

หุ่นยนต์รับส่งเอกสารชนิดเคลื่อนที่ตามเส้น
LINE TRACKING MESSENGING ROBOT



โดย

นาย ดำรงค์ฤทธิ์ ภัณฑารักษ์สกุล

นาย สิทธิชัย เดชพร

นาย สุทัศน์ พริ้งสกุลชัย

เลขหมู่.....
เลขทะเบียน.....42213
วัน, เดือน, ปี.....15 พ.ค. 2545

b.....
i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

หุ่นยนต์รับส่งเอกสารชนิดเคลื่อนที่ตามเส้น
LINE TRACKING MESSENGING ROBOT

โดย

นาย ดำรงค์ฤทธิ์ ภัณฑารักษ์สกุล 41013050

นาย สิทธิชัย เดชพร 41013074

นาย สุทัศน์ พริ้งสกุลชัย 41013077

อาจารย์ที่ปรึกษา

รศ.ดร. ถวิล พึ่งมา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง หุ่นยนต์รับส่งเอกสารชนิดเคลื่อนที่ตามเส้น

LINE TRACKING MESSAGING ROBOT

ผู้จัดทำ

- | | |
|-----------------------------------|----------|
| 1. นาย ดำรงค์ฤทธิ์ ภัณฑารักษ์สกุล | 41013050 |
| 2. นาย สิทธิชัย เดชพร | 41013074 |
| 3. นาย สุทัศน์ พริ้งสกุลชัย | 41013077 |


(รศ.ดร. ทวีต พึงมา)

อาจารย์ที่ปรึกษา



หุ่นยนต์รับส่งเอกสารชนิดเคลื่อนที่ตามเส้น

LINE TRACKING MESSENGING ROBOT

โดย นาย ดำรงค์ฤทธิ์ ภัณฑารักษ์สกุล 41013050
นาย สิทธิชัย เฉลพร 41013074
นาย สุทัศน์ พริ้งสกุลชัย 41013077

อาจารย์ที่ปรึกษา รศ.ดร. ถวิล พึ่งมา

บทคัดย่อ

โครงการนี้เป็นการนำเสนอหุ่นยนต์ที่ใช้ส่งเอกสารหรือพัสดุโดยใช้ไมโครคอนโทรลเลอร์ MCS-51 เป็นส่วนควบคุมให้หุ่นยนต์ตรวจจับเส้น เส้นทางส่งเอกสารหรือพัสดุไปยังจุดที่เรากำหนดไว้ได้ทั้งหมด 256 จุด โดยสามารถสั่งงานได้โดยตรงผ่านสวิทช์ควบคุมบนตัวหุ่นยนต์และมีจอ LCD เป็นตัวแสดงผลในการสั่งงาน

โครงการนี้สามารถไปใช้เป็นต้นแบบการใช้งานแทนคนส่งของตามโรงงานอุตสาหกรรม หรือใช้แทนคนส่งเอกสารในออฟฟิศ โดยหุ่นยนต์สามารถรับน้ำหนักของพัสดุได้ไม่เกิน 30 กิโลกรัม ใช้แบตเตอรี่เป็นแหล่งพลังงานและเนื่องจากวัสดุอุปกรณ์ที่ใช้ในการผลิตสามารถหาได้ภายในประเทศจึงทำให้มีต้นทุนในการผลิตต่ำ

ABSTRACT

The Robot in this project was controlled by the MCS-51, which is used to send or carry the documents to up to 256 assigned destinations. The Robot can be directly commanded by control switches on the robot and display the commanded via LCD.

This project can be regarded as a prototype, which can perform in lieu of the messenger in the Manufacturing Plant or Offices. Furthermore, this Robot, powered by rechargeable battery, can carry documents weigh up to 30 kg. In this project, we used domestic material and components to produce the robot at low cost.

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1 บทนำ

1.1 ทฤษฎีทั่วไปเกี่ยวกับหุ่นยนต์	1
1.2 ที่มาของหุ่นยนต์	2
1.3 เปรียบหุ่นยนต์เป็นคณงาน	3
1.4 สรีระของหุ่นยนต์	4
1.5 หุ่นยนต์อุตสาหกรรม	4
1.6 ประเภทต่างๆ ของหุ่นยนต์	5

บทที่ 2 ทฤษฎีและหลักการ

2.1 หลักการทั่วไป	6
2.2 การทำงานเบื้องต้นของหุ่นยนต์	6
2.3 เรกูเลเตอร์เพาเวอร์ซัพพลาย (Regulator Power Supply)	7
2.4 การใช้ทรานซิสเตอร์ในการควบคุมมอเตอร์ไฟฟ้ากระแสตรง	8
2.5 โครงสร้างและการควบคุมความเร็วรอบ ของมอเตอร์ไฟฟ้ากระแสตรง	9
2.6 ทฤษฎีทางแสง	12
2.7 อุปกรณ์ทางแสง	12
2.8 สเตปปีงมอเตอร์ (Stepping Motor)	21
2.9 โมดูลแสดงผลแบบผลึกเหลว (LCD Module)	23
2.10 สวิตซ์ทรานซิสเตอร์ (Switch Transistor)	28

บทที่ 3 การออกแบบสร้าง Hardware และ Software

3.1 โครงสร้างหุ่นยนต์	31
3.2 ส่วนของการประมวลผล (Microcontroller)	32
3.3 ส่วนของการตรวจจับเส้นทางการเดินทาง	32
3.4 ส่วนของเรกูเลเตอร์ (Regulator Power-supply)	33
3.5 ส่วนของวงจรขับมอเตอร์ไฟฟ้ากระแสตรง	34
3.6 ส่วนของวงจรขับโมดูลแสดงผลแบบผลึกเหลว	35
3.7 ส่วนของวงจรขับสเตปปีงมอเตอร์	36
3.8 โปรแกรมควบคุมการสั่งงานหุ่นยนต์	37

บทที่ 4 การทดลองและผลการทดลอง

4.1 การตรวจสอบวงจรรับแสง	43
4.2 การทดสอบวงจรขับเคลื่อนมอเตอร์	44

	สารบัญ (ต่อ)	
	4.3 การทดลองวงจรเรกเตอร์แบบปรับค่าได้ 2 ระดับ	47
	4.4 การทดลองรัศมีวงเลี้ยวของตัวหุ่นยนต์	47
	4.5 การทดลองระยะห่างระหว่างสถานีรับ – ส่งเอกสาร	48
	4.6 การทดสอบตัวมอเตอร์	49
บทที่ 5	บทวิจารณ์และบทสรุป	
	5.1 ปัญหาทางโครงสร้าง	51
	5.2 ปัญหาของส่วนขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรง	51
	5.3 ปัญหาจากตัวตรวจจับ	51
	5.4 ปัญหาจากโปรแกรมควบคุมหุ่นยนต์	52
	5.5 ปัญหาจากระยะห่างของแต่ละสถานี	52
ภาคผนวก		
กิตติกรรมประกาศ		
หนังสืออ้างอิง		



สารบัญรูปภาพ

หน้าที่

รูปที่ 2.1 ลักษณะการเคลื่อนที่ของหุ่นยนต์โดยอาศัยมอเตอร์ไฟฟ้ากระแสตรง และสแตปปีงมอเตอร์ในการเคลื่อนที่	6
รูปที่ 2.2 การต่อวงจรขยายกระแสให้กับไอซีเรกูเลเตอร์ตระกูล 78XX	7
รูปที่ 2.3 การต่อวงจรเรกูเลเตอร์แบบปรับค่าแรงดันเอาท์พุทได้	8
รูปที่ 2.4 การต่อทรานซิสเตอร์ 4 ตัวเพื่อควบคุมทิศทางการหมุนของมอเตอร์	9
รูปที่ 2.5 หลักการของมอเตอร์ไฟฟ้ากระแสตรง	10
รูปที่ 2.6 (a) วงจรมอเตอร์ไฟฟ้ากระแสตรง (b) วงจรสมมูลย์ของมอเตอร์ไฟฟ้ากระแสตรง	11
รูปที่ 2.7 ตัวตรวจจับแบบใช้หลักการสะท้อนของแสง	13
รูปที่ 2.8 การกำจัดกระแสของแอลอีดีเบื้องต้น	14
รูปที่ 2.9 การเปล่งแสงของแอลอีดีแบบ GaAs และ AlGaAs	14
รูปที่ 2.10 โครงสร้างภายในของโฟโต้โมดูล	15
รูปที่ 2.11 การต่อใช้งาน	15
รูปที่ 2.12 (ก) สัญญาณพัลส์ธรรมดา (ข) สัญญาณโทนเบริสต์	16
รูปที่ 2.13 รายละเอียดแต่ละขาของไอซี 555 ไทมเมอร์แบบ DIL – 8 ขามาตรฐาน	17
รูปที่ 2.14 การทำงาน (ภายในสี่เหลี่ยม) ของไอซี 555 ไทมเมอร์ซึ่งต่อเพื่อใช้ไอซีเป็นตัวกำเนิด โมโนสเตเบิลหรือไทมเมอร์พื้นฐาน	18
รูปที่ 2.15 การหน่วงเวลาของ 555 ไทมเมอร์สำหรับค่าความต้านทานและค่าตัวเก็บประจุต่างๆ	19
รูปที่ 2.16 วงจรอะอสซิลเลตอรีเรกเตอร์พื้นฐานกำเนิดความถี่ 1 KHz	19
รูปที่ 2.17 ความสัมพันธ์ระหว่าง C1,R2 และความถี่ โดยที่ R2 มีค่าสูงมากเมื่อเทียบกับ R1	20
รูปที่ 2.18 โครงสร้างของสแตปปีงมอเตอร์	21
รูปที่ 2.19 แอดเดรส DDRAM ของ LCD Module รุ่น DMC164	27
รูปที่ 2.20 การต่อวงจรสวิตซ์ทรานซิสเตอร์และวงจรสมมูลย์	29
รูปที่ 3.1 ภาพโดยรวมของโครงสร้างหุ่นยนต์	31
รูปที่ 3.2 วงจรภาคส่งสัญญาณโทนเบริสต์	32
รูปที่ 3.3 วงจรภาครับสัญญาณโทนเบริสต์	33
รูปที่ 3.4 วงจรเรกูเลเตอร์แบบคงที่	34
รูปที่ 3.5 วงจรเรกูเลเตอร์แบบปรับค่าได้ 2 ระดับ	34
รูปที่ 3.6 วงจรขับมอเตอร์ไฟฟ้ากระแสตรง	35
รูปที่ 3.7 การต่อโมดูลแสดงผลลิกเหลวเข้ากับ 8255	36
รูปที่ 3.8 วงจรขับสแตปปีงมอเตอร์	36

สารบัญรูปภาพ (ต่อ)

รูปที่ 3.8 วงจรขับเคลื่อนปั๊มมอเตอร์	36
รูปที่ 3.9 โฟลว์ชาร์ทในการเขียนโปรแกรมควบคุมการบังคับลิ้น	37
รูปที่ 3.10 โฟลว์ชาร์ทในการเขียนโปรแกรมควบคุมการเดินหน้าและถอยหลัง	38
รูปที่ 3.11 โฟลว์ชาร์ทที่ใช้ในการเขียนโปรแกรมควบคุมการเดินทางจากสถานีต้นทาง ไปยังสถานีปลายทาง	39
รูปที่ 3.12 โฟลว์ชาร์ทที่ใช้ในการเขียนโปรแกรมควบคุมการเดินทางจากสถานีปลายทาง ไปยังสถานีต้นทาง	40
รูปที่ 3.13 โฟลว์ชาร์ทโดยรวมที่ใช้ในการเขียนโปรแกรมควบคุมหุ่นยนต์	41
รูปที่ 3.14 โฟลว์ชาร์ทโดยรวมที่ใช้ในการเขียนโปรแกรมควบคุมแสดงผลแบบผลึกเหลว	42
รูปที่ 4.1 วงจรที่ใช้ในการทดลองอุปกรณ์ตรวจจับเส้นทาง	43
รูปที่ 4.2 วงจรขับเคลื่อนสแตปปั๊มมอเตอร์ที่ใช้ในการทดลอง	44
รูปที่ 4.3 วงจรขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรงที่ใช้ในการทดลอง	45
รูปที่ 4.4 การต่อวงจรเรกูเลเตอร์แบบปรับค่าได้ 2 ระดับที่ใช้ในการทดลอง	47
รูปที่ 4.5 การทดลองการเคลื่อนที่ของหุ่นยนต์ในเส้นทางวงกลมรัศมีต่างๆ	48
รูปที่ 4.6 การทดลองการเคลื่อนที่ของหุ่นยนต์ที่ระยะห่างของแต่ละสถานีแตกต่างกัน	49
รูปที่ 4.7 การทดลองวัดความเร็วรอบของมอเตอร์โดยใช้ Tachometer	50
รูปที่ 4.8 แผนภูมิความเร็วรอบของมอเตอร์ที่ระดับแรงดันต่างๆ	50

สารบัญตาราง

	หน้าที่
ตารางที่ 2.1 รูปแบบการขับเคลื่อนปั๊มมอเตอร์แบบฟูลสเต็ป 1 เฟส	22
ตารางที่ 2.2 รูปแบบการขับเคลื่อนปั๊มมอเตอร์แบบฟูลสเต็ป 2 เฟส	22
ตารางที่ 2.3 รูปแบบการขับเคลื่อนปั๊มมอเตอร์แบบฮาล์ฟสเต็ป	22
ตารางที่ 2.4 ความสัมพันธ์ของเคอร์เซอร์และตัวอักษร	26
ตารางที่ 3.1 ความสัมพันธ์ของบิตควบคุมกับทิศทางการหมุนของมอเตอร์	35
ตารางที่ 4.1 ค่าลอจิกเอาต์พุตที่ระยะห่างและสีต่างๆ	44
ตารางที่ 4.2 ผลการทดลองของวงจรขับเคลื่อนสเต็ปปั๊มมอเตอร์	44
ตารางที่ 4.3 ผลการทดลองวงจรขับเคลื่อนไฟฟ้ากระแสตรง	46
ตารางที่ 4.4 ผลการทดลองวงจรเรกูเลเตอร์แบบปรับค่าได้ 2 ระดับ	47
ตารางที่ 4.5 การเคลื่อนที่ของหุ่นยนต์ในเส้นทางวงกลมที่รัศมีต่างๆ	47
ตารางที่ 4.6 การเคลื่อนที่ของหุ่นยนต์ที่ระยะห่างของแต่ละสถานีแตกต่างกัน	48
ตารางที่ 4.7 ผลการทดลองวัดความเร็วรอบของมอเตอร์	49



บทที่ 1

บทนำ

1.1 ทฤษฎีทั่วไปเกี่ยวกับหุ่นยนต์

มนุษย์นั้นได้สร้างเครื่องอำนวยความสะดวกขึ้นมา เพื่อให้ใช้ในชีวิตประจำวัน มนุษย์กับเครื่องจักรจึงเป็นสิ่งที่อยู่ควบคู่กันมาโดยตลอดเพราะว่ามนุษย์นั้นในด้านของกายภาพแล้ว มนุษย์จะไม่สามารถทำกิจกรรมที่ซ้ำ ๆ กันได้เหมือนกันทุกครั้ง เช่น ไม่สามารถเขียนอักษรตัวเดิมเท่ากันทุกตัวจึงได้คิดพิมพ์คิดขึ้นมาใช้ หรือ ความสามารถบางอย่างมีขอบเขตจำกัด เช่น ยกของหนัก ๆ มนุษย์จะสามารถยกน้ำหนักได้จำกัด มนุษย์จึงคิดค้นเครื่องจักรหรือกลไกที่ช่วยผ่อนแรงขึ้นมาใช้จะเห็นว่าเครื่องจักรกลกับมนุษย์นั้นเป็นสิ่งที่อยู่คู่กัน

แต่เดิมนั้นเครื่องจักรถูกควบคุมโดยมนุษย์ แต่หลังจากมนุษย์ได้สร้างคอมพิวเตอร์ (Computer) ขึ้นมามนุษย์ก็ได้คิดสร้างเครื่องจักรที่สามารถทำงานได้เองโดยอัตโนมัติ ไม่ต้องอาศัยมนุษย์ควบคุมทุกขั้นตอน ซึ่งเครื่องจักรบางประเภทได้ถูกออกแบบให้เลียนแบบการเคลื่อนไหวของมนุษย์ เครื่องจักรแบบนี้จะเรียกว่า หุ่นยนต์ (ROBOT)

หุ่นยนต์ถูกออกแบบให้เคลื่อนที่โดยการเลียนแบบการเคลื่อนไหวของมนุษย์ เพื่อให้สามารถทำงานแทนมนุษย์ได้ โดยผ่านการควบคุมจากโปรแกรมคอมพิวเตอร์อีกที การเลียนแบบการเคลื่อนไหวและการทำงานจะต้องทำให้คล้ายกับมนุษย์มากที่สุด ดูได้จากเครื่องล้างจานเป็นตัวอย่าง จะเห็นได้ว่าไม่สามารถเรียกว่าหุ่นยนต์ได้ เพราะงานที่ทำแค่ล้างจานเท่านั้น แต่ไม่ได้วางจานลงในอ่างเอง ไม่ใช่ผ้าเช็ดจาน หรือไม่ได้วางจานลงในที่พักจานเสร็จ จินตนาการของมนุษย์ต่อหุ่นยนต์ที่สมบูรณ์ที่สุดคือ “มนุษย์จักรกล” และในปัจจุบันจะพบบ่อยมากขึ้นที่หุ่นยนต์จะคล้ายส่วนหนึ่งของมนุษย์จักรกล แขนกลหุ่นยนต์ส่วนใหญ่ก็เป็นอีกสิ่งที่สร้างเลียนแบบแขนมนุษย์นั่นเอง

หุ่นยนต์ประเภทแขนกล (Robot arm) นั้นส่วนใหญ่ใช้ในงานเชื่อม, งานหีบห่อ, งานพ่นสี, งานจับของวางตามตำแหน่ง และงานประกอบผลิตภัณฑ์ต่าง ๆ ที่ผลิตอยู่ในทุกวันนี้ เช่น การวางตำแหน่งของตัวไอซีลงบนแผ่นวงจรพิมพ์ที่แม่นยำ หรือการประกอบชิ้นส่วนรถยนต์ ซึ่งในปัจจุบันบางส่วนของงานหรือทั้งหมดของกระบวนการทำโดยหุ่นยนต์ทั้งสิ้น

องค์ประกอบของระบบหุ่นยนต์ที่ดูซับซ้อนสามารถแบ่งอย่างชัดเจนได้เป็น 3 กลุ่ม โดยอ้างอิงกับวัฏชะของมนุษย์ได้ดังนี้

- ฮาร์ดแวร์ (Hard Ware) คือ ร่างกายภายนอก อาทิ แขนกลและกริปเปอร์ (Gripper) คือ แขนและมือ

ระบบการมองเห็น (Vision system) คือ ตาทั้งสองข้าง

- คอมพิวเตอร์ คือ สมอง
- การโปรแกรมของซอฟต์แวร์ (Soft Ware) คือ การศึกษาหรือประสบการณ์ที่เรียนรู้

ชิ้นส่วนทางกลของหุ่นยนต์ที่ทำงานทางกายภาพถูกออกแบบให้ทำงานได้ในกลุ่มของงานเฉพาะอย่างและภายในขอบเขตจำกัดที่ขึ้นอยู่กับขนาดของหุ่นยนต์เอง ปริมาณของหุ่นยนต์เคลื่อนที่ที่มีอยู่ส่วนใหญ่มักจะเป็นแบบ Automatically Guided Vehicles (AGVs) โดยการใช้งานหลักของหุ่นยนต์ประเภทนี้คือ

การขนส่งส่วนของวัตถุใด โดยมัลักษณะเป็นระบบสายพานลำเลียง (Conveyer) ที่ฉลาด, เคลื่อนที่ได้และสามารถโปรแกรมได้ด้วย

แขนกลหุ่นยนต์คือ ข้อต่อซึ่งต่อกันเป็นอนุกรมเพื่อทำงานเหมือนสิ่งทีเคลื่อนที่ได้ (Manipulator) อย่างอิสระหรือเปรียบเสมือนแขนมนุษย์ โดยมีมอเตอร์เป็นตัวควบคุมการเคลื่อนที่ของแต่ละข้อต่อ การทำงานของแต่ละข้อต่อไม่จำเป็นต้องเป็นอิสระต่อกันแต่ต้องเป็นส่วนเสริมการทำงานต่าง ๆ ที่เป็นอิสระต่อกัน ถ้าการทำงานไม่เป็นอิสระต่อกันคอมพิวเตอร์จะโปรแกรมการทำงานที่ต้องการให้ดูเหมือนเป็นอิสระต่อกันได้ อย่างไรก็ตามวิธีนี้จะทำให้งานของคอมพิวเตอร์หนักมากขึ้นเพราะอาจต้องมีการคำนวณก่อนการเคลื่อนที่ทุกครั้ง ซึ่งเวลาในการคำนวณนี้ไม่มีมากพอสำหรับการคำนวณในส่วนอื่น ถึงอย่างไรเครื่องจักรกลก็ไม่มีทางเคลื่อนที่ได้อย่างสมบูรณ์, สลับซับซ้อนหรือยืดหยุ่นเท่าแขนของมนุษย์ได้

แขนกลหุ่นยนต์สมัยใหม่ คืออุปกรณ์ที่ตั้งยึดอยู่กับพื้นหรือยึดกับเครื่องจักรที่ใช้งานร่วมกัน ถูกควบคุมการทำงานโดยคอมพิวเตอร์ที่มีประสิทธิภาพสูง ซึ่งโดยปกติจะวางอยู่ใกล้กับตัวหุ่นยนต์และภายในส่วนคอมพิวเตอร์ประกอบด้วยอุปกรณ์ทางลอจิก (Logic component) และส่วนขยาย (Amplifier) ทั้งหมดที่จำเป็นในการทำงานของหุ่นยนต์ หุ่นยนต์ตั้งพื้นสามารถหยิบชิ้นงานที่หนักเกิน 100 กิโลกรัม (Kilogram) หรือ 220 ปอนด์ (Pound) ได้ และวางลงภายใน 1 มิลลิเมตร หรือ 0.04 นิ้ว (Inch) ของตำแหน่งที่ต้องการได้อย่างง่ายดายสามารถเคลื่อนสิ่งของได้ด้วยความเร็วโดยประมาณ 1 เมตร หรือ 3 ฟุตต่อวินาที และวางของลงได้อย่างนุ่มนวลเท่าที่ต้องการ กริปเปอร์หรือมือหุ่นยนต์ในหลายประเภทสามารถเคลื่อนที่และควบคุมทางของสิ่งต่างๆ ได้

1.2 ที่มาของหุ่นยนต์

หุ่นยนต์หรือโรบอต (ROBOT) เกิดจากการที่คนหลาย ๆ คนให้ความเห็นที่ตรงกันว่าจะเรียกสิ่งนั้นว่าเป็น “หุ่นยนต์” หรือมักจะใช้คำว่า “โรโบติก (Robotic)” เพื่ออธิบายถึงเครื่องจักรที่ออกแบบมาเพื่อทำงานโดยอัตโนมัติด้วยตัวมันเอง ถ้าหากมีภรณ์ที่สามารถวิ่งได้เองโดยอัตโนมัติเราก็สามารถเรียกว่า “โรโบติกจีป (Robotic Jeep)” ก็ได้ หรือหุ่นยนต์อุตสาหกรรมถูกเรียกว่าแขนกล หรือ Robotic arm ซึ่งคุณสมบัติอย่างกว้าง ๆ ของหุ่นยนต์ก็มีพอสังเขปดังนี้

- ควบคุมด้วยคอมพิวเตอร์
- มีส่วนประกอบที่เคลื่อนที่ได้
- มีเซอร์โวมอเตอร์ (Servo motor) เป็นส่วนร่วมในการทำงาน
- มียูสเซอร์อินเตอร์เฟซ (User interface) เป็นส่วนติดต่อระหว่างมนุษย์กับตัวหุ่นยนต์
- ตัวอินเตอร์เฟซ (Interface) อาจเป็นแค่ปุ่มเริ่มการทำงานหรือกุญแจเปิดปิด (Keyswitch)
- สามารถโปรแกรมใช้งานต่าง ๆ ได้
- มีการตอบสนองกับสภาพรอบข้างของหุ่นยนต์โดยสัญญาณอินพุตกับเอาต์พุต
- จับยึดหรือตรวจสอบสิ่งภายนอกบางอย่างให้กับตัวหุ่นยนต์เองด้วยวิธีใดวิธีหนึ่งได้

และหากพิจารณาจากคอมพิวเตอร์เล่นหมากรุก จะเห็นว่ามีฉลาดในการเล่นเกมหมากรุกได้เป็นอย่างดี แต่เราไม่เรียกคอมพิวเตอร์นี้ว่าหุ่นยนต์อย่างไรก็ตาม หากคิดแขนหุ่นยนต์เข้าไปและให้คอมพิวเตอร์สั่งงานให้แขนกลจับตัวหมากรุก เกิดการทำงานอย่างต่อเนื่องถึงจะเป็นที่ยอมรับได้

ว่าสิ่งนี้ คือหุ่นยนต์เล่นหมากรุกได้ นี่คือการแสดงให้เห็นว่าการเคลื่อนที่ได้ คือส่วนสำคัญในการเห็น โดยทั่วไปของการยอมรับความเป็นหุ่นยนต์ แต่บางครั้งเราจึงเรียกระบบของการมองเห็นได้ (Vision system) ว่าเป็น “ระบบหุ่นยนต์” (Robotic system) แม้ว่าจะไม่มีการเคลื่อนที่มาเกี่ยวข้องก็ตาม

Robot มาจากคำว่า Creck ซึ่งหมายถึงทาสหรือคนรับใช้ และเข้ามาในศัพท์ภาษาอังกฤษในปี ค.ศ. 1921 โดยนักเขียนบทละครชื่อ KAREL KAPEX ในบทละครแนวล้อเลียน โดยละครเรื่องนี้หุ่นยนต์ก็คือจักรกลที่คล้ายคลึงกับมนุษย์และนำมาทำงานที่น่าเบื่อแทนมนุษย์ แต่ในตอนหลังพวกหุ่นยนต์รวมตัวกันต่อต้านและทำลายมนุษย์

ในช่วงปลายปี 1940 ระหว่างสงครามโลกครั้งที่ 2 งานวิจัยหุ่นยนต์อุตสาหกรรมได้เริ่มเกิดขึ้น ซึ่งเป็นงานวิจัยเกี่ยวกับการพัฒนาเครื่องจักรกลควบคุมระยะไกลสำหรับการขนถ่ายวัสดุคุ้มกันตลุมพราง

ในกลางปี 1950 George C.Devol ได้พัฒนาอุปกรณ์ที่เขาเรียกว่า “Programmed articulated transfer device” ซึ่งเป็นมანიพิวเลเตอร์ (Manipulator) ที่การทำงานของมันสามารถโปรแกรมได้ ซึ่งแนวความคิดนี้ต่อมาได้ถูกนำเข้าสู่อุตสาหกรรม โดยบริษัท Unimation Inc ในปี 1959 หัวใจสำคัญของอุปกรณ์ชนิดนี้ คือการใช้ประโยชน์ของคอมพิวเตอร์ร่วมกับมანიพิวเลเตอร์ เพื่อผลิตเครื่องจักรกลที่ซึ่งสามารถถูกตั้งให้ทำงานที่หลากหลายอย่างอัตโนมัติ โดยการโปรแกรมเข้าใหม่

ในปี 1968 Mccrty และผู้ร่วมงานของเขาที่ The Stanford Artificial Intekelligence Laboratory ได้รายงานการพัฒนาคอมพิวเตอร์กับมือ, ตา, หู (Manipulators, TV cameras, Microphones) โดยระบบนี้สามารถที่จะจดจำข่าวสารในรูปแบบของเสียง และมองวัตถุที่ถูกวางอย่างกระจัดกระจายอยู่บนโต๊ะเพื่อโยกย้ายสิ่งของเหล่านั้นตามคำสั่ง

ในปี 1995 บริษัท IBM ได้พัฒนามანიพิวเลเตอร์ ซึ่งควบคุมโดยคอมพิวเตอร์กับเซนเซอร์แรงและสัมผัส

ในวันนี้เราจะพบว่าหุ่นยนต์เป็นสาขาที่มีขอบเขตกว้างมาก ซึ่งประกอบไปด้วย Kinematics, Dymanics, Planning system, Programming languages, Machine intekelligence

1.3 เปรียบหุ่นยนต์เป็นผลงาน

มนุษย์มีความสนใจอย่างยิ่งที่จะทำงานให้น้อยที่สุดเท่าที่จะน้อยได้แต่ให้งานทั้งหมดสำเร็จ เราออกแบบหุ่นยนต์ขึ้นเพื่อเพิ่มลักษณะบวกและกำจัดลักษณะลบของมนุษย์ โดยส่วนใหญ่ความพยายามทั้งหมดจะอยู่ในทิศทางนี้ ความแตกต่างที่สำคัญระหว่างเครื่องจักรอัตโนมัติปัจจุบันและอดีต คือ เครื่องจักรปัจจุบันสามารถโปรแกรมได้ ณ วันที่สิ่งนี้หุ่นยนต์กำลังขึ้นอยู่กับข้อมูลที่ใส่เข้าไปในตัวหุ่นยนต์ในวันหน้าหุ่นยนต์จะเพิ่มความชาญฉลาดและในที่สุดจะมีความฉลาดที่ยิ่งใหญ่ หุ่นยนต์จะสามารถทำงานของคนให้ได้ประโยชน์สูงสุด และตอบสนองต่อสิ่งที่ไม่คาดคิดและสิ่งรบกวนต่างๆ ที่คาดไม่ถึงได้อย่างหลักแหลม ซึ่งสิ่งรบกวนที่คาดไม่ถึงทั้งหมดจะควบคุมได้ด้วยการปิดเครื่อง, ส่งเสียงเตือน, เรียกหาผู้ปฏิบัติงาน, หรือคัดค้านส่วนออก สำหรับการเปลี่ยนแปลงที่น้อยมากจากวัฏจักรหนึ่งไปยังวัฏจักรหนึ่งและขึ้นส่วนหนึ่งไปยังขึ้นส่วนหนึ่งนั้นยากเกินที่จะควบคุมได้ แต่หุ่นยนต์มีความละเอียดอ่อนและแข็งแรงพอที่จะกำหนด, ตรวจจับ และตอบสนองได้อย่างเหมาะสม

1.4 สรีระของหุ่นยนต์

องค์ประกอบพื้นฐานที่ครอบคลุมการเคลื่อนที่และความสารถของมนุษย์คือ การออกแบบของรูปร่างและความซับซ้อนของซอฟต์แวร์ หุ่นยนต์ถูกออกแบบมาให้มีรูปแบบและขนาดต่างๆ มากมายและซอฟต์แวร์เป็นส่วนทำให้เกิดผลสูงสุดกับความต้องการในแต่ละลักษณะเฉพาะของหุ่นยนต์ ในสภาพแวดล้อมสามมิติของเราในทุกวันนี้ มีวิธีอันหลากหลายที่จะออกแบบหุ่นยนต์ให้ขยับไปถึงทุกจุดในขอบเขตการทำงานของคน ด้วยกฎที่ว่าต้องมีมอเตอร์ (Motor) หนึ่งตัวต่อหนึ่งองศาอิสระ (Degree of freedom) ที่ต้องการจะเจาะจงลงไป ดังนั้นอย่างน้อยที่สุดต้องมีมอเตอร์ สามตัวเพื่อทำงานในตำแหน่งสามมิติ (ระนาบแกน X, Y, และ Z) และอีกสามตัวเพื่อหมุนมือในการหมุนสามแบบที่เป็นไปได้ (roll, pitch และ yaw) ทั้งหมดนี้คือหุ่นยนต์ที่มีพื้นฐานบนแกนสากลทั้งหก (The basic universal six-axis robot)

การออกแบบทางกลของรูปร่างแขนมนุษย์ที่ควรจะสังเกตไว้ได้แก่

ระยะจากไหล่ไปยังข้อศอกมากกว่าระยะจากข้อศอกไปข้อมือ ซึ่งมากกว่าระยะจากข้อมือไปข้อมนิ้ว ที่มากกว่าระยะจากข้อต่อแรกข้อมนิ้วไปยังข้อต่อที่สองข้อมนิ้ว โดยระยะจากข้อต่อที่สองข้อมนิ้วไปยังปลายนิ้วเป็นระยะที่สั้นที่สุด

ความละเอียดของระบบจะมากขึ้น ขณะที่เราเข้าใกล้กับกริปเปอร์และเนืองงานมากขึ้น สำหรับแขนมนุษย์นั้นจำนวนประสาทในแขนจะเพิ่มมากขึ้นจากไหล่ไปยังปลายนิ้ว ในส่วนหลังจากข้อมือไปความยืดหยุ่นเกิดขึ้นเมื่อกางฝ่ามือออก, การกำหมัด, การเคลื่อนที่ข้อมนิ้วต่างๆ ออกจากนิ้วและการประทับรอยนิ้วมือ การมีนิ้วที่ยาวไม่เท่ากัน คือการขัดเกลาให้เกิดความละเอียดที่มากขึ้น

ข้อมูลดังกล่าวให้ข้อบ่งชี้การทำงานของแขนมนุษย์เปรียบเทียบกับลักษณะของแขนมนุษย์เปรียบเทียบกับลักษณะทางกลของหุ่นยนต์ ซึ่งแขนมนุษย์ได้ทำในเกือบทุกสิ่งที่มีมนุษย์ต้องการมาเป็นเวลาล้านๆ ปีแล้ว มีอะไรบางอย่างที่วุ่นวายนี้ได้ออกเราเกี่ยวกับแขนกลหุ่นยนต์

1.5 หุ่นยนต์อุตสาหกรรม

การกำกับการเคลื่อนที่ของมนุษย์ขึ้นอยู่กับงานที่ต้องการให้หุ่นยนต์ทำดังนั้นการออกแบบจะออกมาในรูปของความสามารถขั้นต่ำสุดที่จะทำให้งานนั้นสำเร็จได้ ซึ่งการออกแบบก็จะมีพารามิเตอร์ (Parameter) หลักๆ อาทิ ระยะการเคลื่อนที่ของแต่ละแกน, หน่วยความจำที่ต้องการและจำนวนของตัวตรวจจับ (Sensor) แต่พารามิเตอร์เหล่านี้ควรมีจำนวนน้อยที่สุด เพื่อที่จะสามารถผลิตหุ่นยนต์ที่ประหยัดได้มากที่สุดเท่าที่เป็นไปได้

ลักษณะโครงสร้างหุ่นยนต์ (Robot geometry) แยกออกตามรูปร่างที่ผู้ออกแบบใช้การเลียนแบบขึ้น ซึ่งโครงสร้างของหุ่นยนต์โดยทั่วไปได้แก่

- Articulated : แขนกลหุ่นยนต์แบบนี้จะทำงานเหมือนการทำงานของแขนมนุษย์ การออกแบบที่ยืดหยุ่นแบบนี้จะให้การเคลื่อนที่เหมือนแขนของมนุษย์มากที่สุด
- Cylindrical : โครงสร้างแบบนี้กำกับหุ่นยนต์ซึ่งแกนของหุ่นยนต์ออกแบบมาให้มีลักษณะเฉพาะเหมือน Cylindrical coordinate โดยทั่วไปจะมีแกนหนึ่งแกนเคลื่อนที่ขึ้นลงเหมือนทรงกระบอก และแกนอื่นจะเคลื่อนที่เข้าและออกเหมือนของเวกเตอร์ (Vector)

- Spherical : สำหรับหุ่นยนต์แบบนี้จะใช้มุมสองมุมและการกำหนดเชิงรัศมีเพื่อกำหนดจุดใดๆ ในพื้นที่ที่ใช้งาน
- ระบาย XYZ หรือ Orthogonal : หุ่นยนต์แบบ Orthogonal ก็คือเครื่องจักรแบบระบาย XYZ โดยมีด้านสามด้านซึ่งแต่ละด้านจะอยู่ทางมุมขวาของอีกด้านหนึ่ง
- หุ่นยนต์ Gantry : หุ่นยนต์แบบนี้จะมีกริปเปอร์แขวนอยู่บนโครงสร้างใดๆ ที่ครอบคลุมการทำงานได้ในพื้นที่กว้าง

1.6 ประเภทต่างๆ ของหุ่นยนต์

หุ่นยนต์อาจแยกออกเป็น 2 ประเภทใหญ่ๆ คือ

1.6.1 Fixed Robots คือ หุ่นยนต์ที่ถูกตรึงกับฐานที่ถูกยึดคงที่ ไม่สามารถเคลื่อนที่ได้อิสระ

1.6.2 Mobile Robots คือ หุ่นยนต์ที่ถูกตรึงกับฐานที่สามารถเคลื่อนที่ได้โดยอิสระโดยที่ฐานมีล้อ หรือ ดันระบบ

สรุปได้ว่าหุ่นยนต์คือ “เครื่องจักรที่สามารถโปรแกรมได้ และควบคุมด้วยคอมพิวเตอร์เพื่อเคลื่อนที่สิ่งของผลิตภัณฑ์ต่างๆ และ/หรือเครื่องมือเพื่อทำงาน หุ่นยนต์สามารถตัดสินใจและมีการตอบสนองกับสภาพแวดล้อมได้อย่างชาญฉลาด สามารถรับและส่งข้อมูลให้กับสิ่งรอบข้าง แต่อาจจะเคลื่อนที่ไปยังที่ต่างๆ ได้หรือไม่ก็ได้”

จากทั้งสองประเภทของหุ่นยนต์ คือ Fixed Robot และ Mobile Robot หุ่นยนต์ประเภทที่สองค่อนข้างจะเป็นหุ่นยนต์ที่มีความยืดหยุ่นสูงกว่าประเภทแรก ซึ่งในโครงการนี้ ก็จะเป็นเรื่องเกี่ยวกับ Mobile Robot ดังจะกล่าวต่อไป

บทที่ 2

ทฤษฎีหรือหลักการ

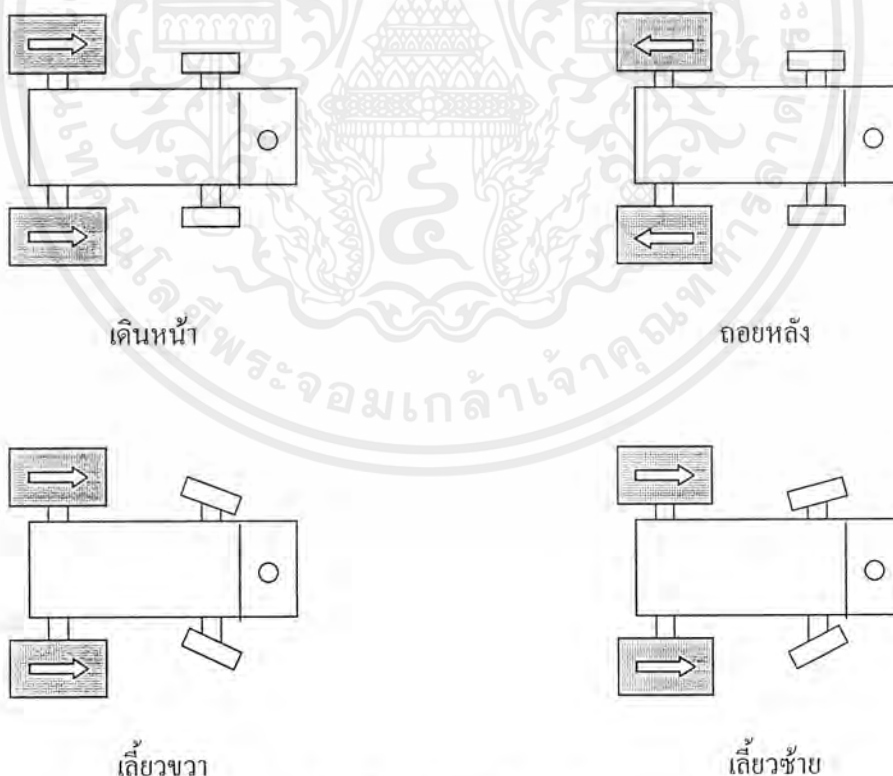
2.1 หลักการทั่วไป

จากที่ได้มีการกล่าวในบทหน้านั้น เราสรุปได้ว่าหุ่นยนต์ คือ “เครื่องจักรที่สามารถโปรแกรมได้ และควบคุมด้วยคอมพิวเตอร์เพื่อเคลื่อนที่สิ่งของหรือเครื่องมือเพื่อทำงาน หุ่นยนต์สามารถตัดสินใจและมีการตอบสนองกับสภาพแวดล้อมได้อย่างชาญฉลาด สามารถรับส่งข้อมูลให้กับสิ่งรอบข้าง แต่อาจจะเคลื่อนที่ไปยังที่ต่าง ๆ ได้หรือไม่ก็ได้”

โครงการนี้จะนำความรู้ที่ได้ศึกษาที่นี้เกือบทั้งหมด มาประยุกต์ใช้งานให้เป็นรูปธรรมโดยนำเทคนิคและทฤษฎีต่าง ๆ มาประยุกต์ใช้ในการสร้าง ลักษณะของหุ่นยนต์มีการเคลื่อนที่โดยใช้ 2 ล้อหลักในการขับเคลื่อน และใช้ 2 ล้อหน้าในการควบคุมทิศทางในการเคลื่อนที่ โดยมีต้นกำเนิดกำลังงานเป็นมอเตอร์ไฟฟ้ากระแสตรง และสเตรปปิงมอเตอร์ตามลำดับ หุ่นยนต์จะทำการเคลื่อนที่ไปในที่ต่าง ๆ โดยอาศัยเส้นสีที่ทำไว้สำหรับนำทางซึ่งเป็นเส้นสีดำหรือสีเข้ม และภายใต้ตัวหุ่นยนต์จะมีอุปกรณ์ในการตรวจจับเส้นทาง (Sensor) ติดตั้งไว้ เพื่อให้หุ่นยนต์เคลื่อนที่ไปในทิศทางที่เราต้องการได้ โดยอาศัยทฤษฎีการสะท้อนของแสง

2.2 การทำงานเบื้องต้นของหุ่นยนต์

มอเตอร์ไฟฟ้ากระแสตรงสำหรับระบบขับเคลื่อนของหุ่นยนต์

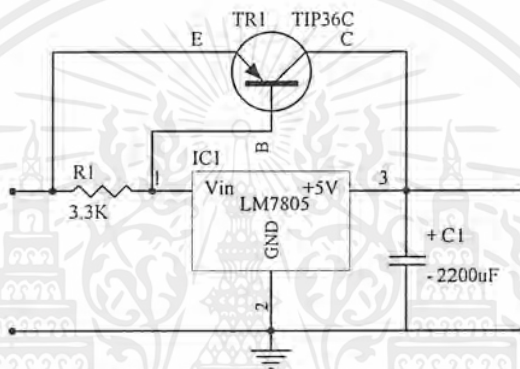


รูปที่ 2.1 ลักษณะการเคลื่อนที่ของหุ่นยนต์โดยอาศัยมอเตอร์ไฟฟ้ากระแสตรง และสเตรปปิงมอเตอร์ในการเคลื่อนที่

มอเตอร์ไฟฟ้ากระแสตรงที่ใช้ในการขับเคลื่อนจัดได้ว่ามีความสำคัญมากส่วนหนึ่งของตัวหุ่นยนต์ จะต้องมีการพิจารณาจุดที่ใช้ในการติดตั้งแล้วทำให้น้ำหนักของตัวหุ่นยนต์สมดุลกัน ในการออกแบบโดยทั่วไปจะใช้มอเตอร์ไฟฟ้ากระแสตรง 2 ตัว ในการขับเคลื่อนล้อหลังทั้ง 2 ข้างแล้วอาศัยสแตปปีงมอเตอร์ในการควบคุมทิศทางการเคลื่อนที่ของหุ่นยนต์ ดังแสดงในรูปที่ 2.1

2.3 เรกูเลเตอร์เพาเวอร์ซัพพลาย (Regulator Power Supply)

2.3.1 วงจรเรกูเลเตอร์แบบพื้นฐานที่ใช้งานกันโดยทั่วไป จะนิยมใช้อิซีเรกูเลเตอร์สำเร็จรูปตระกูล 78XX ในการควบคุมระดับแรงดันเอาต์พุตที่ออกให้มีค่าคงที่ โดยปกติอิซีเรกูเลเตอร์ตระกูล 78XX ธรรมดาจะสามารถจ่ายกระแสได้สูงสุดเพียง 1 แอมป์เท่านั้น ซึ่งเป็นข้อจำกัดของอิซีเรกูเลเตอร์ตระกูลนี้ ดังนั้นจึงต้องมีการขยายกระแสให้กับเรกูเลเตอร์ตระกูล 78XX ก่อนนำไปใช้งาน มีลักษณะการต่อวงจรดังรูปที่ 2.2



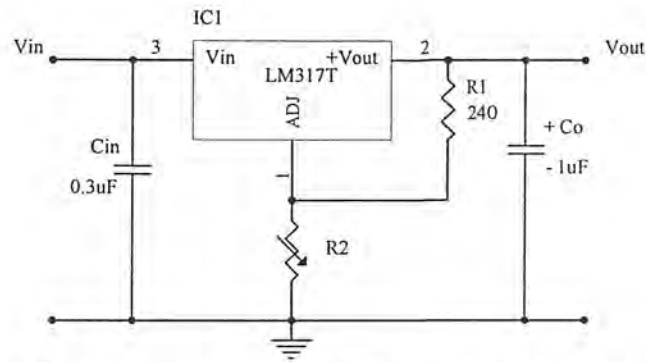
รูปที่ 2.2 การต่อวงจรขยายกระแสให้กับอิซีเรกูเลเตอร์ตระกูล 78XX

อิซีเรกูเลเตอร์ตระกูล 78XX จะมีคุณสมบัติในการป้องกันความร้อนภายใน แต่ในทางปฏิบัติไม่สามารถที่จะป้องกันได้ทั้งหมดที่ ดังนั้นเพื่อเสถียรภาพที่ดีกว่า จึงมีการออกแบบวงจรตามรูปที่ 2.2 คือเมื่อโหลดมีการใช้กระแสเกิน 200 มิลลิแอมป์เมื่อใด ทรานซิสเตอร์ (Transistor) ก็จะทำหน้าที่เป็นตัวผ่านกระแสที่เกินนั้นแทนอิซีเรกูเลเตอร์ ทำให้วงจรสามารถจ่ายกระแสได้สูงสุดถึง 5 แอมป์ และเพื่อความปลอดภัยยิ่งขึ้น ควรติดแผ่นระบายความร้อนให้อุปกรณ์ทั้งสอง ในกรณีที่มีการลัดวงจรทางด้านเอาต์พุตเพียงชั่วครู่ ขนาดของกระแสก็ไม่เกิน 5 แอมป์ เพราะแรงดันไบอัสที่ให้ทรานซิสเตอร์จะถูกจำกัดไว้เนื่องจากเป็น แรงดันทางด้านอินพุต (Input) ให้กับอิซีเรกูเลเตอร์ด้วย

ตัวต้านทาน R_1 จะทำหน้าที่ป้องกันกระแสเกินให้กับอิซีเรกูเลเตอร์เป็นจุดแรก ขณะเดียวกันก็ทำหน้าที่ป้องกันแรงดันไบอัสให้กับทรานซิสเตอร์ด้วย ซึ่งขนาดอัตราทนกำลังของ R_1 คำนวณได้จากค่ากระแสที่ไหลผ่านตัวต้านทานสูงสุด ($P = I^2 R$) ซึ่งจะอยู่ในอัตรา 0.5–5 วัตต์

ตัวเก็บประจุ C_1 จะทำหน้าที่ในการกำจัดริบเบิล (Ripple) ของแรงดันด้านอินพุต หากวงจรมีการใช้กระแสไม่เกิน 1 แอมป์ จะใช้ค่าความจุเท่ากับ 1000 นาโนฟารัด ก็เพียงพอ แต่ถ้าจำเป็นต้องใช้กระแสเอาต์พุต (Output) สูงถึง 5 แอมป์ ก็ควรจะเพิ่มค่าความจุไปเป็น 4700 ไมโครฟารัดแทน

2.3.2 วงจรเรกูเลเตอร์แบบปรับค่าได้พื้นฐานที่ใช้งานโดยทั่วไป นิยมใช้อิซีเรกูเลเตอร์สำเร็จรูปเบอร์ LM 317 ในการควบคุมระดับแรงดันเอาต์พุตที่ออกให้มีค่าคงที่ และสามารถทำการปรับค่าระดับแรงดันตามที่เรากำลังต้องการได้ มีลักษณะการต่อวงจรดังรูปที่ 2.3



รูปที่ 2.3 การต่อวงจรเรกูเลเตอร์แบบปรับค่าแรงดันเอาต์พุตได้

จากรูปที่ 2.3 จะเห็นว่าค่าของแรงดันเอาต์พุตถูกกำหนดโดยค่าความต้านทาน R_1 , R_2 และ กระแส I_{adj} ซึ่งสามารถคำนวณหา V_{out} ได้จากสมการ

$$V_{out} = 1.25V \left[1 + \frac{R_2}{R_1} \right] + [I_{adj} \times R_2]$$

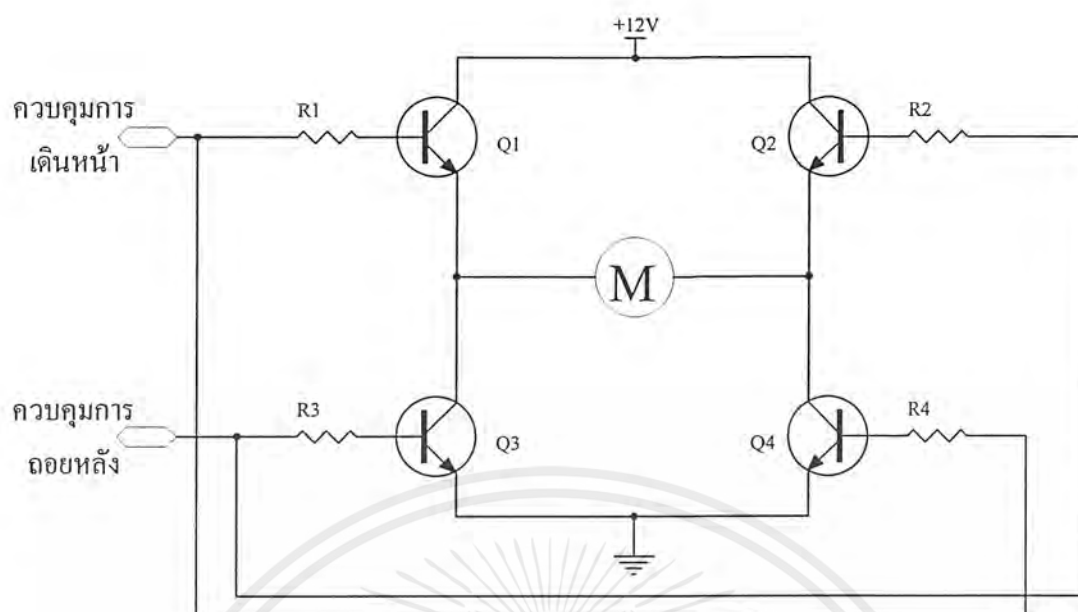
การออกแบบจะใช้ค่าของ R_1 และ I_{adj} ที่กำหนดไว้ในคู่มือ โดยใช้ $R_1 = 240 \Omega$ และ $I_{adj} = 50\mu A$ ในการออกแบบ ดังนั้นเราจึงคำนวณค่าของ R_2 ได้จากสมการ

$$R_2 = \frac{(V_o - 1.25)R_1}{1.25 + I_{adj}R_1}$$

ในการออกแบบถ้าต้องการให้ระดับแรงดันเอาต์พุตในแต่ละตำแหน่งสามารถปรับระดับแรงดันได้ต่ำสุดเท่ากับ $1.25V - 30V$ จากการคำนวณทำให้ได้ค่า $R_2 = 0 \Omega$ ที่แรงดัน $1.25V$ และ $R_2 = 5.47V$ ที่แรงดัน $30V$ และเพื่อให้วงจรสามารถปรับระดับแรงดันเอาต์พุตที่ออกได้ จึงจำเป็นต้องแทน R_2 ด้วยตัวต้านทานปรับค่าได้แทน ส่วน C_{in} มีหน้าที่ในการลดริบเบิลจากวงจรกรองแรงดันและ C_o มีหน้าที่ในการยกระดับการตอบสนองทรานเซียนต์ (Transient)

2.4 การใช้ทรานซิสเตอร์ในการควบคุมมอเตอร์ไฟฟ้ากระแสตรง

การใช้ทรานซิสเตอร์ในการควบคุมมอเตอร์นั้นเป็นการนำทรานซิสเตอร์มาใช้งาน ในลักษณะของสวิตช์ โดยจะใช้ทรานซิสเตอร์เพียง 2 ตัวเท่านั้นในการควบคุมการทำงาน มีการต่อวงจรในลักษณะดังรูปที่ 2.4



รูปที่ 2.4 การต่อทรานซิสเตอร์ 4 ตัวเพื่อควบคุมทิศทางการหมุนของมอเตอร์

จากรูปที่ 2.4 การทำงานในช่วงเวลาหนึ่งเราจะใช้ทรานซิสเตอร์เพียง 2 ตัวเท่านั้นในการนำกระแส คือ Q1 กับ Q4 หรือไม่ก็ Q2 กับ Q3 เท่านั้น ถ้าชุดใดชุดหนึ่งนำกระแสแล้ว อีกชุดต้องหยุดนำกระแส การที่ทรานซิสเตอร์นำกระแสเป็นคู่นี้จะทำให้สามารถบังคับทิศทางการไหลผ่านตัวมอเตอร์ไฟฟ้ากระแสตรงได้ ส่งผลให้สามารถบังคับทิศทางการหมุนของมอเตอร์ได้ด้วย

2.5 โครงสร้างและการควบคุมความเร็วรอบ ของมอเตอร์ไฟฟ้ากระแสตรง

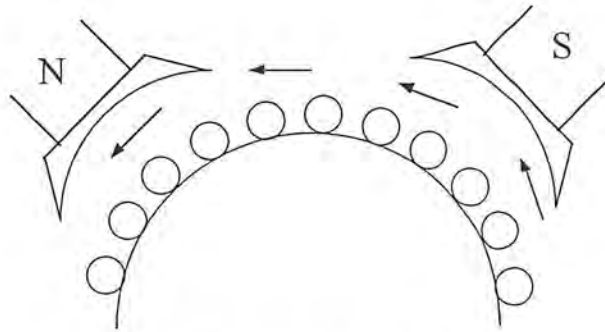
2.5.1 โครงสร้างของมอเตอร์ไฟฟ้ากระแสตรง

หลักการของมอเตอร์ก็คือ การเปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานกล เพื่อนำพลังงานกลที่ได้ไปขับเคลื่อนสิ่งต่างๆ ตามที่ต้องการ อาการทางกลที่เกิดขึ้นนี้จะอาศัยหลักการที่ว่า เมื่อมีกระแสไฟฟ้าไหลเข้าไปในตัวนำซึ่งขณะนั้นอยู่ในสนามแม่เหล็กไฟฟ้า จะทำให้เกิดแรงขึ้นในทิศทางที่สามารถหาได้จากกฎมือซ้ายของเฟลมมิ่ง ขนาดของแรงที่เกิดขึ้น คำนวณได้จากสูตร

$$F = Bil \quad \text{newton.}$$

เมื่อ	$F =$ แรงที่เกิดขึ้นบนตัวนำตัวหนึ่ง	(นิวตัน)
	$B =$ ความหนาแน่นสนามแม่เหล็ก	(เวเบอร์/เมตร ²)
	$i =$ กระแสที่ไหล ในตัวนำ	(แอมป์)
	$l =$ ความยาวของตัวนำ	(เมตร)

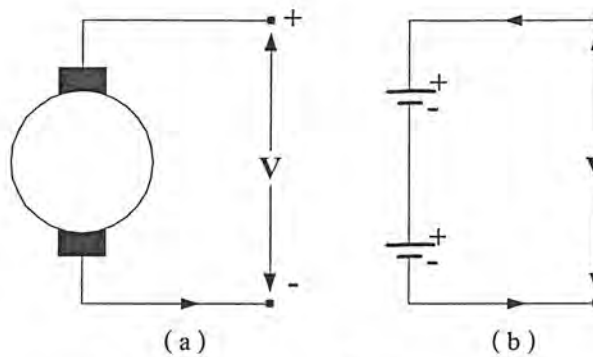
แรง F ที่เกิดขึ้นบนตัวนำ จะเกิดในแนวตั้งฉากกับสนามแม่เหล็ก และกระแสไฟฟ้าที่ไหลผ่านในตัวนำนั้นๆ



รูปที่ 2.5 หลักการของมอเตอร์ไฟฟ้ากระแสตรง

จากรูปที่ 2.5 แสดงส่วนหนึ่งของมอเตอร์ไฟฟ้ากระแสตรงที่มีหลายขั้วแม่เหล็ก เมื่อป้อนไฟเข้าไปที่สนามกระตุ้น จะทำให้เกิดอำนาจแม่เหล็กขึ้นที่ขั้วแม่เหล็ก เมื่อทำการป้อนกระแสไฟฟ้าให้ไหลผ่านในดัดนำที่อาร์เมเจอร์ (Armature) ก็จะทำให้เกิดแรงขึ้นในดัดนำที่อยู่ในสนามแม่เหล็กนั่นคือ ทำให้อาร์เมเจอร์หมุน ในที่นี้ได้กำหนดให้กระแสที่ไหลในดัดนำของอาร์เมเจอร์ที่อยู่ภายใต้ขั้วเหนือ N มีทิศของกระแสพุ่งออกมาข้างนอกดังหัวลูกศรที่เป็นจุด เมื่อเป็นเช่นนี้จึงหาทิศทางการเคลื่อนที่ของดัดนำทุกๆ ดัดนำที่อยู่ภายใต้ขั้วแม่เหล็กทั้ง N และ S โดยใช้กฎมือซ้ายของเฟลมมิ่ง วิธีหาก็คือ กางมือซ้ายออก โดยให้นิ้วหัวแม่มือ นิ้วชี้และนิ้วกลางตั้งฉากซึ่งกันและกัน (ทำแบบเดียวกับกฎมือขวาซึ่งใช้หาทิศแรงเคลื่อนไฟฟ้าที่เกิดขึ้น) จากนั้นให้นิ้วชี้ ชี้ไปตามทิศทางของสนามแม่เหล็ก คือชี้จากขั้ว N ไป S ให้นิ้วกลางชี้ไปตามทิศทางการไหลของกระแส ณ ที่ดัดนำที่ต้องการหาทิศทางการเคลื่อนที่นั่นๆ นั่นคือ นิ้วหัวแม่มือจะชี้ทิศทางการเคลื่อนที่ของดัดนำนั้นๆ จะพบว่าแรงที่เกิดขึ้นบนดัดนำทุกๆ ดัดนำ ภายใต้ขั้วแม่เหล็กเดียวกัน จะมีทิศทางไปในทางเดียวกัน และจะอยู่ในแนวสัมผัสกับเส้นรอบวงของอาร์เมเจอร์นั้น ๆ อีกทั้งยังพบว่าแรงที่เกิดขึ้นภายใต้ขั้วแม่เหล็ก ทุกขั้วที่สลับกันไปนั้นจะมีทิศทางไปในทางเดียวกันทั้งสิ้น โดยแต่ละแรงจะอยู่ในแนวสัมผัสกับเส้น รอบวงของอาร์เมเจอร์นั่นคือ ภายใต้ขั้วแม่เหล็กแต่ละขั้วก็เกิดแรงลัพธ์ขึ้นแรงหนึ่ง ฉะนั้นเมื่อมีหลายขั้วก็จะเกิดแรงขึ้นหลายแรง และทุกๆ แรงต่างก็รวมกันเป็นแรงบิดขึ้นมาแรงหนึ่ง ดังนั้นจึงทำให้มอเตอร์สามารถหมุนได้จากแรงที่เกิดขึ้น

ความสำคัญของแรงเคลื่อนไฟฟ้าสวนกลับ เมื่ออาร์เมเจอร์เริ่มหมุนจะทำให้ดัดนำที่อยู่ในอาร์เมเจอร์ตัดเส้นแรงแม่เหล็กที่มาจากสนามแม่เหล็กหลัก หรือสนามแม่เหล็กลัพท์ที่เกิดอยู่ในมอเตอร์นั้น นั่นคือเกิดการเหนี่ยวนำแรงเคลื่อนไฟฟ้าขึ้นบนดัดนำภายในอาร์เมเจอร์ ซึ่งเป็นไปตามกฎของการเหนี่ยวนำแรงเคลื่อนแม่เหล็กไฟฟ้า ในการหาทิศทางของแรงเคลื่อนไฟฟ้าที่เกิดขึ้นนี้สามารถทำการหาได้โดยใช้กฎมือขวาของ เฟลมมิ่งและพบว่าทิศทางสวนกับแรงเคลื่อนไฟฟ้าที่ป้อนเข้าไปให้กับมอเตอร์ ดังนั้น จึงเรียกแรงเคลื่อนที่เกิดขึ้นว่า แรงเคลื่อนไฟฟ้าสวนกลับ (back e.m.f = E_b) ดังรูปที่ 2.6 (a)



รูปที่ 2.6 (a) วงจรมอเตอร์ไฟฟ้ากระแสตรง

(b) วงจรสมมูลย์ของมอเตอร์ไฟฟ้ากระแสตรง

จากการที่เกิด E_b ขึ้นนี้ จึงเขียนวงจรสมมูลย์ของมอเตอร์ได้ดังรูปที่ 2.6 (b) ซึ่งเหมือนกับว่า มีแบตเตอรี่ E_b ต่อคร่อมอยู่กับแรงเคลื่อนไฟฟ้า V ที่จ่ายให้กับมอเตอร์ นั่นคือในการที่จะให้ I_a ไหลได้ในอาร์เมเจอร์นั้น จะต้องมีการจ่ายไฟฟ้าจำนวนหนึ่งที่ต้องการเอาชนะการสวนนี้ กำลังไฟฟ้าที่เอาชนะกำลังที่สวนนี้ ได้คือ $E_b I_a$ ในกรณีที่ เป็นเซลล์หรือแบตเตอรี่นั้น กำลังที่เอาชนะการสวนนี้ก็คือกำลังที่เปลี่ยนไปเป็นกำลังงานทางเคมีแต่ในมอเตอร์นั้น กำลังที่เอาชนะการสวนนี้ก็คือ กำลังที่เปลี่ยนไปเป็นกำลังกลนั่นเอง

$$\begin{aligned}
 \text{นั่นคือ กระแส } I_a \text{ จะมีค่า} &= \frac{\text{แรงเคลื่อนไฟฟ้าที่ได้ในวงจร}}{\text{ก.ต.ท. ในวงจรของอาร์เมเจอร์}} \\
 &= \frac{\text{net voltage}}{\text{armature circuit resistance}} \\
 &= \frac{V - E_b}{R_a}
 \end{aligned}$$

ตามที่ได้กล่าวมาในตอนต้นแล้วว่า แรงเคลื่อนไฟฟ้าที่เกิดขึ้น จากการหมุนของตัวนำในสนามแม่เหล็กมีค่า $E = \frac{\phi Z S P}{60 a}$ ดังนั้น $E_b = \frac{\phi Z S P}{60 a}$

พบว่า E_b ขึ้นอยู่กับตัวประกอบต่างๆ เช่น ϕ , Z , S , P และ a แต่เนื่องจาก Z , P/a เป็นค่าคงที่ในการพันอาร์เมเจอร์ของแต่ละเครื่องที่แตกต่างกันไป ดังนั้นจึงได้

$$E_b = K_c \phi S$$

$$\text{เมื่อ } K_c = \text{ค่าคงที่ทางไฟฟ้า} = \frac{Z P}{60 a}$$

S = ความเร็วเป็นรอบต่อนาที (r.p.m.)

ϕ = เส้นแรงแม่เหล็กต่อขั้ว (Wb)

จากสมการ E_b ที่กล่าวมานี้ พบว่า E_b จะมากหรือน้อยนั้น ขึ้นอยู่กับค่า ϕ และ S ฉะนั้นเพื่อให้ ϕ คงที่จะพบว่า E_b เป็นสัดส่วนโดยตรงกับความเร็ว S นั่นคือ เมื่อ S มีค่ามาก จะทำให้ E_b มีค่ามาก และย้อนทำให้ I_u มีค่าน้อย (เมื่อ V คงที่ นั่นคือผลต่างของ V E_b มีค่าน้อย เมื่อ $R_u =$ คงที่) ทำให้แรงบิด (Torque) ที่มอเตอร์จ่ายออกมามีค่าน้อยด้วยมอเตอร์ ดังนั้นจะพบว่า E_b เป็นตัวที่คอยควบคุมการไหลของกระแส I_u ในอาร์เมเจอร์ นั่นเอง นั่นคือ E_b เป็นตัวทำให้มอเตอร์มีการควบคุมอยู่ในตัวของมันเอง โดยการดึงกระแสเท่าที่จำเป็นต้องใช้เพื่อให้มอเตอร์หมุนได้ในภาวะที่ต้องการเพื่อให้เกิดการสมดุลย์ตามกฎการทรงมวลนั่นเอง

2.5.2 การควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง

ในการควบคุมความเร็วในการเคลื่อนที่ของหุ่นยนต์ สามารถทำได้โดยการปรับความเร็วรอบของมอเตอร์ แทนที่จะทำการปรับขนาดของล้อยหรืออัตราทดของเฟือง เนื่องจากสามารถทำได้ง่ายและสะดวกกว่า โดยการปรับระดับแรงดันไฟที่ป้อนให้กับตัวมอเตอร์ ความเร็วรอบของมอเตอร์นั้นจะแปรผันตามระดับแรงดันที่ทำการป้อนให้กับตัวมอเตอร์ ถ้าระดับแรงดันมีค่าสูงก็จะทำให้มอเตอร์หมุนเร็ว ในทางตรงกันข้ามถ้ามีการป้อนระดับแรงดันที่มีค่าต่ำก็จะทำให้มอเตอร์หมุนช้าลงนั่นเอง

2.6 ทฤษฎีทางแสง

ทฤษฎีมีอยู่ว่า แสงสีขาวที่สามารถมองเห็นได้นี้ประกอบไปด้วยแสงสีต่าง ๆ หลายสี ผสมกันอยู่ เมื่อแสงตกกระทบวัตถุแล้วทำให้สามารถมองเห็นวัตถุนั้นเป็นสีใดก็ได้ แสดงว่าวัตถุนั้นดูดกลืนแสงหมด ไม่มีการสะท้อนออกมาหรือสะท้อนเป็นส่วนน้อย ในทางตรงกันข้ามหากมีการเห็นวัตถุเป็นสีต่างๆได้นั้น แสดงว่าวัตถุนั้นไม่ดูดกลืนแสง ถ้ามีการมองเห็นวัตถุนั้นเป็นสีขาวแสดงว่าวัตถุนั้นสามารถสะท้อนแสงเกือบทั้งหมดออกมา ซึ่งสามารถนำหลักการสะท้อนของแสงสีนี้มาใช้ในโครงการ โดยใช้แสงอินฟราเรด (Infrared) โดยใช้อุปกรณ์รับ – ส่ง แสงอินฟราเรด ทำหน้าที่ในการตรวจสอบเส้นทางในการเดินทางได้

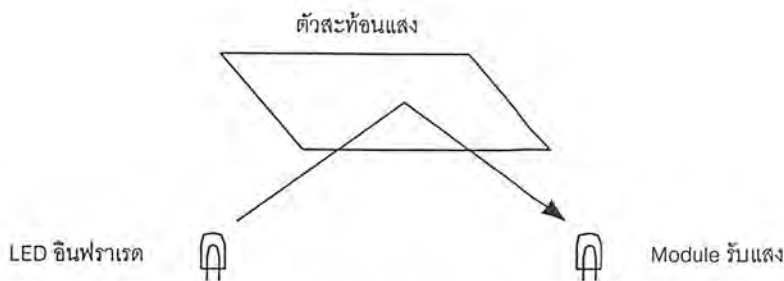
ในตัวกลางที่มีค่าดัชนีการหักเหของแสงค่าเดียวกันนั้นแสงจะเดินทางเป็นเส้นตรง ส่วนตัวกลางที่มีค่าดัชนีการหักเหของแสงต่างกัมนั้นตรงรอยต่อของตัวกลางทั้งสองแสงจะเกิดการหักเหและการสะท้อนกลับ คุณสมบัติการเดินทางเป็นเส้นตรงของแสง คุณสมบัติการหักเหของแสงและคุณสมบัติการสะท้อนกลับของแสง เรียกคุณสมบัติของแสงทั้ง 3 ข้อนี้ว่า กฎ 3 ข้อของแสง ซึ่งปรากฏการณ์เหล่านี้จะสามารถพบเห็นได้บ่อยในชีวิตประจำวัน

ความเร็วของแสงจะต่างกันในตัวกลางที่มีค่าดัชนีการหักเหต่างกัน สำหรับความเร็วของแสงในอากาศนั้นมีค่าประมาณ 3×10^8 เมตรต่อวินาที (ค่าที่ถูกต้องก็คือ 2.9979×10^8 เมตรต่อวินาที)

2.7 อุปกรณ์ทางแสง

2.7.1 อุปกรณ์ตรวจจับแบบใช้หลักการสะท้อนของแสง (Reflective optical sensors)

อุปกรณ์ในการตรวจจับแบบใช้หลักการสะท้อนของแสงมีข้อดี ในการกำหนดพื้นที่การทำงานเพื่อใช้ตรวจสอบเส้นทางได้อย่างแน่นอน ทำให้การออกแบบระบบสามารถทำได้ง่ายและเกิดความคิดพลาดน้อยมาก มีหลักการในการปล่อยแสงและรับแสงดังรูปที่ 2.7



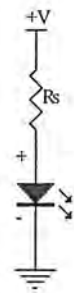
รูปที่ 2.7 ตัวตรวจจับแบบใช้หลักการสะท้อนของแสง

จากรูปที่ 2.7 จะเห็นได้ว่าส่วนของการตรวจจับเส้นทางประกอบไปด้วยแอลอีดีอินฟราเรด (Infrared LED) และโฟโตโมดูล (Photo – Modules) ซึ่งมีการจัดวางอุปกรณ์ทั้งสองในตำแหน่งที่หันเหิศทางไปในทางเดียวกัน การตรวจจับจะอาศัยวัตถุที่สะท้อนแสงได้ เช่น โลหะสะท้อนแสง ซึ่งถูกเรียกว่าตัวสะท้อนแสง (Reflector) มาวางไว้บริเวณด้านหน้าของตัวตรวจจับในระยะที่สามารถทำงานได้ ลำแสงอินฟราเรดจากแอลอีดีจะสะท้อนกลับมายังโมดูลรับแสงอินฟราเรด ส่งผลให้วงจรที่ใช้ในการตรวจจับเส้นทางแสดงเอาท์พุทลอจิกออกมา ซึ่งระดับลอจิก (Logic) ที่ได้จะมีค่าที่แน่นอน อีกทั้งการเปลี่ยนแปลงระดับ ลอจิกใช้เวลานาน ทำให้รูปคลื่นของสัญญาณที่ได้ไม่เกิดการ บิดเบี้ยว เนื่องจากการทำงานอาศัยหลักการสะท้อนของแสงทำให้มีประสิทธิภาพสูง ตัวตรวจจับประเภทนี้จึงนิยมนำมาออกแบบใช้งานกับระบบตัวตรวจจับ 2 ชุด เพราะสามารถนำมาติดตั้งกับงานเข้ารหัสที่ติดตัวสะท้อนแสงได้หลายรูปแบบ เช่น ติดตั้งในด้านเดียวกันแต่คนละตำแหน่ง หรือติดตั้งด้านหน้าหรือด้านหลังของงานเข้ารหัสก็ได้ ทำให้ประยุกต์ใช้งานได้มากขึ้น

2.7.2 อินฟราเรดแอลอีดี (Infrared LED)

แรงดันตกคร่อมที่รอยต่อ P N ของไดโอด (Diode) ต้องมีค่ามากกว่าแรงดันเทอร์ชโฮลต์จึงจะสามารถทำให้ไดโอดนำกระแสได้ สำหรับซิลิคอนไดโอดแรงดันทำงานมีค่าประมาณ 0.6 โวลต์(Volt) ส่วนแอลอีดีที่ให้แสงในย่านมองเห็นได้ ถ้าทำจากสาร GaP ซึ่งให้แสงสีเขียว จะมีค่าแรงดันทำงานประมาณ 2.1 ถึง 2.8 โวลต์ ถ้าเป็นแอลอีดีที่ทำจาก AlGaAs ให้แสงสีแดง มีแรงดันทำงาน 1.75 ถึง 2.5 โวลต์ ส่วนแอลอีดีที่ให้แสงใกล้ย่านอินฟราเรดทำจากสาร GaAs มีแรงดันทำงาน 1.5 โวลต์ โดยให้แสงที่มีความยาวคลื่น 940 นาโนเมตร และถ้าทำจาก AlGaAs จะได้แสงความยาวคลื่น 880 นาโนเมตร ที่แรงดัน 1.75 โวลต์

พลังงานที่ได้จากการเปล่งแสงของแอลอีดีหาได้จากกระแสไบอัสตรงของไดโอดและต้องระมัดระวังไม่ให้กระแสส่วนนี้มีค่าสูงจนเกิดความร้อนจะทำอันตรายต่อชิ้นไดโอดจึงจำเป็นต้องมีการต่อตัวต้านทานอนุกรมที่เหมาะสม เพื่อจำกัดกระแสทำให้สามารถใช้งานแอลอีดีได้อย่างต่อเนื่อง รูปที่ 2.8 แสดงการจำกัดกระแสเบื้องต้นของแอลอีดีโดยสามารถคำนวณได้จากสูตร



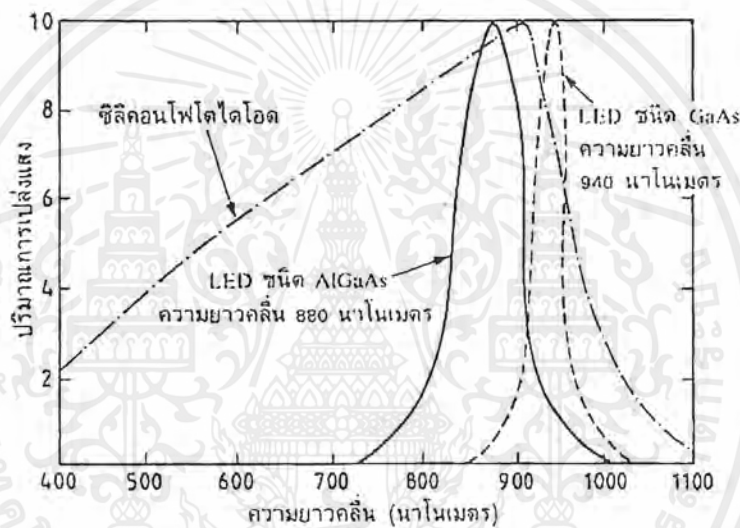
$$R_S = \frac{V_{IN} - V_{LED}}{I_{LED}}$$

+ = แอโนด

- = แคโทด

รูปที่ 2.8 การจำกัดกระแสของแอลอีดีเบื้องต้น

สิ่งที่สำคัญที่สุดของแอลอีดีอินฟราเรดกำลังงานสูงคือ ชั้นสาร AlGaAs ที่ให้ความยาวคลื่น 880 นาโนเมตร และสาร GaAs ซิลิคอนไดโอดที่ให้แสงความยาวคลื่น 940 นาโนเมตร ดังที่แสดงไว้ในรูปที่ 2.9 คือกราฟสเปกตรัมที่เปรียบเทียบความยาวคลื่นกับการเปล่งแสงของแอลอีดีทั้ง 2 ชนิด



รูปที่ 2.9 การเปล่งแสงของแอลอีดีแบบ GaAs และ AlGaAs

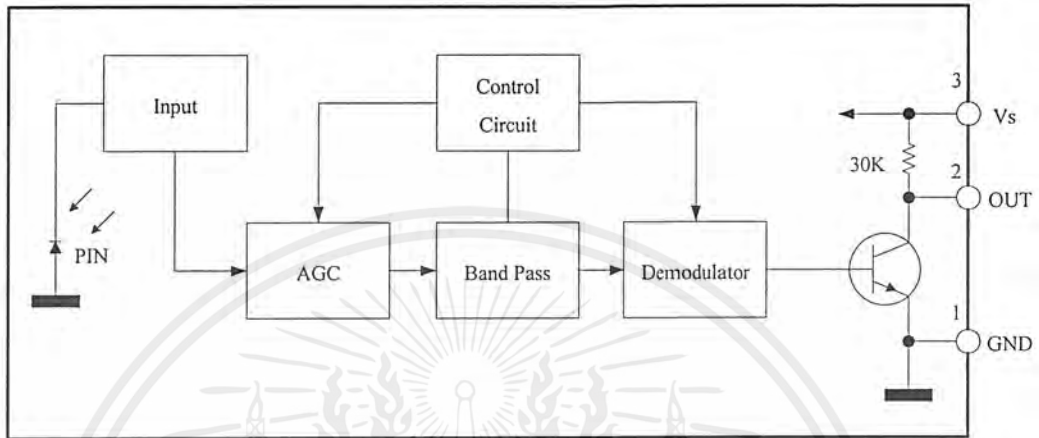
ซิลิคอนแอลอีดีที่ทำจาก GaAs ให้กำลังงานประมาณ 5 มิลลิวัตต์ ที่กระแสไบอัสตรง 100 มิลลิแอมป์ แอลอีดีที่ทำจาก AlGaAs จะให้กำลังงานเป็น 2 เท่า เมื่อให้กระแสไบอัสตรงค่าเดียวกัน ข้อที่ดีกว่าอีกประการหนึ่งของ แอลอีดีชนิด AlGaAs คือ มี rise time และ fall time ที่เร็วกว่าคือประมาณ 0.5 ไมโครวินาที ในขณะที่ GaAs ซิลิคอนไดโอดมีค่า 1.5 ไมโครวินาที

ข้อดีอีกประการหนึ่งคือ การเปล่งแสงแอลอีดีที่ความยาวคลื่น 880 นาโนเมตร (AlGaAs) จะใกล้เคียงกับความยาวคลื่นที่ซิลิคอนโฟโตทรานซิสเตอร์ (Photo Transistor) มีความไวสูงสุดจึงเป็นการเหมาะสมที่จะใช้แอลอีดีที่มีความยาวคลื่น 880 นาโนเมตร แทนแอลอีดีที่มีความยาวคลื่น 940 นาโนเมตร

นอกจากนั้นแอลอีดีที่ให้ความยาวคลื่นแสง 880 นาโนเมตร ยังไม่ถูกดูดกลืนโดยละอองน้ำเหมือนแอลอีดีที่ให้ความยาวคลื่น 940 นาโนเมตร จึงสามารถนำไปใช้ในการตรวจจับไอน้ำในอากาศแอลอีดีชนิด 940 นาโนเมตร ไม่เหมาะกับการสื่อสารด้วยแสงภายนอกเพราะจุดอ่อนเรื่องการถูกดูดกลืนด้วยไอน้ำในอากาศนั่นเอง ส่วนแอลอีดีชนิดซิลิคอนที่ทำจาก GaAs มักจะใช้เป็นแหล่งกำเนิดแสงย่านอินฟราเรด

2.7.3 โฟโต้โมดูล (Photo – Modules)

โฟโต้โมดูล เบอร์ TSOP 4838 ถูกออกแบบมาเพื่อใช้ในการรับสัญญาณอินฟราเรด โดยมีการใช้งานร่วมกับแอลอีดีอินฟราเรด ซึ่งอาศัยสัญญาณดิจิทัลทางด้านอินพุตไปควบคุมวงจรทางด้านเอาต์พุตภายในตัวโฟโต้โมดูลประกอบไปด้วย Pin Diode และวงจร Pre-Amplifier มีโครงสร้างภายในดังรูปที่ 2.10

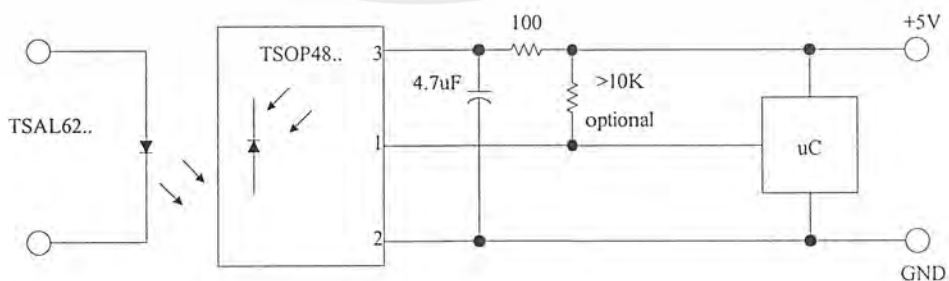


รูปที่ 2.10 โครงสร้างภายในของโฟโต้โมดูล

คุณสมบัติของโฟโต้โมดูล

- มีวงจร Photo Detector และ Pre-Amplifier ภายในตัว
- มีวงจร Band Pass Filter สำหรับกรองความถี่ใช้งาน
- รักษาระดับการทำงานของวงจรต่อสภาพแวดล้อมที่มีการเปลี่ยนแปลงของแสง
- เอาต์พุตทำงานที่ลอจิก "0"
- สามารถส่งข้อมูลต่อเนื่องได้ 800 บิต/วินาที
- Burst length ที่เหมาะสมในการใช้งานต้องไม่ต่ำกว่า 10 cycle/burst

เอาต์พุตที่ได้จากโฟโต้โมดูลสามารถนำไปต่อใช้งานกับไมโครคอนโทรลเลอร์ได้โดยตรง มีลักษณะการต่อวงจรตามรูปที่ 2.11



รูปที่ 2.11 การต่อใช้งาน

2.7.4 สัญญาณโทนเบิร์สต์ (Tone Burst)

สัญญาณโทนเบิร์สต์เป็นสัญญาณที่มีการดัดแปลงรูปแบบให้มีลักษณะคล้ายกับการส่งคลื่นวิทยุ โดย ลักษณะสัญญาณแบบโทนเบิร์สต์แสดงดังรูปที่ 2.12 (ข) ซึ่งสร้างจากสัญญาณพัลส์ที่มีสภาวะสูงของสัญญาณควบคุมแบบธรรมดาในรูปที่ 2.12 (ก) สัญญาณโทนเบิร์สต์ประกอบด้วยพัลส์ความถี่สูงแบบต่อเนื่องตลอดช่วงความกว้างของบิตที่เป็น “1” ในขณะที่บิตข้อมูลอยู่ในสภาวะต่ำสัญญาณจะคงเดิมไม่มีการเปลี่ยนแปลงแต่อย่างใด



(ก)



(ข)

รูปที่ 2.12 (ก) สัญญาณพัลส์ธรรมดา

(ข) สัญญาณโทนเบิร์สต์

สำหรับวงจรรับต้องทำการออกแบบให้มีการตอบสนองต่อสัญญาณโทนเบิร์สต์ในช่วงความถี่ที่เหมาะสมกัน จึงสามารถออกแบบชุดรีโมตคอนโทรล ที่มีค่าความถี่ของโทนเบิร์สต์แตกต่างกัน เพื่อนำมาควบคุมอุปกรณ์ในพื้นที่ใกล้เคียงกันได้ โดยปราศจากการรบกวนซึ่งกันและกัน

ในขณะเดียวกัน ตัววงจรส่งอาจถูกออกแบบให้มีการเปลี่ยนแปลงค่าความถี่ของสัญญาณโทนเบิร์สต์เป็นค่าต่างๆ ได้ซึ่งช่วยให้สามารถใช้เครื่องส่งเพียงตัวเดียวสำหรับการควบคุมอุปกรณ์ไฟฟ้าหลายชนิดได้พร้อมกัน

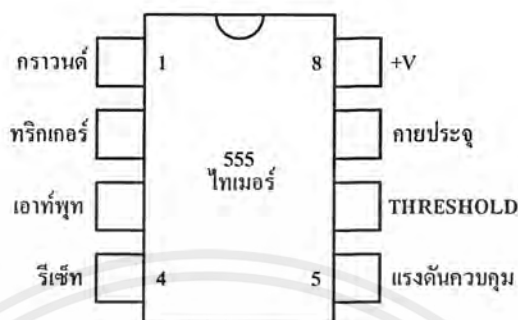
ข้อดีอีกประการของการส่งสัญญาณแบบโทนเบิร์สต์ ก็คือสามารถขจัดสัญญาณรบกวนจากภายนอกได้ดีมาก ตัวอย่างเช่น ระบบควบคุมที่ใช้สัญญาณแสง หรือแสงอินฟราเรดที่ส่งด้วยพัลส์ธรรมดา อาจมีแสงจากภายนอก แสงจากหลอดไฟ หรือแม้กระทั่งแสงอาทิตย์เข้าไปรบกวนที่เครื่องรับจนอาจเกิดข้อผิดพลาดในการรับสัญญาณควบคุมได้ แต่ถ้าหากมีการใช้สัญญาณโทนเบิร์สต์แล้ว สิ่งรบกวนต่างๆ เหล่านี้จะถูกขจัดออกไปโดยสิ้นเชิง

ในการส่งสัญญาณโทนเบิร์สต์ จะต้องมีการเพิ่มวงจรกำเนิดสัญญาณความถี่สูงเข้ามาในวงจรส่งสัญญาณควบคุมด้วย เนื่องจากสิ่งที่ถูกเพิ่มเข้ามาจากสัญญาณพัลส์ธรรมดา ก็คือ พัลส์ความถี่สูง โดยใช้ไอซีช็อคเนียมเบอร์ 555 ต่อเป็นวงจรอะสเตเบิลมัลติไวเบเรเตอร์ ซึ่งสามารถนำเอาที่พิกัดที่ได้ไปขับแอลอีดีอินฟราเรดได้โดยตรง หรือหากต้องการเพิ่มกำลังส่งก็อาจใช้ทรานซิสเตอร์ขยายก่อนนำไปขับแอลอีดีอินฟราเรดก็ได้

2.7.5 หลักการทำงานของ 555 ไทเมอร์

ไอซี 555 ไทเมอร์ทันนี้มีอยู่หลายเบอร์ตามชนิดของลักษณะเฉพาะต่างๆ แต่ปกติจะเรียกรวมกันง่ายๆ ว่า “555 ไทเมอร์” อุปกรณ์นี้มีการบรรจุเป็นหลายรูปแบบซึ่งรวมทั้งแบบ DIL – 8 และ 14 ขา และแบบ TO – 99 มี 8 ขา ในที่นี้วงจรทั้งหมดจะออกแบบโดยใช้อุปกรณ์แบบ DIL – 8 ขา

รูปที่ 2.13 แสดงลักษณะและรายละเอียดแต่ละขาของไอซีแบบ DIL – 8 และรูปที่ 2.13 แสดงแผนภาพการทำงานของอุปกรณ์ (ภายในเส้นคู่) พร้อมทั้งการต่อสำหรับใช้ไอซีเป็นตัวกำเนิดสัญญาณโมโนสเตเบิลพื้นฐานการอธิบายการทำงานของอุปกรณ์ดังที่จะกล่าวต่อไปนี้ สมมติว่าไทเมอร์ถูกใช้ในลักษณะโมโนสเตเบิลตามที่แสดงไว้ในรูปที่ 2.14



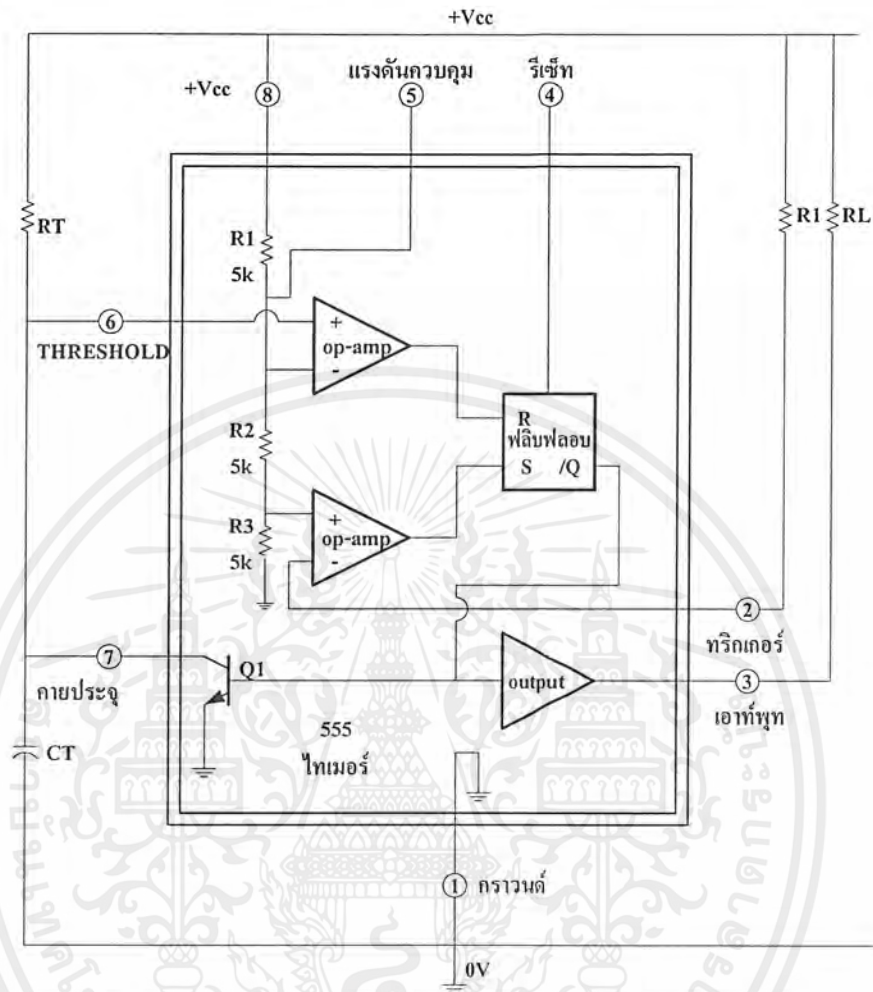
รูปที่ 2.13 รายละเอียดแต่ละขาของไอซี 555 ไทเมอร์แบบ DIL – 8 ขา มาตรฐาน

555 ไทเมอร์บรรจุไดโอด 2 ตัว, ตัวต้านทาน 15 ตัว และทรานซิสเตอร์ 23 ตัว ส่วนประกอบเหล่านี้ถูกจัดในลักษณะที่มีตัวแบ่งศักดาแรงดันอ้างอิงหนึ่งตัว, ออป-แอมป์เปรียบเทียบแรงดันสองตัว, R – S ฟลิป-ฟลอป หนึ่งตัว, ภาควิทยุแบบคอมพลิเมนต์ที่ทำงานที่ค่าหนึ่งตัว และทรานซิสเตอร์สแตฟอีกหนึ่งตัว ตัวแบ่งศักดาแรงดันอ้างอิงประกอบด้วยตัวต้านทาน 5 K Ω สามตัวต่ออนุกรมกัน และถูกต่อกราวด์ไฟเลี้ยงของไอซี ดังนั้นสองในสามของ V_{CC} จะเกิดที่จุดต่อของตัวต้านทานสองตัวบนของตัวแบ่งศักดาและถูกป้อนเข้าขาอินพุตขาหนึ่งของออป-แอมป์เปรียบเทียบแรงดันตัวบนและหนึ่งในสามของ V_{CC} จะเกิดที่จุดต่อของตัวต้านทานสองตัวล่างของตัวแบ่งศักดา และถูกป้อนเข้าขาอินพุตขาหนึ่งของออป-แอมป์เปรียบเทียบแรงดันตัวล่าง เอาต์พุตของตัวเปรียบเทียบทั้งสองจะควบคุม R – S ฟลิป-ฟลอป ซึ่งจะควบคุมสถานะของภาควิทยุแบบคอมพลิเมนต์และทรานซิสเตอร์สแตฟ สถานะของฟลิป-ฟลอปสามารถถูกควบคุมโดยสัญญาณที่ต่อกับขา 4 รีเซ็ตของไอซีอีกด้วย

เมื่อวงจรโมโนสเตเบิลหรือเวลาในรูปที่ 2.14 อยู่ในสถานะคงที่ไม่ได้ขับโหลด ขา 2 ขาทริกเกอร์ของไอซีจะถูกทำให้เป็นสถานะสูงด้วย R_1 ภายใต้เงื่อนไขนี้ Q_1 จะถูกขับให้อิ่มตัว และเกิดการลัดวงจรคร่อมตัวเก็บประจุเวลาภายนอก C_T และขา 3 ขาเอาต์พุตของไอซีจะถูกขับให้อยู่ในสถานะต่ำ การทำงานแบบโมโนสเตเบิลสามารถเริ่มต้นได้โดยการต่อพัลส์ทริกเกอร์ทำงานที่ขา 2 ของไอซี เมื่อพัลส์ที่ป้อนนี้มีค่าต่ำกว่าหนึ่งในสามของ V_{CC} ซึ่งเป็นค่าอ้างอิงของตัวแบ่งศักดาภายใน เอาต์พุตของออป-แอมป์ตัวเปรียบเทียบแรงดันตัวล่างก็จะเปลี่ยนสถานะ และทำให้ R – S ฟลิป-ฟลอปสวิตช์ตรงกันข้ามซึ่งจะคัท-ออฟ Q_1 และขับขา 3 เอาต์พุตของไอซีเป็นสถานะสูง

เมื่อ Q_1 คัท-ออฟตัวเก็บประจุเวลา C_T ก็จะไม่ใช่ถูกลัดวงจร และเริ่มที่จะประจุในลักษณะเอ็กโพเนนเชียลเพิ่มขึ้นสู่ค่าแรงดันของไฟเลี้ยงจนกระทั่ง C_T ประจุถึง $2/3 V_{CC}$ ที่จุดนี้ออป-แอมป์ตัวเปรียบเทียบตัวบนเปลี่ยนสถานะและสวิตช์ให้ R – S ฟลิป-ฟลอปกลับคืนสู่สถานะเดิม ดังนั้นถ้า Q_1 ทำงานและ C_T ภายประจุอย่างรวดเร็วแล้ว ขา 2 เอาต์พุตของไอซีก็จะกลับสู่สถานะต่ำในทันที ลำดับการทำงานของโมโนสเตเบิลก็จะเสร็จสมบูรณ์ สังเกตว่าทันทีที่ทริก วงจรจะไม่ตอบสนองต่อการทริกที่เพิ่มต่อเข้าไปจนกว่าค่าดับ

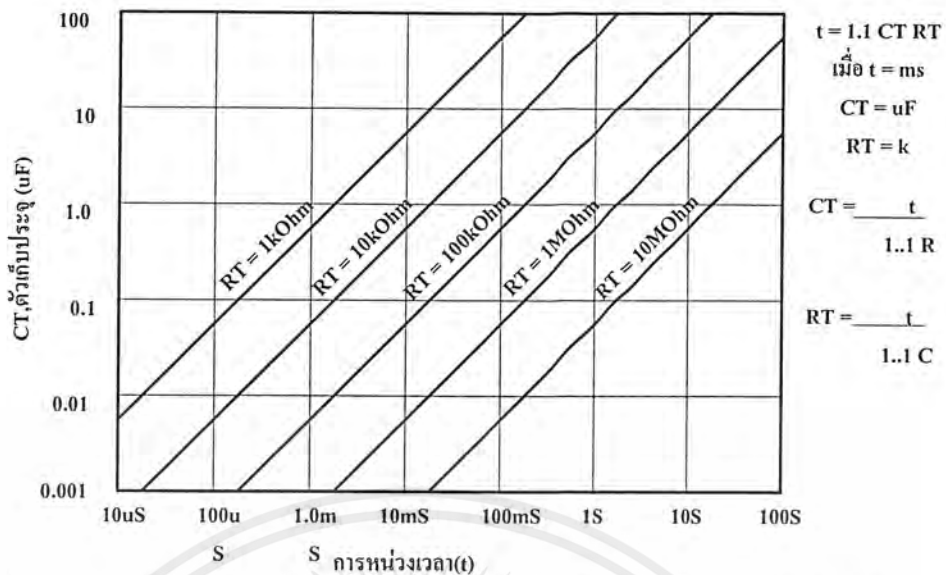
เวลาของชุดแรกจะเสร็จสิ้น แต่ว่าลำดับนี้สามารถจะหลุดเมื่อใดก็ได้ โดยการป้อนพัลส์ทำงานขาลงให้กับขา 4 ของไอซี



รูปที่ 2.14 การทำงาน (ภายในสี่เหลี่ยม) ของไอซี 555 ไทมเมอร์ซึ่งต่อเพื่อ
ใช้ไอซีเป็นตัวกำเนิดโมโนสเตเบิลหรือไทมเมอร์พื้นฐาน
เวลาหนึ่งของวงจรซึ่งหา 3 เอาท์พุทจะเป็นสถานะสูง กำหนดโดย

$$t = 1.1 R_T C_T$$

เมื่อ $t = ms$, $R_T = K\Omega$ และ $C_T = \mu F$ รูปที่ 2.15 แสดงถึงการหน่วงเวลาจาก $10\Omega S$ ถึง 100 วินาที สามารถทำได้โดยการเลือกค่า C_T และ R_T ที่เหมาะสมในช่วง $0.001 \mu F$ ถึง $100 \mu F$ และ $1 K\Omega$ ถึง $1 M\Omega$ ในทางปฏิบัติ R_T ไม่ควรมีค่าต่ำกว่า $1 K\Omega$ หรือมากกว่า $20 M\Omega$ และตัวเก็บประจุ C_T ควรใช้แบบกระแสรั่วไหลน้อย ทั้งนี้คาบเวลาของวงจรไม่ขึ้นอยู่กับค่าของแรงดันไฟเลี้ยงของวงจร แต่คาบเวลาสามารถปรับได้โดยการต่อคสามด้านทานหรือแรงดันปรับค่าได้ระหว่างกราวด์กับขา 5 แรงดันควบคุมของไอซี ข้อดีนี้ทำให้คาบเวลาสามารถถูกมอดูเลทหรือชดเชยได้จากภายนอก

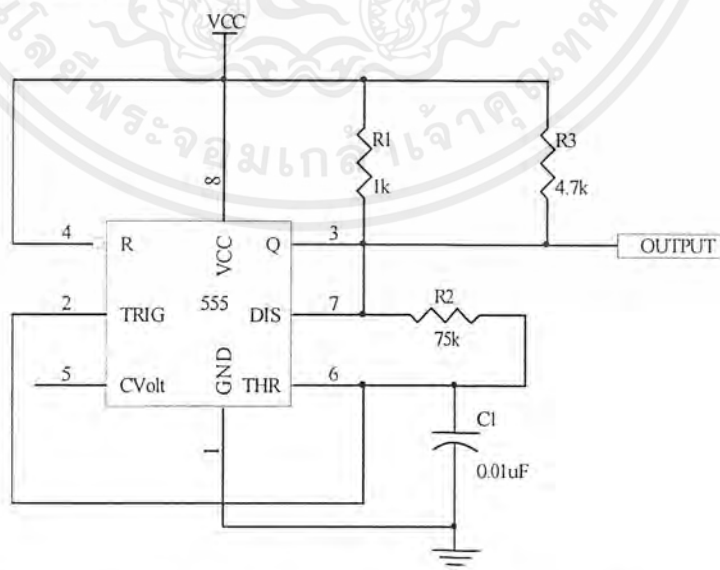


รูปที่ 2.15 การหน่วงเวลาของ 555 ไทเมอร์สำหรับค่าความต้านทานและค่าตัวเก็บประจุต่างๆ

ขา 3 ซึ่งเป็นขาเอาต์พุตของไอซีนั้นปกติจะมีสถานะต่ำ แต่จะสวิตช์เป็นสถานะสูงในระหว่างที่ลำดับโมโนสเตเบิลทำงาน เอาต์พุตสามารถจ่ายหรือรับกระแสได้ถึงค่าสูงสุด 200 mA ดังนั้นภาระภายนอกสามารถต่อระหว่างขา 3 กับไฟเลี้ยงบวกหรือกราวด์ได้อย่างใดอย่างหนึ่งซึ่งขึ้นอยู่กับชนิดการทำงานของภาระตามที่ต้องการ เวลาช่วงขึ้นและช่วงลงที่เอาต์พุตสวิตช์ปกติจะมีค่าประมาณ 100 นาโนวินาที

2.7.6 วงจรอะอสเตเบิล มัลติไวเบเรเตอร์

ไอซี 555 ไทเมอร์ สามารถใช้ทำเป็นอะอสเตเบิล มัลติไวเบเรเตอร์หรือตัวกำเนิดคลื่นรูปสี่เหลี่ยมได้ดีมาก ซึ่งมีความใช้งานถึง 100 KHz 555 อะอสเตเบิลมีเสถียรภาพของวงจรคือ ดิวตี้ไซเคิล (duty cycle) สามารถปรับได้และสัญญาณเอาต์พุต สามารถถูกควบคุมเพื่อใช้ในการมอดูเลชันความถี่ (frequency modulation) ได้โดยขา 5 ของไอซี



รูปที่ 2.16 วงจรอะอสเตเบิล มัลติไวเบเรเตอร์พื้นฐานกำเนิดความถี่ 1 KHz

รูปที่ 2.16 แสดงวงจรออสซิลเลเตอร์ มัลติไวเบเรเตอร์ 555 พื้นฐาน ความถี่ 1 KHz ซึ่งสามารถที่จะคำนวณหาเวลา t_{on} เวลา t_{off} คาบเวลาและความถี่ ได้จากสูตรดังนี้

$$t_{on} = 0.693(R1+R2)C1$$

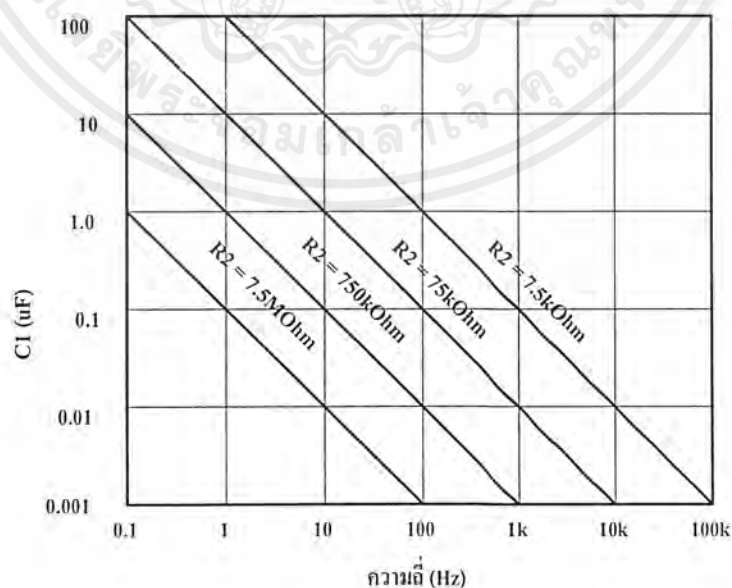
$$t_{off} = 0.693(R2)C1$$

$$T = 0.639(R1+2R2)C1$$

$$f = 1.44/[(R1+2R2)C1]$$

การทำงานของวงจรเมื่อจ่ายไฟเลี้ยงเข้าวงจร C1 จะเริ่มเก็บประจุแบบเอ็กโพเนนเชียล (exponential) ในลักษณะโมโนสเตเบิล (monostable) ปกติ โดยผ่าน R1 และ R2 ที่ต่ออนุกรมกันจนกระทั่ง C1 มีค่าแรงดันถึง $2/3V_{cc}$ ในตอนนี้การทำงานลักษณะโมโนสเตเบิลของไอซีก็จะสิ้นสุดลงและขา 7 ซึ่งเป็นขาขยายประจุจะสวิทช์ไปสู่สถานะต่ำ ดังนั้น C1 ก็จะเริ่มคายประจุแบบเอ็กโพเนนเชียลเข้าสู่ขา 7 ของไอซีโดยผ่าน R2 จนกระทั่งแรงดัน C1 ลดลงถึง $1/3V_{cc}$ และขา 2 ซึ่งเป็นขาทริกเกอร์ (trigger) ของไอซีก็จะทำงานในตอนนี้ลำดับเวลาการทำงานแบบโมโนสเตเบิล ก็จะเริ่มใหม่ และ C1 ก็จะเริ่มเก็บประจุอีกครั้งหนึ่งสู่ค่า $2/3V_{cc}$ โดยผ่าน R1 และ R2 ลำดับการทำงานทั้งหมดจะเกิดขึ้นซ้ำๆกันไปเรื่อยๆด้วยการเก็บประจุของ C1 สู่ค่า $2/3V_{cc}$ โดยผ่าน R1 - R2 และการคายประจุของ C1 สู่ค่า $1/3V_{cc}$ โดยผ่าน R2

ในวงจรรูปที่ 2.16 ถ้าค่า R2 มีค่ามากเมื่อเทียบกับ R1 ความถี่การทำงานของวงจรก็จะกำหนดโดยค่าของ R2 และ C1 และรูปคลื่นเอาต์พุตที่สมมาตรก็จะเกิดขึ้น รูปที่ 2.17 แสดงกราฟความสัมพันธ์โดยประมาณระหว่างความถี่และค่า C1 - R2 ภายในเงื่อนไขข้างบน ในทางปฏิบัติค่า R1 และ R2 ของวงจรจะเปลี่ยนแปลงจาก $1\text{ K}\Omega$ ไปจนถึงหลายสิบล้านโอห์ม (megaohm) อย่างไรก็ตามค่าค่าของ R1 ก็มีความสำคัญต่อการใช้กระแสทั้งหมดของวงจร เนื่องจากขา 7 ของไอซีถูกต่อลงกราวด์ ระหว่างครั้งหนึ่งของลำดับการตั้งเวลา และคิวต์ไซเคิลหรืออัตราส่วนของสัญญาณต่อสัญญาณว่างของวงจรสามารถตั้งค่าที่ทำให้รูปคลื่นสมมาตร โดยการเลือกค่า R1 และ R2 ที่เหมาะสมตามต้องการ



รูปที่ 2.17 ความสัมพันธ์ระหว่าง C1, R2 และความถี่ โดยที่ R2 มีค่าสูงมากเมื่อเทียบกับ R1

2.8 สเตปปีงมอเตอร์ (Stepping Motor)

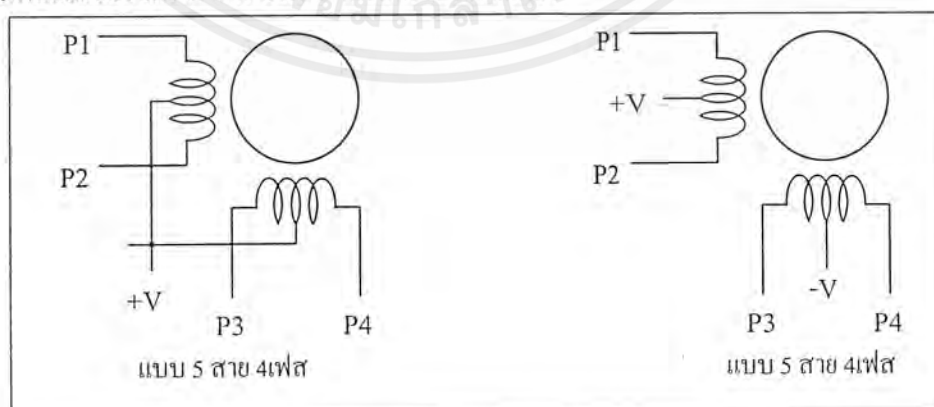
สเตปปีงมอเตอร์ (Stepping motor) เป็นมอเตอร์ที่มีลักษณะการทำงานแตกต่างจากมอเตอร์ทั่วไป เพราะจะต้องป้อนสัญญาณเป็นพัลส์ (Pulse) ให้แก่ขดลวดของมอเตอร์เป็นจังหวะอย่างเหมาะสม และการหมุนของมอเตอร์ชนิดนี้จะหมุนเป็นจังหวะตามพัลส์ที่ป้อนเข้ามา ไม่หมุนต่อเนื่องเหมือนกับมอเตอร์ธรรมดา ทำให้ผู้ควบคุมสามารถเลือกตำแหน่งที่ต้องการให้มอเตอร์หยุดหมุนได้ จังหวะการหมุนของสเตปปีงมอเตอร์ เรียกว่า สเต็ป (Step) ความละเอียดของมอเตอร์กำหนดเป็นองศาที่หมุนไปในหนึ่งสเต็ป หากมอเตอร์มีจำนวนองศาต่อสเต็ปมาก หมายความว่า มอเตอร์ตัวนี้มีความละเอียดของการหมุนต่ำ ยกตัวอย่างการหมุนครบ 1 รอบเท่ากับ 360 องศา หากมอเตอร์มีสเต็ปการหมุนเท่ากับ 7.5 องศาต่อสเต็ป มอเตอร์ตัวนี้มีความละเอียดของการหมุนเท่ากับ 48 ตำแหน่ง แต่ถ้าหากมีสเต็ปการหมุนเท่ากับ 1.8 องศาต่อสเต็ป ความละเอียดของการหมุนเท่ากับ 200 จะเห็นได้ว่ามอเตอร์ตัวหลังมีความละเอียดสูงกว่าตัวแรกมาก ทำให้นำมาใช้ในงานที่ต้องการกำหนดตำแหน่งได้ดีกว่า แน่นอนว่า พนักเข้ากับวงจรขับเคลื่อนแบบครึ่งสเต็ป ความละเอียดของการหมุนจะเพิ่มขึ้นอีก 2 เท่า ทำให้มีความละเอียดของการหมุนกลายเป็น 400 ตำแหน่ง

ขนาดของสเตปปีงมอเตอร์ ที่มีการผลิตและจำหน่ายในท้องตลาด มีตั้งแต่ขนาดแรงดันต่ำ 3 V ไปจนถึง 24 V ส่วนขนาดของกระแสมีตั้งแต่ไม่กี่สิบลิลลิแอมป์เป็นสเตปปีงมอเตอร์มอเตอร์ตัวเล็กไปจนถึงเป็นสิบลแอมป์ ซึ่งแน่นอนขนาดของมอเตอร์ย่อมต้องใหญ่โตขึ้นตามลำดับ

2.8.1 สเตปปีงมอเตอร์ แบบยูนิโพลาร์ (Uni-Polar Stepping Motor)

สเตปปีงมอเตอร์ รับการพัฒนาอย่างต่อเนื่อง จนในปัจจุบันสเตปปีงมอเตอร์ที่นิยมใช้อย่างแพร่หลายมากที่สุดและหาได้ง่ายคือ สเตปปีงมอเตอร์แบบยูนิโพลาร์ (Uni-polar motor) มีลักษณะการพันขดลวดของมอเตอร์แสดงในรูปที่ 2.18

สเตปปีงมอเตอร์แบบนี้มีการพันขดลวด 2 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ (Stator) แต่ละขดแบ่งเป็น 2 เฟส รวมมอเตอร์ทั้งตัวจะมี 4 เฟสคือ เฟส 1, 2, 3 และ 4 มีการต่อสายออกมาจากขดลวดแต่ละขดเพื่อจ่ายไฟเลี้ยง ทำให้สเตปปีงมอเตอร์แบบนี้มีทั้งแบบ 5 สาย และ 6 สาย ถ้าเป็นแบบ 5 สาย จะเป็นการนำสายไฟเลี้ยงของขดลวดทั้งสองมาต่อรวมกันเป็นสายเดียว



รูปที่ 2.18 โครงสร้างของสเตปปีงมอเตอร์

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

ตารางที่ 2.1 รูปแบบการขับเคลื่อนมอเตอร์แบบฟูลสเต็ป 1 เฟส

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

ตารางที่ 2.2 รูปแบบการขับเคลื่อนมอเตอร์แบบฟูลสเต็ป 2 เฟส

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

ตารางที่ 2.3 รูปแบบการขับเคลื่อนมอเตอร์แบบฮาล์ฟสเต็ป

2.8.2 การกระตุ้นและควบคุมการหมุนของมอเตอร์

การกระตุ้นและควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ ซึ่งต้องป้อนเป็นลำดับตามรูปแบบที่ถูกต้อง สามารถแบ่งได้เป็น 3 รูปแบบคือ แบบฟูลสเต็ปหนึ่งเฟส (Full-step 1 phase), แบบฟูลสเต็ป 2 เฟส (Full-step 2 phase) และแบบฮาล์ฟสเต็ป (Half step)

แบบฟูลสเต็ปหนึ่งเฟสเป็นการกระตุ้นที่มีรูปแบบง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งไถ่เรียงถัดกันไป เช่น เริ่มต้นที่ขดที่ 1, 2, 3, 4, แล้ววนกลับมาขดที่ 1 วนไปเรื่อยๆ หรือเริ่มที่ขดที่ 1 แล้วย้อนไปที่ขด 4, 3, 2 แล้วกลับมาขดที่ 1 อีกครั้ง ซึ่งทำให้ทิศทางของการหมุนสวนกัน ในการกระตุ้นรูปแบบนี้จึงมีขดลวดเพียงขดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้นวงจรกระตุ้นแบบเวฟ (Wave) จึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่างๆ แสดงดังตารางที่ 2.1

แบบพูลสเต็ป 2 เฟสมีลักษณะคล้ายแบบหนึ่งเฟส แต่จะทำการกระตุ้นโดยจ่ายแรงดันไปที่ขดลวด 2 ขด ที่อยู่ใกล้กันในเวลาเดียวกัน และเรียงถัดกันไปเช่นเดียวกับแบบหนึ่งเฟส ขดลวดชุดแรกที่ถูกกระตุ้นจะเป็นขดที่ 1 และ 2 ตามด้วยการกระตุ้นขดที่ 2 และ 3 ต่อไปเป็นขดที่ 3 และ 4 ถัดไปเป็นขดที่ 4 และ 1 แล้วกลับมาที่ขดที่ 1 และ 2 วนไปตามลำดับเช่นนี้หรือเริ่มที่ขด 1 และ 4 ตามด้วยขดที่ 4 และ 3 ถัดไปเป็นขดที่ 3 และ 2 ต่อไปเป็นขดที่ 2 และ 1 แล้ววนกลับมาที่ขดที่ 1 และ 4 ทิศทางการหมุนจะสวนทางกัน ดังแสดงขั้นตอนการกระตุ้นในตารางที่ 2.2 การกระตุ้นสเต็ปปั๊มมอเตอร์แบบนี้สามารถเพิ่มแรงบิดหรือทอร์ก (Torque) ได้มากกว่าแบบหนึ่งเฟส แกนแม่เหล็กเคลื่อนที่หรือโรเตอร์ (Rotor) ภายในมอเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจาก 2 ขดลวดที่ถูกกระตุ้นพร้อมกัน และหมุนต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียคือ การกระตุ้นแบบนี้คือ ใช้กำลังไฟฟ้ามากขึ้น

แบบครึ่งสเต็ปเป็นรูปแบบที่ผสมผสานระหว่างการกระตุ้นแบบหนึ่งเฟสและ 2 เฟสเพื่อเพิ่มจำนวนของสเต็ปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเป็นลำดับดังนี้ เริ่มจากขดลวดที่ 1, 1 และ 2, 2, 2 และ 3, 3, 3 และ 4, 4, 4 และ 1 แล้ววนกลับมาขดลวดที่ 1 อีก แสดงขั้นตอนการกระตุ้นดังตารางที่ 2.3 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้น เนื่องจากช่วงของการเคลื่อนที่ในแต่ละสเต็ปมีระยะสั้นลง แต่ละสเต็ปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกันความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น แต่ต้องพึงระวังไว้อีกประการหนึ่งว่า เมื่อกระตุ้นให้ทำงานในรูปแบบนี้จะต้องการหมุนถึง 2 สเต็ป จึงจะได้เท่ากับระยะเท่ากับการกระตุ้นแบบพูลสเต็ป 1 สเต็ป สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้ขนาดเท่ากับแบบ 2 เฟสเป็นอย่างน้อยจึงจะเพียงพอ

2.9 โมดูลแสดงผลแบบผลึกเหลว (LCD Module)

2.9.1 รายละเอียดเกี่ยวกับโมดูล LCD

ในโมดูล LCD จะมีส่วนประกอบหลัก ๆ 3 ส่วน ดังนี้

ตัวแสดงผล (Display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็น โดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองที่แสดงผลบนจอ LCD

ตัวควบคุม (Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะชิป (Chip) ที่นิยมใช้คือ เบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD แบบกราฟิก

ตัวขับ (Driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้เป็นทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น

2.9.2 โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ภายในของชิปควบคุม LCD เบอร์ HD44780 ซึ่งใช้ใน โมดูล LCD แบบอักขระ ประกอบด้วย

บัฟเฟอร์อินพุตเอาต์พุต (Buffer Input/Output) เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจิสเตอร์คำสั่ง (Instruction Register : IR) เป็นรีจิสเตอร์ที่รับข้อมูลคำสั่งจากอุปกรณ์ภายนอกเพื่อนำไปควบคุมการแสดงผล

รีจิสเตอร์ข้อมูล (Data Register : DR) เป็นรีจิสเตอร์ที่รับข้อมูลจากอุปกรณ์ภายนอกเพื่อถ่ายทอดไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผล หรือนำข้อมูลไปสร้างตัวอักษรเพื่อเติมในแรมเก็บตัวอักษร

แรมเก็บข้อมูลแสดงผล (Display Data RAM : DDRAM) เป็นหน่วยความจำแรมทำหน้าที่เก็บข้อมูลที่มาจากรีจิสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดตาราง (Look up-table) ของตัวอักษรที่เก็บไว้ในหน่วยความจำรอม (Rom) และแรม (Ram) เก็บตัวอักษร เพื่อนำไปแสดงที่ตัวแสดงผล

รอมเก็บตัวอักษร (Character Generator ROM : CGROM) เป็นหน่วยความจำรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปแสดงที่ตัวแสดงผลได้ มีขนาด 7,200 บิต โดยจะถูกอ่านด้วยค่าของข้อมูลใน DDRAM

แรมเก็บตัวอักษร (Character Generator RAM : CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGROM ไม่เพียงพอ มีขนาด 512 บิต การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGRAM เอง

แฟล็ก (BUSY) เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่า ตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟล็ก BUSY นี้เสียก่อน

โมดูล LCD ขนาด 16x1 มีขาต่อใช้งานทั้งสิ้น 14 ขา รายละเอียดการทำงานของแต่ละขามีกดังนี้

V_{SS} (ขา 1) : ต่อกราวด์

V_{DD} (ขา 2) : ต่อไฟเลี้ยง + 5 โวลต์

V_O (ขา 3) : เป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา 4) : เป็นขาอินพุตใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี้เป็น “0” ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

$R/\bar{W}$ (ขา 5) : เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับ LCD ถ้าเป็น “0” เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น “1” จะเป็นการอ่านข้อมูล

E (ขา 6) : เป็นขาอินพุต LCD ให้ทำงาน

D0-D7 (ขา 7-14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต

2.9.3 คำสั่งควบคุมโมดูล LCD

ในการเขียนคำสั่งลงในตัวควบคุม แนะนำว่าต้องกำหนดให้ขา RS และ $R/\bar{W}$ เป็น “0” แล้วเขียนคำสั่งตามไป คำสั่งควบคุมโมดูล LCD ของชิปควบคุม HD44780 ที่สำคัญมี 10 คำสั่งดังนี้

1. คำสั่งเคลียร์ตัวแสดงผล (Clear display)

มีข้อมูลคำสั่งเป็น 01H เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่าง หรือ Space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมเอ็กซ์คิวิต์ คำสั่งนี้ จะทำการกำหนดแอดเดรสของ DDRAM เป็น 0 เคอร์เซอร์ (Cursor) จะกลับไปอยู่ที่ตำแหน่งซ้ายมือสุดของจอแสดงผล แล้วเซตบิต I/D ให้เป็น “1”

2. คำสั่ง Return Home

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น “1” เป็นคำสั่งให้เคอร์เซอร์เคลื่อนที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอแสดงผลไม่เปลี่ยนแปลงนั่นคือ ข้อมูลคำสั่งของคำสั่งนี้จะเป็น 02H หรือ 03H ก็ได้

3. คำสั่งเลือกโหมดการป้อนข้อมูล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	0	1	I/D	S

บิต S เป็นบิตที่ใช้ในการกำหนดลักษณะของการแสดงผล เมื่อมีการป้อนข้อมูล ถ้าหากบิต S เป็น “1” เมื่อเกิดข้อมูลใหม่ในตัวแสดงผล ตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักษรข้อมูลเดิมจะถูกดันไปทางซ้าย แต่ถ้าหากบิตนี้เป็น “0” เมื่อเกิดข้อมูลใหม่ตัวเคอร์เซอร์จะเลื่อนไปทางขวามือ

บิต I/D เป็นบิตที่ใช้ในการกำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว ทำให้แอดเดรส (Address) ของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น “1” แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าเป็น “0” แอดเดรสจะลดลง

ดังนั้น ข้อมูลคำสั่งที่เกิดขึ้นสำหรับคำสั่งนี้ได้แก่ 04H – 07H (4 ข้อมูลคำสั่ง) และที่ใช้บ่อยคือ 06H หมายถึง กำหนดให้เมื่อเกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางขวามือและแอดเดรสของ DDRAM เพิ่มขึ้น

4. คำสั่งควบคุมการแสดงผล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	1	D	C	B

บิต D ใช้ควบคุมการเปิดปิดจอแสดงผล ถ้าบิตนี้เป็น “1” จะเป็นการเปิดจอแสดงผล ถ้าเป็น “0” จะเป็นการปิดจอแสดงผล

บิต C ใช้ควบคุมการแสดงตัวเคอร์เซอร์บนจอแสดงผล ถ้าต้องการให้มีเคอร์เซอร์แสดงผลบนจอแสดงผล ต้องกำหนดให้บิตนี้เป็น “1” ถ้ากำหนดให้เป็น “0” จะเป็นการปิดเคอร์เซอร์ หรือไม่แสดงเคอร์เซอร์

บิต B ใช้ควบคุมการกระพริบของเคอร์เซอร์ถ้าบิตนี้เป็น “1” เคอร์เซอร์จะกระพริบ

ดังนั้นจะมีข้อมูลคำสั่งได้ตั้งแต่ 08H – 0FH (8 รูปแบบคำสั่ง) ที่ใช้บ่อยคือ 0CH เป็นการสั่งให้เปิดจอแสดงผล แต่ไม่แสดงเคอร์เซอร์และ 0FH เป็นการสั่งให้เปิดจอแสดงผล แสดงเคอร์เซอร์กระพริบ

5. คำสั่งควบคุมการเลื่อนเคอร์เซอร์และข้อมูลตัวอักษร

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	1	S/C	R/L	*	*

การควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผลขึ้นอยู่กับกำหนดบิต S/C และ R/L ซึ่งสามารถสรุปได้ดังนี้

S/C	R/L	ลักษณะการเลื่อน	ข้อมูลคำสั่ง
0	0	เลื่อนเคอร์เซอร์ไปทางซ้าย	10H – 13H
0	1	เลื่อนเคอร์เซอร์ไปทางขวา	14H – 17H
1	0	เลื่อนตัวอักษรใหม่ไปทางซ้าย	18H – 1BH
1	1	เลื่อนตัวอักษรใหม่ไปทางขวา	1CH – 1FH

ตารางที่ 2.4 ความสัมพันธ์ของเคอร์เซอร์และตัวอักษร

6. คำสั่งอ่านแฟลก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	1	DL	N	F	*	*

บิต DL ใช้กำหนดจำนวนบิตที่ใช้ติดต่อส่งผ่านข้อมูล ถ้าบิตนี้เป็น “0” จะเป็นการติดต่อแบบ 4 บิต แต่ถ้าเป็น “1” จะเป็นแบบ 8 บิต

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล ถ้าเป็น “0” จะแสดงผล 1 บรรทัด ถ้าเป็น “1” จะแสดงผล 2 บรรทัด ในกรณีที่จอแสดงผลสามารถแสดงได้มากกว่า 2 บรรทัด และต้องการให้แสดงผลมากกว่า 2 บรรทัด ก็กำหนดบิต N นี้ให้เป็น “1”

บิต F ใช้เลือกความละเอียดของตัวอักษรให้การแสดงผล ถ้าบิตนี้เป็น “0” จะเป็นการแสดงผลแบบ 5x7 จุด และถ้าเป็น “1” จะแสดงผลเป็นแบบ 5x10 จุด

ข้อมูลคำสั่งที่ใช้บ่อยคือ 38H เป็นการกำหนดให้โมดูล LCD ทำงานในแบบ 8 บิต แสดงผล 2 บรรทัด และเลือกความละเอียดเป็น 5x7 จุด

จุดที่น่าสังเกตคือ โมดูล LCD แบบ 16 ตัวอักษร 1 บรรทัด แม้จะมีบรรทัดการแสดงผลเพียง 1 บรรทัด แต่จะต้องกำหนด N ให้เป็น “1” เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่องคือ 00H และ 40H

7. คำสั่งเลือกแอดเดรสของ CGRAM

เมื่อต้องการกำหนดแอดเดรสของ CGRAM ต้องกำหนดให้บิต 7 เป็น “0” บิต 6 เป็น “1” ส่วนอีก 6 บิตที่เหลือจะแทนด้วยค่าแอดเดรสของ CGRAM จะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ก่อนที่จะอ่านหรือเขียนข้อมูลให้ CGRAM โดยแอดเดรสของ CGRAM อยู่ระหว่าง 00H – 3FH

8. คำสั่งเลือกแอดเดรสของ DDRAM

ใช้ในการเลือกแอดเดรสของ DDRAM ก่อนที่จะทำการอ่านหรือเขียนข้อมูล โดยบิต 7 ต้องเป็น “1” และข้อมูลอีก 7 บิตที่เหลือจะเป็นค่าแอดเดรสของ DDRAM ซึ่งแอดเดรสของ DDRAM จะอยู่ระหว่าง 8CH – 0FFH ทั้งนี้จำนวนแอดเดรสวิ่งขึ้นกับการกำหนดสถานะที่บิต N ด้วย หากบิต N เป็น “0” แอดเดรส

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F

รูปที่ 2.19 แอดเดรส DDRAM ของ LCD Module รุ่น DMC164

ของ DDRAM จะอยู่ระหว่าง 80H – 0CH และถ้าบิต N เป็น “1” แอดเดรสของ DDRAM จะมี 2 ช่วงคือ 8CH – 87H และ 0C0H – 0C7H

9. คำสั่งอ่านแฟล็ก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
BF	A	A	A	A	A	A	A
แอดเดรสไบต์สูง				แอดเดรสไบต์ต่ำ			

เป็นคำสั่งที่ใช้อ่านแฟล็ก BUSY (BF) โดยแฟล็กนี้จะเป็นตัวบอกสถานะของตัวควบคุม LCD ว่าพร้อมจะรับข้อมูลหรือไม่ ถ้าหากบิต BF เป็น “0” แสดงว่าตัวควบคุม LCD พร้อมรับข้อมูลหรือคำสั่ง แต่ถ้าเป็น “1” แสดงว่าขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการทำงานภายในหรือกำลังประมวลผลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง

เมื่อต้องการอ่านแฟล็กต้องกำหนดให้ขา $R/\overline{W}$ เป็น “1” ด้วย แต่สัญญาณที่ RS ยังต้องเป็น “0” อยู่ เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง

นอกจากนี้ ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วย โดยบิต 0 – บิต 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

2.9.4 การเขียนคำสั่งและข้อมูลให้แก่โมดูล LCD

ในการเขียนข้อมูลเพื่อควบคุมให้โมดูล LCD แสดงผลตามที่ผู้ใช้งานต้องการ ต้องส่งคำสั่ง (Instruction) แล้วกำหนดโหมดการทำงานให้แก่โมดูล LCD ก่อน จากนั้นจึงค่อยส่งข้อมูล (Data) ที่ต้องการแสดงผล เนื่องจากบัสข้อมูลของโมดูล LCD มี 8 เส้นคือ D0 – D7 และใช้เป็นทางผ่านของทั้งค่า

ตั้งและข้อมูล ดังนั้นในการส่งคำสั่งและข้อมูลจึงต้องอาศัยการกำหนดสัญญาณลอจิกที่ขา RS ถ้าหากที่ขา RS ได้ลอจิก “0” หมายความว่า ข้อมูลที่ป้อนให้แก่โมดูล LCD ขณะนั้นเป็นคำสั่ง ในทางตรงกันข้าม หากขา RS ได้รับลอจิก “1” ข้อมูลที่ป้อนให้ขณะนั้นเป็นข้อมูลที่ใช้ในการแสดงผล

เมื่อต้องการเขียนหรืออ่านข้อมูลใน CGRAM และ DDRAM เริ่มต้นต้องกำหนดแอดเดรสที่ต้องการอ่านหรือเขียนก่อน โดยใช้คำสั่งหรือแอดเดรส จากนั้นกำหนดให้ขา RS เป็น “1” เพื่อแจ้งให้ตัวควบคุมภายในโมดูล LCD ทราบว่าข้อมูลที่ปรากฏต่อไปนี้เป็นข้อมูลปกติไม่ใช่คำสั่ง

ในกรณีที่ต้องการอ่านข้อมูลต้องกำหนดให้ขา $R/\overline{W}$ เป็น “1” ข้อมูลขนาด 8 บิต (หรือ 4 บิต) ก็จะปรากฏบนบัสข้อมูล (Data Bus) โดยข้อมูลที่อ่านออกมาได้จะเป็นข้อมูลจากแอดเดรสของ CGRAM หรือ DDRAM ตามที่ต้องการ

ในกรณีที่ต้องการเขียนข้อมูล เมื่อกำหนดแอดเดรสและป้อนลอจิก “1” ให้ขา RS แล้ว แล้วต้องกำหนดให้ขา $R/\overline{W}$ เป็น “0” ข้อมูลที่อยู่บนบัสข้อมูลจะถูกเขียนลงในรีจิสเตอร์ DR จากนั้นจึงถ่ายทอดลงใน DDRAM ต่อไป

2.9.5 จังหวะการทำงานของโมดูล LCD

ในการติดต่อกับโมดูล LCD จะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูล เนื่องจากต้องรอให้คอนโทรลเลอร์ภายใน LCD แปลความหมายของรหัสคำสั่งและทำงานตามคำสั่งให้เรียบร้อยก่อน จากนั้นจึงจะรับข้อมูลหรือดำเนินการต่อไป

ดังนั้น ในการใช้งานโมดูล LCD ผู้เขียนโปรแกรมต้องมีโปรแกรมเพื่อหน่วงเวลารอให้โมดูล LCD พร้อมทำงานด้วย โดยเมื่อเริ่มจ่ายไฟให้แก่โมดูล LCD ต้องรอประมาณ 10 มิลลิวินาที เพื่อให้โมดูล LCD ทำการเตรียมความพร้อมหรืออินิเชียล (Initial) หลังจากนั้นก็จะกำหนดลอจิกให้แก่ขา RS ของโมดูล LCD แล้วต้องหน่วงเวลาอีกประมาณ 2 มิลลิวินาที เพื่อให้คอนโทรลเลอร์ใน LCD โมดูลแปลความหมายของลอจิกที่ขา RS ว่า ข้อมูลต่อไปที่จะได้รับนั้นเป็นรหัสคำสั่งหรือเป็นข้อมูลที่ต้องการแสดงผล จากนั้นจะเป็นการส่งข้อมูลมารอที่บัสข้อมูล D0 – D7 (กรณีทำงานในโหมด 8 บิต) ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ไปที่ขา E เพื่ออีนาเบิล (Enable) โมดูล LCD ให้รับข้อมูลจากบัสข้อมูลเข้าไป โดยพัลส์ที่ป้อนเข้าที่ขา E ของ โมดูล LCD ต้องเป็นพัลส์ขอบขาขึ้น จากนั้นทำการหน่วงเวลา 2 มิลลิวินาที

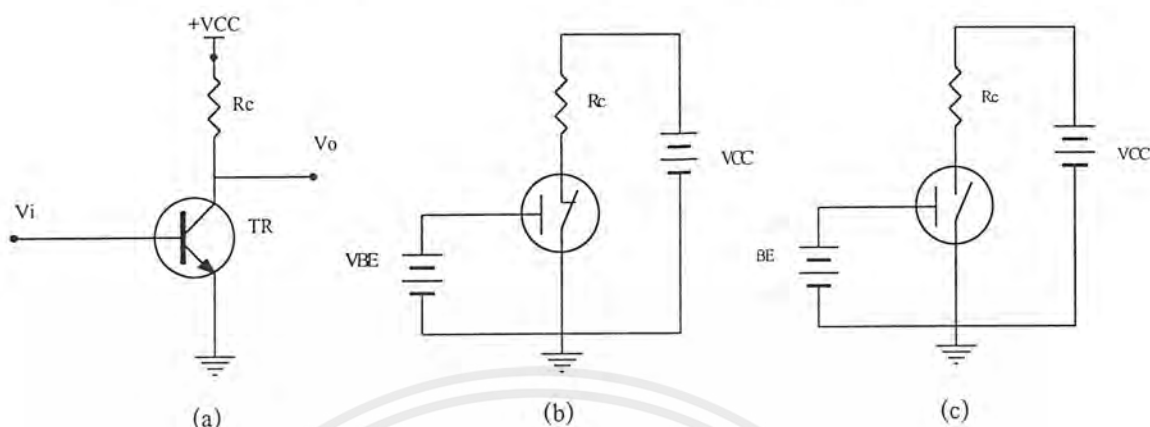
ทั้งหมดที่กล่าวมาคือขั้นตอนและจังหวะในการทำงาน 1 รอบของโมดูล LCD จะเห็นได้ว่ามีโปรแกรมย่อยที่สำคัญอยู่ 3 โปรแกรมย่อยคือ โปรแกรมอินิเชียล LCD, โปรแกรมหน่วงเวลา และโปรแกรมย่อยการส่งพัลส์เพื่ออีนาเบิลโมดูล LCD

2.10 สวิตช์ทรานซิสเตอร์ (Switch Transistor)

เป็นการนำทรานซิสเตอร์มาทำหน้าที่เป็นสวิตช์ ปิด-เปิด วงจรอิเล็กทรอนิกส์จากคุณสมบัติของทรานซิสเตอร์ เมื่อทรานซิสเตอร์ทำงานอยู่ในสภาวะ “OFF” จะถูกแทนด้วยสวิตช์ “เปิด” (Open) และเมื่อทรานซิสเตอร์ทำงานอยู่ในสภาวะ “ON” จะถูกแทนด้วยสวิตช์ “ปิด” (Close)

จากรูปที่ 2.20 (a) เป็นวงจรสวิตช์ทรานซิสเตอร์ในอุดมคติกล่าวคือ เมื่อทรานซิสเตอร์อยู่ในสภาวะ “ON” จะมีแรงดันตกคร่อมที่ขาคอลเลกเตอร์กับขาอีมีตเตอร์เท่ากับ 0 โวลต์ และเมื่อทรานซิสเตอร์

อยู่ในสถานะ “OFF” จะไม่มีกระแสไหล นั่นคือ $I_C = 0$ โวลต์ สำหรับขั้วสวิตช์ของทรานซิสเตอร์ คือ ขาคอลเลกเตอร์กับขาอีมิเตอร์ซึ่งการปิด-เปิดของสวิตช์ถูกควบคุมโดยแรงดันทางด้าอินพุต เมื่อแรงดัน



รูปที่ 2.20 การต่อวงจรสวิตช์ทรานซิสเตอร์และวงจรสมมูล

อินพุตมีค่า $+V$ มากจนทำให้ขาเบสและขาอีมิเตอร์ของทรานซิสเตอร์ได้รับไบอัสตรง (Forward Bias) ทรานซิสเตอร์จะทำงานในสถานะอิ่มตัวทรานซิสเตอร์จะถูกแทนด้วยสวิตช์ “ON” โดยมีแรงดันตกคร่อมระหว่างขาคอลเลกเตอร์และขาอีมิเตอร์ $V_{CE} = 0V$ นั่นเอง นั่นคือแรงดันเอาต์พุต $V_o = 0$ ดังรูปที่ 2.20 (b) ทำให้มีกระแส I_C ไหลจากขั้ว $+V_{CC}$ ผ่าน R_C เกิดแรงดันตกคร่อมเท่ากับ

$$\begin{aligned} V_{RC} + V_o - V_{CC} &= 0 \\ V_{RC} &= V_{CC} - V_o \\ V_{RC} &= V_{CC} \end{aligned} \quad \text{เมื่อ } V_o = 0$$

และถ้าแรงดันอินพุตมีค่า $-V$ จะทำให้ขาเบสกับขาอีมิเตอร์ของทรานซิสเตอร์ได้รับไบอัสกลับ (Reverse Bias) ทรานซิสเตอร์จึงทำงานในสถานะคัทออฟ สามารถแทนทรานซิสเตอร์ด้วยสวิตช์ “OFF” ทำให้ไม่มีกระแสไหลผ่านทรานซิสเตอร์ ($I_C = 0$) ดังรูปที่ 2.20 (c) แรงดันตกคร่อมระหว่างขาคอลเลกเตอร์กับขาอีมิเตอร์มีค่าเท่ากับ

$$\begin{aligned} V_o - V_{CC} + V_{RC} &= 0 \\ V_o &= V_{CC} - V_{RC} \\ V_o &= V_{CC} \end{aligned} \quad \text{เมื่อ } I_C = 0$$

ขณะที่ขาเบสและขาอีมิเตอร์ของทรานซิสเตอร์ได้รับไบอัสตรงจะทำให้กระแส I_B ไหลกระแส ส่วนนี้จะไปควบคุมให้กระแส I_C ไหลด้วยโดยขนาดของกระแส I_C จะมีค่าเท่ากับผลคูณของกระแส I_B กับ อัตราการขยายกระแสตรงซึ่งเขียนสมการได้ คือ

$$I_C = h_{FE} \cdot I_B$$

เมื่อ h_{FE} คือ อัตราขยายกระแสตรงของทรานซิสเตอร์

เมื่อทรานซิสเตอร์ทำงานเป็นสวิตช์จะเกิดการสูญเสียกำลังในรูปของพลังงานความร้อน ซึ่งมีค่ากำลังงานส่วนนี้หาได้จากสูตร

$$P_D = I_C \cdot V_{CE}$$

เมื่อ P_D คือ กำลังที่สูญเสียในรูปของความร้อน (วัตต์)



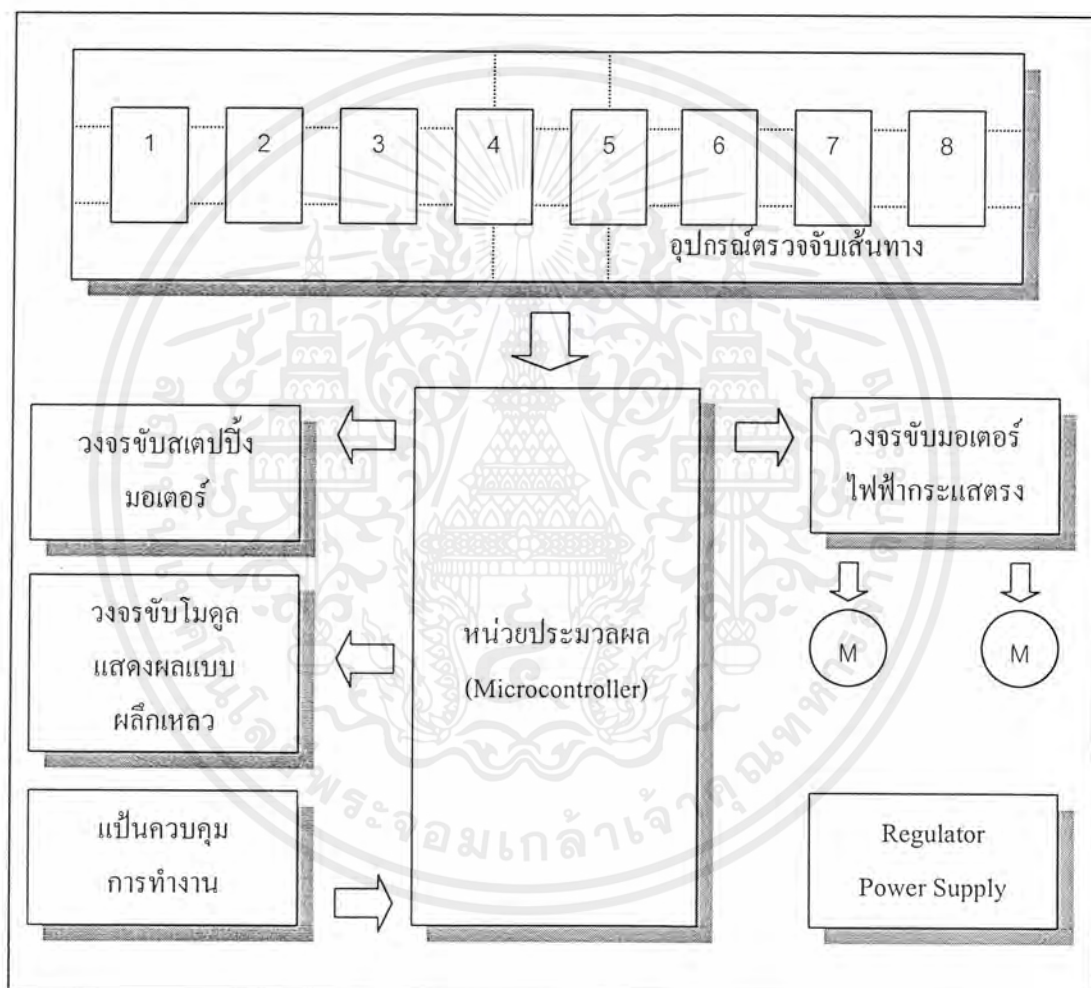
บทที่ 3

การออกแบบสร้าง Hardware และ Software

เนื่องจากในโครงการนี้มีทั้งส่วนที่เป็นโครงสร้าง (Hardware) และส่วนที่เป็นโปรแกรม (Software) ซึ่ง สามารถจำแนกได้ดังนี้

3.1 โครงสร้างหุ่นยนต์

ทำการออกแบบตัวโครงสร้างของหุ่นยนต์ 4 ล้อ โดยจะใช้สองล้อหลังในการขับเคลื่อน และจะใช้สองล้อหน้าในการควบคุมทิศทางการเคลื่อนที่ ดังนั้นการออกแบบจะอาศัยภาพโดยรวมตามส่วนประกอบ ที่มีดังรูปที่ 3.1



รูปที่ 3.1 ภาพโดยรวมของโครงสร้างหุ่นยนต์

ตัวหุ่นยนต์ ประกอบไปด้วยส่วนขับเคลื่อนที่เป็นมอเตอร์ไฟฟ้ากระแสตรงพร้อมเฟืองทด ส่วนของมอเตอร์ไฟฟ้ากระแสตรงที่ใช้ในการ Feed เอกสาร ส่วนของสเตปปี้งมอเตอร์ ที่ใช้ในการควบคุมทิศทางการเคลื่อนที่ ส่วนของอุปกรณ์ตรวจจับเส้นทางจำนวน 8 ชุด ส่วนของจอแสดงผลแบบผลึกเหลว (LCD Module) ที่ใช้ในการแสดงผลการทำงาน และแป้นควบคุมการทำงาน ที่ใช้ในการสั่งงานของหุ่นยนต์ในโครงการนี้

3.2 ส่วนของการประมวลผล (Microcontroller)

การออกแบบจะใช้ไมโครคอนโทรลเลอร์ (Microcontroller) ตระกูล MCS-51 เบอร์ 8031 เนื่องจากเป็นไมโครคอนโทรลเลอร์ที่มีโครงสร้างและคำสั่งที่ง่ายต่อการทำความเข้าใจ โดยทำการต่อเข้ากับ 8255 จำนวน 3 ตัว เพื่อใช้ในการขยายพอร์ต

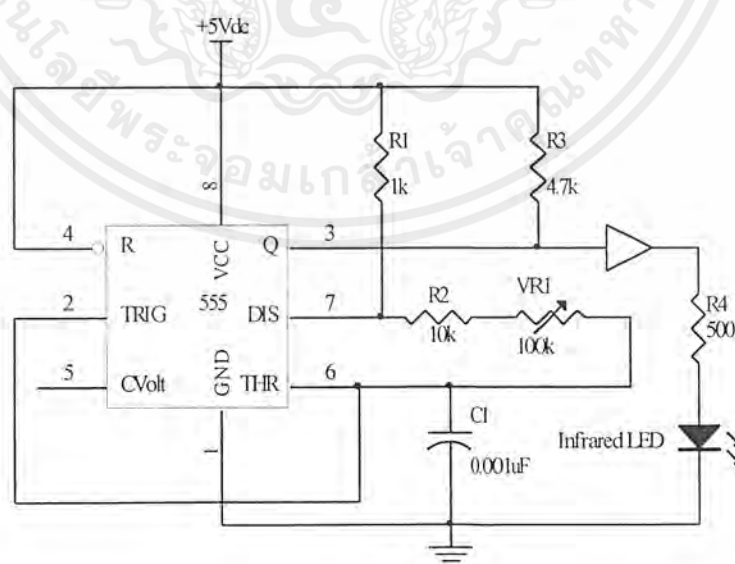
ของไมโครคอนโทรลเลอร์ ให้เพียงพอต่อการใช้งาน ซึ่งมีลักษณะการต่อตัวไมโครคอนโทรลเลอร์เข้ากับ 8255 ทั้ง 3 ตัว โดยที่

- 8255 # 1 POTRA ทำการต่อเข้ากับ D0-D7 ของ LCD Module
 PORTB ทำการต่อเข้ากับขา Enable และ RS ของ LCD Module
 PORTC ทำการต่อเข้ากับวงจรขับสเตปป์มอเตอร์
- 8255 # 2 POTRA ทำการต่อเข้ากับอุปกรณ์ตรวจจับเส้นทาง
 PORTB ทำการต่อเข้ากับตัวแสดงผล สถานะของอุปกรณ์ตรวจจับเส้นทาง
 PORTC ทำการต่อเข้ากับวงจรมอเตอร์ไฟฟ้ากระแสตรงที่ใช้ในการขับเคลื่อน
- 8255#3 PORTA ทำการต่อเข้ากับวงจรมอเตอร์ไฟฟ้ากระแสตรงที่ใช้ในการ Feed เอกสาร
 PORTB ทำการต่อเข้ากับตัวแสดงผล นับแยกของเส้นทางที่หุ่นยนต์เคลื่อนที่ผ่าน
 PORTC ทำการต่อเข้ากับตัวแสดงผล สถานะของอินเตอร์รัพต์ - ไทมเมอร์

ส่วนของเป็นควบคุมการทำงาน จะทำการต่อเข้ากับพอร์ต P1.0 – P1.5 ของตัวไมโครคอนโทรลเลอร์

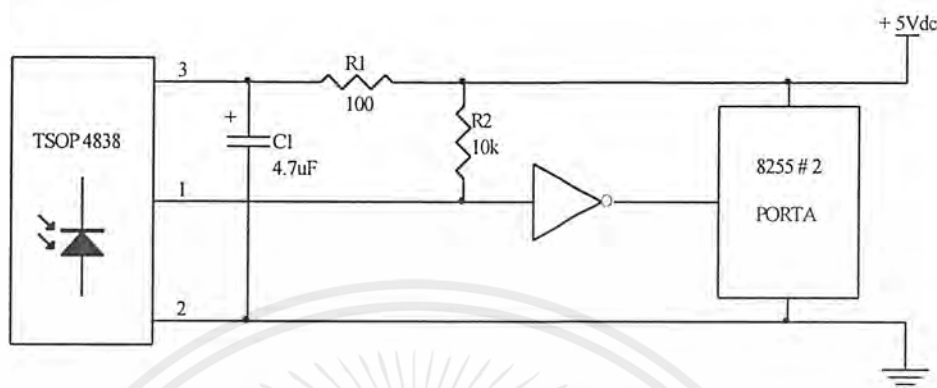
3.3 ส่วนของการตรวจจับเส้นทางการเดินทาง

การออกแบบวงจรให้สามารถตรวจสอบความเข้มของสี และแสดงเอาต์พุตออกมาเป็นลักษณะลอจิกได้ โดยไม่ถูกรบกวนจากสัญญาณรบกวนภายนอก ซึ่งในโครงงานนี้จะใช้ การส่งสัญญาณ โทนเบรสต์ความถี่ 38 KHz ทางด้านส่ง และใช้โฟโตโมดูลในการรับและแสดงผลออกมาเป็นลอจิกทางด้านรับ



รูปที่ 3.2 วงจรภาคส่งสัญญาณโทนเบรสต์

จากรูปที่ 3.2 เป็นวงจรภาคส่งสัญญาณโทนเบรสต์ที่ใช้งานจริงในโครงการ ซึ่งมีการต่อไอซี 555 ไทมเมอร์ เป็นวงจรอะอสเตเบิลล์มัลติไวเบเรเตอร์ กำหนดความถี่ 38 KHz ออกมาเป็นเอาต์พุตที่ขา 3 ของตัวไอซี ป้อนเข้าแอลอีดีอินฟราเรด เพื่อส่งสัญญาณต่อไป ซึ่งสามารถปรับความถี่ใช้งานของสัญญาณได้โดยปรับที่ VR3



รูปที่ 3.3 วงจรภาครับสัญญาณโทนเบรสต์

จากรูปที่ 3.3 เป็นวงจรภาคส่งสัญญาณโทนเบรสต์ที่ใช้งานจริงในโครงการ โดย R_1 มีหน้าที่ในการป้องกันสัญญาณรบกวนจากแหล่งจ่ายไฟ

ในขณะที่ทำการตรวจสอบเส้นทางแล้วไม่พบเส้นทาง (ไม่พบเส้นสีดำ) แสงจากแอลอีดีอินฟราเรดจะกระทบพื้นเกิดการสะท้อนกลับเข้ามาที่โฟโต้โมดูล ทำให้เอาต์พุตลอจิกที่ 8255 มีค่าเป็น “1” นั่นคือ ตรวจสอบแล้วพบเส้นทาง ในทางตรงกันข้ามหากทำการตรวจสอบแล้วพบเส้นทาง (พบเส้นสีดำ) แสงจากแอลอีดีอินฟราเรด ที่ปล่อยออกมาจะถูกดูดซับด้วยเส้นสีดำ ทำให้ไม่มีแสงสะท้อนกลับมายังโฟโต้โมดูล ส่งผลให้ เอาต์พุตลอจิกที่ 8255 มีค่าเป็น “0”

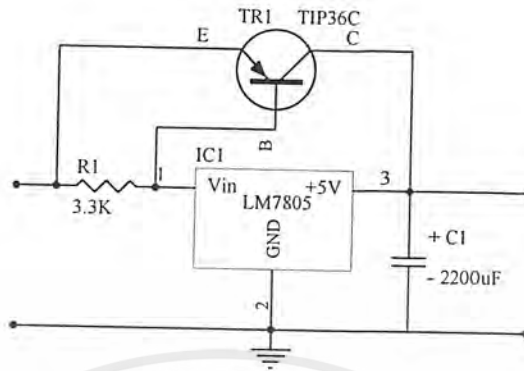
การออกแบบใช้งาน จะต่อไมโครคอนโทรลเลอร์เข้ากับ 8255 # 2 PORTA ในการตรวจสอบระดับเอาต์พุตลอจิกที่ได้จากอุปกรณ์ตรวจจับเส้นทาง ทั้ง 8 ตัว เพื่อนำมาประมวลผลต่อไป

3.4 ส่วนของเรกูเลเตอร์ (Regulator Power-supply)

วงจรเรกูเลเตอร์ที่ใช้ในโครงการจะแบ่งออกเป็น 2 ส่วนคือ ส่วนที่เป็นวงจรเรกูเลเตอร์แบบคงที่ ซึ่งใช้เป็นแหล่งจ่ายไฟ + 5 โวลต์ ให้กับ ตัวไมโครคอนโทรลเลอร์ อุปกรณ์ตรวจจับเส้นทาง เป็นควบคุมการทำงาน วงจรขับเคลื่อนมอเตอร์ และวงจรขับโมดูลแสดงผลแบบผลึกเหลวซึ่งแสดงในรูปที่ 3.3 ส่วนที่สองจะเป็นวงจร เรกูเลเตอร์แบบปรับค่าได้ 2 ระดับ ซึ่งใช้เป็นแหล่งจ่ายไฟให้กับวงจรขับเคลื่อนมอเตอร์ไฟฟ้า กระแสตรง ทำให้สามารถปรับระดับความเร็วการหมุนของมอเตอร์ได้ 2 ระดับซึ่งแสดงในรูปที่ 3.4

จากรูปที่ 3.4 เป็นวงจรเรกูเลเตอร์แบบปรับค่าได้ 2 ระดับ ได้ทำการออกแบบวงจรให้มีความยืดหยุ่นต่อการปรับค่าระดับแรงดันที่ป้อนให้กับมอเตอร์ไฟฟ้ากระแสตรง โดยใช้ตัวต้านทานแบบปรับค่าได้ VR1 มาต่อเข้ากับขาแอดจัสต์ (Adjust) ของไอซี LM317T แทนที่จะทำการต่อดั้ว ตัวต้านทานแบบคงที่ ทรานซิสเตอร์ TR1 และตัวต้านทานแบบปรับค่าได้ VR2 จะใช้ในการเปลี่ยนระดับแรงดัน โดยกรณีที่ TR1 OFF จะส่งผลให้แรงดันเอาต์พุตมีค่าแปรผันตามค่าความต้านทานของ VR1 คือเมื่อค่าความต้านทาน

เพิ่มขึ้นจะส่งผลให้แรงดันเอาต์พุตมีค่าเพิ่มขึ้น และถ้า VR1 มีค่าความต้านทานลดลงแรงดัน เอาต์พุตก็ลดลงเช่นเดียวกันในกรณีที่ TR1 ON เป็นผลให้ VR1 กับ VR2 นั้นต่อขนานกันทำให้ค่าความต้านทานที่



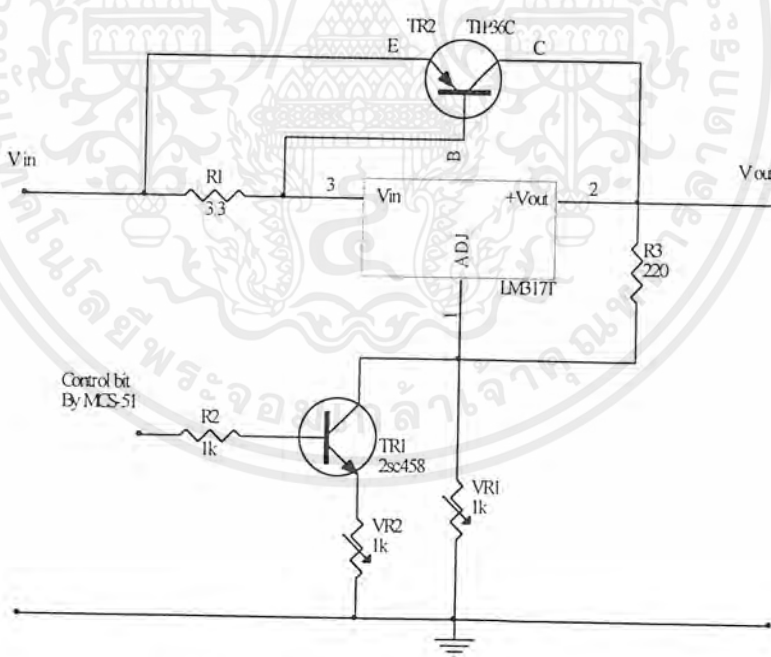
รูปที่ 3.4 วงจรเรกูเลเตอร์แบบคงที่

ต่อกับขาแอดจัสต์ มีค่าลดลงตามไปด้วย ทำให้แรงดันเอาต์พุตมีค่าลดลงด้วย การควบคุมให้ TR1 ON/OFF นั้นสามารถทำได้โดยการส่งลอจิกไปควบคุมที่ขาเบสของ TR1 การออกแบบใช้งานจะทำการต่อตัวไมโครคอนโทรลเลอร์เข้ากับ 8255 #2 ตามพอร์ทใช้งานดังนี้

8255 # 2

PORTC บิตที่ 2 ควบคุมระดับแรงดันไฟให้มอเตอร์ทางด้านขวา

PORTC บิตที่ 3 ควบคุมระดับแรงดันไฟให้มอเตอร์ทางด้านซ้าย

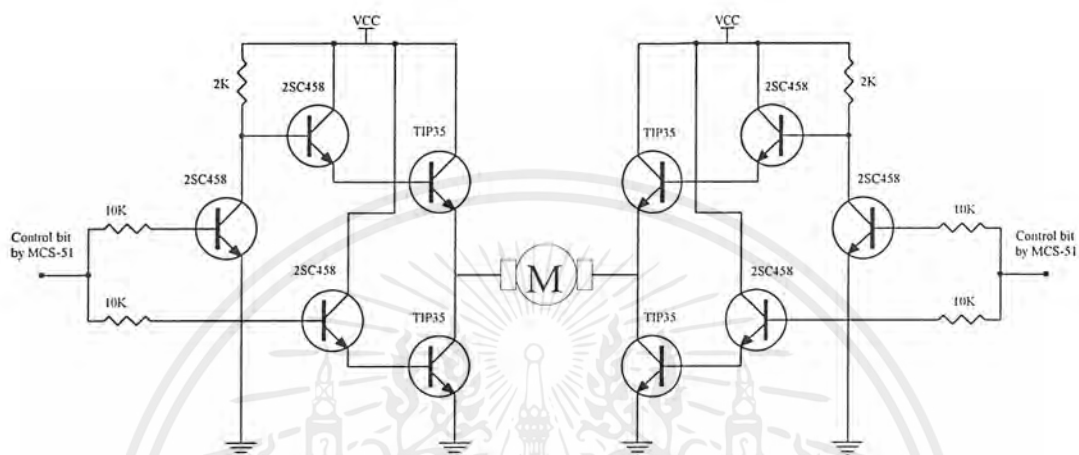


รูปที่ 3.5 วงจรเรกูเลเตอร์แบบปรับค่าได้ 2 ระดับ

3.5 ส่วนของวงจรขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรง

ทำการออกแบบโดยใช้ทรานซิสเตอร์ควบคุมโดยมีลักษณะการต่อวงจรตามรูปที่ 3.6 โดยจะทำการต่อวงจรในลักษณะนี้ เพื่อให้ควบคุมมอเตอร์ขับเคลื่อนล้อหลัง และมอเตอร์ที่ใช้ในการ Feed เอกสาร โดยที่

- 8255 # 2 PORTC บิตที่ 0 และ บิตที่ 1 ใช้ควบคุมทิศทางการหมุนของมอเตอร์ด้านขวา
 PORTC บิตที่ 4 และ บิตที่ 5 ใช้ควบคุมทิศทางการหมุนของมอเตอร์ด้านซ้าย
- 8255 # 3 PORTA บิตที่ 0 และ บิตที่ 1 ใช้ควบคุมมอเตอร์ Feed เอกสาร ชั้นที่ 1
 PORTA บิตที่ 2 และ บิตที่ 3 ใช้ควบคุมมอเตอร์ Feed เอกสาร ชั้นที่ 2
 PORTA บิตที่ 4 และ บิตที่ 5 ใช้ควบคุมมอเตอร์ Feed เอกสาร ชั้นที่ 3



รูปที่ 3.6 วงจรขับมอเตอร์ไฟฟ้ากระแสตรง

จากรูปที่ 3.6 วงจรจะทำงานตามคำสั่งของโปรแกรมที่เขียนขึ้น ถ้ามีระดับลอจิก “0” หรือ “1” ป้อนให้กับวงจร มีค่าเหมือนกันทั้งคู่จะทำให้ระดับแรงดันที่ขั้วต่อของมอเตอร์มีระดับแรงดันที่ไม่แตกต่างกันส่งผลให้มอเตอร์ไม่หมุน ในทางกลับกันถ้ามีการป้อนระดับลอจิกทั้งสองข้างมีค่าแตกต่างกันจะทำให้ระดับแรงดันที่ขั้วต่อของมอเตอร์มีระดับแรงดันที่แตกต่างกัน ส่งผลให้มอเตอร์หมุนไปในทิศทางใดทิศทางหนึ่ง

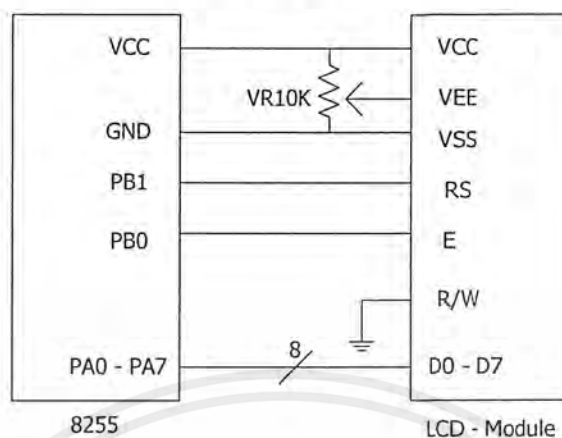
บิตควบคุม		ทิศทางการหมุนของมอเตอร์
0	0	ไม่หมุน
0	1	หมุน
1	0	หมุนตรงกันข้าม
1	1	ไม่หมุน

ตารางที่ 3.1 ความสัมพันธ์ของบิตควบคุมกับทิศทางการหมุนของมอเตอร์

3.6 ส่วนของวงจรขับโมดูลแสดงผลแบบผลิกเหลว

ทำการออกแบบส่วนแสดงผล โดยใช้โมดูลแสดงผลแบบผลิกเหลว ต่อเข้ากับ 8255 ใช้บัสข้อมูล (DATA BUS) ในการเชื่อมต่อ 8 เส้น ซึ่งการต่อลักษณะนี้จะส่งผลให้สามารถเขียนข้อมูลให้กับตัวโมดูลแสดงผลแบบผลิกเหลวได้เพียงอย่างเดียวไม่สามารถอ่านข้อมูลกลับได้ การต่อใช้งานจะต่อโมดูลแสดงผลแบบผลิกเหลวเข้ากับ 8255 # 1 โดยที่

POTRA บิตที่ 0-7 ต่อเข้ากับขาข้อมูล (D0-D7)
 PORTB บิตที่ 0-1 ต่อเข้ากับขา Enable และ RS



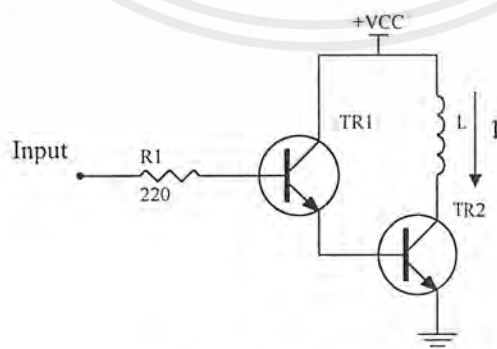
รูปที่ 3.7 การต่อโมดูลแสดงผลลิกเหลวเข้ากับ 8255

3.7 ส่วนของวงจรขับสเตปป์มอเตอร์

ทำการออกแบบโดยนำสเตปป์มอเตอร์แบบยูนิโพลาร์มาใช้ในการควบคุมทิศทางการเคลื่อนที่ของหุ่นยนต์ ในส่วนของวงจรขับสเตปป์มอเตอร์จะใช้วงจรทรานซิสเตอร์สวิตช์ ในการควบคุมการไหลของกระแสที่ผ่านขดลวดแต่ละขดของสเตปป์มอเตอร์ ตามวงจรดังรูปที่ 3.8

ในวงจรจะประกอบด้วยตัวต้านทาน R1 ทรานซิสเตอร์ TR1 และทรานซิสเตอร์ TR2 โดยที่ R1 ทำหน้าที่จำกัดกระแสไบอัส(Bias)ที่เข้ามาทางขาเบสของ TR1 ซึ่งต่อเพื่อทำการเพิ่มกระแสไบอัสให้กับ TR2 โดย TR2 จะทำหน้าที่เป็นตัวควบคุมการไหลของกระแสที่ผ่านขดลวดของสเตปป์มอเตอร์ การทำงานเมื่อมีป้อนลอจิก “1” เข้าทางอินพุตของวงจรจะมีกระแสไหลผ่าน R1 ไปกระตุ้นให้ TR1 ON ทำให้มีกระแส

ไหลจากขาคอลเลกเตอร์มายังขาอีมิเตอร์ของ TR1 ไปกระตุ้นขาเบสของ TR2 ทำให้ TR2 ON และทำให้สามารถนำกระแสไหลผ่านขดลวดขดลวดของสเตปป์มอเตอร์ได้ ในกรณีที่ป้อนลอจิก “0” เข้าที่อินพุตของวงจรจะทำให้ TR1 และ TR2 OFF เป็นผลให้ไม่มีกระแสไหลผ่านขดลวดของสเตปป์มอเตอร์

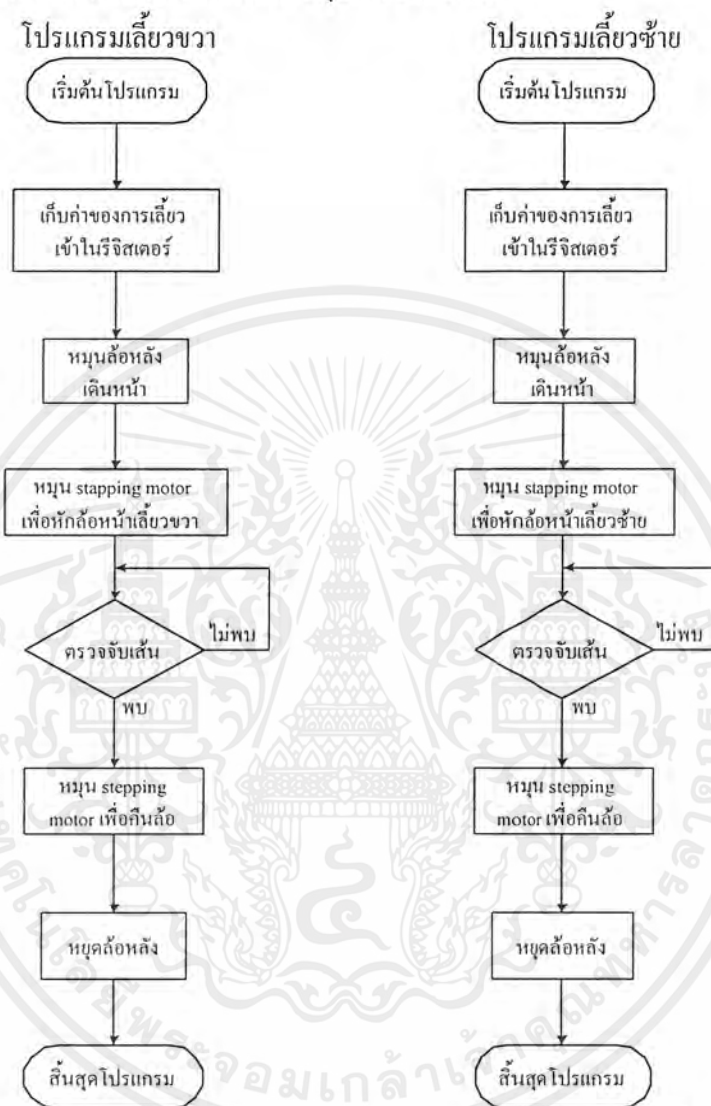


รูปที่ 3.8 วงจรขับสเตปป์มอเตอร์

3.8 โปรแกรมควบคุมการตั้งงานหุ่นยนต์

ทำการเขียนโปรแกรมเพื่อควบคุมการทำงานของตัวหุ่นยนต์ ให้สามารถเคลื่อนที่ไปยังจุดหมายปลายทางได้ตามที่ผู้ใช้ต้องการ

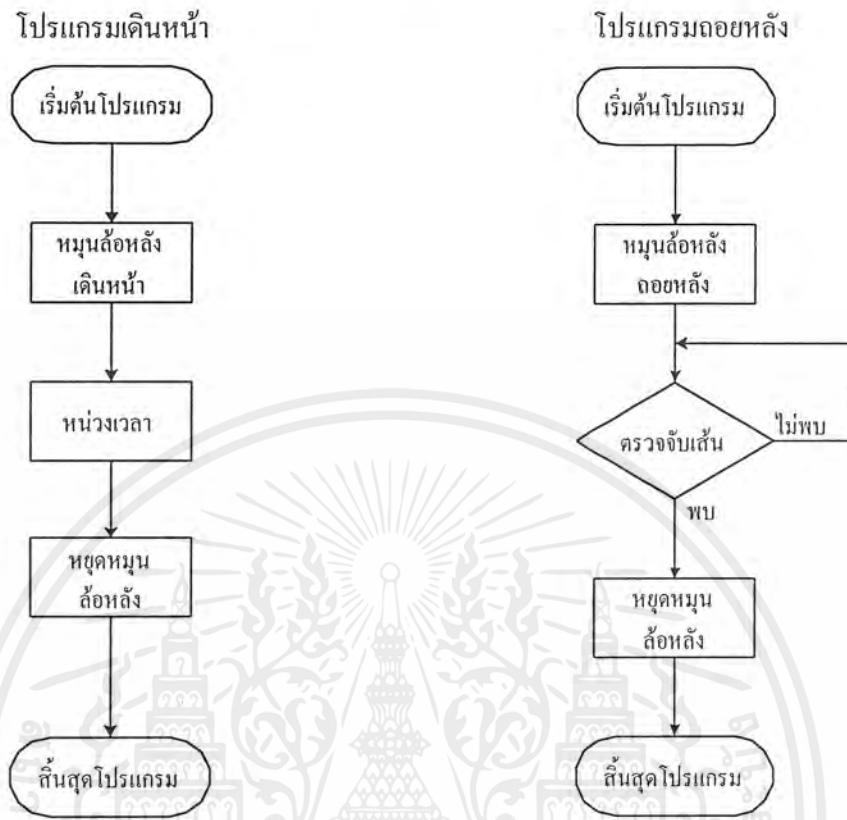
3.8.1 ไฟล์ชาร์ทแสดงโปรแกรมควบคุมการบังคับเลี้ยว



รูปที่ 3.9 ไฟล์ชาร์ทในการเขียนโปรแกรมควบคุมการบังคับเลี้ยว

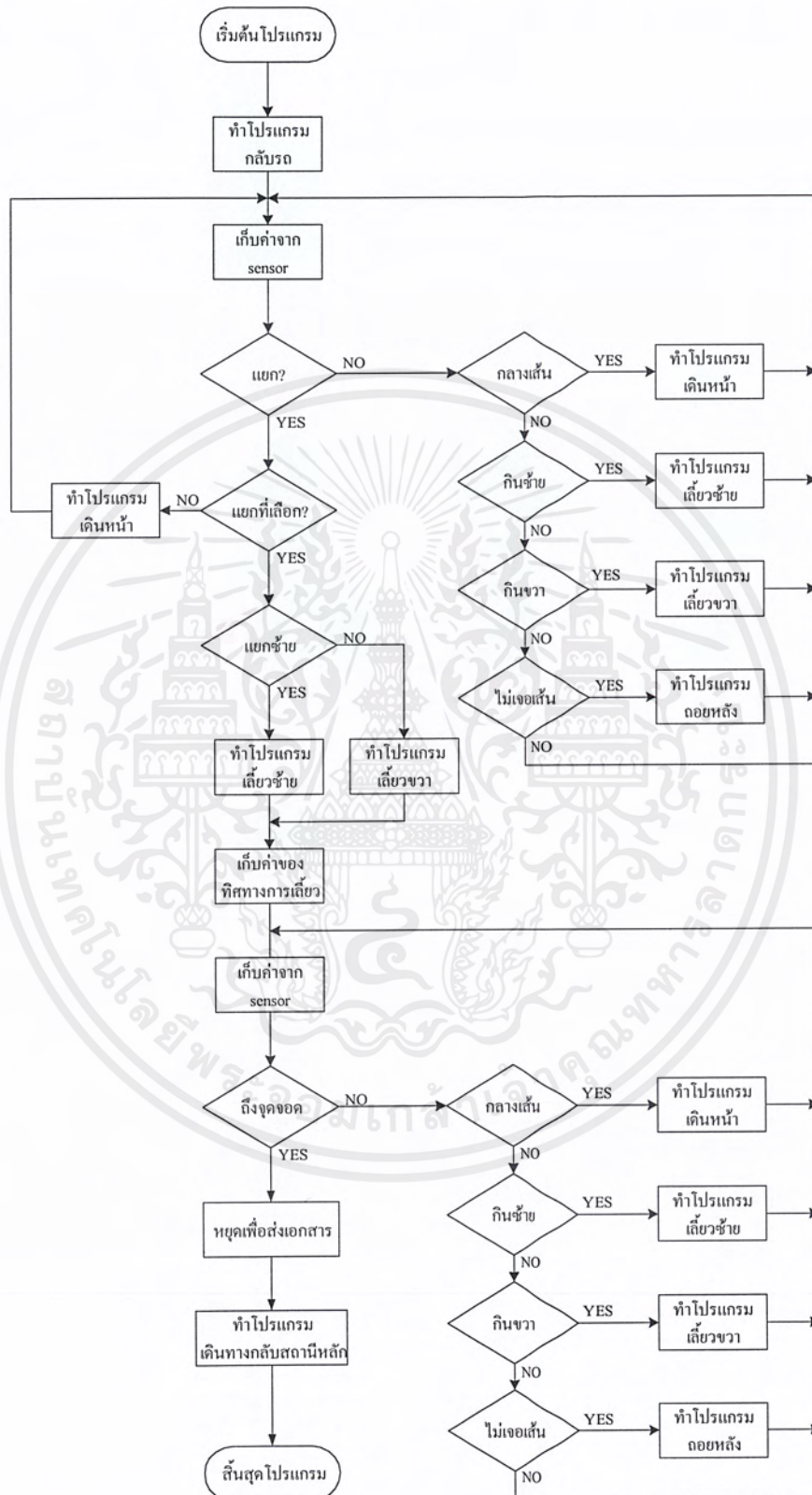
ในทางปฏิบัติ ตัวหุ่นยนต์ไม่ได้มีการเคลื่อนที่เป็นเส้นตรงเพียงอย่างเดียว ในการเดินทางจะต้องเจอแยก หรือโค้งต่าง ๆ ตามที่ผู้ใช้ได้มีการวางเส้นทางไว้ จึงต้องมีการเขียนโปรแกรมควบคุมการบังคับเลี้ยวทั้งซ้าย และ ขวาเพื่อให้ตัวหุ่นยนต์สามารถเคลื่อนที่ได้ตรงตามเส้นทางที่ผู้ใช้ต้องการซึ่งมีไฟล์ชาร์ทดังรูปที่ 3.9

3.8.2 ไฟล์ชาร์ตแสดงโปรแกรมควบคุมการเดินหน้า ถอยหลัง



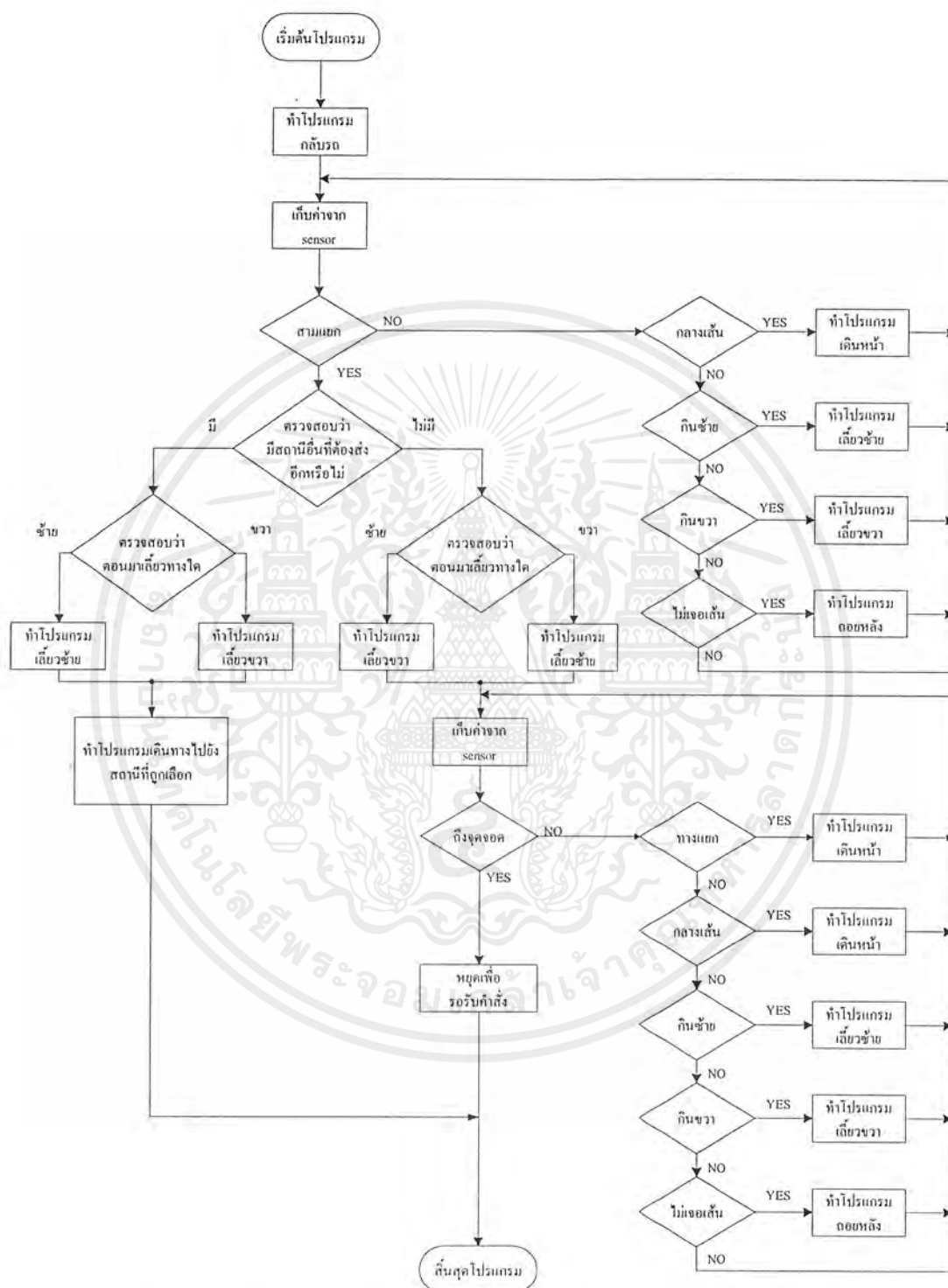
รูปที่ 3.10 ไฟล์ชาร์ตในการเขียนโปรแกรมควบคุมการบังคับเลี้ยว

3.8.3 ไฟล์ชาร์ทแสดงโปรแกรมควบคุมการเดินทางจากสถานีต้นทางไปยังสถานีปลายทาง



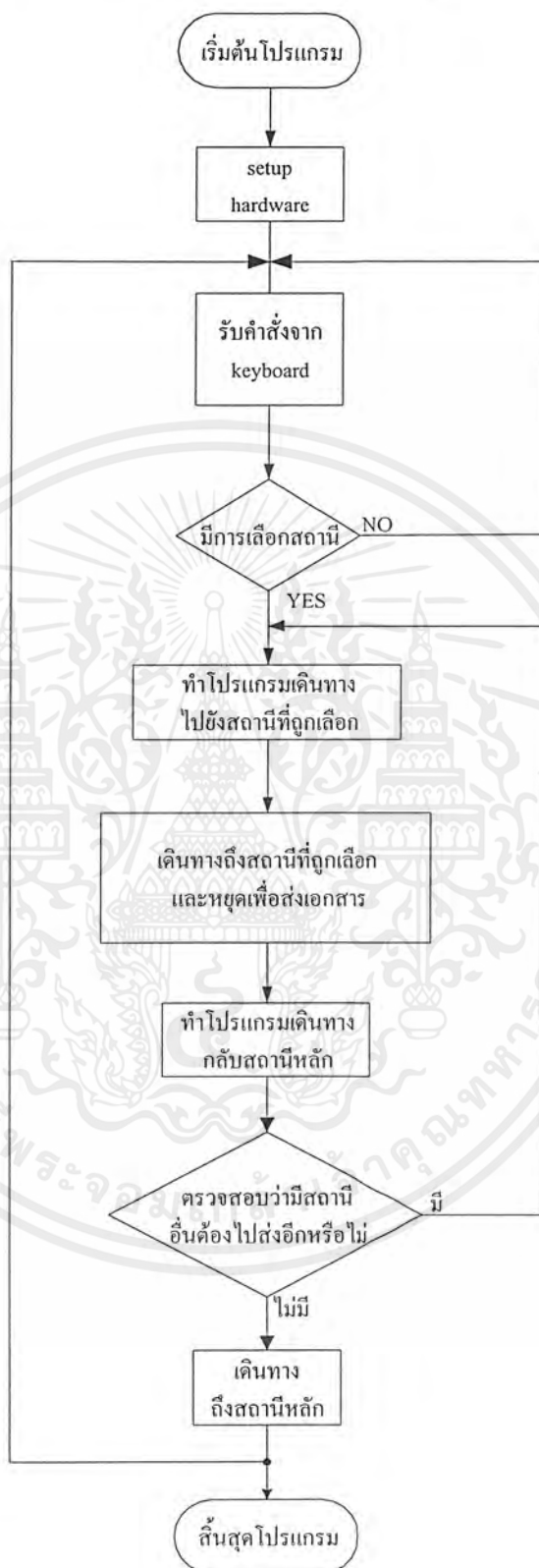
รูปที่ 3.11 ไฟล์ชาร์ทที่ใช้ในการเขียนโปรแกรมควบคุมการเดินทางจากสถานีต้นทางไปยังสถานีปลายทาง

3.8.4 ไฟล์ชาร์ตแสดงโปรแกรมควบคุมการเดินทางจากสถานีปลายทางไปยังสถานีต้นทาง



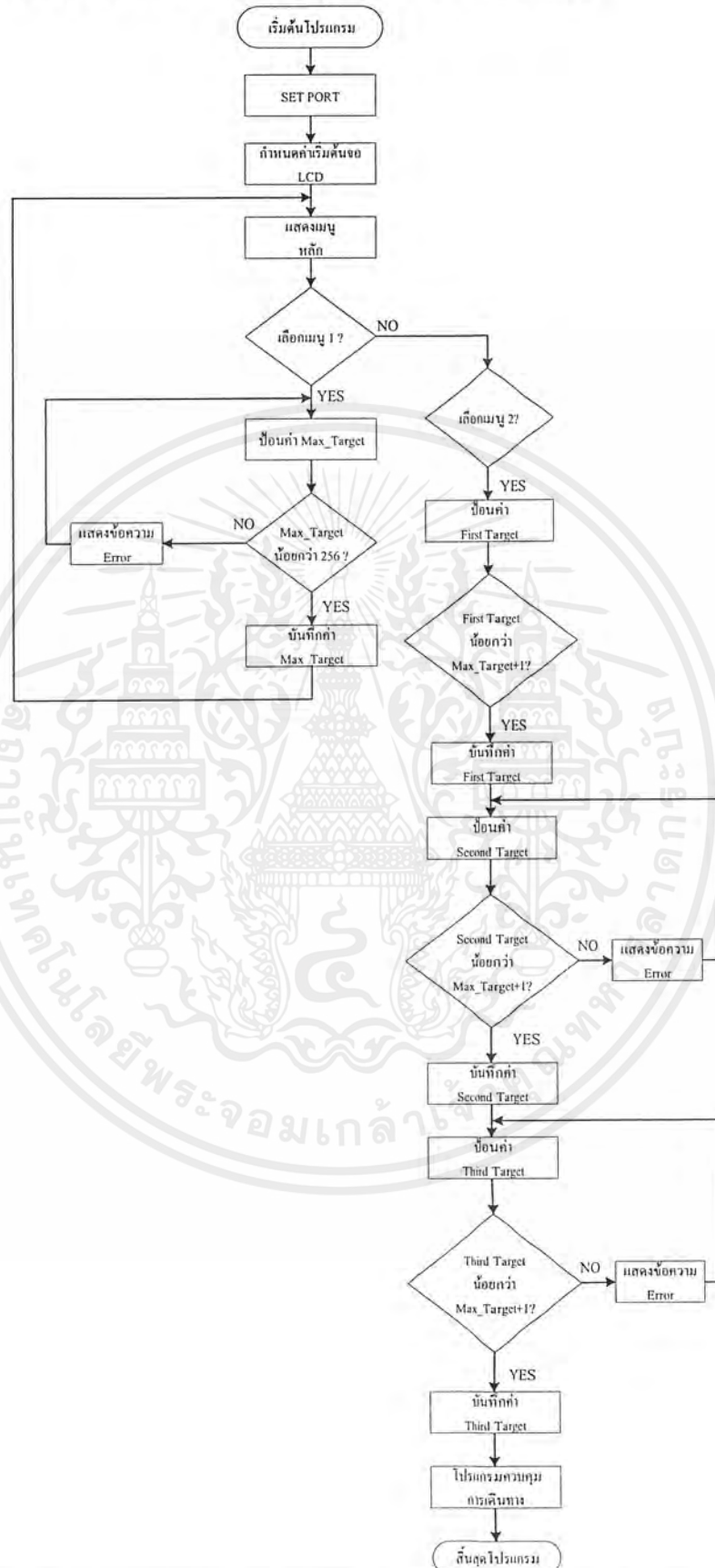
รูปที่ 3.12 ไฟล์ชาร์ตที่ใช้ในการเขียนโปรแกรมควบคุมการเดินทางจากสถานีปลายทางไปยังสถานีต้นทาง

3.8.5 โฟลว์ชาร์ตแสดงโปรแกรมโดยรวมในการควบคุมหุ่นยนต์



รูปที่ 3.13 โฟลว์ชาร์ตโดยรวมที่ใช้ในการเขียนโปรแกรมควบคุมหุ่นยนต์

3.8.6 ไฟล์ชาร์ตแสดงโปรแกรมควบคุมจอแสดงผลแบบผลึกเหลว



รูปที่ 3.14 ไฟล์ชาร์ตโดยรวมที่ใช้ในการเขียนโปรแกรมควบคุมจอแสดงผลแบบผลึกเหลว

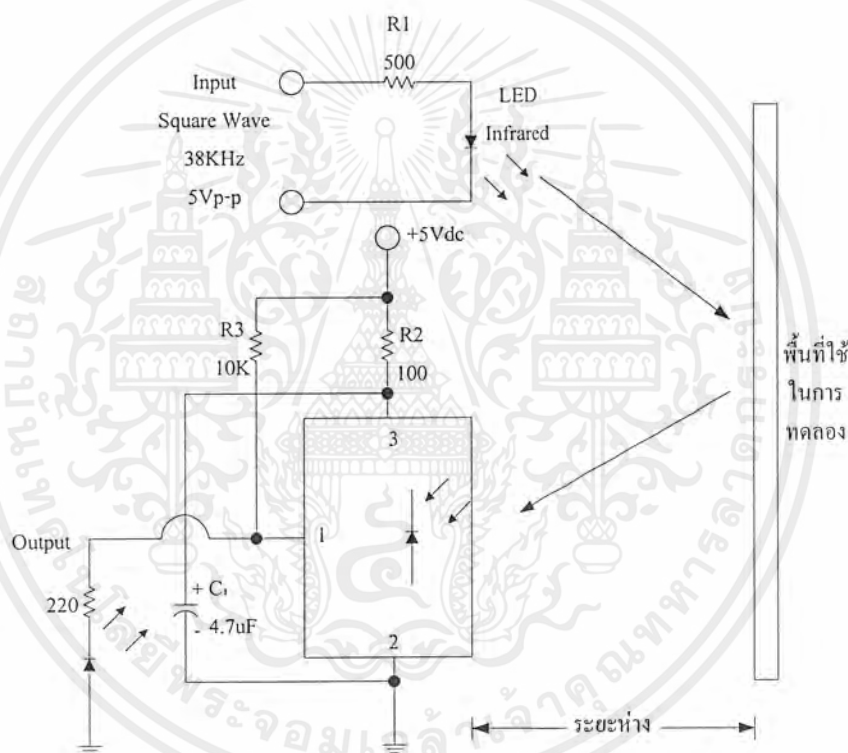
บทที่ 4

การทดลองและผลการทดลอง

ในส่วนของการทดลองนั้นแยกออกเป็นส่วนต่างๆ ดังนี้

4.1 การทดลองอุปกรณ์ตรวจจับเส้นทาง

ทำการวัดระดับเอาท์พุทของตัวโฟโตโมดูลเพื่อจะได้ทราบถึงคุณสมบัติเฉพาะตัวของโฟโตโมดูล ในขณะที่มีสีพื้น และระยะในการสะท้อนแสงต่างๆ กัน ในการทดลองจะค่อยๆ ทำการเปลี่ยนระยะทางระหว่างอุปกรณ์ตรวจจับเส้นทางกับพื้นสีเดียวกันก่อนจากนั้นจึงทำการเปลี่ยนแปลงสีพื้นโดยใช้ระยะทางระหว่างอุปกรณ์ตรวจจับเส้นทางกับพื้นมีระยะทางเท่าเดิม



รูปที่ 4.1 วงจรที่ใช้ในการทดลองอุปกรณ์ตรวจจับเส้นทาง

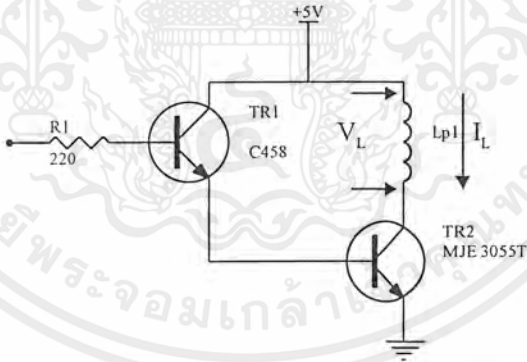
	ลอจิกเอาต์พุตที่แอลอีดี						
	ระยะห่าง	ระยะห่าง	ระยะห่าง	ระยะห่าง	ระยะห่าง	ระยะห่าง	ระยะห่าง
	1.5 cm.	1.75 cm.	2 cm.	2.25 cm.	2.5 cm.	2.75 cm.	3 cm.
ดำ	1	1	1	1	1	1	1
เขียว	1	1	1	1	1	1	1
แดง	0	0	0	1 กับ 0 สลับกัน	0	1	1
น้ำเงิน	0	0	0	0	0	1	1
เหลือง	0	0	0	0	0	1	1
ขาว	0	0	0	0	0	0	1

ตารางที่ 4.1 ค่าลอจิกเอาต์พุตที่ระยะห่างและสีต่างๆ

4.2 การทดสอบวงจรขับเคลื่อนมอเตอร์

4.2.1 การทดลองวงจรขับเคลื่อนสเต็ปมอเตอร์

ทำการทดลองใช้ลอจิกไปควบคุมการสวิทช์ของทรานซิสเตอร์ในวงจรขับเคลื่อนสเต็ปมอเตอร์ และทำการวัดค่าต่างๆตามตารางที่ 4.2 ในการทดลองจะทำการต่อวงจรตามรูปที่ 4.2 ซึ่งเป็นวงจรที่ใช้ขับเคลื่อนสเต็ปมอเตอร์ในแต่ละเฟส ซึ่งในการใช้งานจริงจะประกอบด้วยวงจรตามรูปที่ 4.2 จำนวน 4 วงจร



รูปที่ 4.2 วงจรขับเคลื่อนสเต็ปมอเตอร์ที่ใช้ในการทดลอง

ลอจิก	TR ₁		TR ₂		V _R (โวลต์)	V _L (โวลต์)	I _L (แอมแปร์)
	V _{BE} (โวลต์)	V _{CE} (โวลต์)	V _{BE} (โวลต์)	V _{CE} (โวลต์)			
0	0	4.4	0	5	0	0	0
1	0.75	3.8	0.8	0.15	0.05	4.5	0.85

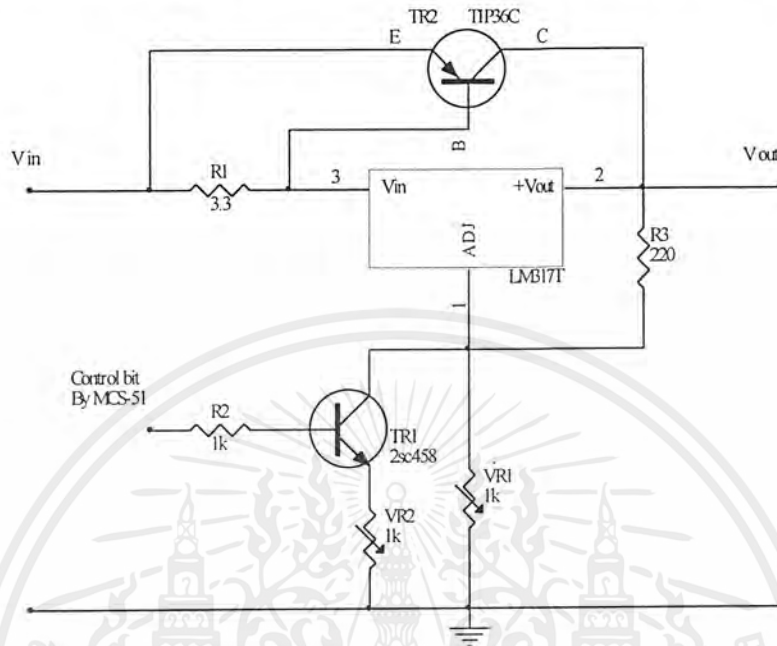
ตารางที่ 4.2 ผลการทดลองของวงจรขับเคลื่อนสเต็ปมอเตอร์

สถานะของรถ	สถานะของบิตควบคุม (Control bit)						แรงดันที่มอเตอร์ซ้าย(โวลท์)				แรงดันที่มอเตอร์ขวา(โวลท์)				กระแสมีโหลด	
	b ₅	b ₄	b ₃	B ₂	b ₁	b ₀	ไม่มีโหลด		มีโหลด		ไม่มีโหลด		มีโหลด		มอเตอร์ซ้าย (แอมแปร์)	มอเตอร์ขวา (แอมแปร์)
							จุด 1	จุด 2	จุด 1	จุด 2	จุด 1	จุด 2	จุด 1	จุด 2		
เดินหน้า (ความเร็วต่ำ)	0	1	1	1	0	1	0	2.7	0.2	2.2	0	2.7	0.2	2.2	0.7	0.7
เดินหน้า (ความเร็วสูง)	0	1	0	0	0	1	0	4	0.2	3.7	0	4	0.2	3.7	0.8	0.8
ถอยหลัง (ความเร็วต่ำ)	1	0	1	1	1	0	2.7	0	2	0.2	2.7	0	2	0.2	0.7	0.7
ถอยหลัง (ความเร็วสูง)	1	0	0	0	1	0	4	0	3.7	0.2	4	0	3.7	0.2	0.8	0.8
หยุด	0	0	0	0	0	0	4	4	4.2	4.2	4	4	4.2	4.2	0	0
เดินหน้า (เลี้ยวซ้าย)	0	1	1	0	0	1	0	2.7	0.2	2.4	0	4	0.2	3.6	0.7	0.8
เดินหน้า (เลี้ยวขวา)	0	1	0	1	0	1	0	4	0.2	3.6	0	2.7	0.2	2.3	0.8	0.7
ถอยหลัง (เลี้ยวซ้าย)	1	0	1	0	1	0	2.4	0	2.4	0.2	4	0	3.6	0.2	0.7	0.8
ถอยหลัง (เลี้ยวขวา)	1	0	0	1	1	0	3.7	0	3.7	0.2	2.7	0	2.3	0.2	0.8	0.7

ตารางที่ 4.3 ผลการทดลองวงจรขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรง

4.3 การทดลองวงจรเรกูเลเตอร์แบบปรับค่าได้ 2 ระดับ

ทำการทดสอบคุณสมบัติของการใช้ลอจิกไปควบคุมการปรับระดับแรงดันของวงจร โดยการทดลองจะทำการต่อวงจรตามรูปที่ 4.4 และ ทำการวัดแรงดันเอาต์พุตและบันทึกค่าลงในตารางที่ 4.4



รูปที่ 4.4 การต่อวงจรเรกูเลเตอร์แบบปรับค่าได้ 2 ระดับที่ใช้ในการทดลอง

สถานะของมอเตอร์	สถานะของลอจิก	TR1	Voutput
High Speed	"0"	OFF	5 V
Low Speed	"1"	ON	3.6 V

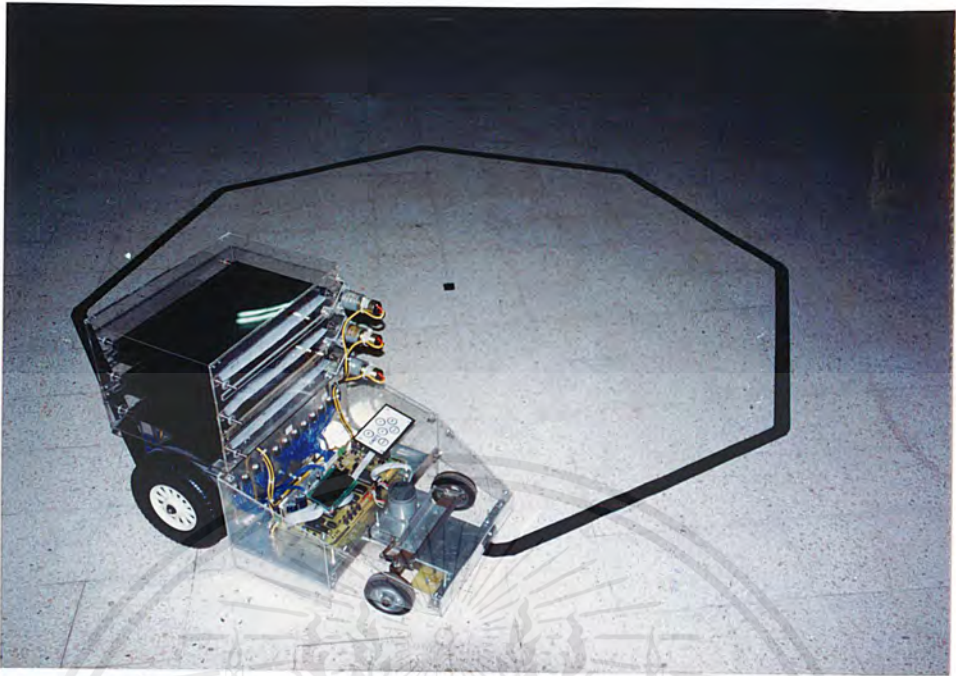
ตารางที่ 4.4 ผลการทดลองวงจรเรกูเลเตอร์แบบปรับค่าได้ 2 ระดับ

4.4 การทดลองรัศมีวงเลี้ยวของตัวหุ่นยนต์

ทำการทดลองโดยให้ตัวหุ่นยนต์เคลื่อนที่ในเส้นทางที่เป็นวงกลมในรัศมีที่มีค่าแตกต่างกัน

รัศมีของวงกลม	ผลการทดลอง
0.50 เมตร	หุ่นยนต์เคลื่อนที่ออกนอกเส้นทาง
0.75 เมตร	หุ่นยนต์เคลื่อนที่ได้โดยไม่หลุดออกนอกเส้นทาง แต่จะเกาะขอบเส้นทางที่ได้ทำการวางไว้
1.00 เมตร	หุ่นยนต์เคลื่อนที่ได้โดยไม่หลุดออกนอกเส้นทาง และตรงตามเส้นทางที่ได้ทำการวางไว้

ตารางที่ 4.5 การเคลื่อนที่ของหุ่นยนต์ในเส้นทางวงกลมที่รัศมีต่างๆ



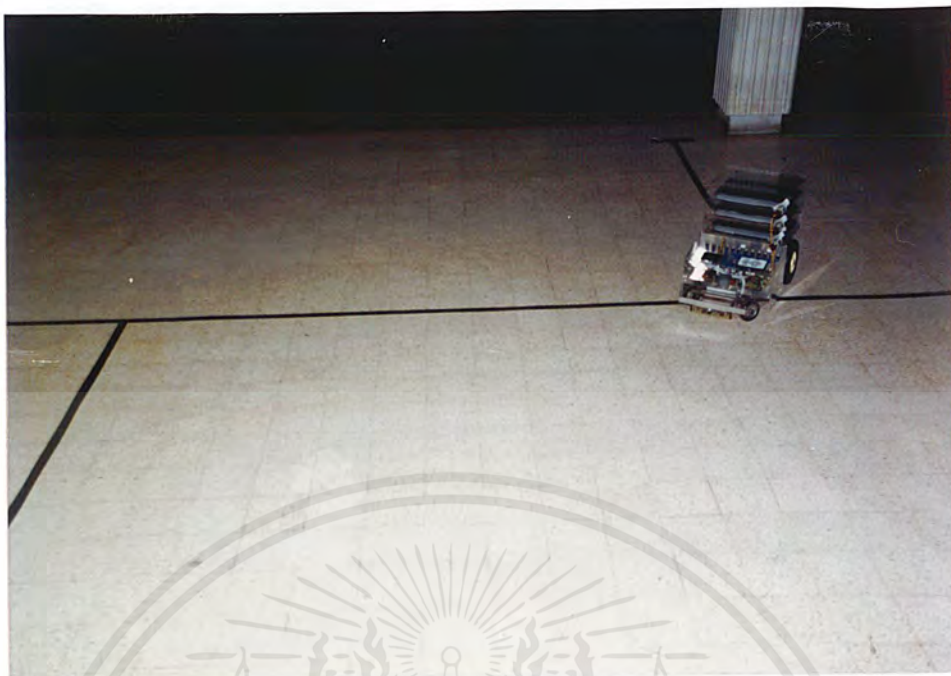
รูปที่ 4.5 การทดลองการเคลื่อนที่ของหุ่นยนต์ในเส้นทางวงกลมที่รัศมีต่างๆ

4.5 การทดลองระยะห่างระหว่างสถานีรับ – ส่งเอกสาร

ทำการทดลองโดยให้ตัวหุ่นยนต์เคลื่อนที่ในเส้นทางใช้งานจริงที่ระยะห่างของแต่ละสถานีแตกต่างกัน

ระยะห่างของแต่ละสถานี	ผลการทดลอง
1.50 เมตร	ไม่สามารถตรวจจับแยกของสถานีถัดไปได้ทัน
2.00 เมตร	สามารถตรวจจับแยกของสถานีถัดไปได้เป็นบางครั้ง บางครั้งไม่สามารถตรวจจับได้
2.50 เมตร	สามารถตรวจจับแยกของสถานีถัดไปได้โดยไม่เกิดความผิดพลาด

ตารางที่ 4.6 การเคลื่อนที่ของหุ่นยนต์ที่ระยะห่างของแต่ละสถานีแตกต่างกัน



รูปที่ 4.6 การทดลองการเคลื่อนที่ของหุ่นยนต์ที่ระยะห่างของแต่ละสถานีแตกต่างกัน

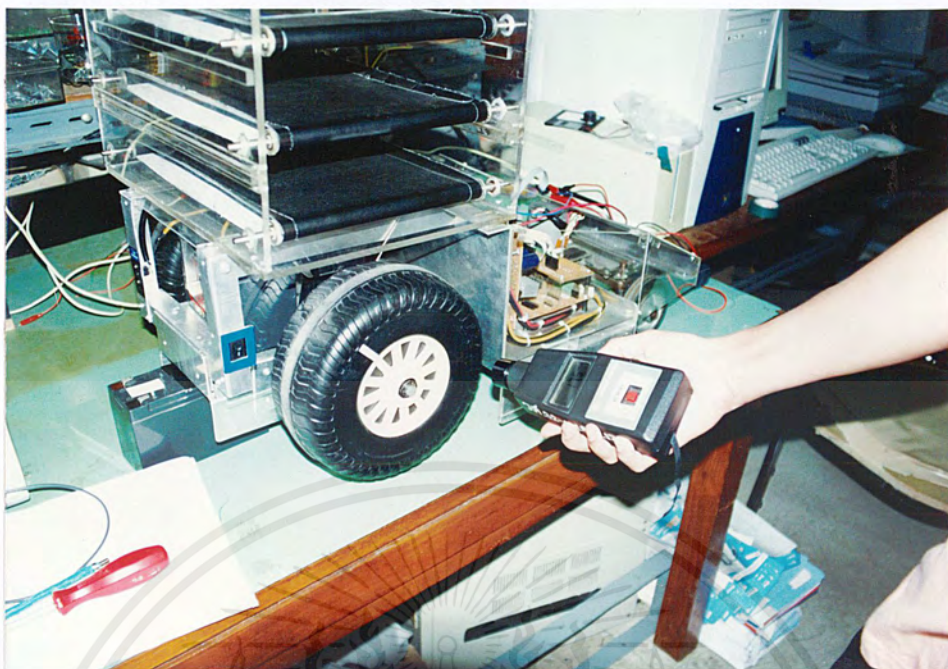
4.6 การทดสอบตัวมอเตอร์

ทำการวัดความเร็วรอบของมอเตอร์ไฟฟ้ากระแสตรงโดยจะป้อนแรงดันที่ระดับต่างๆ ให้กับตัวมอเตอร์แล้วใช้ Tachometer ในการวัดความเร็วรอบของมอเตอร์

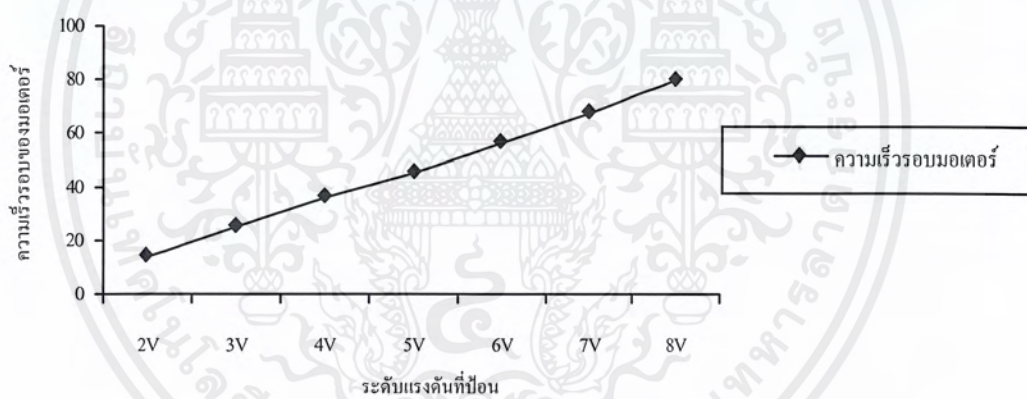
ระดับแรงดันที่ป้อน	ความเร็วรอบมอเตอร์
2 V	13.8 rpm
3 V	25.2 rpm
3.6 V*	31.8 rpm
4 V	36.1 rpm
5 V*	45.3 rpm
6 V	56.2 rpm
7 V	67.3 rpm
8 V	79.8 rpm

ตารางที่ 4.7 ผลการทดลองวัดความเร็วรอบของมอเตอร์

หมายเหตุ * ค่าระดับแรงดันที่ใช้งานจริงในโรงงาน



รูปที่ 4.7 การทดลองวัดความเร็วรอบของมอเตอร์โดยใช้ Tachometer



รูปที่ 4.8 แผนภูมิความเร็วรอบของมอเตอร์ที่ระดับแรงดันต่างๆ

บทที่ 5

บทวิจารณ์และบทสรุป

จากการทดลองและใช้งานจริงของหุ่นยนต์ส่งเอกสารตามเส้น พบว่า สามารถใช้งานในการส่งเอกสารตามจุดต่างๆ ได้อย่างมีประสิทธิภาพสูง ซึ่งในการออกแบบและการสร้างมีปัญหาต่างๆ เกิดขึ้น ดังต่อไปนี้

5.1 ปัญหาทางโครงสร้าง

โครงสร้างหุ่นยนต์ที่ได้ทำการออกแบบในตอนแรกนั้นสร้างจากแผ่นอลูมิเนียม 1 มิลลิเมตร แล้วนำมาพับเป็นโครงหุ่นยนต์ พบว่า เมื่อทำการบรรทุกสิ่งของแล้ว พบว่าโครงที่ออกแบบมีการแอ่นโค้ง ทำให้ในทางปฏิบัติจริงๆ ไม่สามารถนำมาใช้บรรทุกสิ่งของได้

วิธีแก้ไขปัญห โดยการออกแบบโครงสร้างใหม่ โดยมีโครงอลูมิเนียม ตามไว้ตลอดความยาวได้ท้องหุ่นยนต์ และใช้แผ่นอลูมิเนียม 2 มิลลิเมตรในการสร้างโครงแทนแผ่นอลูมิเนียม 1 มิลลิเมตร หลังจากสร้างโครงหุ่นยนต์ที่ได้ออกแบบใหม่ พบว่าไม่เกิดอาการแอ่น โค้งของตัวรถ เหมือนโครงหุ่นยนต์แบบเดิม ทำให้สามารถใช้ในการบรรทุกสิ่งของ เพื่อใช้งานตามวัตถุประสงค์ได้

5.2 ปัญหาของส่วนขับเคลื่อนมอเตอร์ไฟฟ้ากระแสตรง

วงจรที่ได้ทำการออกแบบในครั้งแรกนั้นออกแบบโดยใช้รีเลย์ (Relay) ในการกลับทิศทางการเคลื่อนที่ของมอเตอร์ ทำให้ไม่สามารถใช้งานหุ่นยนต์ได้อย่างเต็มประสิทธิภาพ เนื่องจากวงจรที่ออกแบบจะใช้กระแสไฟในการขับรีเลย์มากส่งผลให้เบตเตอร์หมดเร็ว

วิธีแก้ไขปัญห โดยทำการออกแบบวงจรขับเคลื่อนมอเตอร์ใหม่ที่ใช้ทรานซิสเตอร์ 4 ตัวในการควบคุมทิศทางการหมุนของมอเตอร์แทน ทำให้ประหยัดพลังงานไปได้มาก เนื่องจากทรานซิสเตอร์มีการกินกระแสในช่วงที่ไม่ทำงานน้อยกว่าเมื่อเปรียบเทียบกับรีเลย์ในช่วงที่ไม่ทำงาน

5.3 ปัญหาจากตัวตรวจจบ

5.3.1 ปัญหาจากระยะห่างของอุปกรณ์ตรวจจบเส้นทางและความกว้างของเส้นทางที่ใช้

การออกแบบระยะห่างของ อุปกรณ์ตรวจจบเส้นทางแต่ละชุด และขนาดความกว้างของเส้นทางที่ใช้สำหรับนำทางหุ่นยนต์มีความสำคัญมาก เนื่องจากระยะห่างของอุปกรณ์ตรวจจบเส้นทางแต่ละชุดจะต้องสัมพันธ์กับขนาดของเส้น ในการออกแบบครั้งแรกได้ทำการออกแบบโดยมีระยะห่างของอุปกรณ์ตรวจจบแต่ละชุดมีค่ามากกว่าความกว้างของเส้นทางที่ใช้ในการนำทางเล็กน้อย ก็ยังทำให้ตัวหุ่นยนต์ตรวจสอบเส้นทางได้เป็นว่า ไม่พบเส้นทางทั้งๆ ที่ตัวหุ่นยนต์ยังคร่อมเส้นทางอยู่ ซึ่งมีสาเหตุเนื่องมาจากยังมีแสงอีกส่วนหนึ่งที่ปล่อยจากแอลอีดีอินฟราเรดแล้ว ไม่ถูกดูดซับจากเส้น คือ จะสะท้อนกับพื้นกลับมายังโฟโต้ทรานซิสเตอร์

วิธีแก้ไขปัญห โดยทำการเลือกขนาดของเส้นทางที่ใช้ในการนำทาง และออกแบบระยะห่างของอุปกรณ์ตรวจจบแต่ละชุด ให้มีความเหมาะสมกัน โดยความกว้างของเส้นทางที่ใช้ในการนำทางจะต้องมีค่ามากกว่าระยะห่างของอุปกรณ์ที่ใช้ในการตรวจจบ ไม่ต่ำกว่า 0.5 นิ้ว จึงจะทำให้หุ่นยนต์สามารถทำงานได้โดยไม่เกิดความผิดพลาด

5.3.2 ปัญหาจากสัญญาณรบกวนภายนอก

ในการใช้งานจริงจะพบปัญหาการรบกวนจากแสงภายนอก เช่น แสงจากหลอดไฟ หรือแสงจากดวงอาทิตย์ทำให้อุปกรณ์ตรวจจับเส้นทางทำงานผิดพลาด คือไม่สามารถตรวจพบเส้นทางได้เลยเนื่องจากมีแสงสะท้อนไปยังโฟโตทรานซิสเตอร์ตลอดเวลา

วิธีแก้ไขปัญหานี้ โดยการใช้การส่งสัญญาณแบบโทนเบิร์ตแทนการส่งสัญญาณอินฟราเรดแบบเก่าและใช้โฟโตโมดูลทางด้านรับทำให้แสงจากภายนอกไม่สามารถรบกวนอุปกรณ์ตรวจจับเส้นทางได้

5.4 ปัญหาจากโปรแกรมควบคุมหุ่นยนต์

การเขียนโปรแกรมที่ใช้ในการควบคุมหุ่นยนต์เคยใช้หลักการตรวจสอบ สถานะลอจิกของอุปกรณ์ตรวจจับเส้นทาง แล้วนำมาทำการเปรียบเทียบกับ รูปแบบที่ได้มีการกำหนดไว้ล่วงหน้า ถ้าหากตรวจสอบแล้วตรงตามเส้นทาง โปรแกรมก็จะควบคุมให้มอเตอร์หมุนไปข้างหน้า ถ้าตรวจสอบแล้วไม่พบเส้นทาง โปรแกรมก็จะควบคุมให้มอเตอร์หยุดหมุน ในกรณีที่ตรวจสอบแล้วไม่ตรงตามเส้นทาง โปรแกรมก็จะควบคุมสเต็ปมอเตอร์ให้หมุนเพื่อทำการหักเลี้ยวทิศทางใดทิศทางหนึ่ง เป็นสเต็ปที่ได้มีการกำหนดไว้ แล้วจะทำการหมุนกลับเพื่อให้สเต็ปมอเตอร์กลับมายังตำแหน่งเดิม ซึ่งในขณะที่โปรแกรมทำงานอยู่กับสเต็ปมอเตอร์นี้ ก็จะสามารถตรวจสอบสถานะของอุปกรณ์ตรวจจับเส้นทางได้จนกว่าโปรแกรมจะทำงานเสร็จ ซึ่งในช่วงเวลานั้นหากตัวหุ่นยนต์ผ่านแยกต่าง ๆ ก็จะทำให้ไม่สามารถรับรู้ได้เลยว่าได้เคลื่อนที่ผ่านแยกมาแล้ว ทำให้เกิดความผิดพลาดในเส้นทางเคลื่อนที่ของหุ่นยนต์

วิธีแก้ไขปัญหานี้ ทำการเขียนโปรแกรมโดยใช้ อินเตอร์พพท์ - ไทมเมอร์ ในการตรวจสอบ สถานะลอจิกของอุปกรณ์ตรวจจับเส้นทางแทน ซึ่งจะทำให้การตรวจสอบสถานะลอจิกเป็นช่วงเวลาที่ทำกันโดยตลอด

5.5 ปัญหาจากระยะห่างของแต่ละสถานี

ในการใช้งานจริงจะพบปัญหาหุ่นยนต์ไม่สามารถตรวจจับแยกสถานีถัดไปได้ทัน เนื่องจากมีการวางระยะห่างของแต่ละสถานีสั้นเกินไปแต่วงเลี้ยวของตัวหุ่นยนต์นั้นมีรัศมีกว้าง ทำให้หุ่นยนต์เคลื่อนที่ผ่านแยกของสถานีถัดไปก่อนที่จะทำโปรแกรมในการเลี้ยวเสร็จ ส่งผลให้หุ่นยนต์ไม่พบแยกดังกล่าว

วิธีแก้ไขปัญหานี้ โดยทำการออกแบบเส้นทางเคลื่อนที่ของหุ่นยนต์ให้มีระยะห่างของแต่ละสถานีไม่ต่ำกว่า 2.5 เมตร



```

#####
;           Line Tracking Car Program Version 1.3 can be select 3 section Project by Telecommunication engineer
#####

        org         0000h

;----- define port -----
porta1   equ        0e000h
portb1   equ        0e001h
portc1   equ        0e002h
portp1   equ        0e003h
porta2   equ        0c000h
portb2   equ        0c001h
portc2   equ        0c002h
portp2   equ        0c003h
porta3   equ        0f000h
portb3   equ        0f001h
portc3   equ        0f002h
portp3   equ        0f003h
exram    equ        0d000h

;----- define keyboard -----
ok       bit        p1.0
cancle   bit        p1.1
left1    bit        p1.2
right    bit        p1.3
up       bit        p1.4
down     bit        p1.5

;----- define user register -----
status   equ        031h
temp     equ        032h
temp2    equ        033h
temp1    equ        034h
temp0    equ        035h
max_target equ      036h
digit2   equ        037h
digit1   equ        038h
digit0   equ        039h
count1   equ        03ah
count2   equ        03bh
count3   equ        03ch
count4   equ        03dh
choice   equ        03eh
target   equ        03fh
chk_tar  equ        076h
sure255  equ        077h

;----- define register 2 -----
sec_value equ      046h
backup_a  equ      049h
;r6 use for sample timing
;r7 use for compare condition
;turn_value equ    045h
;r4 use for save turn value
delay1_v  equ      050h

;----- new define register -----
qs        equ      047h
xs        equ      046h
ys        equ      070h

```

```

        zs      equ      071h
        ys1     equ      07ah
        zs1     equ      07bh
        select_tray equ    072h

;-----
        ljmp    start
;-----
;#####
;##### interrupt timer address #####
;#####

        org     000bh
        ljmp    ctime
        org     001bh
        ljmp    rtime

;#####
;##### start program #####
;#####

        org     0030h
start:   mov     p1,#0ffh
        mov     r1,#30h
clriram: mov     @r1,#00h
        inc     r1
        cjne    r1,#7fh,clriram
        mov     a,#80h
        mov     dptr,#portp1
        movx    @dptr,a
        mov     a,#90h
        mov     dptr,#portp2
        movx    @dptr,a
        mov     dptr,#portp3
        mov     a,#80h
        movx    @dptr,a
        mov     a,#00h
        mov     dptr,#portc1 ;clr stepping motor port
        movx    @dptr,a
        mov     dptr,#portc2 ;clr drive motor port
        movx    @dptr,a
        mov     dptr,#porta3 ;clr tray motor port
        movx    @dptr,a
        mov     a,#00h
        clr     a ;correct 2 line here !
        mov     max_target,a
        mov     chk_tar,a

;#####
;##### program lcd pb0 = enable pb1 = rs #####
;#####

start_lcd: clr     a
        mov     p1,#0ffh ;clear port1 for input port
        mov     dptr,#porta1
        movx    @dptr,a ;clear porta
        mov     dptr,#portb1
        movx    @dptr,a ;clear portb

init_lcd: lcall    delay_100ms
        mov     a,#00111000b ;set 8 bit mode
        lcall    lcd_wi

```

```

lcall    delay_10ms
mov      a,#00001000b    ;set display off
lcall    lcd_wi
mov      a,#00000001b    ;set clear display
lcall    lcd_wi
mov      a,#00000110b    ;set entry mode
lcall    lcd_wi
mov      a,#00000010b    ;set return home display
lcall    lcd_wi
mov      a,#0cfh         ;set address 00h [line 1]
lcall    lcd_wi
mov      a,#0eh          ;display on
lcall    lcd_wi
lcall    delay_1s
clr      a
mov      target,a
mov      temp,a
mov      digit2,a
mov      digit1,a
mov      digit0,a
mov      status,a

```

```

mov      a,#'M'
lcall    char_run_l
mov      a,#'E'
lcall    char_run_l
mov      a,#'S'
lcall    char_run_l
mov      a,#'S'
lcall    char_run_l
mov      a,#'E'
lcall    char_run_l
mov      a,#'N'
lcall    char_run_l
mov      a,#'G'
lcall    char_run_l
mov      a,#'i'
lcall    char_run_l
mov      a,#'N'
lcall    char_run_l
mov      a,#'G'
lcall    char_run_l
mov      a,#' '
lcall    char_run_l
mov      a,#'R'
lcall    char_run_l
mov      a,#'O'
lcall    char_run_l
mov      a,#'B'
lcall    char_run_l
mov      a,#'O'
lcall    char_run_l
mov      a,#'T'
lcall    wrchar_lcd
lcall    delay_1s

```

```

lcall    blink_lcd
lcall    delay_1s
mov      a,#01h;clear lcd
lcall    lcd_wi
mov      a,#0c0h          ;set address 40h [ line 2 ]
lcall    lcd_wi
mov      dptr,#title_1
lcall    wrline_lcd
mov      a,#090h          ;set address 10h [ line 3 ]
lcall    lcd_wi
mov      dptr,#title_2
lcall    wrline_lcd
lcall    delay_1s
mov      a,#08h
lcall    lcd_wi
lcall    delay_100ms
lcall    delay_100ms
lcall    delay_100ms
mov      a,#0ch
lcall    lcd_wi
lcall    delay_100ms
lcall    delay_100ms
lcall    delay_1s
;-----
; main menu
;-----
main_menu: mov      a,#01h;clear lcd
lcall    lcd_wi
mov      a,#080h          ;set address 00h [ line 1 ]
lcall    lcd_wi
mov      dptr,#title_3
lcall    wrline_lcd
mov      a,#0c0h          ;set address 40h [ line 2 ]
lcall    lcd_wi
mov      dptr,#title_4
lcall    wrline_lcd
mov      a,#090h          ;set address 10h [ line 3 ]
lcall    lcd_wi
mov      dptr,#title_5
lcall    wrline_lcd
mov      a,#0d0h          ;set address 50h [ line 4 ]
lcall    lcd_wi
mov      dptr,#title_6
lcall    wrline_lcd
mov      a,#0deh          ;set cursor position
lcall    lcd_wi
mov      a,#'1'
lcall    lcd_wd
mov      a,#0fh;curser blink
lcall    lcd_wi
mov      a,#0deh
lcall    lcd_wi
mov      a,#1
mov      choice,a          ;clear register choice
menu:     jnb      up,up_c

```

```

        jnb     down,down_c
        jnb     ok,chk1
        ljmp    menu
up_c:   ljmp    up_choice
down_c: ljmp    down_choice
chk1:   ljmp    check_choice
check_choice: mov a,choice
        cjne   a,#1,next
        ljmp    check_target
next:   cjne   a,#2,menu
        ljmp    setup
wait:   lcall   delay_1s      ;out of program
        ljmp    wait

```

setup robot sub

```

setup:  clr     a
        mov     status,a
        mov     a,#01h      ;clear display
        lcall   lcd_wi
        mov     a,#080h
        lcall   lcd_wi
        mov     dptr,#title_7
        lcall   wrline_lcd
        clr     a
        mov     max_target,a
        lcall   value_target
        mov     max_target,target
        mov     a,max_target
        lcall   conv
        mov     a,#01h
        lcall   lcd_wi
        mov     a,#0c0h      ;set address 40h [ line 2 ]
        lcall   lcd_wi
        mov     dptr,#title_11
        lcall   wrline_lcd
        mov     a,#090h;set address 10h [ line 3 ]
        lcall   lcd_wi
        mov     dptr,#title_12
        lcall   wrline_lcd
        mov     a,#09ch
        lcall   lcd_wi
        mov     a,digit2
        lcall   case_0
        mov     a,#09dh
        lcall   lcd_wi
        mov     a,digit1
        lcall   case_0
        mov     a,#09eh
        lcall   lcd_wi
        mov     a,digit0
        lcall   case_0
        lcall   delay_1s
        inc     status
        ljmp    main_menu

```

```

error1:  lcall    error
         ljmp     check_target
error2:  lcall    error
         ljmp     check_target1
resetup: mov     a,#01h
         lcall    lcd_wi
         mov     a,#0c0h
         lcall    lcd_wi
         mov     dptr,#title_18
         lcall    wrline_lcd
         mov     a,#090h
         lcall    lcd_wi
         mov     dptr,#title_19
         lcall    wrline_lcd
         lcall    delay_1s
         lcall    delay_1s
         lcall    delay_1s
         ljmp     main_menu

```

```

-----
:         check target sub
-----

```

```

check_target: mov     a,status
             cjne    a,#1,resetup
             mov     a,#01h           ;clear display
             lcall    lcd_wi
             mov     a,#080h
             lcall    lcd_wi
             mov     dptr,#title_8
             lcall    wrline_lcd
             lcall    value_target
             lcall    check_error
             jc       error1
             mov     xs,target

```

```

check_target1: mov    a,#01h           ;clear display
                 lcall    lcd_wi
                 lcall    wrline_xs
                 mov     a,#080h
                 lcall    lcd_wi
                 mov     dptr,#title_9
                 lcall    wrline_lcd
                 lcall    value_target
                 lcall    check_error
                 jc       error2
                 mov     ys,target

```

```

check_target2: mov    a,#01h           ;clear display
                 lcall    lcd_wi
                 lcall    wrline_xs
                 lcall    wrline_ys
                 mov     a,#080h
                 lcall    lcd_wi
                 mov     dptr,#title_10
                 lcall    wrline_lcd
                 lcall    value_target
                 lcall    check_error
                 jc       error3

```

```

mov     zs,target
lcall   wrline_xs
lcall   wrline_ys
lcall   wrline_zs
mov     a,#01h           ;clear display
lcall   lcd_wi
mov     a,#0c0h
lcall   lcd_wi
mov     dptr,#title_16
lcall   wrline_lcd
lcall   delay_1s
mov     a,#01h           ;clear display
lcall   lcd_wi
lcall   wrline_xs
lcall   wrline_ys
lcall   wrline_zs
mov     a,#0c0h
lcall   lcd_wi
mov     dptr,#title_17
lcall   wrline_lcd
lcall   delay_1s
lcall   delay_1s
ljmp    start1           ;jump to program start1
error3: lcall   error
ljmp    check_target2

;-----
;      write line xs ys zs sub
;-----
wrlne_xs: mov     a,xs
lcall   conv
mov     a,#0d1h
lcall   lcd_wi
mov     a,##"
lcall   wrchar_lcd
mov     a,#0d2h
lcall   lcd_wi
mov     a,digit2
lcall   case_0
mov     a,#0d3h
lcall   lcd_wi
mov     a,digit1
lcall   case_0
mov     a,#0d4h
lcall   lcd_wi
mov     a,digit0
lcall   case_0
ret
wrlne_ys: mov     a,ys
lcall   conv
mov     a,#0d6h
lcall   lcd_wi
mov     a,##"
lcall   wrchar_lcd
mov     a,#0d7h
lcall   lcd_wi

```

```

        mov     a,digit2
        lcall   case_0
        mov     a,#0d8h
        lcall   lcd_wi
        mov     a,digit1
        lcall   case_0
        mov     a,#0d9h
        lcall   lcd_wi
        mov     a,digit0
        lcall   case_0
        ret

wrlne_zs: mov     a,zs
        lcall   conv
        mov     a,#0dbh
        lcall   lcd_wi
        mov     a,##
        lcall   wrchar_lcd
        mov     a,#0dch
        lcall   lcd_wi
        mov     a,digit2
        lcall   case_0
        mov     a,#0ddh
        lcall   lcd_wi
        mov     a,digit1
        lcall   case_0
        mov     a,#0deh
        lcall   lcd_wi
        mov     a,digit0
        lcall   case_0
        ret

;-----
;               check error sub
;-----
check_error: mov     a,max_target
        mov     b,target
        clr     c
        subb    a,b
        ret

;-----
;               show error_sub
;-----
error:      mov     a,#01h
        lcall   lcd_wi
        mov     a,#0c0h           ;set address 40h [ line 2 ]
        lcall   lcd_wi
        mov     dptr,#title_13
        lcall   wrlne_lcd
        lcall   delay_1s
        mov     a,#01h
        lcall   lcd_wi
        mov     a,#0c0h           ;set address 40h [ line 2 ]
        lcall   lcd_wi
        mov     dptr,#title_14
        lcall   wrlne_lcd
        mov     a,#090h           ;set address 10h [ line 3 ]

```

```

        lcall    lcd_wi
        mov     dptr,#title_15
        lcall    wrline_lcd
        mov     a,max_target
        lcall    conv
        mov     a,#09bh
        lcall    lcd_wi
        mov     a,digit2
        lcall    case_0
        mov     a,#09ch
        lcall    lcd_wi
        mov     a,digit1
        lcall    case_0
        mov     a,#09dh
        lcall    lcd_wi
        mov     a,digit0
        lcall    case_0
        lcall    delay_1s
        lcall    delay_1s
        ret

;-----
;          value_target sub
;-----
value_target: mov     a,#0c7h      ;set center line 3
        lcall    lcd_wi
        mov     a,#0'
        lcall    lcd_wd
        mov     a,#0fh      ;curser blink
        lcall    lcd_wi
        mov     a,#0c7h
        lcall    lcd_wi
        clr     a
        mov     temp2,a
state_1: lcall    delay_100ms
        jnb     ok,keep2      ;start state 1
        jnb     up,up_2
        jnb     cangle,go_main
        jnb     right,state_n
        jb      down,state_1
        ljmp     down_2
state_n: ljmp     state_nd
go_main: ljmp     main_menu
up_2:    ljmp     up_temp2
down_2:  ljmp     down_temp2
keep2:   ljmp     keep_temp2
state_nd: mov     a,#0c8h      ;set center line 3
        lcall    lcd_wi
        mov     a,#0'
        lcall    lcd_wd
        mov     a,#0fh      ;curser blink
        lcall    lcd_wi
        mov     a,#0c8h
        lcall    lcd_wi
        clr     a
        mov     temp1,a

```

```

state_2: lcall    delay_100ms
         lcall    delay_100ms
         jnb     ok,keep1
         jnb     candle,go_main
         jnb     left1,re_temp2
         jnb     right,state_r
         jnb     up,up1
         jnb     down,state_2
         ljmp    down1
state_r: ljmp     state_rd
up1:    ljmp     up_temp1
down1:  ljmp     down_temp1
keep1:  ljmp     keep_temp1
re_temp2: mov    a,#0c8h          ;set center line 3
         lcall    lcd_wi
         mov     a,#' '
         lcall    lcd_wd
         ljmp     value_target
state_rd: mov    a,#0c9h          ;set center line 3
         lcall    lcd_wi
         mov     a,#'0'
         lcall    lcd_wd
         mov     a,#0fh          ;curser blink
         lcall    lcd_wi
         mov     a,#0c9h
         lcall    lcd_wi
         clr     a
         mov     temp0,a
         mov     sure255,a
state_3: lcall    delay_100ms
         lcall    delay_100ms
         mov     a,temp2
         mov     b,#10
         mul     ab
         add     a,temp1
         mov     sure255,a
         cjne    a,#25,state_a
         jnb     ok,keep0
         jnb     candle,go_mainx
         jnb     left1,re_temp1
         jnb     up,up05
         jnb     down,state_3
         ljmp    down05
go_mainx: ljmp    go_main
keep0:   ljmp     keep_temp0
up05:    ljmp     up_temp05
down05:  ljmp     down_temp05
re_temp1: mov    a,#0c9h          ;set center line 3
         lcall    lcd_wi
         mov     a,#' '
         lcall    lcd_wd
         ljmp     state_nd
state_a: lcall    delay_100ms
         lcall    delay_100ms
         mov     b,sure255

```

```

        mov     a,#25
        clr     c
        subb    a,b
        jc      state_b
        jnb     ok,keep0
        jnb     candle,go_mainx
        jnb     left1,re_temp1
        jnb     up,up0
        jnb     down,state_a
        ljmp    down0
up0:    ljmp    up_temp0
down0:  ljmp    down_temp0
state_b: mov     a,#0c9h          ;set center line 3
        lcall   lcd_wi
        mov     a,#' '
        lcall   lcd_wd
        mov     a,#0c8h          ;set center line 3
        mov     a,#0c8h          ;set center line 3
        lcall   lcd_wi
        mov     a,temp1
        lcall   case_0
        mov     a,#0fh           ;curser blink
        lcall   lcd_wi
        mov     a,#0c8h
        lcall   lcd_wi
state_c: lcall   delay_100ms
        lcall   delay_100ms
        jnb     ok,keep1x
        jnb     candle,go_mainx
        jnb     left1,re_temp2x
        jnb     up,up1x
        jnb     down,state_c
        ljmp    down1x
keep1x: ljmp    keep_temp1
re_temp2x: ljmp re_temp2
up1x:    ljmp    up_temp1
down1x:  ljmp    down_temp1
end_value: ret

```

convert number

```

conv:   mov     b,#100
        div     ab
        mov     digit2,a
        mov     a,b
        mov     b,#10
        div     ab
        mov     digit1,a
        mov     digit0,b
        ret

```

show number

```

case_0: cjne     a,#0,case_1
        mov     a,#'0'

```

```

        lcall    wrchar_lcd
        ret
case_1:  cjne    a,#1,case_2
        mov     a,#'1'
        lcall    wrchar_lcd
        ret
case_2:  cjne    a,#2,case_3
        mov     a,#'2'
        lcall    wrchar_lcd
        ret
case_3:  cjne    a,#3,case_4
        mov     a,#'3'
        lcall    wrchar_lcd
        ret
case_4:  cjne    a,#4,case_5
        mov     a,#'4'
        lcall    wrchar_lcd
        ret
case_5:  cjne    a,#5,case_6
        mov     a,#'5'
        lcall    wrchar_lcd
        ret
case_6:  cjne    a,#6,case_7
        mov     a,#'6'
        lcall    wrchar_lcd
        ret
case_7:  cjne    a,#7,case_8
        mov     a,#'7'
        lcall    wrchar_lcd
        ret
case_8:  cjne    a,#8,case_9
        mov     a,#'8'
        lcall    wrchar_lcd
        ret
case_9:  cjne    a,#9,case_0
        mov     a,#'9'
        lcall    wrchar_lcd
        ret

```

```

;-----
;               inc temp0
;-----

```

```

up_temp0: inc     temp0
          mov     a,temp0
zero_up0: cjne    a,#0,one_up0
          mov     a,#'0'
          ljmp    scan_temp0
one_up0:  cjne    a,#1,two_up0
          mov     a,#'1'
          ljmp    scan_temp0
two_up0:  cjne    a,#2,three_up0
          mov     a,#'2'
          ljmp    scan_temp0
three_up0: cjne    a,#3,four_up0
          mov     a,#'3'
          ljmp    scan_temp0

```

```

four_up0: cjne    a,#4,five_up0
          mov     a,#'4'
          ljmp    scan_temp0
five_up0:  cjne    a,#5,six_up0
          mov     a,#'5'
          ljmp    scan_temp0
six_up0:   cjne    a,#6,seven_up0
          mov     a,#'6'
          ljmp    scan_temp0
seven_up0: cjne    a,#7,eight_up0
          mov     a,#'7'
          ljmp    scan_temp0
eight_up0: cjne    a,#8,nine_up0
          mov     a,#'8'
          ljmp    scan_temp0
nine_up0:  cjne    a,#9,reset_up0
          mov     a,#'9'
          ljmp    scan_temp0
reset_up0: clr     a
          mov     temp0,a
          mov     a,#'0'
          ljmp    scan_temp0

;-----
;               dec temp0
;-----
down_temp0: dec    temp0
          mov     a,temp0
nine_down0: cjne    a,#9,eight_down0
          mov     a,#'9'
          ljmp    scan_temp0
eight_down0: cjne    a,#8,seven_down0
          mov     a,#'8'
          ljmp    scan_temp0
seven_down0: cjne    a,#7,six_down0
          mov     a,#'7'
          ljmp    scan_temp0
six_down0:  cjne    a,#6,five_down0
          mov     a,#'6'
          ljmp    scan_temp0
five_down0: cjne    a,#5,four_down0
          mov     a,#'5'
          ljmp    scan_temp0
four_down0: cjne    a,#4,three_down0
          mov     a,#'4'
          ljmp    scan_temp0
three_down0: cjne    a,#3,two_down0
          mov     a,#'3'
          ljmp    scan_temp0
two_down0:  cjne    a,#2,one_down0
          mov     a,#'2'
          ljmp    scan_temp0
one_down0:  cjne    a,#1,zero_down0
          mov     a,#'1'
          ljmp    scan_temp0
zero_down0: cjne    a,#0,set_down0

```

```

        mov     a,#'0'
        ljmp    scan_temp0
set_down0: mov     a,#9
        mov     temp0,a
        mov     a,#'9'
        ljmp    scan_temp0

```

```

;          inc temp05
;

```

```

up_temp05: inc     temp0
        mov     a,temp0
zero_up05: cjne    a,#0,one_up05
        mov     a,#'0'
        ljmp    scan_temp0
one_up05:  cjne    a,#1,two_up05
        mov     a,#'1'
        ljmp    scan_temp0
two_up05:  cjne    a,#2,three_up05
        mov     a,#'2'
        ljmp    scan_temp0
three_up05: cjne    a,#3,four_up05
        mov     a,#'3'
        ljmp    scan_temp0
four_up05: cjne    a,#4,five_up05
        mov     a,#'4'
        ljmp    scan_temp0
five_up05: cjne    a,#5,reset_up05
        mov     a,#'5'
        ljmp    scan_temp0
reset_up05: clr     a
        mov     temp0,a
        mov     a,#'0'
        ljmp    scan_temp0

```

```

;          dec temp05
;

```

```

down_temp05: dec    temp0
        mov     a,temp0
five_down05: cjne    a,#5,four_down05
        mov     a,#'5'
        ljmp    scan_temp0
four_down05: cjne    a,#4,three_down05
        mov     a,#'4'
        ljmp    scan_temp0
three_down05: cjne    a,#3,two_down05
        mov     a,#'3'
        ljmp    scan_temp0
two_down05:  cjne    a,#2,one_down05
        mov     a,#'2'
        ljmp    scan_temp0
one_down05:  cjne    a,#1,zero_down05
        mov     a,#'1'
        ljmp    scan_temp0
zero_down05: cjne    a,#0,set_down05
        mov     a,#'0'

```

```

        ljmp      scan_temp0
set_down05: mov    a,#5
        mov      temp0,a
        mov      a,#'5'
        ljmp      scan_temp0

```

```

;-----
;          show temp0 NO.
;-----

```

```

scan_temp0: lcall   lcd_wd
        mov      a,#0fh
        lcall    lcd_wi
        mov      a,#0c9h
        lcall    lcd_wi
        lcall    delay_100ms
        lcall    delay_100ms
        lcall    delay_100ms
        ljmp     state_3

```

```

;-----
;          inc temp1
;-----

```

```

up_temp1: inc      temp1
        mov      a,temp1
zero_up1: cjne     a,#0,one_up1
        mov      a,#'0'
        ljmp     scan_temp1
one_up1:  cjne     a,#1,two_up1
        mov      a,#'1'
        ljmp     scan_temp1
two_up1:  cjne     a,#2,three_up1
        mov      a,#'2'
        ljmp     scan_temp1
three_up1: cjne     a,#3,four_up1
        mov      a,#'3'
        ljmp     scan_temp1
four_up1: cjne     a,#4,five_up1
        mov      a,#'4'
        ljmp     scan_temp1
five_up1: cjne     a,#5,six_up1
        mov      a,#'5'
        ljmp     scan_temp1
six_up1:  cjne     a,#6,seven_up1
        mov      a,#'6'
        ljmp     scan_temp1
seven_up1: cjne     a,#7,eight_up1
        mov      a,#'7'
        ljmp     scan_temp1
eight_up1: cjne     a,#8,nine_up1
        mov      a,#'8'
        ljmp     scan_temp1
nine_up1: cjne     a,#9,reset_up1
        mov      a,#'9'
        ljmp     scan_temp1
reset_up1: clr      a
        mov      temp1,a
        mov      a,#'0'

```

```

        ljmp      scan_temp1
;-----
;               dec temp1
;-----
down_temp1: dec    temp1
            mov     a,temp1
nine_down1: cjne   a,#9,eight_down1
            mov     a,#'9'
            ljmp    scan_temp1
eight_down1: cjne  a,#8,seven_down1
            mov     a,#'8'
            ljmp    scan_temp1
seven_down1: cjne  a,#7,six_down1
            mov     a,#'7'
            ljmp    scan_temp1
six_down1:  cjne   a,#6,five_down1
            mov     a,#'6'
            ljmp    scan_temp1
five_down1: cjne   a,#5,four_down1
            mov     a,#'5'
            ljmp    scan_temp1
four_down1: cjne   a,#4,three_down1
            mov     a,#'4'
            ljmp    scan_temp1
three_down1: cjne  a,#3,two_down1
            mov     a,#'3'
            ljmp    scan_temp1
two_down1:  cjne   a,#2,one_down1
            mov     a,#'2'
            ljmp    scan_temp1
one_down1:  cjne   a,#1,zero_down1
            mov     a,#'1'
            ljmp    scan_temp1
zero_down1: cjne   a,#0,set_down1
            mov     a,#'0'
            ljmp    scan_temp1
set_down1:  mov     a,#9
            mov     temp1,a
            mov     a,#'9'
            ljmp    scan_temp1
;-----
;               show temp1 NO.
;-----
scan_temp1: lcall   lcd_wd
            mov     a,#0fh
            lcall   lcd_wi
            mov     a,#0 3h
            lcall   lcd_wi
            lcall   delay_100ms
            lcall   delay_100ms
            lcall   delay_100ms
            ljmp    state_2
;-----
;               inc temp2
;-----

```

```

up_temp2: inc    temp2
           mov    a,temp2
zero_up2: cjne   a,#0,one_up2
           mov    a,#'0'
           ljmp   scan_temp2
one_up2:  cjne   a,#1,two_up2
           mov    a,#'1'
           ljmp   scan_temp2
two_up2:  cjne   a,#2,three_up2
           mov    a,#'2'
           ljmp   scan_temp2
three_up2: cjne   a,#3,four_up2
           mov    a,#'3'
           ljmp   scan_temp2
four_up2: cjne   a,#4,five_up2
           mov    a,#'4'
           ljmp   scan_temp2
five_up2: cjne   a,#5,six_up2
           mov    a,#'5'
           ljmp   scan_temp2
six_up2:  cjne   a,#6,seven_up2
           mov    a,#'6'
           ljmp   scan_temp2
seven_up2: cjne   a,#7,eight_up2
           mov    a,#'7'
           ljmp   scan_temp2
eight_up2: cjne   a,#8,nine_up2
           mov    a,#'8'
           ljmp   scan_temp2
nine_up2: cjne   a,#9,reset_up2
           mov    a,#'9'
           ljmp   scan_temp2
reset_up2: clr    a
           mov    temp2,a
           mov    a,#'0'
           ljmp   scan_temp2

```

```

;-----
;               dec temp2
;-----

```

```

down_temp2: dec    temp2
           mov    a,temp2
nine_down2: cjne   a,#9,eight_down2
           mov    a,#'9'
           ljmp   scan_temp2
eight_down2: cjne   a,#8,seven_down2
           mov    a,#'8'
           ljmp   scan_temp2
seven_down2: cjne   a,#7,six_down2
           mov    a,#'7'
           ljmp   scan_temp2
six_down2:  cjne   a,#6,five_down2
           mov    a,#'6'
           ljmp   scan_temp2
five_down2: cjne   a,#5,four_down2
           mov    a,#'5'

```

```

        ljmp      scan_temp2
four_down2: cjne   a,#4,three_down2
        mov      a,#'4'
        ljmp      scan_temp2
three_down2: cjne  a,#3,two_down2
        mov      a,#'3'
        ljmp      scan_temp2
two_down2: cjne   a,#2,one_down2
        mov      a,#'2'
        ljmp      scan_temp2
one_down2: cjne   a,#1,zero_down2
        mov      a,#'1'
        ljmp      scan_temp2
zero_down2: cjne  a,#0,set_down2
        mov      a,#'0'
        ljmp      scan_temp2
set_down2: mov     a,#9
        mov      temp2,a
        mov      a,#'9'
        ljmp      scan_temp2

;-----
;          show temp2 NO.
;-----
scan_temp2: lcall   lcd_wd
        mov      a,#0fh
        lcall    lcd_wi
        mov      a,#0c7h
        lcall    lcd_wi
        lcall    delay_100ms
        lcall    delay_100ms
        lcall    delay_100ms
        ljmp     state_1

;-----
;          show_answer
;-----
keep_temp2: mov    a,temp2
        mov      target,a
        ljmp     end_value
keep_temp1: mov    a,temp2
        mov      b,#10
        mul      ab
        add      a,temp1
        mov      target,a
        ljmp     end_value
keep_temp0: mov    a,temp2
        mov      b,#100
        mul      ab
        mov      temp,a
        mov      a,temp1
        mov      b,#10
        mul      ab
        add      a,temp
        add      a,temp0
        mov      target,a
        ljmp     end_value

```

```

;-----
;               choice sub
;-----
up_choice: inc     choice
            mov     a,choice
            cjne    a,#2,next_m1
            mov     a,#2'
            lcall   show_choice
next_m1:    cjne    a,#1,next_m2
            mov     a,#1'
            lcall   show_choice
next_m2:    mov     a,#1
            mov     choice,a
            mov     a,#1'
            lcall   show_choice
down_choice: dec    choice
            mov     a,choice
            cjne    a,#1,next_m3
            mov     a,#1'
            lcall   show_choice
next_m3:    cjne    a,#2,next_m4
            mov     a,#2'
            lcall   show_choice
next_m4:    mov     a,#2
            mov     choice,a
            mov     a,#2'
            lcall   show_choice
show_choice: lcall   lcd_wd
            mov     a,#0fh
            lcall   lcd_wi
            mov     a,#0deh
            lcall   lcd_wi
            lcall   delay_100ms
            lcall   delay_100ms
            lcall   delay_100ms
            ljmp    menu
;-----
;               define constant
;-----
title_1:    db      "TELECOM. PROJECT"
title_2:    db      "[3T/2]LADKRABANG"
title_3:    db      "  MENU  "
title_4:    db      "1. Select Target"
title_5:    db      "2. Setup Robot "
title_6:    db      " Select Menu: "
title_7:    db      " Add Max_target "
title_8:    db      " First target? "
title_9:    db      " Second target? "
title_10:   db      " Third target? "
title_11:   db      " Maximum "
title_12:   db      " Target is "
title_13:   db      "Selection error!"
title_14:   db      " Don't select "
title_15:   db      " Over than "
title_16:   db      "Select complete!"

```

```

title_17: db      *   Go now!   *
title_18: db      "Before use robot"
title_19: db      " Setup it first "
title_20: db      * Over than 256. *

```

```

; blink lcd 3 time

```

```

blink_lcd: mov     a,#3
           mov     count4,a
blink:     mov     a,#08h
           lcall   lcd_wi
           lcall   delay_100ms
           lcall   delay_100ms
           lcall   delay_100ms
           mov     a,#0ch
           lcall   lcd_wi
           lcall   delay_100ms
           lcall   delay_100ms
           lcall   delay_100ms
           djnz    count4,blink
           ret

```

```

; run char. left side

```

```

char_run_l: lcall   wrchar_lcd
           lcall   delay_100ms
           mov     a,#00011000b
           lcall   lcd_wi
           lcall   delay_100ms
           ret

```

```

; dummy delay lcd

```

```

lcd_delay: mov     count3,#002
lcd_delay_1: mov   count2,#0e6h
lcd_delay_2: nop
           nop
           djnz    count2,lcd_delay_2
           djnz    count3,lcd_delay_1
           ret

```

```

delay_10ms: mov    count3,#010
delay_10ms_1: mov  count2,#0e6h
delay_10ms_2: nop
           nop
           djnz    count2,delay_10ms_2
           djnz    count3,delay_10ms_1
           ret

```

```

delay_100ms: mov   count3,#100
delay_100ms_1: mov count2,#0e6h
delay_100ms_2: nop
           nop
           djnz    count2,delay_100ms_2
           djnz    count3,delay_100ms_1
           ret

```

```

delay_1s:  mov     count1,#100

```

```

delay_1s_1: lcall    delay_10ms
             djnz     count1,delay_1s_1
             ret

```

```

;-----
;               write instruction
;-----

```

```

lcd_wi:      mov      dptr,#porta1
             movx     @dptr,a
             mov      dptr,#portb1
             clr      acc.0           ;clear enable
             clr      acc.1           ;clear rs
             movx     @dptr,a
             setb     acc.0           ;set enable
             movx     @dptr,a
             lcall    lcd_delay
             clr      acc.0           ;clear enable
             movx     @dptr,a
             lcall    lcd_delay
             ret

```

```

;-----
;               write data
;-----

```

```

lcd_wd:      mov      dptr,#porta1
             movx     @dptr,a
             mov      dptr,#portb1
             clr      acc.0           ;clear enable
             setb     acc.1           ;set rs
             movx     @dptr,a
             setb     acc.0           ;set enable
             movx     @dptr,a
             lcall    lcd_delay
             clr      acc.0           ;clear enable
             movx     @dptr,a
             lcall    lcd_delay
             ret

```

```

;-----
;               write characteristic
;-----

```

```

wrchar_lcd: lcall     lcd_wd
             mov      a,#00001100b
             lcall    lcd_wi
             ret

```

```

;-----
;               write line lcd
;-----

```

```

wrlne_lcd:  mov      count4,#16
wrlne_lcd_1: clr      a
             movc     a,@a+dptr
             push     dph
             push     dpl
             lcall    lcd_wd
             pop      dpl
             pop      dph
             inc      dptr
             djnz     count4,wrlne_lcd_1

```

```

        mov     a,#00001100b
        lcall   lcd_wi
        ret

#####
##### start 1 program #####
#####

select1: db     ' You must '
select2: db     ' Select Section '

;-----
start1:  mov     a,xs
        mov     dptr,#portc3
        movx    @dptr,a
        lcall   delay3
        lcall   delay3
        lcall   delay3
        mov     a,ys
        mov     dptr,#portc3
        movx    @dptr,a
        lcall   delay3
        lcall   delay3
        lcall   delay3
        mov     a,zs
        mov     dptr,#portc3
        movx    @dptr,a
        lcall   delay3
        lcall   delay3
        lcall   delay3
        mov     a,#00h
        mov     dptr,#portc3
        movx    @dptr,a
        lcall   delay3
        mov     ys1,#0ffh
        mov     zs1,#0ffh
        mov     a,xs
        cjne    a,#00h,gogogo
        mov     a,#01h        ;display for select zero section
        lcall   lcd_wi
        mov     a,#0c0h
        lcall   lcd_wi
        mov     dptr,#select1
        lcall   wrline_lcd
        mov     a,#090h
        lcall   lcd_wi
        mov     dptr,#select2
        lcall   wrline_lcd
        lcall   delay_1s
        lcall   delay_1s
        ljmp    main_menu      ;jump to main_menu
gogogo:  mov     a,ys
        cjne    a,#00h,gogogo1
        mov     ys1,#00h
gogogo1: mov     a,zs
        cjne    a,#00h,gogogo2
        mov     zs1,#00h
gogogo2: lcall   setw          ;set wheel

```

```

        lcall    retu_c            ;Return car
        mov     4fh,#00h
        mov     qs,#00h          ;count value of section
;----- timer0 setup -----
main3:   mov     ie,#10000010b    ;enable interrupt
        mov     tmod,#01h        ;timer0 mode1
        mov     th0,#00h         ;define timer0
        mov     tl0,#00h
        setb    tr0              ;start timer0
        mov     r6,#00h          ;int value timer
        ljmp    main

;#####
;##### program interrupt timer0 #####
;#####

ctime:   mov     49h,a
        push    dph
        push    dpl
        push    00h
        push    01h
        push    02h
        push    03h
        push    04h
        push    07h
        cjne    r6,#03,ctimex     ;interrupt 3 select 1
        lcall    sect_n
        mov     r6,#00h          ;sample value
ctime:   mov     th0,#00h         ;define timer0
        mov     tl0,#00h
        pop     07h
        pop     04h
        pop     03h
        pop     02h
        pop     01h
        pop     00h
        pop     dpl
        pop     dph
        mov     a,49h            ;@49h use for back up register A
        inc     r6
        reti

;#####
;##### main program #####
;#####

main:    ljmp     s_comp          ;this loop have interrupt timer0
main2:   mov     dptr,#porta2
        movx    a,@dptr
        mov     dptr,#portb3
        .ovx    @dptr,a
        mov     r7,a
        ljmp    con00

;#####
;##### con00 main #####
;#####

con00:   cjne    r7,#11110000b,con01 ;chk condition home to section
        lcall    ffw
        ljmp    main

```

con01:	cjne	r7,#11100000b,con02
	lcall	ffw
	ljmp	main
con02:	cjne	r7,#00001111b,con03
	lcall	ffw
	ljmp	main
con03:	cjne	r7,#00000111b,con04
	lcall	ffw
	ljmp	main
con04:	cjne	r7,#11000000b,con05
	lcall	ffw
	ljmp	main
con05:	cjne	r7,#00000011b,con06
	lcall	ffw
	ljmp	main
con06:	cjne	r7,#01111111b,con07
	mov	45h,#0dh
	lcall	lll
	ljmp	main
con07:	cjne	r7,#11111110b,con08
	mov	45h,#0dh
	lcall	rrr
	ljmp	main
con08:	cjne	r7,#10111111b,con09
	mov	45h,#0bh
	lcall	lll
	ljmp	main
con09:	cjne	r7,#11111101b,con10
	mov	45h,#0bh
	lcall	rrr
	ljmp	main
con10:	cjne	r7,#11011111b,con11
	mov	45h,#09h
	lcall	lll
	ljmp	main
con11:	cjne	r7,#11111011b,con12
	mov	45h,#09h
	lcall	rrr
	ljmp	main
con12:	cjne	r7,#11101111b,con13
	mov	45h,#05h
	lcall	lll
	ljmp	main
con13:	cjne	r7,#11110111b,con14
	mov	45h,#05h
	lcall	rrr
	ljmp	main
con14:	cjne	r7,#00111111b,con15
	mov	45h,#0dh
	lcall	lll
	ljmp	main
con15:	cjne	r7,#11111100b,con16
	mov	45h,#0dh
	lcall	rrr
	ljmp	main

```

con16:  cjne    r7,#10011111b,con17
        mov     45h,#0bh
        lcall   ll
        ljmp    main
con17:  cjne    r7,#11111001b,con18
        mov     45h,#0bh
        lcall   rrr
        ljmp    main
con18:  cjne    r7,#11001111b,con19
        mov     45h,#09h
        lcall   ll
        ljmp    main
con19:  cjne    r7,#11110011b,con20
        mov     45h,#09h
        lcall   rrr
        ljmp    main
con20:  cjne    r7,#00011111b,con21
        mov     45h,#0dh
        lcall   ll
        ljmp    main
con21:  cjne    r7,#11111000b,con22
        mov     45h,#0dh
        lcall   rrr
        ljmp    main
con22:  cjne    r7,#10001111b,con23
        mov     45h,#0dh
        lcall   ll
        ljmp    main
con23:  cjne    r7,#11110001b,con24
        mov     45h,#0dh
        lcall   rrr
        ljmp    main
con24:  cjne    r7,#11000111b,con25
        mov     45h,#07h
        lcall   ll
        ljmp    main
con25:  cjne    r7,#11100011b,con26
        mov     45h,#07h
        lcall   rrr
        ljmp    main
con26:  cjne    r7,#11100111b,con27
        lcall   ffw
        ljmp    main
con27:  cjne    r7,#11111111b,con28
        lcall   bbw
        ljmp    main
con28:  cjne    r7,#00h,con29
        :ljmp   retu_home;stop at station
con29:  ljmp    main
#####
;### this program use turn until to line  sconxx sub prog  ###
#####
sconxx:  mov     dptr,#porta2      ;check line
        movx    a,@dptr
        mov     dptr,#portb3

```

```

movx    @dptr,a
mov     r7,a
scon0:  cjne   r7,#07fh,scon1
        ljmp   sconkk
scon1:  cjne   r7,#0feh,scon2
        ljmp   sconkk
scon2:  cjne   r7,#0bfh,scon3
        ljmp   sconkk
scon3:  cjne   r7,#0fdh,scon4
        ljmp   sconkk
scon4:  cjne   r7,#0dfh,scon5
        ljmp   sconkk
scon5:  cjne   r7,#0fbh,scon6
        ljmp   sconkk
scon6:  cjne   r7,#0efh,scon7
        ljmp   sconkk
scon7:  cjne   r7,#0f7h,scon8
        ljmp   sconkk
scon8:  cjne   r7,#03fh,scon9
        ljmp   sconkk
scon9:  cjne   r7,#0fch,scon10
        ljmp   sconkk
scon10: cjne   r7,#09fh,scon11
        ljmp   sconkk
scon11: cjne   r7,#0f9h,scon12
        ljmp   sconkk
scon12: cjne   r7,#0cfh,scon13
        ljmp   sconkk
scon13: cjne   r7,#0f3h,scon14
        ljmp   sconkk
scon14: cjne   r7,#01fh,scon15
        ljmp   sconkk
scon15: cjne   r7,#0f8h,scon16
        ljmp   sconkk
scon16: cjne   r7,#08fh,scon17
        ljmp   sconkk
scon17: cjne   r7,#0f1h,scon18
        ljmp   sconkk
scon18: cjne   r7,#0c7h,scon19
        ljmp   sconkk
scon19: cjne   r7,#0e3h,scon20
        ljmp   sconkk
scon20: cjne   r7,#0e7h,scon21
        ljmp   sconkk
scon21: cjne   r7,#00h,scon22
        ljmp   sconkk
scon22: ljmp   sconxx
sconkk: ret

```

```

;#####
;##### run forward #####
;#####

```

```

ffw:    mov     dptr,#portc2
        mov     a,#00011101b    ;drive back in forward
        movx    @dptr,a
        lcall   delay2

```

```

        mov     dptr,#portc2
        mov     a,#00000000b      ;stop black wheel
        movx    @dptr,a
        ret

#####
##### run back #####
#####

bbw:    mov     dptr,#portc2
        mov     a,#00101110b      ;drive back
        movx    @dptr,a
        lcall   delay2
        lcall   delay2
        lcall   delay2
        lcall   delay2
        lcall   delay2
        lcall   delay2
        lcall   delay2
        lcall   delay2
        lcall   delay2
        lcall   sconxx
        mov     dptr,#portc2
        mov     a,#00h
        movx    @dptr,a
        ret

#####
##### turn left #####
#####

lll:    mov     r4,45h
        mov     dptr,#portc2
        mov     a,#00011101b
        movx    @dptr,a
        lcall   left
        lcall   sconxx
        mov     dptr,#portc1
        mov     a,#00h
        movx    @dptr,a
        mov     r4,45h
        lcall   righth
        mov     dptr,#portc1
        mov     a,#00h
        movx    @dptr,a
        mov     dptr,#portc2
        mov     a,#00000000b
        movx    @dptr,a
        ret

#####
##### turn righth #####
#####

rrr:    mov     r4,45h
        mov     dptr,#portc2
        mov     a,#00011101b
        movx    @dptr,a
        lcall   righth
        lcall   sconxx
        mov     dptr,#portc1

```

```

mov     a,#00h
movx    @dptr,a
mov     r4,45h
lcall   left
mov     dptr,#portc1
mov     a,#00h
movx    @dptr,a
mov     dptr,#portc2
mov     a,#00000000b
movx    @dptr,a
ret

```

```

#####
##### turn righth sub program #####
#####

```

```

righth:  mov     dptr,#portc1
         mov     a,#09h
         movx    @dptr,a
         lcall   delay2
         mov     dptr,#portc1
         mov     a,#0ch
         movx    @dptr,a
         lcall   delay2
         mov     dptr,#portc1
         mov     a,#06h
         movx    @dptr,a
         lcall   delay2
         mov     dptr,#portc1
         mov     a,#03h
         movx    @dptr,a
         lcall   delay2
         djnz    r4,righth
         ret

```

```

#####
##### turn left sub program #####
#####

```

```

left:    mov     dptr,#portc1
         mov     a,#03h
         movx    @dptr,a
         lcall   delay2
         mov     dptr,#portc1
         mov     a,#06h
         movx    @dptr,a
         lcall   delay2
         mov     dptr,#portc1
         mov     a,#0ch
         movx    @dptr,a
         lcall   delay2
         mov     dptr,#portc1
         mov     a,#09h
         movx    @dptr,a
         lcall   delay2
         djnz    r4,left
         ret

```

```
#####  
##### section left #####  
#####
```

```
sect:    mov     r4,#0eh  
         mov     dptr,#portc2  
         mov     a,#00011001b  
         movx    @dptr,a  
         lcall   s_left  
         mov     50h,#0fh  
         lcall   delay1  
         lcall   sconxx  
         mov     r4,#0eh  
         lcall   s_rigth  
         mov     dptr,#portc2  
         mov     a,#00000000b  
         movx    @dptr,a  
         ret
```

```
#####  
##### section rigth #####  
#####
```

```
secr:    mov     r4,#0eh  
         mov     dptr,#portc2  
         mov     a,#00010101b  
         movx    @dptr,a  
         lcall   s_rigth  
         mov     50h,#0fh  
         lcall   delay1  
         lcall   sconxx  
         mov     r4,#0eh  
         lcall   s_left  
         mov     dptr,#portc2  
         mov     a,#00000000b  
         movx    @dptr,a  
         ret
```

```
#####  
##### section rigth sub program #####  
#####
```

```
s_rigth: mov     dptr,#portc1  
         mov     a,#09h  
         movx    @dptr,a  
         lcall   delay2  
         mov     dptr,#portc1  
         mov     a,#0ch  
         movx    @dptr,a  
         lcall   delay2  
         mov     dptr,#portc1  
         mov     a,#06h  
         movx    @dptr,a  
         lcall   delay2  
         mov     dptr,#portc1  
         mov     a,#03h  
         movx    @dptr,a  
         lcall   delay2  
         djnz    r4,s_rigth  
         mov     dptr,#portc1
```

```

        mov     a,#00h
        movx    @dptr,a
        ret

#####
##### section left sub program #####
#####
s_left:  mov     dptr,#portc1
        mov     a,#03h
        movx    @dptr,a
        lcall   delay2
        mov     dptr,#portc1
        mov     a,#06h
        movx    @dptr,a
        lcall   delay2
        mov     dptr,#portc1
        mov     a,#0ch
        movx    @dptr,a
        lcall   delay2
        mov     dptr,#portc1
        mov     a,#09h
        movx    @dptr,a
        lcall   delay2
        djnz    r4,s_left
        mov     dptr,#portc1
        mov     a,#00h
        movx    @dptr,a
        ret

#####
##### check if find section inc 47h #####
#####
sect_n:  mov     dptr,#porta2      ;move value from sensor
        movx    a,@dptr
        mov     dptr,#portb3
        movx    @dptr,a
        cjne    a,#11000000b,sect_n1
        inc     qs
        mov     4fh,a
        ljmp    sect_e
sect_n1: cjne    a,#00000011b,sect_n2
        inc     qs
        mov     4fh,a
        ljmp    sect_e
sect_n2: cjne    a,#11100000b,sect_n3
        inc     qs
        mov     4fh,a
        ljmp    sect_e
sect_n3: cjne    a,#00000111b,sect_n4
        inc     qs
        mov     4fh,a
        ljmp    sect_e
sect_n4: cjne    a,#11110000b,sect_n5
        inc     qs
        mov     4fh,a
        ljmp    sect_e
sect_n5: cjne    a,#00001111b,sect_e

```

```

        inc        qs
        mov        4fh,a
sect_e:  mov        a,qs
        mov        dptr,#portb2
        movx       @dptr,a
        ret

;#####
;##### compare value if = 46h goto s_con0 #####
;#####

s_comp:  clr        tr0                ;stop timer0
        mov        a,qs
        cjne       a,xs,scomp1
        lcall      s_con0             ;chk righth or left section
        mov        xs,#00h           ; move value for max end
        mov        select_tray,#01h
        ljmp       sgoto              ;jump run go to end part
scomp1:  mov        a,ys1
        cjne       a,#0ffh,scomp2
        mov        a,qs
        cjne       a,ys,scomp2
        lcall      s_con0
        mov        ys,#0
        mov        select_tray,#02h
        ljmp       sgoto
scomp2:  mov        a,zs1
        cjne       a,#0ffh,scomp
        mov        a,qs
        cjne       a,zs,scomp
        lcall      s_con0
        mov        zs,#0
        mov        select_tray,#03h
        ljmp       sgoto
scomp:   setb       tr0                ;start timero
        ljmp       main2              ;qs not equ to xs,ys,zs jmp to main2

;#####
;##### check value of section is righth or left #####
;#####

s_con0:  mov        a,4fh              ;@4fh is condition value from interrupt timer0
        cjne       a,#11110000b,s_con1
        lcall      secr
        mov        4ah,#0ffh         ;if @4ah = 0ffh present to turn righth
        ljmp       s_conxx
s_con1:  cjne       a,#00001111b,s_con2
        lcall      secl
        mov        4ah,#00h          ;if @4ah = 000h present to turn left
        ljmp       s_conxx
s_con2:  cjne       a,#11100000b,s_con3
        lcall      secr
        mov        4ah,#0ffh
        ljmp       s_conxx
s_con3:  cjne       a,#00000111b,s_con4
        lcall      secl
        mov        4ah,#00h
        ljmp       s_conxx
s_con4:  cjne       a,#11000000b,s_con5

```

```

        lcall    secr
        mov     4ah,#0fth
        ljmp    s_conxx
s_con5:  cjne    a,#00000011b,s_conxx
        lcall    secl
        mov     4ah,#00h
s_conxx:  ret                                ;return to s_comp
;#####
;##### sgoto this sub-program runto end #####
;#####
sgoto:    mov     dptr,#porta2
        movx    a,@dptr                    ; move condition from sensor
        mov     dptr,#portb3
        movx    @dptr,a
        mov     r7,a
        ljmp    sgoto00                    ;sgotoxx condition
;#####
;##### sgoto subprogram #####
;#####
sgoto00:  cjne    r7,#11110000b,sgoto01
        lcall    ffw
        ljmp    sgoto
sgoto01:  cjne    r7,#11100000b,sgoto02
        lcall    ffw
        ljmp    sgoto
sgoto02:  cjne    r7,#00001111b,sgoto03
        lcall    ffw
        ljmp    sgoto
sgoto03:  cjne    r7,#00000111b,sgoto04
        lcall    ffw
        ljmp    sgoto
sgoto04:  cjne    r7,#11000000b,sgoto05
        lcall    ffw
        ljmp    sgoto
sgoto05:  cjne    r7,#00000011b,sgoto06
        lcall    ffw
        ljmp    sgoto
sgoto06:  cjne    r7,#01111111b,sgoto07
        mov     45h,#0dh
        lcall    lll
        ljmp    sgoto
sgoto07:  cjne    r7,#11111110b,sgoto08
        mov     45h,#0dh
        lcall    rrr
        ljmp    sgoto
sgoto08:  cjne    r7,#10111111b,sgoto09
        mov     45h,#0bh
        lcall    lll
        ljmp    sgoto
sgoto09:  cjne    r7,#11111101b,sgoto10
        mov     45h,#0bh
        lcall    rrr
        ljmp    sgoto
sgoto10:  cjne    r7,#11011111b,sgoto11
        mov     45h,#09h

```

	lcall	lll
	ljmp	sgoto
sgoto11:	cjne	r7,#11111011b,sgoto12
	mov	45h,#09h
	lcall	rrr
	ljmp	sgoto
sgoto12:	cjne	r7,#11101111b,sgoto13
	mov	45h,#07h
	lcall	lll
	ljmp	sgoto
sgoto13:	cjne	r7,#11110111b,sgoto14
	mov	45h,#07h
	lcall	rrr
	ljmp	sgoto
sgoto14:	cjne	r7,#00111111b,sgoto15
	mov	45h,#0dh
	lcall	lll
	ljmp	sgoto
sgoto15:	cjne	r7,#11111100b,sgoto16
	mov	45h,#0dh
	lcall	rrr
	ljmp	sgoto
sgoto16:	cjne	r7,#10011111b,sgoto17
	mov	45h,#0bh
	lcall	lll
	ljmp	sgoto
sgoto17:	cjne	r7,#11111001b,sgoto18
	mov	45h,#0bh
	lcall	rrr
	ljmp	sgoto
sgoto18:	cjne	r7,#11001111b,sgoto19
	mov	45h,#09h
	lcall	lll
	ljmp	sgoto
sgoto19:	cjne	r7,#11110011b,sgoto20
	mov	45h,#09h
	lcall	rrr
	ljmp	sgoto
sgoto20:	cjne	r7,#00011111b,sgoto21
	mov	45h,#0dh
	lcall	lll
	ljmp	sgoto
sgoto21:	cjne	r7,#11111000b,sgoto22
	mov	45h,#0dh
	lcall	rrr
	ljmp	sgoto
sgoto22:	cjne	r7,#10001111b,sgoto23
	mov	45h,#0dh
	lcall	lll
	ljmp	sgoto
sgoto23:	cjne	r7,#11110001b,sgoto24
	mov	45h,#0dh
	lcall	rrr
	ljmp	sgoto
sgoto24:	cjne	r7,#11000111b,sgoto25

```

        mov     45h,#07h
        lcall   lll
        ljmp    sgoto
sgoto25: cjne   r7,#11100011b,sgoto26
        mov     45h,#07h
        lcall   rrr
        ljmp    sgoto
sgoto26: cjne   r7,#11100111b,sgoto27
        lcall   ffw
        ljmp    sgoto
sgoto27: cjne   r7,#11111111b,sgoto28
        lcall   bbw
        ljmp    sgoto
sgoto28: cjne   r7,#00h,sgoto29
        ljmp    retu_home      ;stop at station
sgoto29: ljmp    sgoto        ;return loop sgoto
;#####
;##### set wheel #####
;#####
setw:   mov     dptr,#port2      ;set wheel center
        mov     a,#0000000b
        movx    @dptr,a
yyy:    mov     dptr,#portb3
        mov     a,#0ffh
        movx    @dptr,a
        mov     50h,#01h
        lcall   delay1
        mov     dptr,#portb3
        mov     a,#00h
        movx    @dptr,a
        mov     50h,#01h
        lcall   delay1
        mov     r4,#20
        lcall   setw0
        mov     r4,#16
        lcall   setw1
        mov     r4,#17
        lcall   setw1
        mov     r4,#16
        lcall   setw0
        ret
setw0:  mov     dptr,#portc1
        mov     a,#03h
        movx    @dptr,a
        lcall   delay3
        mov     dptr,#portc1
        mov     a,#06h
        movx    @dptr,a
        lcall   delay3
        mov     dptr,#portc1
        mov     a,#0ch
        movx    @dptr,a
        lcall   delay3
        mov     dptr,#portc1
        mov     a,#09h

```

```

movx    @dptr,a
lcall   delay3
djnz    r4,setw0
mov     dptr,#portc1
mov     a,#00h
movx    @dptr,a
ret

setw1:  mov     dptr,#portc1
        mov     a,#09h
        movx    @dptr,a
        lcall   delay3
        mov     dptr,#portc1
        mov     a,#0ch
        movx    @dptr,a
        lcall   delay3
        mov     dptr,#portc1
        mov     a,#06h
        movx    @dptr,a
        lcall   delay3
        mov     dptr,#portc1
        mov     a,#03h
        movx    @dptr,a
        lcall   delay3
        djnz    r4,setw1
        mov     dptr,#portc1
        mov     a,#00h
        movx    @dptr,a
        ret

#####
##### delay #####
#####
delay1:  mov     51h,#00h      ;if use this program you must int @50h
ddd:    mov     52h,#00h
        djnz    52h,$
        djnz    51h,ddd
        djnz    50h,delay1
        ret

delay2:  mov     53h,#01h
ddd2:    mov     54h,#0dh
dd2:     mov     55h,#0ffh
        djnz    55h,$
        djnz    54h,dd2
        djnz    53h,ddd2
        ret

delay3:  mov     53h,#03h
delay3_2: mov     54h,#0dh
delay3_1: mov     55h,#0ffh
        djnz    55h,$
        djnz    54h,delay3_1
        djnz    53h,delay3_2
        ret

#####
##### return car #####
#####
retu_c:  mov     dptr,#portc2      ;bbw l=r \

```

```

mov     a,#00100110b
movx    @dptr,a
mov     r4,#16           ;righ\
lcall   righ
mov     50h,#06h         ;delay\
lcall   delay1
lcall   sconxx           ;check line
mov     50h,#02h         ;delay\
lcall   delay1
mov     dptr,#portc2     ;stop\
mov     a,#00h
movx    @dptr,a
mov     50h,#09h         ;deley\
lcall   delay1
mov     r4,#16           ;left\
lcall   left
mov     a,#00011001b     ;ffw l/=r
mov     dptr,#portc2
movx    @dptr,a
mov     r4,#16           ;left\
lcall   left
mov     50h,#10h         ;delay\
lcall   delay1
mov     r4,#16           ;righ\
lcall   righ
mov     dptr,#portc2     ;stop\
mov     a,#00h
movx    @dptr,a
mov     50h,#09h         ;delay
lcall   delay1
mov     a,#00101110b     ;bbw
mov     dptr,#portc2
movx    @dptr,a
mov     50h,#20h
lcall   delay1
mov     a,#00011101b
mov     dptr,#portc2
movx    @dptr,a
lcall   sconxx
mov     dptr,#portc2
mov     a,#00h
movx    @dptr,a
mov     a,#00h
mov     dptr,#portc1
movx    @dptr,a
lcall   setw
mov     dptr,#portc2
mov     a,#00011101b     ;drive back in forward
movx    @dptr,a
lcall   sconxx
mov     dptr,#portc2
mov     a,#00000000b     ;stop black wheel
movx    @dptr,a
ret

```

```

;#####
;##### return to home #####
;#####

data00:  db      ' Please! '
data01:  db      ' receive '
data02:  db      ' message '
data03:  db      ' NOW '
data04:  db      ' Return to '
data05:  db      ' start '
data06:  db      ' Station '
data07:  db      ' NOW '
data08:  db      ' GO to '
data09:  db      ' Next! '
data10:  db      ' Station '
data11:  db      ' NOW '

retu_home:  clr      tr0
            mov      a,#01h      ;show lcd receive message
            lcall    lcd_wi
            mov      a,#010h
            lcall    lcd_wi
            mov      dptr,#data00
            lcall    wrline_lcd
            mov      a,#0c0h
            lcall    lcd_wi
            mov      dptr,#data01
            lcall    wrline_lcd
            mov      a,#090h
            lcall    lcd_wi
            mov      dptr,#data02
            lcall    wrline_lcd
            mov      a,#0d0h
            lcall    lcd_wi
            mov      dptr,#data03
            lcall    wrline_lcd
            lcall    tray

;#####
;##### lcd present ' goto home ' #####
;#####

            mov      a,xs
            cjne     a,#0,nnww1
            mov      a,ys
            cjne     a,#0,nnww1
            mov      a,zs
            cjne     a,#0,nnww1
            mov      a,#01h      ; show lcd goto start
            lcall    lcd_wi
            mov      a,#010h
            lcall    lcd_wi
            mov      dptr,#data04
            lcall    wrline_lcd
            mov      a,#0c0h
            lcall    lcd_wi
            mov      dptr,#data05
            lcall    wrline_lcd
            mov      a,#090h

```

```

lcall    lcd_wi
mov      dptr,#data06
lcall    wrline_lcd
mov      a,#0d0h
lcall    lcd_wi
mov      dptr,#data07
lcall    wrline_lcd
ljmp     nnww2
nnww1:   mov      a,#01h          ;show lcd goto next station
lcall    lcd_wi
mov      a,#010h
lcall    lcd_wi
mov      dptr,#data08
lcall    wrline_lcd
mov      a,#0c0h
lcall    lcd_wi
mov      dptr,#data09
lcall    wrline_lcd
mov      a,#090h
lcall    lcd_wi
mov      dptr,#data10
lcall    wrline_lcd
mov      a,#0d0h
lcall    lcd_wi
mov      dptr,#data11
lcall    wrline_lcd
nnww2:   lcall     retu_c          ;return car
;##### timer1 setup #####
mov      ie,#10001000b
mov      tmod,#10h ;timer1 mode1
mov      th1,#00h
mov      tl1,#00h
setb     tr1
mov      r6,#00h
ljmp     ghome
;#####
;##### program interrupt timer1 #####
;#####
rtime:    mov      49h,a
push     dph
push     dpl
push     00h
push     01h
push     02h
push     03h
push     04h
push     07h
cjne     r6,#02h,rtimex
lcall    gchk
mov      r6,#00h
rtimex:   mov      th1,#00h
mov      tl1,#00h
pop       07h
pop       04h
pop       03h

```

```

        pop        02h
        pop        01h
        pop        00h
        pop        dpl
        pop        dph
        mov        a,49h
        inc        r6
        reti

;#####
;##### check T-section #####
;#####

gchk:    mov        dptr,#porta2
        movx       a,@dptr
        cjne       a,#00h,gchk_out
        mov        44h,#0ffh
        mov        dptr,#portc3
        mov        a,#11000011b
        movx       @dptr,a
gchk_out: ret
;#####
;##### compare value T-section #####
;#####

gcomp:   clr        tr1
        mov        a,44h
        cjne       a,#0ffh,gcomp_out
        ljmp       gcomp_con
gcomp_out: setb     tr1
        ljmp       ghome1
;#####
;##### check to turn left or right #####
;##### when find T-section #####
;#####

gcomp_con: mov     a,xs
        cjne       a,#00h,t_sec
        mov        a,ys
        cjne       a,#00h,t_sec
        mov        a,zs
        cjne       a,#00h,t_sec
        mov        a,4ah
        cjne       a,#0ffh,gcomp_con1
        lcall      sec1
        ljmp       goto_home
gcomp_con1: lcall   secr
        ljmp       goto_home
t_sec:    mov        a,4ah
        cjne       a,#0ffh,t_sec1
        lcall      secr
        mov        44h,#00h
        ljmp       main3
t_sec1:   lcall      sec1
        mov        44h,#00h
        ljmp       main3

```

```

#####
##### go to home station #####
##### run by don't care section #####
#####

```

```

goto_home: mov     dptr,#porta2      ;recive condition from sensor
            movx    a,@dptr
            mov     dptr,#portb3
            movx    @dptr,a
            mov     r7,a
            ljmp    goto_con00

```

```

;----- check condition -----

```

```

goto_con00: cjne    r7,#11110000b,goto_con01
            lcall    ffw
            ljmp    goto_home

```

```

goto_con01: cjne    r7,#11100000b,goto_con02
            lcall    ffw
            ljmp    goto_home

```

```

goto_con02: cjne    r7,#00001111b,goto_con03
            lcall    ffw
            ljmp    goto_home

```

```

goto_con03: cjne    r7,#00000111b,goto_con04
            lcall    ffw
            ljmp    goto_home

```

```

goto_con04: cjne    r7,#11000000b,goto_con05
            lcall    ffw
            ljmp    goto_home

```

```

goto_con05: cjne    r7,#00000011b,goto_con06
            lcall    ffw
            ljmp    goto_home

```

```

goto_con06: cjne    r7,#01111111b,goto_con07
            mov     45h,#0dh
            lcall    lll
            ljmp    goto_home

```

```

goto_con07: cjne    r7,#11111110b,goto_con08
            mov     45h,#0dh
            lcall    rrr
            ljmp    goto_home

```

```

goto_con08: cjne    r7,#10111111b,goto_con09
            mov     45h,#0bh
            lcall    lll
            ljmp    goto_home

```

```

goto_con09: cjne    r7,#111111101b,goto_con10
            mov     45h,#0bh
            lcall    rrr
            ljmp    goto_home

```

```

goto_con10: cjne    r7,#11011111b,goto_con11
            mov     45h,#09h
            lcall    lll
            ljmp    goto_home

```

```

goto_con11: cjne    r7,#11111011b,goto_con12
            mov     45h,#09h
            lcall    rrr
            ljmp    goto_home

```

```

goto_con12: cjne    r7,#11101111b,goto_con13
            mov     45h,#05h

```

```

lcall    lll
ljmp     goto_home
goto_con13: cjne    r7,#11110111b,goto_con14
            mov     45h,#05h
            lcall    rrr
            ljmp     goto_home
goto_con14: cjne    r7,#00111111b,goto_con15
            mov     45h,#0dh
            lcall    lll
            ljmp     goto_home
goto_con15: cjne    r7,#11111100b,goto_con16
            mov     45h,#0dh
            lcall    rrr
            ljmp     goto_home
goto_con16: cjne    r7,#10011111b,goto_con17
            mov     45h,#0bh
            lcall    lll
            ljmp     goto_home
goto_con17: cjne    r7,#11111001b,goto_con18
            mov     45h,#0bh
            lcall    rrr
            ljmp     goto_home
goto_con18: cjne    r7,#11001111b,goto_con19
            mov     45h,#09h
            lcall    lll
            ljmp     goto_home
goto_con19: cjne    r7,#11110011b,goto_con20
            mov     45h,#09h
            lcall    rrr
            ljmp     goto_home
goto_con20: cjne    r7,#00011111b,goto_con21
            mov     45h,#0dh
            lcall    lll
            ljmp     goto_home
goto_con21: cjne    r7,#11111000b,goto_con22
            mov     45h,#0dh
            lcall    rrr
            ljmp     goto_home
goto_con22: cjne    r7,#10001111b,goto_con23
            mov     45h,#0dh
            lcall    lll
            ljmp     goto_home
goto_con23: cjne    r7,#11110001b,goto_con24
            mov     45h,#0dh
            lcall    rrr
            ljmp     goto_home
goto_con24: cjne    r7,#11000111b,goto_con25
            mov     45h,#07h
            lcall    lll
            ljmp     goto_home
goto_con25: cjne    r7,#11100011b,goto_con26
            mov     45h,#07h
            lcall    rrr
            ljmp     goto_home
goto_con26: cjne    r7,#11100111b,goto_con27

```

```

        lcall    ffw
        ljmp     goto_home
goto_con27: cjne    r7,#11111111b,goto_con28
        lcall    bbw
        ljmp     goto_home
goto_con28: cjne    r7,#00h,goto_con29
        ljmp     main_menu          ;go to start program end of runprogram
goto_con29: ljmp     goto_home
#####
##### run out from target to T-section #####
#####
ghome:    ljmp     gcomp              ;check T-section from interrupt timer1
ghome1:   mov      dptr,#porta2        ;recive condition from sensor
        movx     a,@dptr
        mov      dptr,#portb3
        movx     @dptr,a
        mov      r7,a
        ljmp     gcon00
;----- check condition from sensor -----
gcon00:   cjne     r7,#11110000b,gcon01
        lcall    ffw
        ljmp     ghome
gcon01:   cjne     r7,#11100000b,gcon02
        lcall    ffw
        ljmp     ghome
gcon02:   cjne     r7,#00001111b,gcon03
        lcall    ffw
        ljmp     ghome
gcon03:   cjne     r7,#00000111b,gcon04
        lcall    ffw
        ljmp     ghome
gcon04:   cjne     r7,#11000000b,gcon05
        lcall    ffw
        ljmp     ghome
gcon05:   cjne     r7,#00000011b,gcon06
        lcall    ffw
        ljmp     ghome
gcon06:   cjne     r7,#01111111b,gcon07
        mov      45h,#0dh
        lcall    lll
        ljmp     ghome
gcon07:   cjne     r7,#11111110b,gcon08
        mov      45h,#0dh
        lcall    rrr
        ljmp     ghome
gcon08:   cjne     r7,#10111111b,gcon09
        mov      45h,#0bh
        lcall    lll
        ljmp     ghome
gcon09:   cjne     r7,#11111101b,gcon10
        mov      45h,#0bh
        lcall    rrr
        ljmp     ghome
gcon10:   cjne     r7,#11011111b,gcon11
        mov      45h,#09h

```

	lcall	lll
	ljmp	ghome
gcon11:	cjne	r7,#11111011b,gcon12
	mov	45h,#09h
	lcall	rrr
	ljmp	ghome
gcon12:	cjne	r7,#11101111b,gcon13
	mov	45h,#05h
	lcall	lll
	ljmp	ghome
gcon13:	cjne	r7,#11110111b,gcon14
	mov	45h,#05h
	lcall	rrr
	ljmp	ghome
gcon14:	cjne	r7,#00111111b,gcon15
	mov	45h,#0dh
	lcall	lll
	ljmp	ghome
gcon15:	cjne	r7,#11111100b,gcon16
	mov	45h,#0dh
	lcall	rrr
	ljmp	ghome
gcon16:	cjne	r7,#10011111b,gcon17
	mov	45h,#0bh
	lcall	lll
	ljmp	ghome
gcon17:	cjne	r7,#11111001b,gcon18
	mov	45h,#0bh
	lcall	rrr
	ljmp	ghome
gcon18:	cjne	r7,#11001111b,gcon19
	mov	45h,#09h
	lcall	lll
	ljmp	ghome
gcon19:	cjne	r7,#11110011b,gcon20
	mov	45h,#09h
	lcall	rrr
	ljmp	ghome
gcon20:	cjne	r7,#00011111b,gcon21
	mov	45h,#0dh
	lcall	lll
	ljmp	ghome
gcon21:	cjne	r7,#11111000b,gcon22
	mov	45h,#0dh
	lcall	rrr
	ljmp	ghome
gcon22:	cjne	r7,#10001111b,gcon23
	mov	45h,#0dh
	lcall	lll
	ljmp	ghome
gcon23:	cjne	r7,#11110001b,gcon24
	mov	45h,#0dh
	lcall	rrr
	ljmp	ghome
gcon24:	cjne	r7,#11000111b,gcon25

```

mov     45h,#07h
lcall   lll
ljmp    ghome
gcon25: cjne   r7,#11100011b,gcon26
mov     45h,#07h
lcall   rrr
ljmp    ghome
gcon26: cjne   r7,#11100111b,gcon27
lcall   ffw
ljmp    ghome
gcon27: cjne   r7,#11111111b,gcon28
lcall   bbw
ljmp    ghome
gcon28: cjne   r7,#00h,gcon29
lcall   ffw
gcon29: ljmp    ghome
tray:   mov     a,select_tray
tray1:  cjne   a,#01h,tray2
mov     a,4ah
cjne   a,#0ffh,tray11
mov     dptr,#porta3
mov     a,#00100000b
movx    @dptr,a
lcall   delaytray
mov     dptr,#porta3
mov     a,#0
movx    @dptr,a
ljmp    trayex
tray11: mov     dptr,#porta3
mov     a,#00010000b
movx    @dptr,a
lcall   delaytray
mov     dptr,#porta3
mov     a,#0
movx    @dptr,a
ljmp    trayex
tray2:  cjne   a,#02h,tray3
mov     a,4ah
cjne   a,#0ffh,tray22
mov     dptr,#porta3
mov     a,#00001000b
movx    @dptr,a
lcall   delaytray
mov     dptr,#porta3
mov     a,#0
movx    @dptr,a
ljmp    trayex
tray22: mov     dptr,#porta3
mov     a,#00000100b
movx    @dptr,a
lcall   delaytray
mov     dptr,#porta3
mov     a,#0
movx    @dptr,a
ljmp    trayex

```

```

tray3:    mov     a,4ah
          cjne    a,#0ffh,tray33
          mov     dptr,#porta3
          mov     a,#00000010b
          movx    @dptr,a
          lcall   delaytray
          mov     dptr,#porta3
          mov     a,#0
          movx    @dptr,a
          ljmp    trayex
tray33:   mov     dptr,#porta3
          mov     a,#00000001b
          movx    @dptr,a
          lcall   delaytray
          mov     dptr,#porta3
          mov     a,#0
          movx    @dptr,a
trayex:   ret
delaytray: mov     73h,#02fh
delaytray2: mov    74h,#0ffh
delaytray1: mov    75h,#0ffh
          djnz    75h,$
          djnz    74h,delaytray1
          djnz    73h,delaytray2
          ret
          end

```

```

;#####
;##### end of program #####
;#####

```

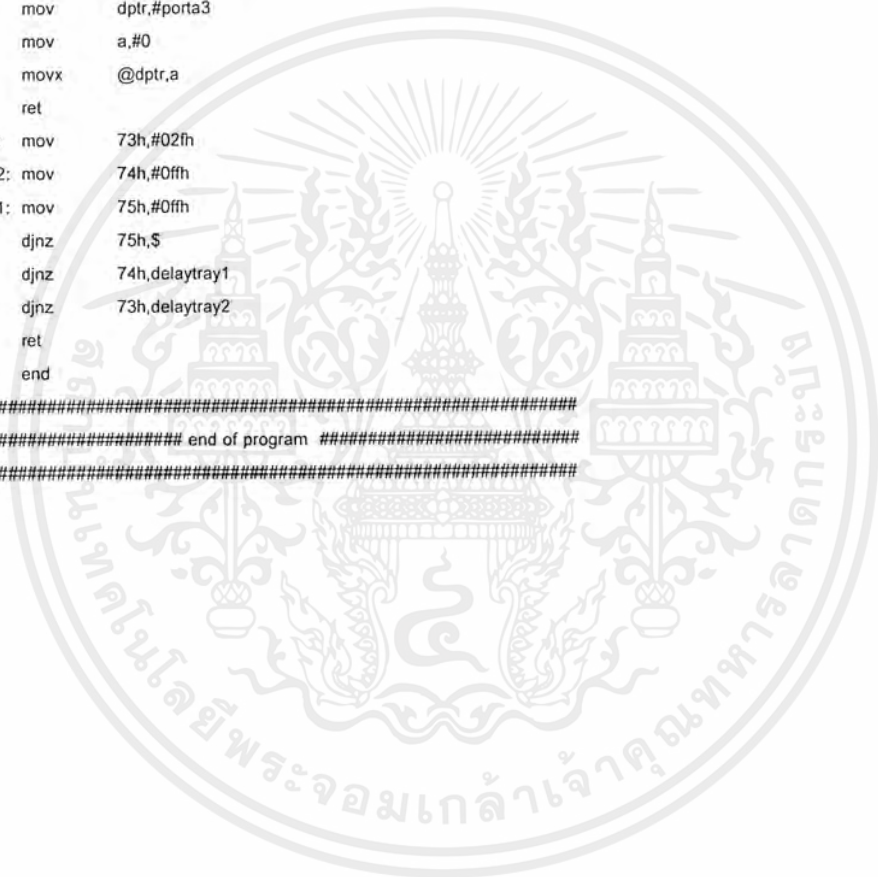


Photo Modules for PCM Remote Control Systems

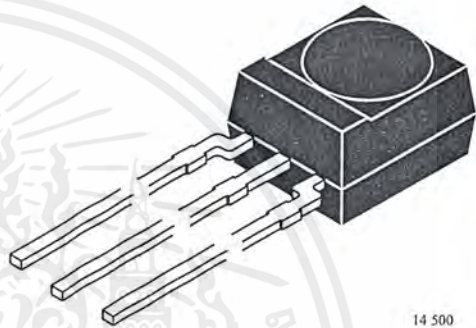
Available types for different carrier frequencies

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

Description

The TSOP48.. – series are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter.

The demodulated output signal can directly be decoded by a microprocessor. TSOP48.. is the standard IR remote control receiver series, supporting all major transmission codes.

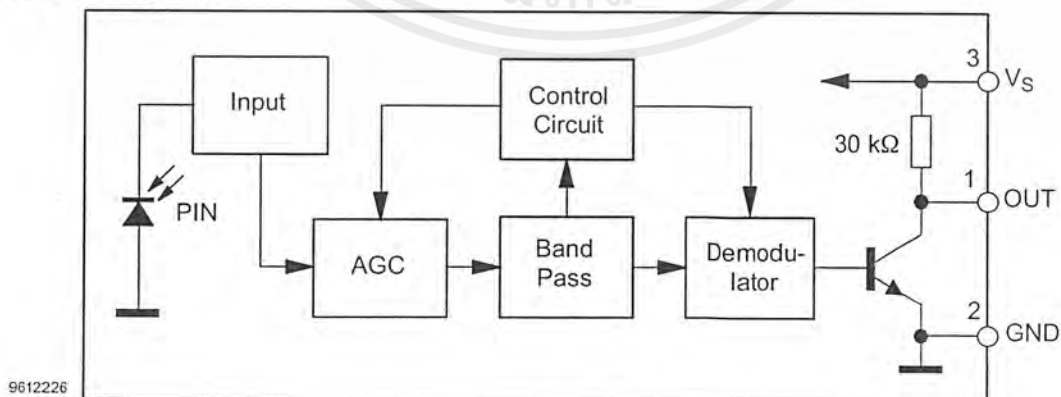


14 500

Features

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

Block Diagram



Absolute Maximum Ratings

T_{amb} = 25°C

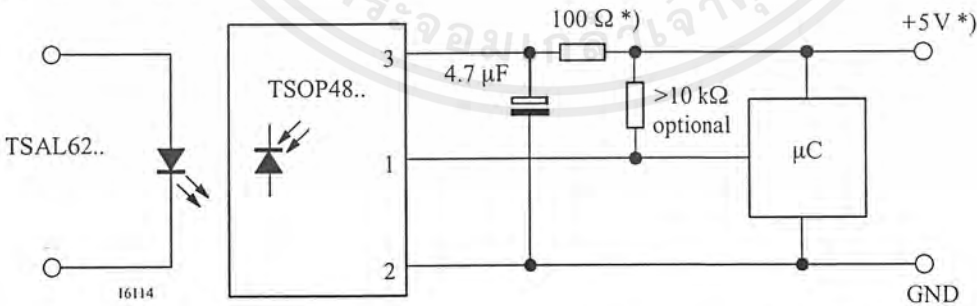
Parameter	Test Conditions	Symbol	Value	Unit
Supply Voltage	(Pin 3)	V _S	−0.3...6.0	V
Supply Current	(Pin 3)	I _S	5	mA
Output Voltage	(Pin 1)	V _O	−0.3...6.0	V
Output Current	(Pin 1)	I _O	5	mA
Junction Temperature		T _j	100	°C
Storage Temperature Range		T _{stg}	−25...+85	°C
Operating Temperature Range		T _{amb}	−25...+85	°C
Power Consumption	(T _{amb} ≤ 85 °C)	P _{tot}	50	mW
Soldering Temperature	t ≤ 10 s, 1 mm from case	T _{sd}	260	°C

Basic Characteristics

T_{amb} = 25°C

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Supply Current (Pin 3)	V _S = 5 V, E _v = 0	I _{SD}	0.8	1.1	1.5	mA
	V _S = 5 V, E _v = 40 klx, sunlight	I _{SH}		1.4		mA
Supply Voltage (Pin 3)		V _S	4.5		5.5	V
Transmission Distance	E _v = 0, test signal see fig.7, IR diode TSAL6200, I _F = 250 mA	d		35		m
Output Voltage Low (Pin 1)	I _{OSL} = 0.5 mA, E _e = 0.7 mW/m ²	V _{OSL}			250	mV
Irradiance (30 – 40 kHz)	Pulse width tolerance: t _{pi} – 5/f _o < t _{po} < t _{pi} + 6/f _o , test signal see fig.7	E _{e min}		0.2	0.4	mW/m ²
Irradiance (56 kHz)	Pulse width tolerance: t _{pi} – 5/f _o < t _{po} < t _{pi} + 6/f _o , test signal see fig.7	E _{e min}		0.3	0.6	mW/m ²
Irradiance	t _{pi} – 5/f _o < t _{po} < t _{pi} + 6/f _o	E _{e max}	30			W/m ²
Directivity	Angle of half transmission distance	φ _{1/2}		±45		deg

Application Circuit



*) recommended to suppress power supply disturbances

Suitable Data Format

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

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

The data signal should fulfill the following condition:

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

Some examples for suitable data format are:

NEC Code, Toshiba Micom Format, Sharp Code, RC5 Code, RC6 Code, R-2000 Code.

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

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

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



Figure A: IR Signal from Fluorescent Lamp with low Modulation

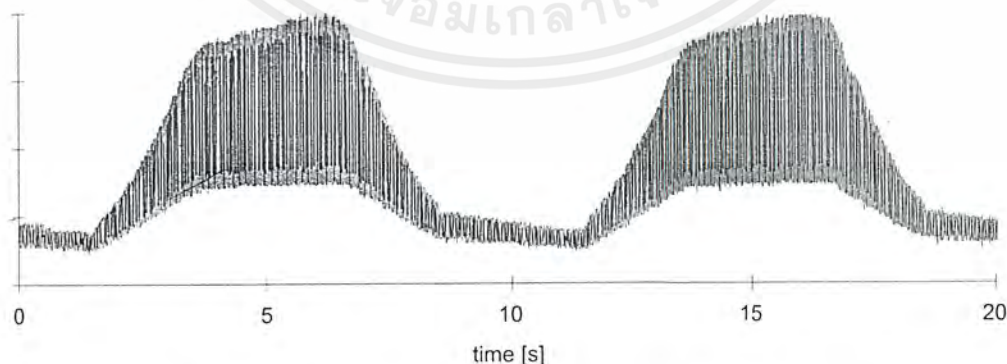


Figure B: IR Signal from Fluorescent Lamp with high Modulation



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

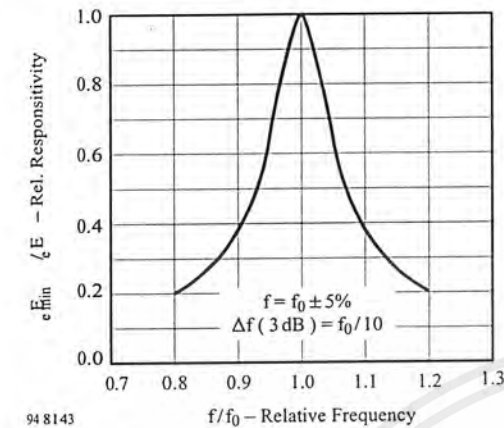


Figure 1. Frequency Dependence of Responsivity

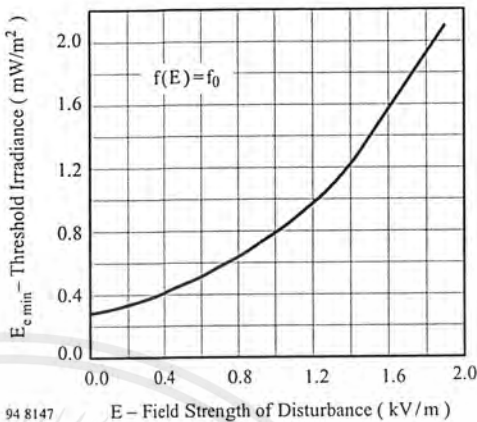


Figure 4. Sensitivity vs. Electric Field Disturbances

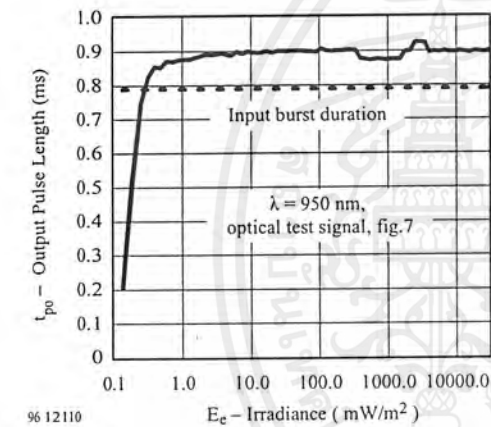


Figure 2. Sensitivity in Dark Ambient

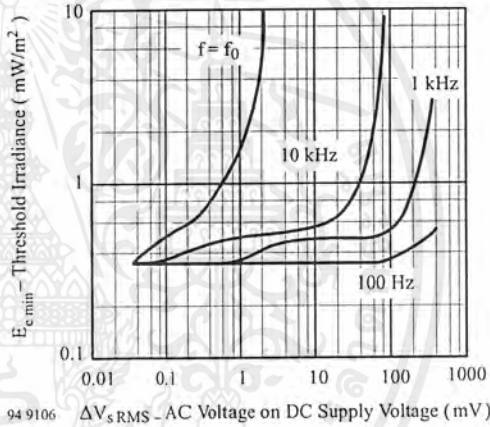


Figure 5. Sensitivity vs. Supply Voltage Disturbances

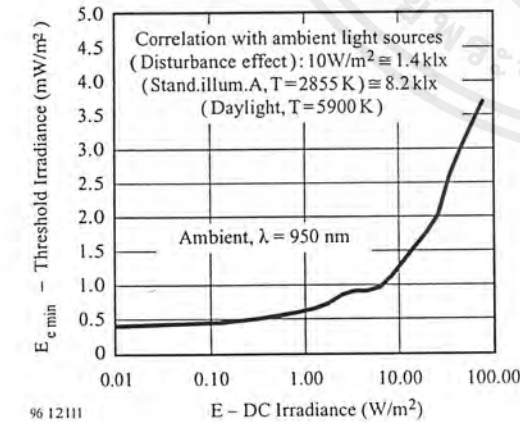


Figure 3. Sensitivity in Bright Ambient

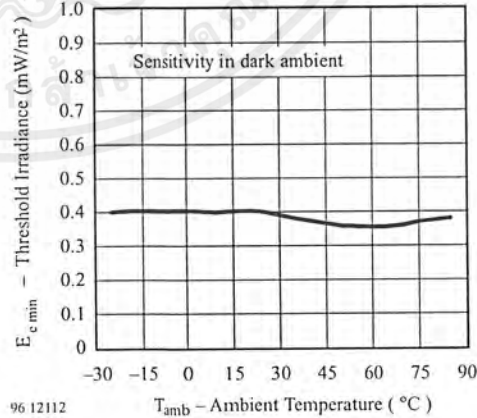


Figure 6. Sensitivity vs. Ambient Temperature

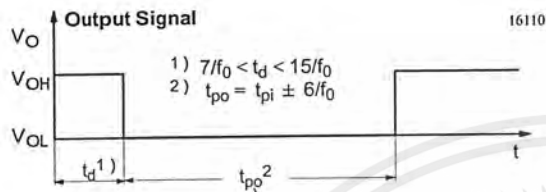
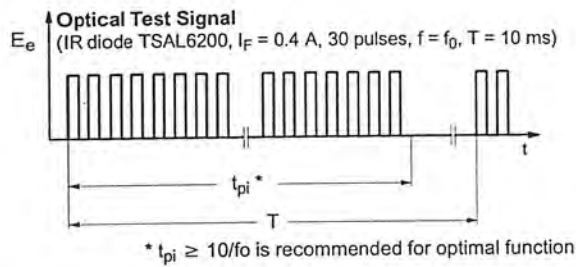


Figure 7.

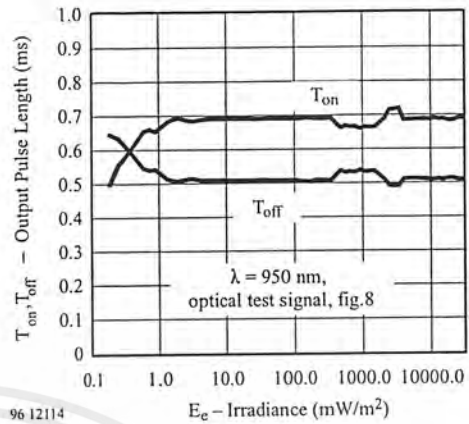


Figure 10. Output Pulse Diagram

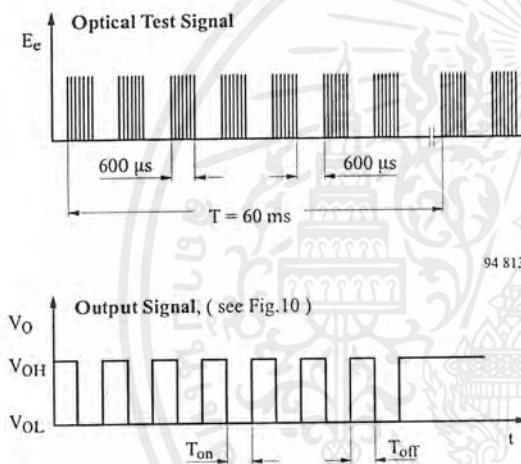


Figure 8. Output Function

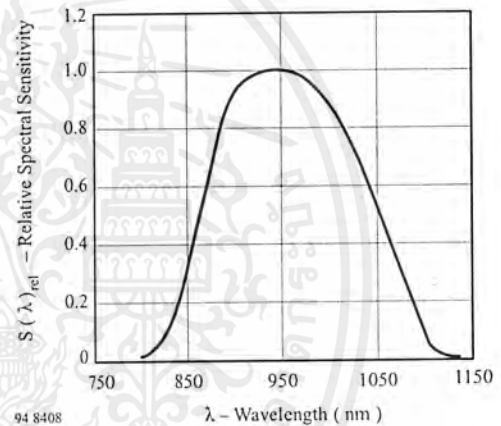


Figure 11. Relative Spectral Sensitivity vs. Wavelength

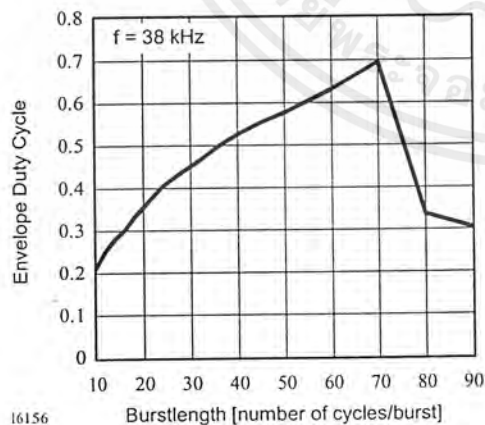


Figure 9. Max. Envelope Duty Cycle vs. Burstlength

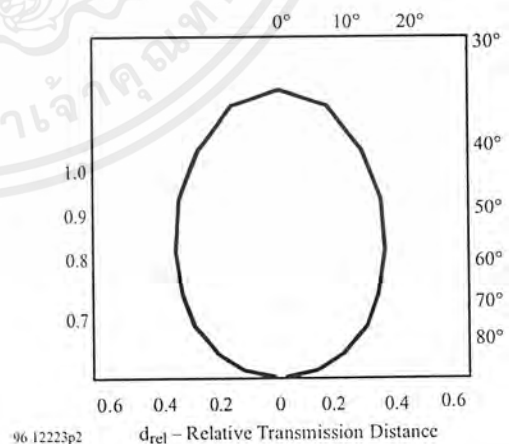


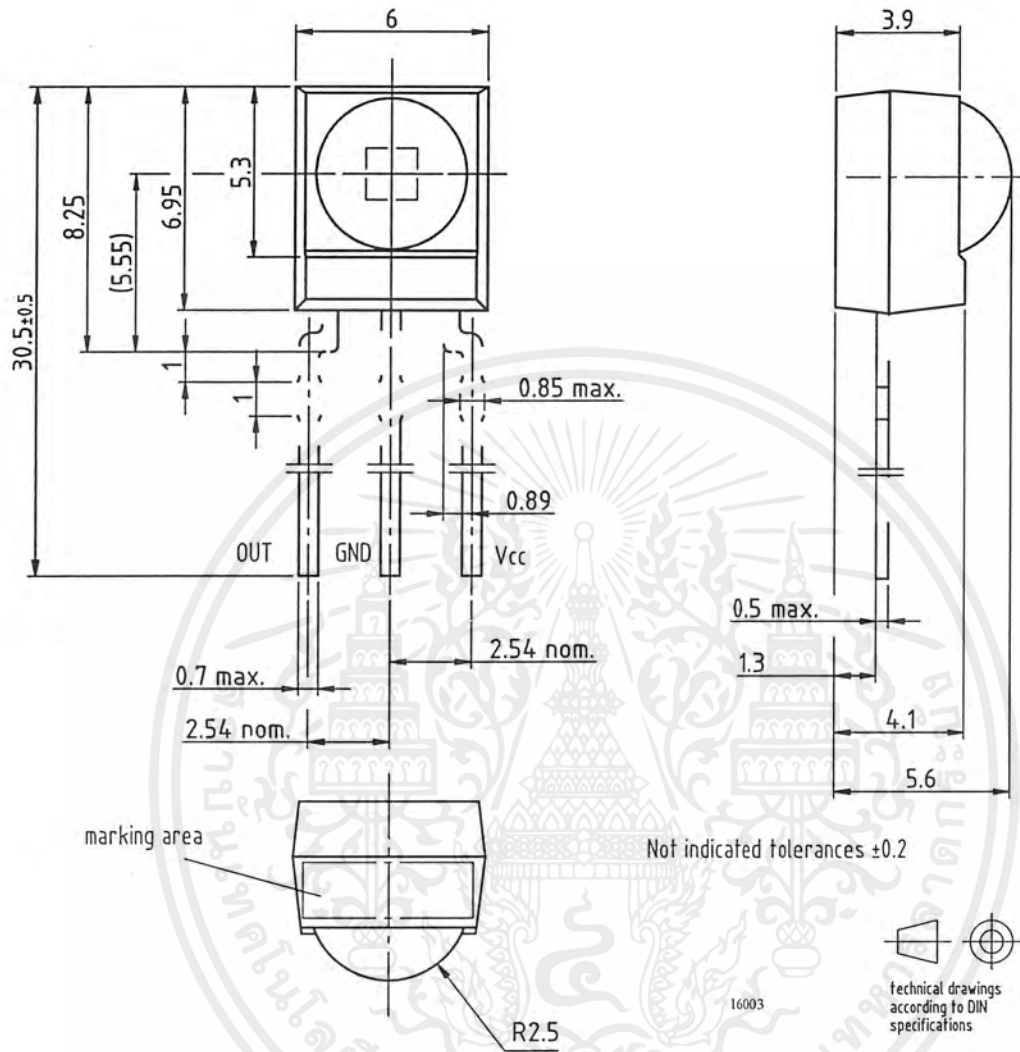
Figure 12. Directivity

TSOP48..

Vishay Telefunken



Dimensions in mm



Ozone Depleting Substances Policy Statement

It is the policy of **Vishay Semiconductor GmbH** to

1. Meet all present and future national and international statutory requirements.
2. Regularly and continuously improve the performance of our products, processes, distribution and operating systems with respect to their impact on the health and safety of our employees and the public, as well as their impact on the environment.

It is particular concern to control or eliminate releases of those substances into the atmosphere which are known as ozone depleting substances (ODSs).

The Montreal Protocol (1987) and its London Amendments (1990) intend to severely restrict the use of ODSs and forbid their use within the next ten years. Various national and international initiatives are pressing for an earlier ban on these substances.

Vishay Semiconductor GmbH has been able to use its policy of continuous improvements to eliminate the use of ODSs listed in the following documents.

1. Annex A, B and list of transitional substances of the Montreal Protocol and the London Amendments respectively
2. Class I and II ozone depleting substances in the Clean Air Act Amendments of 1990 by the Environmental Protection Agency (EPA) in the USA
3. Council Decision 88/540/EEC and 91/690/EEC Annex A, B and C (transitional substances) respectively.

Vishay Semiconductor GmbH can certify that our semiconductors are not manufactured with ozone depleting substances and do not contain such substances.

We reserve the right to make changes to improve technical design and may do so without further notice. Parameters can vary in different applications. All operating parameters must be validated for each customer application by the customer. Should the buyer use Vishay-Telefunken products for any unintended or unauthorized application, the buyer shall indemnify Vishay-Telefunken against all claims, costs, damages, and expenses, arising out of, directly or indirectly, any claim of personal damage, injury or death associated with such unintended or unauthorized use.

Vishay Semiconductor GmbH, P.O.B. 3535, D-74025 Heilbronn, Germany
Telephone: 49 (0) 7131 67 2831, Fax number: 49 (0) 7131 67 2423



MOTOROLA

Order this document by LM317/D

LM317

Three-Terminal Adjustable Output Positive Voltage Regulator

The LM317 is an adjustable 3-terminal positive voltage regulator capable of supplying in excess of 1.5 A over an output voltage range of 1.2 V to 37 V. This voltage regulator is exceptionally easy to use and requires only two external resistors to set the output voltage. Further, it employs internal current limiting, thermal shutdown and safe area compensation, making it essentially blow-out proof.

The LM317 serves a wide variety of applications including local, on card regulation. This device can also be used to make a programmable output regulator, or by connecting a fixed resistor between the adjustment and output, the LM317 can be used as a precision current regulator.

- Output Current in Excess of 1.5 A
- Output Adjustable between 1.2 V and 37 V
- Internal Thermal Overload Protection
- Internal Short Circuit Current Limiting Constant with Temperature
- Output Transistor Safe-Area Compensation
- Floating Operation for High Voltage Applications
- Available in Surface Mount D²PAK, and Standard 3-Lead Transistor Package
- Eliminates Stocking many Fixed Voltages

THREE-TERMINAL ADJUSTABLE POSITIVE VOLTAGE REGULATOR

SEMICONDUCTOR
TECHNICAL DATA

T SUFFIX
PLASTIC PACKAGE
CASE 221A

Heatsink surface
connected to Pin 2.



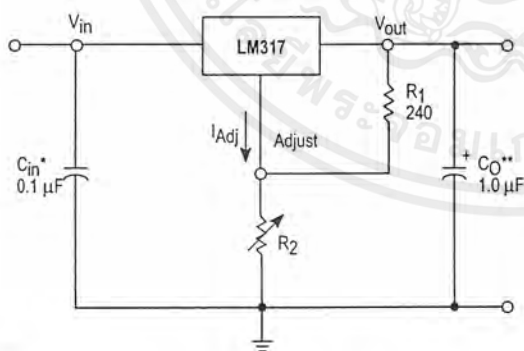
Pin 1. Adjust
Pin 2. V_{out}
Pin 3. V_{in}

D²T SUFFIX
PLASTIC PACKAGE
CASE 936
(D²PAK)

Heatsink surface (shown as terminal 4 in
case outline drawing) is connected to Pin 2.



Standard Application



* C_{in} is required if regulator is located an appreciable distance from power supply filter.

** C_O is not needed for stability, however, it does improve transient response.

$$V_{out} = 1.25 \text{ V} \left(1 + \frac{R_2}{R_1} \right) + I_{Adj} R_2$$

Since I_{Adj} is controlled to less than 100 μA, the error associated with this term is negligible in most applications.

ORDERING INFORMATION

Device	Operating Temperature Range	Package
LM317BD2T	T _J = -40° to +125°C	Surface Mount
LM317BT		Insertion Mount
LM317D2T	T _J = 0° to +125°C	Surface Mount
LM317T		Insertion Mount

LM317

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Input-Output Voltage Differential	$V_I - V_O$	40	Vdc
Power Dissipation Case 221A $T_A = +25^\circ\text{C}$ Thermal Resistance, Junction-to-Ambient Thermal Resistance, Junction-to-Case Case 936 (D ² PAK) $T_A = +25^\circ\text{C}$ Thermal Resistance, Junction-to-Ambient Thermal Resistance, Junction-to-Case	P_D θ_{JA} θ_{JC} P_D θ_{JA} θ_{JC}	Internally Limited 65 5.0 Internally Limited 70 5.0	W $^\circ\text{C/W}$ $^\circ\text{C/W}$ W $^\circ\text{C/W}$ $^\circ\text{C/W}$
Operating Junction Temperature Range	T_J	-40 to +125	$^\circ\text{C}$
Storage Temperature Range	T_{stg}	-65 to +150	$^\circ\text{C}$

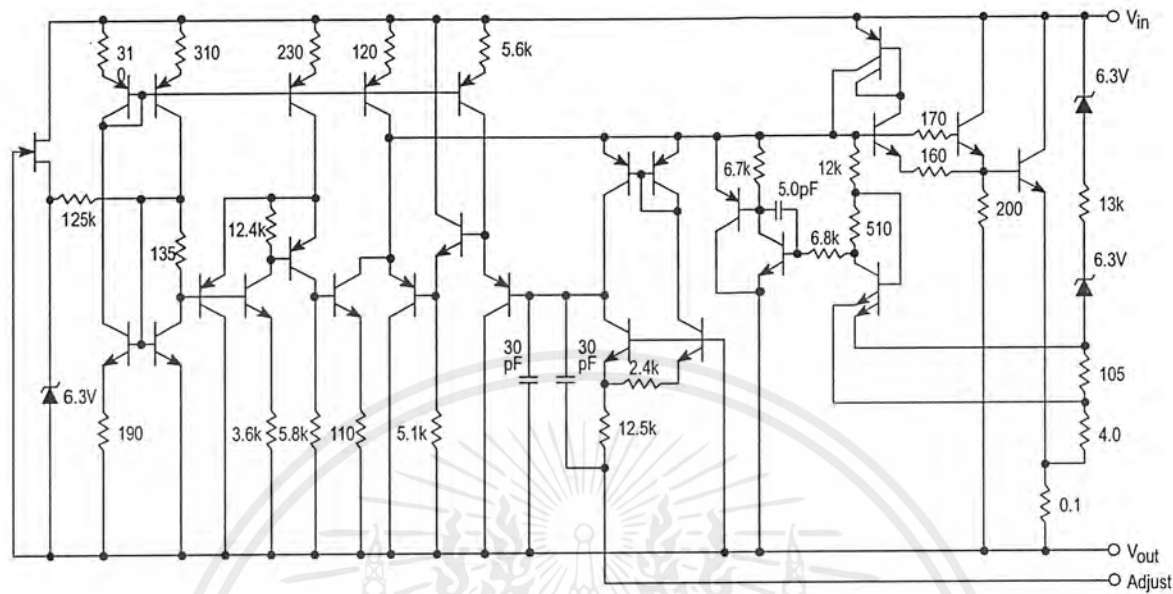
ELECTRICAL CHARACTERISTICS ($V_I - V_O = 5.0\text{ V}$; $I_O = 0.5\text{ A}$ for D2T and T packages; $T_J = T_{\text{low}}$ to T_{high} [Note 1]; I_{max} and P_{max} [Note 2]; unless otherwise noted.)

Characteristics	Figure	Symbol	Min	Typ	Max	Unit
Line Regulation (Note 3), $T_A = +25^\circ\text{C}$, $3.0\text{ V} \leq V_I - V_O \leq 40\text{ V}$	1	Regline	—	0.01	0.04	%/V
Load Regulation (Note 3), $T_A = +25^\circ\text{C}$, $10\text{ mA} \leq I_O \leq I_{\text{max}}$ $V_O \leq 5.0\text{ V}$ $V_O \geq 5.0\text{ V}$	2	Regload	— —	5.0 0.1	25 0.5	mV % V_O
Thermal Regulation, $T_A = +25^\circ\text{C}$ (Note 6), 20 ms Pulse		Regtherm	—	0.03	0.07	% V_O /W
Adjustment Pin Current	3	I_{Adj}	—	50	100	μA
Adjustment Pin Current Change, $2.5\text{ V} \leq V_I - V_O \leq 40\text{ V}$, $10\text{ mA} \leq I_L \leq I_{\text{max}}$, $P_D \leq P_{\text{max}}$	1, 2	ΔI_{Adj}	—	0.2	5.0	μA
Reference Voltage, $3.0\text{ V} \leq V_I - V_O \leq 40\text{ V}$, $10\text{ mA} \leq I_O \leq I_{\text{max}}$, $P_D \leq P_{\text{max}}$	3	V_{ref}	1.2	1.25	1.3	V
Line Regulation (Note 3), $3.0\text{ V} \leq V_I - V_O \leq 40\text{ V}$	1	Regline	—	0.02	0.07	% V
Load Regulation (Note 3), $10\text{ mA} \leq I_O \leq I_{\text{max}}$ $V_O \leq 5.0\text{ V}$ $V_O \geq 5.0\text{ V}$	2	Regload	— —	20 0.3	70 1.5	mV % V_O
Temperature Stability ($T_{\text{low}} \leq T_J \leq T_{\text{high}}$)	3	T_S	—	0.7	—	% V_O
Minimum Load Current to Maintain Regulation ($V_I - V_O = 40\text{ V}$)	3	I_{Lmin}	—	3.5	10	mA
Maximum Output Current $V_I - V_O \leq 15\text{ V}$, $P_D \leq P_{\text{max}}$, T Package $V_I - V_O = 40\text{ V}$, $P_D \leq P_{\text{max}}$, $T_A = +25^\circ\text{C}$, T Package	3	I_{max}	1.5 0.15	2.2 0.4	— —	A
RMS Noise, % of V_O , $T_A = +25^\circ\text{C}$, $10\text{ Hz} \leq f \leq 10\text{ kHz}$		N	—	0.003	—	% V_O
Ripple Rejection, $V_O = 10\text{ V}$, $f = 120\text{ Hz}$ (Note 4) Without C_{Adj} $C_{\text{Adj}} = 10\text{ }\mu\text{F}$	4	RR	— 66	65 80	— —	dB
Long-Term Stability, $T_J = T_{\text{high}}$ (Note 5), $T_A = +25^\circ\text{C}$ for Endpoint Measurements	3	S	—	0.3	1.0	%/1.0 k Hrs.
Thermal Resistance Junction to Case, T Package		$R_{\theta JC}$	—	5.0	—	$^\circ\text{C/W}$

- NOTES:** 1. T_{low} to $T_{\text{high}} = 0^\circ$ to $+125^\circ\text{C}$, for LM317T, D2T. T_{low} to $T_{\text{high}} = -40^\circ$ to $+125^\circ\text{C}$, for LM317BT, BD2T.
2. $I_{\text{max}} = 1.5\text{ A}$, $P_{\text{max}} = 20\text{ W}$
3. Load and line regulation are specified at constant junction temperature. Changes in V_O due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.
4. C_{Adj} , when used, is connected between the adjustment pin and ground.
5. Since Long-Term Stability cannot be measured on each device before shipment, this specification is an engineering estimate of average stability from lot to lot.
6. Power dissipation within an IC voltage regulator produces a temperature gradient on the die, affecting individual IC components on the die. These effects can be minimized by proper integrated circuit design and layout techniques. Thermal Regulation is the effect of these temperature gradients on the output voltage and is expressed in percentage of output change per watt of power change in a specified time.

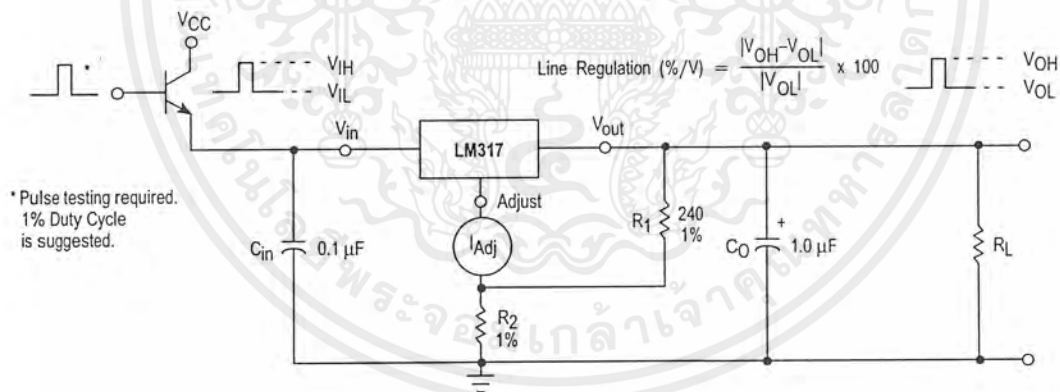
LM317

Representative Schematic Diagram



This device contains 29 active transistors.

Figure 1. Line Regulation and $\Delta I_{Adj}/Line$ Test Circuit



LM317

Figure 2. Load Regulation and $\Delta I_{Adj}/\text{Load}$ Test Circuit

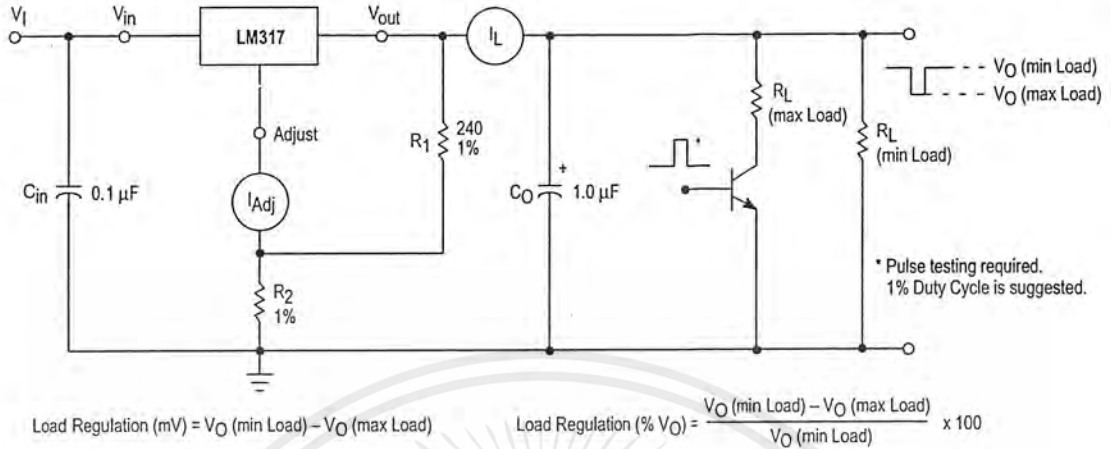


Figure 3. Standard Test Circuit

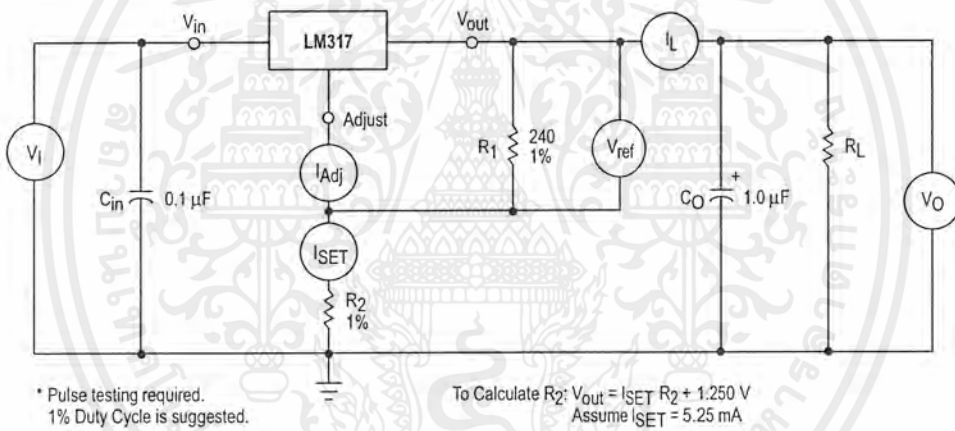


Figure 4. Ripple Rejection Test Circuit

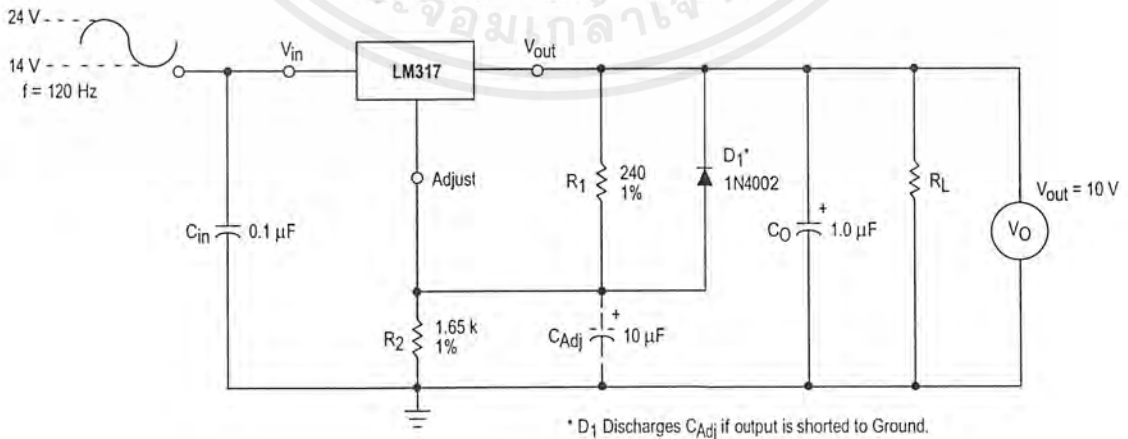


Figure 5. Load Regulation

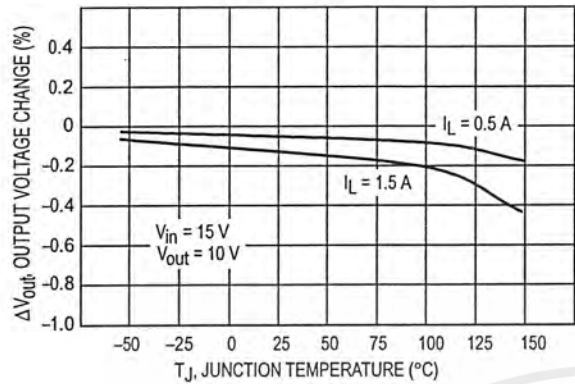


Figure 6. Current Limit

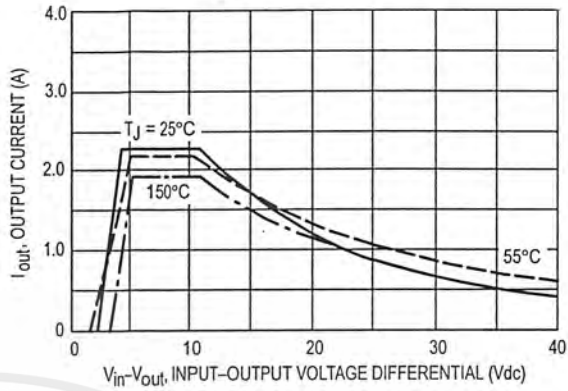


Figure 7. Adjustment Pin Current

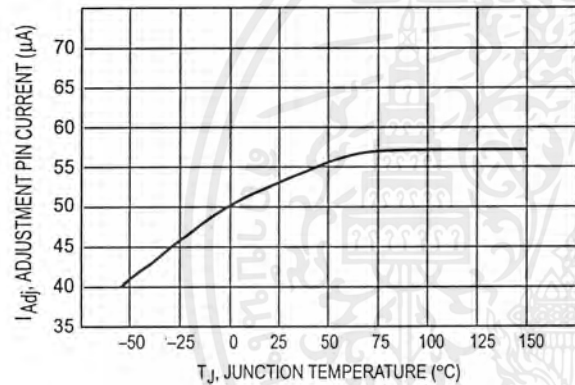


Figure 8. Dropout Voltage

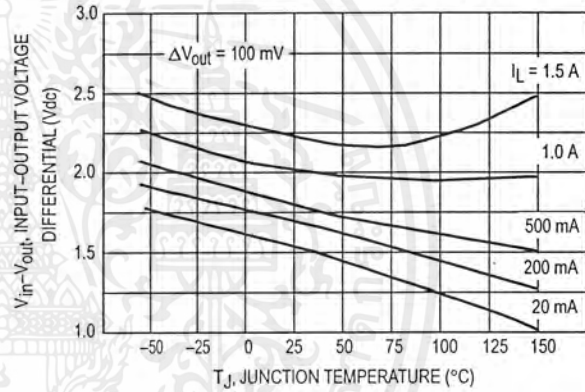


Figure 9. Temperature Stability

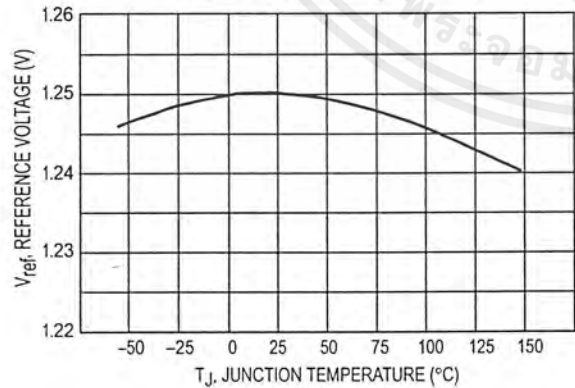


Figure 10. Minimum Operating Current

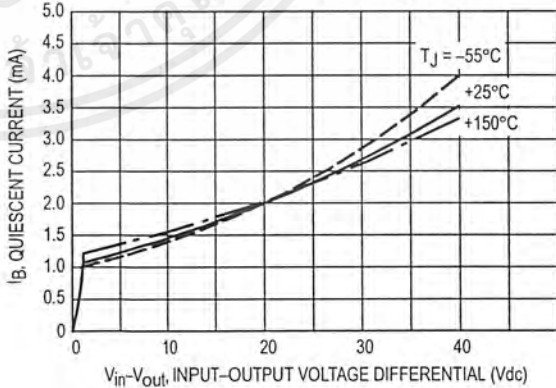


Figure 11. Ripple Rejection versus Output Voltage

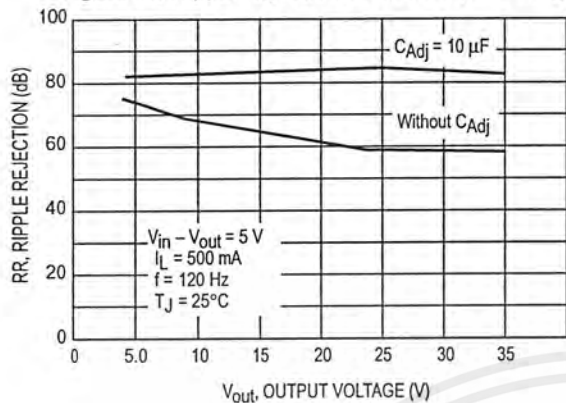


Figure 12. Ripple Rejection versus Output Current

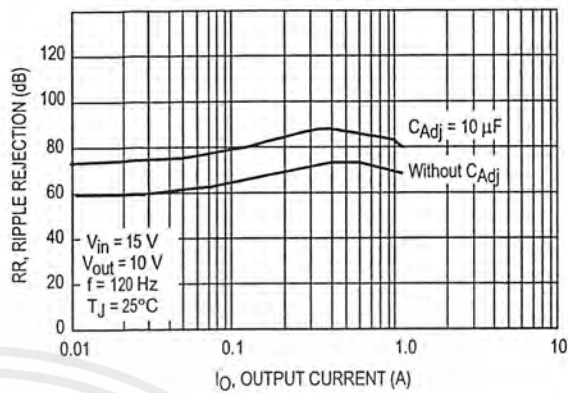


Figure 13. Ripple Rejection versus Frequency

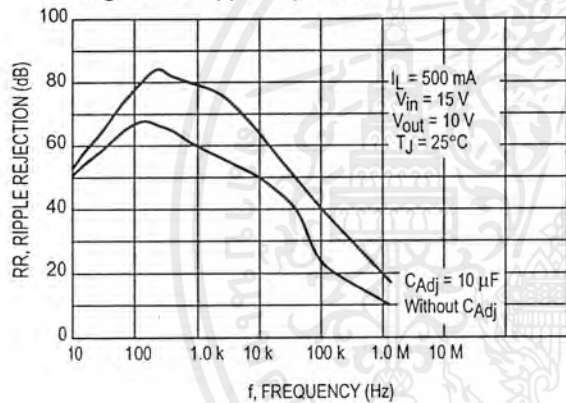


Figure 14. Output Impedance

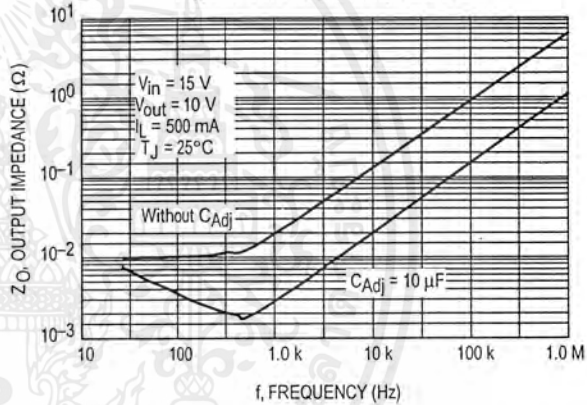


Figure 15. Line Transient Response

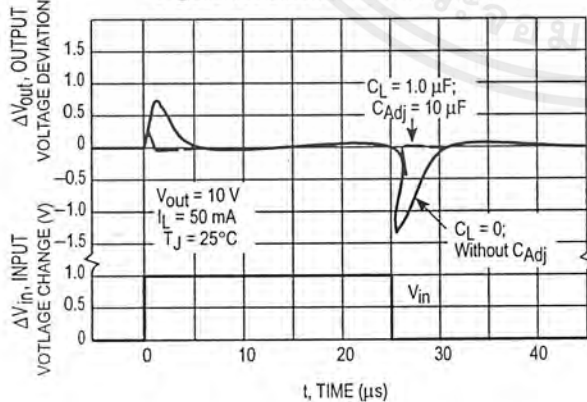
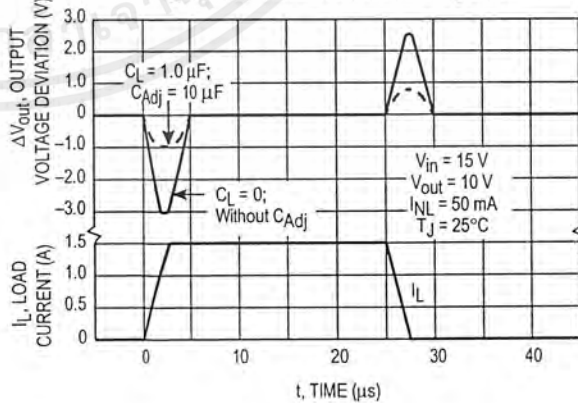


Figure 16. Load Transient Response



LM317

APPLICATIONS INFORMATION

Basic Circuit Operation

The LM317 is a 3-terminal floating regulator. In operation, the LM317 develops and maintains a nominal 1.25 V reference (V_{ref}) between its output and adjustment terminals. This reference voltage is converted to a programming current (I_{PROG}) by R_1 (see Figure 17), and this constant current flows through R_2 to ground.

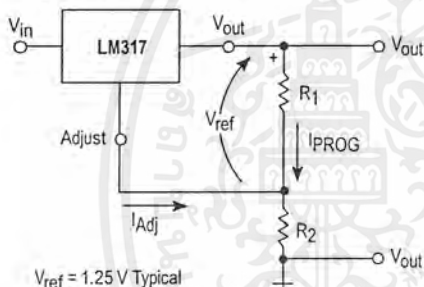
The regulated output voltage is given by:

$$V_{out} = V_{ref} \left(1 + \frac{R_2}{R_1} \right) + I_{Adj} R_2$$

Since the current from the adjustment terminal (I_{Adj}) represents an error term in the equation, the LM317 was designed to control I_{Adj} to less than 100 μA and keep it constant. To do this, all quiescent operating current is returned to the output terminal. This imposes the requirement for a minimum load current. If the load current is less than this minimum, the output voltage will rise.

Since the LM317 is a floating regulator, it is only the voltage differential across the circuit which is important to performance, and operation at high voltages with respect to ground is possible.

Figure 17. Basic Circuit Configuration



Load Regulation

The LM317 is capable of providing extremely good load regulation, but a few precautions are needed to obtain maximum performance. For best performance, the programming resistor (R_1) should be connected as close to the regulator as possible to minimize line drops which effectively appear in series with the reference, thereby degrading regulation. The ground end of R_2 can be returned near the load ground to provide remote ground sensing and improve load regulation.

External Capacitors

A 0.1 μF disc or 1.0 μF tantalum input bypass capacitor (C_{in}) is recommended to reduce the sensitivity to input line impedance.

The adjustment terminal may be bypassed to ground to improve ripple rejection. This capacitor (C_{Adj}) prevents ripple from being amplified as the output voltage is increased. A 10 μF capacitor should improve ripple rejection about 15 dB at 120 Hz in a 10 V application.

Although the LM317 is stable with no output capacitance, like any feedback circuit, certain values of external capacitance can cause excessive ringing. An output capacitance (C_O) in the form of a 1.0 μF tantalum or 25 μF aluminum electrolytic capacitor on the output swamps this effect and insures stability.

Protection Diodes

When external capacitors are used with any IC regulator it is sometimes necessary to add protection diodes to prevent the capacitors from discharging through low current points into the regulator.

Figure 18 shows the LM317 with the recommended protection diodes for output voltages in excess of 25 V or high capacitance values ($C_O > 25 \mu F$, $C_{Adj} > 10 \mu F$). Diode D_1 prevents C_O from discharging thru the IC during an input short circuit. Diode D_2 protects against capacitor C_{Adj} discharging through the IC during an output short circuit. The combination of diodes D_1 and D_2 prevents C_{Adj} from discharging through the IC during an input short circuit.

Figure 18. Voltage Regulator with Protection Diodes

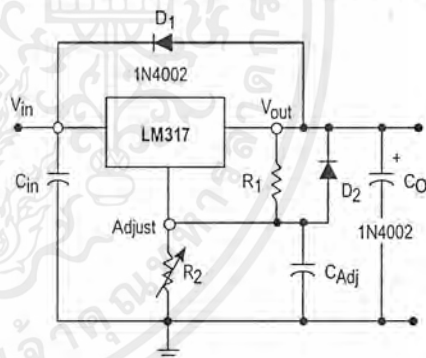
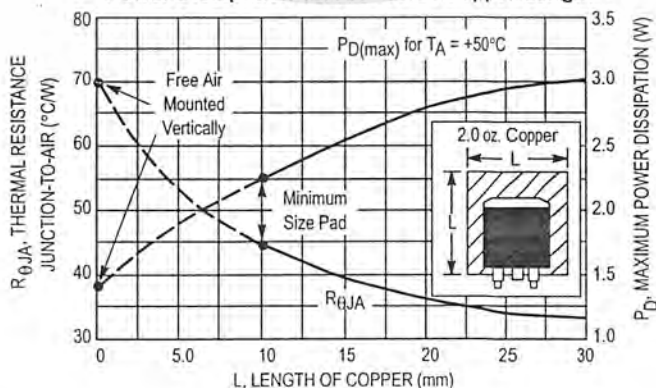


Figure 19. D²PAK Thermal Resistance and Maximum Power Dissipation versus P.C.B. Copper Length



LM317

Figure 20. "Laboratory" Power Supply with Adjustable Current Limit and Output Voltage

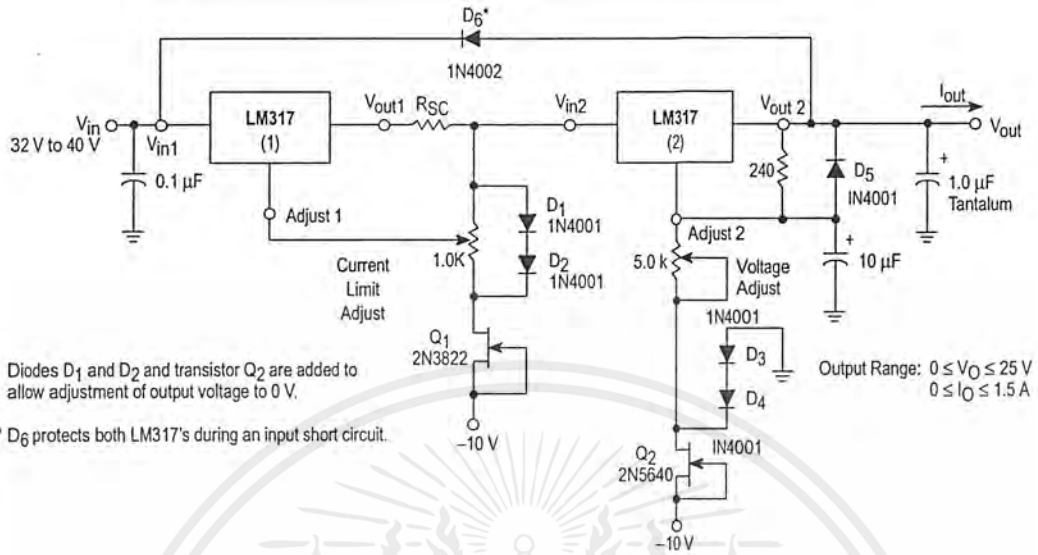


Figure 21. Adjustable Current Limiter

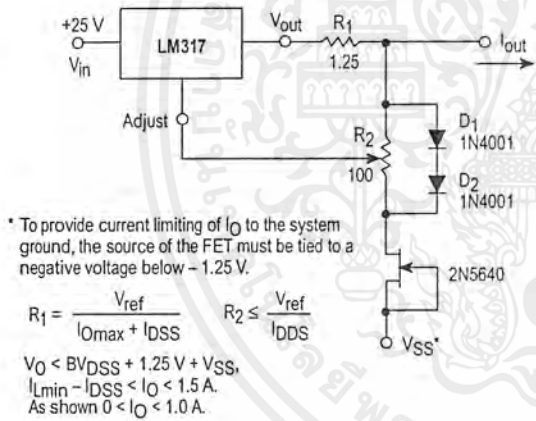


Figure 22. 5.0 V Electronic Shutdown Regulator

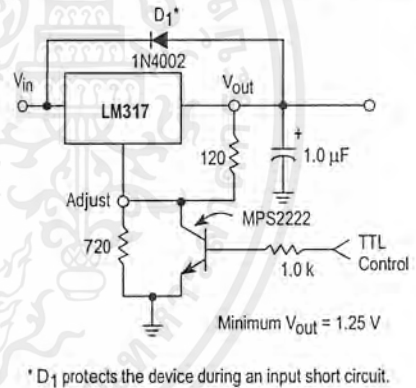


Figure 23. Slow Turn-On Regulator

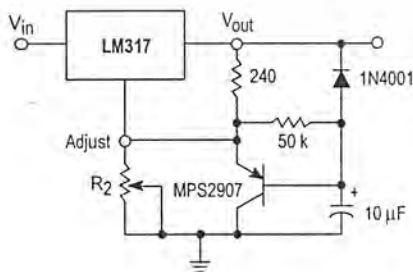
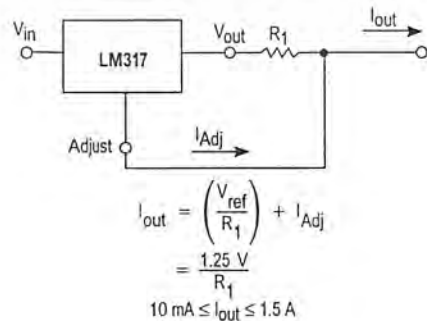


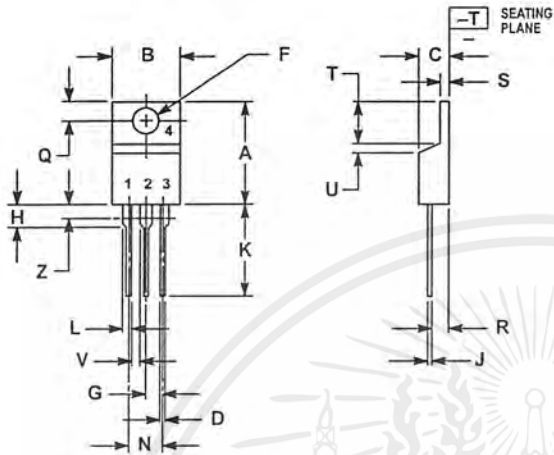
Figure 24. Current Regulator



LM317

OUTLINE DIMENSIONS

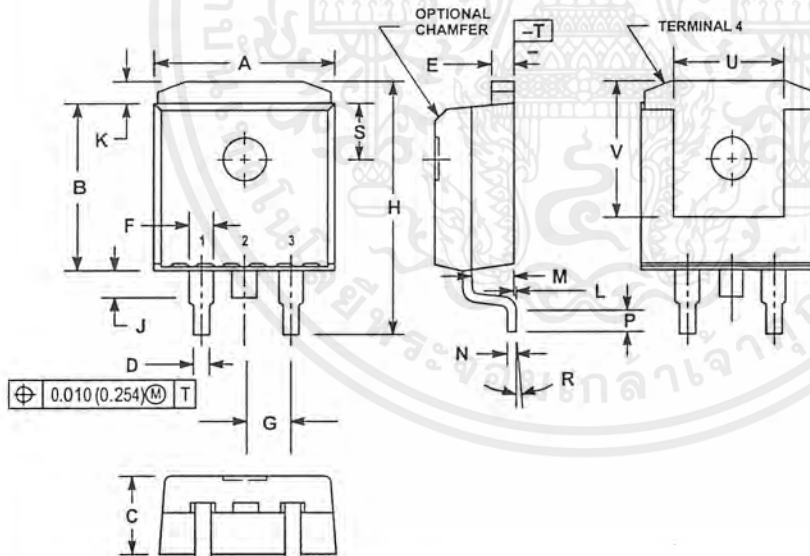
T SUFFIX PLASTIC PACKAGE CASE 221A-06 ISSUE Y



- NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
 2. CONTROLLING DIMENSION: INCH.
 3. DIM Z DEFINES A ZONE WHERE ALL BODY AND LEAD IRREGULARITIES ARE ALLOWED.

DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.570	0.620	14.48	15.75
B	0.380	0.405	9.66	10.28
C	0.160	0.190	4.07	4.82
D	0.025	0.035	0.64	0.88
F	0.142	0.147	3.61	3.73
G	0.095	0.105	2.42	2.66
H	0.110	0.155	2.80	3.93
J	0.018	0.025	0.46	0.64
K	0.500	0.562	12.70	14.27
L	0.045	0.060	1.15	1.52
N	0.190	0.210	4.83	5.33
Q	0.100	0.120	2.54	3.04
R	0.080	0.110	2.04	2.79
S	0.045	0.055	1.15	1.39
T	0.235	0.255	5.97	6.47
U	0.000	0.050	0.00	1.27
V	0.045	-	1.15	-
Z	-	0.080	-	2.04

D2T SUFFIX PLASTIC PACKAGE CASE 936-03 (D²PAK) ISSUE B



- NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
 2. CONTROLLING DIMENSION: INCH.
 3. TAB CONTOUR OPTIONAL WITHIN DIMENSIONS A AND K.
 4. DIMENSIONS U AND V ESTABLISH A MINIMUM MOUNTING SURFACE FOR TERMINAL 4.
 5. DIMENSIONS A AND B DO NOT INCLUDE MOLD FLASH OR GATE PROTRUSIONS. MOLD FLASH AND GATE PROTRUSIONS NOT TO EXCEED 0.025 (0.635) MAXIMUM.

DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	0.386	0.403	9.804	10.236
B	0.358	0.368	9.042	9.347
C	0.170	0.180	4.318	4.572
D	0.026	0.036	0.660	0.914
E	0.045	0.055	1.143	1.397
F	0.051	REF	1.295	REF
G	0.100	BSC	2.540	BSC
H	0.539	0.579	13.691	14.707
J	0.125	MAX	3.175	MAX
K	0.050	REF	1.270	REF
L	0.000	0.010	0.000	0.254
M	0.088	0.102	2.235	2.591
N	0.018	0.026	0.457	0.660
P	0.058	0.078	1.473	1.981
R	5°	REF	5°	REF
S	0.116	REF	2.946	REF
U	0.200	MIN	5.080	MIN
V	0.250	MIN	6.350	MIN



Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

Mfax is a trademark of Motorola, Inc.

How to reach us:

USA / EUROPE / Locations Not Listed: Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217. 303-675-2140 or 1-800-441-2447

JAPAN: Nippon Motorola Ltd.: SPD, Strategic Planning Office, 4-32-1,
Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan. 81-3-5487-8488

Mfax™: RMFAX0@email.sps.mot.com – TOUCHSTONE 602-244-6609
– US & Canada ONLY 1-800-774-1848

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

INTERNET: <http://motorola.com/sps>



MOTOROLA



กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถจัดทำสำเร็จล่วงได้ด้วยดี ก็เพราะได้รับคำแนะนำและคำปรึกษาจาก ท่าน รศ.ดร. ถวิล พึ่งมา และท่าน ศ.มบุญ สุขเกษม คณะผู้จัดทำจึงขอขอบพระคุณเป็นอย่างสูง

คณะผู้จัดทำขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ที่ให้กำลังใจและเป็นผู้ซึ่งให้โอกาสทางการศึกษา และทุกสิ่งทุกอย่างมาโดยตลอด

ขอขอบคุณพี่ พิสิฐ บุญศรีเมือง , สายัณห์ พวงเงิน, นิวัติ เย็นกาย ที่คอยให้คำปรึกษา คำแนะนำ และให้การช่วยเหลือแก้ปัญหาเกี่ยวกับปริญญานิพนธ์ฉบับนี้

ขอขอบคุณ พี่ๆ เพื่อนๆ และน้องๆ ทุกคนที่คอยเป็นกำลังใจในการทำปริญญานิพนธ์มาโดยตลอด

คณะผู้จัดทำ

นาย คำรงค์ฤทธิ์ ภัณฑารักษ์สกุล

นาย สิทธิชัย เดชพร

นาย สุทัศน์ พริ้งสกุลชัย



หนังสืออ้างอิง

- [1] เกียรติศักดิ์ ฐานิกเกษตร และ ประวิทย์ โคมทองชูสกุล, “หลักการดำเนินงานของ 555 ไทเมอร์”, โครงการงานไอซี 110, หน้า 27 – 29 .
- [2] ชัยวัฒน์ ลัมพรจิตรวิไล และ วรพจน์ กรแก้ววัฒนกุล, “การขับโมดูลแสดงผลแบบผลึกเหลว”, เรียนรู้และปฏิบัติการ ไมโครคอนโทรลเลอร์, บริษัท อินโนโวลิตี้ เอ็กเพอริเมนต์ จำกัด, หน้า 283-301, 2541 .
- [3] ณรงค์ ย่างสกุล, “ง่าย ๆ กับการประยุกต์ใช้งานตัวตรวจจับแบบต่าง ๆ”, วารสารเคมีคอนดักเตอร์อิเล็กทรอนิกส์ ฉบับที่ 143, ตุลาคม 2537, หน้า 88 – 94 .
- [4] มงคล เดชนครินทร์ และ ชาดรี ศรีไพพรรณ, “ออกแบบการออกแบบและการใช้งาน”, อิเล็กทรอนิกส์พื้นฐาน, บริษัท ซีเอ็ดดูเคชั่น จำกัด(มหาชน), หน้า 270 – 281, 2538 .
- [5] ยืน ภู่วรรณ, “ออปเปอเรชันัล แอมพลิฟายเออร์”, ทฤษฎีและการใช้งานอิเล็กทรอนิกส์ 2, บริษัท ซีเอ็ด ดูเคชั่น จำกัด(มหาชน), หน้า 233 – 234, 2540 .
- [6] วินัย ชนปรมัตถ์, “แaggeroyการทำหุ่นยนต์”, วารสารเคมีคอนดักเตอร์อิเล็กทรอนิกส์, ฉบับที่ 168 กรกฎาคม 2541 , หน้า 225 – 231 .
- [7] อริคม ฤกษ์บุตร, “การควบคุมด้วยสัญญาณโทนาเบิสต์”, เครื่องควบคุมไร้สาย, บริษัท เมอร์คิวรี ไมโครเทคโนโลยี จำกัด, หน้า 33 – 34 .