ราวตากผ้าอัตโนมัติ

จัดทำโดย

นางสาวนภาพร พิมปรุ รหัสประจำตัว B5109951

นางสาวจันทิมา ชาสิงห์แก้ว รหัสประจำตัว B5132881

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 427499 โครงงานวิศวกรรมโทรคมนาคม

หลักสูตรวิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ประจำภาคการศึกษาที่ 1 ปีการศึกษา 2554

ราวตากผ้าอัตโนมัติ

คณะกรรมการสอบโครงงาน

(ผู้ช่วยศาสตราจารย์ เรืออากาศเอก ดร. ประโยชน์ คำสวัสดิ์)
กรรมการ/อาจารย์ที่ปรึกษาโครงงาน

(อาจารย์ ดร.สมศักดิ์ วาณิชอนันต์ชัย)
กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. มนต์ทิพย์ภา อุฑารสกุล)
กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ปิยาภรณ์ กระฉอดนอก)
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับรายงานโครงงานฉบับนี้ เป็นส่วนหนึ่งของการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม วิชา 427499 โครงงานวิศวกรรมโทรคมนาคม ประจำปีการศึกษา 2553

โครงงาน ราวตากผ้าอัตโนมัติ

จัดทำโดย 1. นางสาวนภาพร พิมปรุ รหัส B5109951

2. นางสาวจันทิมา ชาสิงห์แก้ว รหัส B5132881

อาจารย์ที่ปรึกษา ผศ.ร.อ. ดร.ประโยชน์ คำสวัสดิ์

สาขาวิชา วิศวกรรมโทรคมนาคม

ภาคการศึกษาที่ 1/2554

บทคัดย่อ

(Abstract)

โครงงานนี้นำเสนอราวตากผ้าอัตโนมัติ เพื่อแก้ปัญหาการตากผ้าทิ้งไว้แล้วเปียกฝนเมื่อไม่อยู่บ้าน ซึ่งระบบที่นำเสนอนี้มีอุปกรณ์ที่สำคัญดังนี้คือ บอร์ดไมโครคอนโทรลเลอร์ ตระกูล AVR ATmega128 มอเตอร์ที่ปัดน้ำฝน เซ็นเซอร์แสงและเซ็นเซอร์น้ำฝน โดยการทำงานของระบบเป็นการทำงานแบบอัตโนมัติซึ่งมีหลักการทำงานดังนี้คือ เมื่อฝนตกถูกตัวเซ็นเซอร์จะทำให้ไมโครคอนโทรลเลอร์ทำการสั่งให้มอเตอร์หมุนเพื่อหมุนหลังคาปิดผ้าที่ตากไว้ และเมื่อฝนหยุดไมโครคอนโทรลเลอร์ก็จะทำการสั่งให้มอเตอร์หมุนกลับเพื่อหมุนหลังคาเปิดและตากผ้าเหมือนเดิม และนอกจากนี้ยังสามารถหมุนหลังคาปิดได้เมื่อถึงเวลาตอนเย็นเพื่อไม่ให้ผ้ามีความชื้นและทำการหมุนหลังคาเปิดเมื่อถึงตอนเช้าโดยใช้หลักการเซ็นเซอร์แสง ซึ่งหลักการทำงานดังกล่าวอาจนำไปประยุกต์ใช้ในการทำอุปกรณ์ปิดสิ่งของอย่างอื่นที่ไม่ต้องการให้เปียกฝนได้อีกด้วย

กิตติกรรมประกาศ

จากการที่คณะจัดทำรายงานได้รับมอบหมายให้ทำโครงงานเรื่อง ราวตากผ้าอัตโนมัติ ส่งผลให้คณะจัดทำรายงานได้รับความรู้และประสบการณ์ต่างๆเกี่ยวกับการเขียนโปรแกรมควบคุมบอร์ดไมโครคอนโทรลเลอร์รุ่น ATmega128 บัดนี้โครงงานดังกล่าวพร้อมทั้งรายงานได้สำเร็จลงแล้ว ทั้งนี้ด้วยความร่วมมือและสนับสนุนจากบุคคลต่างๆ ดังนี้

1. ผู้ช่วยศาสตราจารย์ ร.อ. ดร.ประโยชน์ คำสวัสดิ์ (อาจารย์ที่ปรึกษาโครงงาน)
2. นายณัฐวุฒิ พจน์ปริญญา วิศวกร รมศาสตรบัณฑิต (วิศวกรรม โทรคมนาคม)

ข้าพเจ้าใคร่ขอขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่านที่มีส่วนร่วมในการให้ข้อมูลและเป็นที่ปรึกษาในการทำรายงานฉบับนี้จนเสร็จสมบูรณ์ ตลอดจนให้การดูแลและให้ความเข้าใจเกี่ยวกับพื้นฐานการใช้งานโปรแกรม ซึ่งข้าพเจ้าขอขอบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้ด้วย

นางสาวนภาพร พิมปรุ
นางสาวจันทิมา ชาสิงห์แก้ว

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญตาราง

เรื่อง

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

จากการพัฒนาทางด้านเทคโนโลยีทำให้มีการประดิษฐ์อุปกรณ์ต่างๆใช้ในชีวิตประจำวัน เพื่อตอบสนองความต้องการของมนุษย์ โดยการศึกษาการใช้งานไมโครคอนโทรลเลอร์ การเขียนโปรแกรม เพื่อควบคุมการทำงานของอุปกรณ์อิเล็กทรอนิกส์ นอกจากนี้ยังได้ทำการศึกษาการแปลงสัญญาณแอนาลอกเป็นสัญญาณดิจิตอล การออกแบบลายวงจร การทำงานของรีเลย์และการทำงานของมอเตอร์อีกด้วย

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อนำความรู้การเขียนโปรแกรมจากวิชาไมโครคอนโทรลเลอร์ และการเขียนโปรแกรม C++ มาประยุกต์ใช้
2. เพื่อศึกษาการใช้งานบอร์ด AVR ATmega128
3. เพื่อศึกษาและออกแบบวงจรเชื่อมต่อบอร์ด AVR
4. เพื่อศึกษาการทำงานของมอเตอร์ที่ปัดน้ำฝน
5. เพื่อนำสิ่งประดิษฐ์ที่ได้มาใช้อำนวยความสะดวกในชีวิตประจำวัน

1.3 ขอบเขตการดำเนินงาน

1. ศึกษาการใช้งานบอร์ด AVR ATmega128
2. ศึกษาการใช้งานโปรแกรม AVR studio4
3. ศึกษาการทำงานของมอเตอร์ที่ปัดน้ำฝน
4. เขียนโปรแกรมเพื่อควบคุมการทำงานของมอเตอร์

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาค้นคว้าการใช้บอร์ด AVR ATmega128 และการเขียนโปรแกรมคำสั่งให้ทำงาน
2. เขียนโครงการและ เสนอโครงการกับอาจารย์ที่ปรึกษา
3. ศึกษาการทำงานของ step motor และเขียนโปรแกรมควบคุมมอเตอร์หมุน
4. ศึกษาการใช้งานโปรแกรม AVR studio4 และการโหลดโปรแกรมลงบอร์ด
5. สร้างแบบจำลองราวตากผ้าอัตโนมัติ

6. นำอุปกรณ์ทั้งหมดมาเชื่อมต่อกัน ทดลองการทำงาน และบันทึกผลการทดลอง
7. สรุปผลการทดลองและเขียนรายงาน
8. นำเสนอโครงงาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. เข้าใจหลักการทำงานของบอร์ด AVR ATmega128
2. เข้าใจการทำงานของมอเตอร์ที่ปัดน้ำฝน
3. เข้าใจการเขียนโปรแกรมและโหลดโปรแกรมใช้งานได้
4. สามารถใช้งานราวตากผ้าอัตโนมัติได้จริง
5. เพื่อให้ผู้ใช้จะเลิกกังวลว่าเสื้อผ้าจะเปียกฝน เมื่อเวลาที่ไม่อยู่บ้าน

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 บทนำ

การทำงานของราวตากผ้าอัตโนมัติ เป็นการทำงานที่ต้องใช้เซ็นเซอร์ตรวจจับน้ำฝนเพื่อตรวจสอบว่ามีฝนตกหรือไม่ หากตรวจพบว่ามีฝนตกก็จะแสดงการทำงานของไมโครคอนโทรลเลอร์ ซึ่งจะมีการแปลงสัญญาณแอนาลอกเป็นสัญญาณดิจิตอลในการเขียนโปรแกรมควบคุมการหมุนของมอเตอร์ที่แสดงการหมุนไปและหมุนกลับ ซึ่งการหมุนของมอเตอร์จะถูกควบคุมโดยรีเลย์ด้วย

2.2 การใช้งาน AVR ATmega128

ET-BASE AVR ATmega64/128 r3 เป็นบอร์ดไมโครคอนโทรลเลอร์ในตระกูล AVR ของบริษัทAtmel ซึ่งบอร์ดนี้เลือกใช้ MCU เบอร์ ATmega64 และ เบอร์ ATmega128 ขนาด 64 Pin โดยในบอร์ด ET-BASE AVR ATmega64/128 r3 นี้จะเน้นจะเน้นการใช้งานทรัพยากรของตัว MCU เอง เป็นหลัก ซึ่งจะมีการต่อขาสัญญาณ I/O ออกมาจัดเรียงให้เป็นพอร์ ต PA,PB,PC,PD,PE,PF และพอร์ต ET-CLCD เพื่อสะดวกต่อการใช้งาน พร้อมทั้งพอร์ตสำหรับดาวน์โหลดโปรแกรม นอกจากนี้ยังได้เพิ่มวงจร Line Driver RS-232 เข้าไปด้วยเพื่อให้สามารถใช้งานทางด้านพอร์ตอนุกรม RS-232 ได้ง่ายและสะดวกยิ่งขึ้น

คุณสมบัติของบอร์ด

1. เลือกใช้ MCU ตระกูล AVR เบอร์ ATmega64,ATmega128 ของ Atmel ซึ่งเป็น MCU ขนาด 8–Bit โดยเลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาแบบ XTAL ค่า 16 MHz ซึ่งคุณสมบัติเด่น ๆ ของ MCU ได้แก่- มีหน่วยความจำ Flash สำหรับเขียนโปรแกรม 64 KBytes สำหรับ ATmega64 และ 128K Bytes สำหรับ ATmega128 และมี RAM 4 KBytes

- มีหน่วยความจำข้อมูลถาวรแบบ EEPROM ขนาด 2K Bytes สำหรับATmega64 และ 4 K Byte สำหรับ ATmega128 ซึ่งสามารถลบและเขียนซ้ำได้กว่า 100,000 ครั้ง

- จำนวน I/O สูงสุดถึง 53 I/O Pins

- มีวงจรสื่อสาร SPI จำนวน 1 ช่อง, I2C จำนวน 1 ช่อง, Programmable Serial USARTs จำนวน 2 ช่อง

- มี ADC ขนาด 10-Bit จำนวน 8 ช่อง

- มี Timers/Counters 8-Bit จำนวน 2 ช่อง, Timers/Counters 16-Bit จำนวน 2 ช่อง , 8-Bit PWM 2 ช่อง, Watchdog Timer, Real Time Counter

2. I/O PORT 10 PIN จำนวน 6 PORT ดั้งนี้ PA,PB,PC,PD,PE,PF

3. พอร์ต ISP LOAD สำหรับโปรแกรม MCU (ต้องใช้ร่วมกับ ET-AVR ISP หรือเครื่องโปรแกรม ISP อื่นที่มีการจัดเรียงขาสัญญาณเหมือนกัน)

4. วงจร Line Driver สำหรับพอร์ตสื่อสารอนุกรม RS232 จำนวน 2 ช่อง โดยเชื่อมต่อกับสัญญาณ PE0(RXD0) และ PE1(TXD0) จำนวน 1 ช่อง ส่วนที่เหลืออีก 1 ช่อง จะต่อกับสัญญาณ PD2(RXD1) และ PD3(TXD1) เพื่อให้ผู้ใช้สามารถต่อทดลองการติดต่อสื่อสารRS232

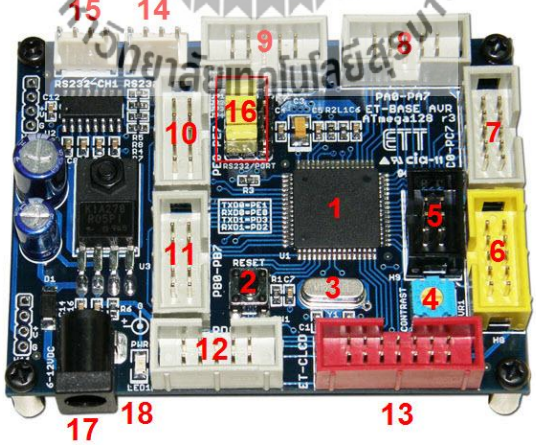
5. วงจรเชื่อมต่อจอแสดงผล LCD แบบ Character (ET-CLCD) พร้อม VR ปรับความเข้มของ LCD ซึ่งใช้การเชื่อมต่อวงจรกับ LCD แบบ 4 Bit Interface

6. วงจร Regulate ขนาด +5V / 2A สำหรับใช้งานเป็นแหล่งจ่ายไฟเลี้ยงวงจรให้กับจอแสดงผล LCD และอุปกรณ์ I/O ต่างๆที่ใช้กับแหล่งจ่ายขนาดขนาด +5V พร้อม LED แสดงสถานะสีแดง

7. ขนาด PCB Size เล็กเพียง 8 X 6 cm

โครงสร้างของบอร์ด

ดังรูปที่ 2.1



รูปที่ 2.1 โครงสร้างบอร์ด AVR ATmega128

- **หมายเลข 1** คือ MCU เบอร์ ATmega64 หรือ ATmega128 ซึ่งเป็น MCU ตระกูล AVRจาก ATMEL

- **หมายเลข 2** คือ Switch RESET ใช้สำหรับ Reset การทำงานของ MCU
- **หมายเลข 3** คือ Crystal ค่า 16 MHz
- **หมายเลข 4** คือ ตัวต้านทานสำหรับปรับค่าความเข้มให้ LCD
- **หมายเลข 5** พอร์ต AVR ISP (6 PIN) ใช้สำหรับดาวน์โหลด Hex File ให้กับ MCU
- **หมายเลข 6** พอร์ต AVR ISP (10 PIN) ใช้สำหรับดาวน์โหลด Hex File ให้กับ MCU
- **หมายเลข 7** คือ PORTC มีขนาด 8 Bit คือ PC0-PC7
- **หมายเลข 8** คือ PORTA มีขนาด 8 Bit คือ PA0-PA7
- **หมายเลข 9** คือ PORTF มีขนาด 8 Bit คือ PF0-PF7
- **หมายเลข 10** คือ PORTE มีขนาด 8 Bit คือ PE0-PE7
- **หมายเลข 11** คือ PORTB มีขนาด 8 Bit คือ PB0-PB7
- **หมายเลข 12** คือ PORTD มีขนาด 8 Bit คือ PD0-PD7
- **หมายเลข 13** คือ พอร์ต ET-CLCD สำหรับเชื่อมต่อกับ LCD ชนิด Character Type ซึ่งใช้การเชื่อมต่อแบบ 4 Bit
- **หมายเลข 14 และ 15** คือ ขั้วต่อ RS232 สำหรับใช้งานทั่วไป
- **หมายเลข 16** คือ จั๊มเปอร์ สำหรับเลือกใช้งาน RS232 หรือ พอร์ต IO
- **หมายเลข 17** คือ ขั้วต่อแหล่งจ่ายไฟสำหรับเลี้ยงวงจรของบอร์ด
- **หมายเลข 18** คือ LED Power ใช้สำหรับแสดงสถานะของแหล่งจ่ายไฟ +5VDC

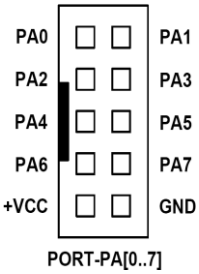
ขั้วต่อสัญญาณต่าง ๆ

สำหรับขั้วต่อสัญญาณของพอร์ต I/O จาก MCU นั้นจะถูกออกแบบและจัดเตรียมไว้ผ่านทางขั้วต่อแบบ IDC-Header ขนาด 10 Pin (2X5) จำนวน 6 ชุด คือ PA,PB,PC,PD,PE,PF ตามลำดับ โดยที่ขั้วต่อสัญญาณแต่ละชุด จะประกอบไปด้วยสัญญาณของ I/O ที่เชื่อมต่อมาจากขาสัญญาณของ MCU โดยตรงทั้งหมด โดยจุดเชื่อมต่อกับสัญญาณภายนอกบอร์ดมีดังนี้

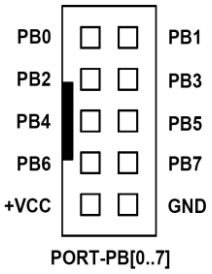
- ขั้วต่อแหล่งจ่ายไฟสำหรับเลี้ยงวงจรของบอร์ด
- ขั้วต่อ PORTA มีขนาด 8 Bit คือ PA0-PA7
- ขั้วต่อ PORTB มีขนาด 8 Bit คือ PB0-PB7
- ขั้วต่อ PORTC มีขนาด 8 Bit คือ PC0-PC7
- ขั้วต่อ PORTD มีขนาด 8 Bit คือ PD0-PD7
- ขั้วต่อ PORTE มีขนาด 8 Bit คือ PE0-PE7

- ขั้วต่อ PORTF มีขนาด 8 Bit คือ PF0-PF7
- ขั้วต่อ ET-CLCD สำหรับเชื่อมต่อกับ LCD ชนิด Character Type
- ขั้วต่อ RS232 จำนวน 2 ช่อง โดยเชื่อมต่อกับสัญญาณ PE0(RXD0) และ PE1(TXD0) จำนวน 1 ช่อง ส่วนที่เหลืออีก 1 ช่อง จะต่อกับสัญญาณ PD2(RXD1) และ PD3(TXD1) เพื่อให้ผู้ใช้สามารถต่อทดลองการติดต่อสื่อสาร RS232
- ขั้วต่อ AVR ISP ใช้สำหรับดาวน์โหลด Hex File ให้กับ MCU

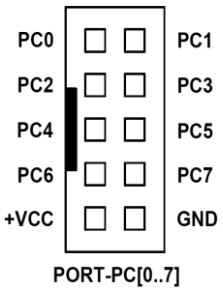
พอร์ต PA มีขนาด 8 บิต



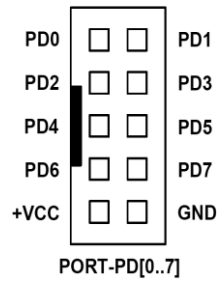
พอร์ต PB มีขนาด 8 บิต



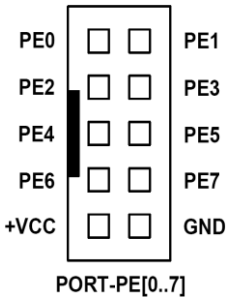
พอร์ต PC มีขนาด 8 บิต



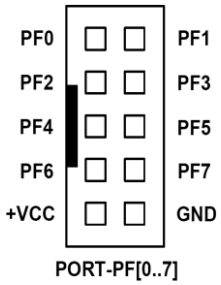
พอร์ต PD มีขนาด 8 บิต



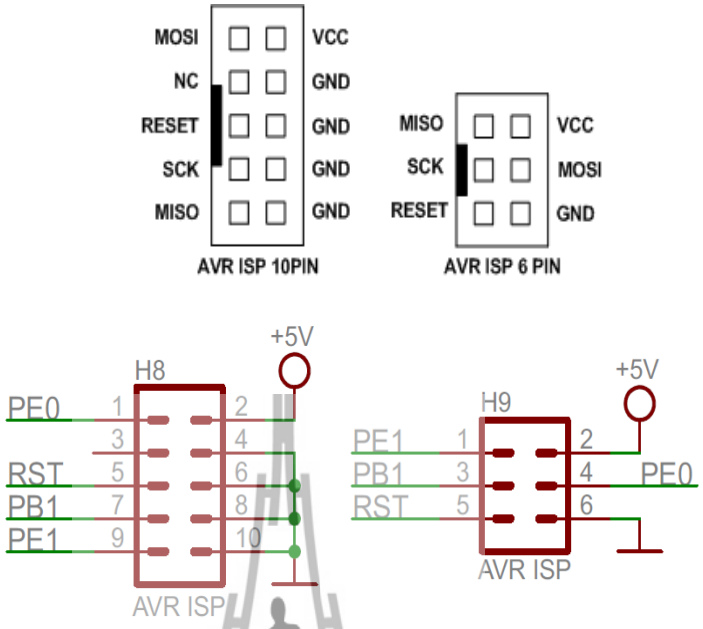
พอร์ต PE มีขนาด 8 บิต



PF มีขนาด 8 บิต

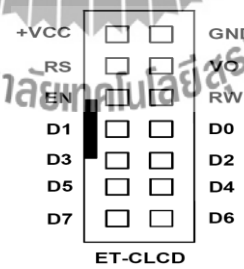


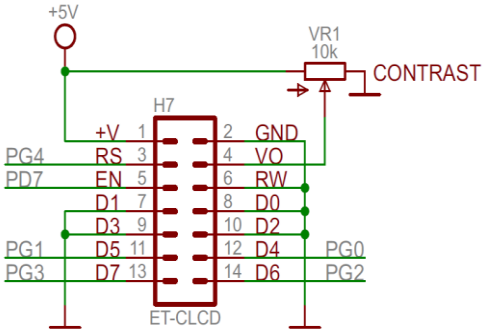
พอร์ต AVR ISP



รูปที่ 2.2 วงจรส่วนที่เชื่อมต่อกับ AVR ISP

พอร์ต ET-CLCD ใช้กับ Character Type LCD โดยใช้การเชื่อมต่อแบบ 4 บิต โดยสัญญาณที่ใช้เชื่อมต่อกับ LCD จะเป็นสัญญาณจากพอร์ต PG และ PD (PD7) โดยในการเชื่อมต่อสายสัญญาณจากขั้วต่อของพอร์ต LCD ไปยังจอแสดงผล LCD นั้นให้ยึดชื่อขาสัญญาณเป็นจุดอ้างอิง โดยให้ต่อสัญญาณที่มีชื่อตรงกันเข้าด้วยกันให้ครบทั้ง 14 เส้น

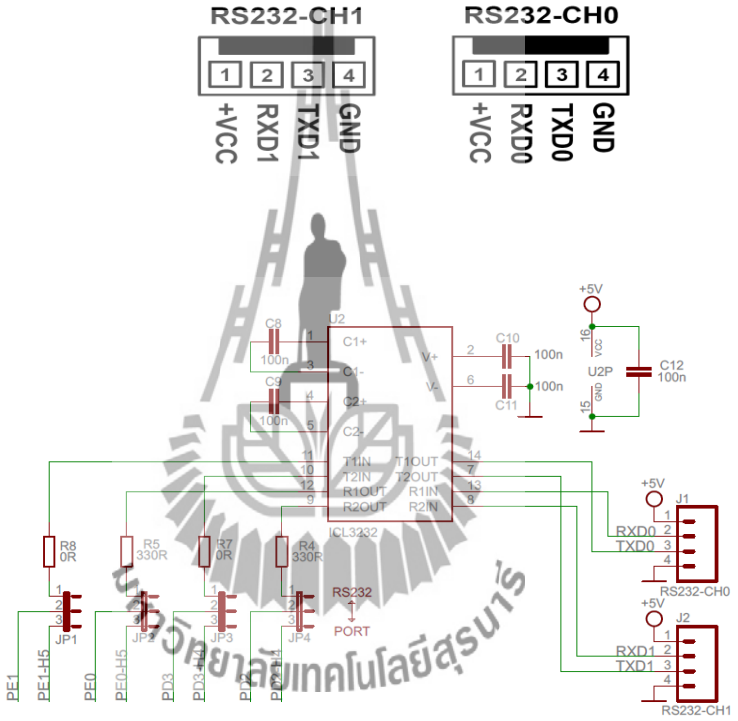




1	2	3	4	5	6	7	8	9	10	11	12	13	14
GND	**+VCC**	**VO**	**RS**	**RW**	**EN**	**D0**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**

การจัดเรียงขาสัญญาณของ Character LCD มาตรฐาน

พอร์ต RS232 จำนวน 2 ช่อง โดยเชื่อมต่อกับสัญญาณ PE0(RXD0) และ PE1(TXD0) จำนวน 1 ช่องส่วนที่เหลืออีก 1 ช่อง จะต่อกับสัญญาณ PD2(RXD1) และ PD3(TXD1)



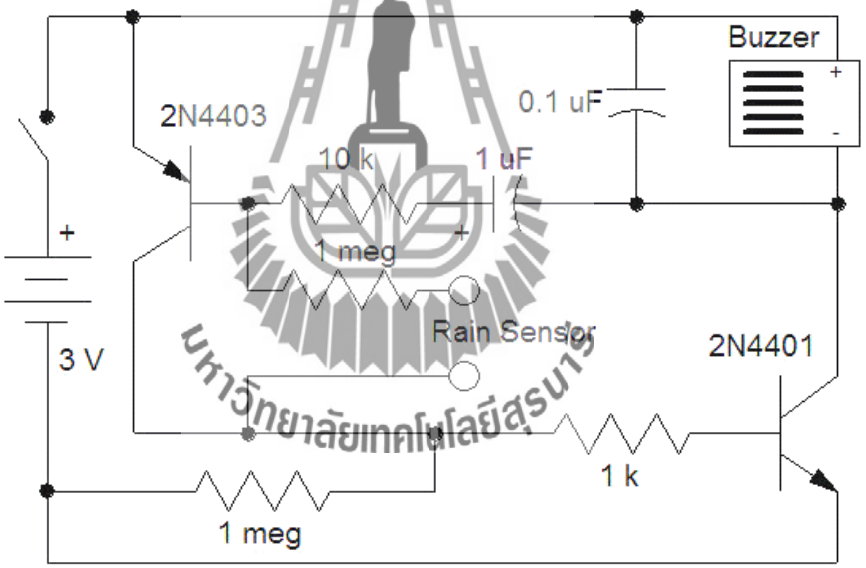
รูปที่ 2.3 วงจรส่วนที่เชื่อมต่อกับ RS232

การดาวน์โหลด Hex File ให้กับ MCU นั้นจำเป็นจะต้องใช้ ET-AVR ISP หรือเครื่องโปรแกรมแบบ ISP อื่นๆ เช่น AVRISP ของ ATMEL เพื่อใช้ในการดาวน์โหลด Hex File ให้กับ MCU ตระกูล AVR ของ Atmel โดยใช้วิธีการแบบ Serial Programming ซึ่งการดาวน์โหลด Hex File ในกรณีที่ใช้ ET-AVR ISP จะกระทำผ่านทางพอร์ตขนานของคอมพิวเตอร์ โดยที่จะต้องใช้งานร่วมกับ ETCAP10P ของอีทีที และ Software ที่ใช้ร่วมกับ ET-AVR ISP ก็คือ PonyProg2000 ซึ่ง PonyProg2000 เป็นโปรแกรม Download ข้อมูลแบบ HEX File ให้กับ CPU ตระกูล AVR โดยใช้วิธีการแบบ Serial Programming ซึ่งสามารถใช้งานกับบอร์ดตระกูล AVR ของ อีทีที ได้เป็นอย่างดี ซึ่งวิธีการใช้งานโปรแกรมโดยทั่วไปนั้น สามารถศึกษาได้จาก Help ของโปรแกรมได้เอง โดยใน

ที่นี้จะขอแนะนำให้ทราบถึงวิธีการ Setup โปรแกรม PonyProg2000 เพื่อใช้งานกับบอร์ดตระกูล AVR ของ อีทีที ซึ่งสามารถใช้งานได้กับบอร์ดตระกูล AVR ทุกรุ่นของ อีทีที

2.3 คุณลักษณะเฉพาะของเครื่องตรวจจับน้ำฝน

เครื่องตรวจจับน้ำฝนนี้ เมื่อมีฝนตกลงมาจะทำให้มีเวลาในการเก็บของหรือปิดหน้าต่าง โดยที่วงจรแบตเตอรี่จะไม่ทำงานเมื่อเซ็นเซอร์แห้งและจะมีการทำงานเมื่อมีน้ำหยดลงบนเซ็นเซอร์ โดยสลับแหล่งจ่ายไฟกับควบคุมแรงดันไฟฟ้าที่ง่ายและลดแรงดันไฟฟ้าที่ 3 โวลต์ โดยใช้ทรานซิสเตอร์ซิลิคอนสามัญ เมื่อวงจรที่จะถูกเรียก buzzer เป็นพัลส์ที่เป็นครั้งต่อวินาทีสำหรับเวลาที่สั้นมากให้มันเป็น"น้ำหยด"มีเสียงเตือนที่เหมาะสม อาจทำให้มีการเตือนที่ช้าลงได้โดยการเพิ่มตัวเก็บประจุ 1uF และเพิ่มตัวต้านทาน 10 K โดยไม่มีการลดอัตราเสียงเตือน แต่ในบางจุดของวงจรจะหยุดทำงานอย่างถูกต้องขึ้นอยู่กับการได้รับของทรานซิสเตอร์



การพิจารณาชิ้นส่วน :

- ทรานซิสเตอร์ที่ทำงาน ในทรานซิสเตอร์สมัยเก่าก็อาจจะมีที่ดีที่สุดที่จะใช้ 4.7 K และ 2.2 uF ในตำแหน่งของ 10 k และ 1uF
- ต้านทานที่สำคัญประเภทหรือขนาดควรปรับการทำงานให้เหมาะสม
- ตัวเก็บประจุ 1uF ชนิดที่มี 16 โวลต์เป็นอลูมิเนียมด้วยไฟฟ้าเป็นที่นิยมมากที่สุด ตัวเก็บประจุจะมีการย้อนกลับ 1 / 2 โวลต์ดังนั้นเวลาที่ตัวเก็บประจุแทนทาลัม 10 โวลต์ หรือสูงกว่าเป็นทางเลือกที่ดีกว่าสามารถจัดการประมาณ 10% ของคะแนน ในทางตรงกันข้าม ไม่มีขั้วประจุเซรามิกยังเป็นทางเลือกที่ดี แต่ไม่ลังเลที่จะใช้ไฟฟ้าราคาถูกเนื่องจากมีความต้านทานขนาดใหญ่
- 0.1uF ไม่ได้เป็นสิ่งสำคัญที่ทุกคนและวงจรทำงานเพียงแค่ปรับไม่ได้ มันจะมีในกรณีที่เสียงกริ่งมีแนวโน้มที่จะ retrigger วงจร
- แบตเตอรี่ที่มีขนาด AA เพียงเซลล์อัลคาไลน์กับสายบัดกรีโดยตรงกับการสิ้นสุดสำหรับการเชื่อมต่อ บัดกรีโดยตรงกับแบตเตอรี่เป็นกระบวนการที่ละเอียดอ่อนและให้ถือแบตเตอรี่สำหรับ solderer มีประสบการณ์น้อยกว่า ต้องทำร่วมกันได้อย่างรวดเร็วซึ่งแบตเตอรี่อาจได้รับความเสียหาย
- กริ่งเป็น 1.5-3 โวลต์, 15mA "เสียงมินิ"
- วงจรที่สร้างขึ้นบนแผงวงจรพรุนและติดตั้งในกล่องพลาสติกเล็ก ๆ น้อย ๆ :

2.4 การแปลงสัญญาณแอนาลอก – ดิจิตอล

สัญญาณที่ใช้ในอุปกรณ์อิเลคทรอนิคส์มี 2 ชนิด คือ สัญญาณแอนาลอกและสัญญาณดิจิตอล สัญญาณแอนาลอกจะใช้ในอุปกรณ์ทั่วๆไปและใช้ในการควบคุมแบบเก่า

ในปัจจุบันมีไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์เข้ามาช่วยในการควบคุมอุปกรณ์ต่างๆมากมาย ซึ่งทำให้การควบคุมนั้นทำได้ง่ายและรวดเร็วยิ่งขึ้น แต่ในการควบคุมนั้นเราจำเป็นต้องใช้ สัญญาณดิจิตอลในการติดต่อกับไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์ แต่ในความเป็นจริงนั้น เราใช้สัญญาณแอนาลอกในการควบคุม ดังนั้นเราจึงจำเป็นต้องมีการเปลี่ยนสัญญาณแอนาลอกเป็นสัญญาณดิจิตอลแล้วจึงนำสัญญาณนั้นเข้ามาสู่ไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์เพื่อใช้ควบคุมระบบต่อไป

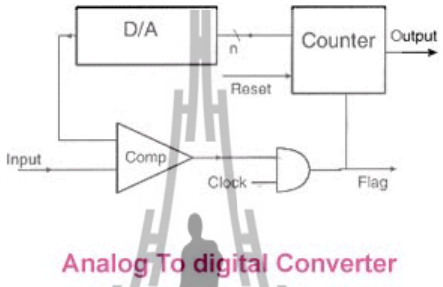
แม้ว่าสัญญาณแอนาลอกนั้นมีความแน่นอนและแม่นยำสูง แต่สัญญาณแอนาลอกนั้นก็ควบคุมได้ยาก เนื่องจากในสภาพแวดล้อมมีสัญญาณรบกวนอยู่มาก และการที่จะทำให้การควบคุมแบบแอนาลอกมีความสามารถควบคุมเท่ากับการควบคุมแบบดิจิตอลนั้นทำได้ยาก เนื่องจากวงจร

ควบคุมแบบแอนาลอกจะต้องมีความซับซ้อนสูง

อย่างไรก็ตาม สัญญาณดิจิตอลก็ไม่สามารถทดแทนความละเอียดของสัญญาณแอนาลอกได้อย่างสมบูรณ์แต่ทำให้การควบคุมนั้นทำให้ง่ายและสะดวกยิ่งขึ้น

2.4.1 Counting Converter

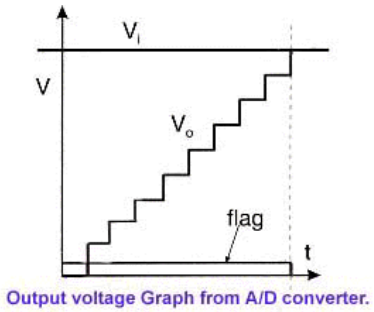
Counting Converter เป็นวิธีที่ง่ายที่สุดของการแปลงสัญญาณ แอนาลอกเป็นสัญญาณดิจิตอล โดยใช้อัลกอริทึมการนับค่าเพิ่มขึ้นเรื่อยๆแล้วนำผลที่ได้จากการนับไปเปรียบเทียบกับค่าที่ต้องการที่ตั้งไว้ ลักษณะการทำงานเป็นดังรูป



รูปที่ 2.4 การทำงานของ Counting Converter

จากวงจร Counter เป็นอุปกรณ์นับค่าที่เพิ่มขึ้นทีละหนึ่ง แล้วส่งค่าที่ได้ให้ D/A มีขา Reset รับสัญญาณ Reset เมื่อต้องการให้เริ่มนับใหม่ D/A เมื่อรับค่าที่นับเพิ่มขึ้นทีละหนึ่งจากตัวนับ ก็แปลงค่าให้เป็นสัญญาณ แอนาลอกที่มีค่าความต่างศักย์ค่าๆหนึ่ง แล้วส่งต่อเข้าไปที่อุปกรณ์ตัวเปรียบเทียบ Comparator จะเป็นอุปกรณ์ตัวเปรียบเทียบค่าความต่างศักย์ของอินพุตและค่าจากที่ตัวนับ ถ้าหากทั้งสองสัญญาณมีค่าเท่ากันส่งค่าความต่างศักย์ 0 โวลต์ออกมา(ลอจิก 0) ถ้าไม่เท่ากันก็จะส่งความต่างศักย์ที่ไม่ใช่ 0 โวลต์ออกมา (ลอจิก 1) ซึ่งค่าความต่างศักย์ที่ออกมาจะนำมาเข้าลอจิกเกต "และ" กับสัญญาณนาฬิกาจะได้ค่าลอจิกออกมา ถ้าผลลัพธ์ออกมาเป็นสัญญาณนาฬิกาแสดงว่ายังไม่ได้ผลลัพธ์เท่าที่ต้องการ สัญญาณนาฬิกาก็จะไปทำให้ตัวนับนับเพิ่มขึ้นต่อไปและเมื่อได้ค่าผลลัพธ์ดิจิตอลที่ต้องการแล้วค่าที่ได้จากตัวเปรียบเทียบจะให้ค่าความต่างศักย์เป็น 0 (ลอจิก 0) ซึ่งเมื่อนำมาเข้าลอจิกเกต "และ" กับสัญญาณนาฬิกาแล้วก็จะให้ลอจิก 0 ซึ่งทำให้ตัวนับไม่นับเพิ่มอีก ก็จะได้ค่าดิจิตอลจากตัวนับที่ต้องการ

จากคำอธิบายข้างต้นจะได้กราฟของ V0 ดังนี้

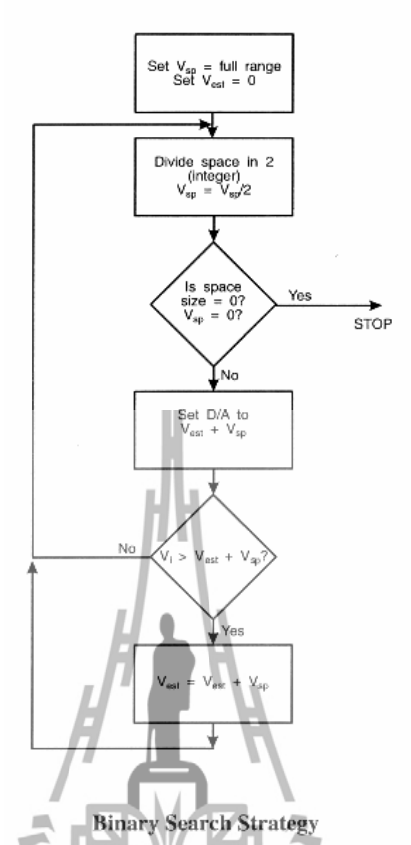


รูปที่ 2.5 ค่าความต่างศักย์ของเอาท์พุต

ข้อเสียของวิธีนี้ คือ การนับต้องเริ่มนับที่ 0 เสมอและนับเพิ่มขึ้นเรื่อยๆทำให้ช้า เอาท์พุตที่ได้จะมี delay จึงไม่ค่อยนิยมใช้เท่าที่ควร จึงได้เปลี่ยนตัวนับเป็นแบบนับลงได้ด้วยซึ่งจะอ้างอิงระดับจากระดับเก่าทำให้ไม่จำเป็นต้องนับ 0 ใหม่ เมื่อมีการเปลี่ยนอินพุตใหม่ แต่ให้อ้างอิงกับผลลัพธ์เดิมทำให้ได้ผลลัพธ์เร็วขึ้น

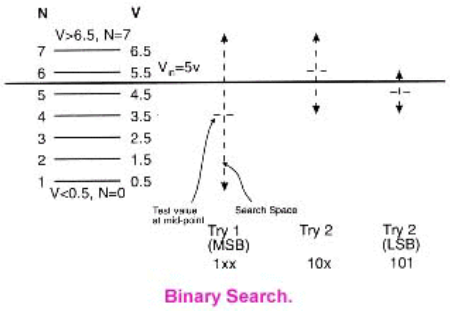
2.4.2 Successive Approximation

ใช้หลักการของ "binary search" ในการหาคำตอบ โดยนำค่าผลลัพธ์มาเปรียบเทียบกับค่ากึ่งกลางของช่วง เพื่อให้ทราบว่าค่านั้นๆมากกว่าหรือน้อยกว่า โดยจะปรับช่วงให้แคบลงมาเรื่อยๆแล้วเปรียบเทียบผลลัพธ์กับค่ากึ่งกลางของช่วงไปเรื่อยๆจนได้ผลลัพธ์ที่ต้องการ เช่น เลขที่เป็นคำตอบคือ 3 จากช่วงของคำตอบที่ 0-7 ครั้งแรกเอาค่า (0+7)/2 = 4 มาเปรียบเทียบได้ผลว่าคำตอบที่ต้องการอยู่ในช่วงที่น้อยกว่า 4 ครั้งที่ 2 ก็เลือกค่า (0+4)/2 = 2 มาเปรียบเทียบได้ผลว่าคำตอบที่ต้องการอยู่ในช่วงที่มากกว่า 2 แต่น้อยกว่า 4 ครั้งที่ 3 ก็เลือกค่า (2+4)/2 = 3 มาเปรียบเทียบได้ผลว่าคำตอบที่ต้องการ จากหลักการที่กล่าวมาอาจเขียน flow chart ได้ดังนี้



รูปที่ 2.6 แผนภูมิการทำงานของ Successive Approximation

ข้อดีของวิธีนี้คือ เวลาที่ใช้ในการหาคำตอบ n รอบ แน่นอน(สำหรับ n bit converter ซึ่งอ้างอิงได้ 2n ระดับ และระดับ Vin ที่คงที่) ซึ่งใช้เวลาน้อยกว่าแบบ " Counting Algorithm" แต่มีข้อเสีย คือ ถ้า Vin เปลี่ยนทันทีทันใด ขณะที่กำลังทำ binary search อยู่นั้น คำตอบที่ได้จะผิดพลาด ตัวอย่างเช่น เปลี่ยน Vin จาก 5 Volt เป็น 2 Volt



รูปที่ 2.7 รูปการค้นหาไบนารี่

ช่วงของ Vin คือ 1-7 ใช้ n=3 (เพราะว่า 23=8)

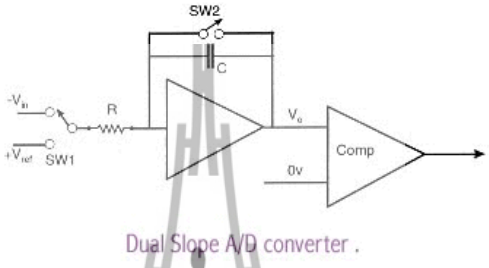
ครั้งแรก ใช้ 4 เปรียบเทียบกับ Vin (ซึ่งเท่ากับ 5 โวลต์) พบว่า อยู่ในช่วง lower ได้ 1xx

ครั้งที่ 2 ใช้ 2 เปรียบเทียบกับ Vin (ซึ่งเท่ากับ 5 โวลต์) พบว่า อยู่ในช่วง upper ได้ 10x

ครั้งที่ 3 ใช้ 3 เปรียบเทียบกับ Vin (ซึ่งเท่ากับ 5 โวลต์) พบว่า ผลลัพธ์ที่ได้จะผิดพลาด ได้ 100

2.4.3 Dual-Slope ADC

ใช้หลักการของวงจร Integrater ทำงานร่วมกับตัว Comparater ดังรูป



รูปที่ 2.8 รูปหลักการของ Dual-Slope ADC

Input Voltage มี 2 ตัว คือ ค่าความต่างศักย์ แอนาลอกที่ต้องการแปลงเป็นดิจิตอล ($-V_{in}$) และความต่างศักย์ที่คงที่ค่าหนึ่ง (V_{ref}) และมีสวิตช์ SW1 ซึ่งทำหน้าที่เลือกค่าสัญญาณ

จากวงจรตอนเริ่มต้นสวิตช์ SW2 ทำหน้าที่คายประจุของตัวเก็บประจุ C แล้วจึงเปิด SW2 ออก เมื่อสวิตช์ SW1 สับมาที่ $-V_{in}$ จากวงจร Intergrater จะพิสูจน์สมการได้ดังนี้

$$I = C \frac{dV_0}{dt}$$

$$-V_{in} + iR - V_0 + V_0 = 0$$

$$-Vin + RC \frac{dV_0}{dt} = 0$$

$$V_{in} = RC \frac{dV_0}{dt}$$

$$\int dV_0 = \int \frac{V_{in}}{RC} dt$$

$$V_0 = \frac{V_{in}(t)}{RC}$$

slope มีค่าเท่ากับ

$$\frac{V_{in}}{RC}$$

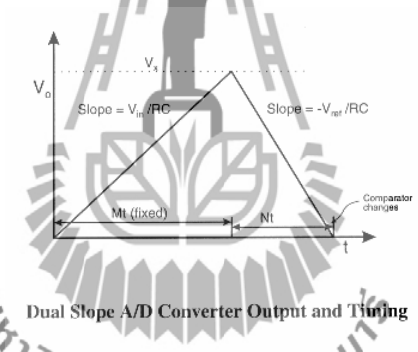
ค่า t ที่ใช้มีค่าคงที่ t_m

เมื่อ t เพิ่มจากศูนย์ถึง t_m ให้SW1 สับไปที่ V_{ref}

จะได้สมการ $V_0 = \frac{V_{ref}}{RC}(t)$

slope มีค่า $\frac{V_{ref}}{RC}$

สมมติ ช่วงเวลาตั้งแต่ความต่างศักย์ที่ t_m จนความต่างศักย์เป็น 0 มีค่าเท่ากับ t_n ได้ดังแสดงในกราฟ



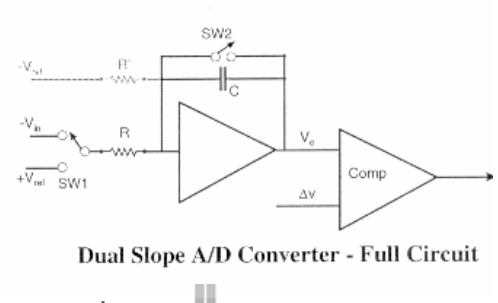
รูปที่ 2.9 กราฟแสดงความต่างศักย์กับเวลาของ Output and Timing

จากหลักของสามเหลี่ยมคล้าย จะได้สมการ $V_{in} = V_{ref}\frac{t_n}{t_m}$

เนื่องจาก V_{ref} และ t_n มีค่าคงที่ สัญญาณแอนาลอกขึ้นกับค่า t_n เพราะการควบคุมการเปลี่ยนสัญญาณดิจิตอล ที่ขึ้นกับค่า t_n การแปลงเป็นสัญญาณดิจิตอลจะทำโดยจับคู่ค่า t_n กับเอาต์พุตค่าๆ หนึ่ง ตามความเหมาะสมสำหรับ V_{ref} นั้นๆ เหมือนการเทียบค่าในตาราง ความเร็วของการแปลงสัญญาณแบบนี้ ขึ้นอยู่กับ V_{in} และ Slope ของวงจร integrater

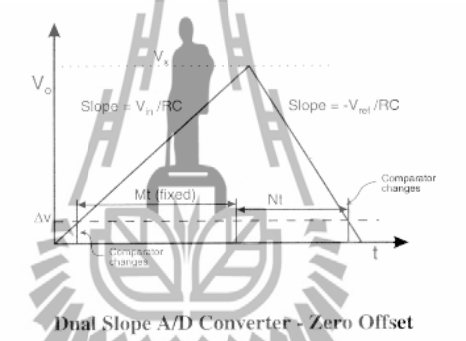
โดยธรรมชาติแล้วลักษณะของตัวเปรียบเทียบเองนั้น จะไม่เป็นอุดมคติคือจะมีผลต่างของความต่างศักย์อยู่ V โวลต์ แม้ว่าจะต่ออินพุตทั้งสองลงกราวด์แล้วก็ตาม ซึ่งถ้า V_{ref} ที่ใช้อยู่นั้นมีค่า

น้อยกว่าค่าผลต่างของความต่างศักย์ที่เกิดจากตัวเปรียบเทียบ ความชันก็จะน้อย ทำให้เวลา t_m ใช้เวลานานมาก กว่าที่จะพ้นค่าความต่างศักย์ที่เกิดจากตัวเปรียบเทียบ เราจึงต้องนำค่าความต่างศักย์มาเพิ่มให้กับ V_{ref} เพื่อหาผลลัพธ์ ดังรูป



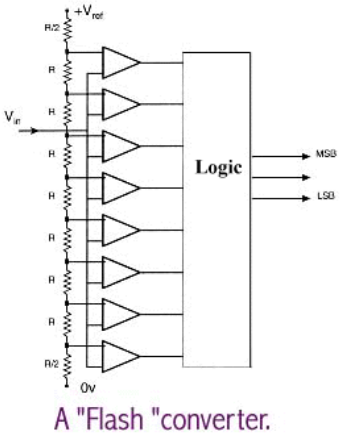
รูปที่ 2.10 รูปวงจร A/D Converter

จากวงจรดังกล่าวทำให้ได้กราฟดังรูป



รูปที่ 2.11 กราฟแสดงความต่างศักย์กับเวลาของ Zero Offset

2.4.4 Flash Converter

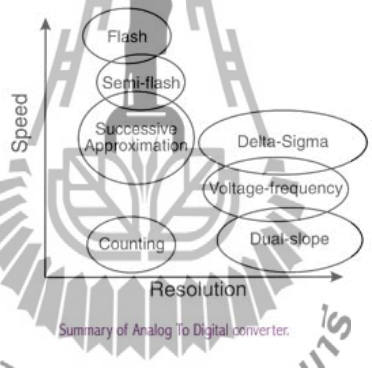


รูปที่ 2.12 รูปการทำงานของ Flash Converter

หลักการของ Flash Converter คือการใช้การแบ่งแรงดันเป็น Voltage หลายๆค่า แล้วเปรียบเทียบกับ Vin เป็นคู่ๆพร้อมกันแล้วกระทำการทาง logic จากรูปมี Voltage เปรียบเทียบ 8bit

ค่าความต่างศักย์จะเพิ่มขึ้นเรื่อยๆจากค่าความต้านทานที่ต่อเพิ่มขึ้น ความต่างศักย์ที่ได้นั้น เมื่อนำไปเปรียบเทียบกับ Vin แล้วมากกว่าก็จะปล่อยลอจิกออกมา ถ้ามากกว่าก็จะให้ลอจิก 1 ถ้าน้อยกว่าหรือเท่ากันก็จะให้ลอจิก 0 ซึ่งวิธี Flash Converter นี้จะเร็วที่สุด แต่ใช้อุปกรณ์ทาง Hardware มากกว่าแบบอื่นๆ

การแปลงสัญญาณแอนาลอก เป็นสัญญาณดิจิตอล มีประโยชน์มากในการควบคุมอุปกรณ์สวิตชิ่ง ซึ่งมีลักษณะการแปลงสัญญาณได้หลายวิธี แต่ละวิธีจะมีอัลกอริทึม ความรวดเร็วในการทำงาน และการใช้อุปกรณ์ฮาร์ดแวร์ต่างกันด้วย ทำให้ขนาด และราคาต่างกัน ขึ้นกับความต้องการของผู้ใช้ที่จะต้องเลือกให้เหมาะสมกับงานที่ใช้ และงบประมาณที่มีอยู่
ลำดับของความเร็ว และความละเอียดของอัลกอริทึมต่างๆ เป็นดังรูป



รูปที่ 2.13 รูปลำดับของความเร็ว และความละเอียดของอัลกอริทึมต่างๆ

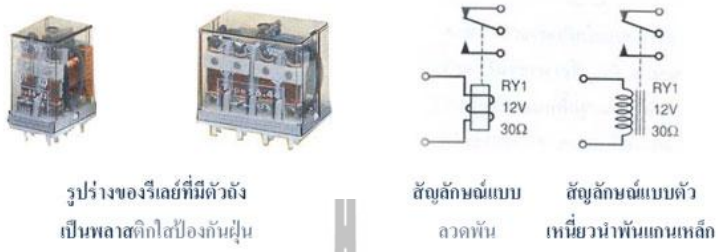
2.5 วัสดุอุปกรณ์ที่ใช้

2.5.1 รีเลย์

รีเลย์ (Relay) เป็นอุปกรณ์ที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานแม่เหล็ก เพื่อใช้ในการดึงดูดหน้าสัมผัสของคอนแท คให้เปลี่ยนสภาวะ โดยการป้อนกระแสไฟฟ้าให้กับขดลวด เพื่อทำการปิดหรือเปิดหน้าสัมผัสคล้ายกับสวิตช์อิเล็กทรอนิกส์ ซึ่งเราสามารถนำรีเลย์ไปประยุกต์ใช้ในการควบคุมวงจรต่าง ๆ ในงานช่างอิเล็กทรอนิกส์มากมาย

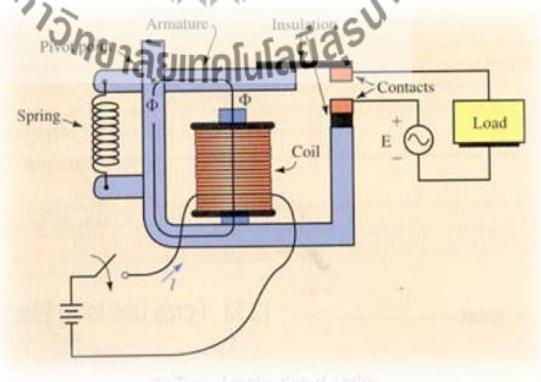
2.5.1.1 หลักการเบื้องต้น

รีเลย์เป็นอุปกรณ์ที่นิยมนำมาทำเป็นสวิตช์ทางด้านอิเล็กทรอนิกส์ โดยจะต้องป้อนกระแสไฟฟ้าให้ไหลผ่านขดลวดจำนวนหนึ่ง เพื่อนำไปควบคุมวงจรกำลังงานสูง ๆ ที่ต่ออยู่กับหน้าสัมผัสหรือคอนแทคของรีเลย์ รูปที่ 2.14



รูปที่ 2.14 รูปร่างและสัญลักษณ์ของรีเลย์

หลักการทำงานเบื้องต้นของรีเลย์แสดงดังรูปที่ 2.15 การทำงานเริ่มจากปิดสวิตช์ เพื่อป้อนกระแสให้กับขดลวด (Coil) โดยทั่วไปจะเป็นขดลวดพันรอบแกนเหล็ก ทำให้เกิดสนามแม่เหล็กไปดูดเหล็กอ่อนที่เรียกว่าอาร์เมเจอร์ (Armature) ให้ต่ำลงมา ที่ปลายของอาร์เมเจอร์ด้านหนึ่งมักยึดติดกับสปริง (Spring) และปลายอีกด้านหนึ่งยึดติดกับหน้าสัมผัส (Contacts) การเคลื่อนที่อาร์เมเจอร์จึงเป็นการควบคุมการเคลื่อนที่ของหน้าสัมผัส ให้แยกจากหรือแตะกับหน้าสัมผัสอีกอันหนึ่งซึ่งยึดติดอยู่กับที่ เมื่อเปิดสวิตช์อาร์เมเจอร์ ก็จะกลับสู่ตำแหน่งเดิม เราสามารถนำหลักการนี้ไปควบคุมโหลด (Load) หรือวงจรอิเล็กทรอนิกส์ต่าง ๆ ได้ตามต้องการ



รูปที่ 2.15 หลักการทำงานเบื้องต้นของรีเลย์

2.5.1.2 ชนิดของรีเลย์

รีเลย์ที่ผลิตในปัจจุบันมีอยู่มากมายหลายชนิด ผู้เรียบเรียงจะขอแนะนำรีเลย์ที่นิยมใช้งานและรู้จักกันแพร่หลาย 4 ชนิดเพื่อเป็นแนวทางในการศึกษา ในระดับสูงต่อไป

-อาร์เมเจอร์รีเลย์ (Armature Relay)

อาร์เมเจอร์ (Armature Relay) คือ รีเลย์ที่ได้อธิบายหลักการทำงานดังในรูปที่ 2.16 ซึ่งเป็นรีเลย์ที่นิยมใช้กันมากที่สุด บางครั้งเรียกรีเลย์แบบนี้ว่า รีเลย์ชนิดแคลปเปอร์ (Clapper Relay)

รูปที่ 2.16 รีเลย์ชนิดอาร์เมเจอร์

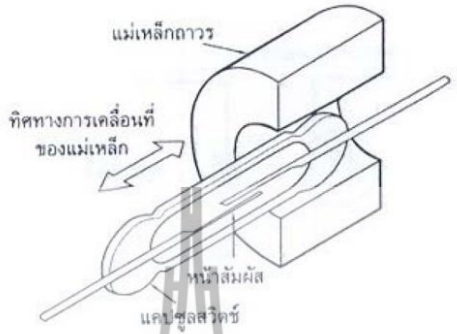
-รีดรีเลย์ (Reed Relay)

รีดรีเลย์ (Reed Relay) เป็นรีเลย์ไฟฟ้าที่มีลักษณะเป็นแคปซูลขนาดเล็ก ในรูปที่ 2.17 แสดงภาพตัดขวางของรีเลย์ ที่ประกอบด้วยส่วนที่เรียกว่ารีดแคปซูล ซึ่งมีคอยล์พันบนแกนบ๊อบบิ้น รีดแคปซูลจะเป็นหลอดแก้ว ภายในบรรจุก๊าซเฉื่อย หน้าสัมผัสเป็นโลหะผสมแผ่นบาง ๆ ปลายตัด 2แผ่น วางซ้อนแต่ไม่สัมผัสกัน เป็นสวิตช์ชุดเดียวทางเดียวหน้าสัมผัสปกติเปิดวงจร (SPST-NO)

รูปที่ 2.17 รีเลย์ชนิดรีดรีเลย์

-รีดสวิตซ์ (Reed Switch)

รีดสวิตช์ (Reed Switch) เป็นรีเลย์อีกชนิดหนึ่งแต่ไม่มีชุดขดลวดสำหรับสร้างสนามแม่เหล็ก การควบคุมการปิดเปิดหน้าสัมผัส ของสวิตช์จะใช้สนามแม่เหล็กจากภายนอกมาควบคุม หน้าสัมผัส โครงสร้างภายในของรีดสวิตช์แสดงดังรูปที่ 2.18



รูปที่ 2.18 รีเลย์ชนิดรีดสวิตซ์

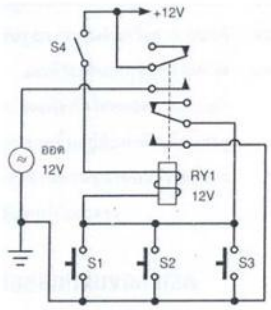
-โซลิดสเตตรีเลย์ (Solid-State Relay)

โซลิดสเตตรีเลย์ (Solid-State Relay) เป็นรีเลย์ที่ไม่มีโครงสร้างทางกลอยู่ภายในมีขั้วต่ออย่างละ 2 ขั้ว ขั้วอินพุต เป็นขั้วสำหรับป้อนสัญญาณควบคุม เพื่อบังคับให้ขั้วเอาต์พุตปิดหรือเปิดวงจร โดยจะมีการแยกกันทางไฟฟ้าระหว่างขั้วอินพุตและเอาต์พุต

รูปที่ 2.19 โซลิดสเตตรีเลย์

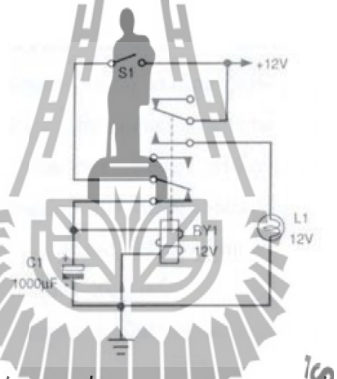
2.5.1.3 การประยุกต์ใช้งานรีเลย์

ปัจจุบันได้มีการนำรีเลย์ไปใช้ในการทำเป็นสวิตช์ทางด้านอิเล็กทรอนิกส์ในวงจรต่าง ๆ มากมาย ผู้เรียบเรียงจะขอยกตัวอย่าง รายละเอียดและรูปวงจรที่พอเป็นแนวทางในการศึกษาค้นคว้าต่อไปดังนี้



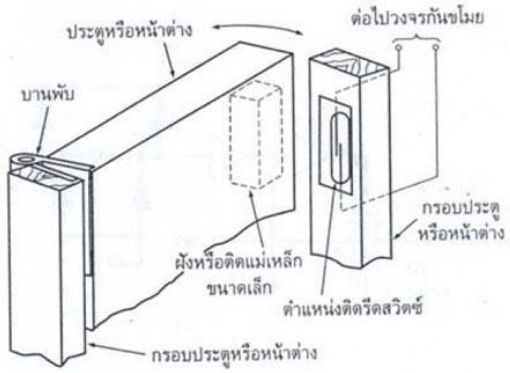
รูปที่ 2.20 การนำรีเลย์ไปต่อเป็นสวิตซ์ในวงจรกันขโมย

รูปที่ 2.20 เป็นการนำรีเลย์ที่มีหน้าสัมผัส 2 ชุดมาต่อเป็นวงจรกันขโมย โดยที่หน้าสัมผัสของสวิตช์ใช้แบบปกติเปิดวงจร (NO) เมื่อมีการกดสวิตช์ S1 , S2 และ S3 ตัวใดตัวหนึ่งจะทำให้ออดส่งเสียงเตือนค้าง โดยมีสวิตช์ S4 ทำหน้าที่รีเซตวงจร



รูปที่ 2.21 การนำรีเลย์มาต่อเป็นวงจรออสซิเลเตอร์เพื่อทำเป็นไฟกระพริบ

รูปที่ 2.21 แสดงการนำรีเลย์มาต่อเป็นวงจรออสซิลเลเตอร์เพื่อทำเป็นไฟกระพริบ ภายในวงจรใช้รีเลย์ขนาด 12 โวลต์ โดยที่หน้าสัมผัสจะจากกันเมื่อแรงดันต่ำกว่า 5 โวลต์ การทำงานของ วงจรเริ่มจากการกดสวิตช์ S1 จะทำให้มีกระแสไหลครบวงจรผ่านขดลวดของรีเลย์ พร้อมทั้งชาร์จไฟเข้าที่ตัวเก็บประจุ C1 ซึ่งจะทำการประจุกระแส จนกระทั่งแรงดันตกคร่อมขดลวดของรีเลย์ RY1 ทำงาน ทำให้หน้าสัมผัสแบบ NC เปิดวงจรออก ตัวเก็บประจุ C1 หยุดการชาร์จ ในขณะเดียวกันก็จะทำให้หน้าสัมผัสซึ่งเป็นแบบ NO ปิดวงจรส่งผลให้หลอดไฟ L1 สว่าง ขณะนี้ตัวเก็บประจุ C1 เริ่มคายประจุให้กับขดลวดแทน มีผลทำให้รีเลย์คงสภาวะการทำงานค้างไว้ จนกระทั่งแรงดันที่คายออกจาก C1 ค่อย ๆ ลดลงจนถึงค่าที่ทำให้ขดลวดไม่สามารถดูดหน้าสัมผัสให้อยู่ได้ จึงทำให้รีเลย์กลับสู่สภาวะเริ่มต้นหรือสภาวะปกติอีกครั้ง ทำให้หน้าสัมผัสของรีเลย์เปิดปิดสลับกันไปตลอดทำให้ไฟติดและดับสลับกัน



รูปที่ 2.22 การนำรีดสวิตช์ไปใช้ในวงจรกันขโมย

รูปที่ 2.22 แสดงการนำรีดสวิตช์ไปใช้ในวงจรกันขโมย โดยฝังสวิตช์ไว้ในกรอบประตูและฝังแม่เหล็กในบานประตู ขณะที่มีการเปิดประตูจะทำให้หน้าสัมผัสของรีดสวิตช์เปิดออกตามลักษณะการเปิดปิดประตู ถ้ามีขโมยเข้ามาก็จะทราบได้ทันที

2.5.2 มอเตอร์

มอเตอร์ไฟฟ้า (Motor) หมายถึง เป็นเครื่องกลไฟฟ้าชนิดหนึ่งที่เปลี่ยนแปลงพลังงานไฟฟ้ามาเป็นพลังงานกล มอเตอร์ไฟฟ้าที่ใช้พลังงานไฟฟ้าเปลี่ยนเป็นพลังงานกลมีทั้งพลังงานไฟฟ้ากระแสสลับและพลังงานไฟฟ้ากระแสตรง

2.5.2.1 หลักการเบื้องต้น

มอเตอร์ไฟฟ้าเป็นอุปกรณ์ที่นิยมใช้กันอย่างแพร่หลายในโรงงานต่างเป็นอุปกรณ์ที่ใช้ควบคุมเครื่องจักรกลต่างๆในงานอุตสาหกรรมมอเตอร์มีหลายแบบหลายชนิดที่ใช้ให้เหมาะสมกับงานดังนั้นเราจึงต้องทราบถึงความหมายและชนิดของมอเตอร์ไฟฟ้าตลอดคุณสมบัติการใช้งานของมอเตอร์แต่ละชนิดเพื่อให้เกิดประสิทธิภาพสูงสุดในการใช้งานของมอเตอร์นั้นๆ และสามารถเลือกใช้งานให้เหมาะสมกับงานออกแบบระบบประปาหมู่บ้านหรืองานอื่นที่เกี่ยวข้องได้

2.5.2.2 ชนิดของมอเตอร์ไฟฟ้า

มอเตอร์ไฟฟ้าแบ่งออกตามการใช้ของกระแสไฟฟ้าได้ 2 ชนิดดังนี้

1. มอเตอร์ไฟฟ้ากระแสสลับ (Alternating Current Motor) หรือเรียกว่าเอ.ซี มอเตอร์ (A.C. MOTOR) การแบ่งชนิดของมอเตอร์ไฟฟ้าสลับแบ่งออกเป็น3 ชนิดได้แก่

1.1 มอเตอร์ไฟฟ้ากระแสสลับชนิด 1 เฟส หรือเรียกว่าซิงเกิลเฟสมอเตอร์ (A.C. Sing Phase)

- สปลิทเฟส มอเตอร์(Split-Phase motor)
- คาปาซิเตอร์ มอเตอร์ (Capacitor motor)
- รีพัลชั่นมอเตอร์ (Repulsion-type motor)
- ยูนิเวอร์แซลมอเตอร์ (Universal motor)
- เช็ดเดดโพล มอเตอร์ (Shaded-pole motor)

1.2 มอเตอร์ไฟฟ้าสลับชนิด 2 เฟสหรือเรียกว่าทูเฟสมอเตอร์ (A.C.Two phas Motor)

1.3 มอเตอร์ไฟฟ้ากระแสสลับชนิด 3 เฟสหรือเรียกว่าทีเฟสมอเตอร์ (A.C. Three phase Motor)

2. มอเตอร์ไฟฟ้ากระแสตรง (Direct Current Motor) หรือเรียกว่าดี.ซี มอเตอร์ (D.C. MOTOR) การแบ่งชนิดของมอเตอร์ไฟฟ้ากระแสตรงแบ่งออกเป็น 3 ชนิดได้แก่

2.1 มอเตอร์แบบอนุกรมหรือเรียกว่าซีรีส์มอเตอร์ (Series Motor)

2.2 มอเตอร์แบบอนุขนานหรือเรียกว่าชันท์มอเตอร์ (Shunt Motor)

2.3 มอเตอร์ไฟฟ้าแบบผสมหรือเรียกว่าคอมเปาวด์มอเตอร์ (Compound Motor)

มอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรง เป็นต้นกำลังขับเคลื่อนที่สำคัญอย่างหนึ่งในโรงงานอุตสาหกรรมเพราะมีคุณสมบัติที่ดีเด่นในด้านการปรับความเร็วได้ตั้งแต่ความเร็วต่ำสุดจนถึงสูงสุด นิยมใช้กันมากในโรงงานอุตสาหกรรม เช่น โรงงานทอผ้า โรงงานเส้นใยโพลีเอสเตอร์ โรงงานถลุงโลหะหรือให้ เป็นต้นกำลังในการขับเคลื่อนรถไฟฟ้า เป็นต้นในการศึกษาเกี่ยวกับมอเตอร์ไฟฟ้ากระแสตรงจึงควรรู้จัก อุปกรณ์ต่าง ๆ ของมอเตอร์ไฟฟ้ากระแสตรงและเข้าใจถึงหลักการทำงานของมอเตอร์ไฟฟ้ากระแสตรงแบบต่าง ๆ

ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรงที่ส่วนประกอบที่สำคัญ 2 ส่วนดังนี้

1. ส่วนที่อยู่กับที่หรือที่เรียกว่าสเตเตอร์ (Stator) ประกอบด้วย

1.1 เฟรมหรือโยค (Frame Or Yoke) เป็นโครงภายนอกทำหน้าที่เป็นทางเดินของเส้นแรงแม่เหล็กจากขั้วเหนือไปขั้วใต้ให้ครบวงจรและยึดส่วนประกอบอื่นๆให้แข็งแรงทำด้วยเหล็กหล่อหรือเหล็กแผ่นหนาม้วนเป็นรูปทรงกระบอก

ขั้วแม่เหล็ก (Pole) ประกอบด้วย 2 ส่วนคือแกนขั้วแม่เหล็กและขดลวด

- ส่วนแรกแกนขั้ว (Pole Core) ทำด้วยแผ่นเหล็กบางๆ กั้นด้วยฉนวนประกอบกันเป็นแท่งยึดติดกับเฟรม ส่วนปลายที่ทำเป็นรูปโค้งนั้นเพื่อโค้งรับรูปกลมของตัวโรเตอร์เรียกว่าขั้วแม่เหล็ก (Pole Shoes) มีวัตถุประสงค์ให้ขั้วแม่เหล็กและโรเตอร์ใกล้ชิดกันมากที่สุดเพื่อให้เกิดช่องอากาศน้อยที่สุด เพื่อให้เกิดช่องอากาศน้อยที่สุดจะมีผลให้เส้นแรงแม่เหล็กจากขั้วแม่เหล็กจาก

ขั้วแม่เหล็กผ่านไปยังโรเตอร์มากที่สุดแล้วทำให้เกิดแรงบิดหรือกำลังบิดของโรเตอร์มากเป็นการทำให้มอเตอร์มีกำลังหมุน (Torque)

- ส่วนที่สอง ขดลวดสนามแม่เหล็ก (Field Coil) จะพันอยู่รอบๆแกนขั้วแม่เหล็กขดลวดนี้ทำหน้าที่รับกระแสจากภายนอกเพื่อสร้างเส้นแรงแม่เหล็กให้เกิดขึ้น และเส้นแรงแม่เหล็กนี้จะเกิดการหักล้างและเสริมกันกับสนามแม่เหล็กของอาเมเจอร์ทำให้เกิดแรงบิดขึ้น

2. ตัวหมุน (Rotor) ตัวหมุนหรือเรียกว่าโรเตอร์ตัวหมุนนี้ทำให้เกิดกำลังงานมีแกนวางอยู่ในตลับลูกปืน (Ball Bearing) ซึ่งประกอบอยู่ในแผ่นปิดหัวท้าย (End Plate) ของมอเตอร์

ตัวโรเตอร์ประกอบด้วย 4 ส่วนด้วยกัน คือ

1. แกนเพลา (Shaft) เป็นตัวสำหรับยึดคอมมิวเตเตอร์ และยึดแกนเหล็กอาร์เมเจอร์ (Armature Core) ประกอบเป็นตัวโรเตอร์แกนเพลานี้จะวางอยู่บนแบริ่ง เพื่อบังคับให้หมุนอยู่ในแนวนิ่งไม่มีการสั่นสะเทือนได้

2. แกนเหล็กอาร์มาเจอร์ (Armature Core) ทำด้วยแผ่นเหล็กบางอาบฉนวน (Laminated Sheet Steel) เป็นที่สำหรับพันขดลวดอาร์มาเจอร์ซึ่งสร้างแรงบิด (Torque)

3. คอมมิวเตเตอร์ (Commutator) ทำด้วยทองแดงออกแบบเป็นซี่แต่ละซี่มีฉนวนไมก้า (mica) คั่นระหว่างซี่ของคอมมิวเตเตอร์ ส่วนหัวซี่ของคอมมิวเตเตอร์ จะมีร่องสำหรับใส่ปลายสาย ของขดลวดอาร์มาเจอร์ ตัวคอมมิวเตเตอร์นี้อัดแน่นติดกับแกนเพลา เป็นรูปกลมทรงกระบอก มีหน้าที่สัมผัสกับแปรงถ่าน (Carbon Brushes) เพื่อรับกระแสจากสายป้อนเข้าไปยังขดลวดอาร์เมเจอร์เพื่อสร้างเส้นแรงแม่เหล็กอีกส่วนหนึ่งให้เกิดการหักล้างและเสริมกันกับเส้นแรงแม่เหล็กอีกส่วน ซึ่งเกิดจากขดลวดขั้วแม่เหล็ก ดังกล่าวมาแล้วเรียกว่าปฏิกิริยามอเตอร์ (Motor action)

4. ขดลวดอาร์เมเจอร์ (Armature Winding) เป็นขดลวดพันอยู่ในร่องสลอ ต (Slot) ของแกนอาร์เมเจอร์ ขนาดของลวดจะเล็กหรือใหญ่และจำนวนรอบจะมากหรือน้อยนั้นขึ้นอยู่กับการออกแบบของตัวโรเตอร์ชนิดนั้นๆ เพื่อที่จะให้เหมาะสมกับงานต่างๆ ที่ต้องการ ควรศึกษาต่อไปในเรื่องการพันอาร์เมเจอร์ (Armature Winding) ในโอกาสต่อไป

ทำด้วยคาร์บอนมีรูปร่างเป็นแท่งสี่เหลี่ยมผืนผ้าในซองแปรงมีสปิงกดอยู่ด้านบนเพื่อให้ถ่านนี้สัมผัสกับซี่คอมมิวเตเตอร์ตลอดเวลาเพื่อรับกระแส และส่งกระแสไฟฟ้าระหว่างขดลวดอาร์ เมเจอร์กับวงจรไฟฟ้าจากภายนอก คือถ้าเป็นมอเตอร์กระแสไฟฟ้าตรงจะทำหน้าที่รับกระแสจากภายนอกเข้าไปยังคอมมิวเตเตอร์ให้ลวดอาร์เมเจอร์เกดแรงบิดทำให้มอเตอร์หมุนได้

หลักการของมอเตอร์กระแสไฟฟ้าตรง (Motor Action)

หลักการของมอเตอร์ไฟฟ้ากระแสตรง (Motor Action) เมื่อเป็นแรงดันกระแสไฟฟ้าตรงเข้าไปในมอเตอร์ ส่วนหนึ่งจะ แปรงถ่านผ่านคอมมิวเตเตอร์เข้าไปในขดลวดอาร์เมเจอร์สร้างสนามแม่เหล็กขึ้น และกระแสไฟฟ้าอีกส่วนหนึ่งจะไหลเข้าไปในขดลวดสนามแม่เหล็ก (Field coil) สร้างขั้วเหนือ-ใต้ขึ้น จะเกิดสนามแม่เหล็ก 2 สนาม ในขณะเดียวกัน ตามคุณสมบัติของเส้นแรง แม่เหล็ก จะไม่ตัดกันทิศทางตรงข้ามจะหักล้างกัน และทิศทางเดียวจะเสริมแรงกัน ทำให้เกิดแรงบิดในตัวอาร์เมเจอร์ ซึ่งวางแกนเพลาและแกนเพลานี้ สวมอยู่กับตลับลูกปืนของมอเตอร์ ทำให้อาร์เมเจอร์นี้หมุนได้ ขณะที่ตัวอาร์เมเจอร์ทำหน้าที่หมุนได้นี้เรียกว่า โรเตอร์ (Rotor) ซึ่งหมายความว่าตัวหมุน การที่อำนาจเส้นแรงแม่เหล็กทั้งสองมีปฏิกิริยาต่อกัน ทำให้ขดลวดอาร์เมเจอร์หรือโรเตอร์หมุนไปนั้นเป็นไปตามกฎซ้ายของเฟลมมิ่ง (Fleming'left hand rule)

สเต็ปมอเตอร์ (STEPPING MOTOR)

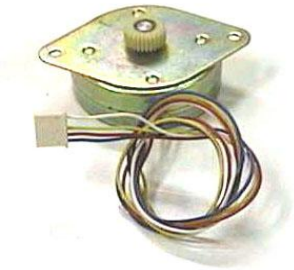
สเต็ปมอเตอร์ เป็นมอเตอร์ที่ขับเครื่องด้วยพัลส์ ลักษณะการขับเคลื่อน จะหมุนรอบแกนได้ 360 องศา มีลักษณะไม่ต่อเนื่อง แต่มีลักษณะเป็นสเต็ป โดยแต่ละสเต็ปจะขับเคลื่อนได้ 1,1.5,1.8 หรือ 2 องศา แล้วแต่ละโครงสร้างของมอเตอร์ลักษณะที่ นำมอเตอร์ไปใช้ จะเป็นงานที่ต้องการตำแหน่งแม่นยำ เช่น ระบบขับเคลื่อนหัวแม่พิมพ์ในเครื่องพิมพ์ (PRINTER)ระบบขับเคลื่อนหัวอ่านในเครื่องอ่านบันทึกเหล็ก ระบบขับเคลื่อนตำแหน่งของปากกาใน X-Y PLOTER เป็นต้น

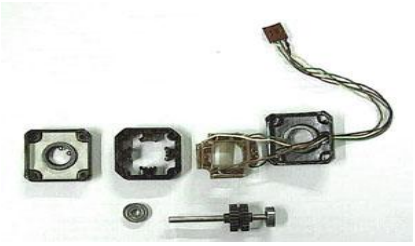
ชนิดและโครงสร้างของสเต็ปมอเตอร์

สเต็ปมอเตอร์มีลักษณะดังรูป

รูปที่ 2.23 สเต็ปมอเตอร์แบบมีสาย 5 เส้น

รูปที่ 2.24 มอเตอร์แบบมีสาย 6 เส้น





รูปที่ 2.25 สเต็ปมอเตอร์หลายแบบไบโพลาร์ **รูปที่ 2.26** ภาพถ่ายโครงสร้างสเต็ปมอเตอร์

สเต็ปมอเตอร์ที่พบในปัจจุบันมี 3 ลักษณะดังนี้

1. **แบบแม่เหล็กถาวร(PERMANENT MAGNET_PM)**

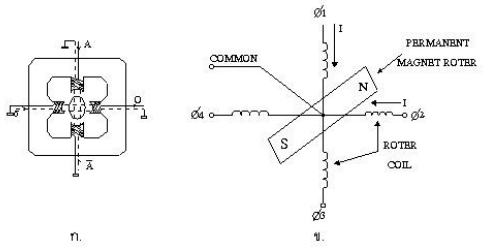
สเต็ปมอเตอร์แบบ PM จะมีสเตเตอร์ (STATOR) ที่พันขดลวดไว้หลายๆโพล โดยมีโรเตอร์ (ROTOR) เป็นรูปทรงกระบอกฟันเลื่อยและโรเตอร์ทำด้วยแม่เหล็กถาวร เพื่อป้อนไฟกระแสตรง ให้กับขดสเตเตอร์จะทำให้เกิดแรงแม่เหล็กไฟฟ้าผลักต่อโรเตอร์ ทำให้มอเตอร์หมุน มอเตอร์แบบ PM จะเกิดแรงฉุดยึดให้โรเตอร์หยุดอยู่กับที่ แม้จะไม่ได้ป้อนไฟเข้าขดลวด

2. **แบบแปรค่ารีลักแตนซ์ (VARIABLE RELUCTANCE- VR)**

สเต็ปมอเตอร์แบบ VR จะมีการหมุนโรเตอร์ได้อย่างอิสระ แม้จะไม่ได้จ่ายไฟให้โรเตอร์ทำจากสารเฟอร์โรแมกเนติก กำลังอ่อน มีลักษณะเป็นฟันเลื่อยรูปทรงกระบอกโดยจะมีความสัมพันธ์ โดยตรงกับจำนวนโพลในสเตเตอร์ แรงบิดที่เกิดขึ้นจะไปหมุนโรเตอร์ไปในเส้นทางของอำนาจแม่เหล็กที่มีค่ารีลักแตนท์ต่ำที่สุด ตำแหน่งที่จะเกิดแน่นอนและมีเสถียรภาพแต่จะเกิดขึ้นได้หลายๆ จุดดังนั้นเมื่อป้อนไฟเข้าขดลวดต่างๆ ในมอเตอร์แตกต่างขดกันไป ก็ทำให้มอเตอร์ หมุนไปตำแหน่งต่างๆ กันโรเตอร์ของ VR จะมีความเฉื่อยของโรเตอร์น้อยจึงมีความเร็วรอบสูงกว่ามอเตอร์แบบ PM

3. **แบบผสม(HYBRID-H)**

สเต็ปมอเตอร์แบบ H จะเป็นลูกผสมของ VR กับ PM โดยจะมีสเตเตอร์คล้ายกับที่ใช้ใน VR โรเตอร์มีหมวกหุ้มปลายซึ่งมีลักษณะของสารแม่เหล็กที่มีกำลังสูง โดยการควบคุมขนาดรูปร่างของหมวกแม่เหล็กอย่างดีทำให้ได้มุม การหมุนและครั้งน้อยและแม่นยำ ข้อดีก็คือ ให้แรงบิดสูง มีขนาดกะทัดรัด และให้แรงฉุดยึดโรเตอร์นิ่งกับที่ตอนไม่จ่ายไฟ



รูปที่ 2.27 (ก) โครงสร้าง (ข) วงจรเทียบเท่า (equivalentcircuit) ของมอเตอร์ ชนิด 4 ขด และมุมของโรเตอร์เทียบกับกระแสไฟฟ้าที่จ่ายแก่เฟสต่าง ๆ8 ตำแหน่ง

เฟสที่จ่ายกระแสไฟฟ้า	ϕ_1	$\phi_1\phi_2$	ϕ_2	$\phi_2\phi_3$	ϕ_3	$\phi_3\phi_4$	ϕ_4	$\phi_4\phi_1$
ตำแหน่งโรเตอร์	⇧	⇗	⇨	⇘	⇩	⇙	⇦	⇖

จากลักษณะของมุมโรเตอร์หมุนกับกระแสไฟที่ป้อนแก่เฟสต่างๆจะสามารถสั่งงานให้ STEPPING MOTOR หมุนได้ 3 อย่าง คือ

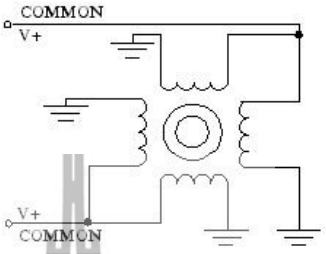
1. **แบบจ่ายกระแสไฟให้เฟสเดียวสวนเวียนกันไป** เรียก ONE-EXCITATION หรือ HALF DRIVE คือ f1 , f2 f3, f4 การ OUT EXCITATION แบบนี้แรงบิดจะน้อย
2. **แบบจ่ายกระแสไฟให้พร้อมกันทีเดียว 2 f** เรียก TWO-EXCTATION หรือFULL STEP คือ f1f2 , f2f3 , f3f4 , f4f1 หมุนเวียนกันไปแบบนี้แรงบิดจะมาก
3. **แบบจ่ายกระแสไฟให้ทีละ 1 เฟส สลับกับ 2 เฟส** เรียก ONE-TWO EXCITATION หรือ HALF STEP เหมือนรูปแสดงของมุมโรเตอร์ แต่แบบนี้จำนวน STEP ทวนเข็มจะเป็นตรงกันข้าม

เฟส	ϕ_4	ϕ_3	ϕ_2	ϕ_1
ϕ_1	1	0	0	1
ϕ_2	0	0	1	0
ϕ_3	0	1	0	0
ϕ_4	1	0	0	0

การตรวจสอบหาสาย COMMON และสาย GROUND ของ STEPPING แบบPM (แบบแกนโรเตอร์เป็นแม่เหล็กถาวร)

โดยทั่วไป SP MOTOR แบบ PM จะมีอยู่ 2 ชนิด

1. ชนิดที่เป็น COMMON ภายนอก SP MOTOR แบบนี้มีสายอยู่ 6 เส้น คือ

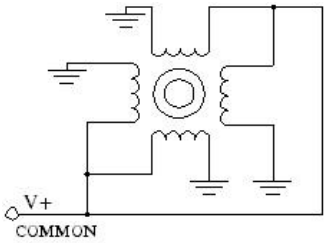


รูปที่ 2.28 สเต็ปมอเตอร์ ชนิดมีสาย 6 เส้น

- สายที่เป็น COMMON 2 เส้น
- สายที่เป็น GROUND 4 เส้น

สาย COMMON 1 เส้น จะ DRIVE GROUND 2 เส้น ในการเช็คให้ใช้มิเตอร์วัดหาสายที่เป็น COMMON ก่อนโดยการตั้ง RANGE ของมิเตอร์ที่ R*1 จับที่สายทีละคู่ ถ้าหากวัดสาย COMMON เทียบกับสาย GROUND ได้ถูกต้องค่าความต้านทานที่อ่านได้จะน้อย แต่ถ้าวัดผิดสาย คือวัดสาย GROUND เทียบกับ GROUNDค่าความ ต้านทานที่อ่านได้จะสูงกว่าแต่ถ้าวัดสาย COMMON เทียบกับสาย GROUND ที่ไม่ใช่คู่กันแล้ว เข็มมิเตอร์ก็จะไม่ กระดิก ให้ทดลองวัดเปรียบเทียบกันทีละคู่ ก็จะทราบว่าสายใดเป็นสาย COMMON สายใดเป็นสาย GROUND

2. ชนิดที่เป็น COMMON ภายใน SP MOTOR แบบนี้มีสายอยู่ 5 เส้น คือ



รูปที่ 2.29 สเต็ปมอเตอร์ชนิดมีสาย 5 เส้น

- สายที่เป็น COMMON 1 เส้น
- สายที่เป็น GROUND 4 เส้น

ในการวัดให้ทำแบบเดียวกับการวัด SP MOTOR ชนิด COMMON ภายนอกแตกต่างกันเพียงแบบ COMMON ภายในสาย COMMON 1 เส้นDRIVE สาย GROUND 4 เส้น ดังนั้นหากสายเส้นใดเมื่อวัดเทียบกับสายเส้นอื่น แล้วมีค่าความต้านทานน้อยที่สุดสายเส้นนั้นเป็นสาย COMMON และที่เหลืออีก4 เส้นจะเป็นสาย GROUND

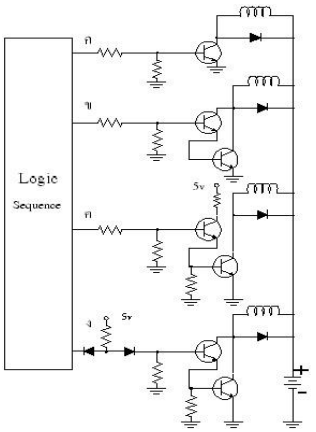
การเรียงเฟสของ STEPPING MOTOR แบบ PM

เมื่อเราทราบว่าสายเส้นใดเป็นสาย COMMON แล้วแต่เรายังไม่ทราบว่าสาย GROUND เส้นใดเป็นเฟสที่ 1 เฟสที่ 2 เฟสที่ 3 และเฟสที่ 4 ในการเรียงเฟสนั้นให้ใช้มิเตอร์วัดโดยนำ V+ เขาที่สาย COMMON วัดเทียบกับสาย GROUNDเส้นใดก็ได้ 1 เส้น จะทำให้แกนโรเตอร์เคลื่อนไปข้างหน้า 1 STEP เมื่อเปลี่ยนสาย GROUND เส้นแรกเป็นเส้นที่ 2 ลาก MOTOR ไม่เคลื่อนที่ไปข้างหน้าแสดงว่าการเรียงเฟสไม่ถูกต้องก็ให้วัดเทียบกับสาย GROUNDเส้นใหม่ต่อไป หาก MOTOR เคลื่อนที่ไปข้างหน้าตามก้น วัดที่สาย GROUND เส้นต่อไปเรื่อย ๆ ก็จะทำให้ทราบว่าสายเส้นใดเป็นเฟสแรก สายเส้นใดเป็นเฟสที่ 2 เฟสที่ 3 และเฟสที่ 4 การเรียงเฟสของ SP MOTOR แบบ PM ทั้งชนิดที่เป็น COMMON ภายนอกและชนิดที่เป็น COMMON ภายใน ใช้หลักการเดียวกัน

วงจรขับสเต็ปมอเตอร์

-วงจรขับ (DRIVE)

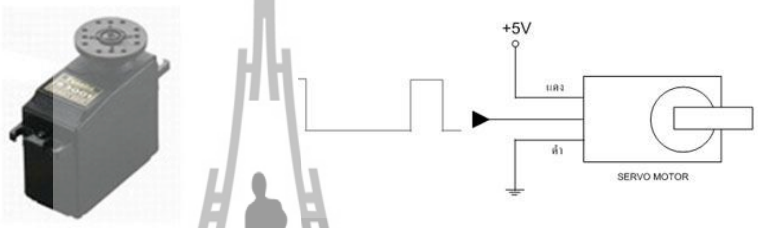
เมื่อเรารู้ซีเควนซ์ของมันแล้วต่อไปเราก็ต้องมีวงจร DRIVE ให้แก่ STEPPING MOTOR วิธีที่ง่ายที่สุดในการ ต่อวงจรซีเควนส์เข้ากับวงจรนั้น คือ การต่อโดยตรง ดังเอาท์พุท ก และ ข แต่ถ้ากระแสเอาท์พุท ของวงจรซีเควนส์ไม่ เพียงพอก็ต้องต่อ บัฟเฟอร์ (BUFFER) เพื่อขยายกระแสดังรูปเอาท์พุท ค และ ง



รูปที่ 2.30 รูปบัฟเฟอร์(BUFFER)

- **SERVO MOTOR**

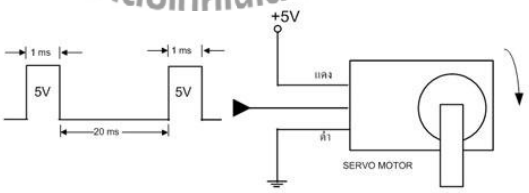
SERVO MOTOR คือ มอเตอร์ไฟฟ้ากระแสตรง DC MOTOR ที่ถูกประกอบร่วมด้วยชุดเกียร์ และส่วนควบคุมต่างๆ ไว้ในโมดูลเดียวกัน โดยจะมีสัญญาณใช้งาน 1 เส้น และอีก2 เส้น เป็น VCC และ GND เท่านั้น ซึ่งสามารถควบคุมให้ตัว SERVO MOTOR หมุนซ้าย หรือ ขวาได้ +90 องศา - 90 องศา (180 องศา) โดยสามารถสั่งงานในการหมุนให้หมุนไปได้ตามองศาต่างๆ ที่ต้องการ ได้ด้วยตัวของ SERVO MOTOR เอง เช่น ต้องการหมุน 1 องศา หรือ 15 องศา ก็ได้ ไม่ต้องมีส่วนควบคุม หรือ SENSOR ใดๆ กลับมาตรวจสอบอีกทำให้ง่าย และสะดวกในการในการนำไปประยุกต์ใช้งานต่างๆ ได้จริง



รูปที่ 2.31 การควบคุมทำงานของเซอร์โวมอเตอร์

การต่อใช้งานเซอร์โวมอเตอร์

ในการควบคุมเซอร์โวมอเตอร์นั้น ทำได้โดยอาศัยความกว้างของพัลส์ที่ทำการป้อนให้เซอร์โวมอเตอร์ โดยสัญญาณพัลส์นี้จะเป็นสัญญาณ TTL จะมีแรงดัน 5VDC และ แรงดัน 0 VDC



รูปที่ 2.32 ตัวอย่างเซอร์โวมอเตอร์หมุนไปทางขวา

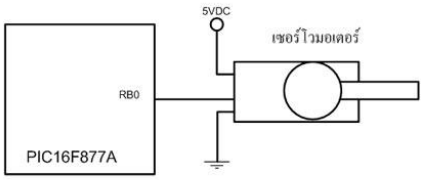
เซอร์โวมอเตอร์หนุนไปทางขวา หรือ ตามเข็มนาฬิกา จะต้องสร้างสัญญาณพัลส์ ความกว้างขนาด 1 ms และมีช่วงระห่างระหว่างพัลส์ 20 ms

เซอร์โวมอเตอร์หนุนไปทางซ้าย หรือ ทวนเข็มนาฬิกา จะต้องสร้างสัญญาณพัลส์ ขนาด 2ms และมีช่วงระห่างระหว่างพัลส์ 20

เซอร์โวมอเตอร์อยู่ในตำแหน่งกึ่งกลาง จะต้องสร้างสัญญาณพัลส์ ขนาด 1.5 ms และมีช่วง

ระห่างระหว่างพัลส์ 20

เซอร์โวมอเตอร์แต่ละตัวจะมีการทำงานของความกว้างของพัลส์ไม่เท่ากัน ค่าดังกล่าวเป็นค่าประมาณเท่านั้น ดังนั้นถ้าต้องการค่าที่ถูกต้อง ต้องดูที่คู่มือด้วยทุกครั้ง และจำนวนลูกสัญญาณพัลส์จะมีผลต่อองศาในการหมุนด้วยเช่นเดียวกัน



รูปที่2.33 รูปเซอร์โวมอเตอร์

การประยุกต์ใช้งาน

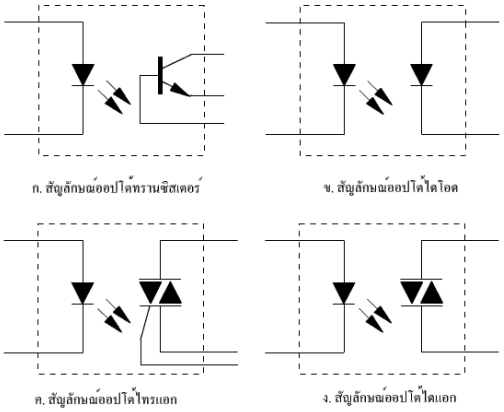
เราสามารถนำเอาเซอร์โวมอเตอร์ไปบังคับควบคุมทิศทาง หรือ ตำแหน่งองศาต่างๆได้ โดยดูแนวทางจากคู่มือของผู้ผลิต

2.5.3 อุปกรณ์เชื่อมต่อทางแสง (Opto-Isolator)

อุปกรณ์เชื่อมต่อทางแสง (Opto-Isolator) หรือที่เรียกว่าออปโต้คัปเปลอร์ (Opto-Coupler) เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในการเชื่อมต่อทางแสงโดยใช้หลักการเปลี่ยนสัญญาณไฟฟ้าเป็นสัญญาณแสงและเปลี่ยนกลับจากแสงเป็นไฟฟ้าตามเดิม ใช้สำหรับการเชื่อมต่อสัญญาณระหว่างสองวงจรที่ต้องการแยกทางไฟฟ้าอย่างเด็ดขาดเพื่อป้องกันการรบกวนกันทางไฟฟ้า แบ่งออกเป็นหลายชนิดแต่ละชนิดจะประกอบด้วย LED ส่งแสงซึ่งปกติจะเป็นชนิดอินฟาเรดและตัวรับแสงที่เป็นโฟโต้ทรานซิสเตอร์หรือโฟโต้ไดโอด โดยจะถูกผลิตรวมอยู่ในตัวเดียวกัน

1. โครงสร้างสัญลักษณ์อุปกรณ์เชื่อมต่อทางแสง

โครงสร้างสัญลักษณ์อุปกรณ์เชื่อมต่อทางแสงจะเหมือนกับอุปกรณ์ประเภท โฟโต้ แต่จะเพิ่มอุปกรณ์ส่งแสงอิ นฟาเรด คือไดโอดเปล่งแสงอินฟาเรดเข้าไปอีกหนึ่งตัว เช่นโฟโต้ทรานซิสเตอร์จะเพิ่มไดโอดเปล่งแสงอินฟาเรดเข้าไปอีกหนึ่งตัวจะได้ ออปโต้ทรานซิสเตอร์ อุปกรณ์ออปโต้ตัวอื่นก็เช่นเดียวกัน

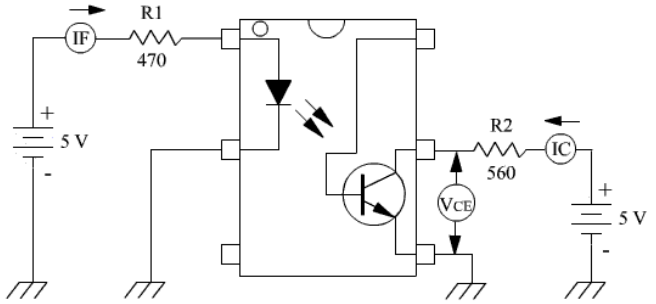


รูปที่ 2.34 สัญลักษณ์อุปกรณ์เชื่อมต่อทางแสงชนิดต่างๆ

ปัจจุบันอุปกรณ์เชื่อมต่อทางแสงถูกสร้างขึ้นมาในรูปของไอซี 6 ขาปิดทึบภายใน ด้านอินพุตจะเป็นแอลอีดีอินฟาเรด (LED Infared) ส่วนทางด้านเอาท์พุตนั้นจะเป็นอุปกรณ์ประเภทโฟโต้ชนิดต่างๆ ซึ่งมีอยู่มากมายเช่น โฟโต้ไดโอด

2. วงจรใช้งานออปโต้คัปเปลอร์

จากรูปที่ 2.35 เป็นวงจรใช้งานเบื้องต้นของออปโต้คัปเปลอร์ โดยมีไดโอดเปล่งแสงเป็นอินพุต และโฟโต้ทรานซิสเตอร์เป็นเอาท์พุตของวงจร เมื่อมีกระแสไหลผ่าน LED โดยมี R_1 เป็นตัวจำกัดกระแส LED จะส่องแสงไปที่โฟโต้ทรานซิสเตอร์ ทำให้โฟโต้ทรานซิสเตอร์นำกระแสมีแรงดันเอาท์พุตตกคร่อมที่ R_2 ซึ่งจะเห็นได้ว่าเอาท์พุตของวงจรจะถูกควบคุมโดยอินพุต โดยทั้งอินพุตและเอาท์พุตแยกกันทางไฟฟ้าโดยสิ้นเชิง วงจรนี้นิยมนำไปใช้ในวงจรควบคุมแรงดันแหล่งจ่ายไฟสวิตชิ่งในเครื่องรับโทรทัศน์ วงจรควบคุมไฟวิ่งวัตต์สูง เป็นต้น



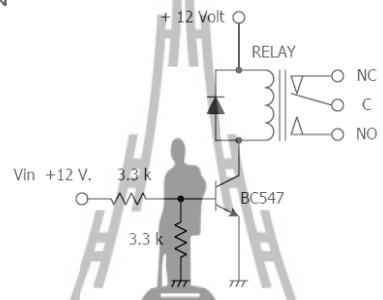
รูปที่ 2.35 วงจรใช้งานออปโต้คัปเปลอร์เบื้องต้น

หัวใจสำคัญของอุปกรณ์เชื่อมต่อทางแสงนั้นคือ " Cerrent Transfer Ratio" (CTR) ซึ่งก็หมายถึง อัตราส่วนระหว่างกระแสอินพุต (I_{in}) ต่อกระแสเอาท์พุต (I_{out}) ดังนั้นสามารถเขียนสมการได้ดังนี้

$$CTR = I_{IN}/I_{OUT}$$

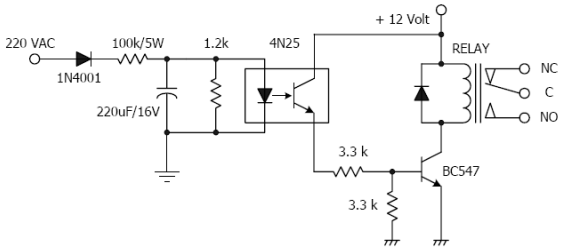
การใช้งานออปโต้คัปเปลอร์

จากรูป 2.36 เป็นการต่อใช้งานสัญญาณสั่งงานควบคุมการทำ งานของอุปกรณ์ผ่านรีเลย์ โดยมีขนาดแรงดันสัญญาณ Vin และรีเลย์ แรงดันไฟต่ำเท่ากัน คือขนาดเท่ากับ 12 โวลต์ อาศัยทรานซิสเตอร์ขับรีเลย์โดยตรง



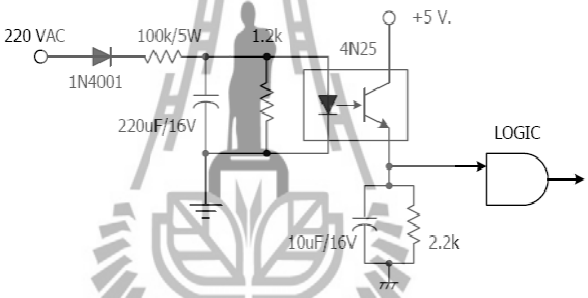
รูปที่ 2.36 วงจรขับรีเลย์โดยใช้ทรานซิสเตอร์

รูปที่ 2.37 เป็นวงจรขับรีเลย์ โดยใช้สัญญาณจากไฟสูงมาขับ OPTO เบอร์ 4N25 มีการเรียงแรงดันเป็นไฟตรงและลดลงเป็นไฟต่ำเพื่อขับ OPTO จากคุณสมบัติของ OPTO จะช่วยแยกส่วนที่เป็นแรงดันไฟสูง ออกจากส่วนที่เป็นแรงดันไฟต่ำโดยสิ้นเชิง ซึ่งถ้าหากส่วนใดเกิดการลัดวงจร จะไม่ทำ ให้เกิดความเสียหายกับวงจรที่เหลือ การต่อวงจรขับด้วย OPTO จะไม่ใช้ขับที่ขารีเลย์โดยตรง เพราะคุณสมบัติของ OPTO ที่มีความจำ กัดของกระแสที่เอาท์พุท ไม่สามารถขับรีเลย์ที่ต้องการกระแสสูงกว่าได้ ดังนั้นจึงต้องต่อขับผ่านตัวทรานซิสเตอร์ BC547 ดังรูปที่ 2.37 ฉะนั้นในการออกแบบวงจรควรพิจารณาคุณสมบัติของ OPTO ที่ต้องการใช้งานให้ดีเสียก่อน



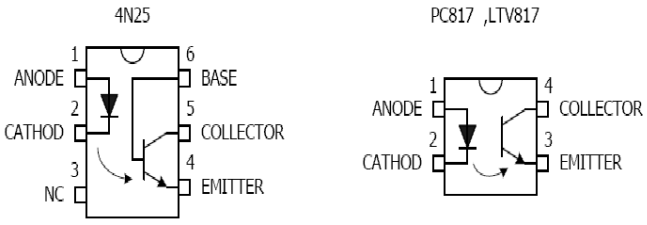
รูปที่ 2.37 วงจรขับรีเลย์โดยใช้ออปโต้คัปเปลอร์กับแรงดันไฟสูง

รูปที่ 2.38 เป็นการต่อวงจรขับโลจิกโดยใช้สัญญาณไฟสูงมาขับวงจร ผ่าน OPTO เบอร์ 4N25 อาศัยแรงดันที่ตกคร่อมบนตัวต้านทาน 2.2k ทำ ให้เกิดสถานะโลจิกเป็น “ 1 “ ไปควบคุม วงจรโลจิกอื่นๆ ตามที่ต้องการได้



รูปที่ 2.38 วงจรขับโลจิกโดยใช้ OPTO กับแรงดันไฟสูง

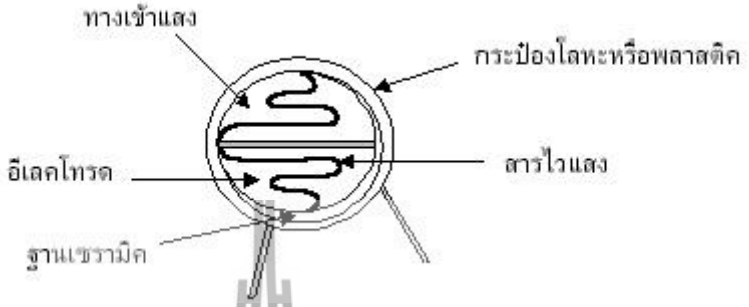
รูปที่ 2.39 เป็นรูปแสดงหน้าที่การทำ งานแต่ละขาของ OPTO #4N25 และมีเบอร์แทนที่ สามารถใช้งานได้ดีและมีจำนวนของขาอุปกรณ์น้อยกว่า สะดวกต่อการใช้งานยิ่งขึ้น



รูปที่ 2.39 หน้าที่แต่ละขา และเบอร์แทนที่ใช้งานได้

ตัวต้านทานไวแสง

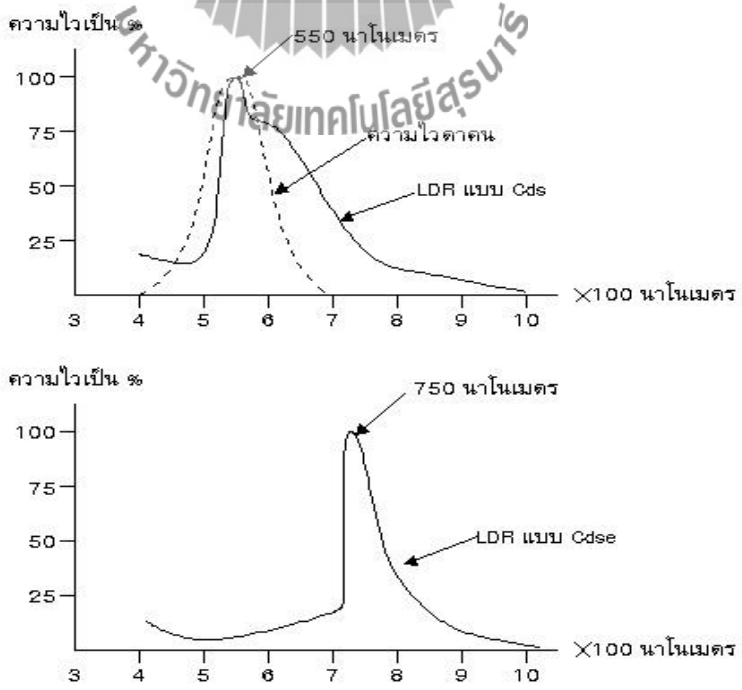
ตัวต้านทานไวแสง (Light Independent Resistor) หรือเรียกสั้น ๆ ว่า LDR ทำมาจากสารแคดเมียมซัลไฟล์ (Cds) หรือแคดเมียมซีลิไนด์ (Cdse) ซึ่งเป็นสารประกอบชนิดกึ่งตัวนำมาฉาบบนแผ่นเซรามิคที่ใช้เป็นฐานรอง แล้วต่อขาจากสารที่ฉาบเอาไว้ออกมาดังโครงสร้างในรูป 2.40



รูปที่ 2.40 โครงสร้างของ LDR

คุณสมบัติทางแสง

LDR ไวต่อแสงในช่วงคลื่น 400-1000 นาโนเมตร (1 นาโนเมตร = 10-9 เมตร) ซึ่งครอบคลุมช่วงคลื่นที่ไวต่อตาคน (400-700 นาโนเมตร) นั่นคือ LDR ไวต่อแสงอาทิตย์ และแสงจากหลอดไฟ หรือ หลอดเรืองแสง และยังไวต่อแสงอินฟาเรดที่ตามองไม่เห็นอีกด้วย (ช่วงคลื่นตั้งแต่ 700 นาโนเมตรขึ้นไป)



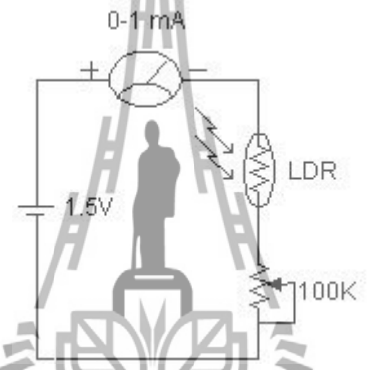
รูปที่ 2.41 กราฟแสดงความไวของ LDR ที่ความยาวคลื่นต่าง ๆ กัน เทียบกับตาคน

คุณสมบัติทางไฟฟ้า

อัตราส่วนของความต้านทาน LDR ขณะที่ไม่มีแสงกับในขณะที่มีแสง อาจมีค่าต่างกัน 100, 1,000, 10,000 เท่า แล้วแต่แบบหรือรุ่นความต้านทานในขณะไม่มีแสงจะอยู่ในช่วงตั้งแต่ 0.5 MW ขึ้นไป และความต้านทานขณะที่มีแสงจะอยู่ในช่วงตั้งแต่ 10 KW ลงมาทนแรงดันสูงสุดได้มากกว่า 100 โวลท์ และทนกำลังไฟได้ประมาณ 50 mW

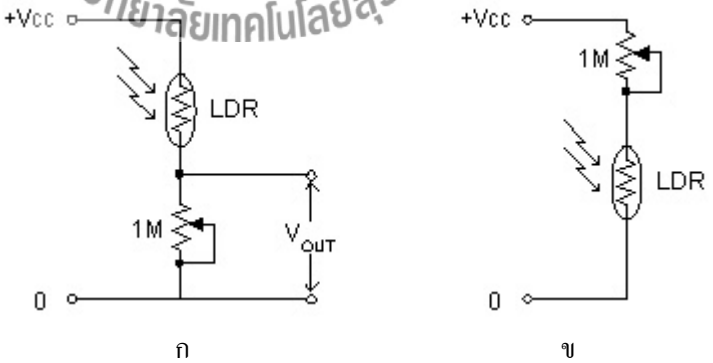
การนำ LDR ไปใช้งาน

จากหลักการดังกล่าวแล้วจะเห็นว่าเมื่อมีแสงสว่างมาตกที่ตัว LDR กระแสที่ไหลผ่านตัว LDR จะสูง เนื่องจากมีความต้านทานต่ำ และเมื่อไม่มีแสงความต้านทานของ LDR มีค่าสูง ทำให้กระแสไหลได้น้อย เราจึงอาจนำ LDR ไปเป็นส่วนประกอบของมิเตอร์วัดความเข้มแสงได้ดังนี้



รูปที่ 2.42 การใช้ LDR เป็นส่วนประกอบของวงจรมิเตอร์วัดแสงอย่างง่าย

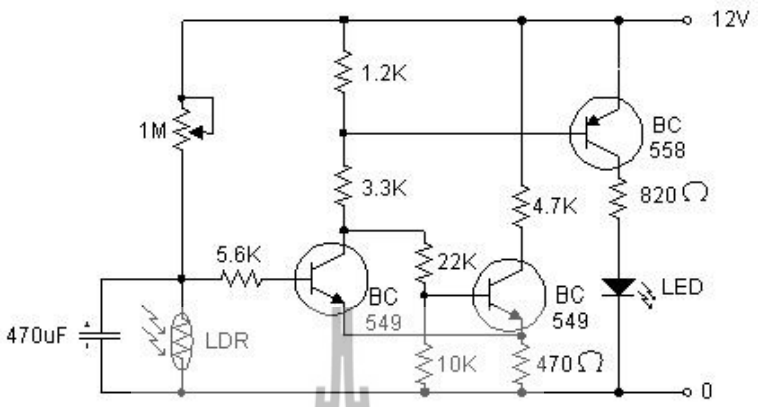
นอกจากนี้เราอาจใช้ LDR ในวงจรควบคุมด้วยแสงได้ดังนี้



รูปที่ 2.43 ตัวต้านทานไวแสงในวงจรควบคุม

ในรูป 2.43 LDR และความต้านทานปรับค่าได้ 1 MW เป็นวงจรแบ่งแรงดันรูป 2.43 ก. แรงดัน Vout จะมีค่าเกือบเท่ากับ Vcc เมื่อมีแสงมาตกและจะมีค่าน้อยเมื่อไม่มีแสงมาตก ส่วนรูป

2.43 ข. เป็นแบบตรงกันข้าม คือ เมื่อมีแสงมาตก แรงดันเอาท์พุทจะมีค่าต่ำ และจะมีค่าสูงเมื่อไม่มีแสงมาตก ตัวอย่างวงจรสวิตซ์ควบคุมด้วยแสงจะทำงานเมื่อไม่มีแสงสว่าง ดังรูปที่ 2.44



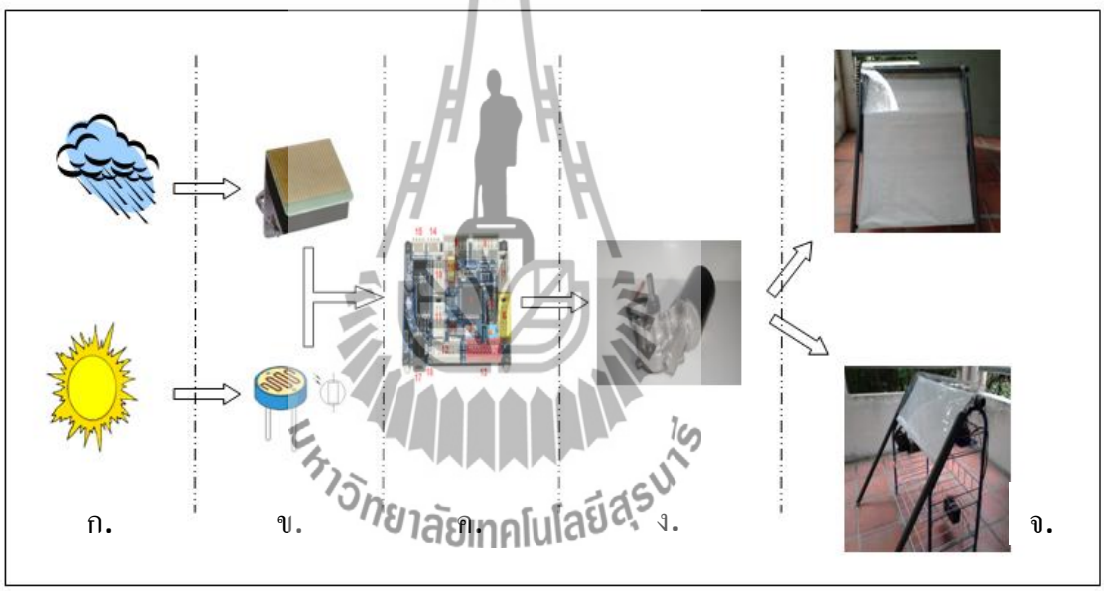
รูปที่ 2.44 วงจรสวิตซ์

บทที่ 3
การออกแบบทางฮาร์ดแวร์

3.1 บทนำ

ในบทนี้จะกล่าวถึงการออกแบบลายวงจรโดยใช้โปรแกรม Protel 99 SE ซึ่งอธิบายการทำงานของโปรแกรมอย่างละเอียด นอกจากนี้ยังแสดงแผนภาพของระบบราวตากผ้าอัตโนมัติโดยอธิบายจุดเด่นของระบบและขั้นตอนการทำแผ่นปริ้น

3.2 แผนภาพระบบราวตากผ้าอัตโนมัติ



รูปที่ 3.1 แผนภาพระบบราวตากผ้าอัตโนมัติ

อธิบายแผนภาพการทำงาน

การทำงานของราวตากผ้าอัตโนมัติมีหลักการทำงานดังนี้

ก. เป็นตัวแปรในการทำงานราวตากผ้าอัตโนมัติซึ่งประกอบด้วย ฝนและแสง

ข. ตรวจจับตัวแปรที่ส่งมาจาก ก. เพื่อส่งไปยังไมโครคอนโทรลเลอร์

ค. ทำหน้าที่ในการรับตัวแปร ข. เพื่อประมวลผลและสั่งให้มอเตอร์ทำงาน

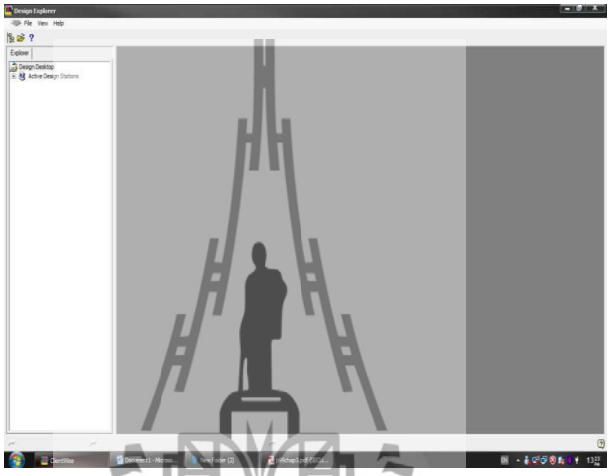
ง. รับคำสั่งจาก ค. เพื่อทำการหมุนเปิด/ปิด หลังคา

จ. รูปแสดงกรณีการเปิดและปิดหลังคา

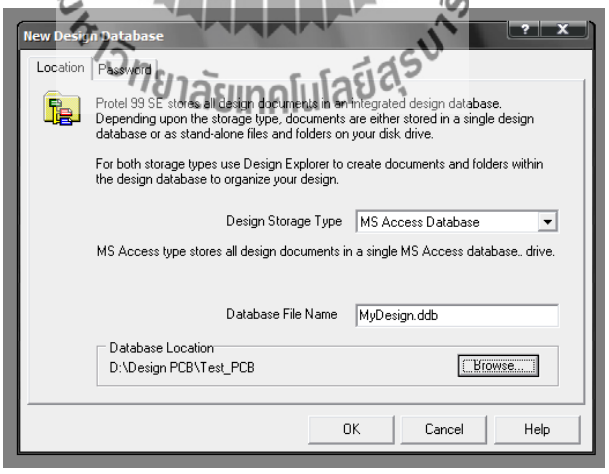
จากแผนภาพการทำงานของระบบราวตากผ้าอัตโนมัติซึ่งมีจุดเด่นในการทำงานคือ สามารถปิดหลังคาลงได้เมื่อฝนตกและเปิดหลังคาขึ้นเมื่อฝนหยุดและมีแสง นอกจากนี้การทำงานของระบบเป็นไปอย่างรวดเร็วและเป็นไปอย่างอัตโนมัติ

3.3 ขั้นตอนการสร้างแผ่นลายวงจรพิมพ์

1. เริ่มต้นการสร้าง Design PCB โดยเปิดโปรแกรม Protel 99 SE จะได้หน้าต่างดังรูป



2. หลังจากนั้นทำการสร้าง New Design ใหม่ขึ้นมา โดยคลิ๊กที่ File --> New จะได้หน้าต่างดังรูป

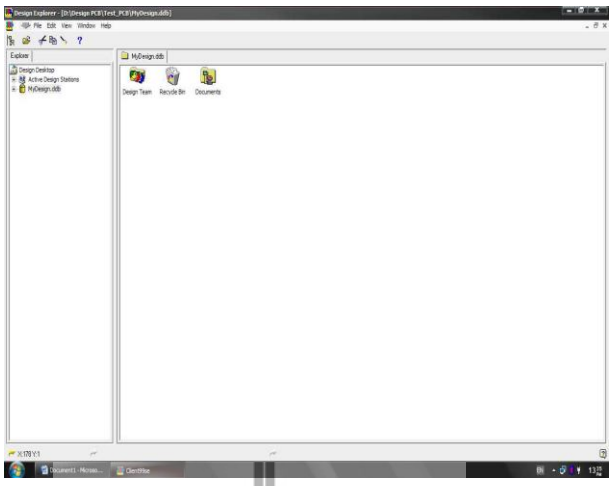


ตั้งชื่อ File และเลือก Location ที่ต้องการบันทึกข้อมูล

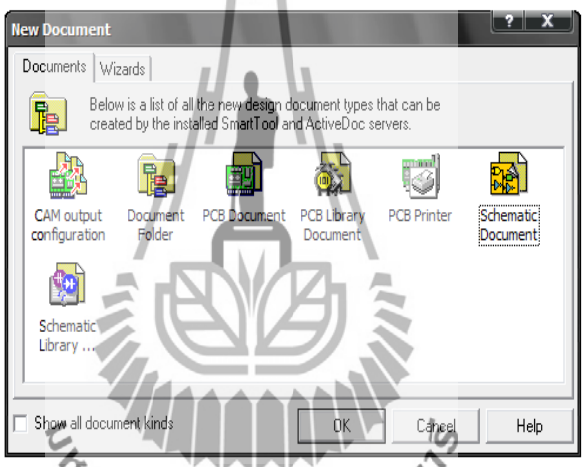
ตั้งชื่อ File เป็น MyDesign.bbd

จัดเก็บ File ไว้ที่ D:\Design PCB\Test_PCB

เสร็จแล้วคลิก OK จะได้หน้าต่างดังรูป

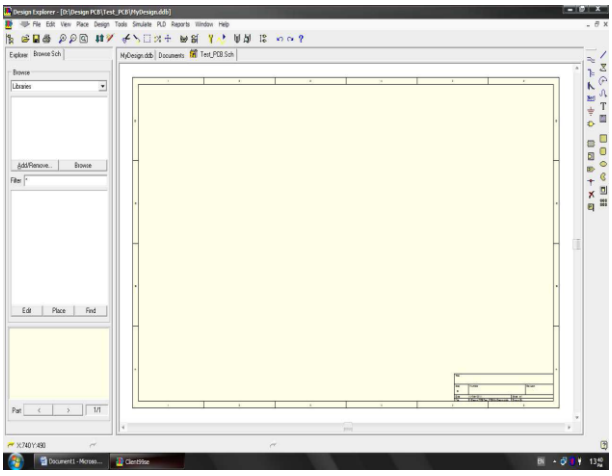


3. เข้าไปที่ Folder Documents แล้วคลิกขวา เลือก NEW….

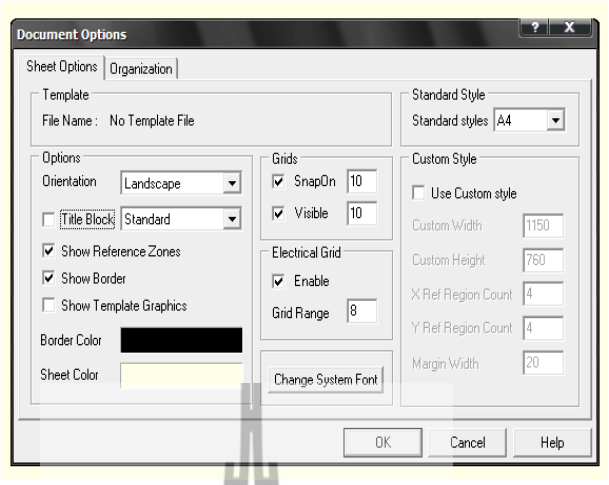


จะแสดงหน้าต่างดังรูป เลือก Schematic Document ทำการตั้งชื่อ Schematic เป็น Test_PCB.Sch

4. เลือก File Test_PCB.Sch ที่อยู่ใน Folder Document จะแสดงหน้าต่างดังรูป

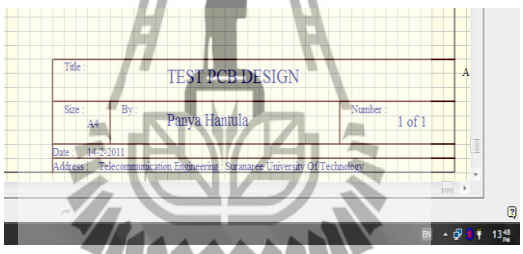


กำหนด Option ของ Schematic โดยเลือกที่ Design --> Option….



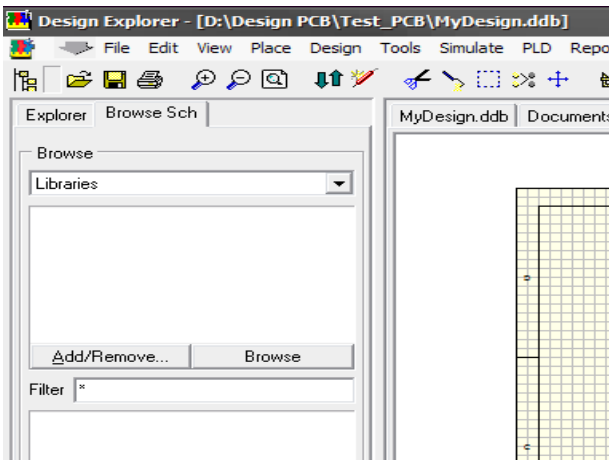
เปลี่ยนขนาดของ schematic เป็น A4

คลิกเครื่องหมายถูกหน้า Title Block เพื่อเอา Title Block ที่เป็น Standard ออก

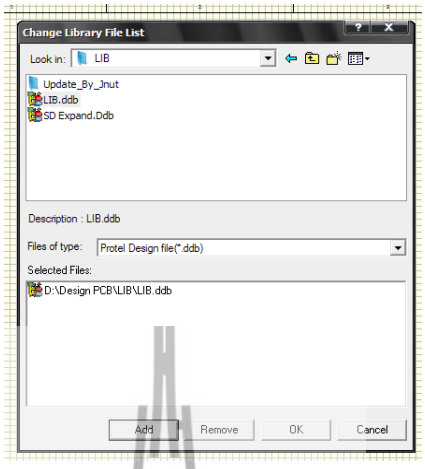


วาด Title Block ใหม่ ตามรูปแบบที่เรากำหนด โดยใช้เครื่องมือ Placel.ine และ PlaceAnnotation (Title Block ใช้แสดงรายละเอียดของงานที่ทำ ไม่จำเป็นเสมอไปที่จะต้องมี)

5. ทำการ Add library โดยการคลิกที่แท็บ Browse Sch --> คลิกที่ Add Remove

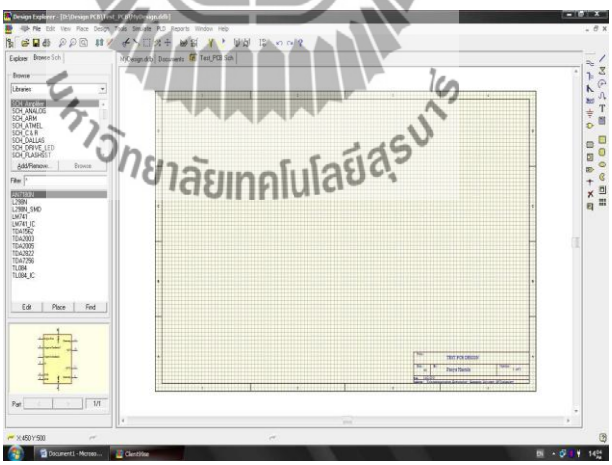


เลือก File Library ที่เก็บอยู่ในเครื่องแล้วคลิกที่ Add จะปรากฏชื่อที่จะทำการ Add Library ในช่อง Selected File ด้านล่าง --> คลิก OK

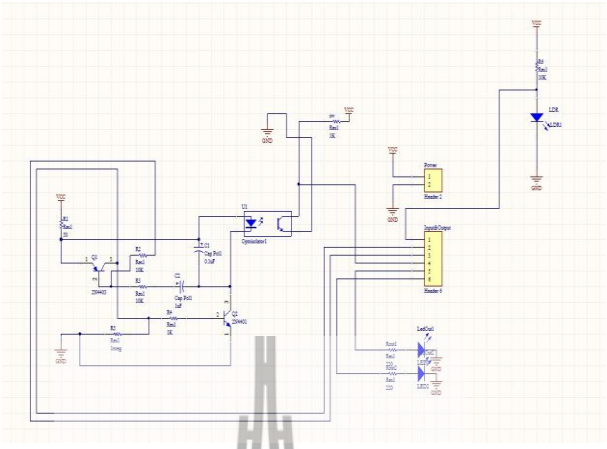


ในกรณีที่ไม่มี Library จะต้องหา Library จะต้องหา Library ที่จะใช้งานก่อนจึงจะสามารถใช้งานได้

6. เมื่อทำการAdd Library เรียบร้อยแล้วจะเป็นอุปกรณ์ที่อยู่ด้านซ้ายมือ ซึ่งสามารถนำมาใช้งานได้โดยการคลิกแล้วนำมาวางที่ schematic

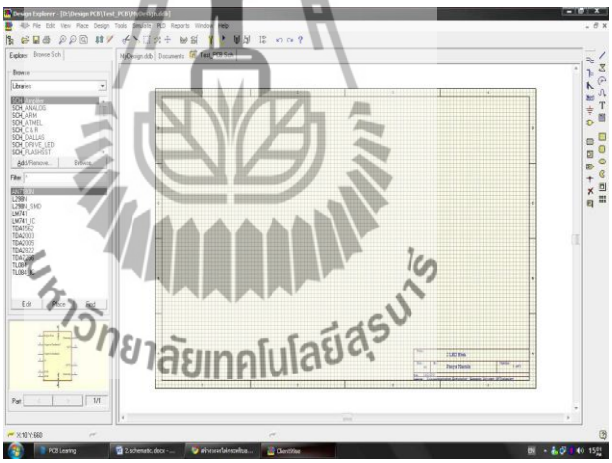


หลังจากที่เราได้รู้จักวิธีการสร้าง New Design ขึ้นมาแล้ว ต่อไปเราจะทำการวาดวงจรที่เราต้องการออกแบบใน Schematic โดยจะทำการออกแบบตามวงจรตัวอย่างดังนี้

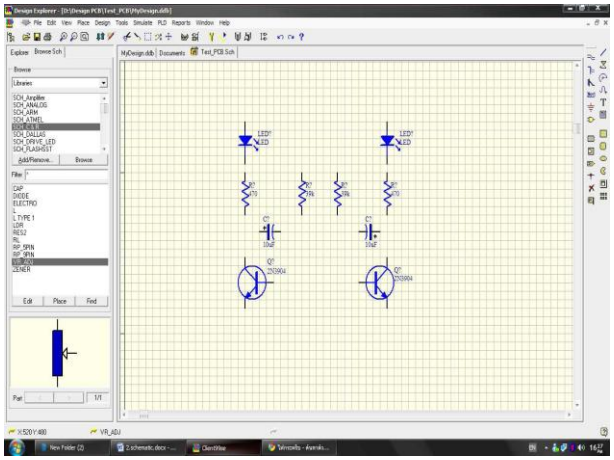


ในวงจรเป็นวงจรของเซ็นเซอร์ ซึ่งการที่จะออกแบบวงจรได้ ควรที่จะเข้าใจการทำงานของวงจรอย่างคราวๆเพื่อที่ว่าเมื่อออกแบบเสร็จแล้วเราสามารถตรวจสอบได้ว่ามีความถูกต้องหรือไม่

1. สร้าง New Design ขึ้นมาตามขั้นตอนที่ได้กล่าวมาแล้ว จะได้หน้าต่างดังรูป



2. จัดวางอุปกรณ์ตามวงจรตัวอย่าง



คำสั่งพื้นฐานในการใช้งาน

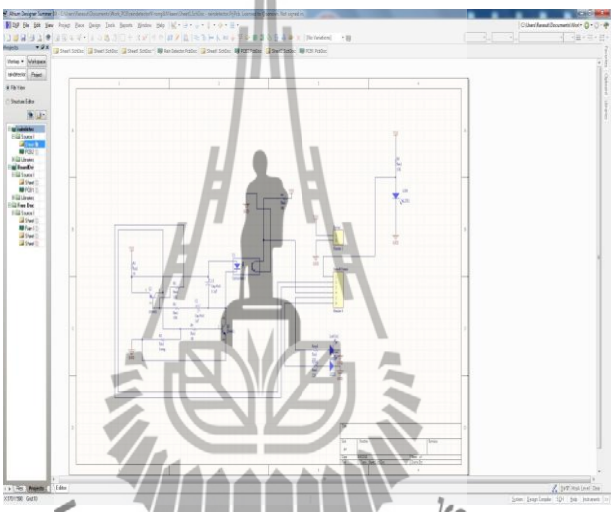
Page Up (Zoom In), Page Down (Zoom Down)

Key X --> File Horizontal , Key Y --> File Vertical

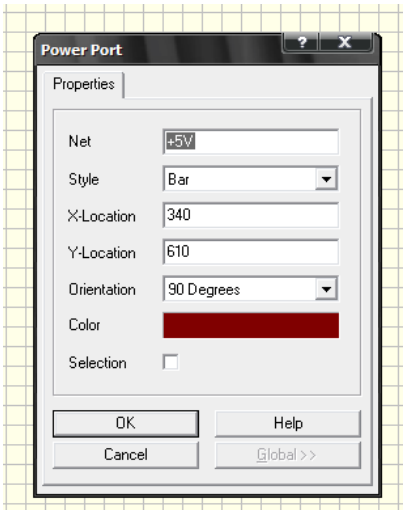
Key Space Bar --> Rotate

**หมายเหตุ ใน Library ที่ต่างกัน อุปกรณ์ที่นำมาวางจะไม่เหมือนกัน

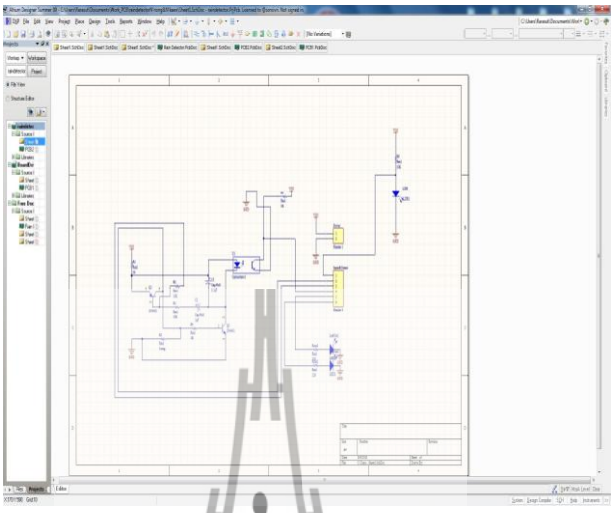
3. เชื่อมต่ออุปกรณ์ด้วยคำสั่ง PlaceWire โดยคลิก Place --> Wire



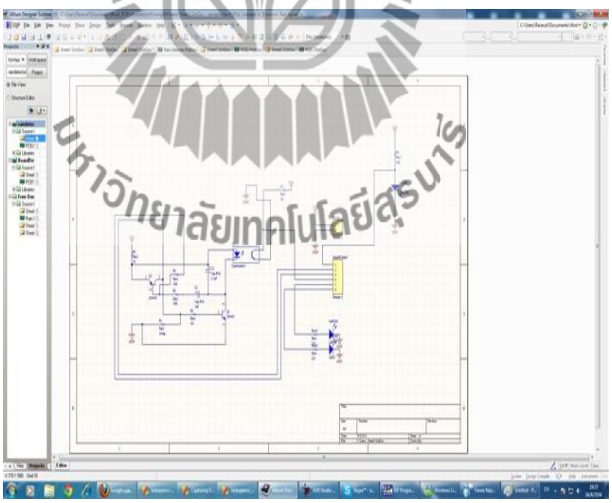
4. ใส่แหล่งจ่ายใครใช้คำสั่ง Place --> Power Port เพื่อเลือกคำสั่ง Power Port แล้วกด Key Tab เพื่อกำหนดคุณสมบัติของ Power Port จะเห็นหน้าต่างดังรูป ใส่ในช่อง Net เป็น +5V และเลือก Style เป็น Bar



5. เมื่อกำหนดแล้ว ให้เพิ่มลงในวงจร และในกรณีของ GND ก็ทำเช่นเดียวกันพียงแต่ใส่ในช่อง Net เป็น GND และเลือก Style เป็น Power Ground



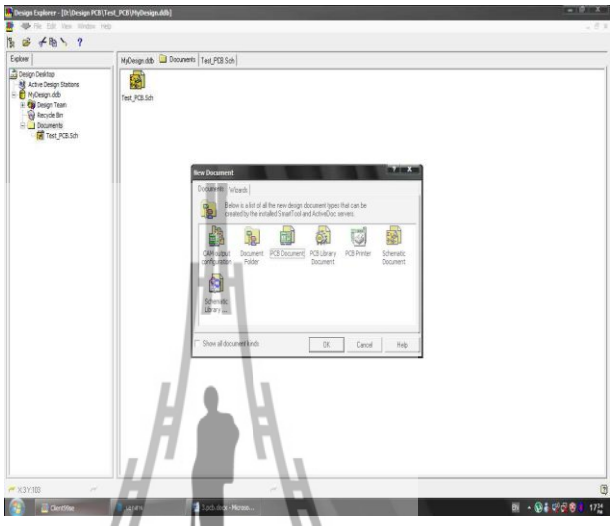
6. ใช้คำสั่ง Annotate เพื่อตั้งชื่อของอุปกรณ์แต่ละตัวไม่ได้ซ้ำกัน โดยไปที่ Tool -- > Annotate แล้วคลิก OK



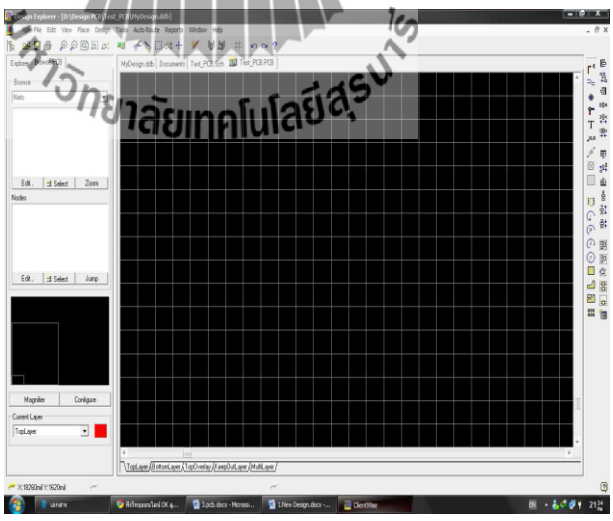
จะเห็นได้ว่า เราได้อุปกรณ์ครบถ้วน และยังมีการตั้งชื่อของอุปกรณ์แต่ละตัวด้วย ถือว่าเป็นการเสร็จในกระบวนการ schematic แล้ว

หลังจากที่ได้ออกแบบวงจรแบบ Schematic เราสามารถสร้าง Artwork ของ PCB ได้จาก Schematic ได้โดยมีขั้นตอนดังนี้

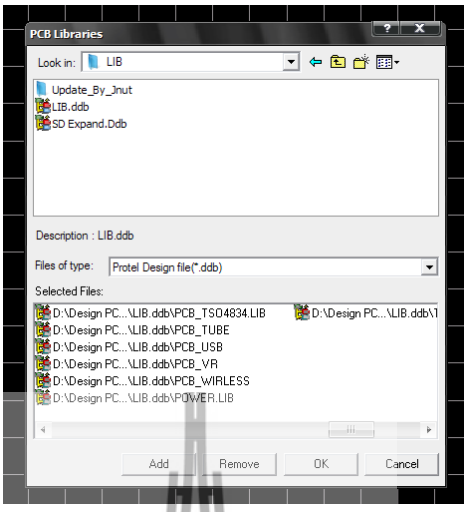
1. เปิด file เราได้สร้าง Schematic จากนั้นไปที่ Folder Documents แล้วคลิกขวา เลือก NEW… เลือก PCB Documents และเปลี่ยนชื่อ File PCB เป็น Test_PCB.PCB



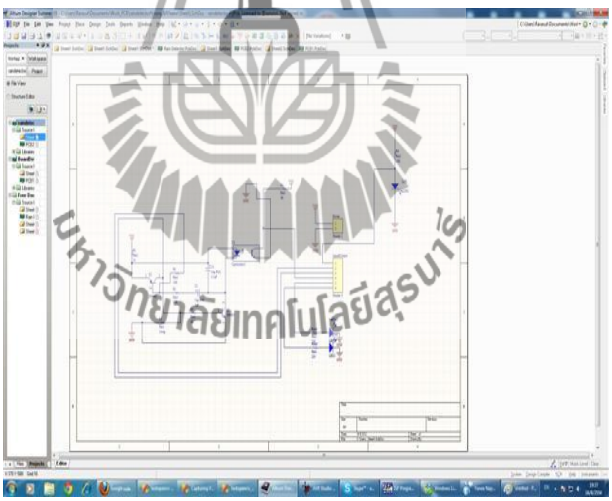
จากนั้นเปิด File Test_PCB.PCB ขึ้นมา จะเห็นหน้าต่างดังรูป



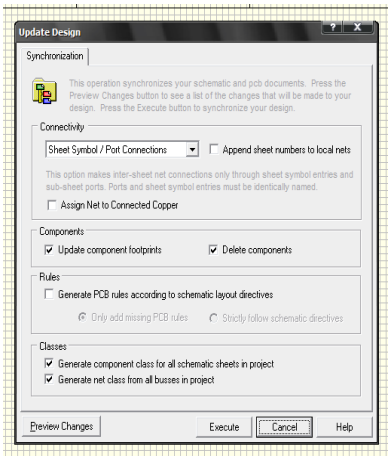
2. ทำการ Add library โดยการคลิกที่ Design -- > Add/Remove Library… และเลือก File Library ที่ต้องการ แล้วคลิกที่ Add -- > คลิก OK

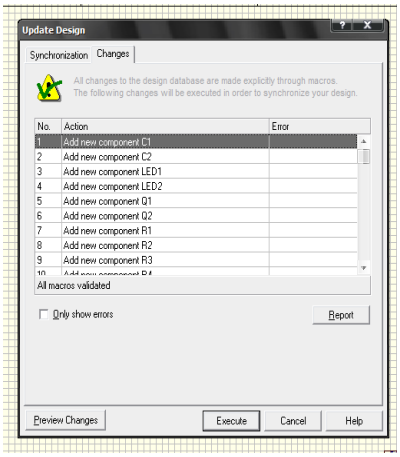


3. เมื่อทำการ Add Library เสร็จแล้ว ให้เปิด File Schematic ขึ้นมาแล้วคลิกที่ Design --> Update PCB

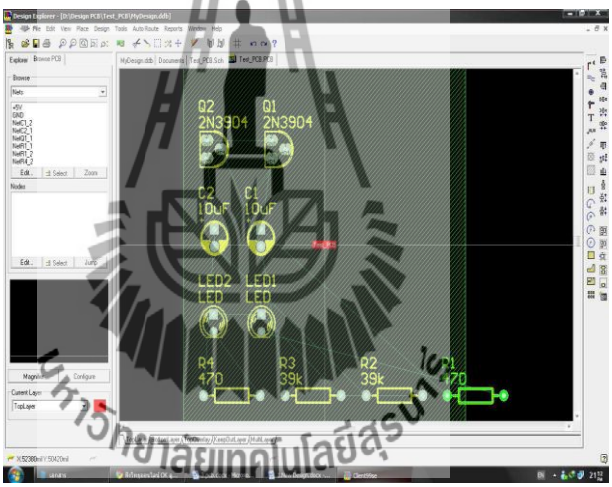


4. คลิกที่ Preview Changes (รูปด้านซ้ายมือ) เพื่อตรวจสอบความผิดพลาด หากตรวจสอบแล้วพบว่าในช่อง Error ไม่มีข้อความเตือน ให้คลิกที่ Executes แต่หากเกิดข้อผิดพลาดให้ทำการแก้ไขก่อน จนกว่าจะไม่เกิด Error



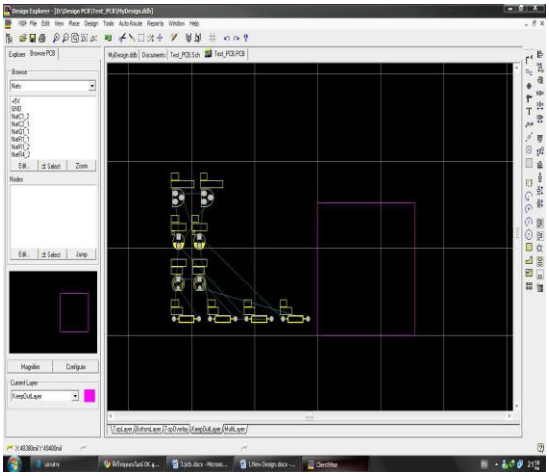


5. เมื่อทำการกด Executes แล้วกลับมาที่ File Test_PCB.PCB อีกครั้ง (ถ้าหากไม่เห็นอุปกรณ์แสดงให้กด Key Z ตามด้วย Key A เพื่อให้แสดงอุปกรณ์)

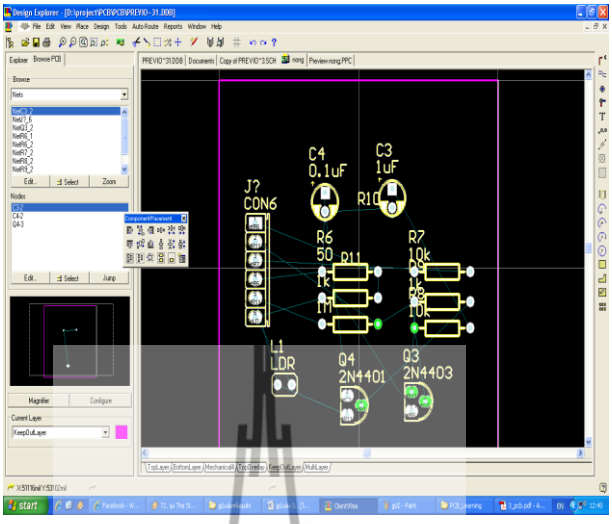


คลิ๊กพื้นที่สีเขียวแล้วกด Key Delete เพื่อลบออก

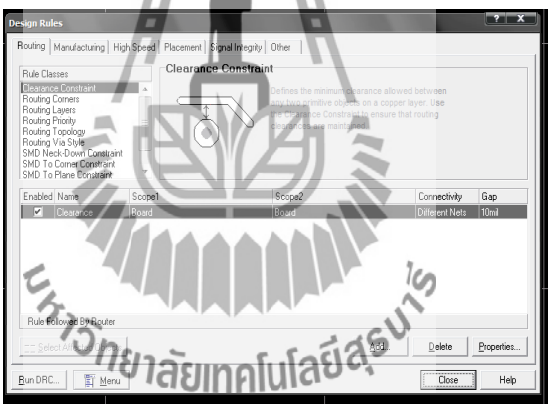
6. เลือกพื้นที่ขอบเขตของแผ่น PCB โดยคลิกที่ KeepOutLayer ด้านล่าง แล้วคลิกที่คำสั่ง Place -- > Line แล้วเลือกพื้นที่ของ PCB



7. จัดวางอุปกรณ์ตามต้องการ (อุปกรณ์นี้คือ Artwork ของ PCB)

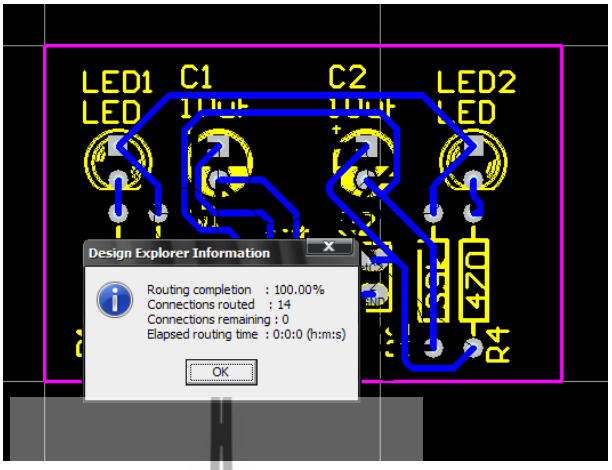


8. ทำการกำหนดค่าต่างๆ โดยคลิกที่ Design --> Rules จะได้หน้าต่างดังรูป



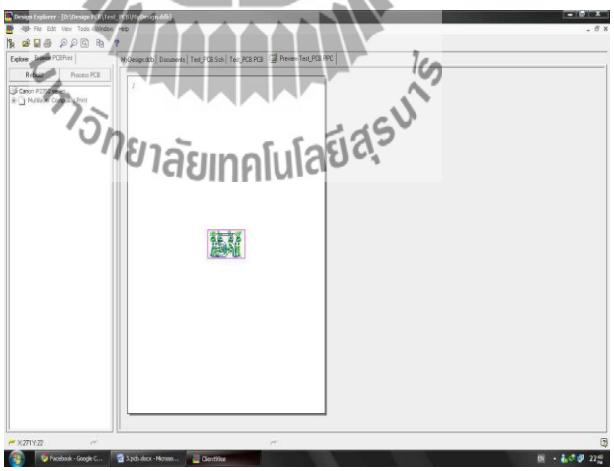
- Clearance Rule คือระยะทางของ Track กับ Pad ปรับค่า Minimum Clearance เป็น 20 mil
- Routing Corners คือลักษณะการเดินเส้นของ Track เมื่อเกิดจุดที่ต้องเลี้ยว
- Routing Layers Rule คือเลือก Layer ที่ต้องการ Rules เช่น หากเป็น PCB แบบ Single Sided จะตั้งค่า Top Layer เป็น Not Used
- Routing Topology คือเทคนิคการ Rule
- Width Constraint คือกำหนดขนาดของเส้นที่ Rule
- Minimum Width คือขนาดของเส้นที่เล็กที่สุดที่สามารถ Rule ได้
- Maximum Width คือขนาดของเส้นที่ใหญ่ที่สุดที่สามารถ Rule ได้
- Preferred Width คือขนาดของเส้นที่ Rule ใน PCB ภายใต้เงื่อนไขที่กำหนด

9. ทำการ Auto Rules โดยใช้คำสั่ง Auto Route -- > All แล้วคลิกที่ Route All

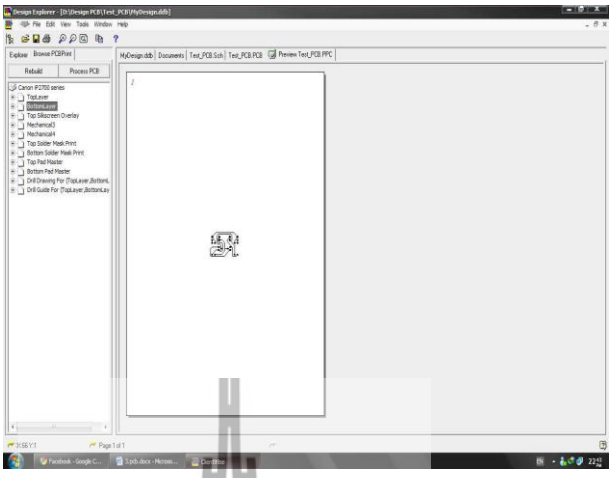


หากสามารถทำ Auto Route ได้หมด จะมีข้อความเตือนว่าสามารถ Rules ได้ 100% และสังเกตลักษณะของเส้นไม่ทับกัน แต่ถ้าหากไม่สามารถทำได้ 100% จะต้องทำการยกเลิกการ Rule โดยใช้คลิกคำสั่ง Tool -- > Un-Route -- > All และปรับการวางอุปกรณ์ใหม่แล้วทำการ Auto Route ใหม่ตามขั้นตอนเช่นเดิม

10. หลังจากที่ได้ Artwork ของ PCB แล้วต้องทำการพิมพ์เพื่อนำไปทำแผ่น PCB ต่อไป คลิกที่ File -- > print/preview จะได้ดังรูป



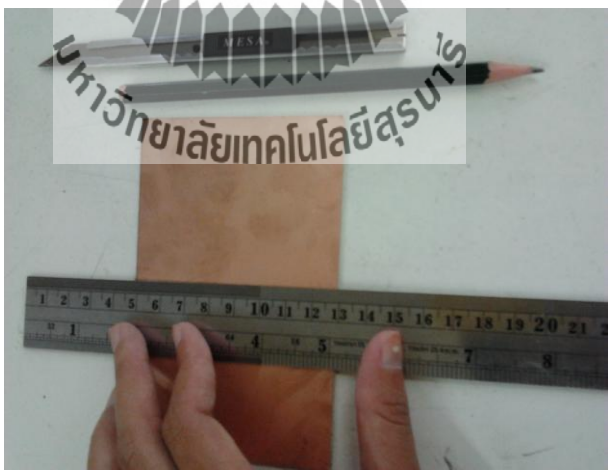
11. คลิกที่คำสั่ง Tool -- > Create Final แล้วเลือก Manu ทางด้านซ้ายมือเพื่อเลือกที่จะพิมพ์ Layer ที่ต้องการ



เพียงเท่านี้เราก็สามารถที่จะสร้างแผ่น PCB ได้ด้วยตัวเอง

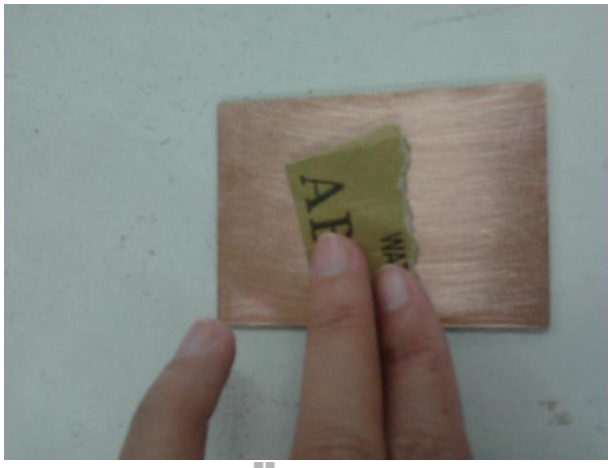
3.4 ขั้นตอนการทำแผ่น PCB วงจร

1. วัดขนาด PCB ที่ต้องการตัด แล้วใช้ปากกาขีดเส้นตามขนาดลงบนแผ่น PCB ขูดด้านบนให้เป็นรอย ใช้ทาบกับขอบโต๊ะให้ตรงตามแนว ออกแรงกดเล็กน้อย แผ่น PCB จะหักตามแนวขูด ดังรูปที่ 3.2



รูปที่ 3.2 การวัดแผ่นปริ้นเพื่อทำการตัด

2. นำมาขัดขอบให้เรียบด้วยกระดาษทราย ใช้กระดาษทรายลบขอบสันด้านข้างด้วย ไม่เช่นนั้นจะทำให้แผ่นใสไม่แนบกับแผ่นทองแดง ดังรูปที่ 3.3



รูปที่ 3.3 การขัดลายวงจรโดยใช้กระดาษทราย

3. นำมาขัดด้วย ใยขัด ให้สะอาด แล้วนำไปล้างด้วยน้ำยาล้างจาน แล้วเช็ดให้แห้ง

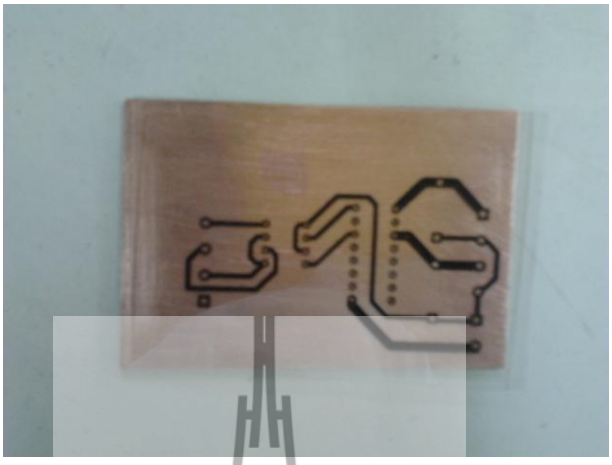
4. ล้างมือให้สะอาด แล้วเช็ดให้แห้ง พยายามอย่าให้มือถูกแผ่นทองแดงอีกเพราะจะทำให้กรดกัดไม่ออก เมื่อมีน้ำมันจากมือไปถูกแผ่นทองแดง

5. นำแผ่น PCB ติดด้วยเทปกาว 2 หน้าชนิดบาง แล้วนำไปติดบนแผ่นไม้

6. นำแผ่นใสมาทาบให้ตรง แล้วติดขอบทั้ง 4 ด้านด้วยเทปกาวใส นำผ้าดิบมาวางทับอีกทีหนึ่งแล้วนำเตารีด มารีด ออกแรงกดเล็กน้อย แล้วรีดไปมาให้ทั่ว ดังรูปที่ 3.4

รูปที่ 3.4 รูปการรีดลายวงจรลงบนแผ่นปริ้น

7. ทิ้งไว้ให้เย็นหรือ นำผ้าชุบน้ำมาดๆ มาเช็ดเพื่อให้เย็นเร็วขึ้นหรือนำไปเป่าด้วยพัดลม เมื่อลอกออก จะเห็นว่าลายจะหลุดติดอยู่บนแผ่นทองแดง ดังรูปที่ 3.5



รูปที่ 3.5 รูปหลังการรีดลายวงจรลงบนแผ่นปริ้น

8. นำปากาเขียนแผ่นใส เติมส่วนที่ขาด เวลาเขียนต้องยกปากาไว้นิดๆ หมึกจะไหลดี หากปากาแห้งให้ นำไปเขียนลงบนกระดาษ โดยออกแรงกดเล็กน้อย แล้วหมุนไปหมุนมา จนหมึกเริ่มไหลดี หากหมึกเลอะปล่อยให้แห้ง แล้วใช้เหล็ก ปลายแหลมขูดออก หรือรูเล็กๆ ถูกหมึกระบายเต็มจนไม่เห็นรู ให้ระบายให้เต็มไปเลย แล้วใช้เหล็ก ปลายแหลม จุดไว้ เวลาเจาะจะได้ง่าย

9. นำไปกัดกรด นำกาวร้อนมาลนไฟ นำมาป้ายไว้ที่แผ่ยึด แล้วติดแผ่น PCB นำไปกัดกรด ตรวจสอบดูว่าตรงไหนกรดไม่กัด สังเกตจากสีจะอ่อนกว่า นำพู่กัน ขนอ่อน ถูบริเวณที่กรดไม่กัดให้ทั่ว เมื่อกรดกัดแผ่นทองแดงส่วนที่ไม่ต้องการ หมดแล้ว นำไปล้างน้ำแล้ว ขัดด้วยใยขัด จนสะอาดแล้วเช็ดให้แห้ง

10. เมื่อนำแผ่นปริ้นที่ไม่สะอาด หรือมีน้ำมันจากมือ ไปกัดกรด หลายๆครั้ง จะทำให้กรดเป็นฝ้าให้นำกรดที่เป็นฝ้า ไปกรองด้วยผ้ากรองก่อนนำมาใช้ใหม่

11. นำแผ่นทองแดงมาเจาะรู ที่เครื่องเจาะรู โดยต้องใช้ความแม่นยำและสมาธิระหว่างการเจาะ ดังรูปที่ 3.6

รูปที่ 3.6 การเจาะแผ่นปริ้น

12. ใส่อุปกรณ์โดยใส่อุปกรณ์ตัวเตี้ยๆก่อน
13. ทำการบัดกรีอุปกรณ์ต่างๆ แล้วทำการทดสอบการทำงานของวงจร ดังรูปที่ 3. 7

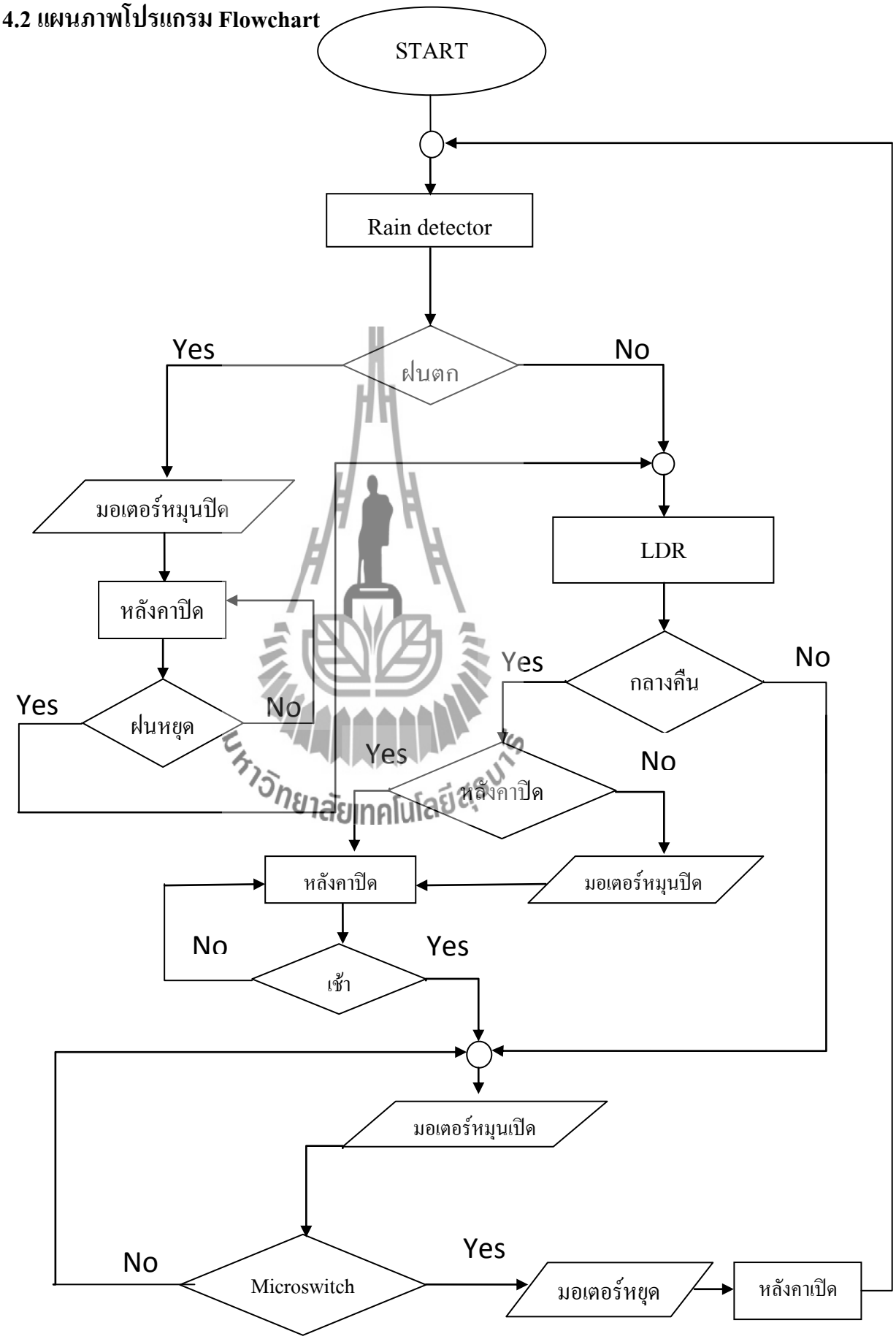
รูปที่ 3.7 การบัดกรีอุปกรณ์ลงบนแผ่นปริ้น

บทที่ 4
การออกแบบทางซอฟแวร์

4.1 บทนำ

การออกแบบทางซอฟแวร์ของระบบราวตากผ้าอัตโนมัติทำงานโดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมซึ่งคำนึงถึงตัวแปรที่รับมาจากเซ็นเซอร์ ในบทนี้จะแสดงแผนภาพโปรแกรมและอธิบายการทำงานของโปรแกรม รวมถึงการพัฒนาโปรแกรมด้วย AVR Studio 4 การเขียนโปรแกรมภาษา C สำหรับไมโครคอนโทรลเลอร์ AVR ซึ่ง AVR Studio จะทำงานร่วมกับ WinAVR ที่ใช้ในการคอมไพเลอร์

4.2 แผนภาพโปรแกรม Flowchart



อธิบายแผนภาพการทำงาน

จากแผนภาพโปรแกรมจะสามารถสรุปการทำงานของระบบราวตากผ้าอัตโนมัติได้ดังนี้

1. เมื่อฝนตกมอเตอร์จะหมุนหลังคาปิดโดยจะหน่วงเวลาการหมุน 8 วินาที

2. เมื่อไม่มีฝนจะต้องทำการตรวจสอบว่าเป็นกลางวันหรือกลางคืน ถ้าเป็นกลางคืนก็จะมีการเช็คอีกว่าขณะนี้หลังคาเปิดหรือปิด ถ้าปิดก็จะยังคงปิดอยู่ หรือถ้าเปิดมอเตอร์ก็จะหมุนหลังคาปิดลง

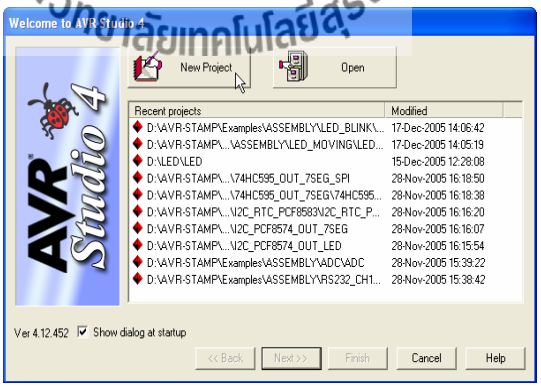
3. เมื่อไม่มีฝนตกและเป็นเวลากลางวันก็จะทำการเปิดหลังคาขึ้นขณะที่หลังคาหมุนขึ้น ถ้าหลังคาชนไมโครสวิตช์ก็จะทำให้มอเตอร์หยุดหมุนทันที

4. กรณีที่เป็นเวลากลางคืนและฝนไม่ตกก็จะทำการเช็คว่าขณะนี้หลังคาเปิดหรือปิดอยู่ ถ้าหลังคาปิดก็จะยังคงปิดอยู่ แต่ถ้าหลังคาเปิดมอเตอร์ก็จะหมุนหลังคาปิดลง

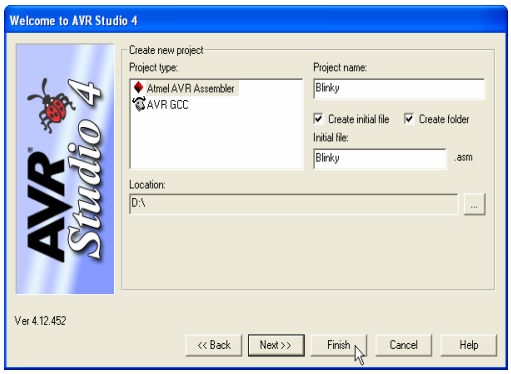
4.3 การพัฒนาโปรแกรมด้วย AVR Studio 4

สำหรับโปรแกรมที่จะใช้ในการเขียนโปรแกรมภาษาแอสเซมบลี ก็คือโปรแกรม AVR Studio 4เนื่องจาก Assembler ตัวนี้ทางบริษัท Atmel แจกให้ลูกค้าใช้ฟรีดังนั้นผู้ทดลองสามารถเข้าไปดาวน์โหลดโปรแกรมได้ที่เว็บไซต์ www.atmel.com แต่อย่างไรก็ตามทางทีมงานได้รวบรวมโปรแกรมนี้ไว้ในแผ่น CDROM แล้ว ซึ่งเป็นโปรแกรม AVR Studio 4 เวอร์ชั่น 4.12 ซึ่งขั้นตอนการใช้งานโปรแกรมมีดังนี้

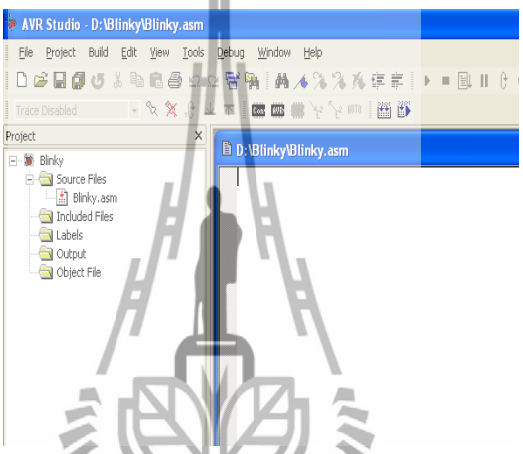
1. ทำการเปิดโปรแกรม AVR Studio จากนั้นจะปรากฏหน้าต่าง Welcome to AVR Studio ให้คลิกที่ New Project เพื่อสร้างโปรเจคใหม่ดังรูป



2. เลือก Project type เป็น Atmel AVR Assembler เพื่อเขียนโปรแกรมเป็นภาษาแอสเซมบลี ทำการตั้งชื่อโปรเจคในช่อง Project name เลือกที่ช่อง Create initial file เพื่อสร้างไฟล์แอสเซมบลีพร้อมกับสร้างไฟล์โปรเจค เลือกที่ช่อง Create folder เพื่อสร้างโฟลเดอร์สำหรับเก็บไฟล์โปรเจค จากนั้นทำการเลือกไดเร็กทอรี่ที่จะเก็บไฟล์โปรเจคและคลิกปุ่ม Finish ดังรูป



3. จากนั้นจะปรากฏหน้าต่าง Text Editor สำหรับเขียนโปรแกรมดังรูป

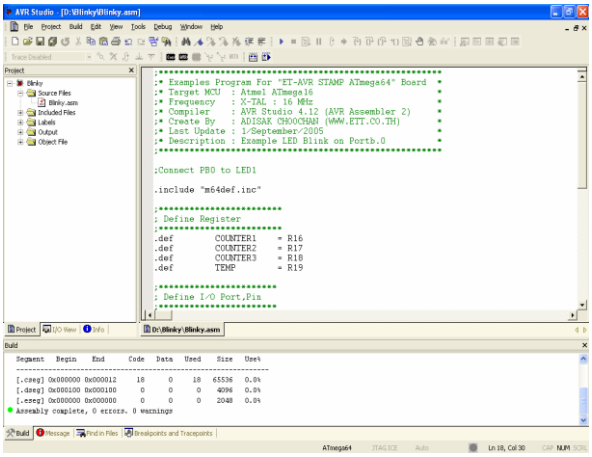


4. ทำการพิมพ์โปรแกรมตัวอย่างภาษาแอสเซมบลี ดังตัวอย่าง ซึ่งเป็นตัวอย่างโปรแกรมไฟกระพริบที่ PORTB.0


```
;*****************************************************
;* Examples Program For "ET-AVR STAMP ATmega64" Board *
;* Target MCU : Atmel ATmega16 *
;* Frequency : X-TAL : 16 MHz *
;* Compiler : AVR Studio 4.12 (AVR Assembler 2) *
;* Create By : ADISAK CHOOCHAN (WWW.ETT.CO.TH) *
;* Last Update : 1/September/2005 *
;* Description : Example LED Blink on Portb.0 *
;*****************************************************
;Connect PB0 to LED1
.include "m64def.inc"
;************************
; Define Register
;************************
.def COUNTER1 = R16
.def COUNTER2 = R17
```

```
.def COUNTER3 = R18
.def TEMP = R19
;***********************
; Define I/O Port,Pin
;***********************
.equ LED = 0
;*******************
; Main Program
;*******************
.CSEG
.ORG 0
RJMP RESET ;Reset Handle
RESET: LDI TEMP,LOW(RAMEND);Initial Stack Pointer
OUT SPL,TEMP
LDI TEMP,HIGH(RAMEND)
OUT SPH,TEMP
SBI DDRB,LED ;Config Portb.0 as output
MAIN: SBI PORTB,LED
RCALL DELAY_200ms
CBI PORTB,LED
RCALL DELAY_200MS
RJMP MAIN
;/*******************
; Delay time
;/*******************
DELAY_1ms: LDI COUNTER1,16
DELAY_1ms_1: LDI COUNTER2,250
DELAY_1ms_2: NOP
DEC COUNTER2
BRNE DELAY_1ms_2
DEC COUNTER1
BRNE DELAY_1ms_1
RET
DELAY_200ms: LDI COUNTER3,200
DELAY_200ms_1: RCALL DELAY_1ms
DEC COUNTER3
BRNE DELAY_200ms_1
RET
```


5. ให้ทำการสั่งแปลโปรแกรมที่เราเขียนขึ้น โดยการคลิกเมาส์ที่เมนูคำสั่ง Build --> Build ซึ่งหลังจากแปลโปรแกรมแล้วได้ผลถูกต้องและไม่เกิดข้อผิดพลาดใด ๆ จะปรากฏข้อความ 0 errors 0 warnings ต่อจากนี้ผู้ใช้ก็สามารถนำ Hex File ที่ได้จากสั่งแปลโปรแกรมนี้ไปทำการ Download ลง MCU ได้ทันที



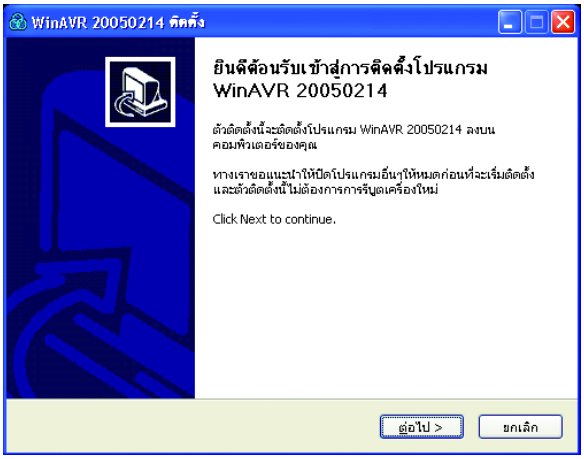
4.4 WinAVR

WinAVR เป็นซอฟต์แวร์ C คอมไพเลอร์หรือตัวแปลโปรแกรมภาษา C สำหรับไมโครคอนโทรลเลอร์ AVR โดย WinAVR เป็นซอฟต์แวร์แบบโอเพ่นซอร์ส (Open source) พัฒนาต่อจาก GNU GCC คอมไพเลอร์ เมื่อทำการติดตั้ง WinAVR แล้ว จะสามารถเชื่อมโยงการทำงานเข้ากับ AVR Studio ได้ ดังนั้นจึงสามารถทำการเขียนโปรแกรมภาษา C บน AVR Studio แล้วทำการคอมไพล์โปรแกรมด้วย WinAVR ได้อย่างต่อเนื่องโดยผลลัพธ์ของการคอมไพล์จะได้เป็นไฟล์นามสกุล .hex อันเป็นไฟล์รหัสภาษาเครื่องหรือที่เรียกว่า "แมชีนโค้ด" โดยเป็นไฟล์ผลลัพธ์ที่ได้จากการพัฒนาสามารถนำไปดาวน์โหลดลงสู่ไมโครคอนโทรลเลอร์ต่อไปได้ทันที

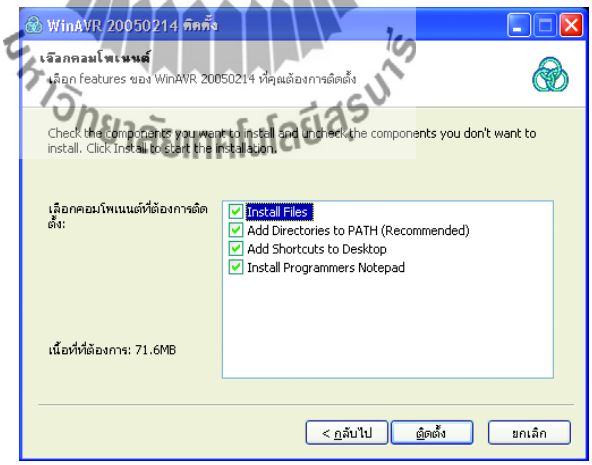
การติดตั้งโปรแกรม WinAVR
มีขั้นตอนโดยสรุปดังนี้

1. ดับเบิ้ลคลิกที่ไฟล์ WinAVR-20050214-install.exe เพื่อติดตั้งโปรแกรม WinAVR
2. เข้าสู่หน้าต่างแรกของการติดตั้งโปรแกรม WinAVR เป็นการเลือกภาษาในการติดตั้ง คลิกที่ปุ่ม OK

3. หลังจากนั้นเข้าสู่หน้าแสดงรายละเอียดการดำเนินการติดตั้งโปรแกรม WinAVR คลิกที่ปุ่มต่อไป >



4. เข้าสู่หน้าต่างแสดงรายละเอียดและเงื่อนไขสำหรับการใช้งานโปรแกรม WinAVR จากนั้นคลิกที่ปุ่ม ยอมรับ>
5. จากนั้นเข้าสู่หน้าต่างแสดงชื่อพาธของคอมพิวเตอร์ที่ต้องการติดตั้งโปรแกรม WinAVR และพื้นที่ที่ต้องการใช้สำหรับการติดตั้งโปรแกรมโดยผู้ติดตั้งสามารถเปลี่ยนพาธได้โดยการคลิกที่ปุ่ม เรียกหา... แล้วกำหนดพาธใหม่ที่ต้องการ
6. ต่อไปเข้าสู่หน้าต่างการกำหนดคุณสมบัติพิเศษ ซึ่งผู้ติดตั้งสามารถเลือกรายการตามที่ต้องการแล้วคลิกที่ปุ่มติดตั้ง>



7. หลังจากนั้นจะเริ่มกระบวนการติดตั้งโปรแกรม WinAVR และแสดงความคืบหน้าตามรูป หลังจากนั้นให้ผู้ติดตั้งรอจนกระทั่งการติดตั้งเสร็จสมบูรณ์ หลังจากนั้นคลิกที่ปุ่ม เสร็จสิ้น เป็นอันว่าการติดตั้งโปรแกรม WinAVR เสร็จสมบูรณ์

บทที่ 5

ผลการทดลอง

5.1 การทดลองที่ 1 ทดสอบการทำงานของวงจรเซ็นเซอร์น้ำฝน

วัตถุประสงค์

1. เพื่อเข้าใจการทำงานของวงจรเซ็นเซอร์น้ำฝน
2. เพื่อเข้าใจหลักการทำงานของมอเตอร์
3. เพื่อเข้าใจการแปลงสัญญาณแอนาลอกเป็นสัญญาณดิจิตอล

อุปกรณ์การทดลอง

1. ราวตากผ้าอัตโนมัติ
2. เครื่องคอมพิวเตอร์
3. สาย RS232

ความรู้พื้นฐาน

ราวตากผ้าอัตโนมัติ (Automatic clothes line) มีหลักการทำงานคือ เมื่อฝนตกถูกตัวเซ็นเซอร์จะทำให้ไมโครคอนโทรลเลอร์ทำการสั่งให้มอเตอร์หมุนเพื่อหมุนหลังคาปิดผ้าที่ตากไว้ และเมื่อฝนหยุดไมโครคอนโทรลเลอร์ก็จะทำการสั่งให้มอเตอร์หมุนกลับเพื่อหมุนหลังคาเปิดและตากผ้าเหมือนเดิม และนอกจากนี้ยังสามารถหมุนหลังคาปิดได้เมื่อถึงเวลาตอนเย็นเพื่อไม่ให้ผ้ามีความชื้นและทำการหมุนหลังคาเปิดเมื่อถึงตอนเช้าอีกด้วย

การทำงานของราวตากผ้าอัตโนมัตินั้นมีการสั่งงานมอเตอร์โดยไมโครคอนโทรลเลอร์ AVR เบอร์ ATmega128 ซึ่งมีโมดูลแปลงสัญญาณแอนาลอกเป็นดิจิตอล หรือ ADC (Analog to Digital Converter) ความละเอียดขนาด 10 บิต (10-bit Resolution) ที่แรงดัน +5V หมายถึงเมื่อแปลงสัญญาณดิจิตอลแล้วจะได้ค่าตัวเลขอยู่ระหว่าง 0-1024 โดยมีรูปแบบการแปลงสัญญาณแอนาลอกเป็นดิจิตอลแบบซัคเซสซีฟ แอพพร็อกซิเมชั่น (Successive Approximation ADC) คือการแปลงแบบประมาณค่า โดยการสุ่มค่าดิจิตอลแล้วแปลงเป็นแรงดันแอนาลอกภายในโมดูล เพื่อใช้เปรียบเทียบกับแรงดันแอนาลอกด้านอินพุต เมื่อเปรียบเทียบได้ค่าแรงดันเท่ากัน โมดูล ADC จะให้ผลลัพธ์ออกมาเป็นค่าดิจิตอล ซึ่งการใช้วิธีการนี้เป็นที่นิยมเพราะมีความเที่ยงตรงสูงและทำงานได้อย่างรวดเร็ว

โดยผลการแปลงสัญญาณแอนาลอกเป็นดิจิตอล คำนวณผลลัพธ์ได้จากสูตรต่อไปนี้

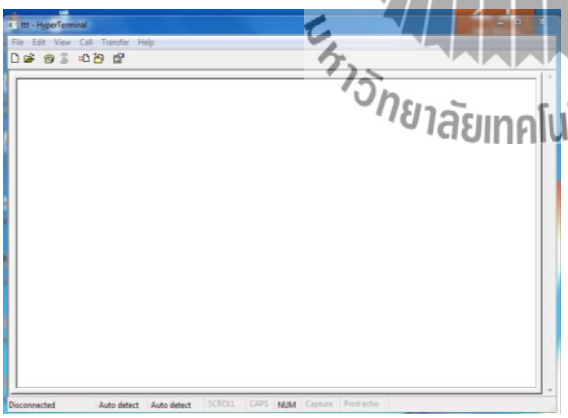
$$ADC = \frac{Vin.1024}{Vref} \qquad \text{(สมการที่ 5.1)}$$

โดยที่ Vin = แรงดันอินพุต

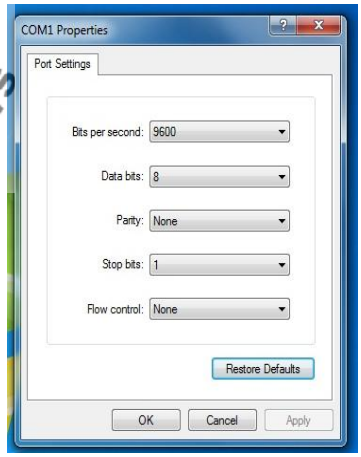
Vref = แรงดันอ้างอิง

ขั้นตอนการทดลอง

1. นำสาย RS232 เชื่อมต่อกับบอร์ด AVR ATmega128 ที่อยู่ในราวตากผ้าอัตโนมัติ
2. เข้าไปที่ Hyper Terminal ทำการตั้งค่า bit per second ให้มีเท่ากับ 9600
3. สังเกตและบันทึกผลการทดลอง
4. นำน้ำมาหยดใส่แผงเซ็นเซอร์น้ำฝน สังเกตและบันทึกผลการทดลอง
5. เช็ดแผงเซ็นเซอร์น้ำฝนให้แห้ง สังเกตและบันทึกผลการทดลอง
6. ทำซ้ำข้อ 4
7. ทำซ้ำข้อ 5

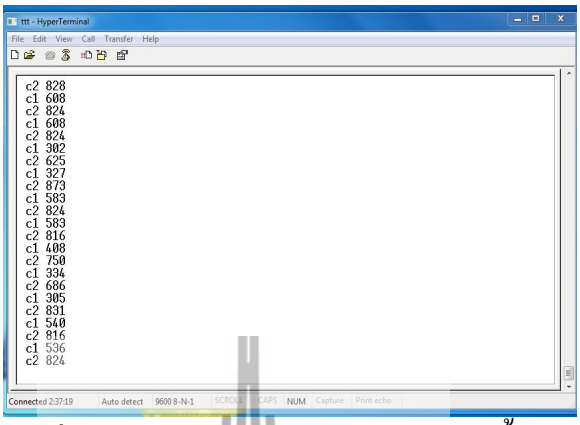


รูปที่ 5.1 หน้าหลักของ Hyper Terminal



รูปที่ 5.2 COM1 Properties

ผลการทดลอง



รูปที่ 5.3 ค่า ADC ของเซ็นเซอร์ตรวจจับน้ำฝน

ตารางที่ 5.1 ผลการทดลองเซ็นเซอร์น้ำฝน

ค่าที่แสดงใน Hyper Terminal(c2)=ADC	แรงดันอินพุต Vin(V)	การเปลี่ยนแปลงของหลังคา (กรณีหลังคาเปิดอยู่)
828	4.043	หลังคาเปิด
824	4.023	หลังคาเปิด
824	4.023	หลังคาเปิด
625	3.051	**หลังคาปิด**
873	4.262	หลังคาเปิด
824	4.023	หลังคาเปิด
816	3.984	หลังคาเปิด
750	3.662	หลังคาเปิด
686	**3.349**	**หลังคาปิด**
831	4.057	หลังคาเปิด

วิเคราะห์ผลการทดลอง

จากการทดลองพบว่า ที่สถานะปกติกรณีที่ยังไม่มีน้ำหยดใส่แผงเซ็นเซอร์น้ำฝนหลังคาจะเปิด ค่าที่อ่านได้ใน Hyper Terminal มีค่าในการสุ่มประมาณ 750 – 873 ซึ่งจะได้ค่าแรงดันอินพุต (Vin) ประมาณ 3.662V – 4.262V และเมื่อทำการหยดน้ำลงบนแผงเซ็นเซอร์น้ำฝนพบว่า ค่าที่อ่านได้ใน Hyper Terminal มีค่าน้อยกว่า 700 จะได้ค่าแรงดันอินพุต (Vin) น้อยกว่า 3.4 V ซึ่งเป็นผลทำให้มอเตอร์ทำงาน หมุนหลังคาปิดลง สามารถหาค่า Vin ได้จากสมการที่ 5.1

สรุปผลการทดลอง

ราวตากผ้าอัตโนมัติเป็นการทำงานโดยมีการเช็คสถานะแรงดันอินพุตที่ทำการแปลงจากสัญญาณแอนาลอกเป็นสัญญาณดิจิตอลซึ่งค่าที่ได้จะอยู่ในเลขฐานสองมีขนาด 10 บิต จากการกำหนดค่าในการเขียนโปรแกรมโดยกำหนดให้ ค่าแรงดันอินพุตที่แสดงใน Hyper Terminal มีค่าตั้งแต่ 0-1024 ถ้ามีค่ามากกว่า 700 จะทำให้หลังคาเปิดขึ้นโดยจะมีการเช็คสถานะก่อนว่าตอนนี้หลังคาเปิดอยู่หรือไม่ ถ้าเปิดอยู่ก็จะไม่มีการเปลี่ยนแปลง แต่ถ้าหลังคาปิดอยู่มอเตอร์ก็จะทำการหมุนหลังคาให้เปิดขึ้น และจะมีการเช็คสถานะเพื่อรอรับค่าอยู่ตลอดเวลา

เมื่อมีฝนตกใส่แผงเซ็นเซอร์ค่าแรงดันอินพุตที่แสดงใน Hyper Terminal จะมีค่าน้อยกว่าหรือเท่ากับ 700 จะทำให้หลังคาปิดลง โดยจะมีการเช็คสถานะก่อนว่าตอนนี้หลังคาปิดอยู่หรือไม่ ถ้าปิดอยู่ก็จะไม่มีการเปลี่ยนแปลง แต่ถ้าหลังคาเปิดอยู่มอเตอร์ก็จะทำการหมุนหลังคาให้ปิดลง และจะทำการวนลูปรอรับค่าต่อไป

5.2 การทดลองที่ 2 ทดสอบการทำงานของเซ็นเซอร์แสง

วัตถุประสงค์

1. เพื่อเข้าใจหลักการทำงานของเซ็นเซอร์แสง
2. เพื่อเข้าใจการแปลงสัญญาณแอนาลอกเป็นสัญญาณดิจิตอล
3. เพื่อเข้าใจหลักการทำงานของมอเตอร์

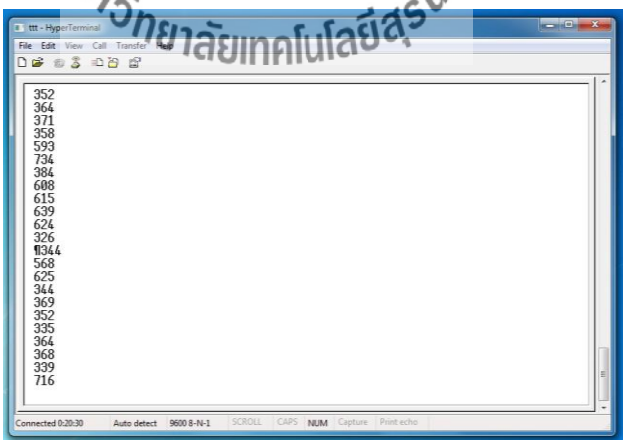
อุปกรณ์การทดลอง

1. ราวตากผ้าอัตโนมัติ
2. เครื่องคอมพิวเตอร์
3. สาย RS232

ขั้นตอนการทดลอง

1. นำสาย RS232 เชื่อมต่อกับบอร์ด AVR ATmega128 ที่อยู่ในราวตากผ้าอัตโนมัติ
2. เข้าไปที่ Hyper Terminal ทำการตั้งค่า bit per second ให้มีเท่ากับ 9600
3. สังเกตและบันทึกผลการทดลอง
4. ทำการปิดไฟหรือปิดตัวเซ็นเซอร์แสงไว้ สังเกตและบันทึกผลการทดลอง

ผลการทดลอง



รูปที่ 5.4 ค่า ADC ของเซ็นเซอร์แสง

ตารางที่ 5.2 ผลการทดลองเซ็นเซอร์แสง

ค่าที่แสดงใน Hyper Terminal(c1)=ADC	แรงดันอินพุต Vin(V)	การเปลี่ยนแปลงของหลังคา (กรณีหลังคาเปิดอยู่)
352	1.718	หลังคาเปิด
371	1.811	หลังคาเปิด
593	2.895	หลังคาเปิด
384	1.875	หลังคาเปิด
615	3.003	หลังคาเปิด
624	3.046	หลังคาเปิด
344	1.679	หลังคาเปิด
625	3.051	หลังคาเปิด
368	1.796	หลังคาเปิด
716	**3.496**	**หลังคาปิด**

วิเคราะห์ผลการทดลอง

จากการทดลองพบว่า ที่สถานะปกติกรณีที่ยังมีแสงสว่างอยู่หลังคาจะเปิด ค่าที่อ่านได้ใน Hyper Terminal เป็นค่าในการสุ่มที่น้อยกว่า 700 ซึ่งจะได้ค่าแรงดันอินพุต (Vin) น้อยกว่า 3.417V เมื่อไม่มีแสงสว่างพบว่า ค่าที่อ่านได้ใน Hyper Terminal มีค่ามากกว่า 700 จะได้ค่าแรงดันอินพุต (Vin) มากกว่า 3.417 V ซึ่งเป็นผลทำให้มอเตอร์ทำงาน หมุนหลังคาปิดลง สามารถหาค่า Vin ได้จากสมการที่ 5.1

สรุปการทดลอง

จากการกำหนดค่าในการเขียนโปรแกรม โดยกำหนดให้เมื่อค่าที่แสดงใน Hyper Terminal ซึ่งเป็นค่าของเลขฐานสองมีขนาด 10 บิต มีค่าน้อยกว่า 700 คือกรณีที่มีแสงสว่างอยู่จะทำให้หลังคาเปิดขึ้น โดยจะมีการเช็คสถานะก่อนว่าตอนนี้หลังคาเปิดอยู่หรือไม่ ถ้าเปิดอยู่ก็จะไม่มีการเปลี่ยนแปลง แต่ถ้าหลังคาปิดอยู่มอเตอร์ก็จะทำการหมุนหลังคาเปิดขึ้น หากมีการเช็คสถานะแล้วพบว่าไม่มีแสงสว่างค่าที่แสดงใน Hyper Terminal จะมีค่ามากกว่า 700 หลังคาก็จะปิดลง และรอรับค่าใหม่

5.3 <u>การทดลองที่ 3</u> ทดสอบการทำงานของวงจรเซ็นเซอร์น้ำฝนและเซ็นเซอร์แสง

วัตถุประสงค์

1. เพื่อเข้าใจการทำงานของวงจรเซ็นเซอร์น้ำฝนและเซ็นเซอร์แสง
2. เพื่อเข้าใจหลักการทำงานของมอเตอร์
3. เพื่อเข้าใจการแปลงสัญญาณแอนาลอกเป็นสัญญาณดิจิตอล

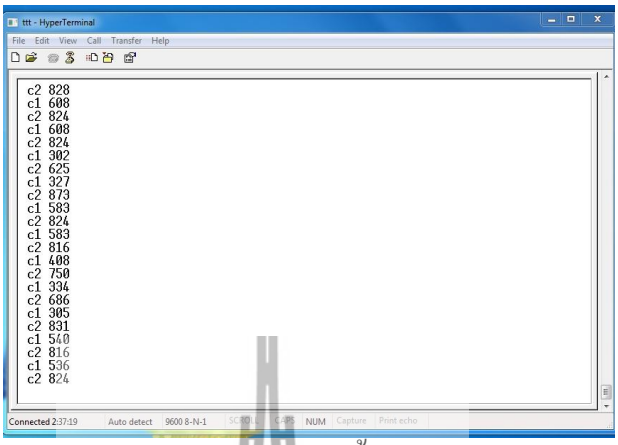
อุปกรณ์การทดลอง

1. ราวตากผ้าอัตโนมัติ
2. เครื่องคอมพิวเตอร์
3. สาย RS232

ขั้นตอนการทดลอง

1. นำสาย RS232 เชื่อมต่อกับบอร์ด AVR ATmega128 ที่อยู่ในราวตากผ้าอัตโนมัติ
2. เข้าไปที่ Hyper Terminal ทำการตั้งค่า bit per second ให้มีเท่ากับ 9600
3. สังเกตและบันทึกผลการทดลองกรณีที่ยัง ไม่มีฝนตกและยังสว่างอยู่
4. นำน้ำมาหยดใส่แผงเซ็นเซอร์น้ำฝน สังเกตและบันทึกผลการทดลอง
5. สังเกตการทำงานขณะที่ยังมีน้ำบนแผงเซ็นเซอร์ แล้วทำการปิดไฟ
6. เมื่อเช็ดแผงเซ็นเซอร์น้ำฝนให้แห้งขณะที่ยังมืดอยู่สังเกตการทำงาน
7. จากนั้นทำการเปิดไฟหรือมีแสงสว่างขณะที่ไม่มีฝนตกสังเกตและบันทึกผลการทดลอง

ผลการทดลอง



รูปที่ 5.5 ค่า ADC ของเซ็นเซอร์น้ำฝนและเซ็นเซอร์แสง

ตารางที่ 5.3 ผลการทดลองเซ็นเซอร์น้ำฝนและเซ็นเซอร์แสง

ค่าที่แสดงใน Hyper Terminal(c1)=ADC	ค่าที่แสดงใน Hyper Terminal(c2)=ADC	แรงดันอินพุต Vin(V)ของ c1	แรงดันอินพุต Vin(V)ของ c2	การเปลี่ยนแปลงของหลังคา
608	824	2.968	4.023	หลังคาเปิด
327	824	1.596	4.023	หลังคาเปิด
583	**625**	**2.846**	**3.051**	**หลังคาปิด**
583	873	2.846	4.262	หลังคาเปิด
408	824	1.992	4.023	หลังคาเปิด
334	816	1.630	3.984	หลังคาเปิด
305	750	1.489	3.662	หลังคาเปิด
540	**686**	**3.349**	**3.349**	**หลังคาปิด**
536	831	2.617	4.057	หลังคาเปิด
715	**846**	**3.491**	**4.130**	**หลังคาปิด**

วิเคราะห์ผลการทดลอง

จากการทดลองพบว่า ที่สถานะเริ่มต้นคือ มีแสงสว่างและฝนไม่ตกค่าที่อ่านได้จาก Hyper Terminal จะมี 2ค่าซึ่งค่า c1 จะมีค่าน้อยกว่า 700 และ c2 มีค่ามากกว่า 700 ทำให้หลังคายังคงเปิดอยู่ แต่เมื่อทำการทดลองโดยการหยดน้ำลงบนเซ็นเซอร์น้ำฝนแต่ยังคงสว่างอยู่พบว่า ค่า c1 ยังน้อยกว่า 700 เหมือนเดิม แต่ค่าของ c2 มีค่าเปลี่ยนไปจากเดิมคือมีค่าน้อยกว่า 700 ซึ่งทำให้หลังคาหมุนปิดลง และทำการทดลองโดยให้แผงเซ็นเซอร์ยังคงเปียกน้ำอยู่แล้วทำการปิดไฟหรือไม่ให้มีแสงสว่าง พบว่า ค่า c1 จะมีค่ามากกว่า 700 และ c2 มีค่าน้อยกว่า 700 หลังคายังคงปิดอยู่ แต่เมื่อทำการทดลองโดยเช็ดแผงเซ็นเซอร์น้ำฝนให้แห้งขณะที่ยังมืดอยู่พบว่า c1 จะยังคงมากกว่า 700 เหมือนเดิม แต่ c2 มีค่ามากกว่า 700 หลังคายังคงปิดอยู่ จากนั้นทำการเปิดไฟและตรวจสอบสถานะอีกครั้งพบว่า ค่า c1 มีค่าน้อยกว่า 700 และ c2 มากกว่า 700 ซึ่งจะทำให้มอเตอร์หมุนหลังคาเปิดขึ้น สามารถหาค่า Vin ได้จากสมการที่ 5.1

สรุปผลการทดลอง

จากการทดลองพบว่ามีทั้งหมด 4 กรณีดังนี้

1. กรณีที่ฝนไม่ตกและยังสว่างอยู่

ค่า c1 น้อยกว่า 700 และ c2 มากกว่า 700 หลังคาเปิด

2. กรณีที่ฝนตกและยังสว่างอยู่

ค่า c1 น้อยกว่า 700 และ c2 น้อยกว่า 700 หลังคาปิด

3. กรณีที่ฝนตกและไม่มีแสงสว่าง

ค่า c1 มากกว่า 700 และ c2 น้อยกว่า 700 หลังคาปิด

4. กรณีที่ฝนหยุดและไม่มีแสงสว่าง

ค่า c1 มากกว่า 700 และ c2 มากกว่า 700 หลังคาปิด

รูปการทำงาน

รูปที่ 5.6 ราวตากผ้าอัตโนมัติขณะที่เปิดหลังคา

รูปที่ 5.7 ราวตากผ้าอัตโนมัติขณะที่ปิดหลังคา

บทที่ 6
บทสรุปและข้อเสนอแนะ

6.1 บทนำ

โครงงานราวตากผ้าอัตโนมัติ เป็นโครงงานที่ใช้งานสำหรับการป้องกันการเปียกฝนของผ้าที่ตากไว้และจะทำการตากผ้าเฉพาะเวลากลางวันมีการทำงานโดยอัตโนมัติ ซึ่งสามารถนำไปใช้อำนวยความสะดวกในชีวิตประจำได้ จากการทดสอบการใช้งานราวตากผ้าอัตโนมัติ สามารถสรุปได้ดังนี้

1. ที่สถานะปกติคือ ไม่มีฝนตกแล้วยังสว่างอยู่ หลังคาก็จะเปิดขึ้นเพื่อทำการตากผ้า
2. เมื่อมีฝนตกขณะที่ยังสว่างอยู่ หลังคาก็จะทำการปิดลงเพื่อกันฝน
3. เมื่อมีฝนตกและไม่มีแสงสว่าง(มืด) หลังคาก็จะยังคงปิดอยู่
4. เมื่อฝนหยุดแต่ไม่มีแสงสว่าง หลังคาก็จะยังคงปิดอยู่

การทำงานดังกล่าวจะเป็นการทำงานแบบอัตโนมัติ โดยไมโครคอนโทรลเลอร์จะรับค่าจากตัวเซนเซอร์แสงและเซ็นเซอร์ตรวจจับน้ำฝนมาประมวลผลตลอดเวลา เพื่อคอยเปิดและปิดหลังคาตามสภาวะแสงและสภาวะอากาศในขณะใช้งาน

6.2 ปัญหาและอุปสรรค

1. ในขณะที่มอเตอร์กำลังทำงานหมุนหลังคาเปิดขึ้น หากเกิดฝนตกลงมาหลังคาก็ยังคงทำการหมุนเปิดให้ถึงสวิตช์ก่อน แล้วจึงรับค่าใหม่เพื่อให้หลังคาปิดลง แต่ก็ไม่เป็นอุปสรรคมากนักเพราะเมื่อรับค่าใหม่จะมีการทำงานที่รวดเร็ว
2. การหมุนลงของหลังคาอาจหมุนลงไม่พอดีกับขนาดที่ทำไว้ อันเนื่องจากความแรงของมอเตอร์ ซึ่งขึ้นอยู่กับแรงดันของแบตตารีอาจอ่อนลงเมื่อใช้งานเป็นเวลานาน แต่การใช้งานจริงอาจทำการเชื่อมต่อกับไฟบ้านได้ แต่ต้องทำการปรับอุปกรณ์ต่างๆเพื่อให้รองรับแรงดันที่เข้ามาได้
3. หากมีการใช้งานเป็นเวลานาน อาจทำให้เซ็นเซอร์น้ำฝนทำงานได้ไม่ดีพอเพราะตัวเซ็นเซอร์ที่ใช้เป็นแผ่น PCB เมื่อถูกน้ำนานๆจะมีการเสื่อมสภาพลง

6.3 สิ่งที่ได้รับจากการทำโครงงาน

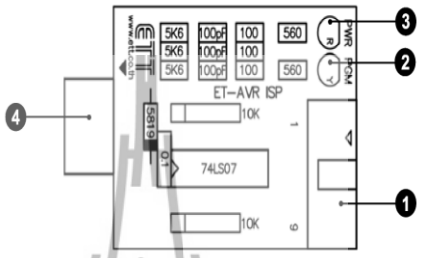
1. ได้ความรู้เกี่ยวกับหลักการทำงานของราวตากผ้าอัตโนมัติ
2. ได้รับความรู้เกี่ยวกับการใช้งานไมโครคอนโทรลเลอร์ในการควบคุมการทำงานของอุปกรณ์ต่าง ๆ และการประยุกต์ใช้งานไมโครคอนโทรลเลอร์สำหรับใช้งานอย่างอื่น
3. ได้รับความรู้เกี่ยวกับการทำลายวงจรต่างๆ
4. ได้รับความรู้เกี่ยวกับการทำงานของรีเลย์ การทำงานของมอเตอร์
5. ได้เรียนรู้การทำงานร่วมกับผู้อื่น
6. สามารถนำความรู้ที่ได้จากการศึกษาทฤษฎีมาปฏิบัติจริงได้
7. สามารถนำความรู้และประสบการณ์ที่ได้จากโครงงานไปประยุกต์ใช้ในชีวิตจริง

ภาคผนวก

ภาคผนวก ก

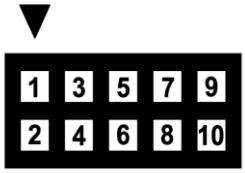
โครงสร้างของบอร์ด ET-AVR ISP

ดังรูปที่ 1



รูปที่ 1

- **หมายเลข 1** คือ พอร์ตสำหรับเชื่อมต่อกับ ET-CAP10P ของอีทีที เพื่อโปรแกรม Hex File ให้กับ MCU
- **หมายเลข 2** คือ LED PGM (สีเขียว) แสดงสถานะของการโปรแกรมหรือดาวน์โหลด Hex File ลง MCU
- **หมายเลข 3** คือ LED PWR (สีแดง) แสดงสถานะของไฟเลี้ยงบอร์ด
- **หมายเลข 4** คือ พอร์ตสำหรับเชื่อมต่อกับบอร์ด Target ซึ่งสามารถใช้โปรแกรม Hex File ให้กับบอร์ด ET-BASE AVR ATmega64/128 r3 โดยเสียบบอร์ด ET-AVR ISP เข้าที่ พอร์ต AVR ISP ซึ่งมีการจัดเรียงขาสัญญาณดังรูป

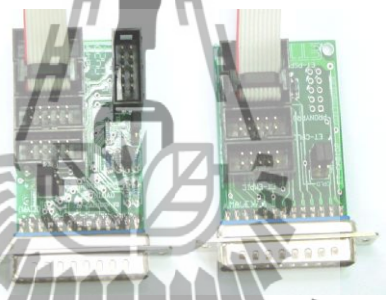


ตำแหน่งขา	ชื่อสัญญาณ
1	MOSI
2	VCC
3	ไม่ได้ใช้งาน
4,6,8,10	GND

5	RESET
7	SCK
9	MISO

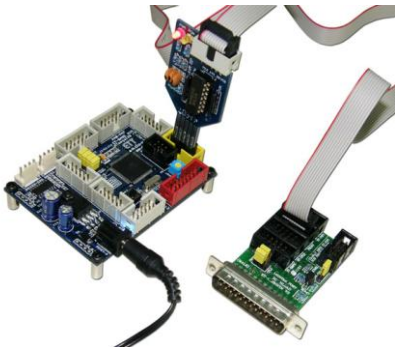
การเชื่อมต่ออุปกรณ์สำหรับโปรแกรม Hex File

การโปรแกรมโค้ด (Hex File) ให้กับ AVR MCU ต้องใช้งานร่วมกับ ET-CAB10PIN และโปรแกรม PonyProg2000 โดยต่อ ET-CAP10PIN เข้ากับพอร์ต Printer พร้อมทั้งเลือก Jumper สำหรับใช้งานกับโปรแกรม PonyProg2000 แล้วต่อสาย Download ที่ขั้วต่อ AVR ISP Download ของบอร์ด พร้อมทั้งจ่ายไฟเข้าบอร์ดให้เรียบร้อย ถ้ามีการต่ออุปกรณ์ภายนอกที่พอร์ต PB ให้ปลดออกก่อน โดยการเชื่อมต่อจะมีลักษณะดังรูปที่ 2 และรูปที่ 3



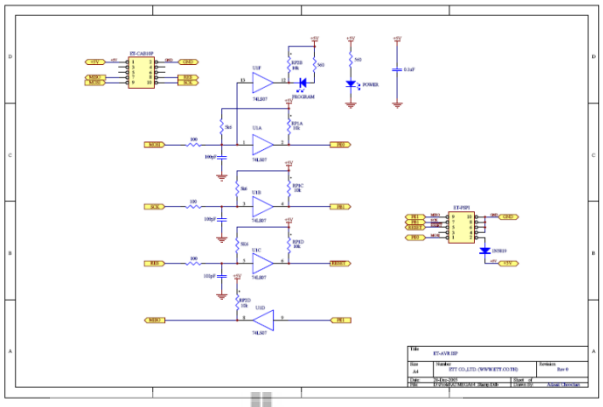
(ซ้าย) ET-CAP10P V2.0 (ขวา) ET-CAP10P V1.0

รูปที่ 2 การเลือก Jumper และการต่อสาย Download ของ ET-CAP10P เพื่อใช้กับ AVR

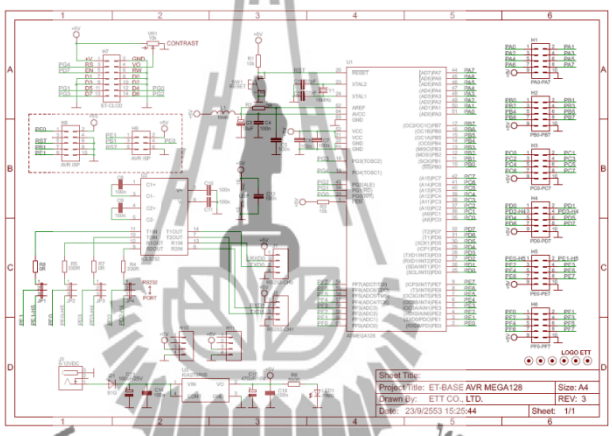


รูปที่ 3 การต่อ ET-AVR ISP เข้ากับ ET-BASE AVR ATmega64/128 r3 โดยการต่อบอร์ดทั้งสองเข้าด้วยกันนั้นจะให้สังเกตที่ตำแหน่งขา 1 จะต้องตรงกัน

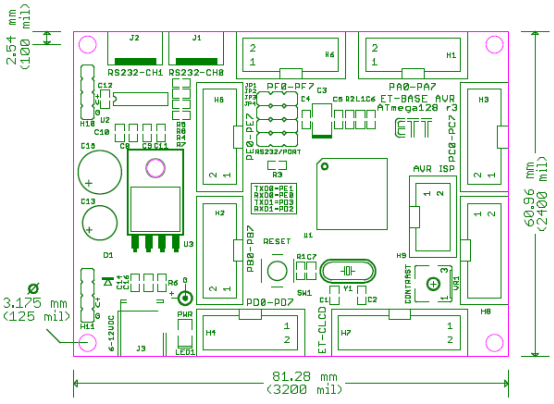
ลักษณะโดยทั่วไปของบอร์ดไมโครคอนโทรลเลอร์รุ่น ATmega128



Dimension



ET-BASE AVR MEGA128



ET-AVR ISP

ภาคผนวก ข

การเขียนโปรแกรม

ภาษา C พื้นฐานกับ AVR

ตัวอย่างที่ 1 แสดงโค้ดโปรแกรมภาษา C สำหรับไมโครคอนโทรลเลอร์ AVR โดยเลือกใช้เบอร์ ATmeag16 แสดงการติดของ LED 1 ดวงที่ขาพอร์ต PA0

โค้ดโปรแกรมตัวอย่างที่ 1 ไฟล์ EX0401.C

```
1 :
/*****************************************************************
2 : #include <avr/io.h>              // AVR decive-specific IO definitions
3
4 : /*****************************************************Main
Function */
5 : #int main(void)
6 : {
7 : DDRA=0xFF;              //PORT A Output all
8 : PORTA=0x00;             //Clear port
9 :
10 :                PORTA=0x01;          //Output high PA0
11 :
12 :                while   (1)          //Loop nothing
13 :                ;
14 :               return 0;
15 :         }
```


อธิบายโค้ดโปรแกรม EX0401.C

1. บรรทัดที่ 1

   คำสั่ง #include เป็นคอนโทรลไดเร็กตีฟในภาษา C ใช้ในการดึงไฟล์เข้ามาร่วมในการคอมไพล์โค้ดโปรแกรม ด้วยคำสั่ง #include จะดึงรายละเอียดของไฟล์ io.h ซึ่งได้กำหนดเบอร์ของ AVR ที่คอมไพล์เลอร์ WinAVR สนับสนุนการใช้งานได้ ขั้นตอนการกำหนดเบอร์ AVR ที่ต้องการใช้งานจะถูกกำหนดตอนสร้างไฟล์โปรเจ คจากตัวอย่างการใช้งาน ATmega16 โดยที่ไฟล์ io.h จะอยู่ภายในโฟลเดอร์ AVR

2. บรรทัดที่ 5

   ฟังก์ชัน main() ฟังก์ชันเริ่มต้นการทำงานของโปรแกรม EX0301

3. บรรทัดที่ 7

   รีจิสเตอร์ DDRA รีจิสเตอร์กำหนดทิศทางการทำงานของพอร์ต A เป็นอินพุตหรือเอาต์พุต จากตัวอย่างกำหนดให้ขาพอร์ต A เป็นเอาต์พุตทุกขาพอร์ต (PA0 ถึง PA7) โดยกำหนดค่า 1 ให้กับรีจิสเตอร์ DDRA บิตที่ตรงกับตำแหน่งบิตของขาพอร์ต A

4. บรรทัดที่ 8

   เคลียร์พอร์ต A ทั้งพอร์ตโดยส่งค่า 0x00 ไปที่ขาพอร์ต A

5. บรรทัดที่ 10

   กำหนดพอร์ต A บิต 0(PA0) โดยส่งค่า 1 ไปที่ขาพอร์ต A ด้วยค่า 0x01 (ฐานสิบหก) หรือ 00000001 (เลขฐานสอง)

6. บรรทัดที่ 12

   วนลูปโปรแกรมโดยไม่ทำงานใดๆ หรือเป็นการวนทำซ้ำอยู่กับที่ ด้วยคำสั่ง while(1) ด้วยเงื่อนไขที่เป็นจริงตลอดการทำงานจากโค้ดโปรแกรมตัวอย่าง จะเห็นได้ว่าการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ คือการกำหนดค่าต่างๆ ให้กับรีจิสเตอร์ที่เกี่ยวข้องกับการใช้งาน ในตัวอย่างเป็นการเปิดการใช้งานพอร์ต A ทั้งหมด โดยมีรีจิสเตอร์ DDRA และ PORTA ที่เกี่ยวข้องเท่านั้นที่ใช้ในการควบคุมขาพอร์ต

ฟังก์ชันอินเตอร์รัปต์

ฟังก์ชันอินเตอร์รัปต์ เป็นฟังก์ชันเฉพาะที่ถูกเรียกใช้งานโดยอัตโนมัติ เมื่อเกิดเหตุการณ์ที่เกี่ยวข้องกับอินเตอร์รัปต์ที่เกี่ยวข้องกับโมดูลที่ใช้งาน โดยจะกระโดดมาทำงานในส่วนของฟังก์ชันที่ได้มีการเปิดการใช้งานอินเตอร์รัปต์ไว้หรือมีการเอ็นเอเบิลอินเตอร์รัปต์ดังกล่าว อินเตอร์รัปแต่ละตัวหรือแต่ละประเภทจะตรงกับการใช้งานของโมดูลนั้น โดยภายในโมดูลดังกล่าวสามารถกำหนดอินเตอร์รัปต์ได้ ซึ่งเป็นอินเตอร์รัปต์ที่เกิดจากสัญญาณภายในของโมดูลเอง เรียกว่าอินเตอร์รัปต์ภายใน นอกจากนี้ยังมีอินเตอร์รัปต์เนื่องจากสัญญาณภายนอก เป็นอินเตอร์รัปต์ที่เกิดขึ้นเมื่อมีสัญญาณเข้ามาที่ขาที่เกี่ยวข้องกับอินเตอร์รัปต์เนื่องจากสัญญาณภายนอก การเขียนฟังก์ชันอินเตอร์รัปต์สำหรับภาษา C ใน WinAVR จะมีรูปแบบดังนี้

```
ISR(หมายเลขอินเตอร์รัปต์ที่ต้องการใช้งาน)
{
        //โค้ดโปรแกรมอินเตอร์รัปต์
}
```

```
ISR(TIMER0_OVF_vect)                    //กำหนดฟังก์ชันอินเตอร์รัปต์เนื่องจากไทเมอร์0 โอ
เวอร์โฟลว์
{
        Count++;                        //เพิ่มค่า count ทีละหนึ่ง
        If(count==6){                   //ถ้า count เท่ากับ 6
                Sbi(PORTA,7);           //เซต บิตที่ 7 ของ PORTA
        }
        If(count==12){                  //ถ้า count เท่ากับ 12
                Cbi(PORTA,7);           //เคลียร์บิตที่ 7 ของ PORTA
                Count=0;                //เคลียร์ count เป็นศูนย์
        }
}
```

โปรแกรมอินเตอร์รัปต์ไทเมอร์ 0

โปรแกรมตัวอย่างที่ 2 แสดงการใช้งานอินเตอร์รัปต์ฟังก์ชันของไทเมอร์ 0 แสดงการติดดับของ LED 1 ดวงที่ขาพอร์ต PA7 โดยการติดดับเนื่องจากการนับของไทเมอร์ 0 เกิดโอเวอร์โฟลว์ขึ้น และในฟังก์ชันหลักยังคงแสดงการติดดับของ LED 1 ดวงที่ขาพอร์ต PA0 ด้วยฟังก์ชันวนลูปหน่วงเวลา

โค้ดโปรแกรมตัวอย่างที่ 2 ไฟล์ EX0303.C


```
1 :
/*****************************************************************
****
2 : #include <avr/io.h>                  // AVR device-specific IO definitions       */
3 : #include <avr/interrupt.h>           //Interupt  Survice routine
4 : #include <compat/depreced.h>         //use sbi, cbi
5 : unsigned  int  count=0;               // counter interrupt
6 :
7 :
/*****************************************************************
**
```

```
8 : void      loop_delay(unsigned long int count1)
9 : {
10 :                  long i ;
11 :                  while (count1-- > 0)  {  // decrease Counter
12 :                        for(i=1;i<100;i++)
13 :                        ;
14 :                  }
15 : }
16 :
/*****************************************************************
***
17 :          int  main (void)
18 : {
19 :                  //PORT A Output all
20 :                  DDRA=(1<<DDA7)|(1<<DDA6)| (1<<DDA5)|(1<<DDA4)|
21 :                        (1<<DDA3)|(1<<DDA2)| (1<<DDA1)|(1<<DDA0);
22 :
23 :                  //Set Timer Prescaler (CLKio/1024)
24 :                  TCCR0=(1<<CS02)| (1<<CS01)| (1<<CS00);

25 :                  //Set Interupt Timer 0
26 :                  TIMSK=  (1<<TOIE0);           // T/C0 Overflow  interrupt
Enable
27 :                  TIFR    =  (1<<TOV0);         // Set T/C0 overflow flag
28 :                  SREG   = 0x80;                // Set I-bit Global interrupt
29 :
30 :                  PORTA=0x00;                   //clear port
31 :
32 :                  while (1) {                   //loop forever
33 :                        sbi(PORTA,0);           // set bit PA0
```

```
34 :                    loop_delay(100);        // delay
35 :                    cbi(PORTA,0);           //clear bit PA0
36 :                    loop_delay(100);        //delay
37 :        }
38 :        return 0;
39 : }
40 :
/*****************************************************************
41 :        //Timer 0 Interupt
42 :        ISR (TIMER0_OVF_vect)
43 :        {
44 :                count++;
45 :                if (count==6) {
46 :                        sbi (PORTA,7);          //Set bit PA7
47 :                }
48 :                if (count==12) {
49 :                        cbi (PORTA,7);          // Clear bit PA7
50 :                        count = 0;              // Clear counter
51 :                }
52 : }
```


1. บรรทัดที่ 3 และ 4

กำหนดเฮดเดอร์ไฟล์ interrupt.h เพื่อเรียกใช้งานมาโคร (Macro) อินเตอร์รัปต์ฟังก์ชัน ISR(vector) และ deprecated.h เพื่อเรียกใช้งานฟังก์ชัน sbi () ในการเซตบิตและ cbi () ในการเคลียร์บิต

2. บรรทัดที่ 5

ตัวแปร count ใช้ในการนับการเกิดโอเวอร์โฟลว์ในไทเมอร์ 0 ควบคุมการติดดับของ LED ในอินเตอร์รัปต์ไทเมอร์ 0

3. บรรทัดที่ 8

ฟังก์ชันหน่วงเวลาด้วยการวนลูปการทำงานของโปรแกรม

4. บรรทัดที่ 17

ฟังก์ชัน main() โปรแกรมหลักติดตั้งการใช้งานพอร์ต A กำหนดปรีสเกลเลอร์สำหรับไทเมอร์ 0 พร้อมกับติดตั้งการใช้งานอินเตอร์รัปต์เนื่องจากไทเมอร์ 0 และวนลูปแสดงการติดดับของ LED 1 ดวงที่ขาพอร์ต PA 0

5. บรรทัดที่ 42

ฟังก์ชันอินเตอร์รัปต์ไทเมอร์ 0 โอเอร์โฟลว์ การทำงานของฟังก์ชันจะรอจนไทเมอร์ 0 นับค่าจนเกิดโอเวอร์โฟลว์ ในแต่ละครั้งตัวแปร count จะเพิ่มค่าทีละ 1 เมื่อเท่ากับ 6 จะแสดงการติดของ LED ที่ขาพอร์ต PA7 และเมื่อนับได้ 12 จะแสดงการดับของ LED พร้อมกับเคลียร์ค่า count เพื่อให้เริ่มต้นนับใหม่เมื่อเกิดโอเวอร์โฟลว์อีกครั้ง

จากตัวอย่างที่ 2 การใช้งานฟังก์ชันอินเตอร์รัปต์ ต้องกำหนดโหมดการทำงานของโมดูลนั้นก่อนจาตัวอย่างเป็นการใช้ไทเมอร์ 0 และกำหนดโหมดการทำงานในบรรทัดที่ 24 กำหนดการใช้งานฟังก์ชันอินเตอร์รัปต์ โดยการเซตบิตในรีจิสเตอร์ที่เกี่ยวข้องกับอินเตอร์รัปต์ที่ต้องการใช้งาน แสดงดังบรรทัดที่ 26 ถึง 28 เขียนฟังก์ชันต์อินเตอร์รัปต์ด้วยมาโคร ISR(Interupt_Vecter) ในบรรทัดที่ 42 โปรแกรมทำงานภายในอินเตอร์รัปต์อยู่ระหว่างบรรทัดที่ 43 ถึง 52

ชนิดข้อมูลของตัวแปร

WinAVR C คอมไพเลอร์ แบ่งชนิดข้อมูลที่ใช้สร้างตัวแปร แบ่งออกเป็นแต่ละชนิดดังตารางที่ 1 ดังนี้

ชนิดข้อมูล	ชื่อที่ใช้แทนชนิดข้อมูลได้	ขนาด
char	-	8-bit signed type
Signed char	Int8_t	8-bit signed type
Unsigned char	Uint8_t	8-bit unsigned type
int	-	16-bit signed type
Signed int	Int16_t	16-bit signed type
Unsigned int	Uint16_t	16-bit unsigned type
Long	-	32-bit signed type
Signed long int	Int32_t	32-bit signed type
Unsigned long int	Uint32_t	32-bit unsigned type

Signed long long int	Int64_t	64-bit signed type
Unsigned long long int	Uint64_t	64-bit unsigned type
float	-	64-bitfloat point number type

โดยที่ชนิดข้อมูล char ใช้กำหนดชนิดข้อมูลแบบอักขระ(Character) และจำนวนตัวเลขก็ได้ เช่น

Char ch;

Ch=36; //กำหนดค่าเป็นจำนวนเต็มแบบตัวเลข

Ch='A'; //กำหนดค่าเป็นตัวอักขระ

คำสั่งควบคุมการทำงานพื้นฐาน

ภาษา C แบ่งคำสั่งควบคุมการทำงานพื้นฐานออกเป็น 3 ส่วน คือ

1. คำสั่งควบคุมการทำงานแบบมีเงื่อนไข ประกอบไปด้วยคำสั่ง if(),if() …else และคำสั่ง switch()

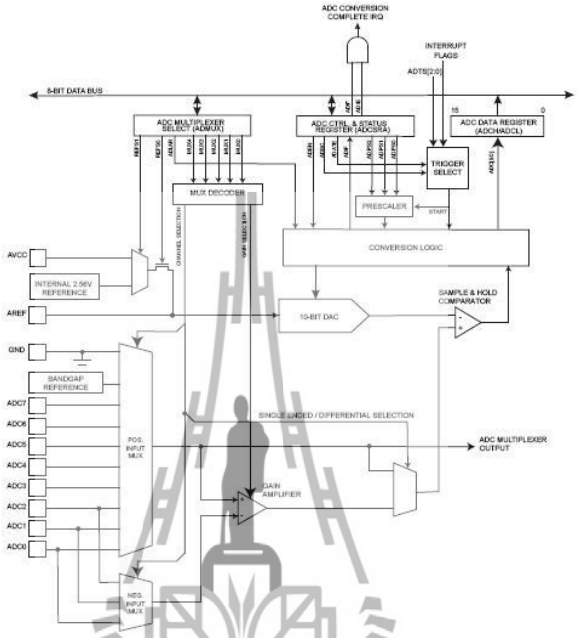
2. คำสั่งควบคุมการทำงานแบบทำซ้ำหรือวนลูป ประกอบไปด้วยคำสั่ง for(),while() และคำสั่ง do…while()

3. คำสั่งควบคุมที่ใช้งานร่วมกับคำสั่งลูปและเงื่อนไข ประกอบไปด้วย goto,break และ continue

การใช้งานโมดูลแปลงสัญญาณแอนาลอกเป็นดิจิตอล

ไมโครคอนโทรลเลอร์ AVR เบอร์ ATmega128 มีโมดูลแปลงสัญญาณแอนาลอกเป็นดิจิตอล หรือ ADC(Analog to Digital Converter) ความละเอียดขนาด 10 บิต (10-bit Resolution) ที่แรงดัน +5V หมายถึงเมื่อแปลงสัญญาณดิจิตอลแล้วจะได้ค่าตัวเลขอยู่ระหว่าง 0-1024 โดยมีรูปแบบการแปลงสัญญาณแอนาลอกเป็นดิจิตอลแบบซัคเซสซีฟ แอพพร็อกซิเมชั่น (Successive Approximation ADC) คือการแปลงแบบประมาณค่า โดยการสุ่มค่าดิจิตอลแล้วแปลงเป็นแรงดันแอนาลอกภายในโมดูล เพื่อใช้เปรียบเทียบกับแรงดันแอนาลอกด้านอินพุต เมื่อเปรียบเทียบได้ค่าแรงดันเท่ากัน โมดูล ADC จะให้ผลลัพธ์ออกมาเป็นค่าดิจิตอล ซึ่งการใช้วิธีการนี้เป็นที่นิยมเพราะมีความเที่ยงตรงสูงและทำงานได้อย่างรวดเร็ว

โมดูล ADC จะมีจำนวน 8 ช่องอินพุตสัญญาณใช้หลักการมัลติเพล็กซ์ (Multiplex) เพื่อเลือกทำงานในแต่ละช่อง กำหนดไว้ที่ขาพอร์ต A โดยมีแรงดันอินพุตระหว่าง 0 V (GND) ถึง VCC (แรงดันอินพุตที่ขา VCC ของไมโครคอนโทรลเลอร์ AVR) ผ่านวงจรสุ่มและเก็บค่า (Sample and Hold) วงจร ADC ดังรูปที่ 1



รีจิสเตอร์ที่เกี่ยวข้องกับโมดูลเปรียบเทียบแรงดันแอนาลอก

1. รีจิสเตอร์ ADMUX (ADC Multiplexer Selection Register)

รีจิสเตอร์กำหนดแรงดันอ้างอิง (Voltage Reference) รูปแบบการเก็บข้อมูลและการกำหนดอินพุตแอนาลอกอ้างอิงด้านบวกและลบ

2. รีจิสเตอร์ ADCSRA(ADC Control and Status Register)

รีจิสเตอร์กำหนดและแสดงสถานการณ์ทำงานของโมดูล ADC

3. รีจิสเตอร์ ADCL และ ADCH(The ADC Data Register)

รีจิสเตอร์เก็บข้อมูลที่ได้จากการแปลงสัญญาณแอนาลอกเป็นดิจิตอล

4. รีจิสเตอร์ SFIOR(Special Function IO Register)

รีจิสเตอร์กำหนดการกระตุ้นจากแหล่งสัญญาณภายนอกให้กับโมดูล ADC

รายละเอียดของรีจิสเตอร์ที่เกี่ยวข้องกับการใช้งานโมดูลมีดังนี้

1. รีจิสเตอร์ ADMUX (ADC Multiplexer Selection Register)

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	REFS 1	REFS 0	ADLA R	MUX 4	MUX 3	MUX 2	MUX 1	MUX 0
Read/Writ e	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

บิตที่ 7:6: บิต REFS1:0 (Reference Selection Bits)

บิตกำหนดแหล่งแรงดันอ้างอิงสำหรับโมดูล ADC การกำหนดรายละเอียดตามตารางที่ 4.1 ดังนี้

ตารางที่ 2 แรงดันอ้างอิงสำหรับโมดูล ADC

REFS1	REFS0	แหล่งแรงดันอ้างอิง
0	0	ปิดการใช้งานแรงดันอ้างอิงภายใน(AREF off)
0	1	ใช้แรงดัน AVCC กับตัวเก็บประจุภายนอกที่ขา AREF
1	0	สงวนไว้
1	1	ใช้แรงดันอ้างอิงภายในที่ 2.56V กับตัวเก็บประจุภายนอกที่ขา AREF

บิตที่ 5: บิต ADLAR (ADC Left Adjust Result)

บิตกำหนดรูปแบบการเก็บข้อมูลในรีจิสเตอร์ ADC Data ขนาด 16 บิต มี 2 รูปแบบคือเก็บชิดบิตสูงสุด (MSB bits) หรือเก็บชิดบิตที่ต่ำสุด (LSB bit) ดูเพิ่มเติมในรีจีสเตอร์ ADC Data Register (ADCH, ADCL

บิตที่ 4:0 บิต MUX4:0 (Analog and Gain Selection Bits)

บิตที่กำหนดขาแรงดันอะนาลอกอินพุตด้านบวกและด้านลบ รายละเอียดแสดงดังตารางที่ 4.2

ตารางที่ 3 การเลือกช่องสัญญาณอินพุตและตัวคูณอัตราขยาย

MUX4..0	อินพุตเดี่ยว	ความแตกต่างของอินพุตด้านบวก	ความแตกต่างของอินพุตด้านลบ	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	N/A	ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x

11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x
11110	1.22V (VBG)			
11111	0 v (GND)			

2. รีจีสเตอร์ ADCSRA (ADC Control and Status Register)

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

บิตที่ 7 : บิต ADEN (ADC Enable)

เซตบิต ADEN เป็น “1”เพื่อเปิดการใช้งานโมดูล ADC

บิตที่ 6 : บิต ADSC (ADC Start Conversion)

เซตบิต ADSC เป็น “1”เพื่อกำหนดให้โมดูลเริ่มต้นแปลงสัญญาณอะนาลอกเป็นดิจิตอล เมื่อแปลงเสร็จสมบูรณ์ บิต ADSC จะถูกเซตเป็น “0”จะไม่มีผลใดๆกับโมดูล

บิตที่ 5 : บิต ADATE (ADC Auto Trigger Enable)

เซตบิต ADATE เป็น “1”เพื่อเปิดการกระตุ้น (Trigger) สัญญาณอัตโนมัติ โดยแหล่งสัญญาณการกระตุ้น กำหนดในบิต ADTS ที่รีจิสเตอร์ SFIOR

บิตที่ 4 : บิต ADIF (ADC Interrupt Flag)

บิต ADIF จะถูกเซตเป็น “1”เมื่อโมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอลเสร็จสมบูรณ์ และข้อมูลได้ถูกเขียนไปรีจิสเตอร์ ADCD (ADCH, ADCL)แล้ว หากมีการเปิดใช้งานอินเตอร์รัปต์เนื่องจากโมดูล ADC และเปิดใช้งานอินเตอร์รัปต์โดยรวมจะส่งผลให้เกิดอินเตอร์รัปต์ขึ้น

บิตที่ 3 : บิต ADIE (ADC Interrupt Enable)

เซตบิต ADIE เป็น “1”เพื่อเปิดการใช้งานอินเตอร์รัปต์เนื่องจากโมดูล ADC (ต้องเปิดอินเตอร์รัปต์โดยรวมด้วย)

บิตที่ 2:0 : บิต ADPS2:0 (ADC Prescaler Select Bits)

บิตกำหนดปรีสเกลเลอร์สำหรับใช้ในการหารสัญญาณนาฬิกาสำหรับโมดูล ADC รายละเอียดแสดงดังตารางที่ 4

ตารางที่ 4 ปรีสเกลเลอร์สำหรับโมดูล ADC

ADPS2	ADPS1	ADPS0	ปรีสเกลเลอร์หารสัญญาณความถี่ (XTAL) และสัญญาณนาฬิกาโมดูลADC
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

3. รีจิสเตอร์ ADCL และ ADCH (The ADC Data Register)

เมื่อ ADLAR=0

บิตที่	15	14	13	12	11	10	9	8
ชื่อบิต	-	-	-	-	-	-	ADC9	ADC8
ชื่อบิต	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC07
บิตที่	7	6	5	4	3	2	1	0
Read/Write	R R	R R	R R	R R	R R	R R	R R	R R
ค่าเริ่มต้น	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0

บิตที่ 9:0 : บิต ADC9:0 (ADC Data Register)

ผลลัพธ์ของข้อมูลที่ได้จากโมดูล ADC จะเป็นทางขวาสุดของข้อมูล โดยการเซตบิต (ADC Left Adjust Result) เป็น “0” ในรีจิสเตอร์ ADMUX

เมื่อ ADLAR = 1

บิตที่	15	14	13	12	11	10	9	8
ชื่อบิต	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2

ชื่อบิต	ADC1	ADC0	-	-	-	-	-	-
บิตที่	7	6	5	4	3	2	1	0
Read/Write	R R	R R	R R	R R	R R	R R	R R	R R
ค่าเริ่มต้น	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0

บิตที่ 15:6 : บิต ADC9:0 (ADC Data Register)

ผลลัพธ์ของข้อมูลที่ได้จากโมดูล ADC จะเก็บทางซ้ายสุดของข้อมูล โดยการเซตบิต (ADC Left Adjust Result) เป็น “1” ในรีจิสเตอร์ ADMUX

4. รีจิสเตอร์ SFIOR (Special Function IO Register)

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

บิตที่ 7:5 : บิต ADTS2:0 (ADC Auto Trigger Source)

บิตกำหนดการกระตุ้นโมดูล ADC จากแหล่งสัญญาณภายนอกโดยขึ้นอยู่กับบิต ADATE ในรีจิสเตอร์ ADCSRA หาก ADATE ถูกเซตเป็น “1”การกำหนดบิตกระตุ้นการทำงานของโมดูล ADC จะขึ้นอยู่กับสัญญาณจากภายนอกตามบิต ADTS2:0 หาก ADCSRA ถูกเซตเป็น “0” บิต ADTS2:0 จะไม่มีผลใดๆกับโมดูล ADC การกำหนดแหล่งกระตุ้นสัญญาณจากภายนอก กำหนดได้ดังตารางที่ 5

ตารางที่ 5 แหล่งกระตุ้นสัญญาณอัตโนมัติของโมดูล ADC

ADTS2	ADTS1	ADTS0	แหล่งกระตุ้นสัญญาณอัตโนมัติ
0	0	0	โหมดทำงานอิสระ
0	0	1	เปรียบเทียบแรงดันอะนาลอก
0	1	0	อินเตอร์รัปต์เนื่องจากสัญญาณภายนอก ช่องที่0
0	1	1	โมดูลเปรียบเทียบสัญญาณของไทเมอร์/เคาน์เตอร์0
1	0	0	ไทเมอร์/เคาน์เตอร์0 โอเวอร์โฟลว์
1	0	1	โมดูลเปรียบเทียบสัญญาณของไทเมอร์/เคาน์เตอร์1

1	1	0	ไทเมอร์/เคาน์เตอร์1 โอเวอร์โฟลว์
1	1	1	โมดูลตรวจจับสัญญาณอินพุตของไทเมอร์/เคาน์เตอร์

บิตที่ 4 : บิต Res(Reserved Bit)

บิตนี้สงวนไว้ ไม่ได้ใช้งาน อ่านค่าได้เป็น “0”

การควบคุมมอเตอร์ (Motor control)

มอเตอร์ที่นิยมนำมาใช้ในงานคอนโทรลปัจจุบันแบ่งออกได้เป็น 3 ประเภทคือ หนึ่ง สเต็ปเปอร์มอเตอร์ (Stepper Motor) สอง เซอร์โวมอเตอร์ (Servo Motor) และสาม มอเตอร์ไฟฟ้ากระแสตรง (DC Motor)

สเต็ปเปอร์มอเตอร์

สเต็ปเปอร์มอเตอร์ เป็นมอเตอร์ที่หมุนตามจำนวนองศาหรือจำนวนรอบที่ต้องการ ทั้งนี้ความละเอียดของการหมุนจะขึ้นอยู่กับจำนวนองศาต่อหนึ่งจังหวะการหมุน เนื่องจากการหมุนของมอเตอร์ชนิดนี้มีการหมุนเป็นจังหวะ จึงเรียกว่า สเต็ปเปอร์มอเตอร์ และที่นิยมใช้จะเป็นชนิดที่เรียกว่า สเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์(uni-polar stepper motor)

สเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์ จะมีการพันขดลวด 2 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ แต่ละขดแบ่งเป็น 2 เฟส ในมอเตอร์ 1 ตัว จึงมี 4 เฟส คือ เฟส 1,2,3 และ เฟส 4 ในการต่อสายไฟเลี้ยงของขดลวดมาต่อรวมกันเป็นสายเดียวไว้

การทำงานของสเต็ปเปอร์มอเตอร์

การทำงานของสเต็ปเปอร์มอเตอร์จะทำงานได้โดยการกระตุ้นขดลวดในแต่ละขดบนสเตเตอร์เพื่อให้มอเตอร์หมุนและในการกระตุ้นขดลวดนี้ต้องป้อนกระแสไฟแบบต่อเนื่อง (Sequacious) ในรูปแบบที่ถูกต้อง จึงจะสามารถขับสเต็ปเปอร์มอเตอร์ให้หมุนได้ตามต้องการ สำหรับการกระตุ้นการหมุนของสเต็ปเปอร์มอเตอร์แบ่งออกเป็น 3 รูปแบบคือ

1. ฟูลสเต็ปแบบ 1 เฟส (Full step 1 phase)
2. ฟูลสเต็ปแบบ 2 เฟส (Full step 2 phase)
3. ฮาล์ฟสเต็ป (half step)

การขับสเต็ปเปอร์มอเตอร์ฟูลสเต็ปแบบ 1 เฟส

การขับสเต็ปเปอร์มอเตอร์ฟูลสเต็ป แบบหนึ่งสเต็ป เป็นการขับสเต็ปมอเตอร์แบบที่ง่ายที่สุด โดยการป้อนกระแสไฟกระตุ้นการทำงานของขดลวดทีละขด เรียงลำดับกันไป แบบนี้จะกินกระแสไฟน้อยที่สุด และให้แรงบิตน้อย ลักษณะการขับสเต็ปเปอร์มอเตอร์แบบหนึ่งเฟส แสดงดังตารางที่ 6

ตารางที่ 6 แสดงการทำงานของขดลวดแบบฟูลสเต็ป 1 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ON	-	-	-
2	-	ON	-	-
3	-	-	ON	-
4	-	-	-	ON

หมายเหตุ เครื่องหมาย “-“ แสดงสภาวะ OFF

การเขียนโค้ดควบคุมการทำงานแบบฟูลสเต็ป 1 เฟส

- หมุนในทิศทางตามเข็มนาฬิกา

```
void  FW_step(int step, unsigned long dl)
{
     STEP_M_POUT = 0x00;                    //clear port
     for (;step>0;step--) {
       MOTOR_HIGHT(M_PINO);  delay_ms(dl);
         MOTOR_LOW(M_PINO);

       MOTOR_HIGHT(M_PIN1);  delay_ms(dl);
         MOTOR_LOW(M_PIN1);

       MOTOR_HIGHT(M_PIN2);  delay_ms(dl);
         MOTOR_LOW(M_PIN2);

                MOTOR_HIGHT(M_PIN2);  delay_ms(dl);
       MOTOR_LOW(M_PIN2);
```

- หมุนในตัวทิศทางเข็มนาฬิกา

```
void  RW_step(int step, unsigned long dl)
```

```
{
    STEP_M_POUT = 0x00;                         //clear port
    for (;step>0;step--)  {
      MOTOR_HIGHT(M_PIN3);  delay_ms(dl);
        MOTOR_LOW(M_PIN3);

      MOTOR_HIGHT(M_PIN2);  delay_ms(dl);
        MOTOR_LOW(M_PIN2);

      MOTOR_HIGHT(M_PIN1);  delay_ms(dl);
        MOTOR_LOW(M_PIN1);

              MOTOR_HIGHT(M_PIN0);  delay_ms(dl);
        MOTOR_LOW(M_PIN0);
    }
}
```

การขับสเต็ปเปอร์มอเตอร์ฟูลสเต็ปแบบ 2 เฟส

การขับสเต็ปเปอร์มอเตอร์แบบฟูลสเต็ปสองเฟส เป็นการขับมอเตอร์ที่จ่ายกระแสไฟให้กับขดลวด 2 ขดพร้อมกัน จะกินกระแสไฟมากกว่าแบบหนึ่งเฟส แต่ก็ให้แรงบิดที่มากขึ้น ลักษณะการขับสเต็ปมอเตอร์แบบสองเฟส แสดงดังตารางที่ 7

ตารางที่ 7 แสดงการทำงานของขดลวดแบบฟูลสเต็ป 2 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ON	ON	-	-
2	-	ON	ON	-
3	-	-	ON	ON
4	ON	-	-	ON

หมายเหตุ เครื่องหมาย “-” แสดงสภาวะ OFF

การขับสเต็ปเปอร์มอเตอร์แบบฮาล์ฟสเต็ป

การขับสเต็ปเปอร์มอเตอร์แบบฮาล์ฟสเต็ปหรือครึ่งจังหวะ เป็นการนำรูปแบบการขับสเต็ปมอเตอร์สองแบบมารวมกัน ทำให้จำนวนสเต็ปของมอเตอร์เพิ่มขึ้นมาอีกหนึ่งเท่าตัว แสดงดังตารางที่ 8

ตารางที่ 8 แสดงการทำงานของขดลวดแบบฮาล์ฟสเต็ป

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ON	-	-	-
2	ON	ON	-	-
3	-	ON	-	-
4	-	ON	ON	-
5	-	-	ON	-
6	-	-	ON	ON
7	-	-	-	ON
8	ON	-	-	ON

หมายเหตุ เครื่องหมาย “-” แสดงสภาวะ OFF

เซอร์โวมอเตอร์

เซอร์โวมอเตอร์เป็นมอเตอร์ที่ประกอบไปด้วยชุดเกียร์ (Gear) มอเตอร์กระแสตรง (DC Motor) และส่วนควบคุมอิเล็กทรอนิกส์ ที่รวมอยู่ภายในตัวมอเตอร์ เซอร์โวมอเตอร์จะทำงานได้ด้วยสัญญาณพัลส์ที่มีคาบความกว้างอยู่ระหว่าง 1 มิลลิวินาที ถึง 2 มิลลิวินาที (พัลส์บวกหรือลอจิก 1) โดยส่งพัลส์ดังกล่าวห่างกันเป็นระยะเวลา 20 มิลลิวินาที (พัลส์ลบหรือลอจิก 0) การส่งสัญญาณพัลส์มีผลให้มอเตอร์หมุน โดยทิศทางการหมุนนั้นจะขึ้นอยู่กับความกว้างของพัลส์บวก

การทำงานของเซอร์โวมอเตอร์

การหมุนของเซอร์โวมอเตอร์จะถูกควบคุมด้วยพัลส์หรือลอจิก 1 (5V) เป็นระยะเวลาที่กำหนดและหน่วงเวลาหรือส่งลอจิก 0 เป็นระยะเวลาคงที่ 20 มิลลิวินาที สลับกันไป ส่งผลให้สามารถควบคุมการหมุนของเซอร์โวมอเตอร์ได้ โดยทิศทางการหมุนเป็นดังนี้

- เซอร์โวมอเตอร์จะหมุนไปทางขวา (ตามเข็มนาฬิกา) เมื่อพัลส์บวกมีความกว้าง 1 มิลลิวินาที (1ms)และพัลส์ลบ 20 มิลลิวินาที

- เซอร์โวมอเตอร์จะหมุนไปทางซ้าย (ทวนเข็มนาฬิกา) เมื่อพัลส์บวกมีความกว้าง 2 มิลลิวินาที (2ms)และพัลส์ลบ 20 มิลลิวินาที
- เซอร์โวมอเตอร์จะอยู่กึ่งกลาง (หยุดหมุน) เมื่อพัลส์บวกมีความกว้าง 1.5 มิลลิวินาที และพัลส์ลบ 20 มิลลิวินาที (หากเซอร์โวมอเตอร์ไม่หยุดหมุน จะต้องปรับค่าความต้านทานของเซอร์โวมอเตอร์ใหม่ โดยที่ช่องปรับจะอยู่ใกล้กับสายไฟของเซอร์โวมอเตอร์)

ความเร็วในการหมุนของเซอร์โวมอเตอร์

ความเร็วในการหมุนของเซอร์โวมอเตอร์จะขึ้นอยู่กับสัญญาณพัลส์ที่มีคาบช่วงบวก ดังนี้

- ความเร็วสูงสุดของการหมุนทวนเข็มนาฬิกา

เมื่อพัลส์คาบช่วงบวกมากกว่า 1.5 ถึง 2 มิลลิวินาที เซอร์โวมอเตอร์จะหมุนทวนเข็มนาฬิกา โดยความเร็วจะเพิ่มขึ้นเมื่อคาบเวลาบวกเพิ่มขึ้นและจะหมุนได้เร็วสูงสุดเมื่อคาบช่วงบวกมีค่าเท่ากับ 2 มิลลิวินาที

- ความเร็วสูงสุดในการหมุนตามเข็มนาฬิกา

เมื่อพัลส์คาบช่วงบวกตั้งแต่ 1 และน้อยกว่า 1.5 มิลลิวินาที เซอร์โวมอเตอร์จะหมุนตามเข็มนาฬิกา โดยความเร็วจะเพิ่มขึ้นเมื่อคาบเวลาบวกลดลงจาก 1.5 มิลลิวินาทีและจะหมุนเร็วได้สูงสุดเมื่อคาบช่วงบวกมีค่าเท่ากับ 1 มิลลิวินาที

ตัวอย่างโปรแกรม สั่งให้เซอร์โวมอเตอร์หมุนไปทางขวา

```
#include <16F877.h>
#fuses HS,NOLVP,NOWDT,NOPROTECT
#use delay (clock = 10000000)
#use fast_io(b)
void main() {
set_tris_b(0X00);
while (TRUE) {
output_b(0X01);
delay_ms(1);                    //--------- สัญญาณพัลส์ 1ms
output_b(0X00);
delay_ms(20);                   //--------- สัญญาณพัลส์ 20ms
```

}

}

มอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรง เป็นมอเตอร์ที่ขับเคลื่อนด้วยไฟฟ้ากระแสตรง โดยควบคุมความเร็วของมอเตอร์ได้ด้วยการจ่ายกระแสไฟฟ้าเป็นช่วงเวลา แต่ไม่สามารถที่จะควบคุมทิศทางการหมุนของมอเตอร์ได้ การที่จะควบคุมทิศทางการหมุนของมอเตอร์จะต้องใช้ไอซีมาช่วยในการขับเคลื่อน โดยใช้หลักการขับเคลื่อนแบบเอชบริดจ์(H-Bridge)

ภาคผนวก ค

โค้ดโปรแกรมที่ใช้งาน

```
#define F_CPU 16000000UL                    // 16 MHz
#include <avr/io.h>                         // ใช้ชุดคำสั่งอินพุตเอาต์พุตพอร์ตของ AVR
#include <util/delay.h>                     // ประกาศเรียกใช้งานเฮดเดอร์ไฟล์ delay
#include <stdio.h>                          // standard I/O
#include <math.h>                           // ฟังก์ชัน math
#include <stdlib.h>                         // Library for dtostrf()

void delay_ms(int i);                       // ฟังก์ชัน delay

unsigned int ADCRead(char);                 // ฟังก์ชัน A/D ใช้หน่วยความจำ 1 ไบต์

void Open(void);                            // ฟังก์ชัน Open

void Close(void);                           // ฟังก์ชัน Close

void int_USART(void);                       // ฟังก์ชันสำหรับการสื่อสารผ่านพอร์ตอนุกรม
static int uart_putchar(char c, FILE *stream);
static FILE uart_str = FDEV_SETUP_STREAM(uart_putchar, NULL,_FDEV_SETUP_WRITE);
/*************************************************************/
```

```
int main ()                                  //ฟังก์ชัน main เป็นฟังก์ชันหลัก
{
int_USART();                                 // ฟังก์ชันการใช้งาน Hyper terminal
DDRD = 0xFF;                                 // PORT D เป็นเอาต์พุต
DDRB = 0;                                    // PORT B เป็นอินพุต
PORTB=0xFF;                                  //กำหนดค่า 1111 1111 ให้กับรีจิสเตอร์ PORTA

 while (1)

 {    unsigned int c1,c2;                    // กำหนด c1 และ c2 เป็น int

c1 = ADCRead(0);                             // แปลงสัญญาณ c1(LDR)เป็น A/D ที่ PORTF0
delay_ms(50);                                // หน่วงเวลา 0.05 วินาที
c2 = ADCRead(4);                             // แปลงสัญญาณ c2(Rain)เป็น A/D ที่ PORTF4
delay_ms(50);                                // หน่วงเวลา 0.05 วินาที
printf("c1 %d  \n ",c1);                     // แสดงค่า c1 ใน Hyper terminal
printf("c2 %d  \n ",c2);                     // แสดงค่า c2 ใน Hyper terminal

if (c2<=700)                                 // ตรวจสอบสถานะ c2<=700
                                                    {
PORTD = (PORTD&~(1<<PD0));                   // LED สีเขียวดับ
Close();                                     // เรียกใช้ฟังก์ชัน Close
                                                    }

        if (c1>=700)                         // ตรวจสอบสถานะ c1>=700
                                                    {

PORTD = (PORTD&~(1<<PD0));                   // LED สีเขียวดับ
Close();                                     // เรียกใช้ฟังก์ชัน Close
```

```
                                                    }

 PORTD = (PORTD|(1<<PD0));                          // LED สีเขียวติด
delay_ms(500);                                      // หน่วงเวลา 0.5 วินาที
PORTD = (PORTD&~(1<<PD0));                          //LED สีเขียวดับ
delay_ms(1000);                                     // หน่วงเวลา 1 วินาที

                                                        }


return 0;
}

void Open(void)                                     // ฟังก์ชัน Open
{
 PORTD = (PORTD|(1<<PD0));                          // LED สีเขียวติด
 PORTD = (PORTD|(1<<PD5));                          // มอเตอร์หมุนเปิด
 while (1)

   {
if ( (PINB&(1<<PINB1))==0 || (PINB&(1<<PINB5))==0)
                                                    // ตรวจสอบสถานะขาพอร์ตอินพุต PINB บิตที่
                                                    1 หรือบิตที่ 5
                                                      {

PORTD = (PORTD&~(1<<PD5));                          // มอเตอร์หยุดหมุน
delay_ms(100);                                      // หน่วงเวลา 0.1 วินาที
main ();                                            // ไปที่ฟังก์ชัน main

                                                      }
```

```
                                        }
}

void Close(void)                        // ฟังก์ชัน Close
{
 PORTD = (PORTD|(1<<PD1));              // LED สีแดงติด
 PORTD = (PORTD|(1<<PD4));              // มอเตอร์หมุนปิด
 delay_ms(8000);                        // หน่วงเวลา 8 วินาที
 PORTD = (PORTD&~(1<<PD4));             // มอเตอร์หยุดหมุน
 delay_ms(200);                         // หน่วงเวลา 0.2 วินาที

 unsigned int c1,c2;                    // รับค่า c1 และ c2

  while (1)

   {
c1 = ADCRead(0);                        // แปลงสัญญาณ c1(LDR)เป็น A/D ที่ PORTF0
c2 = ADCRead(4);                        // แปลงสัญญาณ c2(LDR)เป็น A/D ที่ PORTF4

PORTD = (PORTD|(1<<PD1));               // LED สีแดงติด
delay_ms(500);                          //หน่วงเวลา 0.5 วินาที
PORTD = (PORTD&~(1<<PD1));              // LED สีแดง
delay_ms(1000);                         // หน่วงเวลา 1 วินาที

    if ( !(c2<=700) && !(c1>=700) )     // ตรวจสอบสถานะ
                                          {

PORTD = (PORTD&~(1<<PD1));              // LED สีแดงดับ

Open();                                 // ไปที่ฟังก์ชัน Open
```

```
                                                    }

    }

}

////////////////////////////////////////////////////////////////////////////////

void int_USART(void)                                // ฟังก์ชันเรียกใช้ Hyper terminal
{

   UCSR0A=0x00;
   UCSR0B=0x18;
   UCSR0C=0x86;
   UBRR0H=0x00;
   UBRR0L=0x67;
   stdout = &uart_str;                              // Set address uart_str to stdout
}

//---------- delay -----------//
void delay_ms(int i)                                // ฟังก์ชันหน่วงเวลา
{
for (;i > 0; i--)                                   //คำสั่งลูป for ใช้เพื่อให้โปรแกรมวนลูป
_delay_ms(1);
}


//--- uart putchar -------------//
static int uart_putchar(char c, FILE *stream)
{
```

```
if (c == '\a')
{
fputs("*ring*\n", stderr);
return 0;
}
if (c == '\n')

uart_putchar('\r', stream);
loop_until_bit_is_set(UCSR0A, UDRE);
UDR0 = c;
return 0;
}

      //A-D


unsigned int ADCRead(char Ch)
{

ADMUX = 0x40 | Ch;
ADCSRA = 0xC3;                                  //Start Conversion ADC divide Clock by 16
ADCSRA |= (1<<ADSC);
   while (!(ADCSRA&(1<<ADIF)));
   return(ADCW);

}
```

บรรณานุกรม

[1] http://www.cpe.ku.ac.th/~yuen/204471/conversion/adc/#Successive_Approximate

[2] http://www.techlib.com/electronics/raindetectors.htm

[3] http://kpp.ac.th/elearning/elearning3/book-09.html

[4] http://www.thaimicrotron.com/Trick/PCB/DIYPCB.htm

ประวัติผู้เขียน

นางสาวนภาพร พิมปรุ เกิดเมื่อวันที่ 24 กันยายน พ.ศ.2532 ภูมิลำเนาอยู่ที่ ตำบลธารปราสาท อำเภอโนนสูง จังหวัดนครราชสีมา สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนศรีสุขวิทยา อำเภอโนนสูง จังหวัดนครราชสีมา เมื่อปี พ.ศ. 2550 ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

นางสาวจันทิมา ชาสิงห์แก้ว เกิดเมื่อวันที่ 26 กันยายน พ.ศ.2532 ภูมิลำเนาอยู่ที่ ตำบลในเมือง อำเภอเมือง จังหวัดขอนแก่น สำเร็จการศึกษา ระดับมัธยมปลายจากโรงเรียนสาธิตศึกษาศาสตร์ มหาวิทยาลัยขอนแก่น อำเภอเมือง จังหวัด ขอนแก่น เมื่อปี พ.ศ. 2550 ปัจจุบันเป็น นักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์มหาวิทยาลัยเทคโนโลยีสุรนารี