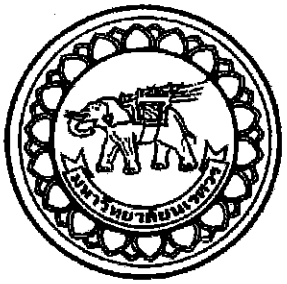
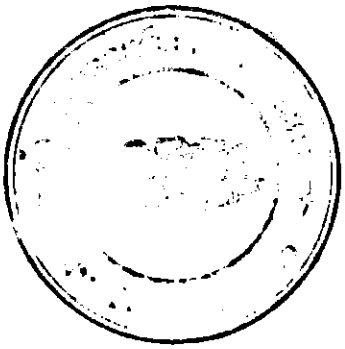




เครื่องควบคุมการรดน้ำอัตโนมัติโดยไมโครคอนโทรลเลอร์ MCS – 51

Watering Timer

นายกองพล โสดา รหัส 47363726

นายมนชิต บุญวงศ์ รหัส 47363999

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2550

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	เครื่องควบคุมการรดน้ำอัตโนมัติโดยไมโครคอนโทรลเลอร์ MCS – 51
ผู้ดำเนินโครงงาน	นายกองพล โสดา รหัส 47363726
	นายมนชิต บุญวงศ์ รหัส 47363999
อาจารย์ที่ปรึกษา	ดร. อัครพันธ์ วงศ์กังแห
สาขาวิชา	วิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2550

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ
(ดร. อัครพันธ์ วงศ์กังแห)

.....กรรมการ
(ดร. ชัยรัตน์ พินทอง)

.....กรรมการ
(อาจารย์ปิยดนัย ภาชนะพรรณ์)

หัวข้อโครงงาน	เครื่องควบคุมการรดน้ำอัตโนมัติโดยไมโครคอนโทรลเลอร์ MCS – 51
ผู้ดำเนินโครงงาน	นายกองพล โสดา รหัส 47363726
	นายมนชิต บุญวงศ์ รหัส 47363999
อาจารย์ที่ปรึกษา	ดร. อัครพันธ์ วงศ์กังแห
สาขาวิชา	วิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2550

บทคัดย่อ

ในปัจจุบันการทำงานในหลายสาขาอาชีพ มักมีเครื่องจักรเข้ามาช่วยผ่อนแรงในการทำงาน ไม่ว่าจะเป็น งานอุตสาหกรรม หรืองานเกษตรกรรม ซึ่งมีเทคโนโลยีที่เข้ามาใช้เพื่อการพัฒนาไปสู่ความทันสมัย และสามารถใช้งานได้ทันตามความต้องการของผู้บริโภค และเนื่องด้วยปัจจุบันผู้ประกอบการแต่ละองค์กรย่อมมีความต้องการสร้างความมั่นคง และช่วยลดค่าใช้จ่ายในเรื่องของทรัพยากรบุคคล สิ่งนี้เองจึงเป็นสิ่งกระตุ้นให้เกิดโครงงานนี้ขึ้น

โครงงานวิจัยฉบับนี้มุ่งเน้นศึกษา การควบคุมการทำงานของการเปิด-ปิดน้ำ ผ่านโซลินอยวาล์วด้วยไมโครคอนโทรลเลอร์ MCS – 51 ซึ่งสามารถนำไปประยุกต์ใช้งานกับโรงเพาะชำทางการเกษตร หรือสวนเกษตรต่างๆ รวมไปถึงการรดน้ำสวนในบ้านอีกด้วย ซึ่งการทำงานโดยปกติจะใช้แรงงานคนในการทำงาน บางครั้งทำให้เสียเวลาและเสียทรัพยากรบุคคลในการปฏิบัติงาน การที่สามารถสร้างอุปกรณ์อิเล็กทรอนิกส์มาช่วยในการลดทรัพยากรบุคคลและประหยัดเวลาในการทำงานจึงมีความสำคัญในการพัฒนากิจการ โดยหลักการทำงานของเครื่องนั้นสามารถกำหนดเวลาในการทำงานได้หลายแบบ ขึ้นอยู่กับความประสงค์ของผู้ประกอบการ

Project Title	Watering Timer.	
Name	Mr. Kongpol Soda	ID. 47363726
	Mr. Monchit Boonwong	ID. 47363999
Project Advisor	Dr. Akaraphunt Vongkunghae	
Major	Electrical Engineering.	
Department	Electrical and Computer Engineering.	
Academic Year	2007	

ABSTRACT

This project is an application of MCS-51 microcontroller and DS 1307 real time clock. The watering time and period can be set by user. This system is used for our convenience in general watering purpose.

กิตติกรรมประกาศ

ขอขอบคุณ อาจารย์ที่ปรึกษาโครงงาน ดร.อัครพันธ์ วงกังแห ที่เมตตารับเป็นอาจารย์ที่ปรึกษาโปรเจคพวกเราแม้ว่าท่านจะมีภาระกิจอันมากมาย ท่านยังมีเวลาให้ปรึกษากับพวกเรา และให้คำชี้แนะที่ดีและมีประโยชน์ รวมไปถึง ดร.ชัยรัตน์ พินทอง และอาจารย์ปิยดนัย ภาชนะพรรณ์ ที่ร่วมเป็นกรรมการคุมสอบด้วย

ขอขอบคุณอาจารย์ทุกๆท่านที่ได้มอบความรู้ที่มีคุณค่าให้แก่พวกเรา และขอขอบคุณน้ำใจของเพื่อนๆทุกคนที่ให้กำลังใจกันตลอดมา รวมทั้งให้ความช่วยเหลือในด้านต่างๆ ไม่ว่าจะเป็นทางด้านการซื้ออุปกรณ์ การเขียนโปรแกรม ขอบใจที่คอยให้คำปรึกษาเรื่อยมา

สุดท้าย พวกเราใคร่ขอกราบขอบพระคุณบิดา มารดา พี่ๆ น้องๆ ทุกคน ที่ให้การสนับสนุนทั้งกายและใจ รวมไปถึงปัจจัยเรื่องเงิน ทั้งหมดทั้งมวลนี้เป็นเหตุให้พวกเราพยายามทำงานให้สำเร็จลุล่วงอย่างทันเวลาที่กำหนด ที่สำคัญคือกำลังใจที่ได้รับและพวกเราจะไม่ขอลืมพระคุณของทุกท่านที่เราได้กล่าวมาข้างต้นตลอดไป

คณะผู้จัดทำโครงงาน
นายกองพล โสดา
นายมนชิต บุญวงศ์

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

บทที่1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

จากแนวพระราชดำรัชเรื่องปรัชญาเศรษฐกิจพอเพียง ในพระบาทสมเด็จพระเจ้าอยู่หัวฯ ที่ทรงเน้นให้พสกนิกรตระหนักถึงความสำคัญในด้านการใช้พลังงาน และด้านความเป็นอยู่ของพสกนิกรให้อยู่อย่างพอเพียง เมื่อเราพิจารณาไปถึงอาชีพเกษตรกรรมและจะเห็นได้ว่า เป็นอาชีพที่ไม่ควรมองข้ามเพราะอาชีพพื้นฐานของประชากรไทยนั้นคือ อาชีพเกษตรกรรม ดังนั้นการพัฒนาการเกษตรให้ทันสมัยเพื่อให้ทันต่อยุคโลกาภิวัตน์นั้น จึงจำเป็นต้องอาศัยความรู้หลายด้านประกอบกัน เช่น ความรู้ด้านวิศวกรรม ด้านสถาปัตยกรรม ด้านเศรษฐศาสตร์ การตลาด การบัญชี เป็นต้น

ซึ่งในงานวิศวกรรมนั้นได้เข้าไปมีบทบาทสำคัญในการพัฒนาขีดความสามารถในการทำงานทั้งในแวดวงอุตสาหกรรม โลจิสติกส์ รวมไปถึงเกษตรกรรม และการเติบโตทางอุตสาหกรรมขนาดใหญ่ย่อมมีเทคโนโลยีใหม่ๆเกิดขึ้นมากมาย แต่เมื่อเราหันกลับมองมายังวิถีชีวิตเกษตรกรรมแล้วพบว่า ยังต้องมีการพัฒนาอีกมากเพื่อให้ทัดเทียมกับนานาประเทศ ในเรื่องของหลักการ และ/หรือขั้นตอนการผลิต การบริหารจัดการ เป็นต้น การประยุกต์ใช้ไมโครคอนโทรเลอร์เพื่อนำมาสั่งการควบคุมการทำงานของเครื่องจักรบางชนิดจึงสามารถช่วยประหยัดทรัพยากรบุคคลได้ ดังนั้นการพัฒนาและการประยุกต์ดังกล่าวจึงเป็นสิ่งที่น่าสนใจอย่างยิ่งในการเพิ่มประสิทธิภาพในการทำงานให้กับผู้ประกอบการต่อไป

ในการทำงานที่ต้องอาศัยน้ำเพื่อเป็นปัจจัยในการเจริญเติบโตนั้น ต้องมีการควบคุมการเปิด-ปิดน้ำ ดังเช่นการทำเกษตรกรรมย่อมต้องมีการรดน้ำอาจจะเป็นแปลงเพาะชำ โรงเพาะเห็ด สวน ไร่ รวมไปถึงการรดน้ำในสวนที่บ้าน ซึ่งโดยปกติแล้วต้องอาศัยคนในการทำงานหรือถ้าเป็นสวนใหญ่ๆก็ต้องมีการจ้างงานเกิดขึ้น สิ่งเหล่านี้จึงทำให้เพิ่มค่าใช้จ่ายในการประกอบการ เหตุผลดังกล่าวเมื่อนำมาพิจารณาแล้วพบว่า การที่เราสามารถควบคุมการเปิด-ปิด รดน้ำแบบอัตโนมัติได้จึงเป็นสิ่งที่ควรนำมาใช้งานในการประกอบการจริงเพราะประโยชน์ที่ได้รับนั้นสามารถช่วยลดทรัพยากรบุคคลในการทำงาน และยังสามารถประหยัดเวลาที่ทำงานได้อีกด้วย

ในโครงงานฉบับนี้มุ่งเน้นศึกษาการทำงานของไมโครคอนโทรลเลอร์ MCS-51 สื่อสารกับ DS1307 ซึ่งเป็นไอซีฐานเวลาจริง กำหนดการตั้งเวลาเปิด-ปิด รวมไปถึงแสดงผลทางหน้าจอLCD และนำมาประยุกต์ใช้งานกับสวนผัก สวนต้นไม้ หรือโรงเพาะชำทุกชนิด เพื่อช่วยลดทรัพยากรบุคคลในการทำงานและเพิ่มประสิทธิภาพในการควบคุมการเปิด-ปิดน้ำอัตโนมัติ ซึ่งสามารถตั้งเวลาและกำหนดเวลาในการทำงานได้ตามความประสงค์ของผู้ประกอบการ

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาทำความเข้าใจถึงการทำงานของไมโครคอนโทรลเลอร์ MCS-51
2. เขียนโปรแกรมประยุกต์ใช้งานกับไมโครคอนโทรเลอร์
3. สร้างแบบจำลอง การทำงานจริงควบคุมโดยไมโครคอนโทรเลอร์
4. เพื่อเป็นการนำความรู้ที่ได้จากการศึกษาไมโครคอนโทรเลอร์ มาประยุกต์กับการทำงานในชีวิตจริง

1.3 ขอบข่ายของโครงงาน

เน้นศึกษาการควบคุมการเปิด-ปิดน้ำผ่านโซลินอยด์วาล์ว ส่งออกไปยังแปลงเกษตรหรือโรงเพาะชำต่างๆ แล้วสามารถนำทฤษฎีที่ได้มาออกแบบและเขียนโปรแกรมควบคุมสั่งการการทำงานลงในบอร์ดไมโครคอนโทรเลอร์ MCS-51 สร้างแบบจำลองการทำงาน และสามารถนำแบบจำลองที่ได้ไปใช้ในสวนเกษตรหรือโรงเพาะชำจริงได้

1.4 ขั้นตอนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงาน

แผนการดำเนินงาน	ปี 2550								ปี 2551			
	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.
1. รวบรวมและเก็บข้อมูล	←	—	—	→								
2. เขียนโปรแกรมเพื่อควบคุมการทำงาน MCS-51				←	—	—	—	→				
3. สร้างแบบจำลองการทำงาน								←	—	—	→	
4. จัดทำรายงานและสรุปผลการทำงาน											←	→

1.5 ผลที่คาดว่าจะได้รับ

1. มีความเข้าใจในสถาปัตยกรรมและหลักการทำงานของไมโครคอนโทรลเลอร์ MCS-51
2. สามารถเขียนโปรแกรมควบคุมการทำงานในไมโครคอลโทรลเลอร์ได้
3. สามารถออกแบบและสร้างชิ้นงานเพื่อนำไปใช้งานในอุตสาหกรรมการเกษตรได้

4. เพื่อให้เห็นถึงความสำคัญในการประยุกต์ใช้ไมโครคอนโทรลเลอร์ MCS-51 ในการทำงานจริงได้

1.6 งบประมาณที่ใช้

1. ค่าถ่ายเอกสารและค่าเข้าเล่ม	1,000	บาท
2. ค่าอุปกรณ์ในการทำงาน	4,000	บาท
รวมเป็นเงิน	5,000	บาท (ห้าพันบาทถ้วน)

(หมายเหตุ ถัวเฉลี่ยทุกรายการ)

บทที่ 2
ทฤษฎีและหลักการทำงาน

ไมโครคอนโทรลเลอร์ได้ถูกนำมาประยุกต์ใช้งานกับการทำงานในหลายสาขาอาชีพ เช่น งานวิศวกรรม งานการแพทย์ เครื่องจักรโรงงาน รวมไปถึงงานเกษตรกรรม ซึ่งเทคโนโลยีดังกล่าวสามารถนำมาใช้งานเพื่อเพิ่มประสิทธิภาพ และขีดความสามารถได้เป็นอย่างดี ในโครงงานฉบับนี้เน้นศึกษาการพัฒนาเทคโนโลยีที่ใช้ในงานด้านการเกษตร ในเรื่องของการจ่ายน้ำเพื่อไปรดให้กับผลิตผลทางการเกษตร เช่น พืช ผัก สวน ไร่ รวมไปถึงแปลงเพาะชำเป็นต้น

ในบทนี้จะกล่าวถึงหลักการและทฤษฎีต่างๆที่เกี่ยวข้องกับการทำงานของระบบการรดน้ำอัตโนมัติ ความสำคัญที่เกิดขึ้นจากการทำโครงงาน ซึ่งมีความจำเป็นอย่างยิ่งในการศึกษาเพื่อทำการทดลองและจะเป็นพื้นฐานองค์ประกอบทางความรู้ให้ผู้อ่านเกิดความเข้าใจในขั้นตอนการปฏิบัติต่อไป

2.1 แนวคิดในการทำงาน

ในการรดน้ำพืชผลทางการเกษตร รวมไปถึงการรดน้ำสวนที่บ้านของเราเอง มักจะทำให้ผู้ที่มีเวลาในการดูแลน้อยนั้น เสียเวลากับสิ่งที่ต้องปฏิบัติเป็นประจำ อย่างเช่น การรดน้ำต้นไม้ย่อมมีการรดน้ำอย่างเป็นเวลา กล่าวคือ รดเช้า รดเย็น หรือ รดเช้า กลางวัน และเย็น ขึ้นอยู่กับชนิดของพันธุ์พืช และยิ่งการประกอบการที่เกี่ยวกับธรรมชาติด้วยแล้วต้องมีการรดน้ำพืชนั้นอย่างเป็นเวลา จึงจะทำให้ผลผลิตที่ได้มีคุณภาพตามประสงค์ของประกอบการ

แนวคิดในการทำโครงงานจึงเกิดขึ้นว่า ในเมื่อการรดน้ำต้นไม้ พืชสวนทางธรรมชาติล้วนต้องกระทำอย่างเป็นเวลาตามวัฏจักร ดังนั้นเราจึงคิดว่าจะมีสิ่งใดที่สามารถกำหนดเวลาตามฐานนาฬิกาจริง แล้วยังสามารถส่งสัญญาณออกมาควบคุมการเปิด-ปิด ของน้ำให้ทำงานตรงเวลาที่ผู้ประกอบการได้ตั้งค่ากำหนดไว้ ซึ่งประโยชน์ที่ได้รับคือจะสามารถประหยัดเวลาในการที่ต้องใช้คนไปทำ เราอาจใช้เทคโนโลยีตัวนี้เป็นตัวกำหนดการปิด-เปิด น้ำรดสวนหรือรดพืชผักในแปลงเกษตร

เมื่อนำหลักการที่ได้มาพัฒนากับอุตสาหกรรมการเกษตรขนาดใหญ่ หรือมีพื้นที่ในการรดน้ำมากๆแล้ว แนวคิดที่ได้นี้จึงเป็นแนวคิดที่ผู้ทำโครงงานคิดว่าน่าจะเป็นประโยชน์อย่างยิ่งในการทำงาน หรือการประกอบอาชีพในสังคมปัจจุบัน

2.2 อุปกรณ์ที่ใช้ในการทำโครงงาน

ผู้จัดทำโครงงานฉบับนี้จะอธิบายชิ้นส่วนต่างๆโดยละเอียดไว้ในบทที่ 3 ในส่วนของขั้นตอนการดำเนินงาน

2.3 ทฤษฎี

2.3.1 ภาษาสั่งงานคอมพิวเตอร์ (Computer Languages)

เนื่องจาก CPU จะถูกสั่งงานด้วยคำสั่งที่อยู่ในรูปของเลขฐานสองเท่านั้นแต่การสั่งงานด้วยเลขฐานสองเป็นเรื่องยากที่คนจะเข้าใจ ดังนั้นจึงจำเป็นต้องมีตัวกลางเข้ามาช่วยในการแปลงภาษาสั่งงานให้อยู่ในรูปของเลขฐานสอง ทำให้ผู้สั่งงานเขียนคำสั่งในรูปของภาษาที่ตัวเองเข้าใจ แล้วตัวกลางจะแปลงให้เป็นคำสั่งในรูปของเลขฐานสองให้คอมพิวเตอร์เข้าใจต่อไป

วิธีการแปลภาษา แบ่งออกเป็น 2 แบบคือ

1. **Compilation** จะมีตัวแปลภาษาคือ Compiler ทำหน้าที่แปลง source code ให้เป็น machine code ประมวลผลการทำงาน ทำให้โปรแกรมประมวลผลทำงานมีขนาดเล็กและทำงานได้เร็ว เนื่องจากการ compile ถูกแยกกระทำก่อนที่โปรแกรมจะทำงาน และ compiler สามารถวิเคราะห์โปรแกรมทั้งโปรแกรมก่อนถึงจะสร้าง machine code เช่น C Pascal

2. **Interpretation** จะมีตัวแปลภาษาคือ Interpreter ทำหน้าที่อ่าน source code ทีละบรรทัดแล้วแปลบรรทัดนั้นเป็น machine code และทำงานทันที ต่อจากนั้นจะอ่าน source code ในบรรทัดต่อไป แล้วทำเช่นเดิมจนจบโปรแกรม การกระทำดังกล่าวเรียกกว่า Interpret เป็นวิธีที่แปลภาษาและโปรแกรมทำงานเกิดขึ้นสลับกันไป โปรแกรมจึงทำงานช้ากว่าการcompiler แต่การแปลภาษาระหว่างโปรแกรมทำงานจะทำให้ได้ภาษาที่มีความยืดหยุ่นในการเขียนโปรแกรมมากกว่า เช่น Prolog Smalltalk

การจำแนกภาษาสั่งงาน ภาษาสั่งงานคอมพิวเตอร์สามารถแยกได้เป็นหลายแบบตามลักษณะงาน กล่าวคืออาจเป็นภาษาที่ออกแบบให้คนใช้ง่ายๆ เขียนโปรแกรมได้ง่าย หรืออาจจะออกแบบให้เครื่องเข้าใจง่ายๆ ทำให้แบ่งภาษาได้ออกเป็น 3 ระดับดังนี้

1. **ภาษาชั้นสูง (High Level Language)** เป็นภาษาเหมือนคำพูดภาษาอังกฤษ ทำให้ง่ายต่อการเข้าใจของคน ใช้ง่าย สะดวกต่อผู้ใช้ เช่น Prolog, Small Talk, Visual Basic เป็นต้น แต่ละคำสั่งจะแทนด้วยกลุ่มของเลขฐานสอง

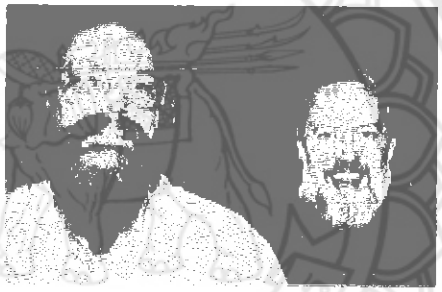
2. ภาษาชั้นกลาง (Medium Level Language) เป็นภาษาที่เหมือนคำพูดอังกฤษซึ่งเหมือนภาษาชั้นสูง แต่ในบางคำสั่งยังคล้ายภาษาชั้นต่ำอยู่ แต่ละคำสั่งยังแทนกลุ่มเลขฐานสองที่สั้นกว่าภาษาชั้นสูง เช่น ภาษา C

3. ภาษาชั้นต่ำ (Low Level Language) เป็นภาษาที่คอมพิวเตอร์เข้าใจง่าย อาจมีการใช้ตัวแปลภาษาหรือไม่มีก็ได้ มี 2 ภาษาคือ

3.1 ภาษา Assembly ต้องมีตัวแปลภาษาเรียกว่า Assembler โดยคำสั่งภาษา Assembly 1 คำสั่งจะแปลงเป็นภาษาเครื่อง 1 คำสั่ง

3.2 ภาษาเครื่อง (Machine Language) เป็นคำสั่งที่อยู่ในรูปของเลขฐานสอง ทำให้คอมพิวเตอร์เข้าใจได้ทันที ไม่ต้องมีการแปลงใดๆ

2.3.2 โปรแกรมภาษา C (C Programming)



รูปที่ 2.1 แสดงรูปของ Ken Thompson และ Dennis Ritchie ผู้คิดค้นภาษาซี

ภาษาที่ได้รับความนิยมสูงสุดตั้งแต่อดีตถึงปัจจุบันคือภาษา C ซึ่งถูกพัฒนาโดย Ken Thompson และ Dennis Ritchie ในปี 1972 โดยทั้งสองได้รับแนวคิดมาจากภาษา BCPL ซึ่งมีชื่อเรียกเล่นๆว่าภาษา B ดังนั้นภาษาที่ต่อจากภาษา B คือภาษา C เนื่องจาก C นั้นมีความเกี่ยวข้องกับระบบปฏิบัติการ UNIX ตั้งแต่เริ่มต้น ดังนั้นความนิยมในตัวระบบปฏิบัติการ UNIX จึงส่งผลกับความนิยมในตัวภาษา C ด้วยเช่นกัน จุดแข็งสำคัญของภาษา C คือโครงสร้างที่เหมาะสมสำหรับโปรแกรมขนาดใหญ่ โปรแกรมเล็กๆที่เขียนจากภาษา C สามารถนำมารวมกันให้เป็นโปรแกรมขนาดใหญ่ได้ และนอกจากนี้ภาษา C ยังสามารถเข้าถึง Hardware ได้ไม่ต่างจาก Assembly ทำให้ภาษา C ถูกจัดให้เป็นภาษาที่มีความสามารถไร้ขอบเขต

พื้นฐานเกี่ยวกับคอมพิวเตอร์ หน่วยสำคัญที่สุดของคอมพิวเตอร์ก็คือ หน่วยประมวลผล หรือที่เรียกกันว่า CPU โดยปกติ CPU จะมีภาษาของตัวเองที่เรียกว่า ภาษาเครื่อง (Machine Language) ซึ่งจะเป็นภาษาที่ประกอบไปด้วยเลขฐานสองมากมาย ดังนั้นการที่จะเขียน

โปรแกรมควบคุมการทำงานของคอมพิวเตอร์ โดยใช้ภาษาเครื่องโดยตรงนั้นจึงทำได้ยาก จึงได้มีการพัฒนาตัวแปรภาษาเครื่องที่เรียกว่า โปรแกรมภาษาระดับสูงขึ้นมา หรือที่เรียกว่า High Level Languages โดยได้อธิบายไว้ข้างต้น ซึ่งภาษาเหล่านี้ จะมีลักษณะรูปแบบการเขียน (Syntax) ที่ทำให้เข้าใจได้ง่ายต่อการสื่อสารกับผู้พัฒนา และถูกออกแบบมาให้ง่ายต่อการใช้งาน และจะเปลี่ยนคำสั่งจากผู้ใช้งาน ไปเป็นเป็นภาษาเครื่อง เพื่อที่จะควบคุมการทำงานของคอมพิวเตอร์ต่อไป ตัวอย่างของโปรแกรมภาษาระดับสูง ได้แก่ COBOL ใช้กันมากสำหรับโปรแกรมทางด้านธุรกิจ, Fortran ใช้กันมากสำหรับการพัฒนาโปรแกรมด้านวิทยาศาสตร์และวิศวกรรมศาสตร์ เพราะง่ายต่อการคำนวณ, Pascal มีใช้กันทั่วไป แต่เน้นสำหรับการพัฒนาเครื่องมือสำหรับการเรียนการสอน, C & C++ ใช้ทั่วไป ปัจจุบันมีผู้เลือกที่จะใช้กันอย่างแพร่หลาย, PROLOG เน้นหนักไปทางด้านงานประเภท AI และ JAVA ใช้ได้ทั่วไป ปัจจุบันเริ่มมีผู้หันมาสนใจกันมากและเพิ่มขึ้นอย่างรวดเร็ว

ก่อนที่จะลงมือพัฒนาโปรแกรมคอมพิวเตอร์ ขั้นแรก เราต้องศึกษารูปแบบความต้องการของโปรแกรมที่จะพัฒนา จากนั้นก็วิเคราะห์ถึงปัญหาตลอดจนวิธีการแก้ปัญหา จากนั้นจึงนำเอาความคิดในการแก้ปัญหาอย่างเป็นขั้นตอน ไปเขียนในรูปแบบของโปรแกรมภาษาในระดับสูง ซึ่งจะอยู่ในรูปแบบของ Source Program หรือ Source Code จากนั้นเราก็จะใช้ Complier ของภาษาที่เราเลือก มาทำการ Compile Source code หรือกล่าวง่ายๆ คือแปลง Source code ของเราให้เป็นภาษาเครื่องนั่นเอง ซึ่งในขั้นตอนนี้ ผลที่ได้ เราจะเรียกว่า Object code จากนั้น Complier ก็จะทำการ Link หรือเชื่อม Object code เข้ากับฟังก์ชันการทำงานใน Libraries ต่างๆ ที่จำเป็นต่อการใช้งาน แล้วนำไปไว้ในหน่วยความจำ แล้วเราก็จะสามารถ Run เพื่อดูผลของการทำงานโปรแกรมได้ หากโปรแกรมมีข้อผิดพลาด เราก็จะทำการแก้ หรือที่เรียกกันในภาษาคอมพิวเตอร์ว่า การ Debug นั่นเอง

แต่ทั้งนี้ภาษาC ก็มีข้อเสียที่มากมายเช่นกันเมื่อเทียบกับภาษาอื่นๆ เช่น Pascal หรือ FORTRAN เพราะ C เป็นภาษาที่มีโครงสร้างที่ซับซ้อน อีกทั้งมีเรื่องของ Pointer มาเกี่ยวข้อง ซึ่งเรื่อง Pointer นี้แม้แต่นักเขียนโปรแกรมมืออาชีพหลายๆคนยังปวดหัว

2.3.2.1 ข้อดีของภาษา C

1. ภาษา C ใช้ได้ในไมโครคอมพิวเตอร์ ตั้งแต่ขนาด 8 บิต 16 บิต 32 บิต มินิคอมพิวเตอร์ หรือ คอมพิวเตอร์ระดับเมนเฟรม มีการพัฒนาการใช้งาน เพื่อให้เป็นมาตรฐาน ไม่ขึ้นกับโปรแกรมจัดระบบงาน หรือ อุปกรณ์ทางอิเล็กทรอนิกส์ (ฮาร์ดแวร์)
2. ภาษา C มีหลายรุ่น มีผู้ผลิตต่างบริษัท แต่มีโครงสร้างคล้ายกัน และสามารถใช้ร่วมกันได้

3. ภาษา C มีความอ่อนตัว สามารถเจาะลงระดับลึกให้เข้ากับฮาร์ดแวร์ ทำงานได้รวดเร็ว และที่สำคัญ ภาษา C เป็นคอมไพเลอร์
4. ภาษา C เป็นภาษาที่มีโครงสร้าง

2.3.2.2 โครงสร้างภาษา C

```
#header
main ( )
{
- กำหนดตัวแปร
- กำหนดค่าตัวแปร
- ฟังก์ชัน ในรูปสเตตเมนท์
- การควบคุม
- คอมเมนท์
}
function a( )
{

....( มีส่วนประกอบเช่นเดียวกับ ฟังก์ชั่น main )
}
function b( )
```

หลักการเบื้องต้นในการเขียนดังแสดงด้านบน ส่วนฟังก์ชันการกำหนดพารามิเตอร์ต่างๆ จะมีรายละเอียดที่มาก ท่านผู้อ่านสามารถศึกษาได้กับหนังสือ การเขียนโปรแกรมภาษาซีได้ต่อไป

เมื่อเราเรียนรู้หลักการการเขียนโปรแกรมจาก 2.3.1ในขั้นตอนเบื้องต้นแล้ว ขั้นตอนต่อไปก็คือการเชื่อมต่อโปรแกรมที่เราออกแบบมาได้เข้ากับคอมพิวเตอร์ หรือไมโครโปรเซสเซอร์ เพื่อที่จะนำตัวโปรแกรมที่ได้ไปสั่งงานให้กับระบบทำงานต่อไป ซึ่งต่อไปจะกล่าวเรื่องของการอินเตอร์เฟสและการเชื่อมต่ออุปกรณ์เข้ากับคอมพิวเตอร์ดังนี้

2.3.3 การอินเตอร์เฟสกับคอมพิวเตอร์หรือไมโครโปรเซสเซอร์

การอินเตอร์เฟสกับคอมพิวเตอร์หรือไมโครโปรเซสเซอร์ คือ การทำงานติดต่อร่วมกันระหว่างซีพียูกับอุปกรณ์อื่นๆกับการโอนถ่ายข้อมูลระหว่างอุปกรณ์ต่างๆ นอกเหนือจากจะต้องทำงานติดต่อกับ RAM, ROM แล้วยังต้องมีการติดต่อกับอุปกรณ์ภายนอก ที่มีการส่งข้อมูลอินพุท/เอาท์พุทอีกทางหนึ่ง ซึ่งเป็นการเพิ่มประสิทธิภาพให้ระบบสมบูรณ์ ในระบบต่างของอุปกรณ์

อิเล็กทรอนิกส์ จะทำงานต่อเนื่องเป็นลูกโซ่ ดังเช่น การส่งรับข้อมูลจากซีพียูไปยังส่วนอื่นๆ เป็นต้น

การที่จะโอนย้ายข้อมูลทุกตัวนั้นจะต้องมีแหล่งที่ส่งข้อมูล และแหล่งที่รับข้อมูลสำหรับขบวนการเหล่านั้น จะมีส่วนที่สำคัญว่า ข้อมูลนั้นเป็น แอดเดรสหรือว่าเป็นดาต้า จะส่งไปยังจุดไหน ตัวอย่างเช่น ส่งไปยังหน่วยความจำ หรืออุปกรณ์อินพุต/เอาท์พุต และจะส่งเมื่อไร การทำงานเหล่านี้โดยทั่วๆไปจะต้องมีสัญญาณ ในการตรวจสอบอุปกรณ์ว่าพร้อมที่จะส่ง/รับข้อมูลหรือยังก่อนเสมอ เนื่องจากจุดที่ส่งและรับข้อมูล จะต้องมีสัญญาณตรวจสอบความพร้อมเสมอ เพื่อที่จะให้ข้อมูลที่เราใช้งานนั้นๆ เป็นระเบียบ ตัวอย่างเช่น ส่งข้อมูลจากซีพียูไปที่อุปกรณ์รอบข้าง เป็นต้น ซึ่งจุดรับส่งคู่หนึ่งๆ อาจจะเป็นระหว่างซีพียูด้วยกัน ซีพียูกับหน่วยความจำ ซีพียูกับอุปกรณ์รอบข้าง ระหว่างอุปกรณ์รอบข้างด้วยกัน หรือ ระหว่างหน่วยความจำกับอุปกรณ์รอบข้างก็ได้ สำหรับข้อมูลที่โอนย้ายไปมานั้นจะอยู่ในลักษณะของเลขฐานสอง ตัวอย่างเช่น $-->01101100_2$ ซึ่งเลขแต่ละตัวจะแทนด้วย 1 bit อาจเป็น 8 bit หรือ 16 bit ก็ขึ้นอยู่กับของระบบนั้นๆ ถ้าหากเป็นการต่อจากพอร์ตพีซีไม่ว่าจะเป็น Serial หรือ Parallel ในสัญญาณที่ส่งมาจะมีระบบแรงดันไฟฟ้า

2.3.4 การเชื่อมติดต่ออุปกรณ์เข้ากับคอมพิวเตอร์

1. **Connector** ที่อยู่ภายนอก ส่วนใหญ่จะอยู่ข้างหลังเครื่องคอมพิวเตอร์
   - พอร์ตต่อคีย์บอร์ด หรือ อาจเรียกกันว่า PS/2, mini-DIN
   - พอร์ตต่อเมาส์ หรือ อาจเรียกกันว่า PS/2, mini-DIN
   - พอร์ตต่อจอภาพ
   - พอร์ตต่ออนุกรม อาจเรียก Serial Port, Com port (Com1, Com2) ใช้ในระบบติดต่อสื่อสาร RS-232
   - พอร์ตต่อขนาน อาจเรียก Parallel Port, Printer port (LPT1, LPT2) ส่วนใหญ่จะใช้พ่วงต่อกับเครื่องพิมพ์ (Printer)
   - พอร์ตต่อจอยสติก ส่วนมากที่เห็นจะอยู่ที่เซาว์การ์ดเป็นส่วนใหญ่
   - พอร์ตต่อโมเด็ม ตัวคอนเน็กเตอร์จะเป็นประเภทเดียวกับสายสัญญาณโทรศัพท์
   - พอร์ตUSB (Universal Serial Bus) เป็นพอร์ตรุ่นใหม่ที่สามารถพ่วงอุปกรณ์ได้มากเริ่มวางตลาด เช่น เมาร์, คีย์บอร์ด, โมเด็ม, กล้องดิจิตอล เป็นต้น
   - พอร์ตเชื่อมต่อระบบเครือข่าย จะมากับการ์ดแลค์ เรียก พอร์ตRJ-45
   - พอร์ต SCSI (Small Computer System Interface) มักใช้เชื่อมต่อกับอุปกรณ์ที่ต้องการความเร็วสูง อาจจะได้ยินมาจากชนิดของฮาร์ดดิส

2. **Connector** ที่อยู่ภายนอก ส่วนใหญ่จะอยู่ภายในเครื่องคอมพิวเตอร์

- EIDE (Enhanced Integrated Drive Electronics) สายเชื่อมต่อกับฮาร์ดดิสก์
- SCSI (Small Computer System Interface) โดยมากจะมากับการ์ดที่เป็นแบบสกัซซี
- ฟลอปปี้ไดรฟ์ คอมพิวเตอร์ทุกเครื่องจะมีไว้ต่อฟลอปปี้ไดรฟ์
- คอนเน็กเตอร์อนุกรม มี 10 เข็มอยู่ที่แผงวงจรเมนบอร์ด
- คอนเน็กเตอร์ขนาน มี 26 เข็มอยู่ที่แผงวงจรเมนบอร์ด ถ้าต้องการขยาย พอร์ตขนาน ก็สามารถนำไปใส่เพิ่มได้
- Card I/O 8255 มาเสียบเข้าไปใน Slot ซึ่งเป็นประเภท Card PCI สำหรับ IC I/O 8255 จะมีพอร์ตเพิ่มขึ้นมาได้ 3 พอร์ตต่อ 1 ชุด IC 8255

2.3.5 ระบบที่ใช้ติดต่อสื่อสารข้อมูลของคอมพิวเตอร์

- USB (Universal Serial Bus) รวมถึง Firmware (IEEE-1348) เป็นระบบใช้ติดต่อสื่อสารข้อมูลแบบใหม่ ที่มีความเร็วสูง อีกทั้งเฉลียวฉลาดมากขึ้น ระหว่างเครื่องคอม พิวเตอร์ หรือคอมพิวเตอร์กับอุปกรณ์ภายนอก (Hardware) ซึ่ง USB ได้ถูกนำเข้ามาแทน การติดต่อแบบ RS-232 และ Centronics Printer Ports ดังจะเห็นได้จากอุปกรณ์ โมเด็ม หรืออุปกรณ์ตัวอื่นๆ เป็นต้น

- Fire wire มันได้ถูกออกแบบเพื่อรองรับการสื่อสารสำหรับข้อมูลที่เป็น สัญญาณภาพ, เสียง,วีดีโอรวมถึงขนาดบล็อคของอุปกรณ์ที่มีขนาดใหญ่

- Micro wire, SPI, I2C Interface การติดต่อสื่อสารเป็นแบบ Synchronous Serial เหมาะสำหรับใช้ในระยะสั้นๆซึ่งMicrocontrollerส่วนใหญ่แล้วจะติดต่อแบบนี้

- Ethernet หลายท่านคงจะรู้จักกัน เพราะใช้ติดต่อสื่อสารในระบบเครือข่ายหรือที่เรียกระบบแลนด์ ที่มีเครื่องคอมพิวเตอร์ต่อกันไปมากๆ เป็นระบบ ที่มีความเร็วสูง และความจุแต่ละอุปกรณ์ (Hardware) และโปรแกรม (Software) ซึ่งจะมีความซับซ้อน อีกทั้งราคาสูงกว่าระบบการติดต่อสื่อสารแบบอื่นๆในที่นำมากล่าวทั้งหมดนี้

- Centronics Parallel Printer Port Interface สามารถส่งข้อมูลได้หลายบิตสำหรับการส่งหนึ่งครั้ง ซึ่งมีความเร็วสูงสุด มักจะนิยมใช้สำหรับการติดต่อสื่อสาร ระหว่างพีซีกับเครื่องพิมพ์ (Printer) เครื่องแสกนเนอร์ เครื่องเก็บข้อมูลแบบภายนอก (Data Acquisition Devices) เป็นต้น

- IrDA (Interface Data Association) เป็นการสื่อสารข้อมูลแบบไร้สาย โดยใช้แสงอินฟราเรด ซึ่งใช้ได้ในระยะทางสั้นๆในที่สายเคเบิลไม่สามารถติดตั้งได้/เข้าไปไม่ถึง ที่สามารถพบเห็นในชีวิตประจำวัน ได้แก่ รีโมทโทรทัศน์หรือวีดีโอ,เมาส์หรือคีย์บอร์ดอินฟราเรด เป็นต้น

- MIDI (Musical Instrument Digital Interface) ใช้สำหรับการสื่อสารแบบอนาล็อกในเครื่องมือด้านเครื่องเสียง, เครื่องมือด้านดนตรี (ซินติไซเซอร์/เปอร์คัดชั่น/กีตาร์แอฟเฟ็ก), เครื่องมือ

ควบคุมเสียงในโรงภาพยนต์ (มิกเซอร์/อีเควอไรเซอร์/แอฟเฟ็กต่างๆ) ซึ่งมันจะใช้กระแสไฟประมาณ 5 mA ที่ความเร็ว 31.5 kb

2.3.6 รูปแบบการรับส่งข้อมูล

การรับส่งข้อมูลถ้าแบ่งตามรูปแบบการรับส่งจะแบ่งได้เป็น 3 ประเภทคือ

1. Simplex
2. Half Duplex
3. Full Duplex

2.3.6.1 การส่งผ่านแบบทิศทางเดียว (Simplex)

การส่งผ่านแบบทิศทางเดียว (Simplex) หมายถึง รูปแบบการส่งสัญญาณให้ด้านรับได้ฝ่ายเดียว โดยผู้รับไม่สามารถโต้ตอบผ่านทางการติดต่อได้เช่น การกระจายเสียงของวิทยุหรือสัญญาณโทรทัศน์ เป็นต้น

รูปที่ 2.2 แสดงการส่งผ่านแบบทิศทางเดียว (Simplex)

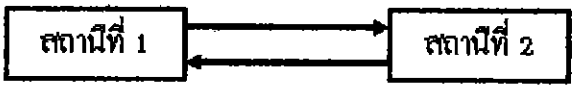
2.3.6.2 การส่งผ่านแบบสองทิศทางแต่ต่างเวลากัน (Half-Duplex)

การส่งผ่านแบบสองทิศทางแต่ต่างเวลากัน (Half-Duplex) หมายถึง รูปแบบสัญญาณที่สถานีทั้งสองฝ่ายสามารถรับและส่งสัญญาณระหว่างกันได้โดยกำหนดว่าต้องมีด้านใดด้านหนึ่งเป็นตัวรับเสมอเช่น การใช้วิทยุสมัครเล่นในการติดต่อสื่อสารเป็นต้น

รูปที่ 2.3 แสดงการส่งผ่านแบบสองทิศทางแต่ต่างเวลากัน (Half-Duplex)

2.3.6.3 การส่งผ่านแบบสองทิศทางในเวลาเดียวกัน (Full-Duplex)

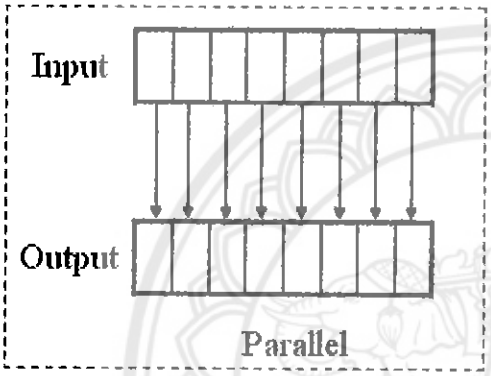
การส่งผ่านแบบสองทิศทางในเวลาเดียวกัน (Full-Duplex) หมายถึง รูปแบบการส่งสัญญาณที่ทั้งด้านส่งและด้านรับสามารถที่จะส่งสัญญาณในเวลาเดียวกันได้โดยไม่จำเป็นต้องสลับด้านกันด้วย เช่น การสนทนาทางโทรศัพท์ เป็นต้น



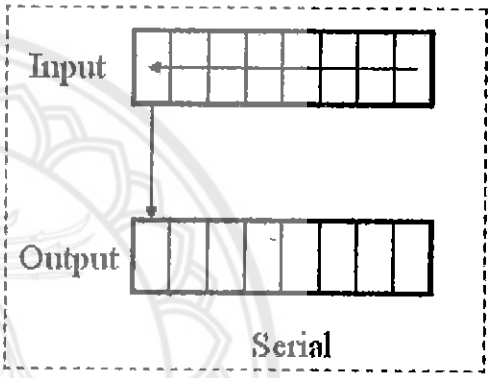
รูปที่ 2.4 แสดงการส่งผ่านแบบสองทิศทางในเวลาเดียวกัน (Full-Duplex)

2.3.7 การรับส่งข้อมูล (Data Communication) ปกติแล้วในการสื่อสารสำหรับรับส่งข้อมูลโดยทั่วไปจะแบ่ง ออกเป็น 2 ประเภทคือ

- แบบขนาน (Parallel Communications)
- แบบอนุกรม (Serial Communications)



รูปที่ 2.5 แสดงการรับส่งข้อมูลแบบขนาน



รูปที่ 2.6 แสดงการรับส่งข้อมูลแบบอนุกรม

2.3.7.1 การรับส่งข้อมูลแบบขนาน

การรับส่งข้อมูลแบบขนาน คือ การเคลื่อนย้ายข้อมูลทุกๆบิท ใน 1 คำในเวลาเดียวกัน ไม่ว่าจะเป็นการรับหรือการส่งก็ตาม เช่น เมื่อมีสัญญาณควบคุมเอาท์พุท ถูกส่งมาจากซีพียู ข้อมูล D0-D7 จะถูกส่งไปยังรีจิสเตอร์พร้อมกันทุกบิท ดังรูปที่ %

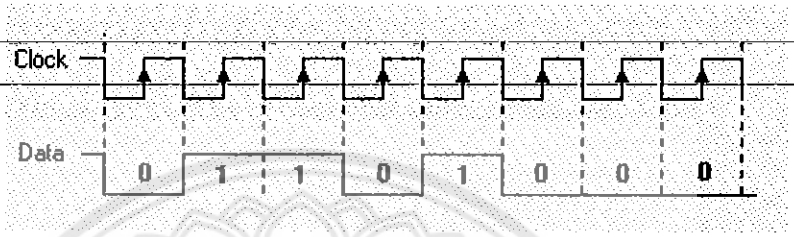
2.3.7.2 การรับส่งข้อมูลแบบอนุกรม

สำหรับการรับส่งข้อมูลแบบอนุกรม จะมีการรับส่งข้อมูลทีละบิต แล้วเวียนจนครบจำนวนบิทที่ต้องการรับส่ง ซึ่งข้อดีของระบบการรับส่งข้อมูลแบบอนุกรมคือจะใช้จำนวนสายน้อยกว่าการรับส่งแบบขนาน และสามารถสายสัญญาณรับส่งข้อมูลได้ระยะทางไกลมากกว่าแบบขนาน โดยทั่วไปการรับส่งข้อมูลแบบอนุกรมแบ่งออกเป็น 2 ประเภท คือ

1. Synchronous Serial Communication การรับส่งข้อมูลแบบขึ้นกับเวลา การสื่อสารแบบซิงโครนัสนี้จะใช้สัญญาณนาฬิกาควบคุมการรับส่งสัญญาณ เช่น สายคีย์บอร์ดคอมพิวเตอร์ โดยจะมีสายสัญญาณเส้นหนึ่งเป็นสายสัญญาณนาฬิกา ส่วนอีกเส้นหนึ่งเป็นสายของข้อมูล (และ

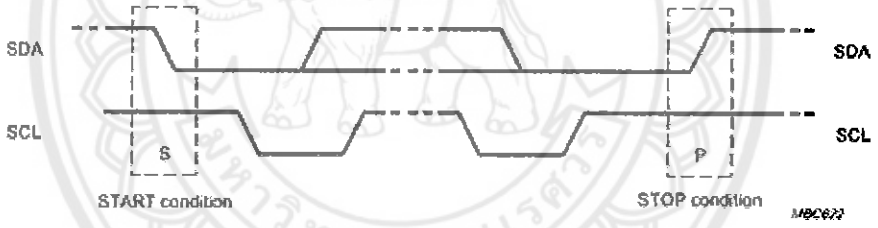
มักจะมีสาย กราวน์ด้วย) สำหรับการสื่อสารแบบซิงโครนัสนี้เหมาะสำหรับการทำงานในระยะใกล้ ข้อมูลที่จะส่งมีไม่มากนัก เพราะถ้าระยะทางไกลขึ้นจะทำให้สัญญาณนาฬิกามีปัญหา อีกทั้ง ต้องมีสายหลายเส้นทำให้สิ้นเปลืองมาก

ตัวอย่าง การรับส่งข้อมูลจะมีสัญญาณนาฬิกา ซึ่งเป็นตัวกำหนดจังหวะเวลาการส่งข้อมูล ร่วมอยู่ด้วยอีกเส้นหนึ่ง ใช้คู่กับสัญญาณข้อมูล ตัวอย่างเช่น การส่งสัญญาณจากคีย์บอร์ด เป็นต้น



รูป 2.7 แสดงตัวอย่างการรับส่งข้อมูลแบบซิงโครนัส

Synchronous Serial Communication



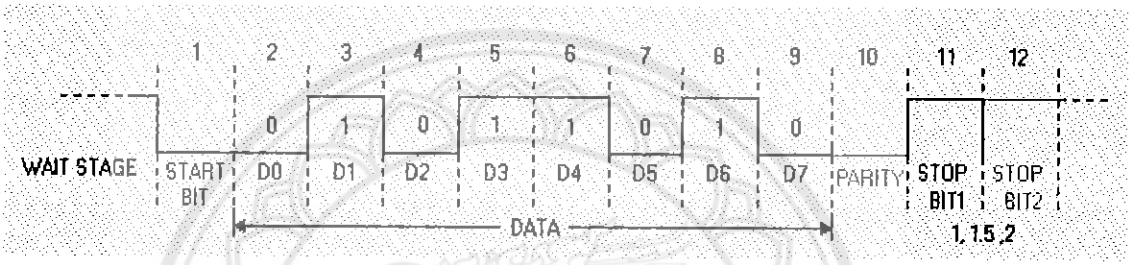
รูปที่ 2.8 แสดง Diagram การรับส่งข้อมูลแบบขึ้นกับเวลา

2. **Asynchronous Serial Communication** เป็นการรับส่งข้อมูลแบบไม่ขึ้นกับเวลา การสื่อสารแบบอะซิงโครนัสนั้น จะใช้สายสัญญาณเพียงตัวเดียวแต่จะใช้รูปแบบการส่งข้อมูล หรือBit-Pattern เป็นตัวกำหนดว่าส่วนไหนเป็นตัวเริ่มต้นข้อมูล ส่วนไหนเป็นตัวข้อมูล ส่วนไหนจะเป็นตัวตรวจสอบความถูกต้องของข้อมูล และส่วนไหนเป็นส่วนปิดท้ายของข้อมูล โดยต้องกำหนดให้สัญญาณนาฬิกาเท่ากันทั้งภาครับและภาคส่งซึ่งจะมีอุปกรณ์พิเศษที่เรียกว่าUART หรือ Universal Asynchronous Reciever/Trasmitter คอยควบคุมการรับและการส่งข้อมูล

2.1 รูปแบบข้อมูลแบบอะซิงโครนัส ประกอบด้วย 4 ส่วนคือ

1. บิตเริ่มต้น (Start bit) มีขนาด1 บิต
2. บิตข้อมูล (Data) มีขนาด 5,6,7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity bit) มีขนาด 1 บิตหรือไม่มี
4. บิตหยุด (Stop bit) มีขนาด 1, 1.5, 2 บิต

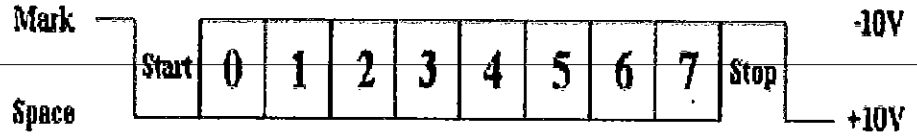
ตัวอย่าง



รูปที่ 2.9 แสดงตัวอย่างการส่งข้อมูลแบบอะซิงโครนัส

- เมื่อไม่มีการส่งข้อมูล ขา data จะมีสถานะเป็นโลจิก "1" หรือ สถานะหยุดรอ (Waiting stage)
- เมื่อเริ่มต้นส่งข้อมูลจะให้ขา data เป็นโลจิก "0" เป็นจำนวน 1 บิต เรียกว่าบิตเริ่มต้น (Start bit)
- จากนั้นก็จะเริ่มต้นส่งข้อมูล โดยส่งบิตต่ำไปก่อน (LSB)
- แล้วตามด้วยพาริตี้บิต (จะมีหรือไม่ก็ได้ ขึ้นอยู่กับการติดตั้งค่า ของทั้งสองฝ่าย)
- สุดท้ายตามด้วยโลจิก "1" อย่างน้อย 1 บิต (มีขนาด 1, 1.5, หรือ 2 บิต) เพื่อแสดงว่าสิ้นสุดข้อมูล

Asynchronous Serial Communication



รูปที่ 2.10 แสดงDiagram เป็นการรับส่งข้อมูลแบบไม่ขึ้นกับเวลา

2.2 หน้าที่สำคัญของสื่อสารแบบอะซิงโครนัส (Asynchronous)

รับสัญญาณ

1. เปลี่ยนสัญญาณ Input ที่เข้ามาแบบอนุกรมให้เป็นแบบขนาน
2. ตรวจสอบความผิดพลาดของสัญญาณที่รับ
3. ตัดสต็อปบิต (Stop Bit) และพาริตี้บิต (Parity Bit)
4. สัญญาณให้ CPU รับรู้ว่า ได้รับสัญญาณไว้แล้ว

ส่งสัญญาณ

1. เปลี่ยนสัญญาณแบบขนาดจาก CPU ค่อยๆ ทยอยส่งออกเป็นแบบอนุกรม
2. เพิ่มสต็อปบิตและพาริตี้บิต
3. เพิ่มสัญญาณควบคุมโมเด็มที่เชื่อมต่อ (ถ้ามี)

3. องค์ประกอบของการรับส่งข้อมูลแบบอนุกรม

3.1 Start Bit (ขนาด 1 บิต) จะใส่ที่จุดเริ่มต้นเสมอเพื่อเตือนอุปกรณ์ฝ่ายรับว่าข้อมูลกำลังจะมาถึง

3.2 Data Character (ขนาด 7 บิต หรือ 8 บิต) การส่งบิต ข้อมูลจะส่งเป็นกลุ่มๆ โดยทั่วไปจะส่งเป็น 7 บิต หรือ 8 บิต ซึ่งเพียงพอสำหรับการส่ง ASCII Word

3.3 Parity Bit (ขนาด 1 บิต) ใช้สำหรับตรวจสอบความถูกต้องของข้อมูลที่ส่งเราจะใส่บิตพาริตี้เข้าไป บิตพาริตี้มีหลายแบบดังนี้

- พาริตี้คู่ (Even Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกันทุกๆบิตของข้อมูลแล้วจะต้องมีจำนวน บิตที่เป็นเลข 1 เป็นเลขคู่ตัวอย่างเช่นข้อมูล 1000111 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 0

- พาริตี้คี่ (odd Parity) ค่าของบิตพาริตี้นี้เมื่อรวมกันทุกๆบิตของข้อมูลแล้วจะต้องมีจำนวน บิตที่เป็นเลข 1 เป็นเลขคี่ ตัวอย่างเช่นข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 1

- ไม่มีพาริตี้ (None) ถ้าตั้งค่าบิตพาริตี้เป็น None ทั้งภาครับและภาคส่งจะไม่มีการตรวจ สอบ บิตพาริตี้

3.4 Stop Bit (ขนาด 1 บิต หรือ 2บิต) เป็นบิตที่ส่งมาปิดท้ายข้อมูล

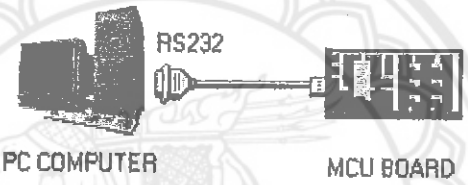
2.3.8 อัตราเร็วในการรับส่งข้อมูล (Baud rate)

อัตราเร็วในการรับส่งข้อมูลแบบอนุกรม การที่อุปกรณ์ 2 อย่างจะติดต่อสื่อสารกันได้นั้นจะต้องทำงานด้วยอัตราเร็วเท่ากัน ซึ่งอัตราเร็วในการสื่อสารแบบ อะซิงโครนัสคือค่าบลัดเรต (Baud Rate) มีหน่วยเป็นบิตต่อวินาทีซึ่งค่าอัตราเร็วในการสื่อสารแบบอนุกรมสำหรับมาตรฐาน RS-232 นั้นมี ใช้ ดังนี้ 110, 150, 300, 600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที

2.3.9 มาตราฐาน RS-232

RS-232 เป็นมาตรฐานการเชื่อมต่อข้อมูลแบบอนุกรมที่มีคนนิยมใช้มากที่สุด กำหนดโดย EIA (Electronics Industry Association) หรือสมาคมผู้ประกอบการอุตสาหกรรมอิเล็กทรอนิกส์ของอเมริกา ตั้งแต่ปี 1969 โดยมีจุดเริ่มต้นจากความต้องการที่จะกำหนดมาตรฐานการเชื่อมต่อระหว่างคอมพิวเตอร์กับโมเด็มในสมัยนั้น ตัวมาตรฐานจะกำหนดสิ่งที่เกี่ยวข้องกับการเชื่อมต่อนี้ด้วยกันทั้งหมด 4 หัวข้อหลักๆ ด้วยกันคือ

1. คุณสมบัติทางไฟฟ้าของสัญญาณ
2. คุณสมบัติทางกลของการเชื่อมต่อ ซึ่งหมายถึงตัวคอนเน็กเตอร์ (Connector) นั่นเอง
3. หน้าที่การทำงานของวงจรสำหรับแลกเปลี่ยนข้อมูล
4. มาตรฐานการเชื่อมต่อสำหรับระบบสื่อสารเฉพาะอย่าง



รูปที่ 2.11 แสดงภาพการเชื่อมต่อ MCU Board กับ PC Computer โดยใช้ RS-232

2.3.9.1 ช่องเชื่อมต่อแบบ RS232

เครื่อง PC ทุกเครื่องจะมีช่องเชื่อมต่อสื่อสารกับอุปกรณ์ภายนอกแบบอนุกรม ที่เรียกว่า คอมพอร์ท (Com Port = Communication Port) อยู่อย่างน้อยหนึ่งช่อง ทั่วไปแล้วจะมีสองช่อง โดยมีชื่อว่า com1 และ com2

ในอดีตช่อง com1 มักจะไว้ใช้เสียบเม้าส์ มาบัดนี้เม้าส์กลายเป็นแบบหัว OS/2 และ USB ไปเสียเป็นส่วนมากแล้ว ช่อง com1 จึงมักจะว่างอยู่ คนที่มีโมเด็มแบบภายนอก ก็อาจจะใช้โมเด็มต่อกับช่อง com1 นี้ได้ ช่อง com1 และ com2 สามารถใช้เชื่อมต่อกับอุปกรณ์คอมพิวเตอร์ และอุปกรณ์อิเลคโทรนิคต่างๆ ได้กว้างขวาง เช่นกระดานเขียนภาพ และเครื่องมือวัดเป็นต้น ที่เป็นไปได้เช่นนั้นก็เพราะอุปกรณ์ต่างๆ ดังกล่าวใช้มาตรฐานในการเชื่อมต่อแบบเดียวกัน คือมาตรฐาน RS232

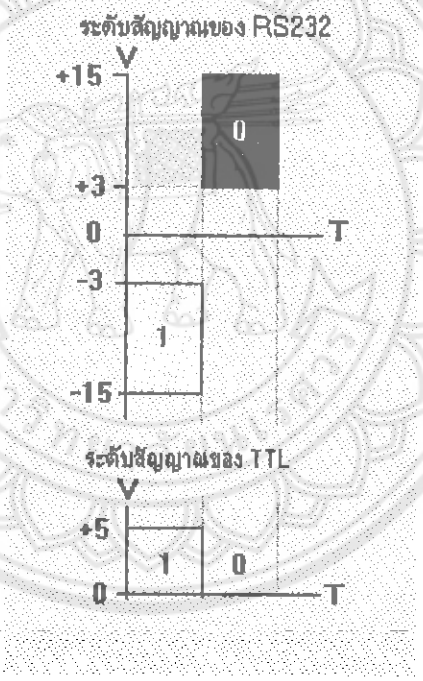
ข้อกำหนดในมาตรฐาน RS232 ครอบคลุมถึงลักษณะของสัญญาณ ระดับแรงดันกระแสไฟฟ้า และการจัดเรียงขั้วในหัวเชื่อมต่อด้วย ถ้าอุปกรณ์ใดมีช่องเชื่อมต่อที่ออกแบบมาตามข้อกำหนดนี้ ก็แน่ใจได้ว่าจะสามารถทำงานร่วมกันได้

ตัวขั้วเชื่อมต่อ (Connector) สำหรับ com1 ที่ติดอยู่ด้านหลังเครื่อง PC นิยมใช้แบบเก้าเข็ม หรือที่เรียกว่า DB9/M ส่วน com2 อาจจะเป็น DB9/M เช่นเดียวกับ com1 หรืออาจเป็นแบบ ตัวใหญ่ 25 เข็ม ที่เรียกว่า DB25/M ก็มีให้เห็นอยู่บ้างเหมือนกัน โดยเฉพาะในเครื่องเก่าๆ

2.3.9.2 ระดับสัญญาณของ RS232

สัญญาณรบกวนที่เกิดขึ้น ในสายนำสัญญาณ มักจะมีแรงดันเป็นบวก เมื่อเทียบกับกราวน์ โดยมีจุดประสงค์ คือ

- เพื่อป้องกันสัญญาณรบกวนนี้ จึงออกแบบแรงดัน ของโลจิก "1" เป็นลบ คืออยู่ในช่วง -3V ถึง -15V ส่วนแรงดัน ของโลจิก "0" อยู่ในช่วง +3V ถึง +15V
- และเหตุที่ ระดับสัญญาณ ของ RS232 อยู่ในช่วง +15V ถึง -15V ก็เพื่อให้ต่อสายสัญญาณไป ได้ไกลขึ้น



รูปที่ 2.12 แสดงระดับสัญญาณของ RS232 และระดับสัญญาณของ TTL

ดังนั้นจึงจำเป็นจะต้องมีวงจรเปลี่ยนระดับแรงดันของ RS232 มาเป็นระดับแรงดันของTTL ส่วนตัวอินพุต และ เอาต์พุตที่ใช้ในการติดต่อ หรือ ควบคุม ไมโครคอนโทรลเลอร์ ด้วยสัญญาณจะใช้สายอย่างน้อย 3 เส้น คือ

- สายส่งสัญญาณ TX
- สายรับสัญญาณ RX
- และสาย GND

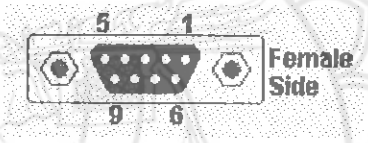
พอร์ตอนุกรมของ PC จะเป็นคอนเน็คเตอร์ (Connector) แบบ DB9 ตัวผู้ (Male)

พอร์ตอนุกรมของอุปกรณ์ภายนอก จะเป็นคอนเน็คเตอร์(Connector) แบบ DB9 ตัวเมีย (Female)

รูปที่ 2.13 แสดงภาพพอร์ตอนุกรมของ PC DB9 ตัวผู้ (Male)

รูปที่ 2.14 แสดงภาพพอร์ตอนุกรมของอุปกรณ์ภายนอก DB9 ตัวเมีย (Female)

เมื่อเราพิจารณา DB9 ตัวเมีย เมื่อมองจากด้านหลังจะนับขาที่หนึ่งถึงเก้า ตามรูปที่ 2.15 ด้านล่าง



รูปที่ 2.15 ภาพแสดงหัวต่อ RS232 แบบเก้าเข็ม

ตารางที่ 2.1 แสดงการอธิบายหน้าที่ของแต่ละช่องของ RS232

Pin 1	Received Line Signal Detector (Data Carrier Detect)
Pin 2	Received Data
Pin 3	Transmit Data
Pin 4	Data Terminal Ready
Pin 5	Signal Ground
Pin 6	Data Set Ready
Pin 7	Request To Send
Pin 8	Clear To Send
Pin 9	Ring Indicator

Pin ใน RS-232 สามารถอธิบายได้ดังนี้

Pin 1 DCD (carrier detect, หรือ data carrier detect)

Pin 2 Received Data ขารับสัญญาณ

Pin 3 Transmit Data ขาส่งสัญญาณ

Pin 4 DTR (data terminal ready) เป็นสายสำหรับการทำแฮนด์เชคจาก DTE ไปยัง DCE

Pin 5 Signal Ground ขากราวด์

Pin 6 DSE (data set ready) เป็นสายสำหรับการทำแฮนด์เชคจาก DCE ไปยัง DTE

Pin 7 RTS (request to send) เมื่ออุปกรณ์ DTE ต้องการส่งข้อมูลมันจะส่งสัญญาณที่ขานี้เป็นลอจิก "Low" ออกไป

Pin 8 CTS (clear to send) เป็นสายสัญญาณสำหรับการทำแฮนด์เชคจาก DCE ไปยัง DTE

Pin 9 RI (ring indicator) เป็นสายที่ใช้โดยโมเด็มเพื่อบอกว่าตัวมันเองได้รับสัญญาณเรียกเข้ามา

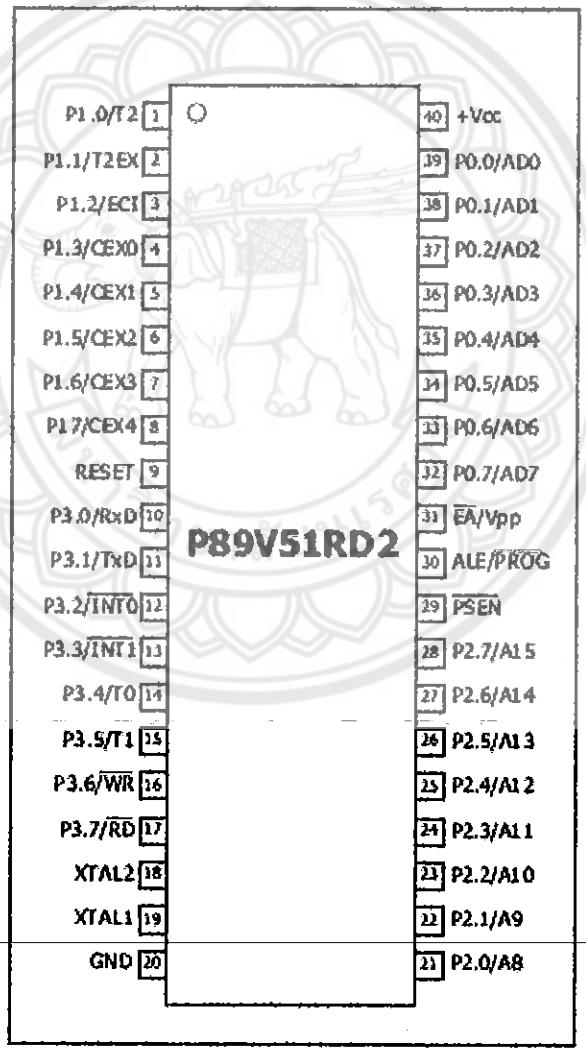
2.3.10 ไมโครคอนโทรลเลอร์ P89V51RD2

ไมโครคอนโทรลเลอร์เบอร์ P89V51RD2 เป็นตัวไมโครคอนโทรลเลอร์ที่ใช้กับบอร์ดการทดลองของโครงงาน ซึ่งมีหน่วยความจำแบบแฟลช (flash memory) ของบริษัท Philips Semiconductor โดยมีคุณสมบัติที่สำคัญดังนี้

- เป็นไมโครคอนโทรลเลอร์ 8 บิต ที่เข้ากันได้กับไมโครคอนโทรลเลอร์ MCS-51 พื้นฐาน
- มีหน่วยความจำโปรแกรมในตัวไมโครคอนโทรลเลอร์เป็นแบบแฟลช ลบและเขียนใหม่ได้ถึงหนึ่งหมื่นครั้ง ขนาดของหน่วยความจำ 64 กิโลไบต์
- หน่วยความจำข้อมูลแรมภายในมีขนาด 1 กิโลไบต์
- โปรแกรมข้อมูลลงในหน่วยความจำโปรแกรมแบบ ในระบบ (ISP : In-system programming)
- ความถี่สัญญาณนาฬิกาสูงสุด 40 MHz ในกรณีทำงานด้วยสัญญาณนาฬิกาภายใน 12 ลูกต่อแม-ชีนไซเกิลและ 20 MHz ในกรณีทำงานด้วยสัญญาณนาฬิกาภายใน 6 ลูกต่อแมชีนไซเกิล
- ขาพอร์ต 8 บิต 4 พอร์ต แบบกึ่งสองทิศทาง (quasi-bidirectional) เป็นได้ทั้งอินพุตและเอาต์พุต
- อุปกรณ์เพอริเฟอรัลภายในไมโครคอนโทรลเลอร์สามารถทำงานด้วยความเร็ว 12 สัญญาณนาฬิกาต่อแมชีนไซเกิลได้ แม้ว่าซีพียูจะทำงานด้วยความเร็ว 12 สัญญาณนาฬิกาภายในต่อแม-ชีนไซเกิล
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิต 3 ตัว (ไทเมอร์ 0, 1 และ 2)
- มีรีจิสเตอร์ตัวชี้ตำแหน่งข้อมูลหรือ DPTR 2 ตัว
- สามารถรองรับแหล่งกำเหนิดอินเตอร์รับได้ 8 ประเภท

- กำหนดนัยสำคัญของการตอบสนองอินเตอร์รัปต์ได้ 4 ระดับ
- สามารถติดต่อหน่วยความจำภายนอกได้สูงสุด 64 กิโลไบต์
- มีวอตซ์ด็อกไทเมอร์
- มีโมดูลวงจรนับโปรแกรมได้ (PCA : Programmable Counter Array) ซึ่งบรรจุวงจรตรวจจับสัญญาณ (Capture) และเปรียบเทียบสัญญาณ (Compare) มีวงจรมอดูเลชั่นทางความกว้างพัลส์(PWM) และวอตช์ด็อกไทเมอร์ (Watchdog Timer)

การจัดการขาของ P89V51RD2 แสดงในรูปที่2.16 ส่วนตารางที่ 2.2 แสดงรายละเอียดขั้นต้นของขาต่อใช้งานของ P89V51RD2



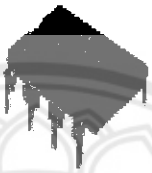
รูปที่ 2.16 แสดงการจัดการขาของ P89V51RD2

ตารางที่ 2.2 แสดงรายละเอียดขั้นต้นของขาต่อใช้งานของ P89V51RD2

ชื่อขา	ขาที่	ชนิด	หน้าที่และการทำงาน
Vcc	40	อินพุต	ต่อไฟเลี้ยง +5V
GND	20	อินพุต	ต่อกราวด์
P0.0-P0.7	39-32	อินพุต/เอาต์พุต	- ใช้งานเป็นขาพอร์ตอินพุตเอาต์พุต ถ้าต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ทำให้มีสถานะเลอย (float) ค่าอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ - ใช้ในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยใช้การมัลติเพล็กซ์เข้าช่วยเพื่อสลับการทำงานให้เป็นได้ทั้งขาติดต่อแอดเดรสและขาข้อมูลในการติดต่อกับหน่วยความจำภายนอก
P1.0-P1.7	1-8	อินพุต/เอาต์พุต	- ใช้งานเป็นขาพอร์ตอินพุตเอาต์พุตสำหรับใช้งานทั่วไป เฉพาะขา P1.5 ถึง P1.7 สามารถรับกระแสได้สูง 16mA ต่อขา - เป็นขาสัญญาณของไทเมอร์ 2 และขาสัญญาณของโมดูล PCA ดังมีรายละเอียดต่อไปนี้ T2 (P1.0 : ขา 1) เป็นขาอินพุตสำหรับนับค่าของไทเมอร์ 2 และขาเอาต์พุตสัญญาณนาฬิกาโปรแกรมแบบได้ T2EX (P1.1 : ขา 2) เป็นขาอินพุตสำหรับควบคุมการทำงานของไทเมอร์/เคาน์เตอร์ 2 ECI (P1.2 : ขา 3) เป็นขาอินพุตสัญญาณนาฬิกาจากภายนอกสำหรับโมดูล PCA CEX0 (P1.3 : ขา 4) เป็นขาอินพุตเอาต์พุตภายนอกของวงจรตรวจจับและเปรียบเทียบสัญญาณสำหรับ PCA โมดูล 0 CEX1 (P1.4 : ขา 5) เป็นขาอินพุตเอาต์พุตภายนอกของวงจรตรวจจับและเปรียบเทียบสัญญาณสำหรับ PCA โมดูล 1 CEX2 (P1.5 : ขา 6) เป็นขาอินพุตเอาต์พุตภายนอกของวงจรตรวจจับและเปรียบเทียบสัญญาณสำหรับ PCA โมดูล 2 CEX3 (P1.6 : ขา 7) เป็นขาอินพุตเอาต์พุตภายนอกของวงจรตรวจจับและเปรียบเทียบสัญญาณสำหรับ PCA โมดูล 3 CEX4 (P1.7 : ขา 8) เป็นขาอินพุตเอาต์พุตภายนอกของวงจรตรวจจับและเปรียบเทียบสัญญาณสำหรับ PCA โมดูล 4
P2.0-P2.7	21-28	อินพุต/เอาต์พุต	- ใช้งานเป็นขาพอร์ตอินพุตเอาต์พุตสำหรับใช้งานทั่วไป - ใช้ต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15) เมื่อติดต่อด้วย
P3.0-P3.7	10-17	อินพุต/เอาต์พุต	- ใช้งานเป็นขาพอร์ตอินพุตเอาต์พุตสำหรับใช้งานทั่วไป - ใช้งานเป็นขาพอร์ตหน้าที่พิเศษ ดังมีรายละเอียดต่อไปนี้ RxD (P3.0 : ขา 10) ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม TxD (P3.1 : ขา 11) ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม INT0 (P3.2 : ขา 12) ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 0 INT1 (P3.3 : ขา 13) ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 T0 (P3.4 : ขา 14) ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 0 T1 (P3.5 : ขา 15) ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 $\overline{WR}$ (P3.6 : ขา 16) ใช้เป็นขาสัญญาณ $\overline{WR}$ ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก $\overline{RD}$ (P3.7 : ขา 17) ใช้เป็นขาสัญญาณ $\overline{RD}$ ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก
RESET	9	อินพุต	ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณลอจิก "1" อย่างน้อยเป็นเวลา 2 แมชีนไซเกิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ
ALE	30	เอาต์พุต	*Address Latch Enable* ออกมาทุกๆ แมชีนไซเกิล อย่างไรก็ตาม สามารถดิสเอเบิลสัญญาณพัลส์นี้ได้ โดยการเซตบิต 0 ของรีจิสเตอร์ AUXR
$\overline{PSEN}$	29	เอาต์พุต	*Program Store Enable* : ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก ไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้ง นอกจากนี้ยังใช้ประกอบในการอ่าน-เขียนข้อมูลในหน่วยความจำโปรแกรมด้วยกระบวนการ ISP - สำหรับเบอร์ P89C51RD+ ให้ต่อขานี้ลงกราวด์ แล้วป้อนไฟ +12V (±0.5V) เข้าที่ขา $\overline{EA}$/Vpp - สำหรับเบอร์ P89C51RD2 ให้ต่อขานี้ลงกราวด์, ป้อนลอจิก "1" เข้าที่ขา P2.7 และป้อนแรงดัน +5V เข้าที่ขา $\overline{EA}$/Vpp
$\overline{EA}$/Vpp	31	อินพุต	*External Access enable/Programming voltage Input* : ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ "0" เลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก "1" เลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายใน นอกจากนี้ ที่ขานี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันสำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ - สำหรับเบอร์ P89C51RD+ ต้องการแรงดัน +12V (±0.5V) - สำหรับเบอร์ P89C51RD2 ต้องการแรงดัน +5V
XTAL1	19	อินพุต	ขาอินพุตรับสัญญาณจากวงจรขยายออสซิลเลเตอร์ (ขา XTAL2) และจากภายนอก ในการใช้งานปกติ ขานี้และขา XTAL2 ต่อเข้ากับคริสตอลและตัวเก็บประจุชุดเซรามิกค่าน้อยๆ
XTAL2	18	เอาต์พุต	ขาเอาต์พุตของวงจรขยายออสซิลเลเตอร์ภายในไมโครคอนโทรลเลอร์ ในการใช้งานปกติ ขานี้และขา XTAL1 ต่อเข้ากับคริสตอลและตัวเก็บประจุชุดเซรามิกค่าน้อยๆ

2.3.11 ไอซีฐานเวลาจริง DS 1307 (Real Time Controller DS1307)

ไอซีฐานเวลาจริง DS 1307 (Real Time Controller DS1307) เป็นอุปกรณ์ที่ใช้ในการสร้างฐานเวลาจริงให้กับไมโครคอนโทรลเลอร์ ใช้สายสัญญาณในการติดต่อสองเส้นแบบบัส I^2C ให้ข้อมูลเกี่ยวกับเวลาอย่างครบถ้วน เช่น วันที่ เดือน ปี รวมไปถึงเวลาแสดงค่าเป็น วินาที นาที และชั่วโมง ทั้งแบบ 24 ชั่วโมง และแบบ 12 ชั่วโมง พร้อมระบุค่า AM/PM นอกจากนี้ยังมีหน่วยความจำแบบนอนโวลาไทล์ (nonvolatile RAM) อีก 56 ไบต์ซึ่งสามารถใช้เก็บข้อมูลได้ โดยลักษณะภายนอกแสดงดังรูปข้างล่าง



รูปที่ 2.17 แสดงลักษณะภายนอกของไอซี DS 1307

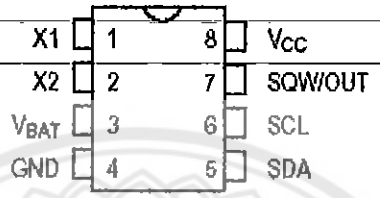
2.3.11.1 คุณสมบัติของไอซี DS 1307

การจัดการขาของไอซี DS 1307 จัดการเชื่อมต่อในระบบบัส I^2C โดยทำงานเป็นอุปกรณ์สเลฟ(Slave)เสมอ ส่วนประกอบหลักที่สำคัญคือ วงจรออสซิเลเตอร์ถือเป็นหัวใจหลักของไอซี เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลฐานเวลาจริง มีการเก็บค่าของเวลาไว้ในหน่วยความจำนอนโวลาไทล์แรม(nonvolatile RAM) ซึ่งมีขนาดรวม 64 ไบต์แต่จัดสรรให้ใช้เก็บข้อมูลเวลา 8 ไบต์และเป็นหน่วยความจำสำหรับเก็บข้อมูลทั่วไป สำหรับผู้ใช้งานอีก 56 ไบต์ วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสภาวะของไฟเลี้ยงไอซี หากไฟเลี้ยงต่ำกว่า 1.25 x Vbat ก็จะควบคุมให้ DS 1307 หยุดการทำงาน ทำให้ไม่สามารถติดต่อกับ DS 1307 ได้ ดังนั้นในการใช้งานต้องระมัดระวังอย่าให้ไฟเลี้ยงต่ำกว่า 1.25 x Vbat หรือประมาณ 3.75 V ถ้าหากไฟเลี้ยงมีค่าต่ำกว่า Vbat ไอซี DS 1307 จะเข้าสู่โหมดสำรองข้อมูลกระแสต่ำทันที แต่วงจรสร้างฐานเวลายังคงทำงานเพื่อให้ค่าของเวลาเดินไปอย่างไม่ผิดพลาด ซึ่งจากข้อมูลข้างต้นสามารถสรุปได้ดังนี้

1. DS 1307 เป็นไอซีแบบ 8 ขา กินพลังงานต่ำมาก โดยกินกระแสน้อยกว่า 500 นาโนแอมป์ ในโหมดแบตเตอรี่สำรอง
2. นับสัญญาณนาฬิกาเป็นวินาที นาที ชั่วโมง วัน วันที่ เดือน และปีได้อย่างถูกต้องไปถึง ค.ศ. 2010
3. มีหน่วยความจำภายในแบบนอนโวลาไทล์ (Nonvolatile RAM) ขนาด 56 ไบต์ ไว้เก็บข้อมูลเวลาภายใน

4. เชื่อมต่อกับไมโครคอนโทรลเลอร์โดยใช้บัสแบบ I^2C
5. สามารถโปรแกรมให้สร้างคลื่นรูปสี่เหลี่ยม (square wave) ออกมาได้
6. สามารถเลือกใช้รุ่นที่ใช้งานในอุตสาหกรรมได้โดยสามารถใช้อุณหภูมิได้ในช่วง –40 ถึง +85

2.3.11.2 รายละเอียดของขาต่อใช้งาน



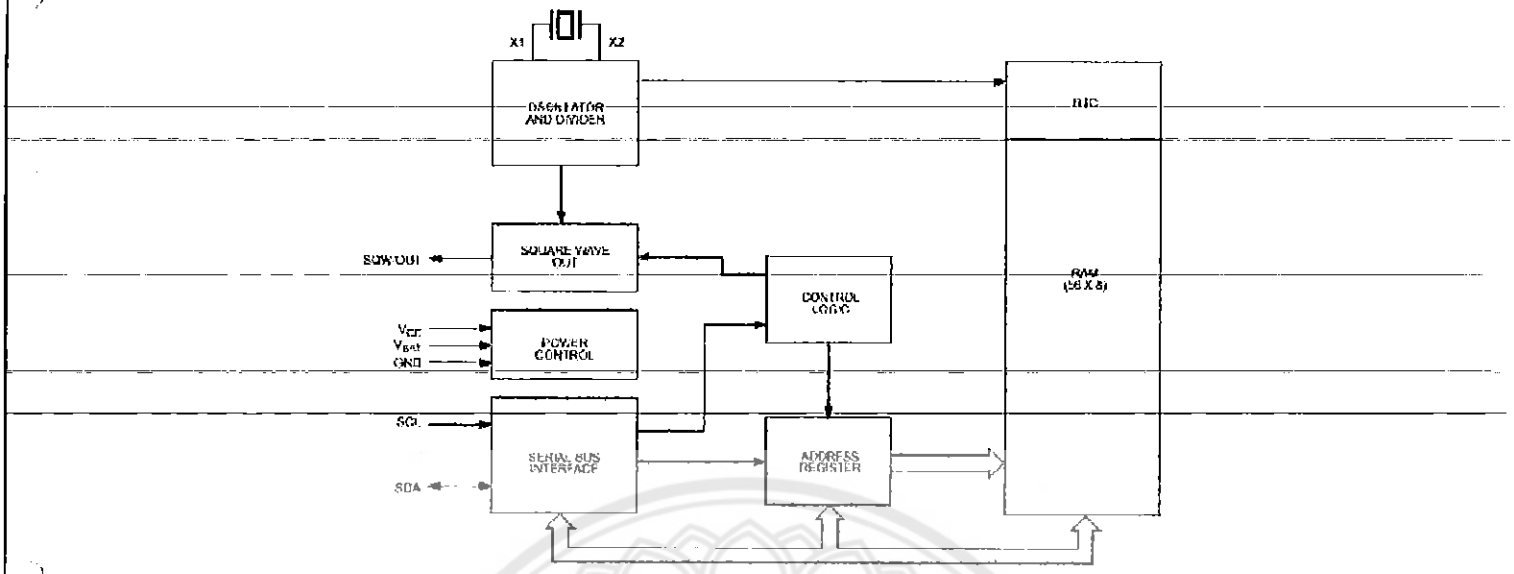
รูปที่ 2.18 แสดงขาต่อใช้งานของ DS 1307

รายละเอียดการจัดการของขาสัญญาณ สามารถอธิบายได้ในตารางที่

ตารางที่ 2.3 รายละเอียดการจัดการของขาสัญญาณ

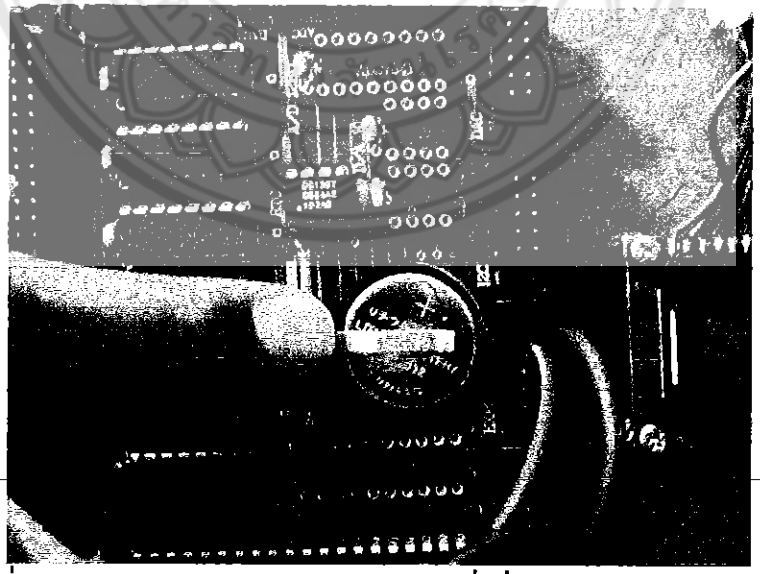
ตำแหน่ง	สัญลักษณ์	รายละเอียด
1	X1	ขาต่อ Xtal 32.768 KHz
2	X2	ขาต่อ Xtal 32.768 KHz
3	VBAT	ขาต่อแบตเตอรี่ 3 V
4	GND	ขาต่อกราวด์
5	SDA	สัญญาณ Data แบบอนุกรม
6	SCL	สัญลักษณ์ Clock แบบอนุกรม
7	SQW/OUT	สัญญาณเอาท์พุทเป็นแบบสี่เหลี่ยมหรือเป็นลอจิก
8	Vcc	ต่อเข้ากับแหล่งจ่ายไป

ส่วนประกอบหลักของ DS 1307 สามารถแสดงแบบโครงสร้างได้ดังรูป 2.19



รูปที่ 2.19 แสดงถึงส่วนประกอบหลักของ DS1307

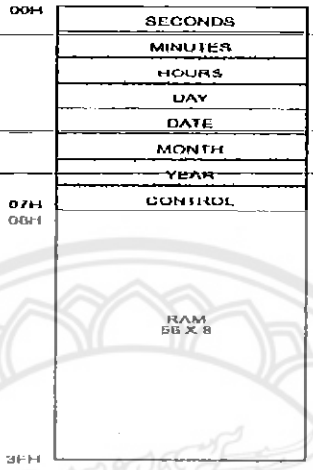
บอร์ด CP-SPI/RD2 V3.0 ที่เราใช้ทำโครงงาน จะใช้ DS1307 ของ DALLAS เป็น RTC ซึ่งสามารถแสดงค่าเวลาได้เป็น ปี เดือน วันที่ของเดือน วันที่ของสัปดาห์ ชั่วโมง นาที และ วินาที โดยภายในยังมี RAM อีก 56 ไบต์ เพื่อใช้เก็บข้อมูล โดยทุกครั้งที่มีการใช้งาน DS1307 เราจะต้องใส่ BAT 3 V เข้าไปด้วย



รูปที่ 2.20 แสดงตำแหน่งของการใส่แบตเตอรี่ เมื่อต้องการใช้งาน DS1307

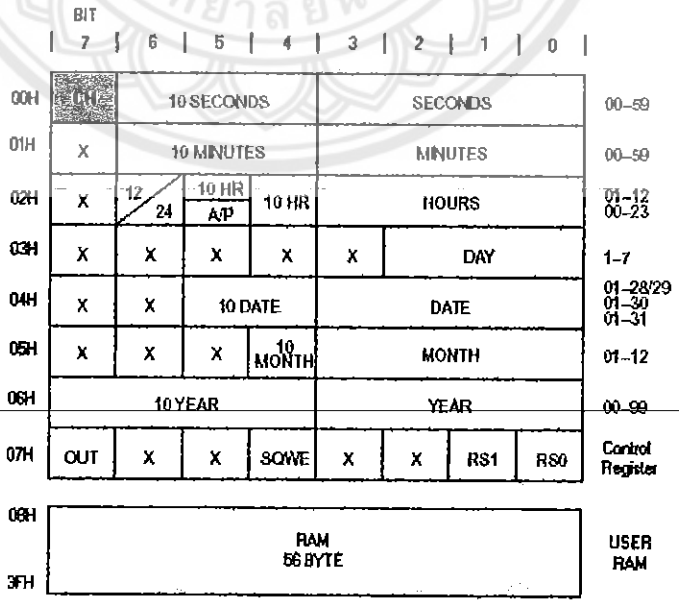
DS1307 มีการสื่อสารข้อมูลกับไมโครคอนโทรลเลอร์โดยผ่านทาง I^2C ซึ่งใช้สายสัญญาณเพื่อ สื่อสาร 2 เส้นคือสายสัญญาณ DATA (SDA) และสายสัญญาณ CLOCK (SCL)

2.3.11.3 RTC and RAM Address Map เป็นตำแหน่งที่อยู่ของนาฬิกาและ RAM ของตัว DS1307 แสดงในรูปที่ 2.20 โดยตำแหน่งของนาฬิกาอยู่ที่ 00H ถึง 07H โดยตำแหน่งของ RAM อยู่ที่ตำแหน่ง 08H ถึง 3FH เมื่อมีการเข้าถึงหน่วยความจำแบบ Multibyte ถ้าการเข้าถึงหน่วยความจำถึง 3FH และจะกลับไปที่ 00 H ซึ่งเป็นตำแหน่งเริ่มต้น



รูปที่ 2.21 แสดงตำแหน่งของนาฬิกาและ RAM ของ DS1307

การจัดสรรหน่วยความจำภายในของ DS 1307 ที่ใช้เก็บข้อมูลวันเวลาจะมีจำนวน 7 ไบต์ โดยตำแหน่ง 00H จะเก็บเวลาเป็นวินาที ตำแหน่ง 01 เก็บเวลาเป็นนาที โดยข้อมูลที่จัดเก็บจะอยู่ในรูปของรหัส BCD



รูปที่ 2.22 แสดงโครงสร้างของหน่วยความจำภายในของ DS1307

ตารางที่ 2.4 แสดง Address Nameและ Description ของ DS 1307

Address	Name	DESCRIBTION
00H	SECOND	เก็บค่าวินาที โดยบิตที่สำคัญ คือบิต 7 CH (Clock Halt) เมื่อบิต 7 นี้เป็น 1 จะทำ ให้สัญญาณ นาฬิกา หยุดเดิน
01H	MINUTES	เก็บค่านาที 00-59
02H	HOUR	เก็บค่าชั่วโมง เมื่อบิต 6 เป็นบิตที่กำหนดการแสดงผแบบ 12/24 ช.ม เมื่อบิต 6 =1 จะเก็บค่า 01-12 ,เมื่อบิต 6 =0 จะเก็บค่า 01-24
03H	DAY	เก็บค่าวัน 1-7
04H	DATE	เก็บค่าที่ 01-31
05H	MONTH	เก็บค่าเดือน 01-12
06H	YEAR	เก็บค่าปี 00-99
07H	Control Register	รีจิสเตอร์ควบคุมการทำงานของขา SQW \OUT
08H-3FH		หน่วยความจำใช้งานทั่วไป 56 byte

ถ้าบิต 4 SQWE = 1 กำหนดให้ขา SQW \OUT กำเนิดความถี่ ตามความถี่ที่กำหนดโดย RS1, RS0

ถ้าบิต 4 SQWE = 0 กำหนดให้ขา SQW \OUT จะมีค่าตามบิต 7 คือขา OUT

ซึ่งการกำหนดความถี่ของคลื่นนั้นสามารถกำหนดขาได้ดังนี้

ตารางที่ 2.5 แสดงการกำหนดความถี่ที่ขา RS1 และ RS0 ของ DS 1307

RS1	RS0	ความถี่
0	0	1 Hz
0	1	4.096 kHz
1	0	8.192 kHz
1	1	32.768 kHz

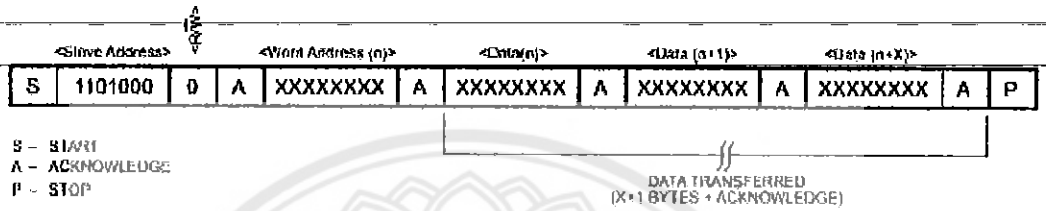
2.3.11.4 DS1307 Timekeeping Register

Control Register เป็นรีจิสเตอร์ควบคุมเป็นส่วนที่ใช้ควบคุมขาสัญญาณ SQW/OUT ถ้าบิต 4 หรือบิต SQWE มีลอจิก 1 จะทำให้ขา SQW/OUT สร้างสัญญาณพัลส์ออกมาตามความถี่ที่กำหนดในบิต RS0และ RS1แต่ถ้าบิต 4 มีลอจิก 0 จะทำให้ขา มีลอจิกตามบิต 7 หรือบิต OUT

Squarewave Output Frequency เป็นการกำหนดค่าความถี่ที่ต้องการให้ออกที่ขา SQWE มีลอจิก 1 สามารถกำหนดได้จากบิทRS0 และRS1 ตามตารางข้างบน

2.3.11.5 การเขียนข้อมูลลง DS1307 (Data Write)

การเขียนข้อมูลลงใน DS1307 จะถูกนำมาใช้เมื่อต้องการตั้งเวลา การกำหนดให้สัญญาณ Pulse ออกที่ขา SQW/OUT หรือแม้กระทั่งการเขียนเข้าไปเก็บไว้ในหน่วยความจำส่วน RAM ที่อยู่ภายใน DS1307แนวทางการเขียนข้อมูลเข้าใน DS1307 นั้นจะใช้หลักการของการสื่อสารข้อมูลแบบ I^2C คือการเขียนจะต้องเริ่มต้นจากที่ไมโครคอนโทรลเลอร์ส่งเงื่อนไข Start ไปให้กับ DS1307 แล้วจึงส่งข้อมูลต่อ-ไปอีก 1 ไบต์ โดยข้อมูลไบต์นี้ใช้เก็บ Address และบิท R/W



รูปที่ 2.23 แสดง Diagram การเขียนข้อมูลลง DS1307

DS1307 ถูกกำหนดให้มีตำแหน่ง Address อยู่ที่ 110100B (ขนาด 7 บิท) ซึ่งในการเขียนข้อมูลจะต้องกำหนดให้บิต R/W เป็น 0 เพราะฉะนั้นเมื่อรวม Address ขนาด 7 บิท เข้ากับบิทR/W จะได้ข้อมูลไบต์ที่จะต้องส่งเป็น 11010000B

หลังจากที่ส่งไบต์ Address ไปให้กับ DS1307 แล้วถ้า DS1307 รับข้อมูลได้ถูกต้อง ตัวDS1307 จะส่ง บิท ACK ออกมาให้กับไมโครคอนโทรลเลอร์ ซึ่งบิท ACK ที่ DS1307 ส่งออกมาให้กับตัวไมโครคอนโทรลเลอร์จะเป็น 0 แต่ถ้าไมโครคอนโทรลเลอร์ตรวจสอบบิทนี้แล้วเป็น 1 แสดงว่า DS1307 ไม่ได้รับตำแหน่ง Address ที่ถูกต้อง ต้องกลับไปเริ่มขบวนการ Start ใหม่

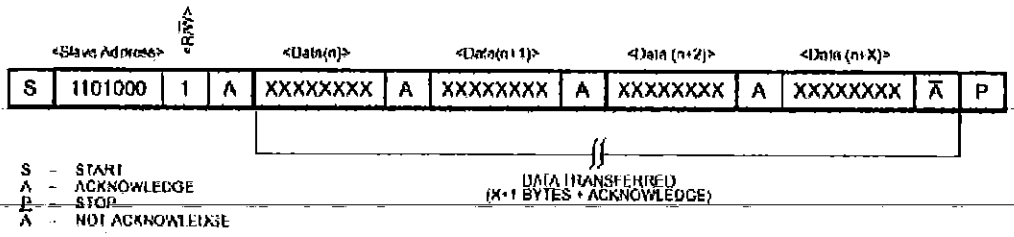
หลังจากที่ไมโครคอนโทรลเลอร์ได้รับ บิท ACK ตอบกลับออกมาจาก DS1307 แล้ว ตัวไมโครคอนโทรลเลอร์ จะต้องส่งข้อมูลไปอีก 1 ไบต์โดยข้อมูลไบต์นี้จะเป็นตำแหน่งหน่วยความจำภายใน DS1307 ซึ่งอยู่ที่ตำแหน่ง0-3F การส่งข้อมูลไบต์นี้จะเหมือนกับการส่งข้อมูลในไบต์แรกคือเมื่อ DS1307ได้รับข้อมูลถูกต้องแล้ว ตัว DS1307 จะส่งบิท ACK กลับออกมา

หลังจากที่ DS1307 ได้รับตำแหน่งของหน่วยความจำที ต้องการเขียนเรียบร้อยแล้วไมโครคอนโทรลเลอร์สามารถส่งข้อมูลที่ต้องการเขียนไปยังตำแหน่งดังกล่าวได้เลย ซึ่งเมื่อส่งข้อมูลได้ออกไป 1ไบต์ ตัว DS1307 จะเพิ่มค่าตำแหน่ง 1 ค่า และส่งบิท ACK กลับออกมา

ถ้าต้องการเขียนข้อมูลในตำแหน่งที่เรียงกันไป สามารถทำได้โดยส่งข้อมูลต่อไปเรื่อยๆจนครบแล้วจึงส่งเงื่อนไขSTOP เพื่อหยุดการทำงาน แต่ถ้าหากต้องการเปลี่ยนตำแหน่งที่จะติดต่อหรือไม่ทราบว่า ตัวชี้ตำแหน่งของ DS1307 ชี้อยู่ที่ตำแหน่งใด ก็สามารถทำได้โดยการส่งตำแหน่งใหม่ให้กับ DS1307

2.3.11.6 การอ่านข้อมูลจาก DS1307 (Data Read)

การอ่านข้อมูลจาก DS1307 จะมีลักษณะคล้ายกับการเขียนข้อมูลให้กับ DS1307 โดยจะต้องให้บิท R/W เป็น 1 เพราะฉะนั้นข้อมูลไบต์ Address จะมีค่าเป็น 11010001B



รูปที่ 2.24 แสดง Diagram การอ่านข้อมูลออกจาก DS1307

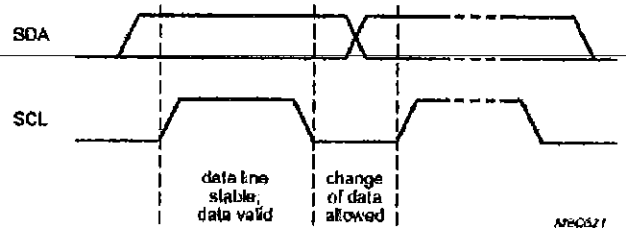
2.3.12 ระบบบัสแบบ I2C

ระบบบัสแบบ I^2C ย่อมาจาก Inter – IC Communication พัฒนาโดยห้องวิจัยของ Phillips ในประเทศ Netherlands เมื่อปี ค.ศ. 1980 ดังนั้นอุปกรณ์หลายๆตัวที่มีการสื่อสารแบบ I^2C จึงถูกผลิตออกมาจากบริษัท Phillips การสื่อสารข้อมูลด้วยระบบ I^2C บัสเป็นอีกมิติหนึ่งของการสื่อสารระหว่างไมโครคอนโทรลเลอร์ และอุปกรณ์ประกอบร่วม

ระบบ I2C บัสเป็นการสื่อสารแบบ 2 ทาง โดยใช้สายสัญญาณในการสื่อสารเพียง 2 เส้น โดยสายที่ใช้สื่อสารนี้คือ SDA ซึ่งเป็นสายสัญญาณข้อมูล และ SCL ซึ่งเป็นสายสัญญาณ Clock โดยสัญญาณทั้ง 2 เส้นนี้สามารถต่อเข้ากับอุปกรณ์ได้มากกว่า 1 ตัว ซึ่งทำให้การใช้งานไมโครคอนโทรลเลอร์มีประสิทธิภาพมากในแง่ของความสิ้นเปลืองพอร์ตของไมโครคอนโทรลเลอร์

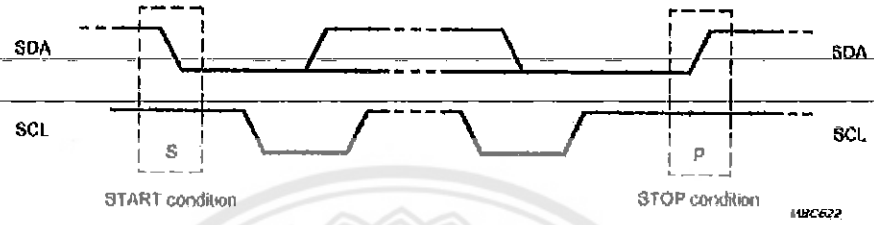
ทางบริษัทผู้พัฒนาระบบ I2C ได้ให้คำจำกัดความของการสื่อสารแบบ I2C ไว้เพื่อให้การศึกษาเกิดความเข้าใจเดียวกัน โดยมีรายละเอียดต่างๆดังนี้

2.3.12.1 Bit Transfer ข้อมูล 1 บิทจะถูกส่งออกไปด้วยช่วงเวลา 1 CLOCK โดยข้อมูลที่สาย SDA จะต้องคงที่ในขณะที่ CLOCK เป็นลอจิก 1



รูปที่ 2.25 แสดง Timing Diagram ของ Bit Transfer

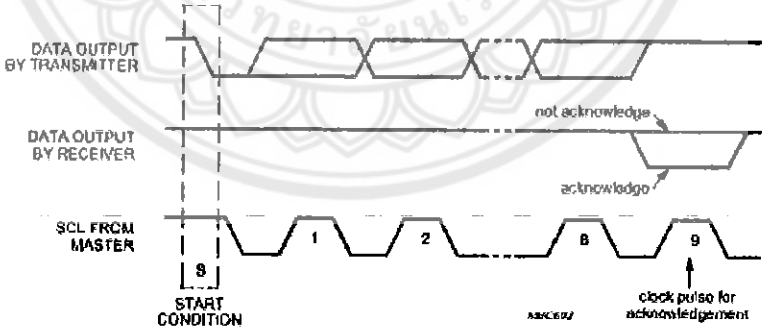
2.3.12.2 Start and Stop Conditions ทั้งสายสัญญาณ SDA และสายสัญญาณ SCL ถ้าอยู่ในสถานะไม่ BUSY จะเป็นลอจิก 1 การเปลี่ยนแปลงจากลอจิก 1 เป็นลอจิก 0 ของสายสัญญาณ SDA ขณะที่สายสัญญาณ SCL เป็น 1 เรียกว่าการกำหนดเงื่อนไข START แต่การเปลี่ยนแปลงจาก 0 เป็น 1 ของสายสัญญาณ SDA ในขณะที่สายสัญญาณ SCL เป็น 1 เรียกว่าการกำหนดเงื่อนไข STOP



รูปที่ 2.26 แสดง Timing Diagramของ Start and Stop Conditions

2.3.12.3 System configuration อุปกรณ์ที่ส่งข้อมูลเรียกว่า TRANSMITTER ส่วนอุปกรณ์ที่รับข้อมูลเรียกว่า RECEIVER และอุปกรณ์ที่ใช้ควบคุมทิศทางการสื่อสารข้อมูลเรียกว่า MASTER และอุปกรณ์ที่ถูกควบคุมจาก MASTERเรียกว่า SLAVE

2.3.12.4 Acknowledge จำนวนไบต์ของข้อมูลที่ถูกส่งระหว่างตัวรับและตัวส่งมีได้ไม่จำกัด ซึ่งเมื่อส่งข้อมูลครบ 1 ไบต์จะต้องส่งบิท ACK ตามออกไป 1 บิท



รูปที่ 2.27 แสดง Timing Diagram ของ Acknowledge

2.3.13 โซลินอยด์วาล์ว (Solenoid Valve)

โซลินอยด์วาล์ว (Solenoid Valve) คือวาล์วควบคุมชนิดกลไกนำและวาล์วจะปิดในเวลาทำงานปกติ มันจะใช้พลังงานของของเหลวในระบบเป็นผู้ช่วยในการเปิด/ปิด ภายใต้สถานะความดันที่ถูกต้องเมื่อจ่ายไฟฟ้าให้กับคอยล์จะทำให้เกิดสนามแม่เหล็กขึ้นบริเวณส่วนบนสุดของก้านวาล์ว พลังแม่เหล็กจะดึงก้านวาล์วและลิ้นวาล์วของวาล์วนำขึ้น ของเหลวที่อยู่ด้านบนไดอะแฟรม

จะถูกขับออกผ่านทางรูระบายเล็กๆ(orifice)ไปยังทางออกของท่อหลัก ในขณะเดียวกันความดันทางเข้าบริเวณส่วนล่างของไดอะแฟรมจะยกไดอะแฟรมขึ้น และเมื่อเปิดลิ้นวาล์วหลักแล้ว ของเหลวก็จะไหลเข้าตลอดแนวทางเดินของวาล์ว

เมื่อหยุดจ่ายไฟฟ้าให้กับคอยล์ สปริงภายในตัวโซลินอยด์ (ไม่ได้สัมผัสกับของเหลว) จะผลักให้ก้านวาล์วและลิ้นปิดช่องทางเดินของวาล์วน้ำ ซึ่งจะทำให้ความดันเริ่มสะสมบริเวณด้านบนของตัวไดอะแฟรมมมากขึ้นจนผลักดันไดอะแฟรมเลื่อนลงปิดบ่าวาล์วของวาล์วหลัก

ด้วยหลักการดังกล่าวโซลินอยด์วาล์ว จึงนับว่ามีความสำคัญกับระบบการรดน้ำอัตโนมัติเป็นอย่างมาก และถ้าเรามีจำนวนจุดใช้งานหลายจุด การที่เราจะเดินไปเปิด-ปิดวาล์วน้ำด้วยตัวเองทุกครั้งที่เราต้องการใช้งานสปริงเกอร์คงจะลำบากและเสียเวลา แต่ถ้าเราติดตั้งระบบจ่ายน้ำอัตโนมัติ โซลินอยด์วาล์วไฟฟ้าจะเข้ามามีบทบาทกับการใช้งานเป็นอย่างมากแต่ โซลินอยด์วาล์ว ก็มีหลายแบบทำจากวัสดุที่แตกต่างกัน ทั้งที่ทำจากทองเหลือง สแตนเลส พลาสติก ควบคุมการเปิด-ปิดวาล์วด้วยมือ ลม หรือไฟฟ้า ทั้งนี้การเลือกใช้งานก็ต้องเลือกตามความเหมาะสมด้วย แต่ที่นิยมใช้ในระบบรดน้ำคือ โซลินอยด์วาล์วแบบไฟฟ้าที่ทำจากพลาสติกหรือทองเหลือง

รูปที่ 2.28 แสดงโซลินอยด์วาล์ว

ขั้นตอนต่อจะเป็นวิธีการดำเนินงาน การทำลองและหลักการออกแบบในการทำงานซึ่งท่านผู้อ่านจะได้ทราบถึงรายละเอียดต่อไปในบทที่ 3

บทที่ 3
วิธีการดำเนินงาน

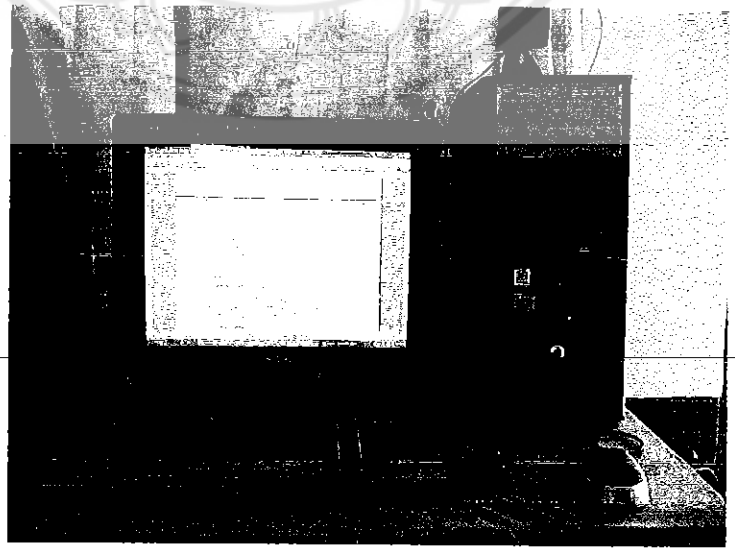
จากบทที่ 2 ทำให้เราทราบถึงหลักการ และทฤษฎีบทที่เกี่ยวข้องกับการทำโครงงาน ซึ่งในบทนี้จะพูดถึงขั้นตอนการดำเนินงานและการออกแบบ แบบจำลองที่ใช้งานจริง โดยเนื้อหาจะมีรายละเอียดดังนี้

- อุปกรณ์ที่ใช้ในการทำชิ้นงาน
- ทำความเข้าใจกับบอร์ด MCS-51 และการทำงานของ DS 1307
- ขั้นตอนการเขียนโปรแกรม การทำไฟล์ .HEX ของไมโครคอนโทรเลอร์
- ขั้นตอนการนำไฟล์ .HEX ลงบอร์ด MCS-51
- ออกแบบและติดตั้งอุปกรณ์เชื่อมต่อไมโครคอนโทรเลอร์เข้ากับอุปกรณ์อื่นๆที่เกี่ยวข้อง
- ขั้นตอนการทำงานของระบบที่ออกแบบ
- ทดสอบการทำงานของระบบ

3.1 เตรียมอุปกรณ์ที่ใช้ในการทำโครงงาน

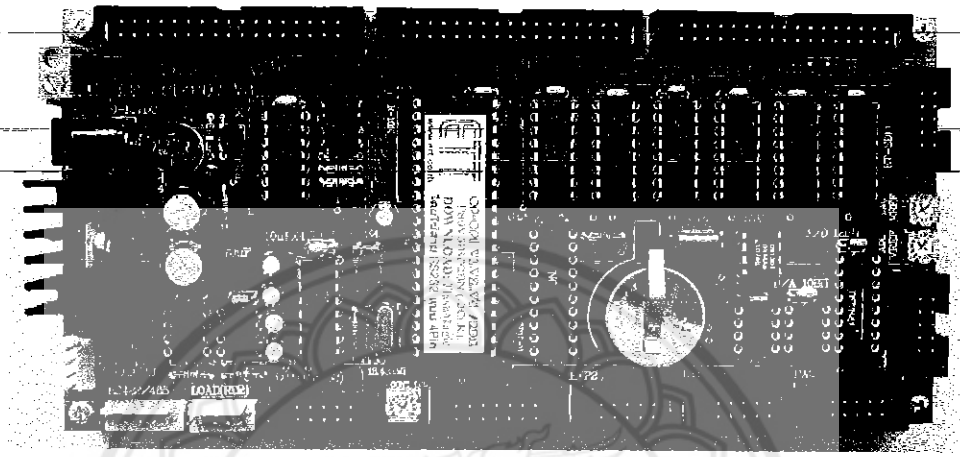
ขั้นตอนแรกคือการซื้ออุปกรณ์ที่เกี่ยวข้องกับการทดลองซึ่งอุปกรณ์ที่ใช้ทำโครงงานโดยรายละเอียดสามารถแสดงได้ดังรูปด้านล่างต่อไปนี้

3.1.1 คอมพิวเตอร์ สามารถใช้ได้ทั้งแบบตั้งโต๊ะและแบบพกพา



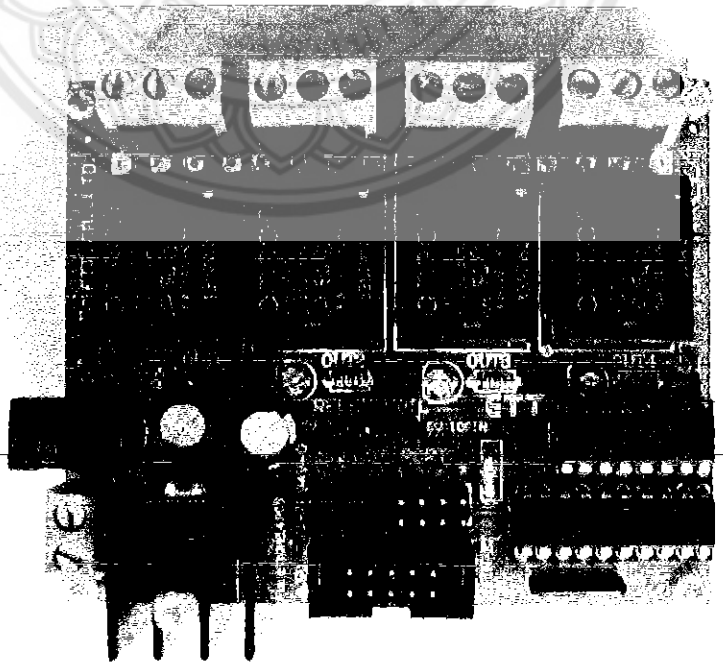
รูปที่ 3.1 แสดงคอมพิวเตอร์ PC

3.1.2 บอร์ดไมโครคอนโทรเลอร์ MCS-51 รุ่น CP-SPI/RD2 V3.0 ภายในบอร์ดจะมีส่วนประกอบที่สำคัญ คือ ตัวไมโครคอนโทรเลอร์ PV89V51RD2 ไอซีฐานเวลาจริง D31307 พร้อมถ่าน black up 3V โดยตัวบอร์ดจะใช้ไฟ 12 VDC



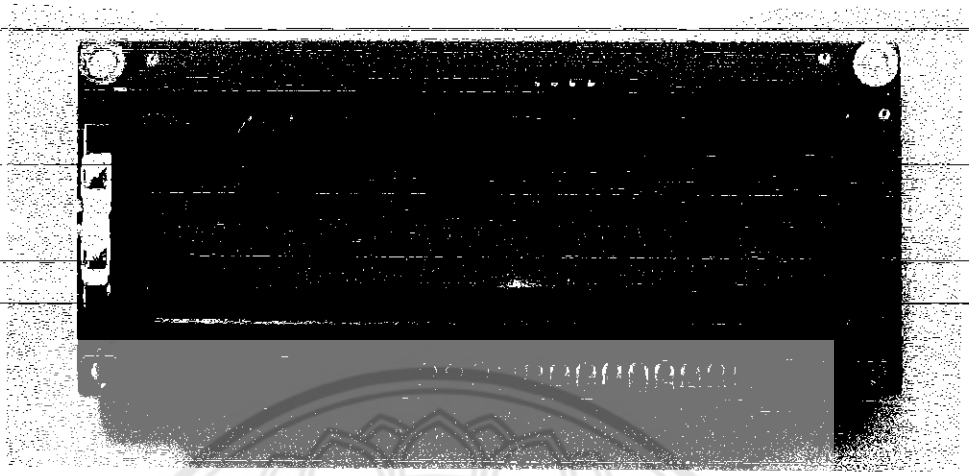
รูปที่ 3.2 แสดงบอร์ดไมโครคอนโทรลเลอร์ MCS-51 รุ่น P89V51RD2BN

3.1.3 บอร์ดควบคุมการทำงานของรีเลย์ ซึ่งจะเป็นส่วนควบคุมการเปิด-ปิดวงจรที่จ่ายให้กับโซลินอยด์วาล์วต่อไป



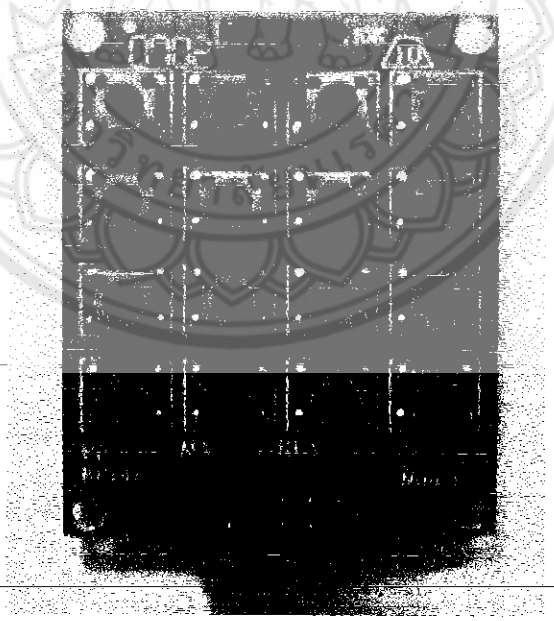
รูปที่ 3.3 แสดงบอร์ดควบคุมการทำงานของรีเลย์

3.1.4 จอแสดงผล LCD ใช้เป็นจอแสดงผลเวลาในการสั่งให้เครื่องควบคุมทำงาน โดยเป็นจอแสดงผลขนาด 16 ตัวอักษร 1 บรรทัด



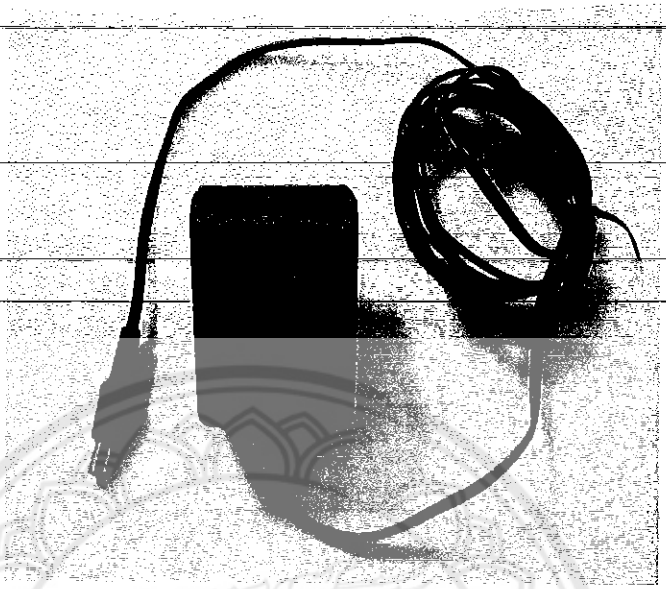
รูปที่ 3.4 แสดงจอแสดงผล LCD

3.1.5 ปุ่มคีย์บอร์ด Matrix ซึ่งในโครงงานการทดลองนี้ใช้ Matrix 4 x 4



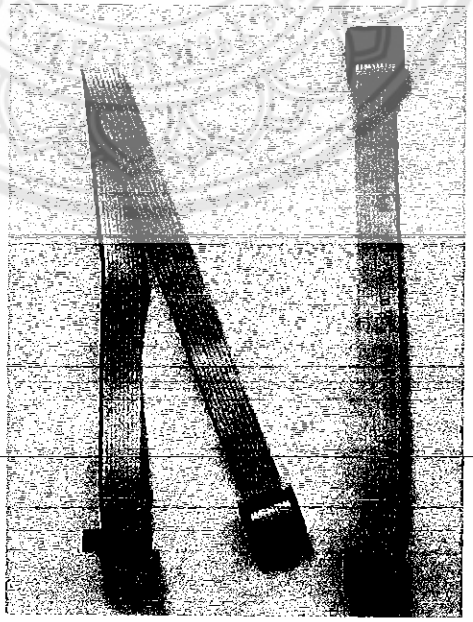
รูปที่ 3.5 แสดงปุ่มคีย์บอร์ด Matrix 4 x 4

3.1.6 Adaptor เป็นตัวแปลงไฟจาก 22VAC ไปเป็น 12 VAC เพื่อจ่ายไปให้กับบอร์ด MCS-51



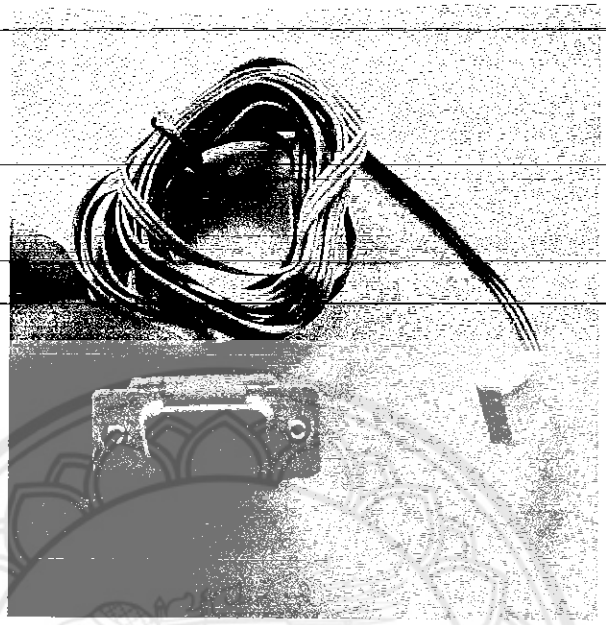
รูปที่ 3.6 แสดงตัว Adaptor 220 VAC แปลงเป็น 12 VAC

3.1.7 สายแพร10 Pin ทำหน้าที่เชื่อมต่อกับบอร์ดควบคุมการทำงานของรีเลย์เข้ากับบอร์ด MCS-51 และเชื่อมต่อกับปุ่มคีย์บอร์ด Matrix 4 x 4



รูปที่ 3.7 แสดงสายแพร10 Pin

3.1.8 สาย RS 232 ที่ใช้เชื่อมต่อสื่อสารระหว่างบอร์ดไมโครคอนโทรเลอร์ MCS-51 กับเครื่อง Computer



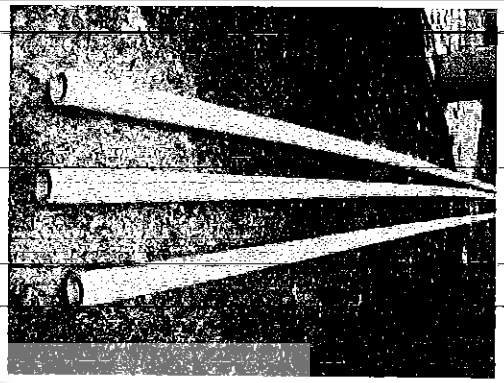
รูปที่ 3.8 แสดงสาย RS232 แบบ 9 ช่อง

3.1.9 โซลินอยด์วาล์ว เป็นวาล์วเปิด – ปิดน้ำด้วยการจ่ายไฟฟ้าให้กับอุปกรณ์ ในโครงงานนี้ใช้ 220 VAC

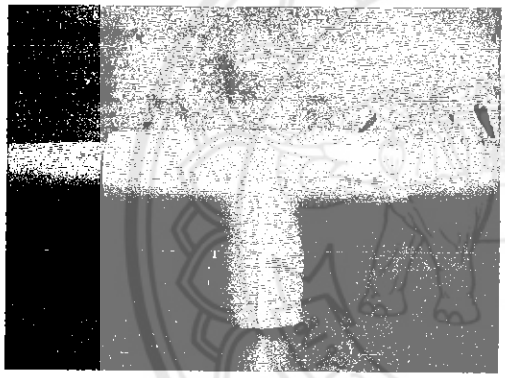
รูปที่ 3.9 แสดงโซลินอยด์วาล์ว 220 VAC

3.1.10 ท่อน้ำ ในโครงงานฉบับนี้เลือกใช้ขนาดท่อที่ต่อเข้ากับโซลินอยด์วาล์วมีขนาด 3/4 นิ้ว หรือ 6 หุนแล้วแปลงลงมาเป็น 1/2 นิ้ว เพื่อความสะดวก

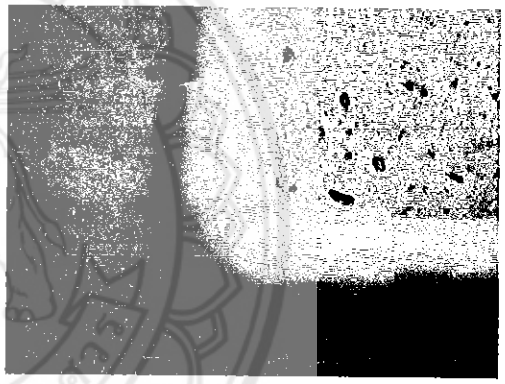
รูปที่ 3.10.1 ข้อต่อเกรียว



รูปที่ 3.10.2 ท่อน้ำยาว

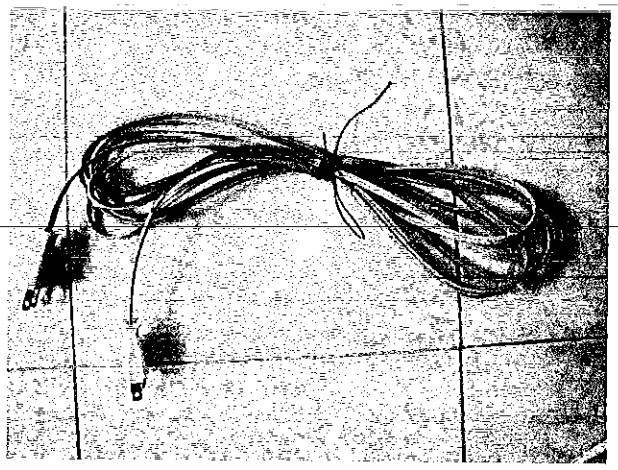


รูปที่ 3.10.3 ข้อต่อสามทาง



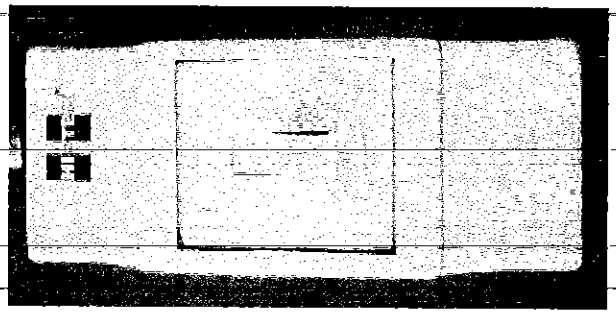
รูปที่ 3.10.4 ข้องอ

3.1.11 สายไฟ ใช้เชื่อมต่ออุปกรณ์เข้ากับไฟบ้าน และจากอุปกรณ์ไปอุปกรณ์เอง



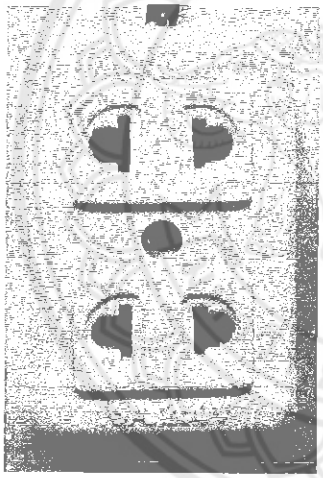
รูปที่ 3.11 แสดงสายไฟที่ถูกนำมาใช้งาน

3.1.12 เบรกเกอร์ ทำหน้าที่เปิด-ปิดไฟที่รับมาจากไฟบ้าน เพื่อเชื่อมต่อกับวงจรบนบอร์ดไมโครคอนโทรลเลอร์

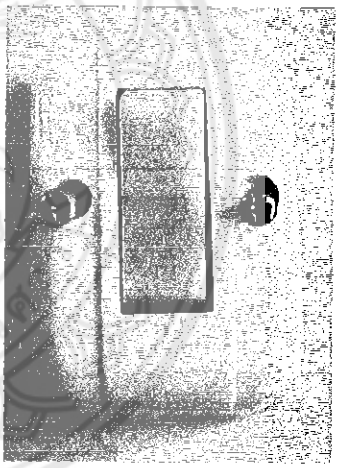


รูปที่ 3.12 แสดงเบรกเกอร์

3.1.13 สวิสต์ไฟและเต้าปลักไฟ ใช้เพื่อช่วยควบคุมการทำงานของวงจร การเปิด-ปิดการจ่ายไฟเพื่อสร้างความปลอดภัยให้กับระบบมากยิ่งขึ้น



รูปที่ 3.13.1 เต้าปลักไฟ



รูปที่ 3.13.2 สวิสต์ไฟ

3.2 ศึกษาทำความเข้าใจกับอุปกรณ์และการทำงานของอุปกรณ์

จากบทที่ผ่านมาได้อธิบายเกี่ยวกับทฤษฎีและหลักการทำงานของตัวบอร์ดไมโครคอนโทรเลอร์ ไอซี DS 1307 และโซลินอยด์วาล์ว ไว้ในเบื้องต้นทำให้ท่านผู้อ่านเกิดความเข้าใจในการทำงานมาพอสมควรและเป็นประโยชน์ในการทดลองต่อไป

โดยสิ่งที่เน้นเป็นพิเศษคือ

1. ตัวโปรแกรมที่ออกแบบโดยภาษาซีซึ่งจะได้กล่าวละเอียดในหัวข้อถัดไป
2. ขาของไอซี ไมโครคอนโทรเลอร์ PV89V51RD2 และขาของไอซีฐานเวลาจริง DS 1307

3.3 ขั้นตอนการออกแบบและการใช้งานโปรแกรม

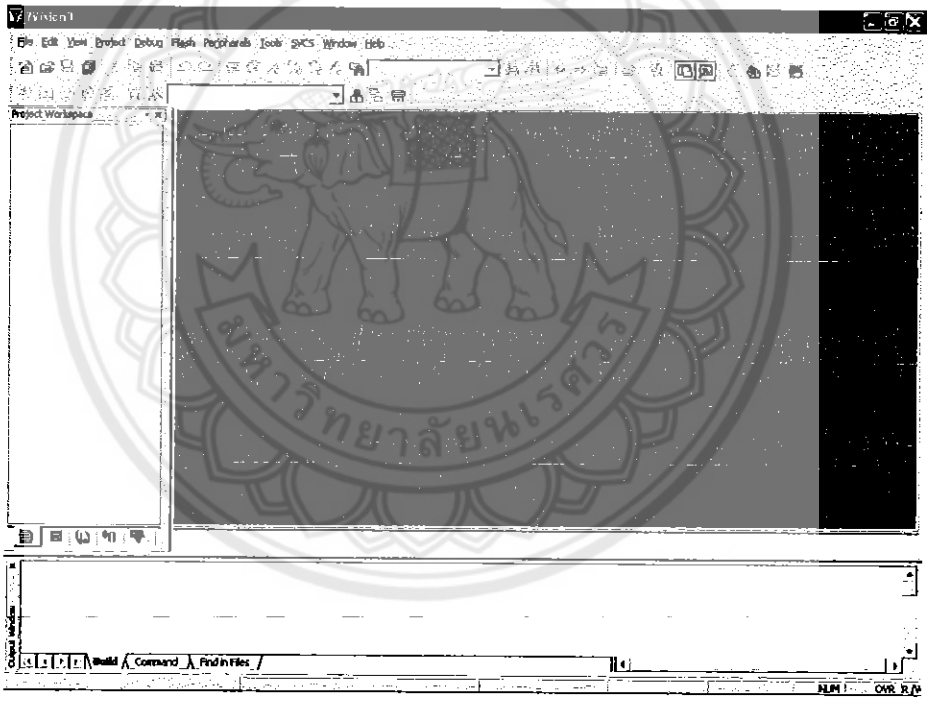
ในขั้นตอนนี้จัดได้ว่าเป็นส่วนประกอบที่กสำคัญที่สุดในการทำโครงงาน เพราะจะเป็นหัวใจในการสั่งการให้กับตัวไมโครคอนโทรเลอร์ ให้ทำงานตามที่เราต้องการหรือตามที่เรากำหนดได้ โดยการเขียนโปรแกรมสามารถแยกออกได้เป็น 2 ส่วน นั่นคือ

1. ตัวโปรแกรมที่ทำการแปลงมาเป็น .HEX
2. การ คอมพายร์(Compile) ลงบอร์ดไมโครคอนโทรเลอร์

ในขั้นตอนพื้นฐานเราต้องสามารถเขียนโปรแกรมสั่งงานลงบอร์ดให้ได้ ในที่นี้จะใช้โปรแกรม Keil uVision3 โดยภาษาที่ใช้เขียนจะใช้ภาษาซีทั้งหมด

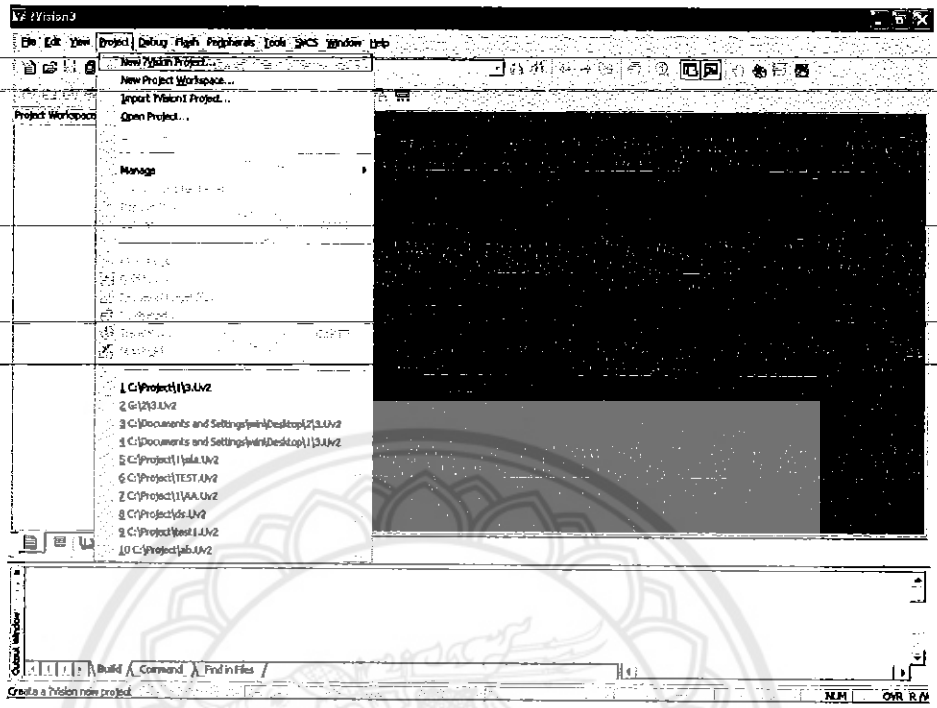
3.3.1 การใช้งานโปรแกรม Keil uVision3

1. รันโปรแกรม Keil uVision 3



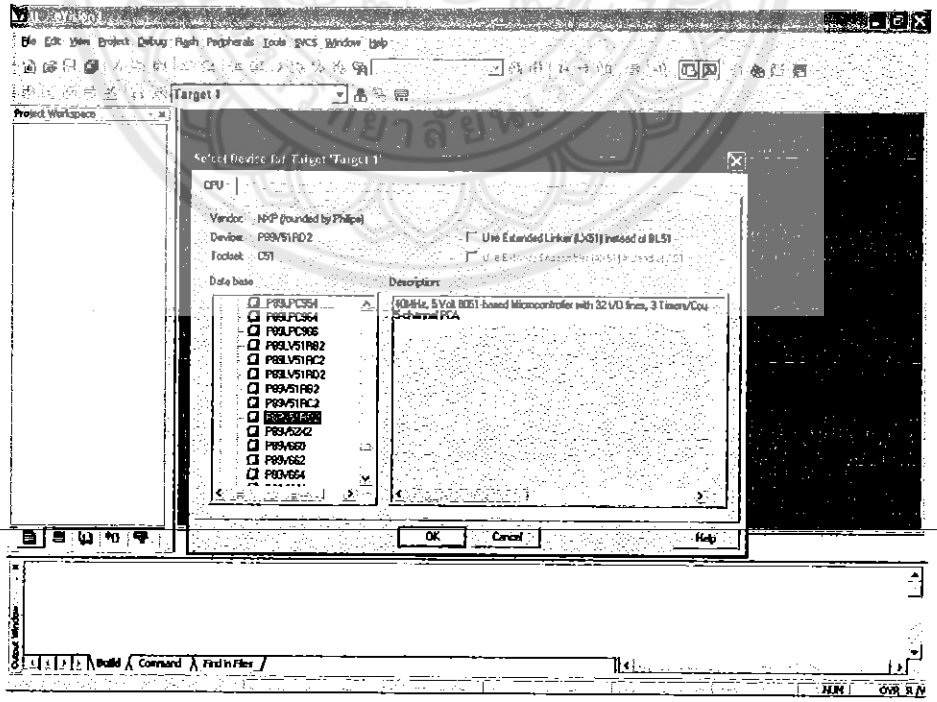
รูปที่ 3.13 แสดงหน้าต่างของโปรแกรม Keil uVision 3

2. สร้าง New Project



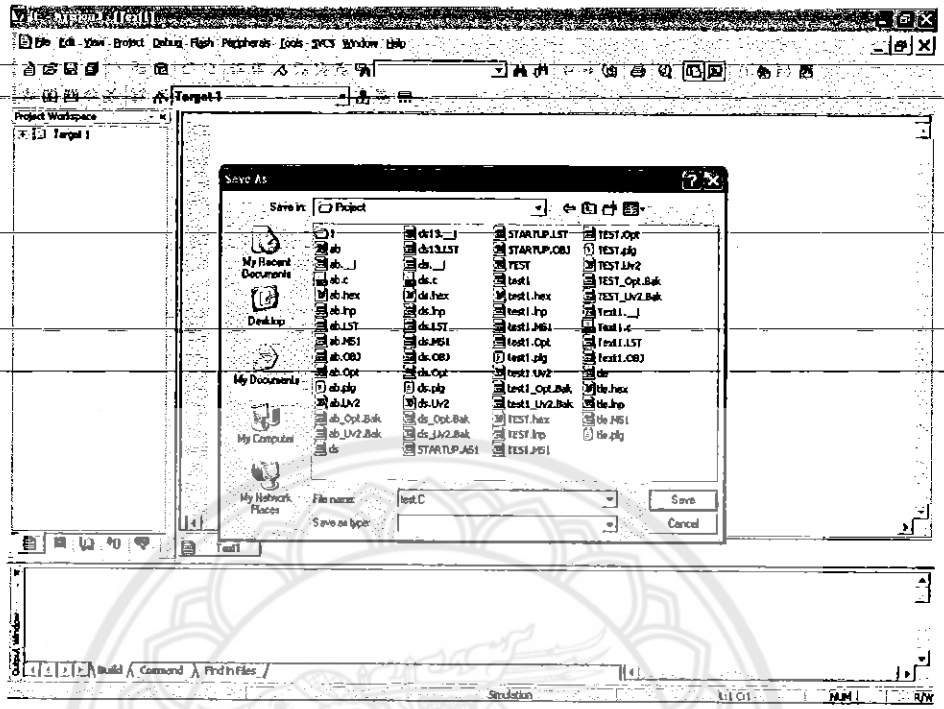
รูปที่ 3.14 การสร้าง New Project

3. เลือก PHILIPS และ 89V51RD2



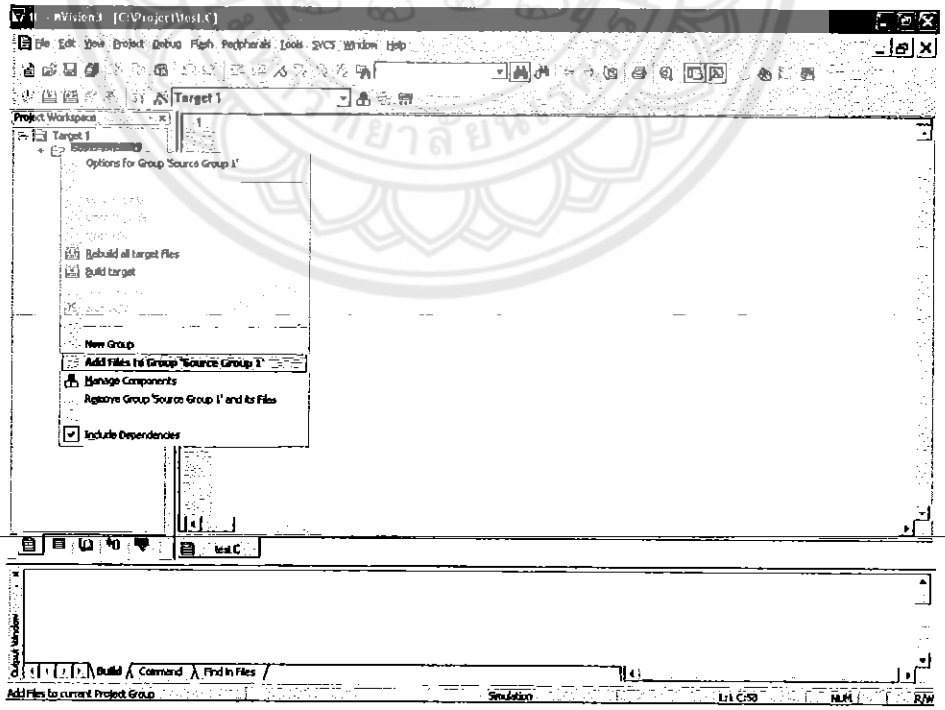
รูปที่ 3.15 การเลือก PHILIPS และ 89V51RD2

4. เลือก File / NEW จากนั้นเลือก File / Save แล้วตั้งชื่อ File เป็น .c



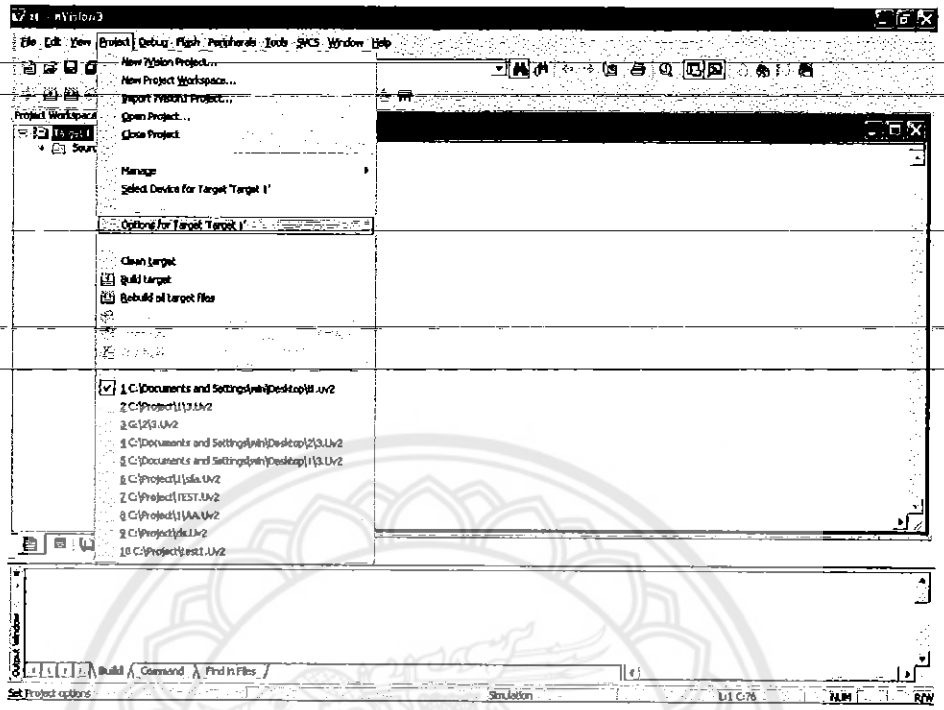
รูปที่ 3.16 การเลือก File / NEW จากนั้นเลือก File / Save

5. เพิ่ม File เข้าไปใน Group (Click ขวา ที่ Source Group1)



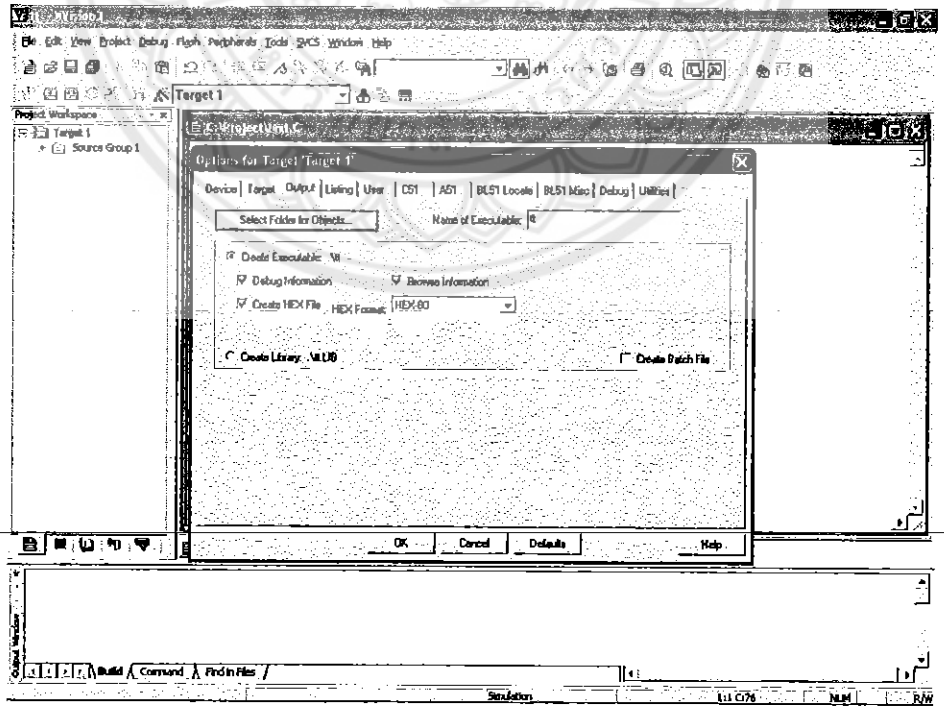
รูปที่ 3.17 การเพิ่ม File เข้าไปใน Group โดย Click ขวา ที่ Source Group1

6. ใช้ Mouse คลิกที่ Taget1 จากนั้น เลือก Options for Target “Target 1”



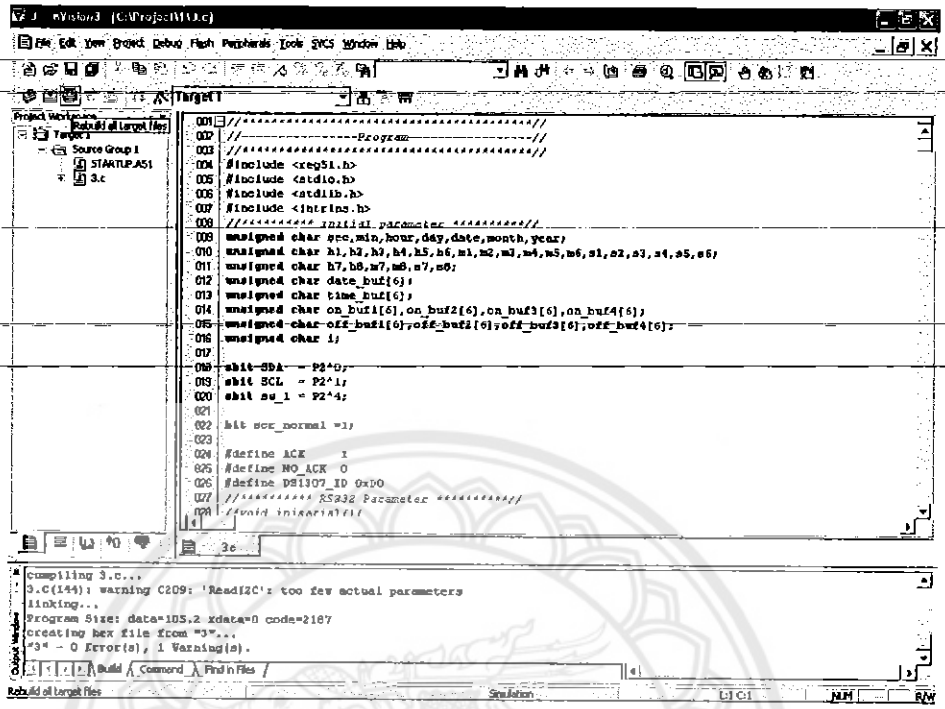
รูปที่ 3.18 เลือก Options for Target “Target 1”

7. เลือก Create HEX File



รูปที่ 3.19 เลือก Create HEX File

8. เขียนโปรแกรมจนสมบูรณ์ เมื่อต้องการแปลงให้เลือกที่ Rebuild all target files

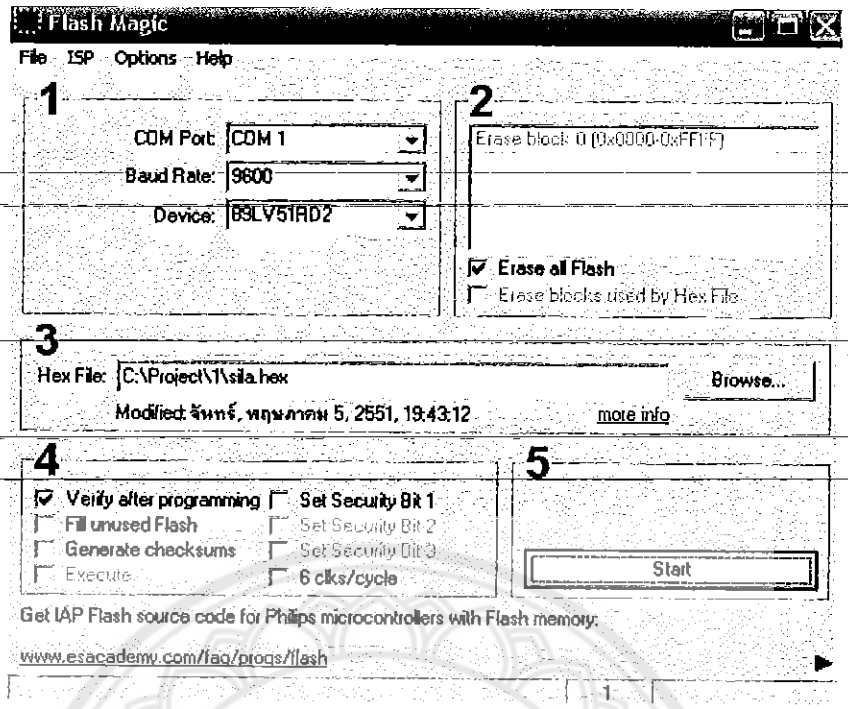


รูปที่ 3.20 เมื่อต้องการแปลงให้เลือกที่ Rebuild all target files

เมื่อเราได้โปรแกรมสำหรับเขียนโค้ตภาษาซีแล้ว เราจะได้ไฟล์ที่เป็น .HEX ออกมา เพื่อนำมา Compile หรือสั่งให้กับตัวไมโครคอนโทรเลอร์ที่อยู่บนบอร์ดต่อไป ดังนั้นจึงต้องมีโปรแกรมสำหรับ Compile ในโครงงานฉบับนี้จะใช้ของ Flash magic ซึ่งการนำไฟล์.HEX ลงโปรแกรม Flash magic มีขั้นตอนดังต่อไปนี้

3.3.2 การนำไฟล์ . HEX ลงบอร์ดไมโครคอนโทรเลอร์โดยใช้โปรแกรม **Flash magic**

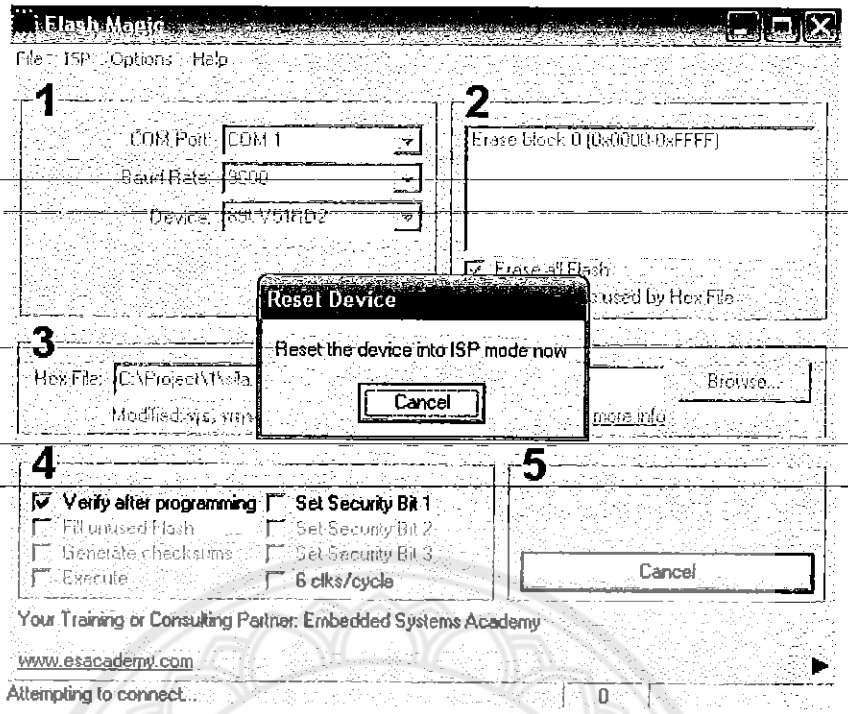
ในกรณีที่ใช้ MCU เบอร์ P89V51RD2 ของ Philips นั้น วิธีการ Download โปรแกรมให้กับบอร์ดจะสามารถทำได้โดยง่ายกว่า MCU เบอร์อื่นๆ เนื่องจากสามารถใช้วงจรสื่อสารอนุกรม RS232 ปรกติเหมือนกับการใช้งานทั่วๆไป ซึ่งในการ Download โปรแกรมสามารถทำได้ทันที โดยไม่ต้องจัดวงจรควบคุมอื่นๆให้กับไมโครคอนโทรเลอร์อีก สำหรับในกรณีที่ต้องการใช้งานกับบอร์ดไมโครคอนโทรลเลอร์ต่างๆของ ETTนั้นจะสามารถทำ การ Downloadโปรแกรมผ่านทางขั้วต่อ RS232 แบบ 4 Pin ได้ทันที โดยมีหน้าตาของโปรแกรมดังรูป



รูปที่ 3.21 แสดงหน้าต่างของโปรแกรม Flash Magic

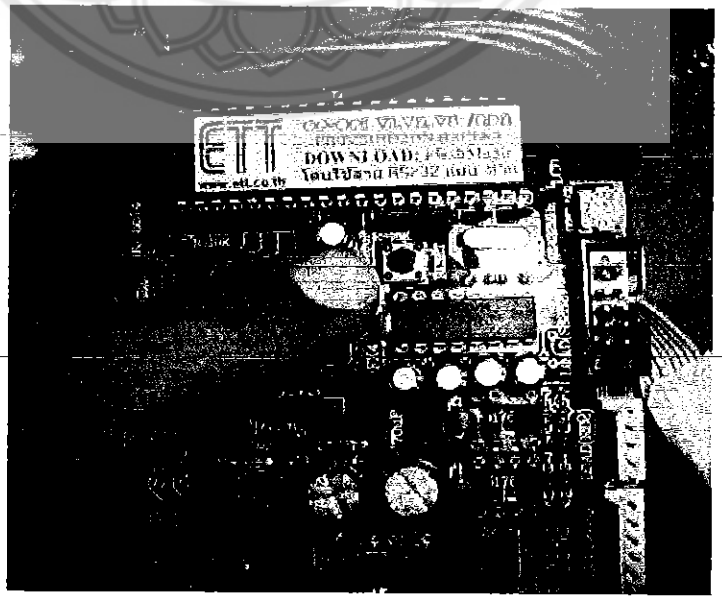
3.3.2.1 ขั้นตอนการใช้งาน อธิบายได้ดังนี้

1. ต่อสายสัญญาณ ET-RS232 ระหว่างคอมพิวเตอร์ PC กับ บอร์ดไมโครคอนโทรลเลอร์ที่ตำแหน่งของขั้วต่อสาย ET-RS232 (4 Pin) พร้อมกับจ่ายไฟให้กับบอร์ดเพื่อพร้อมรับคำสั่ง
2. สั่ง Run โปรแกรม Flash Magic
3. เลือกกำหนด Comport ตามที่ต่อสายไว้จริง และ เลือกกำหนด Baud rate เป็น 9600
4. เลือกกำหนด Device ตามที่ใช้จริง คือ P89V51RD2
5. เลือกกำ หนด รูปแบบการลบข้อมูล ซึ่งถ้าไม่แน่ใจว่า CPU ถูก Lock ไว้หรือไม่ ให้เลือก Erase All Flash
6. เลือกกำ หนด Hex File ที่ต้องการ Download ตามต้องการ
7. เลือก Verify After Programming
8. เลือก Start เพื่อสั่ง Download ข้อมูลให้กับ CPU ซึ่งจะปรากฏหน้าต่างบอกให้ RESET การทำงานของ MCU ให้เริ่มต้นทำงานใน ISP Mode ดังรูป



รูปที่ 3.22 Reset Device

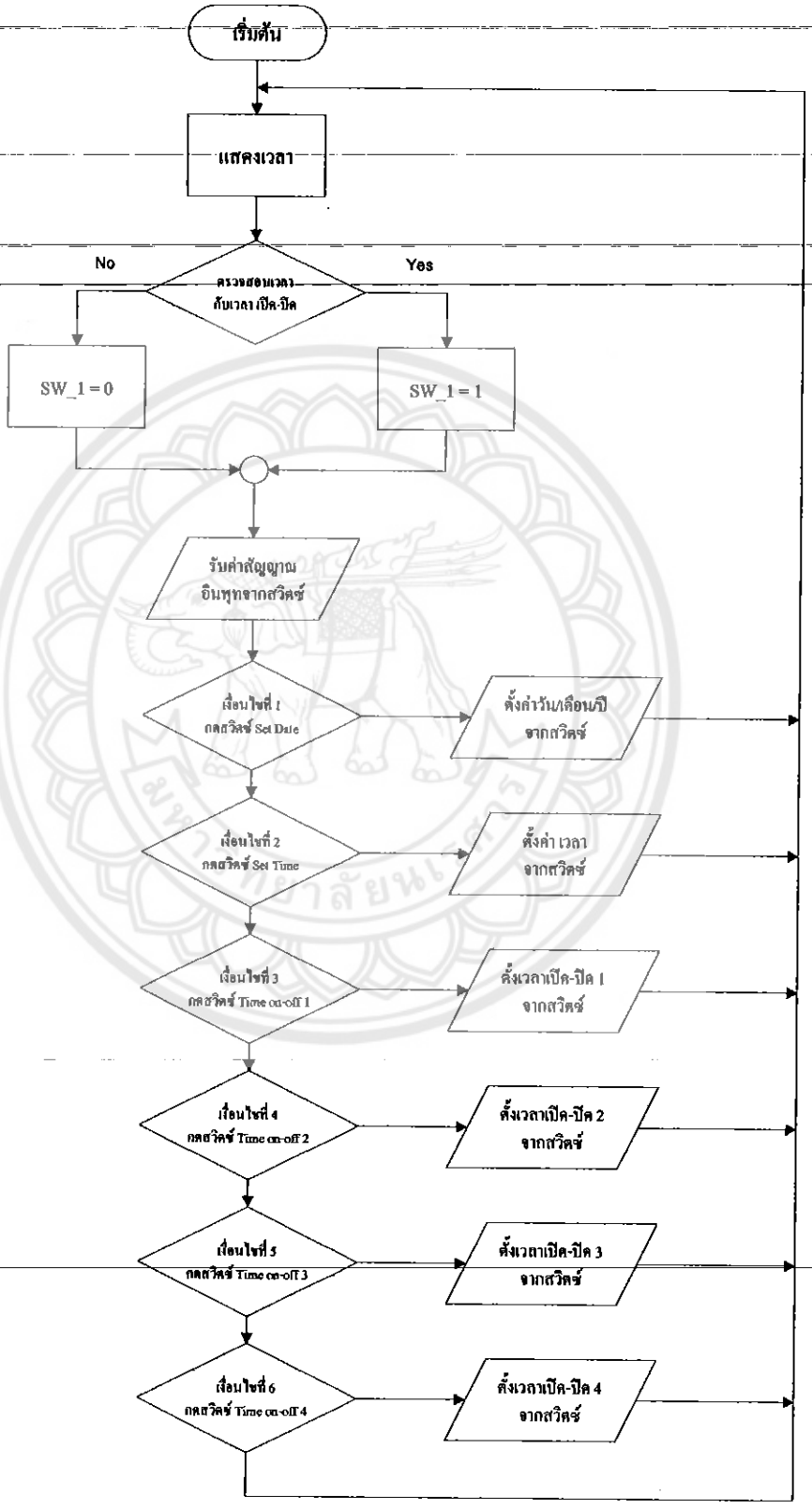
9. ให้ทำการกดสวิตช์ RESET ที่บอร์ดไมโครคอนโทรลเลอร์ ซึ่งหน้าต่างข้างต้นจะหายไป และสังเกตุเห็นโปรแกรมเริ่มต้นทำงาน ตามขั้นตอนต่างๆที่เลือกไว้ รอจนเสร็จขั้นตอน
10. ให้ทำการกดสวิตช์ RESET ที่บอร์ดไมโครคอนโทรลเลอร์อีกครั้งหนึ่งเพื่อให้ MCU เริ่มต้นทำงาน ซึ่งจะสังเกตุเห็น MCU เริ่มต้นทำงานตามโปรแกรมที่ได้ทำการ Download ไปแล้วทันที



รูปที่ 3.23 แสดงตำแหน่งปุ่มสวิตช์ Reset บนบอร์ด MCU

3.3.3 การออกแบบและเขียนโปรแกรม

- การเขียนโปรแกรมสามารถอธิบายในรูปแบบเงื่อนไขแสดงเป็น Flowchart ได้ดังนี้



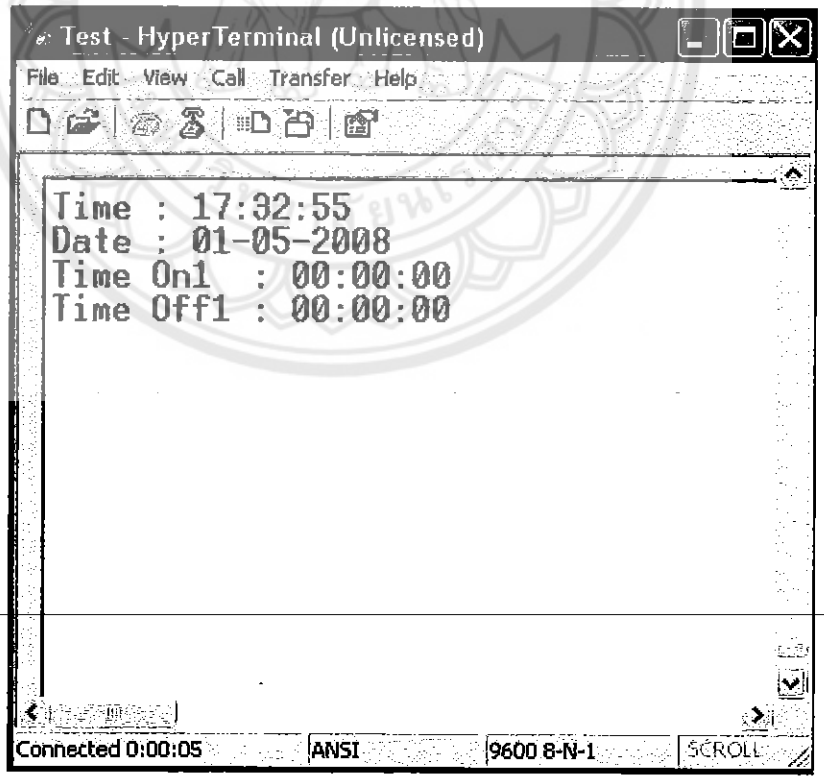
รูปที่ 3.24 แสดง Flowchart จำลองการทำงานของโปรแกรม

ในการออกแบบการเขียนโปรแกรมการควบคุมการทำงานของไมโครคอนโทรลเลอร์จะใช้โปรแกรม Keil uVision 3 โดยมีขั้นตอนในการเขียนโปรแกรม ดังแสดงไว้ในรูปที่ 3.25 ซึ่งสามารถอธิบายได้ดังนี้ คือ เริ่มต้นโปรแกรมด้วยการแสดงเวลาปัจจุบัน จากนั้นทำการเปรียบเทียบเงื่อนไขว่าเวลาปัจจุบันตรงกับเวลาเปิด-ปิด ที่ได้ตั้งไว้หรือไม่ ถ้าเงื่อนไขเป็นจริงไมโครคอนโทรลเลอร์จะสั่งให้ SW_1 = 1 รีเลย์ก็จะทำงาน แต่ถ้าเงื่อนไขเป็นเท็จไมโครคอนโทรลเลอร์จะสั่งให้ SW_1 = 0 รีเลย์ก็จะไม่ทำงาน เมื่อโปรแกรมทำคำสั่งที่อยู่ภายใต้เงื่อนไขเสร็จแล้ว โปรแกรมก็จะไปเริ่มต้นรอรับค่าอินพุตจากสวิตซ์เพื่อทำการเปรียบเทียบเงื่อนไขโดย

- เงื่อนไขที่ 1 เมื่อกดสวิตซ์ Set Date จะเป็นการตั้งค่า วัน เดือนและปี
- เงื่อนไขที่ 2 เมื่อกดสวิตซ์ Set Time จะเป็นการตั้งค่าเวลา ชั่วโมง นาทีและวินาที
- เงื่อนไขที่ 3-6 เมื่อกดสวิตซ์ Time On-Off จะเป็นการตั้งเวลาเปิด-ปิด ชั่วโมงและนาที

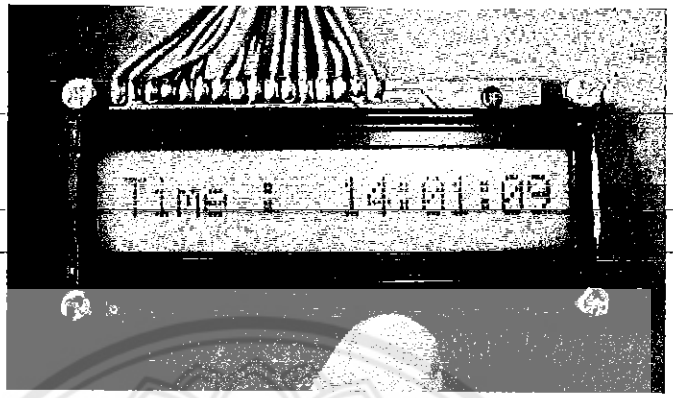
จาก Flowchart จะเห็นได้ว่าสามารถตั้งเวลาได้ถึง 4 ครั้ง และหลังจากที่ทำคำสั่งตามเงื่อนไขเสร็จแล้วโปรแกรมก็จะกลับไปแสดงเวลาปัจจุบันอีกครั้ง

เมื่อออกแบบ Flowchart จำลองการทำงานของโปรแกรมแล้ว จึงทำการเขียนโปรแกรมติดต่อกับ DS 1307 เพื่อให้ได้ฐานเวลาจริงออกมาแสดงในโปรแกรม hyper terminal ดังรูป ซึ่งตั้งเวลาและวันที่ตามจริงแล้ว



รูปที่ 3.25 แสดงผล DS 1307 ที่ตั้งเวลาแล้วใน hyper terminal

เมื่อทำการเขียนโปรแกรมติดต่อกับ DS 1307 ได้แล้ว และตั้งเวลากับวันที่เรียบร้อยแล้ว จึงทำการเขียนโปรแกรมสั่งให้ข้อมูลฐานเวลาจริงที่ได้แสดงผลในหน้าจอ LCD ซึ่งในโครงงานฉบับนี้จะใช้ LCD 1 บรรทัดในการแสดงผล



รูปที่ 3.26 แสดงข้อมูลฐานเวลาจริงที่ได้แสดงผลในหน้าจอ LCD

จากนั้นจึงทำการเขียนโปรแกรมติดต่อกับคีย์บอร์ด Matrix 4 x 4 เพื่อกำหนดค่าให้กับปุ่มทั้งหมด 16 ปุ่ม ซึ่งกำหนดพารามิเตอร์ไว้ดังตาราง

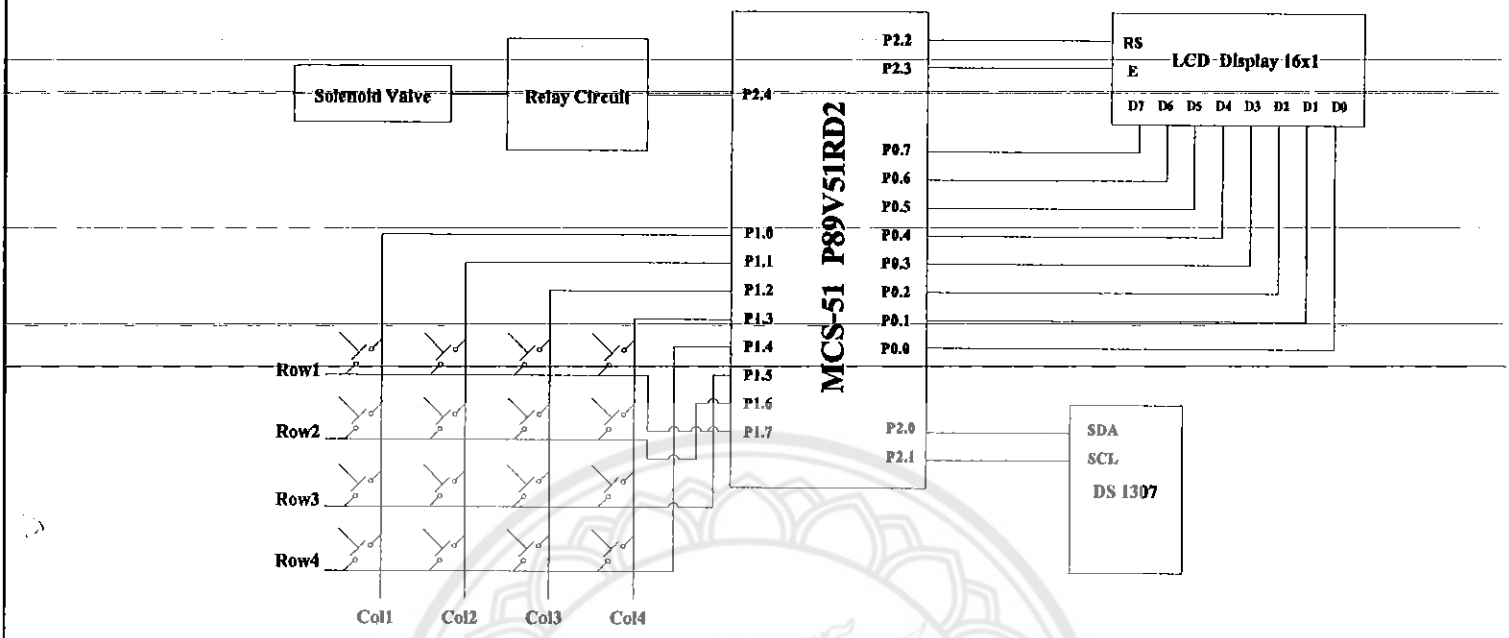
ตารางที่ 3.1 แสดงตำแหน่งปุ่มกดของสวิตซ์

แสดงตำแหน่งปุ่มกดของสวิตซ์ 4X4			
1	2	3	4
5	6	7	8
9	**0**	**Set Date**	**Set Time**
Set Time on-off 1	**Set Time on-off 2**	**Set Time on-off 3**	**Set Time on-off 4**

จากตารางจะแสดงให้เห็นว่าตำแหน่งปุ่มบนคีย์บอร์ดแต่ละปุ่มเป็นปุ่มอะไรบ้าง

- แถวแรกหรือแถวบนสุด เป็นปุ่ม 1 , 2 , 3 และ 4
- แถวที่ 2 เป็นปุ่ม 5 , 6 , 7 และ 8
- แถวที่ 3 เป็นปุ่ม 9 , 0 , Set Time และ Set Date
- แถวที่ 4 เป็นปุ่ม Set Time On-Off 1 ถึง 4

3.3.4 การออกแบบและต่อวงจรเข้ากับบอร์ดไมโครคอนโทรลเลอร์



รูปที่ 3.27 แสดงการต่อวงจรเข้ากับพอร์ตบนบอร์ดไมโครคอนโทรลเลอร์

อุปกรณ์หรือวงจรที่ต่อเข้ากับพอร์ตต่างๆของไมโครคอนโทรลเลอร์สามารถอธิบายได้ดังนี้

LCD Display

RS (ขา 4) ซึ่งเป็นขาอินพุตใช้แยกชนิดของข้อมูล ต่อเข้ากับพอร์ต P2.2

E (ขา 6) ซึ่งเป็นขาสำหรับรับสัญญาณพัลส์ ต่อเข้ากับพอร์ต P2.3

D0-D7 (ขา 7-14) ซึ่งเป็นขาข้อมูลขนาด 8 บิต ต่อเข้ากับพอร์ต P0.0-P0.7

IC DS1307

SDA (ขา 5) ซึ่งเป็นสัญญาณ Data แบบอนุกรม ต่อเข้ากับพอร์ต P2.0

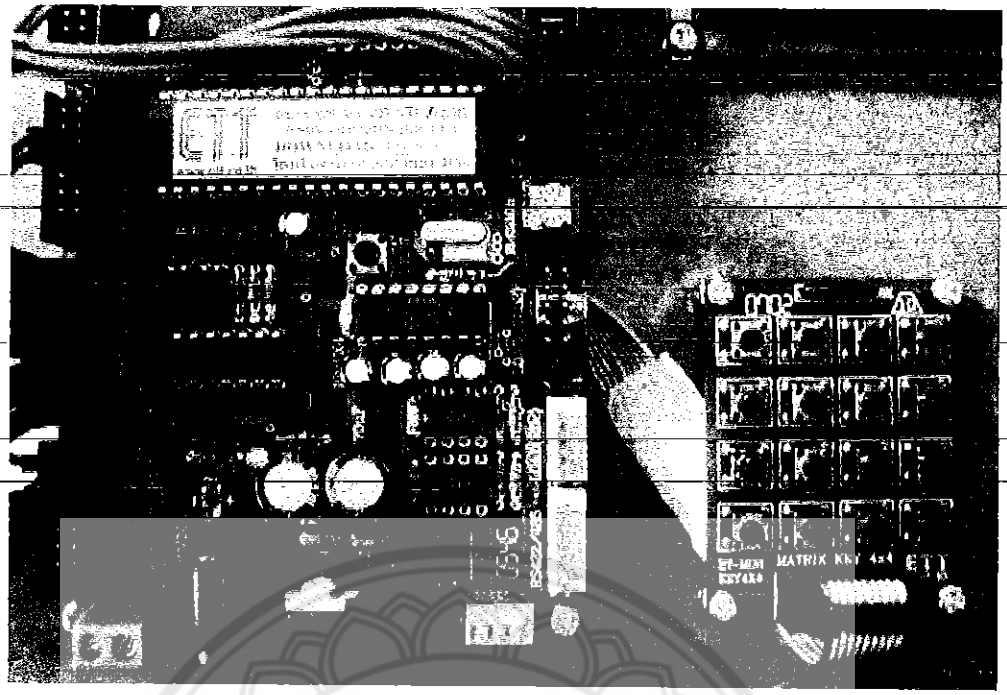
SCL (ขา 6) ซึ่งเป็นสัญลักษณ์ Clock แบบอนุกรม ต่อเข้ากับพอร์ต P2.1

Matrix Keyboard 4x4

Col1 ต่อเข้ากับพอร์ต P1.0	Row1 ต่อเข้ากับพอร์ต P1.7
Col2 ต่อเข้ากับพอร์ต P1.1	Row2 ต่อเข้ากับพอร์ต P1.6
Col3 ต่อเข้ากับพอร์ต P1.2	Row3 ต่อเข้ากับพอร์ต P1.5
Col4 ต่อเข้ากับพอร์ต P1.3	Row4 ต่อเข้ากับพอร์ต P1.4

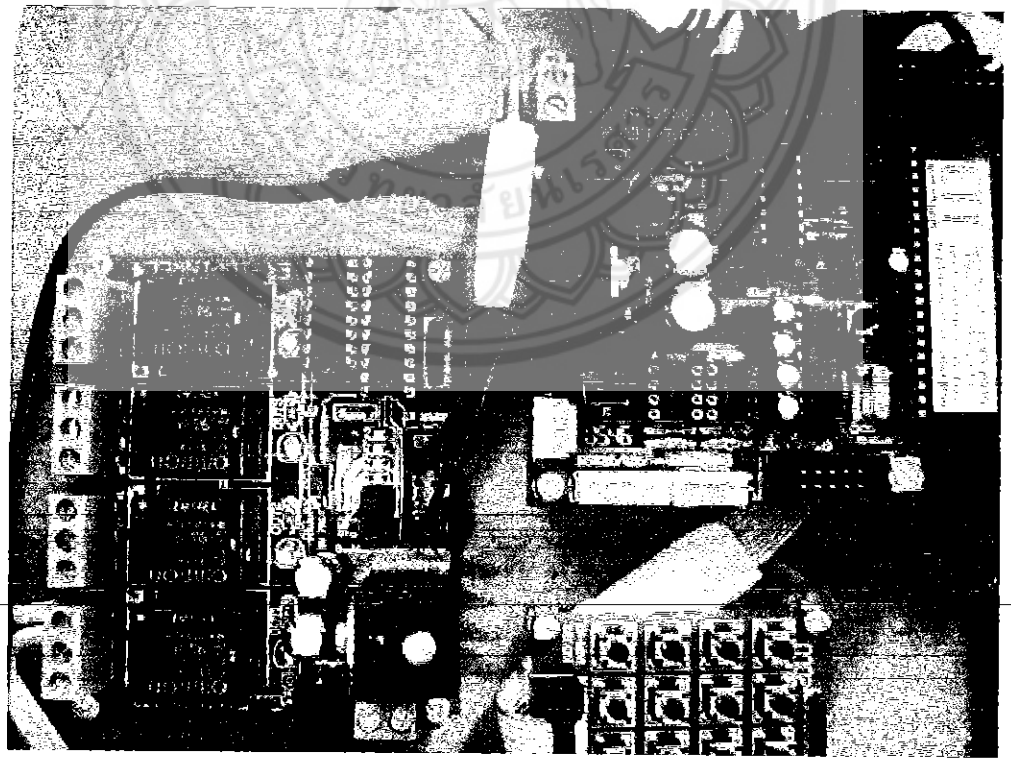
Relay Circuit

Relay ตัวที่ 1ซึ่งใช้เป็นสวิตซ์ ต่อเข้ากับพอร์ต P2.4



รูปที่ 3.28 แสดงการติดต่อไมโครคอนโทรลเลอร์กับคีย์บอร์ด Matrix 4 x 4

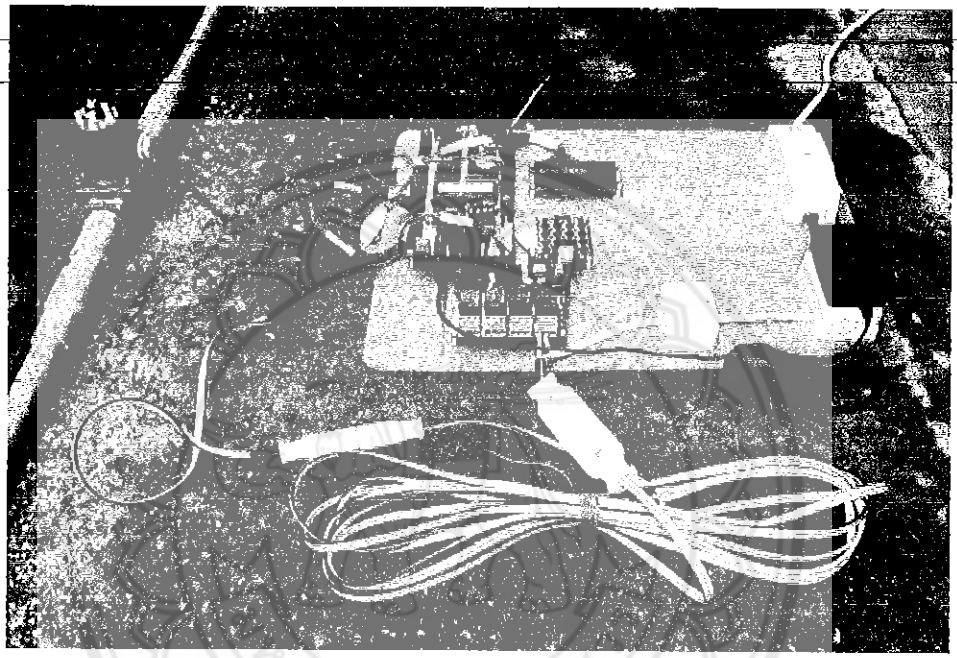
- เขียนโปรแกรมติดต่อกับรีเลย์ ให้รีเลย์ทำงานตามคำสั่งที่เราป้อนเข้าไป



รูปที่ 3.29 แสดงภาพรีเลย์เมื่อถูกสั่งให้ทำงาน

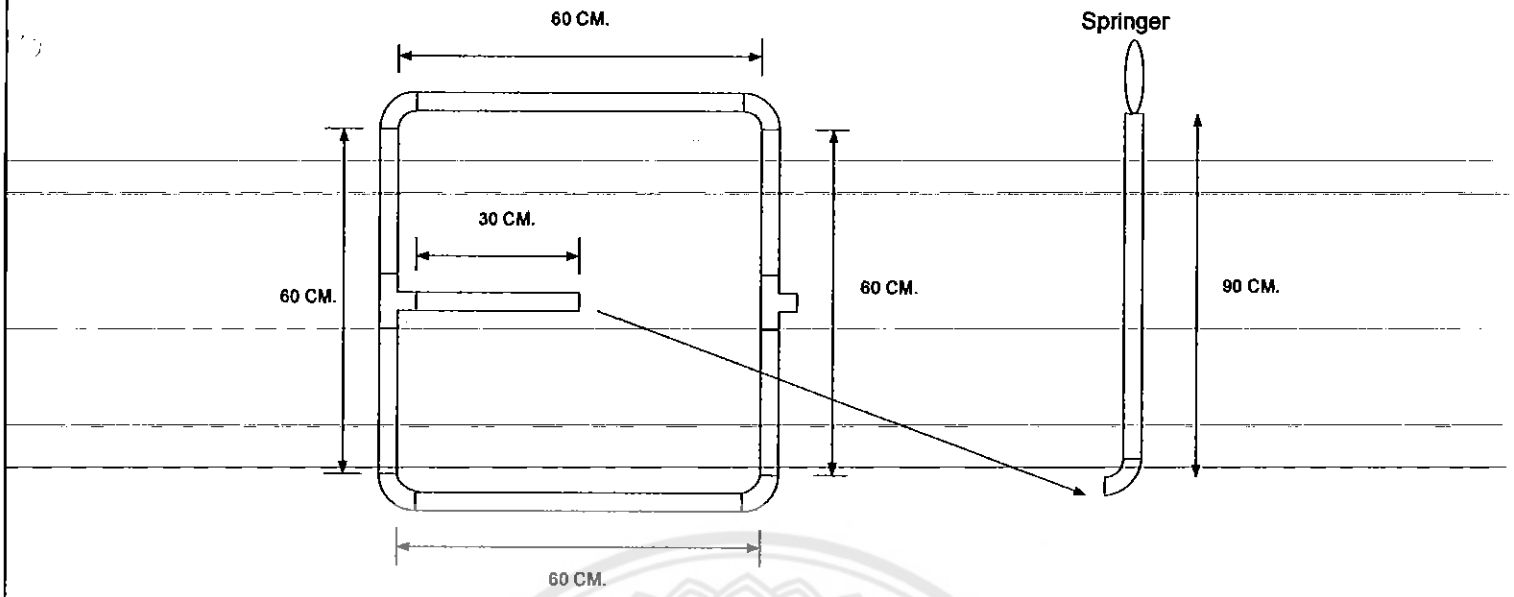
3.3.5 การออกแบบและการติดตั้งบอร์ดไมโครคอนโทรลเลอร์เข้ากับโซลินอยด์วาล์ว

เมื่อเราสามารถเขียนโปรแกรมควบคุมการทำงานของรีเลย์ ซึ่งทำหน้าที่คล้ายสวิตช์ ปิด-เปิดไฟแล้ว เราจึงนำมาเชื่อมต่อกับโซลินอยด์วาล์ว ซึ่งมีคุณสมบัติปิด-เปิดน้ำด้วยกระแสไฟฟ้า โดยหลักการคือถ้าโซลินอยด์วาล์วอยู่ในสภาวะปกติหรือไม่มีกระแสไฟ ตัวโซลินอยด์วาล์วจะอยู่ในสภาวะปิด แต่ถ้าทำการป้อนกระแสไฟเข้าไปแล้ว ตัวโซลินอยด์วาล์วจะอยู่ในสภาวะเปิดซึ่งทำให้น้ำสามารถไหลผ่านไปได้ โดยการออกแบบและติดตั้งแสดงดังรูป



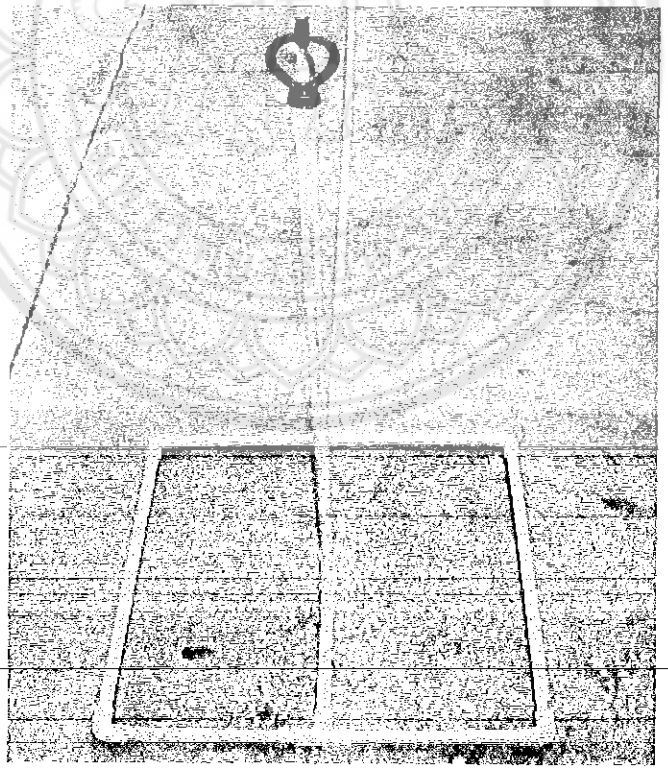
รูปที่ 3.30 แสดงการติดตั้งบอร์ดไมโครคอนโทรลเลอร์เข้ากับโซลินอยด์วาล์ว

การออกแบบเครื่องรดน้ำต้นไม้ในส่วนของท่อน้ำที่ใช้สามารถนำมาเป็นฐานให้กับตัวสปริงเกอร์ ซึ่งการออกแบบสามารถอธิบายได้ดังรูป อย่างไรก็ตามการออกแบบมีความยืดหยุ่นสามารถดัดแปลงให้เหมาะสมกับการใช้งานจริงของผู้ประกอบการได้

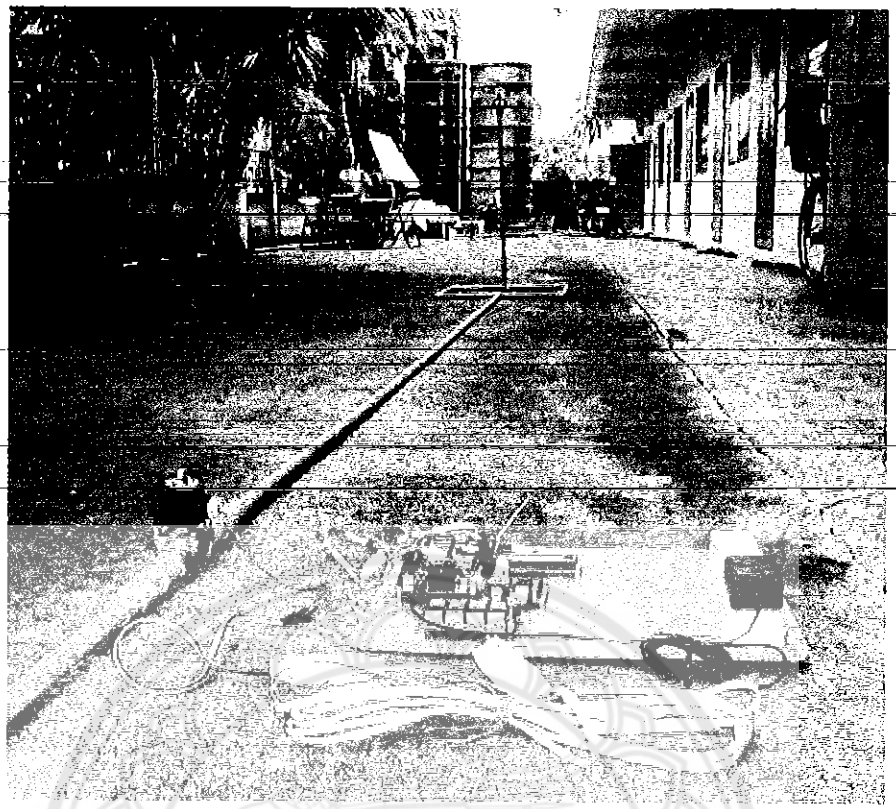


ส่วนฐานของสปริงเกอร์ ส่วนของสปริงเกอร์ที่ใช้พ่นน้ำ

รูปที่ 3.31 แสดงโครงสร้างการออกแบบท่อน้ำ



รูปที่ 3.32 แสดงการติดตั้งท่อน้ำและสปริงเกอร์



รูปที่ 3.33 แสดงการติดตั้งเครื่องรดน้ำอัตโนมัติกับท่อน้ำ

3.3.6 ขั้นตอนการทำงานของระบบ

เมื่อเราทำการติดตั้งอุปกรณ์เสร็จเรียบร้อยแล้ว การทำงานของระบบที่เราออกแบบ คือ จะทำการสั่งงานผ่านตัวคีย์บอร์ด Matrix 4 x 4 โดยมีข้อมูล 2 ประการที่จะสั่งงานคือ

1. เวลาที่ต้องการให้เปิดน้ำ
2. จำนวนเวลาที่ต้องการให้ไหลนานเท่าใด

แล้วส่งข้อมูลผ่านไปยังบอร์ดไมโครคอนโทรลเลอร์ แล้วรอดูผลการทำงานของระบบเมื่อถึงเวลาที่ตั้งไว้ ซึ่งการตั้งเวลาให้กับเครื่องรดน้ำต้นไม้อัตโนมัตินั้น จะขึ้นอยู่กับความประสงค์ของผู้ใช้งานหรือผู้ประกอบการว่าจะสั่งให้เปิด-ปิดน้ำเวลาใด โดยสามารถตั้งเวลาได้มากสุด 4 ครั้งต่อ 1 การสั่งงาน

<u>กรณีที่</u> 1 เราต้องการสั่งให้รดน้ำ 1 ครั้งเมื่อถึงเวลา 07.00 น. เป็นเวลา30 นาที

<u>วิธีการทดลอง</u>

1. ทำการตั้งเวลาและวันที่ (ตั้งครั้งแรกที่ใช้งานซึ่งระบบสามารถจำและเก็บข้อมูลไว้ได้) ซึ่งสมมติว่าขณะนี้เวลา 06.00 น.
2. ตั้งเวลาให้ระบบเริ่มทำงาน Time On : 07.00
3. กำหนดเวลาที่ระบบหยุดทำงาน Time Off : 07.30
4. รอดูผลการทำงาน

กรณีที่ 2 เราต้องการสั่งให้รดน้ำ 3 ครั้ง โดยมีความต้องการดังต่อไปนี้

- เมื่อถึงเวลา 07.00 น. เป็นเวลา 15 นาที
- เมื่อถึงเวลา 12.00 น. เป็นเวลา 30 นาที
- เมื่อถึงเวลา 17.00 น. เป็นเวลา 20 นาที

วิธีการทดลอง

1. ตั้งเวลาให้ระบบเริ่มทำงาน Time On : 07.00
2. กำหนดเวลาที่ระบบหยุดทำงาน Time Off : 07.15
3. ตั้งเวลาให้ระบบเริ่มทำงาน Time On : 12.00
4. กำหนดเวลาที่ระบบหยุดทำงาน Time Off : 12.30
5. ตั้งเวลาให้ระบบเริ่มทำงาน Time On : 17.00
6. กำหนดเวลาที่ระบบหยุดทำงาน Time Off : 07.20
7. รอดูผลการทำงาน

หมายเหตุ ถ้ามีการพิมพ์ข้อมูลผิดพลาดสามารถแก้ไขโดยการกดปุ่ม Reset ที่บอร์ดไมโครคอนโทรลเลอร์ และเริ่มทำการกำหนดเวลาที่ต้องการใหม่

การยกตัวอย่างทั้ง 2 กรณีนี้เป็นเพียงการสมมติขึ้นเพื่อให้เห็นภาพ ซึ่งการตั้งเวลาการทำงานจริงนั้น ผู้ที่ใช้งานสามารถกำหนดได้ตามชนิดของพืชที่ใช้ รวมไปถึงระยะเวลาที่เหมาะสม ซึ่งหลักการและขั้นตอน ในการตั้งเวลาให้กับเครื่องรดน้ำต้นไม้อัตโนมัตินั้น มีขั้นตอนที่เหมือนกัน

บทที่ 4

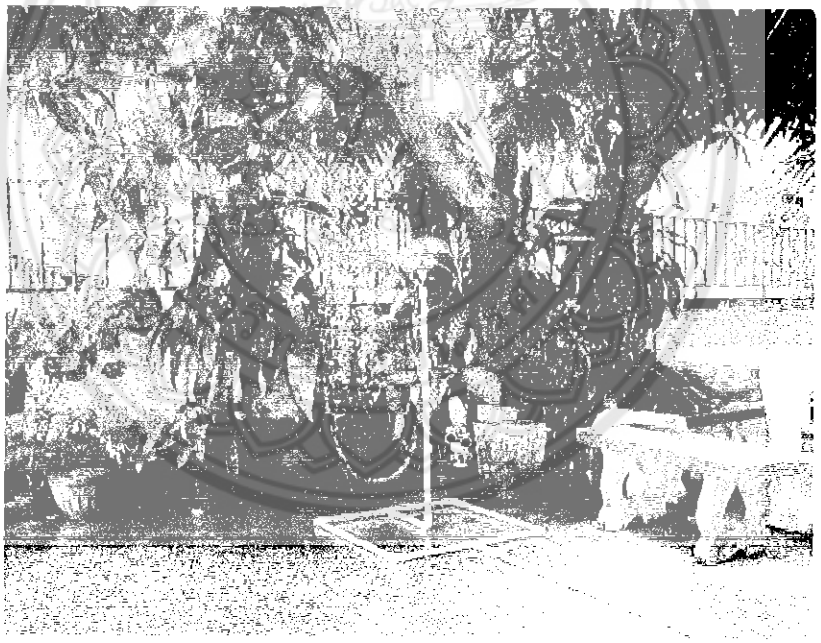
ผลการทดลองและการวิเคราะห์

ในบทที่ผ่านมาเราได้ทำการออกแบบระบบ เขียนโปรแกรม และติดตั้งอุปกรณ์เข้ากับชิ้นส่วนที่เกี่ยวข้องต่างๆแล้ว ทำการจำลองการทำงานของระบบให้ทำงานในเวลาที่เรากำหนดไว้ ซึ่งผลการทดลองแยกเป็นกรณีได้สามารถอธิบายได้ในบทนี้

4.1 ผลการทดลอง

<u>กรณีที่</u> 1 เราต้องการสั่งให้รดน้ำ 1 ครั้งเมื่อถึงเวลา 07.00 น. เป็นเวลา 30 นาที

เมื่อถึงเวลาที่กำหนด ระบบถูกทำงานอย่างตรงเวลา กล่าวคือ เมื่อถึงเวลา 07.00 ระบบสั่งการออกมาให้น้ำไหลผ่านโซลินอยด์วาล์วออกทางสปริงเกอร์ และเมื่อถึงเวลา 07.30 ระบบก็สั่งการให้น้ำหยุดไหล



รูปที่ 4.1 แสดงการทำงานของสปริงเกอร์รดน้ำกรณีที่ 1 เมื่อเวลา 07.00 น.

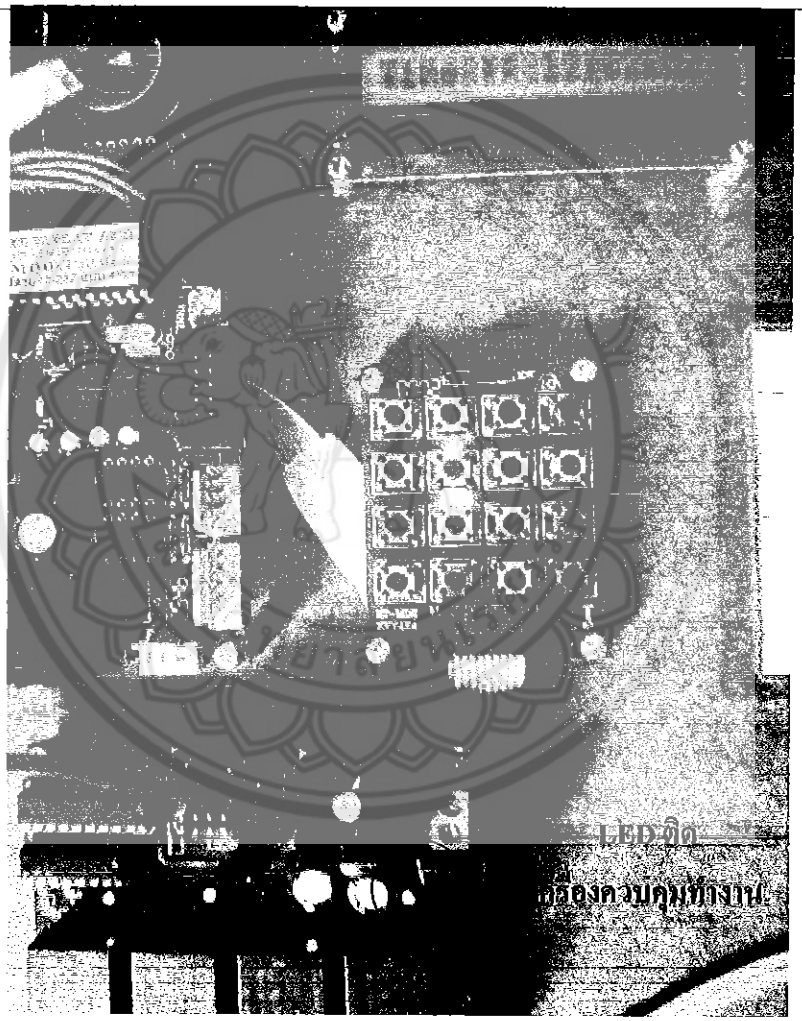
<u>กรณีที่</u> 2 เราต้องการสั่งให้รดน้ำ 3 ครั้ง โดยมีความต้องการดังต่อไปนี้

- เมื่อถึงเวลา 07.00 น. เป็นเวลา 15 นาที
- เมื่อถึงเวลา 12.00 น. เป็นเวลา 30 นาที
- เมื่อถึงเวลา 17.00 น. เป็นเวลา 20 นาที

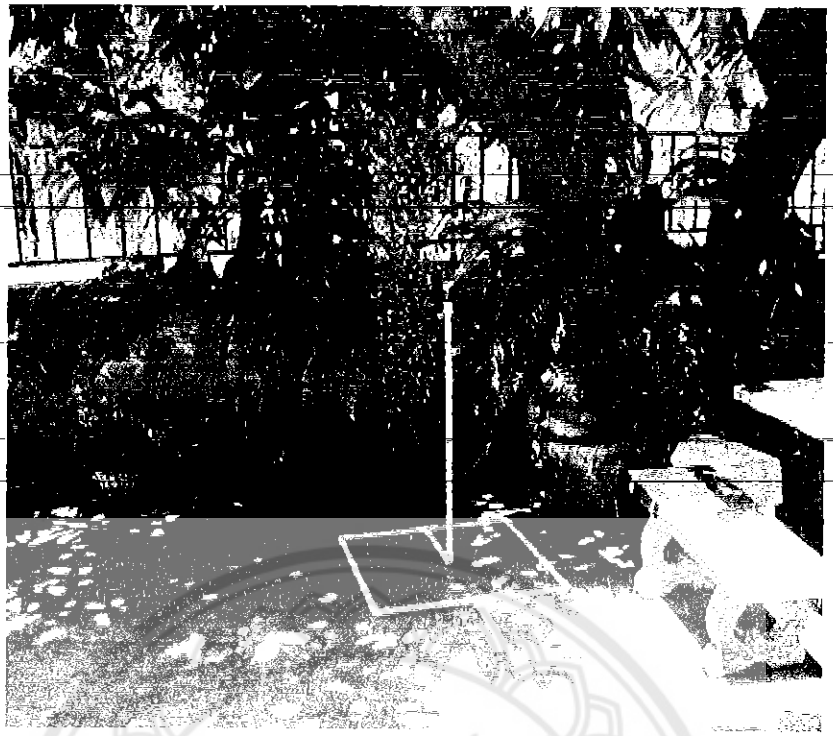
1. เมื่อถึงเวลาที่กำหนด ระบบถูกทำงานอย่างตรงเวลา กล่าวคือ เมื่อถึงเวลา 07.00 ระบบสั่งการออกมาให้น้ำไหลผ่านโซลินอยด์วาล์ว ออกทางสปริงเกอร์ และเมื่อถึงเวลา 07.15 ระบบก็สั่งการให้น้ำหยุดไหล

2. เมื่อถึงเวลา 12.00 ระบบสั่งการออกมาให้น้ำไหลผ่านโซลินอยด์วาล์ว ออกทางสปริงเกอร์ และเมื่อถึงเวลา 12.30 ระบบก็สั่งการให้น้ำหยุดไหล

3. เมื่อถึงเวลา 17.00 ระบบสั่งการออกมาให้น้ำไหลผ่านโซลินอยด์วาล์ว ออกทางสปริงเกอร์ และเมื่อถึงเวลา 07.20 ระบบก็สั่งการให้น้ำหยุดไหล



รูปที่ 4.2 แสดงการทำงานของเครื่องควบคุมในกรณีที่ 2 เมื่อเวลา 12.02 น.



รูปที่ 4.3 แสดงการทำงานของสปริงเกอร์รดน้ำทำงานในกรณีที่ 2 เมื่อเวลา 12.02 น.

4.2 การวิเคราะห์ผลการทดลอง

จากการทดลองทั้งสองกรณี เครื่องรดน้ำต้นไม้อัตโนมัติสามารถควบคุมการเปิด-ปิดน้ำได้อย่างตรงเวลาและมีความแม่นยำในการทำงาน ซึ่งเราสามารถตั้งเวลาในหนึ่งรอบวันหรือ 24 ชั่วโมงได้ ระบบสามารถตั้งได้มากสุด 4 ครั้ง ซึ่งน่าจะเพียงพอกับการใช้งานในหนึ่งวัน โดยเทียบกับการรดน้ำทั่วๆไป เมื่อสามารถทำได้ดังนี้แล้วเราก็สามารถประหยัดเวลา ประหยัดคนในการทำงานได้ ประโยชน์ที่ได้รับจึงสามารถนำออกมาให้เห็นเป็นรูปธรรมได้อย่างชัดเจน

บทที่ 5
บทสรุป

จากการทดลองในบทที่ 3 และผลการทดลองในบทที่ 4 เราสามารถนำผลการทดลองมาสรุปผล พร้อมบอกถึงปัญหาที่พบในระหว่างการทำโครงงาน และข้อเสนอแนะเพื่อที่จะสามารถนำไปพัฒนาให้มีประสิทธิภาพดียิ่งขึ้นต่อไปในอนาคต

5.1 สรุปผลการทดลอง

เมื่อเราตั้งเวลาให้กับเครื่องรดน้ำต้นไม้อัตโนมัติ ตามจุดประสงค์ของผู้ประกอบการหรือผู้ใช้งานแล้ว ผลการทดลองที่ได้เป็นที่น่าพอใจ กล่าวคือ เป็นไปตามฐานเวลาที่เรากำหนด เมื่อถึงเวลาเครื่องรดน้ำก็สามารถทำงานได้อย่างตรงเวลาทั้ง 4 ครั้งตามที่ตั้งเวลาไว้

5.2 ปัญหาที่พบ

1. ในกรณีแรกผู้จัดทำคิดที่จะควบคุมอุณหภูมิและความชื้นด้วยแต่ด้วยข้อจำกัดของเวลาและค่าใช้จ่ายจึงไม่สามารถทำได้อย่างทันเวลาที่กำหนด
2. การตั้งเวลาการทำงานสามารถตั้งได้มากสุด 4 ครั้งต่อการตั้งการทำงาน ด้วยเหตุจำกัดของตัวไมโครคอนโทรลเลอร์
3. ในเรื่องของสถานที่ในการทดลอง สถานที่ไม่ใช่สวนผักหรือแปลงผัก แต่เป็นเพียงการแสดงให้เห็นว่าน้ำนั้นไหลตามเวลาที่กำหนดไว้จริง และสามารถควบคุมได้จริง

5.3 ข้อเสนอแนะ

ในบอร์ดไมโครคอนโทรลเลอร์ MCS-51 นั้น ยังมีฟังก์ชันที่น่าสนใจอีกหลายฟังก์ชันและตัวที่น่าสนใจในการต่อยอดโครงงานนี้คือ การควบคุมอุณภูมิและความชื้น ซึ่งตัวอย่างที่ใช้งานได้แก่ การทำฟาร์มเห็ด ซึ่งในโรงเพาะเห็ดนั้นจำเป็นต้องมีการควบคุมอุณภูมิและความชื้น โดยหลักการคือ ชนิดของเห็ดนางฟ้าต้องการความชื้นประมาณ 80 % และอุณภูมิที่เหมาะสมคือ 24-26 องศาเซลเซียส ซึ่งถ้าความชื้นต่ำก็จะสั่งการไปยังตัวไมโครคอนโทรลเลอร์เพื่อสั่งให้พ่นน้ำในอากาศ เพื่อเพิ่มความชื้นโดยอัตโนมัติ จากกรณีตัวอย่างดังกล่าวจะเห็นได้ว่าเป็นแนวคิดที่สามารถนำไปพัฒนาในการทำงานได้จริง ซึ่งเพิ่มจากการรดน้ำในเวลาปกติไปด้วย

ภาคผนวก

```
//********** Program Automatic Watering Controller **********//
#include <reg51.h>
#include <stdio.h>
#include <stdlib.h>
#include <intrins.h>
//********** initial parameter **********//
unsigned char sec,min,hour,day,date,month,year;
unsigned char h1,h2,h3,h4,h5,h6,h7,h8,m1,m2,m3,m4,m5,m6,m7,m8,s1,s2,s3,s4,s5,s6,s7,s8;
unsigned char date_buf[6];
unsigned char time_buf[6];
unsigned char on_buf1[6],on_buf2[6],on_buf3[6],on_buf4[6];
unsigned char off_buf1[6],off_buf2[6],off_buf3[6],off_buf4[6];
unsigned char i;

sbit SDA = P2^0;                    // connect to SDA
sbit SCL = P2^1;                    // connect to SCL
sbit RS  = P2^2;                    // connect RS pin of LCD
sbit E   = P2^3;                    // connect Enable pin of LCD
sbit sw_1 = P2^4;

bit normal =1;
#define ACK         1
#define NO_ACK      0
//********** FUNCTION **********//
//********** Delay Ms **********//
void DelayMs(unsigned int count)
{
  unsigned int i;
  while(count)
       {
    i = 115;
                while(i>0) i--;
```

```
            count--;
        }
    }
    //********** start I2C **********//
    void Start(void)
    {
      SDA = 1;
        SCL = 1;
        _nop_();
        _nop_();
        SDA = 0;
        _nop_();
        _nop_();
        SCL = 0;
        _nop_();
        _nop_();
    }
    //********** stop I2C **********//
    void Stop(void)
    {
        SDA = 0;
        _nop_();
        _nop_();
        SCL = 1;
        _nop_();
        _nop_();
        SDA = 1;
    }
    //********** Read I2C **********//
    unsigned char ReadI2C(bit ACK_Bit)
    {
      unsigned char Data=0;
```

```
        SDA = 1;
        for (i=0;i<8;i++)
        {
            SCL  = 1;
            Data<<= 1;
            Data  = (Data | SDA);
            SCL  = 0;
            _nop_();
        }
    if (ACK_Bit == 1)
        SDA = 0;
    else
        SDA = 1;
    _nop_();
    _nop_();
    SCL = 1;
    _nop_();
    _nop_();
    SCL = 0;
    return Data;
}
//********** Write I2C **********//
void WriteI2C(unsigned char Data)
{
    for (i=0;i<8;i++)
    {
  SDA = (Data & 0x80) ? 1:0;
        SCL=1;SCL=0;
        Data<<=1;
    }
    SCL = 1;
    _nop_();
```

```
        _nop_();
        SCL = 0;
    }
    //********** Read RTC **********//
    void ReadRTC()
    {
        Start();
        WriteI2C(0xD0);
        WriteI2C(0x00);

        Start();
        WriteI2C(0xD1);
        sec=ReadI2C(ACK);
        min=ReadI2C(ACK);
        hour=ReadI2C(ACK);
        day=ReadI2C(ACK);
        date=ReadI2C(ACK);
        month=ReadI2C(ACK);
        year=ReadI2C(NO_ACK);
        Stop();
    }
    //********** Read data from DS1307 **********//
    unsigned char Read1307(unsigned char addr)
    {
        unsigned char ret;
        Start();
        WriteI2C(0xD0);
        WriteI2C(addr);
        Start();
        WriteI2C(0xD0+1);
        ret=ReadI2C();
        Stop();
```

```
        return(ret);
    }
    //********** Write RTC Time  **********//
    void WriteRTCtime()
    {
        hour = (time_buf[1]<<4)| (time_buf[2]);
        min = (time_buf[3]<<4)| (time_buf[4]);
        sec = (time_buf[5]<<4)| (time_buf[6]);
        Start();
        WriteI2C(0xD0);
        WriteI2C(0x00);
        WriteI2C(sec);
        WriteI2C(min);
        WriteI2C(hour);
        Stop();
    }
    //********** Write RTC Date **********//
    void WriteRTCdate()
    {
        date = (date_buf[1]<<4) | (date_buf[2]);
        month = (date_buf[3]<<4) | (date_buf[4]);
        year = (date_buf[5]<<4) | (date_buf[6]);
        Start();
        WriteI2C(0xD0);
        WriteI2C(0x04);
        WriteI2C(date);
        WriteI2C(month);
        WriteI2C(year);
        Stop();
    }
    //********** scan key 4x4 **********//
    sbit Col1 = P1^0;
```

```
sbit Col2 = P1^1;
sbit Col3 = P1^2;
sbit Col4 = P1^3;
sbit Row1 = P1^7;
sbit Row2 = P1^6;
sbit Row3 = P1^5;
sbit Row4 = P1^4;

unsigned char scankey(void)
{
    unsigned char ret = 0xff;
    Col1 = 0;
    if(Row1==0)
    {
        DelayMs(500);
        ret = 1;
    }
    if(Row2==0)
    {
        DelayMs(500);
        ret = 5;
    }
    if(Row3==0)
    {
        DelayMs(500);
        ret = 9;
    }
    if(Row4==0)
    {
        DelayMs(500);
        ret = 13;
    }
```

```
        Col1 = 1;
        Col2 = 0;
        if(Row1==0)
        {
                DelayMs(500);
                ret = 2;
        }
        if(Row2==0)
        {
                DelayMs(500);
                ret = 6;
        }
        if(Row3==0)
        {
                DelayMs(500);
                ret = 0;
        }
        if(Row4==0)
        {
                DelayMs(500);
                ret = 14;
        }
        Col2 = 1;
        Col3 = 0;
        if(Row1==0)
        {
                DelayMs(500);
                ret = 3;
        }
        if(Row2==0)
        {
                DelayMs(500);
```

```
            ret = 7;
        }
        if(Row3==0)
        {
            DelayMs(500);
            ret = 11;
        }
        if(Row4==0)
        {
            DelayMs(500);
            ret = 15;
        }
        Col3 = 1;
        Col4 = 0;
        if(Row1==0)
        {
            DelayMs(500);
            ret = 4;
        }
        if(Row2==0)
        {
            DelayMs(500);
            ret = 8;
        }
        if(Row3==0)
        {
            DelayMs(500);
            ret = 12;
        }
        if(Row4==0)
        {
            DelayMs(500);
```

```
                ret = 16;
            }
            Col4 = 1 ;
            return(ret);
        }
        //********** send command 1 byte to LCD          **********//
        void lcd_command(unsigned char com)
        {
            RS = 0;
            E = 1;
            P0 = com;
            DelayMs(1);
            E = 0;
            DelayMs(1);
        }
        //********** send character 1 byte to LCD **********//
        void lcd_text(unsigned char text)
        {
            RS = 1;
            E = 1;
            P0 = text;
            DelayMs(1);
            E = 0;
            DelayMs(1);
        }
        //********** Display String to LCD **********//
        void lcd_puts(char addr,char *ptr)
        {
            lcd_command(0x02);
            lcd_command(addr);
            while(*ptr)
            {
```

```
                lcd_text(*ptr);
                ptr++;
            }
        }
//********** function Initial LCD **********//
void lcd_init()
{
        DelayMs(500);
        lcd_command(0x38);
        lcd_command(0x0C);
        lcd_command(0x01);
}
//********** convert to ascii and send to LCD **********//
void send_to_lcd(unsigned char value)
{
        unsigned char buf = 0;
        buf = value & 0xF0;
        buf = (buf>>4)|(0x30);
        lcd_text(buf);
        buf = value & 0x0F;
        buf = buf | 0x30;
        lcd_text(buf);
}
//********** Display Show Time **********//
void display_time(void)
{
        sec=Read1307(0);
        min=Read1307(1);
        hour=Read1307(2);
        day=Read1307(3);
        date=Read1307(4);
        month=Read1307(5);
```

```
        year=Read1307(6);

        lcd_command(0x02);
        lcd_command(0x0C);
        lcd_puts(0x80,"Time ::");
        lcd_command(0xC0);
        send_to_lcd(hour);
        lcd_text(':');
        send_to_lcd(min);
        lcd_text(':');
        send_to_lcd(sec);
    }
    //********** Setting Date **********//
    void display_setdate(void)
    {
        lcd_command(0x01);
        lcd_puts(0x80,"Date :");
        lcd_puts(0xC0,"dd/mm/yy");
        lcd_command(0x02);
        lcd_command(0x0F);
        lcd_command(0xC0);
    }
    //********** Setting Time **********//
    void display_settime(void)
    {
        lcd_command(0x01);
        lcd_puts(0x80,"Time :");
        lcd_puts(0xC0,"hh/mm/ss");
        lcd_command(0x02);
        lcd_command(0x0F);
        lcd_command(0xC0);
    }
```

```
//********** Setting Time On **********//
void display_settimeon(void)
{
        lcd_command(0x01);
        lcd_puts(0x80,"Time On");
        lcd_puts(0xC0,"hh:mm");
        lcd_command(0x02);
        lcd_command(0x0F);
        lcd_command(0xC0);
}
//********** Setting Time Off **********//
void display_settimeoff(void)
{
        lcd_command(0x01);
        lcd_puts(0x80,"Time Off");
        lcd_puts(0xC1,"hh:mm");
        lcd_command(0x02);
        lcd_command(0x0F);
        lcd_command(0xC1);
}
//********** Main Program **********//
void main(void)
{
        unsigned char x_key = 0;
        unsigned char push_key = 0;
        unsigned char position = 1;
        sw_1 = 0;
        P0 = 0x00;
        lcd_init();
        while(1)
        {
                if(scankey()==13)               // set time on-off 1
```

```
position = 4;
if(scankey()==14)                    // set time on-off 2
position = 5;
if(scankey()==15)                    // set time on-off 3
position = 6;
if(scankey()==16)                    // set time on-off 4
position = 7;
if(scankey()==11)                    //      set date
position = 2;
if(scankey()==12)                    //      set time
position = 3;

if((((hour>=h7)&(hour<=h8))&&((min>=m7)&(min<=(m8-
1))))||(((hour>=h5)&(hour<=h6))&&((min>=m5)&(min<=(m6-
1))))||(((hour>=h3)&(hour<=h4))&&((min>=m3)&(min<=(m4-
1))))||(((hour>=h1)&(hour<=h2))&&((min>=m1)&(min<=(m2-1)))))
{
sw_1 = 1;
}
else
{
sw_1 = 0;
}

h1=     (on_buf1[1]<<4)| (on_buf1[2]);
m1=     (on_buf1[3]<<4)| (on_buf1[4]);
s1=     0;
h2=     (off_buf1[1]<<4)| (off_buf1[2]);
m2=     (off_buf1[3]<<4)| (off_buf1[4]);
s2=     0;
h3=     (on_buf2[1]<<4)| (on_buf2[2]);
m3=     (on_buf2[3]<<4)| (on_buf2[4]);
```

```
        s3=     0;
        h4=     (off_buf2[1]<<4)| (off_buf2[2]);
        m4=     (off_buf2[3]<<4)| (off_buf2[4]);
        s4=     0;
        h5=     (on_buf3[1]<<4)| (on_buf3[2]);
        m5=     (on_buf3[3]<<4)| (on_buf3[4]);
        s5=     0;
        h6=     (off_buf3[1]<<4)| (off_buf3[2]);
        m6=     (off_buf3[3]<<4)| (off_buf3[4]);
        s6=     0;
        h7=     (on_buf4[1]<<4)| (on_buf4[2]);
        m7=     (on_buf4[3]<<4)| (on_buf4[4]);
        s7=     0;
        h8=     (off_buf4[1]<<4)| (off_buf4[2]);
        m8=     (off_buf4[3]<<4)| (off_buf4[4]);
        s8=     0;

switch(position)
{
case 0 : break;
case 1 : if(normal)
         {
                lcd_command(0x01);
                normal = 0;
         }
         ReadRTC();
         DelayMs(50);
         display_time();
         break;
case 2 : display_setdate();
         while(x_key < 6)
         {
```

```
                push_key = scankey();
                if(push_key !=0xFF)
                {
                 x_key++;
                 date_buf[x_key] = push_key;
                 push_key = push_key | 0x30;
                      lcd_text(push_key);
                if(x_key==2)
                {
                        lcd_command(0x02);
                        lcd_command(0xC3);
                }
                if(x_key==4)
                {
                        lcd_command(0x02);
                        lcd_command(0xC6);
                }
                }
               }
               WriteRTCdate();
               x_key = 0;
               normal = 1;
               position = 1;
               break;
        case 3 : display_settime();
                 while(x_key < 6)
                 {
                  push_key = scankey();
                  if(push_key !=0xFF)
                  {
                        x_key++;
                   time_buf[x_key] = push_key;
```

```
                push_key = push_key | 0x30;
                    lcd_text(push_key);
                if(x_key==2)
                {
                        lcd_command(0x02);
                        lcd_command(0xC3);
                }
                if(x_key==4)
                {
                        lcd_command(0x02);
                        lcd_command(0xC6);
                }
              }
            }
            WriteRTCtime();
            x_key = 0;
            position = 1;
            normal = 1;
            break;
    case 4 : display_settimeon();
            while(x_key < 4)
            {
             push_key = scankey();
             if(push_key !=0xFF)
             {
               x_key++;
               on_buf1[x_key] = push_key;
               push_key = push_key | 0x30;
                    lcd_text(push_key);
               if(x_key==2)
               {
                        lcd_command(0x02);
```

```
                    lcd_command(0xC3);
                }
                }
                }
                x_key = 0;
                display_settimeoff();
                while(x_key < 4)
                {
                 push_key = scankey();
                 if(push_key !=0xFF)
                 {
                        x_key++;
                 off_buf1[x_key] = push_key;
                 push_key = push_key | 0x30;
                        lcd_text(push_key);
                 if(x_key==2)
                 {
                        lcd_command(0x02);
                        lcd_command(0xC4);
                 }
                 }
                }
                x_key = 0;
                position = 1;
                normal = 1;
                break;
        case 5 : display_settimeon();
                while(x_key < 4)
                {
                 push_key = scankey();
                 if(push_key !=0xFF)
                 {
```

```
                x_key++;
                on_buf2[x_key] = push_key;
                push_key = push_key | 0x30;
                        lcd_text(push_key);
                if(x_key==2)
                {
                                lcd_command(0x02);
                                lcd_command(0xC3);
                }
              }
            }
            x_key = 0;
            display_settimeoff();
            while(x_key < 4)
            {
             push_key = scankey();
             if(push_key !=0xFF)
             {
                    x_key++;
              off_buf2[x_key] = push_key;
              push_key = push_key | 0x30;
                    lcd_text(push_key);
              if(x_key==2)
              {
                                lcd_command(0x02);
                                lcd_command(0xC4);
              }
             }
            }
            x_key = 0;
            position = 1;
            normal = 1;
```

```
                    break;
            case 6 :  display_settimeon();
                    while(x_key < 4)
                    {
                     push_key = scankey();
                     if(push_key !=0xFF)
                     {
                      x_key++;
                      on_buf3[x_key] = push_key;
                      push_key = push_key | 0x30;
                            lcd_text(push_key);
                      if(x_key==2)
                      {
                              lcd_command(0x02);
                              lcd_command(0xC3);
                      }
                     }
                    }
                   x_key = 0;
                   display_settimeoff();
                   while(x_key < 4)
                    {
                     push_key = scankey();
                     if(push_key !=0xFF)
                     {
                            x_key++;
                      off_buf3[x_key] = push_key;
                      push_key = push_key | 0x30;
                            lcd_text(push_key);
                      if(x_key==2)
                      {
                              lcd_command(0x02);
```

```
                    lcd_command(0xC4);
                }
                }
            }
            x_key = 0;
            position = 1;
            normal = 1;
            break;
case 7 :  display_settimeon();
          while(x_key < 4)
          {
           push_key = scankey();
           if(push_key !=0xFF)
           {
            x_key++;
            on_buf4[x_key] = push_key;
            push_key = push_key | 0x30;
                lcd_text(push_key);
            if(x_key==2)
            {
                    lcd_command(0x02);
                    lcd_command(0xC3);
            }
           }
          }
          x_key = 0;
          display_settimeoff();
          while(x_key < 4)
          {
           push_key = scankey();
           if(push_key !=0xFF)
           {
```

```
                x_key++;
            off_buf4[x_key] = push_key;
            push_key = push_key | 0x30;
                lcd_text(push_key);
            if(x_key==2)
            {
                    lcd_command(0x02);
                lcd_command(0xC4);
            }
          }
        }
        x_key = 0;
        position = 1;
        normal = 1;
        break;
        }
    }
}
```

ประวัติผู้เขียนโครงงาน

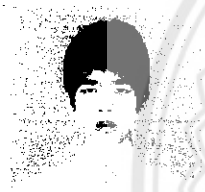
ชื่อ นายกองพล โสดา

ภูมิลำเนา 491/246 ม.2 ต.วังนกแอ่น อ.วังทอง จ.พิษณุโลก

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนพิษณุโลกพิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E-mail: **soda_ee@hotmail.com**



ชื่อ นายมนชิต บุญวงศ์

ภูมิลำเนา 258 ม.7 ต.บางระกำ อ.บางระกำ จ.พิษณุโลก

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนพิษณุโลกพิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E-mail: **alisto_tles@hotmail.com**