;
100     nRF_CS = 0; // Set CSN low, init SPI transaction
101     status = nRF_SPI_RW(reg); // Select register to write to and read status byte
102     for(byte_ctr=0; byte_ctr<bytes; byte_ctr++){ // then write all byte in buffer(*pBuf)
103         nRF_SPI_RW(*pBuf++);
104     }
105     nRF_CS = 1; // Set CSN high again
106     return(status); // return nRF24L01 status byte
107 }
108 unsigned char LO1_RD_RX_PU_n(unsigned char pipe)
109 {
110     return nRF_SPI_Read(RX_PU_PO + pipe);
111 }
112 void LO1_Write_TX_Pload(unsigned char *pBuf, unsigned char piWidth)
113 {
114     nRF_SPI_Write_Buf(WR_TX_LOAD, pBuf, piWidth);
115 }
116 unsigned char Send_Packet() // Send one data packet, controlled by Button1(SW1)
117 {
118     LO1_Write_TX_Pload(&TX_Pload, TX_LOAD_WIDTH); // Write new TX payload
119     CE_Pin_Active(CE_PULSE); // Enable CE pulse, 127s
120     //nRF_Ctr = 0; // Reset Try_Ctr before nex transmit..
121     return 1;
122 }

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อการใช้งานเฉพาะบุคคลเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าในรูปแบบใดก็ตาม หากท่านมีให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

132 void LOI_Flush_RX(void)
133 {
134     nRF_SPI_RW_Reg(FLUSH_RX,0);
135 }
136 void LOI_Flush_TX(void)
137 {
138     nRF_SPI_RW_Reg(FLUSH_TX,0);
139 }
140 unsigned char LOI_Get_FIFO(void)
141 {
142     return nRF_SPI_Read(FIFO_STATUS);
143 }
144 unsigned char nRF_SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
145 {
146     unsigned char status, byte_ctr;
147
148     nRF_CS = 0; // Set CSN low, init SPI transaction
149     status = nRF_SPI_RW(reg); // Select register to write to and read status byte
150
151     for(byte_ctr=0;byte_ctr<bytes;byte_ctr++) // Perform SPI_RW to read byte from nRF24L01
152     {
153         pBuf[byte_ctr] = nRF_SPI_RW(0);
154     }
155
156     nRF_CS = 1; // Set CSN high again
157
158     return(status); // return nRF24L01 status byte
159 }
160 unsigned char nRF_SPI_Read(unsigned char reg)
161 {
162     unsigned char reg_val;
163
164     nRF_CS = 0; // CSN low, initialize SPI communication...
165     nRF_SPI_RW(reg); // Select register to read from..
166     reg_val = nRF_SPI_RW(0); // ..then read register value
167     nRF_CS = 1; // CSN high, terminate SPI communication
168
169     return(reg_val); // return register value
170 }
171 unsigned char LOI_Get_Status(void)
172 {
173     return nRF_SPI_Read(STATUS);
174 }
175 unsigned char LOI_Get_Current_Pipenum(void)
176 {
177     return ((LOI_Get_Status() < RX_P_NO) >> 1);
178 }
179 unsigned int LOI_Read_RX_Pload(unsigned char *pBuf)
180 {
181     unsigned char pWidth, pipe;
182     pWidth = LOI_RD_RX_PU_n(pipe = LOI_Get_Current_Pipenum()); // Read current pipe's payload width
183     nRF_SPI_Read_Buf(RD_RX_PLOAD, pBuf, pWidth); // Then get RX data
184
185     return ((pipe << 8) + pWidth); // return pipe# & pipe#.pWidth
186 }
187 /*unsigned char nRF_SPI_RW_Reg_IRQ(unsigned char reg, unsigned char value)
188 {
189     unsigned char status;
190
191     nRF_CS = 0; // CSN low, init SPI transaction
192
193     SSPIBUF = reg;
194     //while (SSPSTAT.BF == 0)
195     Delay_us(10);
196     status = SSPIBUF; // free the register
197
198     SSPIBUF = value;
199     //while (SSPSTAT.BF == 0)
200     Delay_us(10);
201     reg = SSPIBUF; // free the register
202
203     nRF_CS = 1; // CSN high again
204
205     return(status); // return nRF24L01 status byte
206 }*/
207 void Tx_Mode ()
208 {
209     CE_Pin_Active(CE_LOW); // Set CE pin low to enable stanby mode
210     Delay_ms(200);
211     LOI_Flush_TX();
212     LOI_Flush_RX();
213     //LOI_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
214     nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
215     //ucIRQ_Source = CLEAR;
216     //ucLastStat = ucLinkStat = LINK_ESTABLISH; // LINK_ESTABLISH = 0x02
217
218     nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x32); // Enable Auto.Ack:Pipes 0-5
219     nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x32); // Enable Pipes 0-5
220     nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a); // 5007s + 867s, 10 retrans...
221     nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
222     nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03); // Select Address
223     nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08); // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
224     nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
225
226     nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Writes TX Address to nRF24L01
227     nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // RX_Addr0 same as TX_Adr for Auto.Ack
228     CE_Pin_Active(CE_HIGH);
229 }
230
231 /*void Tx_Mode_runtime()
232 {
233     CE_Pin_Active(CE_LOW); // Set CE pin low to enable stanby mode
234     //Delay_ms(200);
235     LOI_Flush_TX();
236     LOI_Flush_RX();
237     //LOI_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
238     //nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
239     //ucIRQ_Source = CLEAR;
240     //ucLastStat = ucLinkStat = LINK_ESTABLISH; // LINK_ESTABLISH = 0x02
241
242     //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x32); // Enable Auto.Ack:Pipes 0-5
243     //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x32); // Enable Pipes 0-5
244     //nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a); // 5007s + 867s, 10 retrans...
245     //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
246
247     //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08); // TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
248     //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
249
250     //nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Writes TX Address to nRF24L01
251     //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // RX_Addr0 same as TX_Adr for Auto.Ack
252     CE_Pin_Active(CE_HIGH);
253     Delay_ms(50);
254     //CE_Pin_Active(CE_LOW); //max 1 ms
255 }*/
256 void Rx_Mode ()
257 {
258     CE_Pin_Active(CE_LOW); // Set CE pin low to enable stanby mode
259     Delay_ms(1); // 200ms
260     LOI_Flush_TX();
261     LOI_Flush_RX();
262
263     //LOI_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าในรูปแบบใดๆก็ตาม หากท่านมีให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

264
265 nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack:Pipes 0-5
266 nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3F); // Enable Pipes 0-5
267 nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
268 nRF_SPI_RW_Reg(WRITE_REG + SETUP_AV, 0x03); // Select Address
269 nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x02); // TX_PU:0dBm, DataRate:2Mbps, LNA:HCURR
270 nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x02); // Set PUR_UP bit, enable CRC (2 bytes) & Prim:RX, RX_DR enabled..
271
272 nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Use the same address on the RX device as the
273 nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));
274
275 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
276 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
277 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
278 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);
279
280 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_PO, RX_LOAD_WIDTH); // Select same RX payload width as TX Payload width
281 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_LOAD_WIDTH);
282 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_LOAD_WIDTH);
283 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_LOAD_WIDTH);
284 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_LOAD_WIDTH);
285 nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_LOAD_WIDTH);
286
287 CE_Pin_Active(CE_HIGH);
288
289 }
290 void Rx_Mode_Runtime()
291 {
292 //CE_Pin_Active(CE_LOW); // Set CE pin low to enable stanby mode
293 //Delay_ms(1); // 200ms
294 //LO1_Flush_TX();
295 //LO1_Flush_RX();
296
297 //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
298
299 //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack:Pipes 0-5
300 //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3F); // Enable Pipes 0-5
301 //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
302 //nRF_SPI_RW_Reg(WRITE_REG + SETUP_AV, 0x03); // Select Address
303 //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x02); // TX_PU:0dBm, DataRate:2Mbps, LNA:HCURR
304 //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x02); // Set PUR_UP bit, enable CRC (2 bytes) & Prim:RX, RX_DR enabled..
305
306 //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Use the same address on the RX device as
307 //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));
308
309 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
310 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
311 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
312 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);
313
314 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_PO, RX_LOAD_WIDTH); // Select same RX payload width as TX Payload width
315 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_LOAD_WIDTH);
316 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_LOAD_WIDTH);
317 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_LOAD_WIDTH);
318 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_LOAD_WIDTH);
319 //nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_LOAD_WIDTH);
320
321 CE_Pin_Active(CE_HIGH);
322 Delay_ms(30);
323 }
324 void Rx_Real_Mode()
325 {
326 Rx_Mode_Runtime();
327 do
328 {
329 //PO0 = ~PO0;
330 u1RX_inFo = LO1_Read_RX_Pload(RX_pload); // Read current payload
331 // temp = LO1_Get_FIFO() & 0x02; temp >> 1;
332
333 }
334 while(!(LO1_Get_FIFO() & RX_EMPTY)); // ..until FIFO empty
335 //while( temp == 0 );
336 temp = LO1_Get_FIFO() & 0x01; // temp >> 1;
337
338 //P14 = temp;
339
340 //u1RX_inFo = LO1_Read_RX_Pload(RX_pload);
341 LO1_Clear_IRQ(MASK_RX_DR_FLAG);
342
343 /*if (RX_pload[0] == 0x31) { PO0 = ~PO0; RX_pload[0] = 0; }
344 if (RX_pload[1] == 0x32) { PO1 = ~PO1; RX_pload[1] = 0; }
345 if (RX_pload[2] == 0x33) { PO2 = ~PO2; RX_pload[2] = 0; }
346 if (RX_pload[3] == 0x34) { PO3 = ~PO3; RX_pload[3] = 0; }
347 if (RX_pload[4] == 0x35) { PO4 = ~PO4; RX_pload[4] = 0; }*/
348 CE_Pin_Active(CE_LOW);
349 }
350 void Tx_Real_Mode()
351 {
352 Send_Packet();
353 P13 = ~P13;
354 Delay_ms(1);
355 LO1_Clear_IRQ(MASK_TX_DS_FLAG);
356
357 /*void Decode_output()
358 {
359 for
360 int DO;
361 unsigned char Data_All;
362 DO = (3*(((Address_Route>>4) & 9))+1);
363 if ( Send_Decode > Data_All )
364 {
365 Send_Decode = 0;
366 }
367 switch ( Send_Decode )
368 {
369 case 0:
370 break;
371 }
372 Send_Decode++;
373 }*/
374 void Decode_input()
375 {
376 int DI;
377 DI = (3*(((Address_Route>>4) & 9))+1);
378 switch ( (RX_pload[DI] & 0xF0) >> 4 )
379 {
380 case 0:
381 /*SOBUF = RX_pload[2];
382 while(TIO)
383 {
384 TIO = 0;
385 }*/
386 break;
387 case 1:
388 break;
389 }
390 }

```

เอกสารนี้เป็นเอกสารสงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าในรูปแบบใดก็ตาม หากมีให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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] & 0x20;
    }
    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] & 0x20;
    }
    TX_pload[9] = 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] & 0x20;
    }
    TX_pload[6] = Address_Route;
  }
  else if (CD == 3)
  {
    Rx_Real_Mode();
    for (set=0; set<=9; set++)
      TX_pload[set] = RX_pload[set];
    for (set=13; 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] & 0x20;
    }
    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] & 0x20;
    }
    TX_pload[13] = 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] & 0x20;
    }
    TX_pload[18] = 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] & 0x20;
    }
    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] & 0x20;
    }
    TX_pload[24] = 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] & 0x20;
    }
    TX_pload[34] = Address_Route;
  }
}

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าในรูปแบบใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

547 void Read_Data ()
548 {
549     int SD;
550     SD = ((Address_Route >> 4) & 9);
551     if(SD == 0)
552     {
553         if(((Address_Route & 0xF0) == (RX_pload[0] & 0xF0) && (((Address_Route & 0x02)+2) == (RX_pload[6] & 0x02)) || ((Address_Route & 0x02)+15) == (RX_pload[30] & 0x02)))
554         {
555             Decode_input();
556             /*if(RX_pload[2] == 0x01)
557             {
558                 Delay_ms(100); P00 = -P00; Check = 1; }*/
559             //P00 = -P00;
560             Check = 1;
561         }
562         if(RX_pload[6] == 0x00)
563         {
564             if(((Address_Route & 0xF0) == (RX_pload[0] & 0xF0) && (((Address_Route & 0x02)+1) == (RX_pload[3] & 0x02)) || ((Address_Route & 0x02)+15) == (RX_pload[30] & 0
565             //P02 = -P02;
566             Decode_input();
567             Check = 1;
568         }
569     }
570     else if(SD == 1)
571     {
572         if(((Address_Route & 0xF0) == (RX_pload[3] & 0xF0) && (((Address_Route & 0x02)-1) == (RX_pload[0] & 0x02)))
573         //if(((Address_Route >> 4) == (RX_pload[3] >> 4) && (((Address_Route & 0x02)-1) && (RX_pload[0] & 0x02)))
574         {
575             Decode_input();
576             /*if(RX_pload[5] == 0x01)
577             {
578                 Delay_ms(100); P00 = -P00; Check = 1; }*/
579             Check = 1;
580         }
581     }
582     else if(SD == 2)
583     {
584         if(((Address_Route & 0xF0) && (RX_pload[6] & 0xF0) && (((Address_Route & 0x02)-1) == (RX_pload[3] & 0x02)))
585         {
586             Decode_input();
587             /*if(RX_pload[8] == 0x01)
588             {
589                 Delay_ms(100); P00 = -P00; Check = 1; }*/
590             if (TX_pload[7] == 0x00) P00 = -P00;
591             if (TX_pload[8] == 0x01) P01 = -P01;
592             Check = 1;
593         }
594     }
595     else if(SD == 3)
596     {
597         if(((Address_Route & 0xF0) && (RX_pload[9] & 0xF0) && (((Address_Route & 0x02)+2) == (RX_pload[15] & 0x02)) || ((Address_Route & 0x02)-2) == (RX_pload[3] & 0x02)
598         {
599             if(RX_pload[11] == 0x01)
600             {
601                 Delay_ms(100); P00 = -P00; Check = 1; }
602             }
603         if(RX_pload[15] == 0x00)
604         {
605             if(((Address_Route & 0xF0) && (RX_pload[9] & 0xF0) && (((Address_Route & 0x02)+1) == (RX_pload[12] & 0x02)) || ((Address_Route & 0x02)-2) == (RX_pload[3] & 0x
606             P02 = -P02;
607         }
608     }
609     else if(SD == 4)
610     {
611         if(((Address_Route & 0xF0) && (RX_pload[12] & 0xF0) && (((Address_Route & 0x02)-1) == (RX_pload[6] & 0x02)))
612         {
613             if(RX_pload[14] == 0x01)
614             {
615                 Delay_ms(100); P00 = -P00; Check = 1; }
616             }
617     }
618     else if(SD == 5)
619     {
620         if(((Address_Route & 0xF0) && (RX_pload[15] & 0xF0) && (((Address_Route & 0x02)-1) == (RX_pload[12] & 0x02)))
621         {
622             if(RX_pload[17] == 0x01)
623             {
624                 Delay_ms(100); P00 = -P00; Check = 1; }
625             }
626     }
627     else if(SD == 6)
628     {
629         if(((Address_Route & 0xF0) && (RX_pload[18] & 0xF0) && (((Address_Route & 0x02)+2) == (RX_pload[24] & 0x02)) || ((Address_Route & 0x02)-3) == (RX_pload[6] & 0x0
630         {
631             if(RX_pload[20] == 0x01)
632             {
633                 Delay_ms(100); P00 = -P00; Check = 1; }
634             }
635         if(RX_pload[24] == 0x00)
636         {
637             if(((Address_Route & 0xF0) && (RX_pload[9] & 0xF0) && (((Address_Route & 0x02)+1) == (RX_pload[21] & 0x02)) || ((Address_Route & 0x02)-3) == (RX_pload[2] & 0x
638             P02 = -P02;
639         }
640     }
641     else if(SD == 7)
642     {
643         if(((Address_Route & 0xF0) && (RX_pload[21] & 0xF0) && (((Address_Route & 0x02)-1) == (RX_pload[18] & 0x02)))
644         {
645             if(RX_pload[23] == 0x01)
646             {
647                 Delay_ms(100); P00 = -P00; Check = 1; }
648             }
649     }
650     else if(SD == 8)
651     {
652         if(((Address_Route & 0xF0) && (RX_pload[24] & 0xF0) && (((Address_Route & 0x02)-1) == (RX_pload[21] & 0x02)))
653         {
654             if(RX_pload[26] == 0x01)
655             {
656                 Delay_ms(100); P00 = -P00; Check = 1; }
657             }
658     }
659 }
660 void RF_Mode() interrupt 9 using 1
661 {
662     RF = 0;
663     P07 = -P07;
664     Rx_flag = 1;
665     RF = 1;
666 }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

658 void main()
659 {
660     PODIR = 0x10;
661     P1DIR = 0xff;
662
663     if (P12 == 1)
664     {
665         SW = 0x61;
666         Do = 1;
667     }
668     if (P12 == 0)
669     {
670         SW = 0x62;
671         Do = 1;
672     }
673
674     //INTEXP = 0x18;
675     //IP1 = 1;
676     //IPO = 1;
677
678     SOCON = 0x50;           //Mode 1 8-bit UART.
679     SOBUF = 0x00;
680     SORELH = 0x03;
681     SORELL = 0xF3;
682     ADCON = 0x80;
683
684     PCON = PCON| 0x80;
685
686     RF = 0;
687     //ESO = 1;
688     RIO = 0;
689     TIO = 0;
690     //EXO = 1;
691     //EX1 = 1;
692     //EA = 1;
693     nRF_CE = 0;
694     nRF_CS = 1;
695
696     RFCKEN = 1;
697     Delay_ms(500);
698
699     CE_Pin_Active(CE_LOW);           // Set CE pin low to enable standby mode
700     Delay_ms(200);
701     LO1_Flush_TX();
702     LO1_Flush_RX();
703
704     Delay_ms(500);
705     Rx_Mode();
706
707     while(1)
708     {
709         if (P12 == 1 && SW == 0x62)
710         {
711             SW = 0x61;
712             Do = 1;
713             onetime = 1;
714         }
715         if (P12 == 0 && SW == 0x61)
716         {
717             SW = 0x62;
718             Do = 1;
719             onetime = 1;
720         }
721     }
722
723     if(Do == 1)
724     {
725         //PO6 = ~PO6;
726         //Decode_output();           TX_pload
727         TX_pload[0] = Address_Route;
728         TX_pload[1] = 0x00;
729         if(onetime != 0x63)
730         {
731             TX_pload[2] = SW;
732         }
733         else (TX_pload[2] = onetime ; )
734
735
736
737         /*for (p=3;p<32;p++)
738         {
739             TX_pload[p] = 0x00;
740         }*/
741
742         Tx_Mode();                   //Tx_mode
743         Tx_Real_Mode();
744         Delay_ms(1000);
745         //PO6 = ~PO6;
746         Do = 0;
747         Delay_ms(500);
748         onetime = 0x63 ;
749
750
751
752
753     }
754
755     /*if(Rx_flag == 1)
756     {
757         //PO0 = ~PO0;
758         Check_Device();
759         Read_Data();
760         //CE_Pin_Active(CE_HIGH);
761         Rx_flag = 0;
762     }
763
764     Rx_Mode(); //ถ้ามีให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเข้าของเอกสารทุกครั้งที่มีการนำไปใช้
765     Delay_ms(10);
766 }

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อการเรียนการสอนเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าในรูปแบบใดก็ตาม หากมีให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



```

132 void LO1_Flush_TX(void)
133 {
134     nRF_SPI_RW_Reg(FLUSH_TX,0);
135 }
136 unsigned char LO1_Get_FIFO(void)
137 {
138     return nRF_SPI_Read(FIFO_STATUS);
139 }
140 unsigned char nRF_SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
141 {
142     unsigned char status, byte_ctr;
143     nRF_CS = 0; // Set CSN low, init SPI transaction
144     status = nRF_SPI_RW_Reg(reg); // Select register to write to and read status byte
145     for (byte_ctr=0;byte_ctr<bytes;byte_ctr++) // Perform SPI_RW to read byte from nRF24L01
146     {
147         pBuf[byte_ctr] = nRF_SPI_RW(0);
148     }
149     nRF_CS = 1; // Set CSN high again
150     return(status); // return nRF24L01 status byte
151 }
152 unsigned char nRF_SPI_Read(unsigned char reg)
153 {
154     unsigned char reg_val;
155     nRF_CS = 0; // CSN low, initialize SPI communication...
156     nRF_SPI_RW_Reg(reg); // Select register to read from..
157     reg_val = nRF_SPI_RW(0); // ..then read register value
158     nRF_CS = 1; // CSN high, terminate SPI communication
159     return(reg_val); // return register value
160 }
161 unsigned char LO1_Get_Status(void)
162 {
163     return nRF_SPI_Read(STATUS);
164 }
165 unsigned char LO1_Get_Current_Pipenum(void)
166 {
167     return ((LO1_Get_Status() & RX_P_NO) >> 1);
168 }
169 unsigned int LO1_Read_RX_Pload(unsigned char *pBuf)
170 {
171     unsigned char piWidth, pipe;
172     piWidth = LO1_RX_RX_PW_n(pipe = LO1_Get_Current_Pipenum()); // Read current pipe's payload width
173     nRF_SPI_Read_Buf(RD_RX_PLOAD, pBuf, piWidth); // Then get RX data
174     return ((pipe << 8) + piWidth); // return pipe# & pipe#*piWidth
175 }
176 /*unsigned char nRF_SPI_RW_Reg_IRQ(unsigned char reg, unsigned char value)
177 {
178     unsigned char status;
179     nRF_CS = 0; // CSN low, init SPI transaction
180     SSPIBUF = reg;
181     //while (SSP1STAT.BF == 0)
182     Delay_us(10);
183     status = SSPIBUF; // free the register
184     SSPIBUF = value;
185     //while (SSP1STAT.BF == 0)
186     Delay_us(10);
187     reg = SSPIBUF; // free the register
188     nRF_CS = 1; // CSN high again
189     return(status); // return nRF24L01 status byte
190 }*/
191 void Tx_Mode ()
192 {
193     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
194     Delay_ms(200);
195     LO1_Flush_TX();
196     LO1_Flush_RX();
197     //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
198     nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
199     //ucIRQ_Source = CLEAR;
200     //ucLastStat = ucLinkStat = LINK_ESTABLISH; // LINK_ESTABLISH = 0x02
201     nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack:Pipes 0-5
202     nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3F); // Enable Pipes 0-5
203     nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a); // 500µs + 66µs, 10 retrans...
204     nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
205     nRF_SPI_RW_Reg(WRITE_REG + SETUP_AW, 0x03); // Select Address
206     nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08); // TX_PWR:0dBm, DataRate:2Mbps, LNA:HCURR
207     nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
208     nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Writes TX_Address to nRF24L01
209     nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // RX_Addr0 same as TX_Adr for Auto.Ack
210     CE_Pin_Active(CE_HIGH);
211 }
212 /*void Tx_Mode_runtime()
213 {
214     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
215     //Delay_ms(200);
216     LO1_Flush_TX();
217     LO1_Flush_RX();
218     //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
219     //nRF_SPI_RW_Reg(WRITE_REG + STATUS, MASK_IRQ_FLAGS);
220     //ucIRQ_Source = CLEAR;
221     //ucLastStat = ucLinkStat = LINK_ESTABLISH; // LINK_ESTABLISH = 0x02
222     //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack:Pipes 0-5
223     //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3F); // Enable Pipes 0-5
224     //nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x2a); // 500µs + 66µs, 10 retrans...
225     //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
226     //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x08); // TX_PWR:0dBm, DataRate:2Mbps, LNA:HCURR
227     //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT & TX_DS enabled..
228     //nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR, &ADDRESS_PO, sizeof(ADDRESS_PO)); // Writes TX_Address to nRF24L01
229     //nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_PO, &ADDRESS_PO, sizeof(ADDRESS_PO)); // RX_Addr0 same as TX_Adr for Auto.Ack
230     Delay_ms(50); //max 1 ms
231     //CE_Pin_Active(CE_LOW);
232 }*/
233 void Rx_Mode ()
234 {
235     CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
236     Delay_ms(1); // 200ms
237     LO1_Flush_TX();
238     LO1_Flush_RX();
239     //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
240 }

```

เอกสารนี้เป็นไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าในรูปแบบใดก็ตามโปรดเปลี่ยนเนื้อหาให้ตรงข้อเท็จจริงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

251 nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack:Pipes 0-5
252 nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3F); // Enable Pipes 0-5
253 nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
254 nRF_SPI_RW_Reg(WRITE_REG + SETUP_AV, 0x03); // Select Address
255 nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x0F); // TX_PWR:0dBm, DataRate:2Mbps, LNA:HCURR
256 nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x02); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..
257
258 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
259 nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));
260
261 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
262 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
263 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
264 nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);
265
266 nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P0, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
267 nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P1, RX_PLOAD_WIDTH);
268 nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P2, RX_PLOAD_WIDTH);
269 nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P3, RX_PLOAD_WIDTH);
270 nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P4, RX_PLOAD_WIDTH);
271 nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P5, RX_PLOAD_WIDTH);
272
273 CE_Pin_Active(CE_HIGH);
274
275 }
276
277 void Rx_Mode_Runtime()
278 {
279 //CE_Pin_Active(CE_LOW); // Set CE pin low to enable standby mode
280 //Delay_ms(1); // 200ms
281 //LO1_Flush_TX();
282 //LO1_Flush_RX();
283
284 //LO1_Clear_IRQ(MASK_IRQ_FLAGS); // Clear interrupts
285
286 //nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3F); // Enable Auto.Ack:Pipes 0-5
287 //nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3F); // Enable Pipes 0-5
288 //nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40); // Select RF channel 40
289 //nRF_SPI_RW_Reg(WRITE_REG + SETUP_AV, 0x03); // Select Address
290 //nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x0F); // TX_PWR:0dBm, DataRate:2Mbps, LNA:HCURR
291 //nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x02); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..
292
293 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
294 nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, &ADDRESS_P1, sizeof(ADDRESS_P1));
295
296 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03);
297 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04);
298 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05);
299 //nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06);
300
301 //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P0, RX_PLOAD_WIDTH); // Select same RX payload width as TX Payload width
302 //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P1, RX_PLOAD_WIDTH);
303 //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P2, RX_PLOAD_WIDTH);
304 //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P3, RX_PLOAD_WIDTH);
305 //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P4, RX_PLOAD_WIDTH);
306 //nRF_SPI_RW_Reg(WRITE_REG + RX_PU_P5, RX_PLOAD_WIDTH);
307
308 CE_Pin_Active(CE_HIGH);
309 Delay_ms(30);
310
311 }
312
313 void Rx_Read_Mode()
314 {
315 Rx_Mode_Runtime();
316 do
317 {
318 //POD = -POD;
319 //uiRX_info = LO1_Read_RX_Pload(RX_pload); // Read current payload
320 // temp = LO1_Get_FIFO() & 0x02; temp >> 1;
321 }
322 while(!((LO1_Get_FIFO() & RX_EMPTY)); // ..until FIFO empty
323 //while( temp == 0 );
324 temp = LO1_Get_FIFO() & 0x01; // temp >> 1;
325 //P14 = temp;
326 //uiRX_info = LO1_Read_RX_Pload(RX_pload);
327 LO1_Clear_IRQ(MASK_RX_DR_FLAG);
328
329 /*if (RX_pload[0] == 0x31) (POD = -POD; RX_pload[0] = 0; )
330 if (RX_pload[1] == 0x32) (PO1 = -PO1; RX_pload[1] = 0; )
331 if (RX_pload[2] == 0x33) (PO2 = -PO2; RX_pload[2] = 0; )
332 if (RX_pload[3] == 0x34) (PO3 = -PO3; RX_pload[3] = 0; )
333 if (RX_pload[4] == 0x35) (PO4 = -PO4; RX_pload[4] = 0; )*/
334 CE_Pin_Active(CE_LOW);
335
336 }
337
338 void Tx_Read_Mode()
339 {
340 Send_Packet();
341 Delay_ms(1);
342 LO1_Clear_IRQ(MASK_TX_DS_FLAG);
343 }
344
345 unsigned char spi_master_read_write(unsigned char pload)
346 {
347 Delay_ms(1);
348 SPINDAT = pload; // Write data to SPI master
349 while(!((SPINSTAT & 0x04))); // Wait for data available in rx_fifo
350 return SPINDAT; // Return data register
351 }
352
353 long ADE_Read (unsigned char add, unsigned char num)
354 {
355 unsigned char i;
356 unsigned long Value;
357
358 spi_master_read_write(add);
359 for(i = 0; i < num; i++)
360 {
361 Read[i] = spi_master_read_write(0x00);
362 }
363
364 Value = 0;
365 for(i = 0; i < num; i++)
366 {
367 Value = Value << 8;
368 Value |= Read[i];
369 }
370 return (Value);
371 }
372
373 void ADE_Write (unsigned char add, unsigned char num, unsigned char *write)
374 {
375 unsigned char i;
376
377 spi_master_read_write(add | 0x10);
378 for(i = 0; i < num; i++)
379 {
380 spi_master_read_write(*write);
381 write++;
382 }
383 //ADE_Write(add,num,(unsigned char *) &ui); //How to
384
385 void Relay_onoff (int point ,int income)
386 {
387

```

เอกสารนี้เป็นเอกสารที่ให้บริการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าในรูปแบบใดก็ตาม และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

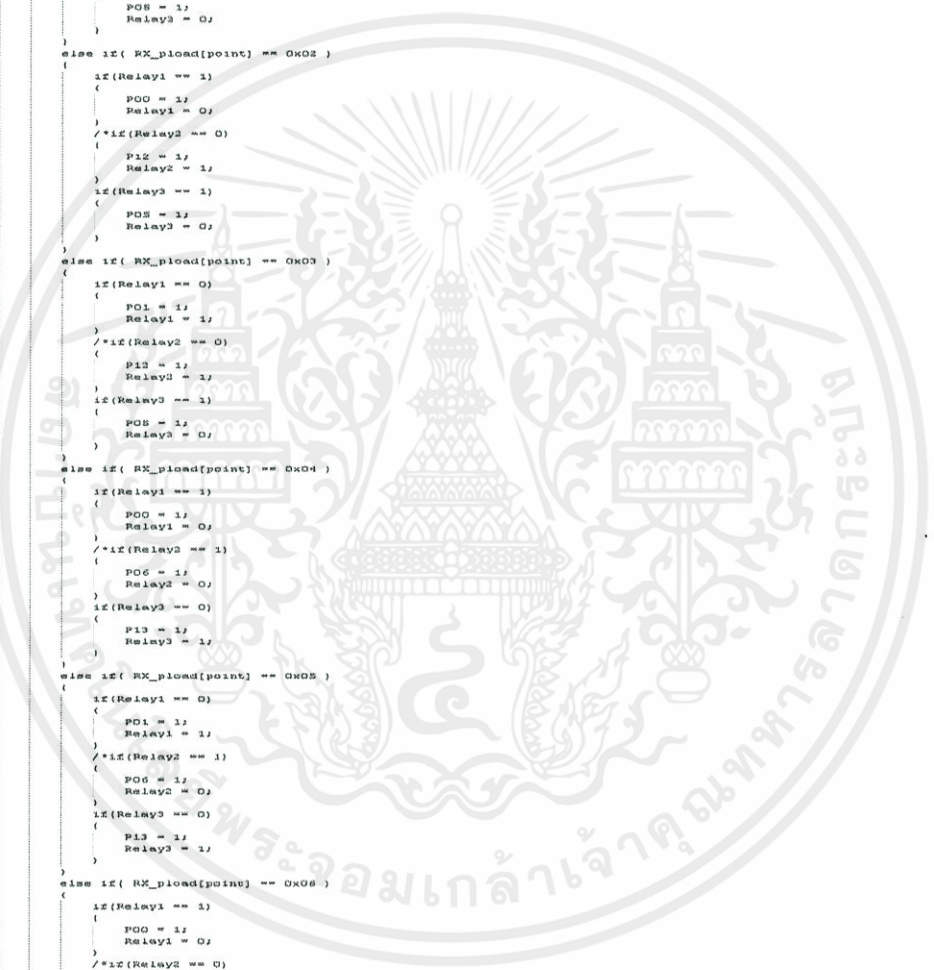
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538

```

RX_pload[point] = income;

/*if( RX_pload[point] == 0x00 )
{
  if(Relay1 == 1)
  {
    PG0 = 1;
    Relay1 = 0;
  }
  /*if(Relay2 == 1)
  {
    PG6 = 1;
    Relay2 = 0;
  }
  if(Relay3 == 1)
  {
    PG8 = 1;
    Relay3 = 0;
  }
}
else if( RX_pload[point] == 0x01 )
{
  if(Relay1 == 0)
  {
    PG1 = 1;
    Relay1 = 1;
  }
  /*if(Relay2 == 1)
  {
    PG6 = 1;
    Relay2 = 0;
  }
  if(Relay3 == 1)
  {
    PG8 = 1;
    Relay3 = 0;
  }
}
else if( RX_pload[point] == 0x02 )
{
  if(Relay1 == 1)
  {
    PG0 = 1;
    Relay1 = 0;
  }
  /*if(Relay2 == 0)
  {
    P12 = 1;
    Relay2 = 1;
  }
  if(Relay3 == 1)
  {
    PG8 = 1;
    Relay3 = 0;
  }
}
else if( RX_pload[point] == 0x03 )
{
  if(Relay1 == 0)
  {
    PG1 = 1;
    Relay1 = 1;
  }
  /*if(Relay2 == 0)
  {
    P13 = 1;
    Relay2 = 1;
  }
  if(Relay3 == 1)
  {
    PG8 = 1;
    Relay3 = 0;
  }
}
else if( RX_pload[point] == 0x04 )
{
  if(Relay1 == 1)
  {
    PG0 = 1;
    Relay1 = 0;
  }
  /*if(Relay2 == 1)
  {
    PG6 = 1;
    Relay2 = 0;
  }
  if(Relay3 == 0)
  {
    P13 = 1;
    Relay3 = 1;
  }
}
else if( RX_pload[point] == 0x05 )
{
  if(Relay1 == 0)
  {
    PG1 = 1;
    Relay1 = 1;
  }
  /*if(Relay2 == 1)
  {
    PG6 = 1;
    Relay2 = 0;
  }
  if(Relay3 == 0)
  {
    P13 = 1;
    Relay3 = 1;
  }
}
else if( RX_pload[point] == 0x06 )
{
  if(Relay1 == 1)
  {
    PG0 = 1;
    Relay1 = 0;
  }
  /*if(Relay2 == 0)
  {
    P12 = 1;
    Relay2 = 1;
  }
  if(Relay3 == 0)
  {
    P13 = 1;
    Relay3 = 1;
  }
}
else if( RX_pload[point] == 0x07 )
{
  if(Relay1 == 0)
  {
    PG1 = 1;
    Relay1 = 1;
  }
  /*if(Relay2 == 0)
  {
    P12 = 1;
    Relay2 = 1;
  }
  if(Relay3 == 0)
  {
    P13 = 1;
    Relay3 = 1;
  }
}
}

```

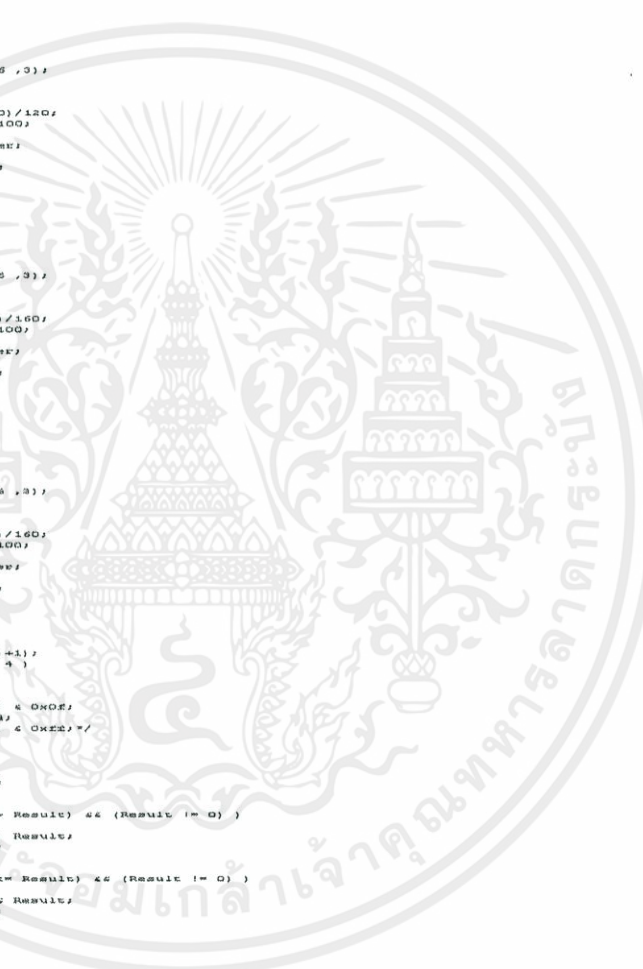


เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์ไว้เพื่อให้บริการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่าจะใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

5333 void Decode_output ()
5334 {
5335     int DO ,output;
5336     DO = (3*((Address_Route>>4) & 3)+1);
5337     for (output=0;output<3;output++)
5338     {
5339         TX_pload[output] = RX_pload[output];
5340     }
5341     /*if ( Send_Decode < Data_All )
5342     {
5343         Send_Decode = 0;
5344     }
5345     output = (RX_pload[3] & 0x03);*/
5346     /*if(output == 0x00)
5347     {
5348         TX_pload[DO] = 0x00;
5349         TX_pload[DO+1] = 0x00;
5350         Decode_out = Decode_out & 0x0000;
5351
5352         if( Relay1 == 1 )
5353         | Decode_out |= 0x0001;
5354         else
5355         | Decode_out |= 0x0000;
5356         if( Relay2 == 1 )
5357         | Decode_out |= 0x0002;
5358         else
5359         | Decode_out |= 0x0000;
5360         if( Relay3 == 1 )
5361         | Decode_out |= 0x0004;
5362         else
5363         | Decode_out |= 0x0000;
5364         TX_pload[DO+1] = Decode_out & 0x0F;
5365     }
5366     /*else if(output == 0x01)
5367     {
5368         TX_pload[DO] = 0x10;
5369         TX_pload[DO+1] = 0x00;
5370
5371         on_on1;
5372         delay_nop
5373         delay_nop
5374         Irms1 = ADE_Read (0x16 ,3);
5375         on_off1;
5376         delay_nop
5377         Irms1 = (Irms1 - 1300)/120;
5378         Power = (Vrms*Irms1)/100;
5379         TX_pload[DO+1] |= Power;
5380         Power = Power >> 8;
5381         TX_pload[DO] |= Power;
5382     }
5383     else if(output == 0x02)
5384     {
5385         TX_pload[DO] = 0x20;
5386         TX_pload[DO+1] = 0x00;
5387
5388         on_on2;
5389         delay_nop
5390         delay_nop
5391         Irms2 = ADE_Read (0x16 ,3);
5392         on_off2;
5393         delay_nop
5394         Irms2 = (Irms2 - 1500)/160;
5395         Power = (Vrms*Irms2)/100;
5396         TX_pload[DO+1] |= Power;
5397         Power = Power >> 8;
5398         TX_pload[DO] |= Power;
5399     }
5400     else if(output == 0x03)
5401     {
5402         TX_pload[DO] = 0x30;
5403         TX_pload[DO+1] = 0x00;
5404
5405         on_on3;
5406         delay_nop
5407         delay_nop
5408         Irms3 = ADE_Read (0x16 ,3);
5409         on_off3;
5410         delay_nop
5411         Irms3 = (Irms3 - 1800)/160;
5412         Power = (Vrms*Irms3)/100;
5413         TX_pload[DO+1] |= Power;
5414         Power = Power >> 8;
5415         TX_pload[DO] |= Power;
5416     }
5417     }*/
5418 }
5419 void Decode_input ()
5420 {
5421     int DI;
5422     DI = (3*((Address_Route>>4) & 3)+1);
5423     switch ( (RX_pload[DI] & 0xF0) >> 4 )
5424     {
5425         case 0x00:
5426             /*Decode_in = RX_pload[DI] & 0x0F;
5427             Decode_in = Decode_in << 4;
5428             Decode_in = RX_pload[DI+1] & 0x0F;*/
5429             /*if( Result == 0 )
5430             {
5431                 Sum = RX_pload[DI+1];
5432                 Relay_onOFF (DI+1,Sum);
5433                 Result = Sum;
5434             }
5435             else if( (RX_pload[DI+1] > Result) && (Result != 0) )
5436             {
5437                 Sum = RX_pload[DI+1] | Result;
5438                 Relay_onOFF (DI+1,Sum);
5439                 Result = Sum;
5440             }
5441             else if( (RX_pload[DI+1] <= Result) && (Result != 0) )
5442             {
5443                 Sum = RX_pload[DI+1] & Result;
5444                 Relay_onOFF (DI+1,Sum);
5445                 Result = Sum;
5446             }
5447             */
5448             /*Sum = RX_pload[DI+1];
5449             //Relay_onOFF (DI+1 ,Sum);
5450             switch (RX_pload[DI])
5451             {
5452                 case 0x01:
5453                     P00 = 0;
5454                     P01 = 1;
5455                     break;
5456                 case 0x02:
5457                     P00 = 1;
5458                     P01 = 0;
5459                     break;
5460             }
5461             Delay_ms (50);
5462             /*P07 = 1;
5463             P06 = 0;
5464             P05 = 0;
5465             P10 = 0;
5466             P12 = 0;
5467             P13 = 0;*/
5468             break;
5469     }

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

692 void Check_Device()
693 {
694     int CD, set, pload;
695     CD = ((Address_Route >> 4) & 9);
696     if (CD == 0)
697     {
698         Rx_Real_Mode();
699         for (set=3; set<=31; set++)
700             TX_pload[set] = RX_pload[set];
701         for (pload=3; pload<=24; pload++)
702             if ((pload & 3) == 0)
703                 TX_pload[pload] = TX_pload[pload] & 0x20;
704         TX_pload[0] = Address_Route;
705     }
706     else if (CD == 1)
707     {
708         Rx_Real_Mode();
709         for (set=0; set<=2; set++)
710             TX_pload[set] = RX_pload[set];
711         for (set=6; set<=24; set++)
712             TX_pload[set] = RX_pload[set];
713         for (pload=0; pload<=30; pload++)
714             if ((pload & 3) == 0)
715                 TX_pload[pload] = TX_pload[pload] & 0x20;
716         TX_pload[3] = Address_Route;
717     }
718     else if (CD == 2)
719     {
720         Rx_Real_Mode();
721         for (set=0; set<=5; set++)
722             TX_pload[set] = RX_pload[set];
723         for (set=9; set<=24; set++)
724             TX_pload[set] = RX_pload[set];
725         for (pload=0; pload<=30; pload++)
726             if ((pload & 3) == 0)
727                 TX_pload[pload] = TX_pload[pload] & 0x20;
728         TX_pload[6] = Address_Route;
729     }
730     else if (CD == 3)
731     {
732         Rx_Real_Mode();
733         for (set=0; set<=8; set++)
734             TX_pload[set] = RX_pload[set];
735         for (set=12; set<=24; set++)
736             TX_pload[set] = RX_pload[set];
737         for (pload=0; pload<=30; pload++)
738             if ((pload & 3) == 0)
739                 TX_pload[pload] = TX_pload[pload] & 0x20;
740         TX_pload[9] = Address_Route;
741     }
742     else if (CD == 4)
743     {
744         Rx_Real_Mode();
745         for (set=0; set<=11; set++)
746             TX_pload[set] = RX_pload[set];
747         for (set=15; set<=24; set++)
748             TX_pload[set] = RX_pload[set];
749         for (pload=0; pload<=30; pload++)
750             if ((pload & 3) == 0)
751                 TX_pload[pload] = TX_pload[pload] & 0x20;
752         TX_pload[12] = Address_Route;
753     }
754     else if (CD == 5)
755     {
756         Rx_Real_Mode();
757         for (set=0; set<=14; set++)
758             TX_pload[set] = RX_pload[set];
759         for (set=18; set<=24; set++)
760             TX_pload[set] = RX_pload[set];
761         for (pload=0; pload<=30; pload++)
762             if ((pload & 3) == 0)
763                 TX_pload[pload] = TX_pload[pload] & 0x20;
764         TX_pload[15] = Address_Route;
765     }
766     else if (CD == 6)
767     {
768         Rx_Real_Mode();
769         for (set=0; set<=17; set++)
770             TX_pload[set] = RX_pload[set];
771         for (set=21; set<=24; set++)
772             TX_pload[set] = RX_pload[set];
773         for (pload=0; pload<=30; pload++)
774             if ((pload & 3) == 0)
775                 TX_pload[pload] = TX_pload[pload] & 0x20;
776         TX_pload[18] = Address_Route;
777     }
778     else if (CD == 7)
779     {
780         Rx_Real_Mode();
781         for (set=0; set<=20; set++)
782             TX_pload[set] = RX_pload[set];
783         for (set=24; set<=30; set++)
784             TX_pload[set] = RX_pload[set];
785         for (pload=0; pload<=30; pload++)
786             if ((pload & 3) == 0)
787                 TX_pload[pload] = TX_pload[pload] & 0x20;
788         TX_pload[21] = Address_Route;
789     }
790 }

```



เอกสารนี้เป็นเอกสารราชการ ใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
 ไม่ว่าในรูปแบบใดก็ตาม หากมีข้อสงสัยหรือต้องการข้อมูลเพิ่มเติม กรุณาติดต่อฝ่ายประชาสัมพันธ์ โทร. 02-254-4000

```

else if(CD == 0)
{
  RX_Real_Mode();
  for(set=0;set<=23;set++)
  TX_pload[set] = RX_pload[set];
  for(set=27;set<=30;set++)
  TX_pload[set] = RX_pload[set];
  for(pload=0;pload<=30;pload++)
  {
    if((pload & 3) == 0)
      TX_pload[pload] = TX_pload[pload] & 0x20;
    TX_pload[24] = Address_Route;
  }
  CE_Pin_Active(CK_LCW);
}
void Read_Data ()
{
  int SD;
  SD = ((Address_Route>>4) & 9);
  if(SD == 0)
  {
    if(((Address_Route & 0x20) == (RX_pload[0] & 0x20)) && (((Address_Route & 0x2E)+2) == (RX_pload[6] & 0x2E)) || (((Address_Route & 0x2E)+15) == (RX_pload[10] & 0x2E)))
    {
      Decode_input();
      /*if(RX_pload[2] == 0x01)
      { Delay_ms(100); P00 = -P00; Check = 1; }*/
      //P00 = -P00;
      Check = 1;
    }
    if(RX_pload[6] == 0x0D)
    {
      Decode_input();
      Check = 1;
      if(((Address_Route & 0x20) == (RX_pload[0] & 0x20)) && (((Address_Route & 0x2E)+1) == (RX_pload[3] & 0x2E)) || (((Address_Route & 0x2E)+15) == (RX_pload[30] & 0x2E)))
      {
        Decode_input();
        Check = 1;
      }
    }
  }
  else if(SD == 1)
  {
    if(((Address_Route & 0x20) == (RX_pload[3] & 0x20)) && (((Address_Route & 0x2E)-1) == (RX_pload[0] & 0x2E)))
    //(((Address_Route>>4) == (RX_pload[3]>>4)) && (((Address_Route&0x2E)-1) && (RX_pload[0]&0x2E)))
    {
      Decode_input();
      /*if(RX_pload[5] == 0x01)
      { Delay_ms(100); P00 = -P00; Check = 1; }*/
      //P00 = -P00;
      Check = 1;
    }
  }
  else if(SD == 2)
  {
    if(((Address_Route & 0x20) && (RX_pload[6] & 0x20)) && (((Address_Route & 0x2E)-1) == (RX_pload[3] & 0x2E)))
    {
      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;
      //P00 = -P00;
      Check = 1;
    }
  }
  else if(SD == 3)
  {
    if(((Address_Route & 0x20) && (RX_pload[9] & 0x20)) && (((Address_Route & 0x2E)+2) == (RX_pload[15] & 0x2E)) || (((Address_Route & 0x2E)-2) == (RX_pload[3] & 0x2E)))
    {
      if(RX_pload[11] == 0x01)
      { Delay_ms(100); P00 = -P00; Check = 1; }
      if(RX_pload[16] == 0x0D)
      {
        if(((Address_Route & 0x20) && (RX_pload[9] & 0x20)) && (((Address_Route & 0x2E)+1) == (RX_pload[12] & 0x2E)) || (((Address_Route & 0x2E)-2) == (RX_pload[3] & 0x2E)))
        {
          //P02 = -P02;
        }
      }
    }
  }
  else if(SD == 4)
  {
    if(((Address_Route & 0x20) && (RX_pload[12] & 0x20)) && (((Address_Route & 0x2E)-1) == (RX_pload[6] & 0x2E)))
    {
      if(RX_pload[14] == 0x01)
      { Delay_ms(100); P00 = -P00; Check = 1; }
    }
  }
  else if(SD == 5)
  {
    if(((Address_Route & 0x20) && (RX_pload[15] & 0x20)) && (((Address_Route & 0x2E)-1) == (RX_pload[3] & 0x2E)))
    {
      if(RX_pload[17] == 0x01)
      { Delay_ms(100); P00 = -P00; Check = 1; }
    }
  }
  else if(SD == 6)
  {
    if(((Address_Route & 0x20) && (RX_pload[16] & 0x20)) && (((Address_Route & 0x2E)+2) == (RX_pload[24] & 0x2E)) || (((Address_Route & 0x2E)-3) == (RX_pload[6] & 0x2E)))
    {
      if(RX_pload[20] == 0x01)
      { Delay_ms(100); P00 = -P00; Check = 1; }
      if(RX_pload[24] == 0x0D)
      {
        if(((Address_Route & 0x20) && (RX_pload[9] & 0x20)) && (((Address_Route & 0x2E)+1) == (RX_pload[12] & 0x2E)) || (((Address_Route & 0x2E)-3) == (RX_pload[2] & 0x2E)))
        {
          //P02 = -P02;
        }
      }
    }
  }
  else if(SD == 7)
  {
    if(((Address_Route & 0x20) && (RX_pload[21] & 0x20)) && (((Address_Route & 0x2E)-1) == (RX_pload[16] & 0x2E)))
    {
      if(RX_pload[23] == 0x01)
      { Delay_ms(100); P00 = -P00; Check = 1; }
    }
  }
  else if(SD == 8)
  {
    if(((Address_Route & 0x20) && (RX_pload[24] & 0x20)) && (((Address_Route & 0x2E)-1) == (RX_pload[21] & 0x2E)))
    {
      if(RX_pload[26] == 0x01)
      { Delay_ms(100); P00 = -P00; Check = 1; }
    }
  }
}
void RF_Mode() interrupt 0 using 1
{
  RF = 0;
  //P07 = -P07;
  Rx_Flag = 1;
  RF = 1;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

void main() {
//int LOOP,TEST;
RF = 1;
EA = 1;

SPIMCON0 = 0;
SOCON = 0x50; //Mode 1 8-bit UART.
SBUF = 0x00;
SORELH = 0x03;
SORELL = 0xF3;
ADCON = 0x80;

PCON = PCON| 0x80;
SPIMCON0 = 0x55;

nRF_CE = 0;
nRF_CS = 1;
PODIR = 0x10;

SPIMCON0 = 0; //Mode 1 8-bit UART.
SOCON = 0x50;
SBUF = 0x00;
SORELH = 0x03;
SORELL = 0xF3;
ADCON = 0x80;

PCON = PCON| 0x80;
SPIMCON0 = 0x55;

nRF_CE = 0;
nRF_CS = 1;
PODIR = 0x10;
PIDIR = 0x00; // for SPI PIDIR = 0x0f;

PG2 = 0;
PG7 = 0;
PG6 = 0;
PG5 = 0;
PI0 = 0;
PI2 = 0;
PI3 = 0;

RFCKEN = 1;
//Delay_ms(500);

CE_Pin_Active(CE_LOV); // Set CE pin low to enable standby mode
//Delay_ms(200);
LQ1_Flush_TX();
LQ1_Flush_RX();

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

//TIO = 0;

cs_on1;
delay_nop
delay_nop
ui = 0x0040;
ADE_Write(0x09,2,(unsigned char *)&ui);
Delay_ms(200);
ui = 0x000c;
ADE_Write(0x09,2,(unsigned char *)&ui);
uc = 0x00;
ADE_Write(0x02,1,(unsigned char *)&uc);
uc = 0x00;
ADE_Write(0x0d,1,(unsigned char *)&uc);
uc = 0x00;
ADE_Write(0x0e,1,(unsigned char *)&uc);
cs_off1;
delay_nop

cs_on2;
delay_nop
delay_nop
ui = 0x0040;
ADE_Write(0x09,2,(unsigned char *)&ui);
Delay_ms(200);
ui = 0x000c;
ADE_Write(0x09,2,(unsigned char *)&ui);
uc = 0x00;
ADE_Write(0x02,1,(unsigned char *)&uc);
uc = 0x00;
ADE_Write(0x0d,1,(unsigned char *)&uc);
uc = 0x00;
ADE_Write(0x0e,1,(unsigned char *)&uc);
cs_off2;
delay_nop

cs_on3;
delay_nop
delay_nop
ui = 0x0040;
ADE_Write(0x09,2,(unsigned char *)&ui);
Delay_ms(200);
ui = 0x000c;
ADE_Write(0x09,2,(unsigned char *)&ui);
uc = 0x00;
ADE_Write(0x02,1,(unsigned char *)&uc);
uc = 0x00;
ADE_Write(0x0d,1,(unsigned char *)&uc);
uc = 0x00;
ADE_Write(0x0e,1,(unsigned char *)&uc);
cs_off3;
delay_nop

while(1)
{
if(Rx_Flag == 1)
{
PO2 = ~PO2;
Check_Device();
Read_Data();

if(Check == 1)
{
RF = 0;
Decode_output();
Tx_Mode();
Tx_Real_Mode();
Delay_ms(10);
Check = 0;
RF = 1;
}
Rx_Mode();
Rx_Flag = 0;
}
}
}

```

เอกสารนี้เป็นเอกสารที่จัดทำขึ้นเพื่อการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ หากมีให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้