ชุดพัฒนาระบบหุ่นยนต์

Robot development

โดย

นายเมธี นิลภารักษ์

นายวันลาภ พฤทธ์กิตติวงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชา วิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ชุดพัฒนาระบบหุ่นยนต์

Robot Development

ผู้จัดทำ

1. นายเมธี นิลภารักษ์ รหัส 42010278
2. นายวันลาภ พฤทธ์กิตติวงศ์ รหัส 42010322

.....อาจารย์ที่ปรึกษา

(อ.สุมิตร พนาอุดมทรัพย์)

ชุดพัฒนาระบบหุ่นยนต์

นายเมธี นิลภารักษ์
นายวันลาภ พฤทธ์กิตติวงศ์
อ. สุมิตร พนาอุดมทรัพย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการวิจัยและพัฒนาโปรแกรมต้นแบบที่ใช้ในการควบคุมหุ่นยนต์ที่ได้สร้างเอาไว้ โปรแกรมต้นแบบที่สร้างขึ้นมานั้นทำจากภาษาวิชวลเบสิก 6 (Visual Basic V.6) เป็นภาษาที่บุคคลที่ซึ่งไม่มีพื้นฐานทางด้านการเขียนโปรแกรมก็สามารถใช้โปรแกรมนี้ควบคุมหุ่นยนต์ได้โดยง่ายหลังจากเขียนโปรแกรมเสร็จ ก็สามารถที่จะดาวน์โหลดโปรแกรมไปยังหุ่นยนต์และทำงานตามโปรแกรมที่เขียนไว้ โดยตัวหุ่นยนต์มีการเคลื่อนไหวแบบอัตโนมัติ

Robot Development

Metee Nilparak
Wanlarp Preutkittiwong
Sumit Panaudomsub Advisor
2002

Abstract

This thesis presents a research and development of prototype program which uses to control the robot that has built for test. The prototype program makes from Visual Basic V.6 and It is the program for someone who hasn't the basic on programming to able to use this program to control the robot very easy. After use this program that can download the program to the robot and can work follow the program and the robot can move automatic.

สารบัญ

	หน้า
บทคัดย่อ	I
สารบัญรูป	III
บทที่ 1 บทนำ	1
1.1 ที่มาของหุ่นยนต์	1
1.2 ทฤษฎีทั่วไปเกี่ยวกับหุ่นยนต์	1
1.3 ประเภทต่างๆของหุ่นยนต์	1
1.4 ความเป็นมาของโครงงาน	2
1.5 วัตถุประสงค์ของโครงงาน	2
1.6 ขอบเขตของโครงงาน	2
บทที่ 2 ทฤษฎี	3
2.1 กลไกเคลื่อนไหว	4
2.1.1 DC Motor	4
2.1.2 Servo Motor	5
2.2 ส่วนตรวจจับหรือเซนเซอร์	7
2.2.1 Limit Switch	7
2.2.2 เซนเซอร์แสง	7
2.3 ส่วนควบคุม	10
2.3.1 P89C51RD2	10
2.3.2 89C51	11
2.3.3 89C2051	11
2.3.4 การทำงานแบบต่างๆของไมโครคอนโทรลเลอร์ที่ใช้ในโปรเจกต์	13
2.3.4.1 ไทเมอร์/เคาน์เตอร์ของไมโครคอนโทรลเลอร์ MCS-51	13
2.3.4.2 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของไทเมอร์/เคาน์เตอร์ 0 และ 1	14

หน้า

2.3.4.3 กระบวนการอินเตอร์รัปต์ของ
ไมโครคอนโทรลเลอร์ MCS-51 18
2.3.4.4 รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์
ในไมโครคอนโทรลเลอร์ MCS-51 19
2.3.4.5 การใช้งานไทเมอร์ภายในไมโครคอนโทรลเลอร์
MCS-51 กำเนิดสัญญาณพัลส์แบบปรับช่วง
เวลา on-off ได้ 20
2.4 ส่วนรับและแสดงผลของหุ่นยนต์ 23
2.4.1 ส่วนของการรับคำสั่งจากแป้นกดที่ตัวหุ่น 23
2.4.2 ส่วนของการแสดงผลออกหน้าจอ LCD 24
2.5 การเชื่อมต่อพอร์ตอนุกรมของคอมพิวเตอร์ 31
2.6 แหล่งจ่ายไฟ 35
2.6.1 แบตเตอรี่แห้ง 12 v 1.3AH 35
2.6.2 แบตเตอรี่แห้ง 7.2 v 2700 mAh 35
2.7 โปรแกรมที่ใช้ในการควบคุมหุ่นยนต์ 36
บทที่ 3 การออกแบบ 44
3.1 ส่วนของโครงสร้างทางเครื่องกล 44
3.1.1 ส่วนของการเคลื่อนที่ของหุ่นยนต์ 44
3.1.2 ส่วนของแขนกลของหุ่นยนต์ 45
3.1.2.1 แกนที่ 1 46
3.1.2.2 แกนที่ 2 47
3.1.2.3 แกนที่ 3 49
3.2 ส่วนของการควบคุมของหุ่นยนต์ 50
3.2.1ไมโครคอนโทรลเลอร์ P89C51RD2 50
3.2.2 ไมโครคอนโทรลเลอร์ 89C51 51
3.2.3 ไมโครคอนโทรลเลอร์ 89C512051(มี 2 ตัว ซ้ายและขวา) 51
3.2.4 คีย์แพด (Keypad) 52
3.2.5 จอแสดงผล LCD 54

หน้า

3.2.6 วงจรขับมอเตอร์ด้วยรีเลย์ 56

3.2.7 วงจรขับมอเตอร์ด้วย IC L298 56

3.2.8 ลิมิตสวิตช์ 57

3.2.9 เซอร์โวมอเตอร์ 58

3.2.10 เซนเซอร์ 58

3.3 ส่วนของโปรแกรมควบคุมหุ่นยนต์ 66

3.3.1 Program Boxes 67

3.3.2 Robot Development Code 68

3.3.3 Assembly Code 69

3.3.4 Download Part 69

3.4 วิธีการใช้งานของหุ่นยนต์ 75

บทที่ 4 การทดลองและผลการทดลอง 77

บทที่ 5 บทวิจารณ์และสรุป 79

ภาคผนวก

กิตติกรรมประกาศ

เอกสารอ้างอิง

สารบัญรูป

		หน้า
รูปที่ 2-1	ผังการทำงานโดยรวมเบื้องต้นของหุ่นยนต์อัตโนมัติ	3
รูปที่ 2-2	ตัวมอเตอร์ที่ใช้ขับเคลื่อนของหุ่นยนต์	4
รูปที่ 2-3	มือจับวัตถุของหุ่นยนต์โดยใช้เซอร์โวมอเตอร์ในการขับเคลื่อน	5
รูปที่ 2-4	แสดงเซอร์โวมอเตอร์ที่ใช้ในการควบคุมแขน	5
รูปที่ 2-5	แสดงตัวเซอร์โวมอเตอร์	6
รูปที่ 2-6	วงจร Limit Switch ที่ใช้ในหุ่นยนต์	7
รูปที่ 2-7	วงจรเซนเซอร์ที่ใช้ในหุ่นยนต์	7
รูปที่ 2-8	ลิมิตสวิตช์ของหุ่นยนต์เพื่อตรวจจับตำแหน่งของแขนหุ่น	8
รูปที่ 2-9	ลิมิตสวิตช์ที่ติดอยู่ที่มือจับของหุ่น	8
รูปที่ 2-10	การวางตำแหน่งเซนเซอร์ทั้ง 5 ตัวของหุ่น	9
รูปที่ 2-11	รูปแสดงหลักการทำงานของเซนเซอร์	9
รูปที่ 2-12	IC L298 ที่ใช้ในการขับ DC Motor ของหุ่นยนต์	12
รูปที่ 2-13	วงจรไมโครคอนโทรลเลอร์ทั้งหมดที่ใส่ไว้ในกล่อง	12
รูปที่ 2-14	รูปแสดงไมโครคอนโทรลเลอร์ชนิดต่างๆ	12
รูปที่ 2-15	ตารางแสดงฟังก์ชั่นและ TMOD ของไทเมอร์ 0	17
รูปที่ 2-16	Flow chat ของโปรแกรมการสร้างสัญญาณพัลส์	22
รูปที่ 2-17	แผนผังการเชื่อมต่อ LCD กับไมโครคอนโทรลเลอร์	31
รูปที่ 2-18	ไอซี MAX232 และโครงสร้างภายในของ ไอซี	32
รูปที่ 2-19	แสดงการเชื่อมต่อกับพอร์ตอนุกรมของไอซี	32
รูปที่ 2-20	การเชื่อมต่อของ Keypads, LCD, พอร์ตอนุกรม	33
รูปที่ 2-21	ผังการทำงานเบื้องต้นของส่วนควบคุมหลักในหุ่นยนต์อัตโนมัติ	34
รูปที่ 2-22	แบตเตอรี่แห้ง 12 V.	35
รูปที่ 2-23	แบตเตอรี่แห้ง 7.2 V.	35
รูปที่ 2-24	หน้าต่างการเลือกชนิดโปรเจกต์	36
รูปที่ 2-25	หน้าต่างหลักของโปรแกรม Visual Basic 6	37
รูปที่ 2-26	เมนูบาร์	37

		หน้า
รูปที่ 2-27	ทูลบาร์	38
รูปที่ 2-28	ทูลบ็อกซ์	38
รูปที่ 2-29	หน้าต่างโปรเจ็กต์	39
รูปที่ 2-30	หน้าต่างคุณสมบัติ	39
รูปที่ 2-31	หน้าต่างฟอร์ม	40
รูปที่ 2-32	หน้าต่าง Code Editor	40
รูปที่ 3-1	ทิศทางการหมุนของล้อเพื่อให้ได้ทิศทางต่างๆตามต้องการ	44
รูปที่ 3-2	แสดงรูปของล้อทั้ง 4 ของตัวหุ่นยนต์	45
รูปที่ 3-3	รูปแสดงแขนกล 3 แกนที่ออกแบบไว้ในหุ่นยนต์	45
รูปที่ 3-4	รูปแสดงการหมุนรอบแกน X-Y-Z	46
รูปที่ 3-5	รูปแสดงการสัมผัสลิมิตสวิตช์ของลูกปืน	46
รูปที่ 3-6	ภาพด้านข้างของแขนกลในตำแหน่งที่เริ่มต้น	47
รูปที่ 3-7	ภาพด้านข้างของแขนกลในตำแหน่งที่หมุนลงมาด้านหน้า	47
รูปที่ 3-8	รูปแสดงแขนหุ่นขณะหมุนยกขึ้นมาข้างบน	48
รูปที่ 3-9	รูปแสดงแขนหุ่นขณะหมุนลงมาหยิบของ	48
รูปที่ 3-10	รูปแสดงแขนกลยึดกับเซอร์โวมอเตอร์	48
รูปที่ 3-11	เซอร์โวมอเตอร์หมุนตามเข็มนาฬิกา	49
รูปที่ 3-12	เซอร์โวมอเตอร์หมุนทวนเข็มนาฬิกา	49
รูปที่ 3-13	รูปแสดงมือจับของแขนกลบีบเข้ามาสุด	50
รูปที่ 3-14	รูปแสดงมือจับของแขนกลอ้าออกไปสุด	50
รูปที่ 3-15	รูปแสดงวงจร Keypad	52
รูปที่ 3-16	โฟลว์ชาร์ตโปรแกรมย่อยการอ่านค่าคีย์แพด	53
รูปที่ 3-17	รูปแสดงวงจรจอ LCD MODULE	54
รูปที่ 3-18	โฟลวชาร์ตการส่งพัลส์เอ็นเอเบิลให้แก่โมดูล LCD	54
รูปที่ 3-19	โฟลวชาร์ตของการอินิเชียลโมดูล	55
รูปที่ 3-20	แสดงวงจรขับมอเตอร์ด้วยรีเลย์	56
รูปที่ 3-21	รูปแสดงวงจรลิมิตสวิตช์	57
รูปที่ 3-22	รูปแสดงวงจรเซนเซอร์	58

		หน้า
รูปที่ 3-23	การวางตำแหน่งของเซนเซอร์บนหุ่นยนต์ และแสดงถึงกรณีที่หุ่นยนต์วิ่งไปเจอสี่แยกแล้ว	59
รูปที่ 3-24	โฟลว์ชาร์ตการเก็บค่าอ้างอิงระหว่างสีพื้นกับสีเส้นของเซนเซอร์	61
รูปที่ 3-25	โฟลว์ชาร์ตแสดงการทำงานเพื่อเช็คว่าเป็นสีพื้นหรือสีเส้น	62
รูปที่ 3-26	วงจรส่วนประมวลผลกลาง 1	63
รูปที่ 3-27	วงจรส่วนประมวลผลกลาง 2	64
รูปที่ 3-28	วงจรแหล่งจ่ายไฟ 5 V.	65
รูปที่ 3-29	หน้าต่างหลักของโปรแกรม Robot Development	66
รูปที่ 3-30	รูปแสดงส่วนของ Program Boxes	68
รูปที่ 3-31	หน้าต่าง Robot Development ซึ่งมีลำดับของคำสั่งจาก Program Boxes	68
รูปที่ 3-32	หน้าต่าง Assembly Code ซึ่งแปลงโค้ดมาจาก หน้าต่างRobot Development Code	69
รูปที่ 3-33	รูปแสดงหน้าต่างการเซฟโปรแกรมไว้ใน Text File	69
รูปที่ 3-34	รูปแสดงหน้าต่างของ Download Parts	70
รูปที่ 3-35	รูปแสดงหน้าต่างแสดงผลของการแปลง Hex file	70
รูปที่ 3-36	รูปแสดงโปรแกรม Micro ISP	70
รูปที่ 3-37	โฟลว์ชาร์ตขั้นตอนการทำงานระบบรวม	74
รูปที่ 3-38	รูปแสดงหน้าจอ LCD เมื่อเปิดสวิตช์ Power	75
รูปที่ 3-39	รูปแสดงหน้าจอ LCD เมื่อกดปุ่ม PGM	75
รูปที่ 3-40	รูปแสดงจอLCD เมื่อกดปุ่ม RUN	75
รูปที่ 3-41	รูปแสดงจอLCD เมื่อกดปุ่ม C2	76
รูปที่ 3-42	รูปแสดงจอLCD เมื่อกดปุ่ม Str	76
รูปที่ 3-43	รูปแสดงจอ LCD เมื่อกดปุ่ม CCL.	76
รูปที่ 4-1	ผังจำลองสนามในการใช้ของหุ่นยนต์	77
รูปที่ 4-2	รูปแสดงเส้นทางเดินที่ให้หุ่นยนต์เดินไป	77

บทที่ 1
บทนำ

1.1 ที่มาของหุ่นยนต์

มนุษย์นั้นได้สร้างเครื่องอำนวยความสะดวกขึ้นมา เพื่อให้ใช้ในชีวิตประจำวันและสร้างสิ่งต่างๆ ได้ มนุษย์กับเครื่องจักรนั้นจึงเป็นสิ่งที่อยู่ควบคู่กันมาโดยตลอด เพราะ ทางด้านกายภาพแล้วมนุษย์ไม่สามารถทำกิจกรรมที่ซ้ำๆกันได้เหมือนกันทุกครั้ง เช่น ไม่สามารถเขียนหนังสือได้อักษรตัวเดิมเท่ากันทุกตัวก็ได้คิดพิมพ์ดีดขึ้นมาใช้

ปัจจุบันหุ่นยนต์มีบทบาทสำคัญมากในโรงงานอุตสาหกรรมต่างๆ อีกทั้งยังสามารถให้ความสะดวกสบายแก่คนในสังคมได้เช่น แขนหุ่นยนต์ในโรงงานผลิตรถยนต์ หุ่นยนต์เดินสองขาของฮอนด้าที่มีความสามารถคล้ายมนุษย์ เป็นต้น ซึ่งในการที่จะสร้างและควบคุมหุ่นยนต์ดังที่กล่าวข้างต้นนั้นผู้สร้างจะต้องมีความรู้หลักๆอยู่ 3 ด้าน คือ ความรู้ด้านเครื่องกล ด้านฮาร์ดแวร์ของหุ่นยนต์ ด้านซอฟแวร์ของหุ่นยนต์ ซึ่งพอสร้างหุ่นยนต์ขึ้นมาได้ตัวหนึ่งก็จะเจอปัญหาการเขียนโปรแกรมที่ซับซ้อนดังนั้นบุคคลที่จะควบคุมนั้นทำให้จำกัดอยู่ในวงแคบ บุคคลที่ไม่มีความรู้ด้านที่กล่าวมาก็ไม่สามารถที่จะควบคุมหุ่นยนต์ได้ แต่ถ้าเราสามารถที่จะสร้างหุ่นยนต์ที่มีระบบควบคุมที่คนที่ไม่มีความรู้ด้านดังกล่าวก็สามารถที่จะควบคุมหุ่นยนต์ได้ซึ่งมันก็จะทำให้เกิดความสะดวกสบายแก่ผู้ที่ควบคุมหุ่นยนต์นั้นมากขึ้น

1.2 ทฤษฎีทั่วไปเกี่ยวกับหุ่นยนต์

มีคุณสมบัติอย่างกว้างๆ ของหุ่นยนต์ก็มีพอสังเขปดังนี้

- ควบคุมด้วยคอมพิวเตอร์
- มีส่วนประกอบที่เคลื่อนไหวได้
- มียูสเซอร์อินเตอร์เฟส (User Interface) เป็นส่วนติดต่อระหว่างมนุษย์กับตัวหุ่นยนต์
- ตัวอินเตอร์เฟส (Interface) อาจเป็นแค่ปุ่มเริ่มการทำงานหรือกุญแจเปิดปิด
- สามารถโปรแกรมให้งานต่างๆได้

1.3 ประเภทต่าง ๆ ของหุ่นยนต์

หุ่นยนต์อาจแบ่งออกเป็น 2 ประเภทใหญ่ๆ คือ

1.3.1 Fixed Robots คือ หุ่นยนต์ที่ถูกตรึงกับฐานที่ถูกยึดคงที่ ไม่สามารถเคลื่อนที่ได้อิสระ

1.3.2 Mobile Robots คือ หุ่นยนต์ที่ถูกตรึงกับฐานที่สามารถเคลื่อนที่ได้โดยอิสระ โดยที่ฐานมีล้อหรือต้นระบบ

1.4 ความเป็นมาของโครงงาน

เนื่องจากหุ่นยนต์ได้รับความสนใจจากสังคมปัจจุบันมากขึ้น และเยาวชนรุ่นใหม่จะเป็นกำลังสำคัญของชาติในการพัฒนาเทคโนโลยีด้านหุ่นยนต์ ซึ่งกระผมได้เห็นชุดทดลองหุ่นยนต์ของต่างประเทศเค้าสามารถทำให้เยาวชนรุ่นเล็กๆสามารถที่จะใช้ความคิดในการออกแบบ และเขียนโปรแกรมให้กับหุ่นยนต์ ซึ่งผมก็อยากจะให้เด็กไทยสามารถที่จะได้พัฒนาและปลูกฝังความเป็นวิศวกรรมให้แก่เยาวชนรุ่นต่อๆไป

1.5 วัตถุประสงค์ของโครงงาน

การสร้างชุดพัฒนาหุ่นยนต์ขึ้นมานี้มีวัตถุประสงค์หลักดังนี้

1. ช่วยให้ผู้ที่มีความรู้ด้านควบคุมหุ่นยนต์อยู่แล้วสามารถควบคุมได้ง่ายมากยิ่งขึ้น
2. บุคคลที่ไม่มีความรู้ด้านการควบคุมหุ่นยนต์ก็สามารถควบคุมได้โดยไม่ยาก
3. สามารถประหยัดเวลาในการเขียนโปรแกรมในการควบคุมหุ่นยนต์ได้
4. สามารถที่เป็นชุดทดลองให้เยาวชนได้เรียนรู้ในการควบคุมหุ่นยนต์

1.6 ขอบเขตของโครงงาน

project นี้จะแบ่งออกเป็น 2 ส่วน คือ

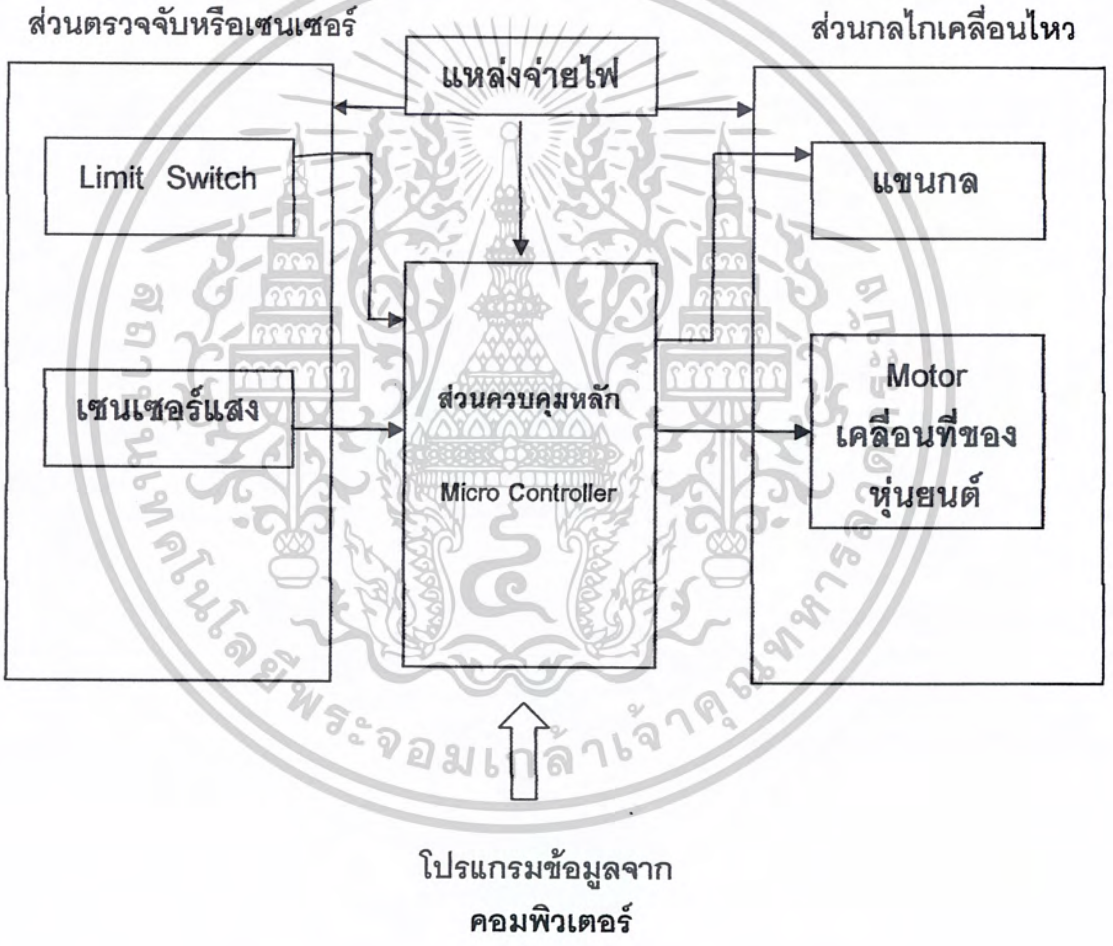
ส่วนที่ 1 เป็นการสร้างเครื่องจักรกลของตัวหุ่นยนต์และฮาร์ดแวร์ของหุ่นยนต์ ที่ประกอบด้วย

- DC Motor, Servo Motor, Limit Switch, Sensor
- วงจรไมโครคอนโทรลเลอร์เบอร์ P89C51RD2,89C51,89C2051,วงจร Drive Motor, LCD, Keypad

ส่วนที่ 2 เป็นการสร้างโปรแกรมที่ใช้ในการควบคุมหุ่นยนต์ ซึ่งสร้างขึ้นมาจากโปรแกรม VISUAL BASIC ลักษณะของโปรแกรมที่สร้างขึ้นมาจะเป็น Flow Chart พอเขียนเสร็จแล้วก็จะ Download เข้าไปสู่หุ่นยนต์ได้

บทที่ 2
ทฤษฎี

ในโครงงาน "ชุดพัฒนาระบบหุ่นยนต์" นี้ได้แบ่งส่วนประกอบของตัวหุ่นของเป็นส่วนต่างๆ ออกเป็น 2 ส่วนใหญ่ คือ ส่วนของฮาร์ดแวร์ และ ซอฟต์แวร์ โดย หลักการทำงานเบื้องต้นของหุ่นยนต์ที่ใช้ในวิทยานิพนธ์นี้ จะมีแผนผังการทำงานหลักๆดังนี้



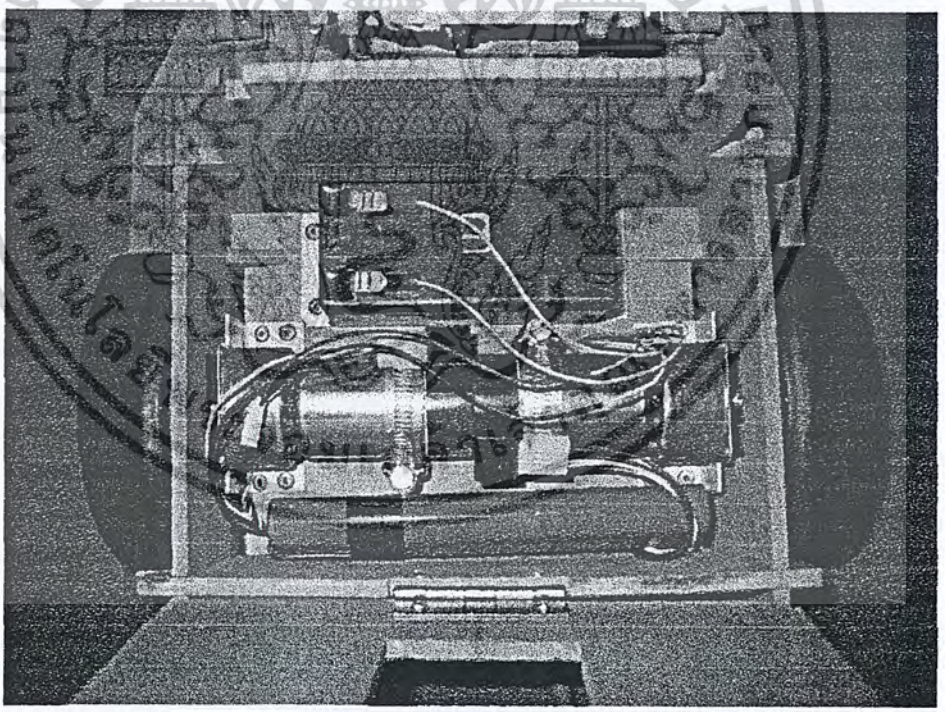
รูปที่ 2-1 ผังการทำงานโดยรวมเบื้องต้นของหุ่นยนต์อัตโนมัติ

2.1. กลไกเคลื่อนไหว

นับได้ว่าเป็นส่วนสำคัญที่ขาดไม่ได้สำหรับหุ่นยนต์อัตโนมัติ เพราะหุ่นยนต์ต้องมีการเคลื่อนไหว จะเคลื่อนไหวในบางส่วน หรือทั้งหมดก็ได้ ในกลไกเคลื่อนไหวจะมีส่วนประกอบย่อยที่สำคัญอีก 2 ส่วนคือ แหล่งกำเนิดการเคลื่อนไหวและกลไก

Project นี้ได้ใช้แหล่งกำเนิดการเคลื่อนไหวอยู่ 2 แบบคือ

2.1.1 DC Motor จะใช้ในส่วนของจุดหมุนตรงฐานของแขนกล จะใช้ DC Motor ในการกำเนิดการเคลื่อนที่ และ ส่วนแหล่งกำเนิดการเคลื่อนที่ของตัวหุ่นยนต์จะใช้ DC Motor ที่มี Encoder 2 ตัว ซ้ายและขวาข้างละตัว ซึ่งจุดเด่นในการเลือกใช้ Motor ชนิดนี้เพราะว่าสามารถนำสัญญาณของ Encoder มาทำ Feedback control ได้ทำให้เราสามารถควบคุมมอเตอร์ได้ดียิ่งขึ้น การเลือกซื้อมอเตอร์มีความสำคัญมากเพราะถ้าเราเลือกซื้อมอเตอร์ไม่เหมาะสมกับการใช้งานจะทำให้การทำงานของตัวหุ่นผิดพลาดหรือไม่สามารถทำงานได้เลย

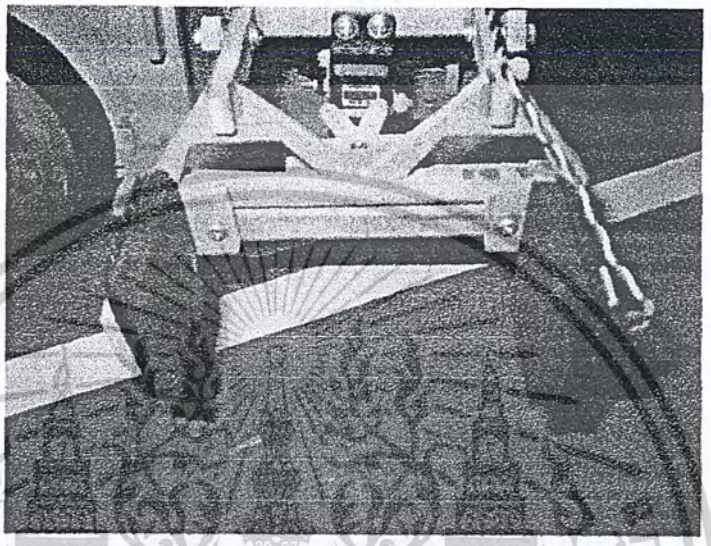


รูปที่ 2-2 รูปแสดงตัวมอเตอร์ที่ใช้ขับเคลื่อนของหุ่นยนต์

2.1.2 Servo Motor

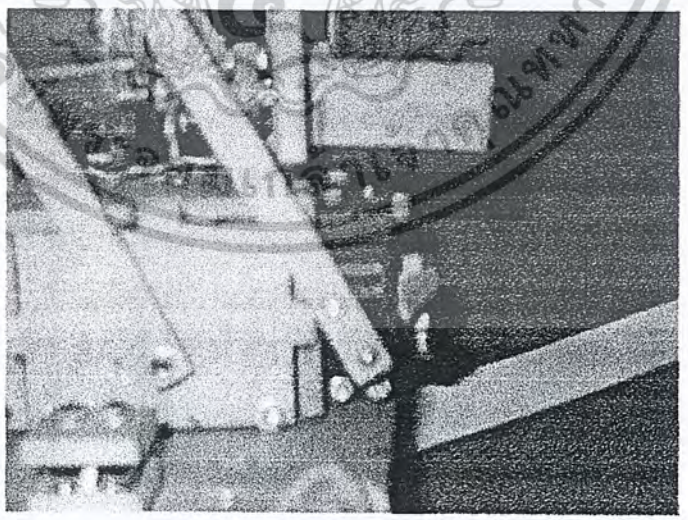
ในส่วนของแขนกลของหุ่นยนต์จะใช้ Servo Motor อยู่ 2 จุดคือ

มือจับวัตถุ ไว้สำหรับจับวัตถุเพื่อใช้ในการเคลื่อนย้ายวัตถุ



รูปที่ 2-3 รูปแสดงมือจับวัตถุของหุ่นยนต์โดยใช้เซอร์โวมอเตอร์ในการขับเคลื่อน

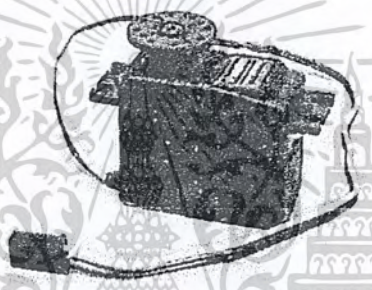
ชุดยกของแขน ไว้สำหรับยกวัตถุและเอื้อมมือเพื่อที่จะนำไปจับวัตถุ



รูปที่ 2-4 แสดงเซอร์โวมอเตอร์ที่ใช้ในการควบคุมแขน

เซอร์โวมอเตอร์ (Servo motor)

เซอร์โว เป็นมอเตอร์ทดเฟืองขนาดเล็กโดยมีการส่งกำลัง 1 อัน โดยปกติเซอร์โวจะหมุนได้เพียง 180 องศาเท่านั้นหรือบางตัวอาจจะถึง 210 องศาขึ้นอยู่กับบริษัทผู้ผลิตและการส่งกำลังนี้สามารถที่จะควบคุมทิศทางที่หันไปหรือตำแหน่งที่ต้องการได้ โดยการส่งรหัสควบคุมให้กับตัวเซอร์โว ตำแหน่งของแกนส่งกำลังจะอยู่มุมเดิมได้นั้นจะต้องมีการส่งสัญญาณดังกล่าวเข้าที่ขาอินพุทตลอดเวลา เซอร์โวสามารถนำไปประยุกต์ใช้งานกับอุปกรณ์ได้หลากหลาย ส่วนใหญ่เราจะเห็นอยู่กับเครื่องเล่นที่บังคับด้วยวิทยุทั้งหลายเช่น นำมาบังคับเลี้ยวและควบคุมอัตราวิ่งของรถบังคับวิทยุ , ควบคุมปีกของเครื่องบิน , หรือหางเสือเรือ และการเดินของหุ่นยนต์



รูปที่ 2-5 แสดงตัวเซอร์โวมอเตอร์

หลักการทำงานของเซอร์โวมอเตอร์

มอเตอร์ของเซอร์โวมีวงจรควบคุมการหมุนโดยใช้ตัวต้านทานปรับค่าได้หรือ VR ค่า 5 kΩ ตัวต้านทานนี้จะติดอยู่กับแกนส่งกำลัง เพื่อใช้วัดระยะองศาของแกนหมุนโดยใช้ร่วมกับวงจรควบคุมการกำหนดมุมของเซอร์โว ถ้าแกนอยู่ในมุมที่ถูกต้องมอเตอร์ก็จะปิดเองโดยอัตโนมัติ แต่ถ้าวงจรตรวจสอบพบว่ามุมยังไม่ถูกต้องมอเตอร์ก็จะหมุนไปในทางที่ถูกจนกระทั่งได้มุมที่ถูกต้องและจะหยุดอยู่ตรงนั้นตลอดไปจนกว่าจะมีการป้อนเข้ามาใหม่ ตัวแกนของเซอร์โวนี้สามารถหมุนได้เพียง 180 – 210 องศา (เนื่องจาก VR หมุนได้เท่านี้) และสามารถสั่งให้เซอร์โวทำการกำหนดองศาของการหมุนได้โดยการส่งสัญญาณพัลส์ค่าระหว่าง 1.5 มิลิวินาที ถึง 2 มิลลิวินาที

จากรูปจะสรุปได้ว่าถ้าเราส่งพัลส์ที่มีขนาด 1.5 มิลลิวินาที จะทำให้เซอร์โวปรับหมุนไปที่ตำแหน่งกึ่งกลางของแกนหรือที่ 90 องศา ส่วนถ้าป้อนสัญญาณพัลส์ที่มีค่าน้อยกว่า 1.5 มิลลิวินาทีเซอร์โวจะหมุนไปทางซ้ายหรือเข้าสู่ 0 องศา และถ้าป้อนสัญญาณพัลส์มากกว่า 1.5 มิลลิวินาที ก็จะทำให้เซอร์โวหมุนไปทางขวาหรือเข้าสู่องศาที่ 180 นั่นเอง สัญญาณพัลส์ที่ส่งก็คือ

สัญญาณ pulse width modulation (PWM) ซึ่งใน project นี้ ผมจะใช้ไทเมอร์ของไมโครคอนโทลเลอร์เป็นตัวสร้างพัลส์ให้แก่เซอร์โวมอเตอร์

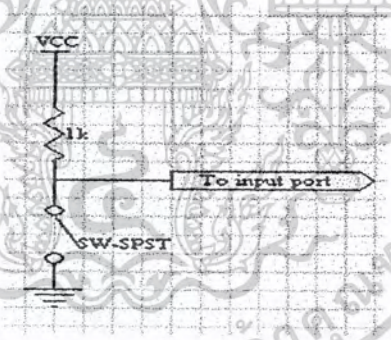
2.2. ส่วนตรวจจับหรือเซนเซอร์

อาจกล่าวได้ว่าส่วนนี้มีความสำคัญยิ่งยวด หุ่นยนต์จะทำงานได้ถูกต้องหรือไม่จะขึ้นอยู่กับการทำงานของอุปกรณ์ตรวจจับนี้ เพราะในการทำงานของหุ่นยนต์จะต้องรับสัญญาณอินพุตมากำหนดเงื่อนไขในการทำงาน เช่น ให้เดินตามเส้น ก็ต้องมีตัวจับเส้น หากตัวตรวจจับเส้นไม่ทำงานหรือหรือทำงานผิดพลาด หุ่นยนต์ก็จะไม่ทำงานหรือทำงานผิดพลาดตามไปด้วย

หน้าที่ของอุปกรณ์ตรวจจับคือ ทำการตรวจจับสัญญาณหรือการเปลี่ยนแปลงของปริมาณทางวิทยาศาสตร์ แล้วรายงานให้ส่วนควบคุมรับทราบ อาทิ ในหุ่นยนต์เดินตามแสง ตัวตรวจจับแสงต้องทำหน้าที่ตรวจจับว่ามีแสงเกิดขึ้นหรือไม่ หรือวัดความเข้ม แล้วส่งผลการตรวจจับนั้นไปให้ส่วนควบคุม เพื่อตัดสินใจว่า จะขับเคลื่อนหุ่นยนต์ไปตามแสงหรือไม่ หรือในหุ่นยนต์เดินตามเส้น ตัวตรวจจับเส้นก็ต้องคอยตรวจจับว่า ขณะนี้ยังสามารถตรวจจับพบเส้นที่ใช้ในการเคลื่อนที่หรือไม่ แล้วส่งสัญญาณไปแจ้งส่วนควบคุมเพื่อขับเคลื่อนหุ่นยนต์ให้อยู่บนเส้นที่กำหนดให้ได้ เป็นต้น

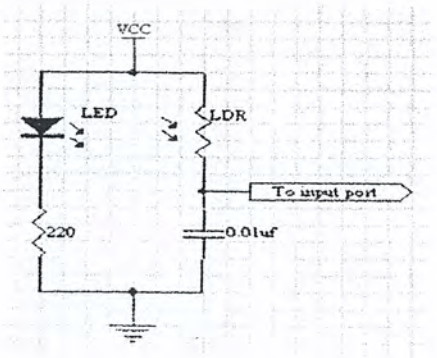
Project นี้ได้ใช้ตัวตรวจจับหรือเซนเซอร์อยู่ 2 แบบคือ

2.2.1 Limit switch



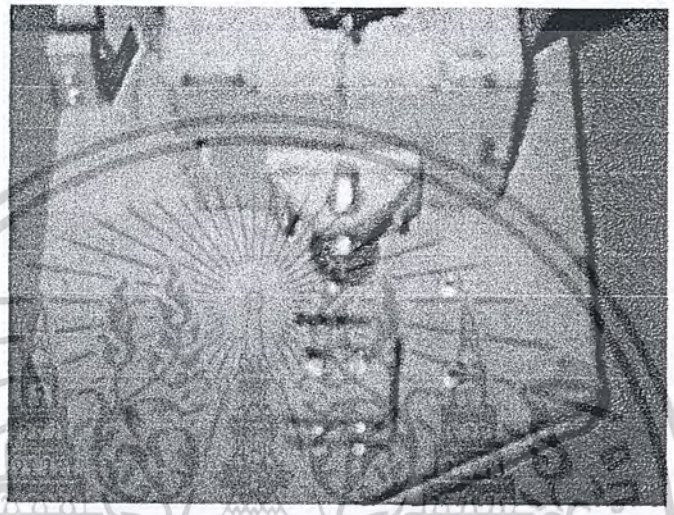
รูปที่ 2-6 วงจร Limit Switch ที่ใช้ในหุ่นยนต์

2.2.2 เซนเซอร์แสง



รูปที่ 2-7 วงจรเซนเซอร์ที่ใช้ในหุ่นยนต์

- Limit switch ในหุ่นยนต์ project นี้มี 4 ตัว โดย 3 ตัวมีหน้าที่เพื่อตรวจจับตำแหน่งการหมุนของแขนกลให้มีตำแหน่งให้แขนกลหยุดอยู่ที่ตรงกลางหมุนขวา 90 องศา และหมุนซ้าย 90 องศา จะติดไว้ที่ส่วนฐานของแขนหุ่น ส่วนอีกตัว จะติดไว้ที่มือของหุ่น มีหน้าที่เพื่อใช้ในการตรวจจับว่าได้ทำการจับวัตถุแล้วเพื่อไม่ทำให้บีบวัตถุแรงเกินไปเพราะอาจทำให้วัตถุเสียหายได้



รูปที่ 2-8 รูปแสดงลิมิตสวิตช์ของหุ่นยนต์เพื่อตรวจจับตำแหน่งของแขนหุ่น

รูปที่ 2-9 รูปแสดงลิมิตสวิตช์ที่ติดอยู่ที่มือจับของหุ่น

- **เซนเซอร์แสง** เซนเซอร์แสงที่ออกแบบใน project นี้จะเป็นเซนเซอร์ที่ใช้การ charge และการ discharge ของประจุ C แล้วใช้ไมโครคอนโทรลเลอร์ในการนับค่าเวลาการ charge ของประจุ C โดยใช้หลักการของสูตร $V_0 = V_{in}\, e^{(-t/RC)}$

Vin คือ ค่าแรงดันอินพุต(5V)

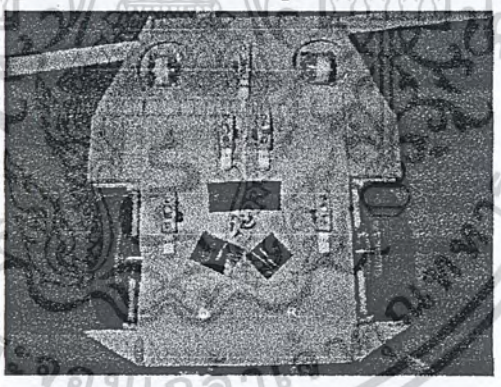
Vo คือ ค่าแรงดันที่ขา Port ไมโครคอนโทรลเลอร์

C คือ ค่าประจุ C ในวงจรเซนเซอร์

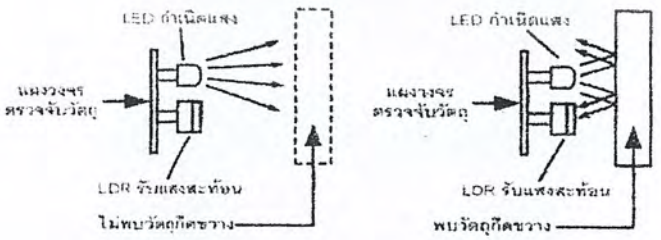
R คือ ค่าความต้านทานของ LDR

t คือ คาบเวลาในการวนลูปนับค่าของโปรแกรม

จะสังเกตความสัมพันธ์ของสูตรคือ ถ้า ค่า C เรากำหนดให้คงที่ส่วน R ในสูตรคือ LDR ในวงจร ซึ่งจะแปรเปลี่ยนไปตามความเข้มของแสง เพราะฉะนั้นค่า t ซึ่งเป็นค่าคาบเวลาการนับของโปรแกรมก็จะมีค่าที่ไม่เท่ากัน ยกตัวอย่าง ถ้าเขียนโปรแกรมวนลูป 00H ไปจนถึง FFH ถ้าวางเซนเซอร์อยู่บนพื้นสีขาวแล้วใช้เวลาในการ charge ประจุเท่ากับ 05H แต่ถ้าวางบนพื้นสีดำจะใช้เวลาในการ charge ประจุเท่ากับ FAH หลังจากนั้นก็นำมาบวกกันหารสองก็จะได้ค่า reference ของสองสี ดังนั้นก็จะสามารถรับรู้ได้ว่าขณะนี้เซนเซอร์อยู่บนพื้นสีดำ หรือพื้นสีขาว



รูปที่ 2-10 รูปแสดงการวางตำแหน่งเซนเซอร์ทั้ง 5 ตัวของหุ่น



รูปที่ 2-11 รูปแสดงหลักการทำงานของเซนเซอร์

2.3. ส่วนควบคุม

ในหุ่นยนต์อัตโนมัติสมัยใหม่จะใช้อุปกรณ์อิเล็กทรอนิกส์ที่เรียกว่า "ไมโครคอนโทรลเลอร์" (microcontroller) เป็นอุปกรณ์หลักในการควบคุมและประมวลผล โดยในส่วนควบคุมนี้จะบรรจุโปรแกรมควบคุมที่ผู้ใช้งานเขียนขึ้นลงในหน่วยความจำ และสามารถเปลี่ยนแปลงแก้ไขได้ โดยส่วนควบคุมนี้จะมีส่วนเชื่อมต่ออุปกรณ์ภายนอก 2 ลักษณะคือ พอร์ตอินพุต (input port) และพอร์ตเอาต์พุต (output port)

ใน project นี้ได้ใช้**ไมโครคอนโทรลเลอร์ ถึง 4 ตัว**ในการควบคุมส่วนต่างๆของหุ่นยนต์ดังนี้

2.3.1 P89C51RD2 เป็นไมโครคอนโทรลเลอร์ตัวหลักที่จะคอยทำงานในส่วนของการ Download โปรแกรมใหม่ๆที่สร้างขึ้นมา เพราะว่าไมโครคอนโทรลเลอร์เบอร์นี้มีคุณสมบัติในการ Download โปรแกรมเข้าสู่ในคอนโทลเลอร์ได้เลยเพราะว่าในคอนโทลเลอร์เบอร์นี้จะมีระบบ ISP(In-System Programming) อีกทั้งในพอร์ตต่างๆยังทำงาน ในการรับสัญญาณเซนเซอร์,รับสัญญาณลิมิตสวิตช์, เป็นตัวสร้างพัลส์ส่งสัญญาณพัลส์ไปให้แก่เซอร์โวมอเตอร์, ส่งสัญญาณไปติดต่อกับไมโครคอนโทรลเลอร์ 89C2051 ซึ่งเป็นส่วนที่ติดต่อกับการทำงานของมอเตอร์ที่ใช้เคลื่อนที่ของหุ่นยนต์ และยังติดต่อกับไมโครคอนโทรลเลอร์ 89C51ในส่วนของการรับส่งข้อมูลของ Keypad และการแสดงผลของจอ LCD

คุณสมบัติการโปรแกรมในระบบหรือ ISP (In-System Programming)

จุดเด่นที่ไมโครคอนโทรลเลอร์ P89C51RD2 มีเหนือไมโครคอนโทรลเลอร์ตระกูล MCS-51 อีกประการหนึ่งคือ ความสามารถในการเขียน-อ่าน-ลบข้อมูลในหน่วยความจำโปรแกรมในระบบหรือในวงจรโดยไม่ต้องถอดตัวไมโครคอนโทรลเลอร์ออกมาโปรแกรมด้วยเครื่องโปรแกรมภายนอก หรือที่เรียกย่อๆว่า ISP (In-System Programming) การโปรแกรมแบบ ISP ของไมโครคอนโทรลเลอร์ P89C51RD2 แตกต่างจากอนุกรม AT89Sxxxx ตรงที่การถ่ายทอดข้อมูลจะกระทำผ่านขาพอร์ตอนุกรมคือ TxD และ RxD จึงทำให้ไม่ต้องใช้วงจรรับข้อมูลจากคอมพิวเตอร์แล้วถ่ายทอดข้อมูลในแบบ SPI (Synchronous Peripheral Interface) เพียงต่อขาสัญญาณ จากไมโครคอนโทรลเลอร์ผ่านวงจรแปลงระดับสัญญาณ RS-232 ก็ใช้ได้แล้ว

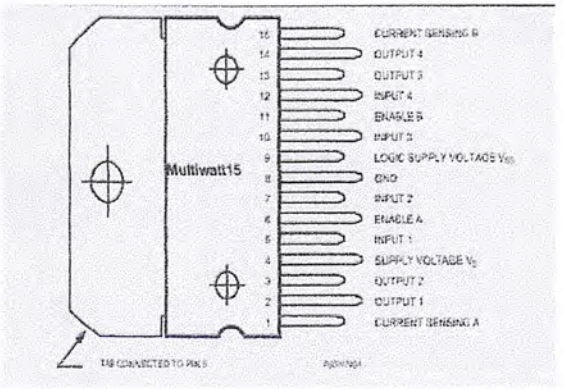
หัวใจสำคัญของกระบวนการ ISP ในไมโครคอนโทรลเลอร์ P89C51RD2 คือ หน่วยความจำบูตรอมที่บรรจุโปรแกรมสำหรับอ่าน-เขียนหน่วยความจำโปรแกรมภายในไมโครคอนโทรลเลอร์โดยปกติจะไม่สามารถเข้าถึงได้เว้นแต่ผู้ใช้งานต้องการให้ไมโครคอนโทรลเลอร์เข้าสู่โหมดการโปรแกรมแบบ ISP การเข้าสู่โหมดโปรแกรมของไมโครคอนโทรลเลอร์ P89C51RD เล็กน้อย ดังมีรายละเอียดต่อไปนี้

<u>การเข้าสู่โหมดโปรแกรมแบบ ISP ของไมโครคอนโทรลเลอร์ P89C51RD2</u>

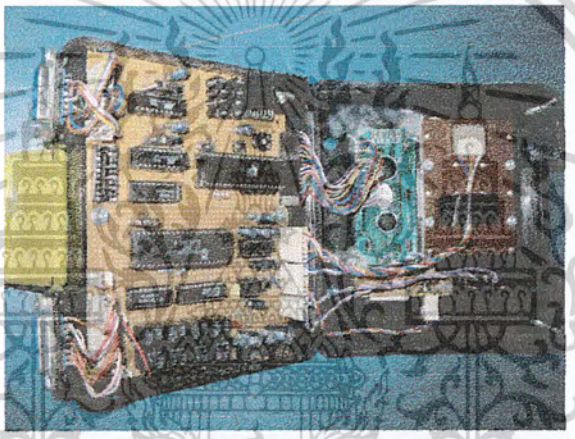
1. ต่อขา PSEN ลงกราวด์
2. จ่ายไฟ +5V เข้าที่ขา EA/Vpp
3. ป้อนลอจิก "1" เข้าที่ขา P2.7
4. รีเซตไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์จะเข้าสู่โหมดโปรแกรมแบบ ISP.
5. หลังจากที่ติดต่อกับหน่วยความจำโปรแกรมเรียบร้อยแล้ว ให้ต่อ PSEN กลับไปยังลอจิก"1"จากนั้นทำการรีเซตไมโครคอนโทรลเลอร์อีกครั้ง ขณะนี้ไมโครคอนโทรลเลอร์กลับมาทำงานในโหมดรันปกติแล้ว

2.3.2 89C51 ไมโครคอนโทรลเลอร์เบอร์นี้ในส่วนของการควบคุมจะทำหน้าที่ในการรับค่าของ Keypad และแสดงสถานะต่างๆของหุ่นยนต์แสดงออกจอ LCD อีกทั้งยังมีการรับ – ส่งข้อมูลกับไมโครคอนโทรลเลอร์ P89C51RD2 อีกด้วย

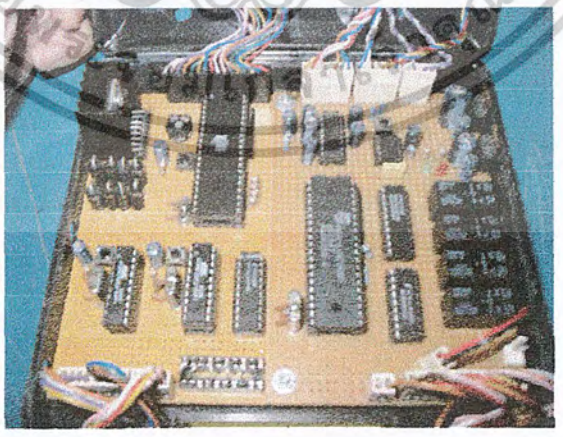
2.3.3 89C2051 ไมโครคอนโทรลเลอร์เบอร์นี้ใช้ถึง 2 ตัวในส่วนของการควบคุมเพราะ 89C2051 แต่ละตัวจะมีหน้าที่ในการควบคุมมอเตอร์ที่ใช้เคลื่อนที่ในแต่ละข้างมันจะรับข้อมูลมาจาก ไมโครคอนโทรลเลอร์P89C51RD2 พอรับข้อมูลเสร็จ จากนั้นมันก็จะสร้างพัลส์ให้แก่มอเตอร์แต่ละข้างและสัญญาณพัลส์ที่เกิดจาก Encoder ของมอเตอร์แต่ละข้างนั้น เคานท์เตอร์ของไมโครคอนโทรลเลอร์ก็จะเก็บค่าพัลส์แล้วนำไปประมวลผล ซึ่งก่อนที่พัลส์จะส่งไปถึงมอเตอร์จะต้องผ่านวงจรขับมอเตอร์ก่อน ซึ่งใน project นี้ได้ใช้ IC L298 เป็นตัวขับมอเตอร์เพราะใช้ง่ายและสามารถทนกระแสของมอเตอร์ได้ตามที่ต้องการ



รูปที่ 2-12 IC L298 ที่ใช้ในการขับ DC Motor ของหุ่นยนต์



รูปที่ 2-13 รูปแสดงวงจรไมโครคอนโทรลเลอร์ทั้งหมดที่ใส่ไว้ในกล่อง



รูปที่ 2-14 รูปแสดงไมโครคอนโทรลเลอร์ชนิดต่างๆ

2.3.4 การทำงานแบบต่าง ๆของไมโครคอนโทรลเลอร์ที่ใช้ในโปรเจกต์

ใน project นี้การใช้ไทเมอร์ และเคาท์เตอร์ของไมโครคอนโทลเลอร์นั้นเป็นส่วนที่สำคัญมากเพราะต้องนำไปใช้ในการสร้างพัลส์ในการควบคุมเซอร์โวมอเตอร์ การปรับความเร็วของ DC Motor การรับสัญญาณพัลส์ Encoder จาก DC Motor การสร้างโปรแกรม charge และ discharge ของเซนเซอร์ รวมๆแล้วแทบจะต้องใช้ไทเมอร์ เคาท์เตอร์ของไมโครคอนโทรลเลอร์

2.3.4.1 ไทเมอร์/เคาน์เตอร์ของไมโครคอนโทรลเลอร์ MCS-51

ไทเมอร์/เคาน์เตอร์ (Timer/Counter) เป็นอีกหนึ่งส่วนประกอบที่สำคัญของไมโครคอนโทรลเลอร์ เนื่องจากในการทำงานของไมโครคอนโทรลเลอร์จะต้องมีการเก็บและตรวจสอบค่าของเวลาและจำนวนของสัญญาณนาฬิกาอยู่ตลอดเวลา ทั้งนี้เพื่อประโยชน์ในการสร้างฐานเวลา สร้างสัญญาณพัลส์ เปรียบเทียบค่าเวลา หรือเปรียบเทียบค่าของการนับ รวมไปถึงการกำหนดอัตราเร็วในการสื่อสารข้อมูลของพอร์ตอนุกรมด้วย

การทำงานเป็นไทเมอร์

เมื่อกำหนดให้ทำงานเป็นตัวตั้งเวลาหรือไทเมอร์ ค่าของรีจิสเตอร์จะเพิ่มขึ้นในทุกๆ แมทชีนไซเกิล ดังนั้นเมื่อทำงานเป็นไทเมอร์ รีจิสเตอร์จะทำการนับค่าของแมทชีนไซเกิลนั่นเอง และเนื่องจากแมทชีนไซเกิลประกอบด้วยคาบเวลาของวงจรกำเนิดสัญญาณนาฬิกา 12 คาบเวลา ดังนั้นอัตราในการนับของรีจิสเตอร์จึงเท่ากับ 1/12 ของความถี่สัญญาณนาฬิกา

การทำงานเป็นเคาน์เตอร์

เมื่อทำงานเป็นตัวนับหรือเคาน์เตอร์ค่าของรีจิสเตอร์จะเพิ่มขึ้นก็ต่อเมื่อมีการเปลี่ยนแปลงของระดับลอจิกจาก “1” เป็น “0” เกิดขึ้นที่ขาอินพุตทางฮาร์ดแวร์ของวงจรไทเมอร์/เคาน์เตอร์ ซึ่งก็คือขา T0 และ ขา T1 สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C51 รวมทั้งขา T2 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และอนุกรม AT89Sxx โดยจะมีการสุ่มรับสัญญาณจากขาอินพุตในทุกๆคาบเวลาที่2ของสเตตที่ 5 (S5P2) ในแต่ละแมชีนไซเกิล

เมื่อสัญญาณอินพุตเปลี่ยนแปลงจาก “1” เป็น “0” เป็นเวลาหนึ่งไซเกิล ในไซเกิลต่อมาค่าของการนับจะเพิ่มขึ้นหนึ่งค่า และจะไปปรากฏในรีจิสเตอร์ภายในคาบเวลาที่ 1 ของสเตตที่3 (S3P1) ของแมชีนไซเกิลต่อไปหลังจากที่ตรวจจับพบการเปลี่ยนแปลงที่ขาไทเมอร์อินพุตแล้ว เมื่อเป็นเช่นนี้ในกระบวนการการตรวจจับการเปลี่ยนแปลงของสัญญาณอินพุตที่ขาไทเมอร์จะต้องใช้ 2 แมทชีนไซเกิล อัตราการนับของเคาน์เตอร์จึงเท่ากับ 1/24 ของความถี่สัญญาณนาฬิกา ดังนั้น

ความถี่สูงสุดของสัญญาณอินพุตที่ไทเมอร์/เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถตรวจจับได้จึงเท่ากับความถี่ของของสัญญาณนาฬิกาหารด้วย 24 ยกตัวอย่าง ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C51 สามารถใช้สัญญาณนาฬิกาได้สูงสุด 24 MHz ดังนั้นความถี่สูงสุดของสัญญาณอินพุตที่ไทเมอร์/เคาน์เตอร์สามารถตรวจจับได้คือ 1 MHz

2.3.4.2 รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของไทเมอร์/เคาน์เตอร์ 0 และ 1

ในการทำงานของไทเมอร์/เคาน์เตอร์ 0 และ 1 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช มีรีจิสเตอร์ที่ต้องเกี่ยวข้องเป็นพื้นฐานอยู่ 6 ตัว ดังมีรายละเอียดต่อไปนี้

รีจิสเตอร์ไทเมอร์

มีด้วยกัน 4 ตัวคือ TL0 มีแอดเดรสอยู่ที่ 8AH,TH0 มีแอดเดรสอยู่ที่ 8CH,TL1 มีแอดเดรสอยู่ที่ 8BH และ TH1 มีแอดเดรสอยู่ที่ 8 DH รีจิสเตอร์ทั้ง 4 ตัวจะอยู่ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR รีจิสเตอร์แต่ละตัวมีขนาด 8 บิต แต่ในการใช้งานโดยทั่วไปมักใช้ร่วมกันโดยจัดเป็นคู่คือ TL0 กับ TH0 รวมกันเป็นรีจิสเตอร์ Timer 0 ขนาด 16 บิต และ TL1 กับ TH1 รวมกันเป็นรีจิสเตอร์ Timer 1 ขนาด 16 บิต โดยใน TL0 และ TL1 เก็บข้อมูล 8 บิตล่าง ส่วน TH0 และ TH1 เก็บข้อมูล 8 บิตบน รีจิสเตอร์ไทเมอร์ทั้ง 2 คู่ เมื่อนำมาใช้ร่วมกันจะสามารถเก็บค่าของการนับได้สูงสุด 65,536 หรือ FFFFH เมื่อนับถึงค่านี้แล้วจะวนไปเริ่มนับ 0000H ใหม่ และเมื่อเกิดการนับรอบใหม่ บิต TF0 หรือ TF1 ในรีจิสเตอร์ TCON ที่ใช้ควบคุมการทำงานของไทเมอร์จะเกิดการเซต เพื่อแจ้งให้ทราบว่า นับเกินค่าสูงสุดแล้ว การเซตบิต TF0 หรือ TF1 ขึ้นอยู่กับว่าเลือกใช้งานรีจิสเตอร์ไทเมอร์ตัวใด

รีจิสเตอร์ควบคุมการทำงานของไทเมอร์/เคาน์เตอร์หรือ TCON (Timer/Counter Control register)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 88H ในพื้นที่ของรีจิสเตอร์ SFR สามารถเข้าถึงได้ในระดับบิตมีรายระเอียดการทำงานดังนี้

บิต0	บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 (Timer 1 overflow flag): เซตด้วยกระบวนการทางฮาร์ดแวร์ เมื่อค่าของรีจิสเตอร์ Timer 1 เกิดการนับเกินหรือเกิดโอเวอร์โฟลว์ การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางฮาร์ดแวร์เช่นกัน โดยบิตนี้จะเคลียร์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น

TR1 (Timer 1 run control bit): ใช้ในการเปิดปิดการทำงานของไทเมอร์ 1 (เอ็นเอเบิลหรือดิสเอเบิล)ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ไทเมอร์ 1 ทำงานต้องเซตบิตนี้ให้เป็น "1"

TF0 (Timer 0 overflow flag): เซตด้วยกระบวนการทางฮาร์ดแวร์ เมื่อค่าของรีจิสเตอร์ Timer 0 เกิดการนับเกินหรือเกิดโอเวอร์โฟลว์ การเคลียร์บิตนี้ทำได้ด้วยกระบวนการทางฮาร์ดแวร์เช่นกัน โดยบิตนี้จะเคลียร์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น

TR0 (Timer 0 run control bit): ใช้ในการเปิดปิดการทำงานของไทเมอร์ 0 (เอ็นเอเบิลหรือดิสเอเบิล) ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการให้ไทเมอร์ 0 ทำงานต้องเซตบิตนี้ให้เป็น "1"

IE1 (External Interrupt 1 edge flag): บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณอินเตอร์รัปต์จากภายนอกที่ขาอินพุตอินเตอร์รัปต์ 1 (INT1) ได้และจะทำการเคลียร์เมื่อมีการบริการอินเตอร์รัปต์เกิดขึ้น

IT1 (Interrupt 1 type control bit): บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ โดยใช้ในการเลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุตอินเตอร์รัปต์ 1 (INT1) การเซตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์

"0" เลือกขอบขาลงของสัญญาณ (falling edge)

"1" เลือกระดับลอจิกต่ำ (low level triggered)

IE0 (External Interrupt 0 edge flag): บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อสามารถตรวจจับขอบขาของสัญญาณอินเตอร์รัปต์จากภายนอกที่ขาอินพุตอินเตอร์รัปต์ 0 (INT0) ได้ และจะทำการเคลียร์เมื่อมีการบริการอินเตอร์รัปต์เกิดขึ้น

IT0 (Interrupt 0 type control bit) : บิตนี้จะใช้ในกระบวนการอินเตอร์รัปต์ โดยใช้ในการเลือกลักษณะของสัญญาณอินเตอร์รัปต์จากภายนอกที่ต้องการให้ทำการตอบสนองสำหรับขาอินพุตอินเตอร์รัปต์ 0 (INT0) การเซตและเคลียร์ทำได้ด้วยกระบวนการซอฟต์แวร์

"0" เลือกขอบขาลงของสัญญาณ (falling edge)

"1" เลือกระดับลอจิกต่ำ (low level triggered)

รีจิสเตอร์เลือกโหมดการทำงานของไทเมอร์/เคาน์เตอร์หรือTMOD (Timer/Counter Mode Control Register)

เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 89H ในพื้นที่ของรีจิสเตอร์ SFR ไม่สามารถเข้าถึงได้ในระดับบิต แบ่งการทำงานออกเป็น 2 ส่วน คือ 4 บิตล่าง ใช้ในการเลือกโหมดการทำงานของไทเมอร์ 0 และ 4 บิตบนใช้ในการเลือกโหมดการทำงานของไทเมอร์ 1 ดังนั้นในการอธิบายการทำงานจะขออธิบายเพียงส่วนเดียวดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
GATE	C/T	M1	M0	GATE	C/T	M1	M0
ไทเมอร์ 1				ไทเมอร์ 0			

GATE : ใช้เลือกลักษณะการควบคุมการทำงานของไทเมอร์/เคาน์เตอร์

"0" ไทเมอร์/เคาน์เตอร์จะทำงานเมื่อบิต TRx ในรีจิสเตอร์ TCON เป็น "1" เรียกการควบคุมแบบนี้ว่าการควบคุมทางซอฟต์แวร์

"1" ไทเมอร์/เคาน์เตอร์จะทำงานเมื่อบิต TRx ในรีจิสเตอร์ TCON เป็น "1" และสถานะลอจิกที่ขาอินพุตอินเตอร์รัปต์ INT0 และ INT1 เป็น "1" เรียกการควบคุมแบบนี้ว่า การควบคุมทางฮาร์ดแวร์

C/T (Timer or Counter selecter) : ใช้เลือกลักษณะการทำงานของไทเมอร์/เคาน์เตอร์

"0" เลือกให้ทำงานเป็นไทเมอร์ โดยใช้สัญญาณอินพุตจากสัญญาณนาฬิกาภายในไมโครคอนโทรลเลอร์

"1" เลือกให้ทำงานเป็น เคาน์เตอร์ โดยรับสัญญาณ อินพุตทางขา T0 หรือ T1

M1,M0 (Mode selector bit) : ใช้เลือกโหมดการทำงานของไทเมอร์/เคาน์เตอร์

"00" เลือกให้ทำงานในโหมด ไทเมอร์/เคาน์เตอร์ 13 บิต

"01" เลือกให้ทำงานในโหมด ไทเมอร์/เคาน์เตอร์ 16 บิต

"10" เลือกให้ทำงานในโหมด ไทเมอร์/เคาน์เตอร์ขนาด 8 บิต แบบตั้งค่ากึ่งอัตโนมัติ

"11" สำหรับไทเมอร์ 0 เลือกให้ทำงานในโหมดไทเมอร์/เคาน์เตอร์แยกส่วน โดยแยกเป็นไทเมอร์/เคาน์เตอร์ 8 บิต 2 ตัว รีจิสเตอร์ TL0 จะได้รับการควบคุมการเปิดปิดจากบิต TR0 ในรีจิสเตอร์ TCON และรีจิสเตอร์ TH0 ซึ่งเป็นไทเมอร์/เคาน์เตอร์ 8 บิต อีกตัวหนึ่ง จะได้รับการ

ควบคุมจากบิต TR1 ในรีจิสเตอร์ TCON ในกรณีของไทเมอร์ 1 เป็นการสั่งให้ไทเมอร์/เคาน์เตอร์ 1 หยุดทำงาน (ดิสเอเบิล)

โหมดการทำงานของไทเมอร์/เคาน์เตอร์ 0 และ 1

ไทเมอร์ 0 และไทเมอร์ 1 สามารถเลือกโหมดการทำงานได้ 4 โหมดคือ โหมด 0 : ไทเมอร์/เคาน์เตอร์ 13บิต (13bit timer/counter), โหมด 1 : ไทเมอร์/เคาน์เตอร์ 16บิต (16 bit timer/counter), โหมด 2 : ตั้งค่าอัตโนมัติขนาด 8 บิต (8 bit auto-reload timer/counter) และ โหมด 3 : ไทเมอร์/เคาน์เตอร์แยกส่วน (split timer/counter) หรือ อาจเรียกว่าโหมดไทเมอร์/เคาน์เตอร์ 8 บิต ก็ได้ ในขณะที่ไทเมอร์ 2 มีโหมดการทำงาน 3 โหมดคือ โหมดแคปเจอร์หรือตรวจจับสัญญาณ (capture), โหมดตั้งค่าอัตโนมัติ (auto-reload) และโหมดกำเนิดอัตราเร็วในการสื่อสารข้อมูลอนุกรมหรืออัตราบอด (baud rate generator)

การเลือกโหมดการทำงานของไทเมอร์/เคาน์เตอร์ 0 และ 1 สามารถทำได้ที่รีจิสเตอร์ TCON และ TMOD ร่วมกัน โดย TCON ใช้ในการเอ็นเอเบิลหรือดิสเอเบิลไทเมอร์/เคาน์เตอร์ ส่วน TMOD ใช้ในการเลือกโหมดและลักษณะการทำงาน ในขณะที่การทำงานของไทเมอร์ 2 จะอธิบายแยกต่างหาก

การทำงานในโหมด 1 : ไทเมอร์/เคาน์เตอร์ 16 บิต (ใช้ใน project นี้)

มีไดอะแกรมการทำงานแสดงในรูปที่ 6-2 ในที่นี้จะใช้ไทเมอร์ 1 ในการอธิบาย การทำงานในโหมดนี้จะคล้ายกับโหมด 0 แต่จะใช้งานรีจิสเตอร์ TL1 และ TH1 ครบ 8 บิตดังนั้นในโหมดนี้ค่าของการนับจะมีขนาด 16 บิต คือ 0000H-FFFFH เมื่อทำการนับครบรอบ ค่าของการนับเปลี่ยนจาก FFFFH เป็น 0000H ก็จะเซตบิต TF1 ในรีจิสเตอร์ TCON ส่วนการทำงานในโหมดนี้ของไทเมอร์ 0 มีลักษณะเหมือนกันทุกประการ เพียงแต่เปลี่ยนรีจิสเตอร์และขาสัญญาณที่เกี่ยวข้องให้เป็นของไทเมอร์ 0

โหมด	ฟังก์ชั่นของ ไทเมอร์ 0	TMOD	
		การควบคุม จากภายใน	การควบคุม จากภายนอก
0	ไทเมอร์/เคาน์เตอร์ 13 บิต	00H	08H
1	ไทเมอร์/เคาน์เตอร์ 16 บิต	01H	09H
2	8 บิตตั้งค่าอัตโนมัติ	02H	0AH
3	ไทเมอร์/เคาน์เตอร์แยกส่วน	03H	0BH

รูปที่ 2-15 ตารางแสดงฟังก์ชั่นและ TMOD ของไทเมอร์ 0

2.3.4.3 กระบวนการอินเตอร์รัปต์ของ ไมโครคอนโทรลเลอร์ MCS-51

การอินเตอร์รัปต์ (interrupt) เป็นชื่อเรียกกระบวนการที่เข้ามาขัดจังหวะการทำงานโดยปกติของไมโครคอนโทรลเลอร์ ในไมโครคอนโทรลเลอร์ MCS-51 สามารถตอบสนองการอินเตอร์รัปต์ที่เกิดขึ้นได้จาก 5 แหล่ง กำเนิดสำหรับเบอร์ AT89C51 ประกอบด้วย การรับสัญญาณอินเตอร์รัปต์จากภายนอกผ่านทางขา INT0 และ INT1 , สัญญาณอินเตอร์รัปต์จากไทเมอร์/เคาน์เตอร์ T0 และ T1 และ สัญญาณอินเตอร์รัปต์จากพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ ในขณะที่ไมโครคอนโทรลเลอร์ เบอร์ AT89C52 และในอนุกรม AT89Sxx สามารถตอบสนองการอินเตอร์รัปต์ได้จาก 6 แหล่งกำเนิด โดยเพิ่มการรับสัญญาณอินเตอร์รัปต์จากไทเมอร์/เคาน์เตอร์ 2 อีกแหล่งกำเนิด

การจัดการอิเตอร์รัปต์ในไมโครคอนโทรลเลอร์ MCS-51

เมื่อมีการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ MCS-51 เกิดขึ้น และมีการเอ็นเอเบิลการตอบสนอง การอินเตอร์รัปต์ไว้ กระบวนการหลังจากนั้นซีพียูจะทำการกระโดไปยังแอดเดรสในหน่วยความจำที่กำหนดไว้เรียกตำแหน่งแอดเดรสนี้ว่า แอดเดรสอินเตอร์รัปต์เวกเตอร์ (interrupt vector address) ดังนั้นจะต้องมีการเขียนโปรแกรมย่อยการบริการอินเตอร์รัปต์ไว้ที่แอดเดรสอินเตอร์รัปต์เวกเตอร์นี้ โดยค่าของแอดเดรสอินเตอร์รัปต์เวกเตอร์จะแตกต่างกันไปในการอินเตอร์รัปต์แบบต่างๆ ดังมีรายละเอียดต่อไปนี้

การอินเตอร์รัปต์ภายนอกที่ขา INT0 มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 003H

การอินเตอร์รัปต์จากไทเมอร์ 0 มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 000BH

การอินเตอร์รัปต์ภายนอกที่ขา INT1 มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 0013H

การอินเตอร์รัปต์จากไทเมอร์ 1 มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 001BH

การอินเตอร์รัปต์จากพอร์ตอนุกรม มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 0023H

การอินเตอร์รัปต์จากไทเมอร์ 2 มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 002BH

2.3.4.4 รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ MCS-51

การอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช มีรีจิสเตอร์ที่ต้องเกี่ยวข้องอยู่ 2 ตัว ดังมีรายละเอียดดังต่อไปนี้

รีจิสเตอร์เอ็นเอเบิลการอินเตอร์รัปต์หรือ IE (Interrupt Enable register)

มีแอดเดรสอยู่ที่ A8H ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต สามารถเข้าถึงได้ในระดับบบิต ใช้ในการเอ็นเอเบิลการตอบสนองการอินเตอร์รัปต์ในแบบต่างๆมีรายละเอียดการทำงานดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA (Global enable/disable interrupt) : ใช้เอ็นเอเบิลและดิสเอเบิลการตอบสนองการอินเตอร์รัปต์ทั้งหมด

"0" ดิสเอเบิลการอินเตอร์รัปต์ นั่นคือ กำหนดให้ไมโครคอนโทรลเลอร์ไม่ตอบสนองการอินเตอร์รัปต์

"1" เอ็นเอเบิลการอินเตอร์รัปต์ นั่นคือ กำหนดให้ไมโครคอนโทรลเลอร์สามารถตอบสนองการอินเตอร์รัปต์จากแหล่งกำเนิดต่างๆ

นั่นคือ ถ้าต้องการให้ไมโครคอนโทรลเลอร์ตอบสนองการอินเตอร์รัปต์ไม่ว่าจะแหล่งกำเนิดใด จะต้องเซตบิตนี้ก่อนเสมอ สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

ET2 (Timer 2 interrupt enable) : ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวหรือการแคปเจอร์ในไทเมอร์/เคาน์เตอร์ 2 จะมีเฉพาะในเบอร์ AT89C52 และในอนุกรม AT89Sxx เท่านั้น บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

ES (serial port interrupt enable bit) : ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากการรับส่งข้อมูลทางพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

ET1 (Timer 1 interrupt enable) : ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวในไทเมอร์/เคาน์เตอร์ 1 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

EX1 (External interrupt 1 enable bit) : ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา INT บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

ET0 (Timer 0 interrupt enable) : ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวในไทเมอร์/เคาน์เตอร์ 0 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

EX0 (External interrupt 1 enable bit) : ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา INT0 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์สำหรับบิต 6 ของรีจิสเตอร์ IE ไม่มีการใช้งาน ต้องกำหนดให้เป็น "0" เสมอ

2.3.4.5 การใช้งานไทเมอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 กำเนิดสัญญาณพัลส์แบบปรับช่วงเวลา on-off ได้

การใช้งานไทเมอร์/เคาน์เตอร์ 0 ในไมโครคอนโทรลเลอร์ MCS-51 ในการกำเนิดสัญญาณพัลส์สี่เหลี่ยม โดยผู้ใช้งานสารถเลือกค่าเวลาของสัญญาณพัลส์บวกหรือ T_{on} หรือ Mask และช่างเวลาของพัลส์ลบหรือ T_{off} หรือ Space ได้อย่างอิสระ ซึ่งจะเป็นแนวทางให้สามารถใช้ไทเมอร์/เคาเตอร์ในการสร้างสัญญาณรูปแบบอื่นๆได้ตามที่ต้องการ

แนวคิด

ในการใช้งานไทเมอร์ในการทดลองนี้จะเน้นการใช้งานไทเมอร์ 0 และไทเมอร์ 1 เป็นหลัก เพราะมีอยู่ในไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์ ในขณะนี้ไทเมอร์ 2 จะมีเฉพาะบางเบอร์เท่านั้นเนื่องจากไทเมอร์1 ทักจะถูกนำไปใช้ในการสร้างอัตราบอดสำหรับการสื่อสารผ่านพอร์ตอนุกรม ดังนั้นในการทดลองนี้จึงใช้ไทเมอร์ 0 เป็นตัวกำเนิดสัญญาณ โดยต้องกำหนดให้ทำงานในโหมด 1 คือเป็นไทเมอร์ 16 บิต

แนวความคิดในการสร้างสัญญาณสี่เหลี่ยม คือ ให้ไทเมอร์เป็นฐานเวลา 1 ms จากนั้นให้ทำการรับค่าเวลา T_{on} และ T_{off} เข้ามา แล้วทำการเซตขาสัญญาณ T0 เป็น "1" แล้วนำค่าเวลา T_{on} ที่ได้เป็นตัวกำหนดวงรอบในการนับจำนวนโอเวอร์โฟลงของไทเมอร์ 0 จากนั้นให้ทำการเคลียร์ขาสัญญาณ T0 เป็น "0" และนำค่าเวลา T_{off} ที่ได้มาเป็นตัวกำหนดรอบในการนับจำนวนโอเวอร์โฟลวของไทเมอร์ 0 ทำการวนโปรแกรมเช่นนี้ไปเรื่อยๆก็จะได้สัญญาณต่อเนื่องเป็นรูปสี่เหลี่ยมออกมา

ตัวอย่างเช่น ถ้าตั้งค่า T_{on} ไว้ที่ 50 และ T_{off} ไว้ที่ 25 ในช่วงที่ขาสัญญาณถูกเซตเป็น "1" จะต้องทำให้เกิดการโอเวอร์โฟลว์ 50 ครั้ง ซึ่งแต่ละครั้งมีค่าเวลาเท่ากับ 1 ms ทำให้ได้ค่าเวลา T_{on} เป็น 50 ms และในช่วงเวลาที่ขาสัญญาณถูกเคลียร์เป็น "0" จะใช้ทำให้เกิดการโอเวอร์โฟลว์ 25 ครั้ง ทำให้ได้ค่าเวลา T_{off} เป็น 25 ms ตามต้องการ

ผู้ใช้งานสามารถคำนวนค่า TH0 และ TL0 เพื่อกำหนดฐานเวลาที่ต้องการได้จากช่วงเวลาที่ไทเมอร์ 0 ทำงานจนเกิดการโอเวอร์โฟลว เท่ากับ

$$T_{overflow} = \frac{12}{F_{osc}} \times 65536 - [(TH0 \times 256) + TL0]$$

วิธีการคำนวนหรือกำหนดค่าของ TH0 และ TL0

เริ่มต้นด้วยการกำหนดค่าความถี่สัญญาณที่ต้องการ ในที่นี้กำหนดเป็น 500 HZ ที่ค่าความถี่นี้จะมีคาบเวลาเท่ากับ 2 มิลลิวินาที โดยคำนวนจาก T=1/F (โดย F คือค่าความถี่ในหน่วย HZ และ T คือค่าเวลาในหน่วยวินาที) กำหนดให้สัญญาณสี่เหลี่ยมที่ต้องการมีดิวตี้ไซเกิล 50% นั่นคือ มีช่วงเวลาที่มีและไม่มีสัญญาณเท่ากัน ดังนั้น Ton จึงเท่ากับ Toff ทำให้

$$T_{period} = T_{on} + T_{off}$$

$$T_{period} = 2(T_{on}) \times \text{ค่า } T_{overflow}$$

แทนค่า T_{period} ด้วย 2 ms จะได้

$$2ms = 2T_{on}$$

$$T_{on} = 1ms$$

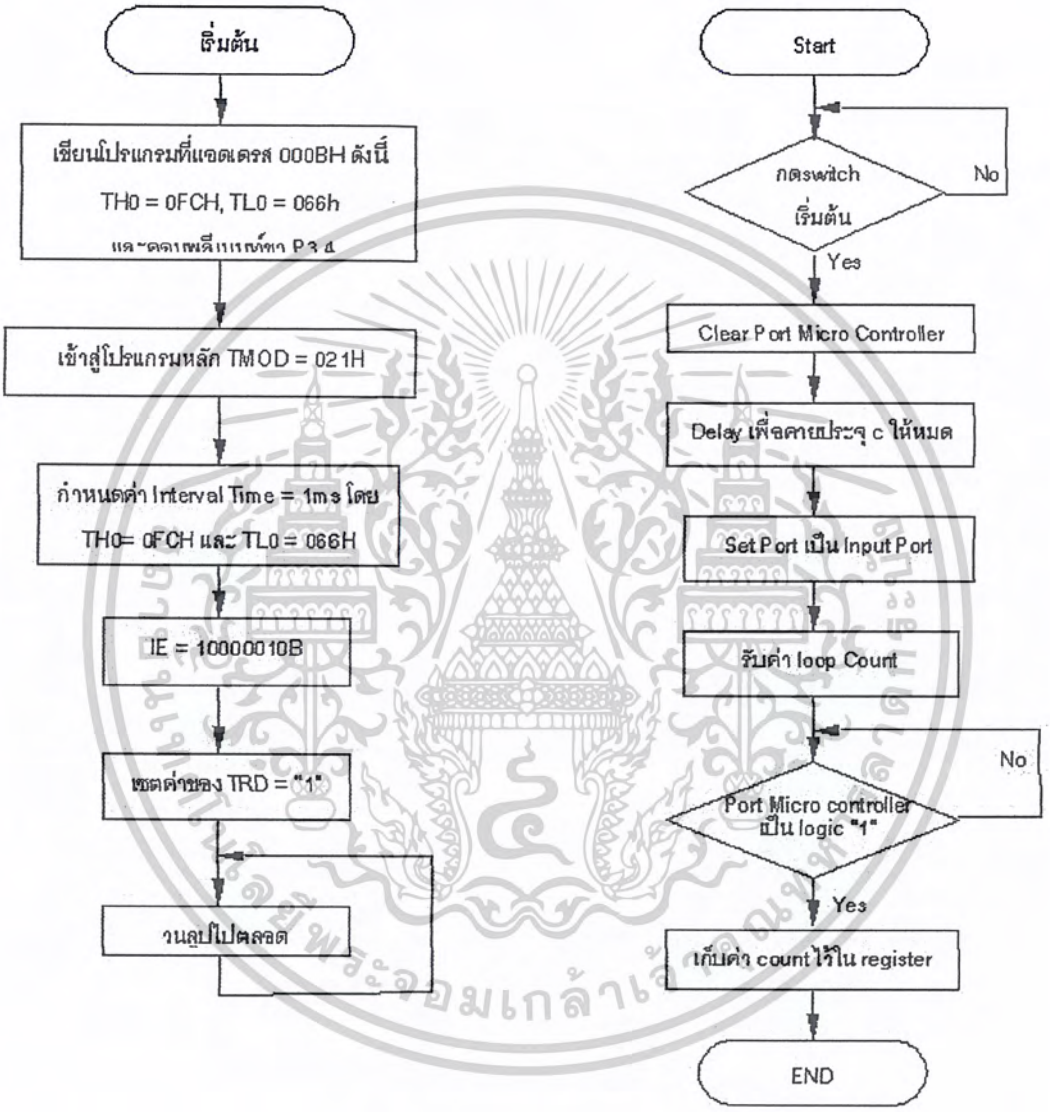
ดังนั้นค่า $T_{overflow}$ จึงเท่ากับ 1 ms นำไปคำนวนหาค่า TH0 และ TL0 จากสมการ

$$T_{overflow} = \frac{12}{F_{osc}} \times 65536 - [(TH0 \times 256) + TL0]$$

แทนค่า $T_{overflow}$ เท่ากับ 1ms และ f_{osc} เท่ากับ 11.0592 MHz และกำหนดให้ (TH0 x 256) + TL0 เป็นค่า reload จะได้

$$1 \times 10^{-3} = \frac{12}{11.0592 \times 10^{6}} \times (65536 - \text{reload})$$

ดังนั้นค่า reload จะได้เท่ากับ 64,614 นำมาหารด้วย 256 จะได้ผลลัพธ์เป็น 252 เศษ 102 ผลหาร 252 ก็คือค่าของ TH0 และเศษ 102 คือค่าของ TL0 นำค่าทั้งสองมาแปลงเป็นเลขฐานสิบหก จะได้ค่า TH0 เท่ากับ 0FCH และค่า TL0 เป็น 066H



รูปที่ 2-16 ด้านซ้ายแสดง Flow chat ของโปรแกรมการสร้างสัญญาณพัลส์สี่เหลี่ยมจากไทเมอร์ 0 ของไมโครคอนโทรลเลอร์ และด้านขวาแสดง Floe chat ของโปรแกรมการนับเวลาการ charge ของประจุ C และเก็บค่าเวลาการ charge ไว้ใน register

2.4 ส่วนรับและแสดงผลของหุ่นยนต์

หุ่นยนต์โดยทุกชนิดจำเป็นต้องมีส่วนรับคำสั่งจากผู้ใช้อย่างน้อยก็ต้องมีสวิตช์ปิด-เปิด แต่ส่วนของการแสดงผลก็มีความจำเป็นในการแสดงสถานะของตัวหุ่นเพื่อที่จะได้รู้ถึงลำดับการทำงานของหุ่นในขณะนั้น

2.4.1 **ส่วนของการรับคำสั่งจากแป้นกดที่ตัวหุ่น**(key pad) เป็นส่วนของฮาร์ดแวร์ที่ติดอยู่กับตัวหุ่นออกแบบมาเพื่อติดต่อกับผู้ใช้โดยตรงเพื่อเป็นการกดเริ่มคำสั่งทำงานของตัวหุ่นและเป็นการเซตระบบต่างๆของตัวหุ่น ซึ่งจะมีไมโครคอนโทรลเลอร์ 89C51 เป็นตัวควบคุมและรับค่าการกดออกคำสั่งของปุ่มเหล่านี้

การอ่านค่าหรือรับค่าการกดสวิตช์เป็นอีกงานหนึ่งที่ไมโครคอนโทรลเลอร์ต้องสามารถรองรับและเชื่อมต่อใช้งานร่วมด้วยได้ วงจรของสวิตช์ที่ใช้ใน project นี้ใช้การต่อวงจรแบบเมตริกซ์ (matrix switch) ดังในรูป สวิตช์จะถูกต่อกันในแนวแกนตั้งและแกนนอน จะเรียกแนวตั้งว่า หลักหรือคอลัมน์ (column) ในขณะที่แนวนอนจะเรียกว่า แถวหรือโรว์ (row) ดังนั้นค่าของสวิตช์จะต้องประกอบด้วย ตำแหน่งในแนวหลักและแถว กระบวนการที่จะทำให้ได้มาซึ่งค่าของสวิตช์มีขั้นตอนซับซ้อนพอสมควร แต่วงจรของสวิตช์แบบนี้มีข้อดีคือสามารถรองรับการเพิ่มของสวิตช์ได้อย่างสะดวก เพียงเพิ่มเติมจำนวนสวิตช์และแก้ไขซอฟต์แวร์อีกเล็กน้อยเท่านั้น ทำให้วงจรสวิตช์แบบเมตริกซ์เป็นที่นิยมใช้มากในระบบควบคุมอัตโนมัติหรือกึ่งอัตโนมัติที่มีจำนวนสวิตช์มากกว่า 8 ตัว ในการใช้งานทั่วไปจะเรียกสวิตช์แบบเมตริกซ์นี้ว่า คีย์แพด (keypad)

การเชื่อมต่อคีย์แพดเข้ากับไมโครคอนโทรลเลอร์ MCS-51

จากวงจรที่แสดงในรูป จะใช้พอร์ต 2 ของไมโครคอนโทรลเลอร์เชื่อมต่อเข้ากับคีย์แพดทั้ง 7 เส้นคือ สายของคอลัมน์ 3 สาย C0-C2 และสายทางโรว์หรือแถวอีก 4 สายคือ R0-R3 โดยเฉพาะที่ขาพอร์ต P2.0-P2.3 จะต้องต่อตัวต้านทานพูลอัปไว้เพื่อกำหนดสภาวะเริ่มต้นที่ไม่มีการกดคีย์ โดยไมโครคอนโทรลเลอร์จะทำการส่งข้อมูล "0" ไปยัง P2.6,P2.5 และ P2.4 ตามลำดับ ในทุกครั้งที่มีการส่งข้อมูลไปยังสายคอลัมน์ของคีย์แพดไมโครคอนโทรลเลอร์จะทำการอ่านค่าที่ P2.0-P2.3 เข้ามาด้วย หากไม่มีการกด ค่าของ P2.0-P2.3 ก็จะเป็น "1"

2.4.2 ส่วนของการแสดงผลออกหน้าจอ LCD มีไว้เพื่อแสดงสถานะต่างๆ ของหุ่นยนต์และเพื่อติดต่อกับผู้ใช้ได้โดยตรง โดยจอ LCD ที่ใช้ใน Project นี้ใช้ โมดูล LCD แบบ 2 บรรทัด 16ตัวอักษร

รายละเอียดเกี่ยวกับโมดูล LCD ในโมดูล LCD จะมีส่วนประกอบหลักๆ 3 ส่วนดังนี้

ตัวแสดงผล (display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

ตัวควบคุม (controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือ เลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะ ชิปที่นิยมใช้คือ เบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD แบบกราฟิก

ตัวขับ (driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น

โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ซึ่งโมดูลที่ใช้ในโครงการนี้เป็น แบบ อักขระ ดังรูป เป็นบล็อกไดอะแกรมภายในของชิปควบคุม LCD เบอร์ HD44780 ซึ่งใช้ในโมดูล LCD แบบอักขระ ประกอบด้วย

บัฟเฟอร์อินพุตเอาต์พุต เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอกเพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจิสเตอร์คำสั่ง (Instruction Register : IR) เป็นรีจิสเตอร์ใช้รับข้อมูลคำสั่งจากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

รีจิสเตอร์ข้อมูล (Data Register : DR) เป็นรีจิสเตอร์ใช้รับข้อมูลจากอุปกรณ์ภายนอก เพื่อถ่ายทอดต่อไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผล หรือนำไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

แรมเก็บข้อมูลแสดงผล (Display Data Ram : DDRAM) เป็นหน่วยความจำแรมทำหน้าที่เก็บข้อมูลที่มาจากรีจิสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดตาราง

(Look up-lable) ของตัวอักษรที่เก็บไว้ในหน่วยความจำรอมและแรมเก็บตัวอักษร เพื่อนำไปแสดงที่ตัวแสดงผล

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

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

แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานการณ์ทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่า ตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน

โมดูล LCD ขนาด 16 ตัวอักษร 2 บรรทัด (LCD 16X2)

สำหรับโมดูล LCD ที่ยกมาใช้ในการเรียนรู้ในการทดลอง เป็นขนาด 16 ตัวอักษร 2 บรรทัด เนื่องจากราคาถูก หาง่าย และเป็นโมดูล LCD ที่มีโครงสร้างเป็นมาตรฐาน มีผู้ผลิตหลายราย และมีการระบุเบอร์แตกต่างกันออกไปตามผู้ผลิต แต่อย่างไรก็ตามคอนโทรลเลอร์ที่ใช้คือเบอร์เดียวกันนั่นคือเบอร์ HD44750 ของฮิตาชิ

โมดูล LCD ขนาด 16X2 มีขาต่อใช้งานทั้งสิ้น 14 ขา มีการจัดขาดังในรูปที่ สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

V_{SS} (ขา 1) : ต่อกราวด์

V_{DD} (ขา 2) : ต่อไฟเลี้ยง +5 โวลท์

V_O (ขา 3) : เป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล

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

R/W (ขา 5) : เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับโมดูล LCD ถ้าเป็น "0" เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น "1" จะเป็นการอ่านข้อมูล

E (ขา 6) : เป็นขาสำหรับรับสัญญาณพัลส์เอ็นเอเบิลโมดูล LCD ให้ทำงาน

D0-D7 (ขา 7-14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิตอนึ่งขา RS,R/W และ E จะใช้งานร่วมกัน โดยมีความสัมพันธ์แสดงในตาราง ที่

คำสั่งควบคุมโมดูล LCD

ในการเขียนคำสั่งลงในตัวควบคุม แน่นอนว่าต้องกำหนดให้ขา RS และ R/W เป็น " 0 " แล้วเขียนคำสั่งตามไป คำสั่งควบคุมโมดูล LCD ของชิปควบคุม HD44780 ที่สำคัญมี 10 คำสั่งดังนี้

1. **คำสั่งเคลียร์ตัวแสดงผล (clear display)**

มีข้อมูลคำสั่งเป็น 01H เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่าง หรือ space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมเอ็กซิคิวต์คำสั่งนี้ จะทำการกำหนดแอดเดรสของ DDRAM เป็น 0 เคอร์เซอร์จะกลับไปอยู่ที่ตำแหน่งซ้ายมือสุดของจอแสดงผล แล้วเซตบิต I/D (ซึ่งจะกล่าวถึงภายหลัง) ให้เป็น "1"

2. **คำสั่ง return home**

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น "1" เป็นคำสั่งให้เคอร์เซอร์เคลื่อนที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอแสดงผลไม่เปลี่ยนแปลง นั่นคือ ข้อมูลคำสั่งของคำสั่งนี้จะเป็น 02H หรือ 03H ก็ได้

3. **คำสั่งเลือกโหมดการป้อนข้อมูล (Entry mode Set)**

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" เมื่อเกิดข้อมูลใหม่บนจอแสดงผล ตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักษรข้อมูลเดิมจะถูกดันไปทางซ้าย แต้ถ้าหากบิตนี้เป็น "0" เมื่อเกิดข้อมูลใหม่ตัวเคอร์เซอร์จะเลื่อนไปทางขวามือ

บิต I/D เป็นบิตที่ใช้กำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น "1" แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าเป็น "0" แอดเดรสจะลดลง

ดังนั้นข้อมูลคำสั่งที่เกิดขึ้นสำหรับคำสั่งนี้ได้แก่ 04H – 07H (4 ข้อมูลคำสั่ง) และที่ใช้บ่อย คือ 06H หมายถึง กำหนดให้เมื่อเกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางขวามือ และแอดเดรสของ 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" จะเป็นการปิดแสดงผล

บิต C ใช้ควบคุมการแสดงตัวเคอร์เซอร์บนจอแสดงผล ถ้าต้องการให้มีเคอร์เซอร์แสดงผลบนจอแสดงผล ต้องกำหนดให้บิตนี้เป็น "1" ถ้ากำหนดให้เป็น "0" จะเป็นการปิดเคอร์เซอร์ หรือไม่แสดงเคอร์เซอร์

บิต B ใช้ควบคุมการกระพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น "1" เคอร์เซอร์จะกระพริบ

ดังนั้นจะมีข้อมูลคำสั่งได้ตั้งแต่ 08H – 0FH (8 รูปแบบคำสั่ง) ที่ใช้บ่อยคือ 0CH เป็นการสั่งให้เปิดจอแสดงผล แต่ไม่แสดงเคอร์เซอร์ และ 0FH เป็นการสั่งให้เปิดจอแสดงผล แสดงเคอร์เซอร์ และ สั่งให้เคอร์เซอร์กระพริบ

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	*	*

การควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผล ขึ้นอยู่กับการกำหนดบิต 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	1C – 1FH

6. คำสั่งกำหนดฟังก์ชั่นการทำงาน

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 แต่ ถ้าเป็น "1" จะเป็นแบบ 8 บิต

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล ถ้าเป็น "0" จะแสดงผล 1 บรรทัดถ้าเป็น "1" จะแสดง2บรรทัด ในกรณีที่จอแสดงผลสามารถแสดงได้มากกว่า 2 บรรทัด และต้องการให้แสดงผลมากกว่า 2 บรรทัดก็กำหนดบิต N ให้เป็น "1" จุดที่น่าสังเกตคือ โมดูล LCD แบบ 16 ตัวอักษร 1 บรรทัด แม้จะมีบรรทัดการแสดงผลเพียง1บรรทัดแต่จะต้องกำหนด N ให้เป็น "1" เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่องคือ 00H และ 40H

บิต F ใช้เลือกความละเอียดของตัวอักษรให้การแสดงผล ถ้าบิตนี้เป็น "0" จะเป็นการแสดงผลแบบ 5x7 จุดและถ้าเป็น "1" จะแสดงผลเป็นแบบ 5x10 จุด

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

7. คำสั่งเลือกแอดเดรสของ CGRAM

เมื่อต้องการกำหนดแอดแดรสของ CGRAM ต้องกำหนดให้บิต 7 เป็น "0" บิต 6 เป็น "1" ส่วนอีก 6 บิตที่เหลือจะแทนด้วยค่าแอดเดรสของ CGRAM จะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ ก่อนที่จะอ่านหรือเขียนข้อมูลให้ 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 ทั้งนี้จำนวนแอดเดรสยังขึ้นกับการกำหนดสถานะที่บิต N ด้วยหาก N เป็น "0" แอดเดรสของDDRAM จะอยู่ระหว่าง 80H-0CFH และถ้าบิต N เป็น "1" แอดเดรสของ DDRAM จะมี 2 ช่วง 8CH-87H และ 0C0H-0C7H

9. คำสั่งการอ่านแฟลก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
BF	A	A	A	A	A	A	A

เป็นคำสั่งที่ใช้อ่านแฟลก BUSY (BF) โดยแฟลกนี้จะเป็นตัวบอกสถานะของตัวควบคุม LCD ว่าพร้อมจะรับข้อมูลอยู่หรือไม่ ถ้าหากบิต BF เป็น "0" แสดงว่าตัวควบคุม LCD พร้อมรับข้อมูลหรือคำสั่งแต่ถ้าเป็น "1" แสดงว่า ขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการทำงานภายในหรือกำลังประมวลผลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง เมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา R/W เป็น "1" ด้วย แต่สัญญาณที่ RS ยังต้องเป็น "0" อยู่เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง นอกจากนี้ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรส CGRAM และ DDRAM ด้วยโดยบิต 0-บิต 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

การเขียนคำสั่งและข้อมูลให้แก่โมดูล LCD

ในการเขียนข้อมูลเพื่อควบคุมให้โมดูล LCD แสดงผลตามที่ผู้ใช้งานต้องการ ต้องส่งคำสั่ง (instruction) แล้วกำหนดโหมดการทำงานให้แก่โมดูล LCD ก่อน จากนั้นจึงค่อยส่งข้อมูล (data) ที่ต้องการแสดงผล เนื่องจากบัสข้อมูลของโมดูล LCD มี 8 เส้น คือ D0-D7 และใช้เป็นทางผ่านของทั้งคำสั่งและข้อมูล ดังนั้นในการส่งคำสั่งและข้อมูลจึงต้องอาศัยการกำหนดสัญญาณลอจิกที่ขา RS ได้รับลอจิก "1" ข้อมูลที่ป้อนให้ขณะนั้นเป็นข้อมูลที่ใช้ในการแสดงผล

เมื่อต้องการเขียนหรืออ่านข้อมูลใน CGRAM และ DDRAM เริ่มต้นต้องกำหนดแอดเดรสที่ต้องการอ่านหรือเขียนก่อน โดยใช้คำสั่งเลือกแอดเดรสจากนั้นกำหนดให้ขา RS เป็น "1" เพื่อแจ้งให้ตัวควบคุมภายในโมดูล LCD ทราบว่าข้อมูลที่ปรากฏต่อไปนี้เป็นข้อมูลปกติไม่ใช่คำสั่ง

ในกรณีที่ต้องการอ่านข้อมูลต้องกำหนดให้ขา R/W เป็น "1" ข้อมูลขนาด 8 บิต (หรือ 4 บิต) ก็จะปรากฏบนบัสข้อมูล โดยข้อมูลที่อ่านออกมาได้จะเป็นข้อมูลจากแอดเดรสของ CGRAM หรือ DDRAM ตามที่ต้องการ

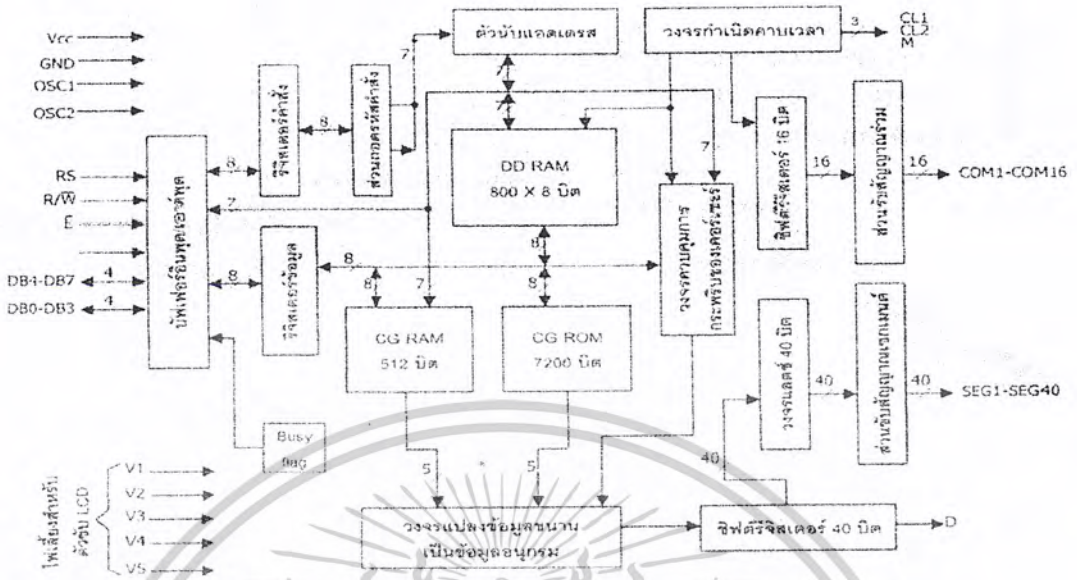
ในกรณีที่ต้องการเขียนข้อมูล เมื่อกำหนดแอดเดรสและป้อนลอจิก "1" ให้ขา RS แล้วแล้วต้องกำหนดให้ขา R/W เป็น "0" ข้อมูลที่อยู่บนบัสข้อมูลจะถูกเขียนลงในรีจิสเตอร์ DR จากนั้นจึงถ่ายทอดลงใน DDRAM ต่อไป

จังหวะการทำงานของ LCD โมดูล

ในการติดต่อกับโมดูล LCD จะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูล เนื่องจากต้องรอให้คอนโทรลเลอร์ภายใน LCD โมดูล แปลความหมายของรหัสคำสั่งและทำงานตามคำสั่งให้เรียบร้อยก่อนจากนั้นจึงจะรับข้อมูลหรือดำเนินการต่อไป

ดังนั้นในการใช้งานโมดูล LCD ผู้เขียนโปรแกรมต้องมีโปรแกรมเพื่อหน่วงเวลารอให้โมดูล LCD พร้อมทำงานด้วย โดยเมื่อเริ่มจ่ายไฟให้แก่โมดูล LCD ต้องรอประมาณ10 มิลลิวินาที เพื่อให้โมดูล LCD ทำการเตรียมความพร้อมหรืออินิเซียล (Initial) หลังจากนั้นก็จะกำหนดลอจิกให้แก่ขา RS ว่า ข้อมูลต่อไปที่จะได้รับนั้นเป็นรหัสคำสั่งหรือเป็นข้อมูลที่ต้องการแสดงผล จากนั้นจะเป็นการส่งข้อมูลมารอที่บัสข้อมูล D0-D7 (กรณีทำงานในโหมด 8 บิต) ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ไปที่ขา E เพื่อเอ็นเอเบิลโมดูล LCD ให้รับข้อมูลจากบัสข้อมูลเข้าไป โดยพัลส์ที่ป้อนเข้าที่ขา E ของโมดูล LCD ต้องเป็นพัลส์ขอบขาขึ้น จากนั้นทำการหน่วงเวลา 2 มิลลิวินาทีเพื่อให้คอนโทรลเลอร์ในโมดูล LCD แปลความหมายของลอจิกที่ขา RS ว่าข้อมูลต่อไปที่จะได้รับนั้นเป็นรหัสคำสั่งหรือเป็นข้อมูลที่ต้องการแสดงผล จากนั้นจะเป็นการส่งข้อมูลมารอที่บัสข้อมูล D0-D7 (กรณีทำงานในโหมด 8 บิต) ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ไปที่ขา E เพื่อเอ็นเอเบิลโมดูล LCD ให้รับข้อมูลจากบัสข้อมูลเข้าไป โดยพัลส์ที่ป้อนเข้าที่ขา E ของโมดูล LCD ต้องเป็นพัลส์ขอบขาขึ้น จากนั้นทำการหน่วงเวลา 2 มิลลิวินาที

ทั้งหมดที่กล่าวมาคือขั้นตอนและจังหวะในการทำงาน 1 รอบของโมดูล LCD จะเห็นได้ว่ามีโปรแกรมย่อยที่สำคัญอยู่ 3 โปรแกรมย่อยคือ โปรแกรมอินิเซียล LCD โปรแกรมหน่วงเวลา และ โปรแกรมย่อยการส่งพัลส์เพื่อเอ็นเอเบิลโมดูล LCD

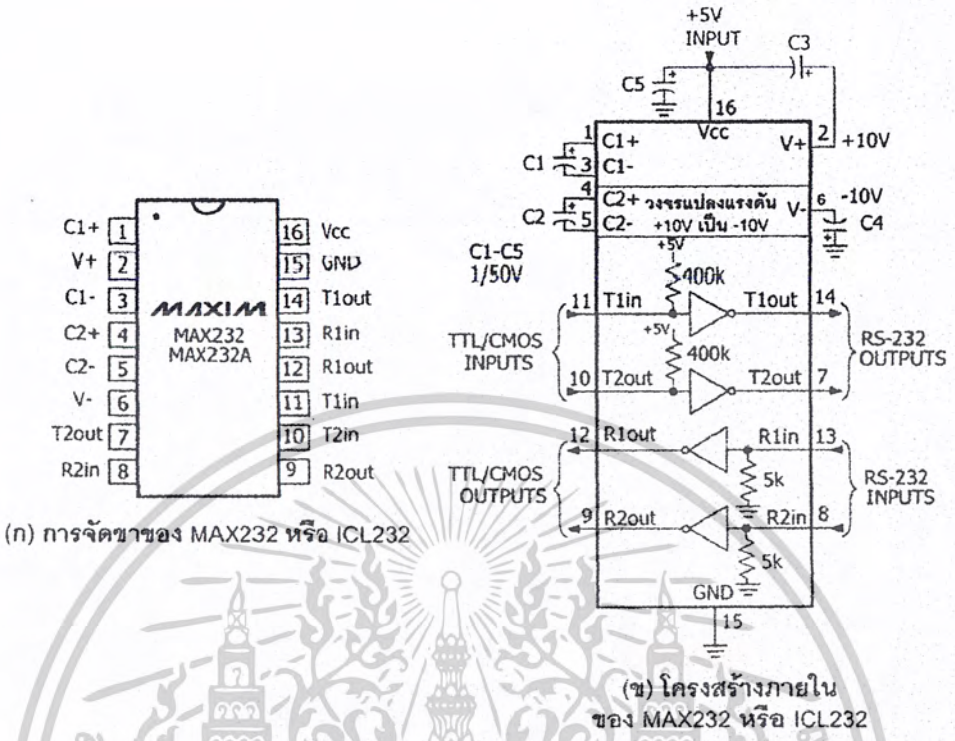


รูปที่ 2-17 แผนผังการเชื่อมต่อ LCD กับไมโครคอนโทรลเลอร์

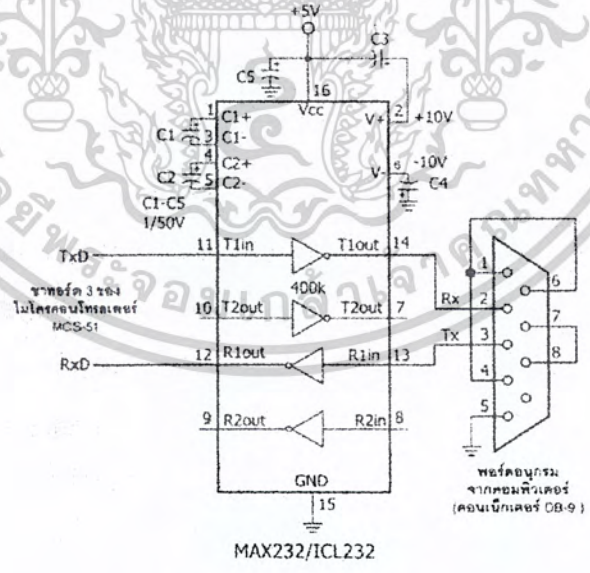
2.5 การเชื่อมต่อพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมาตรฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ ± 3 ถึง ± 12 V ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์ MCS-51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณ

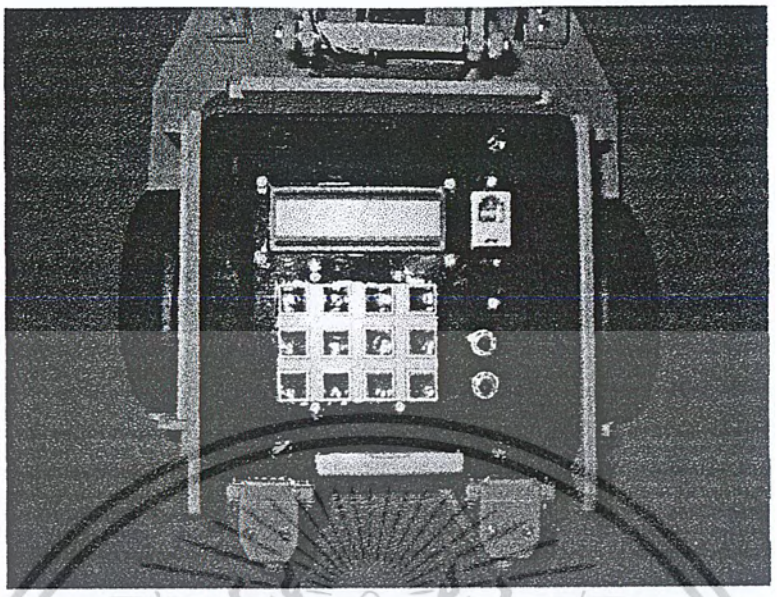
ไอซีที่ทำหน้าที่ในการแปลงสัญญาณนี้ต้องทำการแปลงข้อมูลส่งของไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับทีทีแอลเพื่อให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS-51 ได้อย่างสมบูรณ์ ไอซีดังกล่าวมีด้วยกันหลายเบอร์จากหลายผู้ผลิต อาทิ MAX232 จาก MAXIM หรือ ICL232 จาก HARRIS เป็นต้น



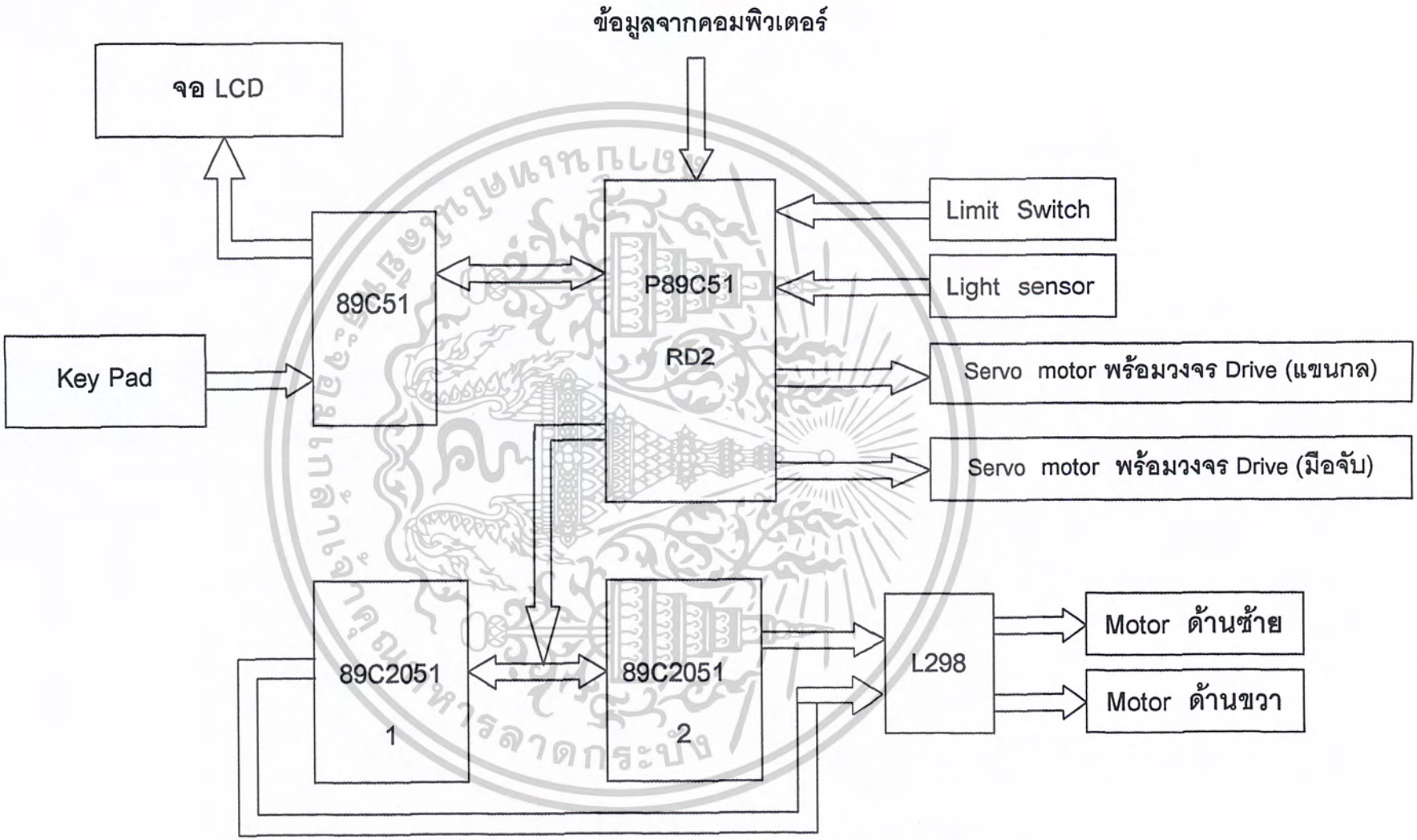
รูปที่ 2-18 รูปแสดงไอซี MAX232 และโครงสร้างภายในของ ไอซี



รูปที่ 2-19 แสดงการเชื่อมต่อกับพอร์ตอนุกรมของไอซี



รูปที่ 2-20 รูปแสดง การเชื่อมต่อของ Key pads ,LCD , พอร์ตอนุกรม



รูปที่ 2-21 แสดงผังการทำงานเบื้องต้นของส่วนควบคุมหลักในหุ่นยนต์อัตโนมัติ

2.6 แหล่งจ่ายไฟ

ในการออกแบบ project นี้ได้ใช้แบตเตอรี่แห้งในการจ่ายไฟ 2 แบบด้วยกันดังนี้

2.6.1 แบตเตอรี่แห้ง 12 v 1.3AH 1 ก้อนใช้ในการจ่ายให้กับมอเตอร์

รูปที่ 2-22 รูปแสดงแบตเตอรี่แห้ง 12 V.

2.6.2 แบตเตอรี่แห้ง 7.2 v 2700 mAh 1ก้อนใช้ในการจ่ายไฟเลี้ยงวงจร

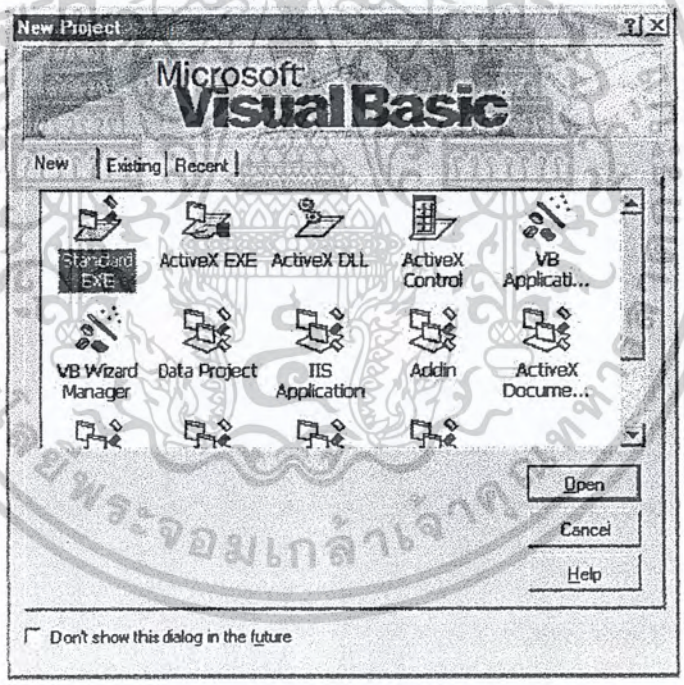
รูปที่ 2-23 รูปแสดงแบตเตอรี่แห้ง 7.2 V.

2.7 โปรแกรมที่ใช้ในการควบคุมหุ่นยนต์

ในที่นี้ทางผู้จัดทำได้เลือกใช้โปรแกรม Visual Basic 6.0 ซึ่งเป็นโปรแกรมที่ใช้ติดต่อกับผู้ใช้ผ่านทางคอมพิวเตอร์ เนื่องจากเป็นเครื่องมือแอปพลิเคชั่นบนวินโดว์ ที่มีประสิทธิภาพสูงสุดตัวหนึ่ง ซึ่งสามารถรองรับความสามารถต่างๆที่อยู่ในวินโดว์ได้อย่างครบถ้วน และยังง่ายในการใช้งาน ทำให้สามารถพัฒนาแอปพลิเคชั่นต่างๆ ได้เวลาอันรวดเร็ว ไม่ว่าจะเป็นส่วนของติดต่อกับผู้ใช้ การเชื่อมโยงฐานข้อมูล ตลอดจนการติดต่อสื่อสารผ่านทางพอร์ตอนุกรม โดยหลักการทำงานของโปรแกรม Visual Basic 6.0มีดังนี้

รู้จักส่วนประกอบต่าง ๆ ของ VB6

สำหรับก่อนอื่นให้เรารันโปรแกรม VB6 ขึ้นมาก่อน เมื่อเรารันขึ้นมาแล้วจะปรากฏหน้าจอดังรูป ซึ่งโดยทั่วไปแล้ว เราจะเลือก Standard.EXE ซึ่งเป็นการใช้ VB6 ในการโปรแกรมที่รันบนวินโดวส์ทั่วไป

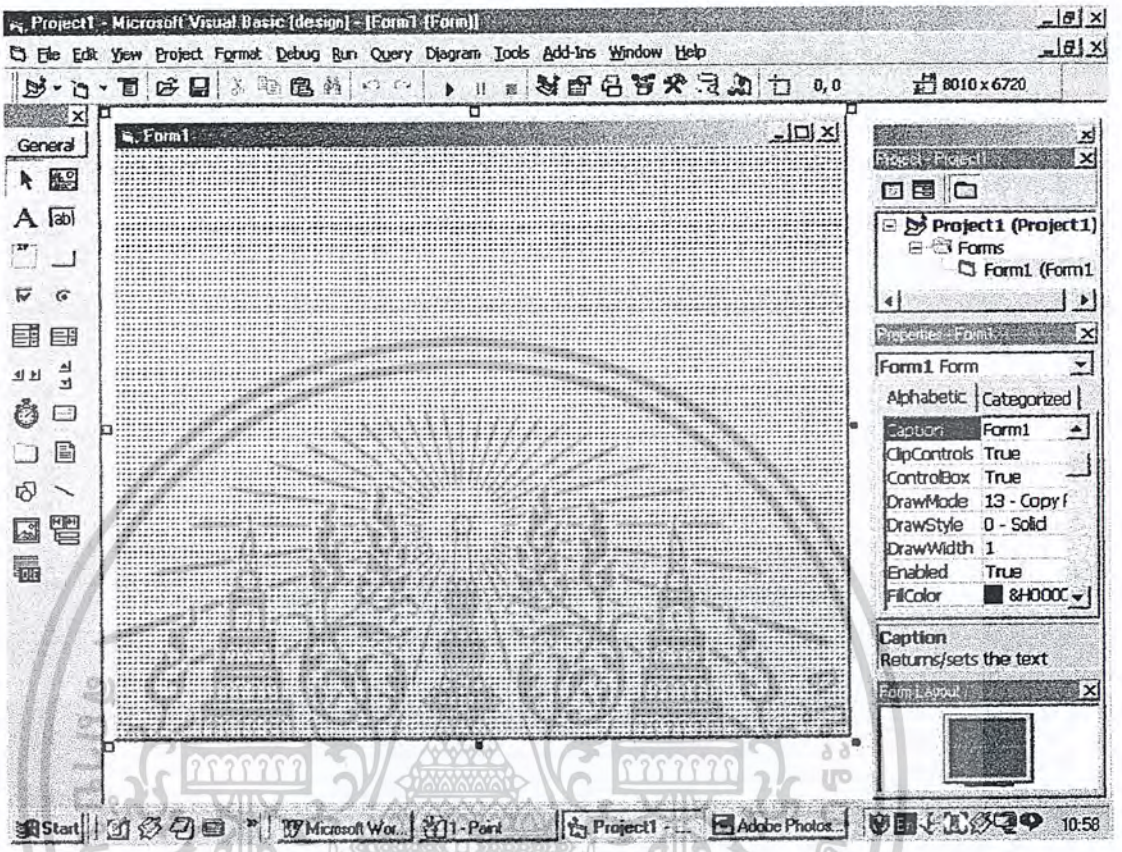


รูปที่ 2-24 หน้าต่างการเลือกชนิดโปรเจกต์

เนื่องจากโปรเจ็กต์ชนิด Standard.EXE นั้น เป็นการเขียนโปรแกรมทั่วไปที่รันบนวินโดวส์ โดยโปรเจ็กต์คือกลุ่มของไฟล์ที่เราจะนำมารวมกันเพื่อสร้างโปรแกรมใน VB6

เมื่อเราเลือกเสร็จแล้ว ต่อไปจะปรากฏหน้าจอของ Visual Basic 6 โดยมีชื่อส่วนประกอบต่างๆ ที่เราจำเป็นต้องทราบในตอนนี้ ดังนี้

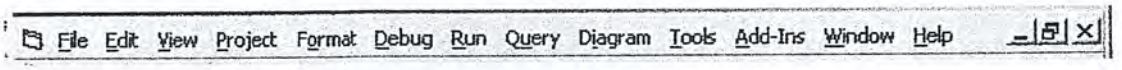
หน้าต่างหลัก (Main Window)



รูปที่ 2-25 หน้าต่างหลักของโปรแกรม Visual Basic 6

หน้าต่างหลักเป็นหน้าต่างแรกที่ปรากฏเมื่อเรียกใช้โปรแกรม VB6 ซึ่งประกอบด้วยส่วนต่างๆคือ Menubar, Toolbar, Toolbox, Project Explorer, Properties Window, Form โดยมีรายละเอียดดังนี้

- เมนูบาร์



รูปที่ 2-26 เมนูบาร์

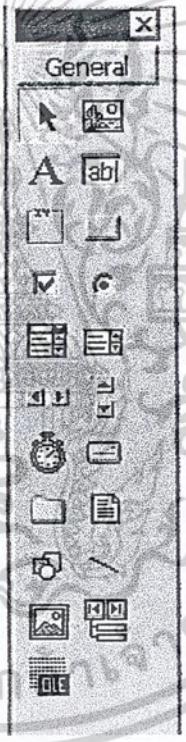
เมนูบาร์เป็นคำสั่งที่เราสามารถใช้งานได้ทั้งหมดใน VB6 ประกอบไปด้วยเมนูทำงานกับ File, View และ Windows

- ทูลบาร์

รูปที่ 2-27 ทูลบาร์

ทูลบาร์เป็นที่แสดงเครื่องมือต่างๆ ที่ช่วยให้เราใช้คำสั่งของ VB6 ได้รวดเร็วยิ่งขึ้น

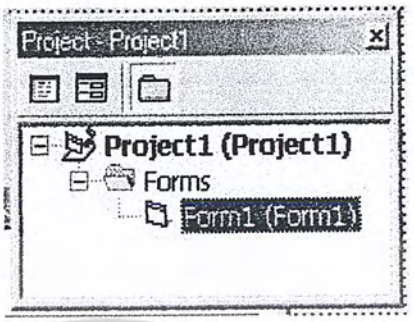
- ทูลบ็อกซ์



รูปที่ 2-28 ทูลบ็อกซ์

ทูลบ็อกซ์เป็นที่แสดงเครื่องมือต่างๆ ที่เราเรียกว่า คอนโทรล ซึ่งเป็นเครื่องมือที่เราสามารถนำไปวางลงบนฟอร์มได้ เพื่อออกแบบหน้าจอของโปรแกรม (เรียกว่าส่วนติดต่อกับผู้ใช้ หรือ User Interface)

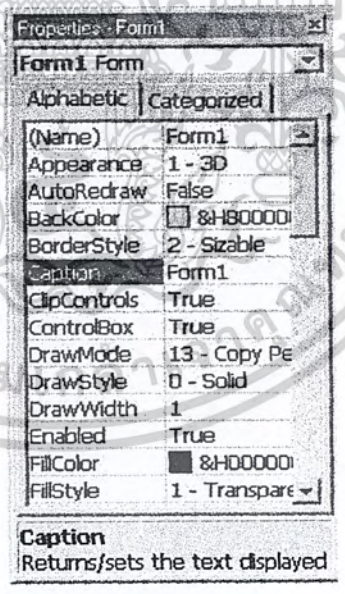
- **หน้าต่างโปรเจกต์**



รูปที่ 2-29 หน้าต่างโปรเจ็กต์

หน้าต่างโปรเจ็กต์เป็นหน้าต่างที่แสดงโมดูล (Modules) ต่างๆ ที่มีอยู่ในโปรเจ็กต์เราทั้งหมด เราสามารถกดปุ่ม <CTRL>+<R> เพื่อให้แสดงหน้าต่างนี้ได้

- **หน้าต่างคุณสมบัติ**

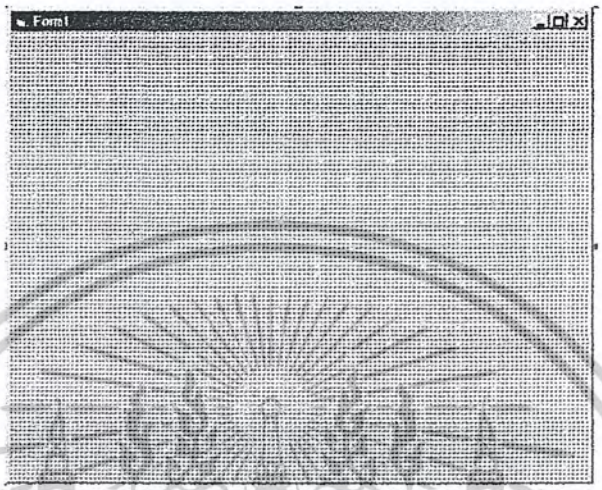


รูปที่ 2-30 หน้าต่างคุณสมบัติ

หน้าต่างคุณสมบัติเป็นหน้าต่างที่แสดงคุณสมบัติของคอนโทรลที่เลือกอยู่ในขนาดนั้น เราสามารถปุ่ม <F4> เพื่อให้แสดงหน้าต่างนี้ได้

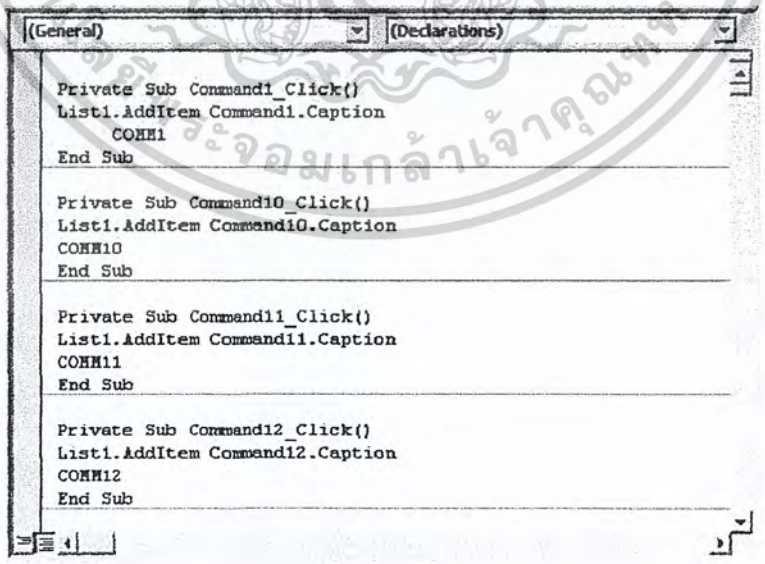
คุณสมบัติ (Properties) คือลักษณะต่างๆ ของคอนโทรลที่ถูกนำมาวางบนฟอร์ม ที่เราสามารถกำหนดได้ เช่น ข้อความที่ปรากฏบนคอนโทรล รูปแบบฟอร์มของคอนโทรล

- **ฟอร์ม**



รูปที่ 2-31 หน้าต่างฟอร์ม

ฟอร์มเป็นฟอร์มที่ใช้ในการออกแบบ เป็นหน้าต่างที่ใช้ในการออกแบบหน้าจอโปรแกรม สำหรับส่วนประกอบที่มีความสำคัญอีกส่วนคือ หน้าต่าง Code Editor ซึ่งเป็นหน้าต่างที่ให้เราใส่คำสั่ง เพื่อควบคุมการทำงานของโปรแกรม เราสามารถเรียกหน้าต่าง Code Editor ขึ้นมาโดยคลิกไอคอน view code ที่หน้าต่างโปรเจ็กต์ หรือดับเบิ้ลคลิกที่ฟอร์มเลยก็ได้



รูปที่ 2-32 หน้าต่าง Code Editor

<u>รู้จักกับคุณสมบัติ เมดถอด และอีเว็นต์</u>

ในการทำงานกับฟอร์ม และคอนโทรล เราจะต้องทำความคุ้นเคยกับ 3 คำต่อไปนี้ คือ

1. คุณสมบัติ (properties) ที่ให้เรากำหนดลักษณะต่างๆของฟอร์ม และคอนโทรล
2. เมดถอด (Method) เป็นการสั่งให้ฟอร์ม และคอนโทรลทำงานตามที่เราร้องขอไป
3. อีเว็นต์ (Events) เป็นเหตุการณ์ที่เกิดขึ้นกับฟอร์มหรือคอนโทรลที่เราสามารถใส่คำสั่ง เพื่อตอบสนองได้

ในที่นี้จะขอพูดถึง คุณสมบัติ เมดถอด อีเว็นต์ ที่ใช้งานมากเท่านั้น ซึ่งโครงงานนี้ใช้คุณสมบัติ เมดถอด อีเว็นต์ ที่สำคัญดังนี้

รู้จักกับฟอร์ม (Forms)

ฟอร์มจะเป็นเครื่องมือที่เราจะต้องทำงานด้วยบ่อยมาก ในการสร้างโปรแกรมด้วย VB6 โดยทีฟอร์มจะเป็นหน้าต่างที่ผู้ใช้ติดต่อทำงานด้วย ผ่านทางคอนโทรลต่างๆ ที่เราวางบนฟอร์ม

คุณสมบัติที่สำคัญของฟอร์ม

Name*	กำหนดชื่อของฟอร์มที่เราใช้อ้างอิงถึงในโปรแกรม
BorderStyle	กำหนดลักษณะในการเปลี่ยนขนาดของฟอร์มเป็นอย่างไร
MinButton, MaxButton	กำหนดให้ฟอร์มสามารถ Maximize, Minimize ได้หรือไม่
Caption*	กำหนดข้อความที่แสดงบนไตเติ้ลบาร์ของฟอร์ม
Icon	กำหนดรูปไอคอนของฟอร์มเมื่อ Minimize ฟอร์ม
Height, Width*	กำหนดความสูง และความกว้างของฟอร์ม
Left, Top*	กำหนดตำแหน่งของฟอร์มโดยคิดจากตำแหน่งบนซ้ายของหน้าจอ
Movable	กำหนดว่าฟอร์มจะสามารถเคลื่อนย้ายได้หรือไม่
WindowState	กำหนดว่าเมื่อเริ่มต้นเรียกฟอร์ม จะให้อยู่ในแบบ Maximize, Minimize หรือธรรมดา
Enabled*	กำหนดให้ฟอร์มสามารถตอบสนองอีเว็นต์ได้หรือไม่

คุณสมบัติที่มีเครื่องหมาย * เป็นคุณสมบัติที่ใช้กับคอนโทรลอื่นได้โดยที่จะมีความหมายใกล้เคียงกัน

เมดถอด และอีเว็นต์ที่สำคัญของฟอร์ม

Load	เป็นอีเว็นต์ที่เกิดเมื่อ เรียกฟอร์มขึ้นมาครั้งแรก
Click	เป็นอีเว็นต์ที่เกิดเมื่อมีการ Click mouse บนฟอร์ม
Hide	เป็นเมดถอดที่ใช้ซ่อนฟอร์มไว้

รู้จักกับปุ่มคำสั่ง (Command Button)

เราคงพอจะได้ทราบกันมาบ้างแล้วว่า ปุ่มคำสั่งนั้นมีหน้าที่หลักคือ การตอบสนองต่อการ Click mouse ของผู้ใช้ที่สั่งงานมายังโปรแกรมว่า ต้องการให้โปรแกรมทำอะไรตอบกลับไป ดังนั้น

คุณสมบัติของปุ่มคำสั่ง

Style	กำหนดให้เป็นปุ่มคำสั่งธรรมดา หรือมีรูปภาพบนปุ่มด้วย
Picture	กำหนดรูปภาพที่แสดงบนปุ่มคำสั่ง (ใช้ได้เมื่อคุณสมบัติ Style เท่ากับ 1)
DownPicture	กำหนดรูปภาพที่แสดงบนปุ่มคำสั่ง เมื่อปุ่มถูกกด
DisabledPicture	กำหนดรูปภาพที่แสดงบนปุ่มคำสั่ง เมื่อคุณสมบัติ Enabled เท่ากับ False
ToolTipText	เป็นข้อความที่แสดง เมื่อหยุดตัวชี้เมาส์ที่ปุ่มคำสั่งช่วงเวลาหนึ่ง
Caption	ข้อความที่ปรากฏบนปุ่มคำสั่ง
Enabled	กำหนดให้ปุ่มคำสั่งใช้งานได้ หรือไม่ได้
Visible	กำหนดให้คอนโทรลสามารถมองเห็นได้ในตอนรันหรือไม่

รู้จักกับเท็กบ็อกซ์ (TextBox)

คอนโทรลเท็กบ็อกซ์เป็นคอนโทรลที่เราใช้งานบ่อยมากในการรับข้อมูลจากผู้ใช้ที่ป้อนเข้ามาในโปรแกรมของเรา รวมทั้งสามารถแสดงผล และให้ผู้ใช้แก้ไขข้อมูลได้ด้วย

คุณสมบัติของเท็กบ็อกซ์

Maxlength	กำหนดความยาวของข้อความที่สามารถรับได้ของเท็กบ็อกซ์เป็นเท่าไร
MultiLine	กำหนดให้แสดงและรับข้อความในเท็กบ็อกซ์ในรูปหลายบรรทัดได้ หรือไม่ได้
ScrollBars	กำหนดให้แสดงสกรอลบาร์ด้วยหรือไม่และแสดงในแบบใด (แนวนอน แนวตั้ง หรือทั้งสองแนว)

Alignment	กำหนดให้จัดรูปแบบข้อความในคอนโทรลอย่างไร คุณสมบัตินี้จะใช้ได้ก็ต่อเมื่อคุณสมบัติ MultiLine มีค่าเป็น True
Locked	กำหนดให้ผู้ใช้แก้ไขข้อความในเท็กบ็อกซ์ ได้หรือไม่

บทที่ 3
การออกแบบ

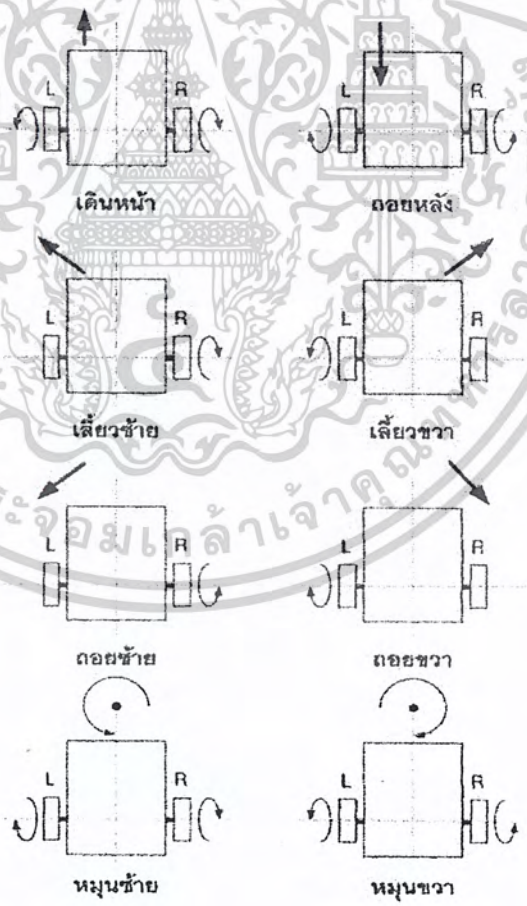
3.1 ส่วนของโครงสร้างทางเครื่องกล

ในการออกแบบโปรเจกท์นี้ได้ออกแบบหุ่นยนต์เพื่อใช้ในการเป็นต้นแบบเพื่อการเคลื่อนย้ายสิ่งของโดยการนำสิ่งของที่อยู่จากที่หนึ่งไปสู่อีกที่หนึ่งโดยตัวหุ่นยนต์สามารถเคลื่อนที่ไปตามเส้นทางที่เราสามารถกำหนดให้ได้โดยถูกต้อง

ดังนั้นส่วนสำคัญของฮาร์ดแวร์ของตัวหุ่นยนต์มีหลักๆอยู่ 2 อย่างใหญ่คือ

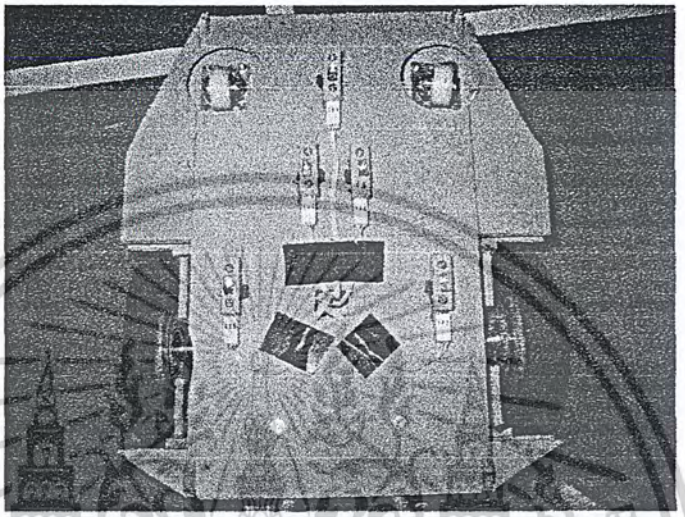
3.1.1 ส่วนของการเคลื่อนที่ของหุ่นยนต์

ส่วนของการเคลื่อนที่ของหุ่นยนต์เราใช้ DC Motor ในการขับล้อทั้งสองข้างเพื่อที่จะใช้ควบคุมหุ่นยนต์ให้เคลื่อนที่ไปข้างหน้า,เลี้ยวซ้าย,เลี้ยวขวา,หมุนซ้าย,หมุนขวา,ถอยซ้ายหรือถอยขวา



รูปที่ 3-1 แสดงทิศทางการหมุนของล้อเพื่อให้ได้ทิศทางต่างๆตามต้องการ

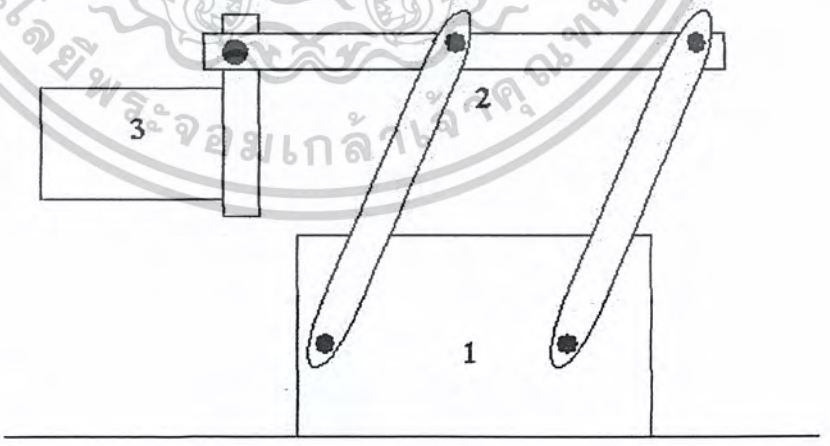
ระบบขับเคลื่อนของหุ่นในโปรเจกต์นี้ นั้นมี 4 ล้อ โดยใช้ 2 ล้อหลังเป็นตัวขับเคลื่อนโดยใช้ DC Motor 12 V. เป็นตัวขับดังรูปที่ 3-2 โดยได้มีการหุ้มล้อยางเพื่อเพิ่มประสิทธิภาพในการเกาะพื้น ส่วนสองล้อหน้าเป็นล้ออิสระสามารถเคลื่อนได้รอบทิศทาง



รูปที่ 3-2 แสดงรูปของล้อทั้ง 4 ของตัวหุ่นยนต์

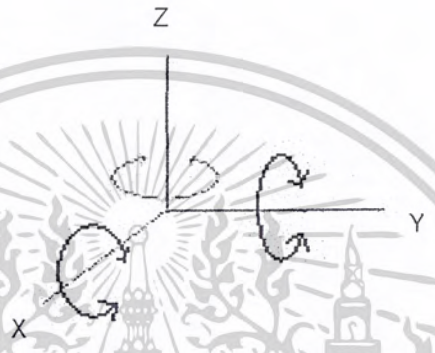
3.1.2 ส่วนของแขนกลของหุ่นยนต์

ส่วนของแขนกลของหุ่นยนต์ในโปรเจกต์นี้ใช้แบบแขนหุ่นยนต์ 3 แกน โดยจะอธิบายทีละแกนดังนี้

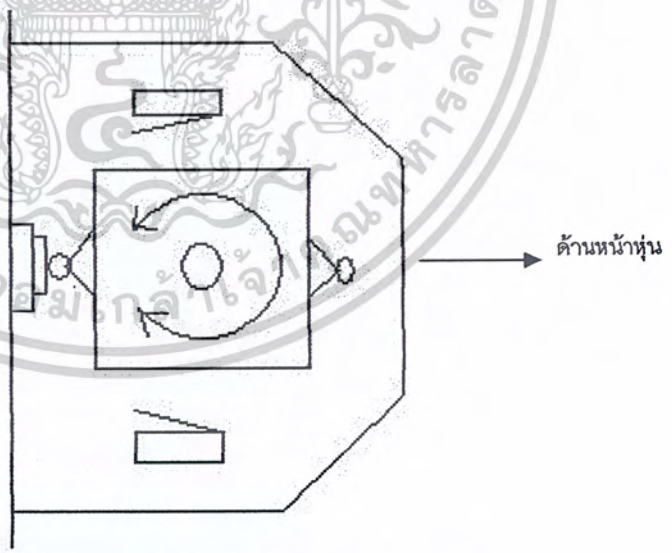


รูปที่ 3-3 รูปแสดงแขนกล 3 แกนที่ออกแบบไว้ในหุ่นยนต์

3.1.2.1 แกนที่ 1 เป็นฐานที่รองรับแขนกล ซึ่งเปรียบเสมือนต้นแขนของหุ่นหมุนได้รอบแกน Z ใช้ DC Motor เป็นตัวขับเคลื่อนโดยหมุนได้ 3 ระดับ 0 องศา 90องศา 180 องศา โดยใช้ลิมิตสวิตช์3 ตัวเป็นตัวตรวจจับตำแหน่งให้อยู่ตามองศาที่กำหนด โดยที่ลูกปืนที่อยู่ด้านหน้าของฐานแขนกลจะทำหน้าที่สัมผัสกับลิมิตสวิตช์ด้านซ้ายและขวา ส่วนลูกปืนด้านหลังจะทำหน้าที่สัมผัสกับลิมิตสวิตช์ตัวกลาง

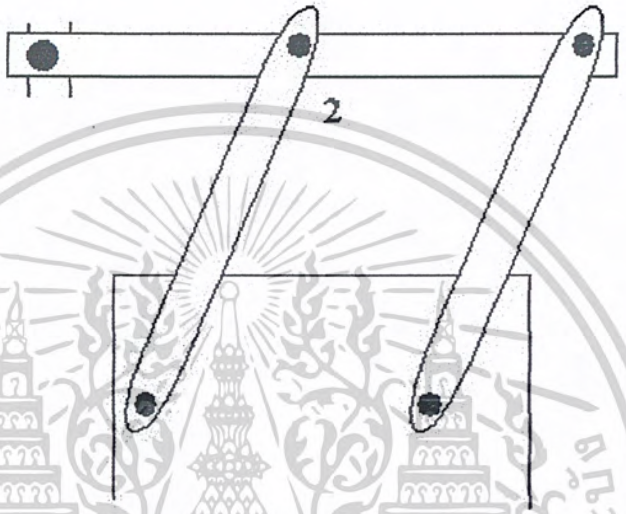


รูปที่ 3-4 รูปแสดงการหมุนรอบแกน X-Y-Z

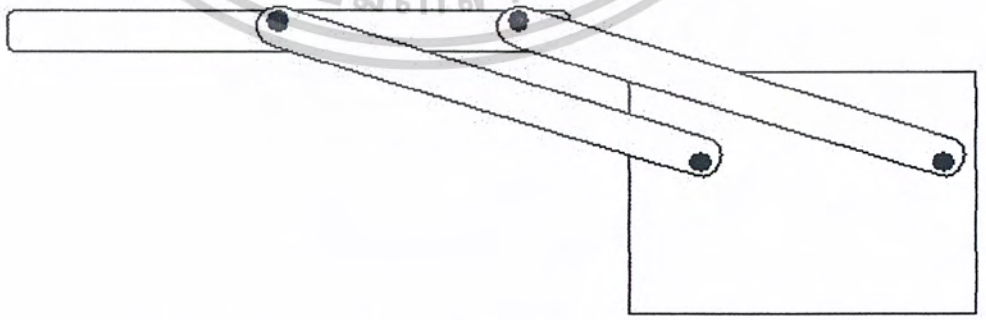


รูปที่ 3-5 รูปแสดงการสัมผัสลิมิตสวิตช์ของลูกปืน

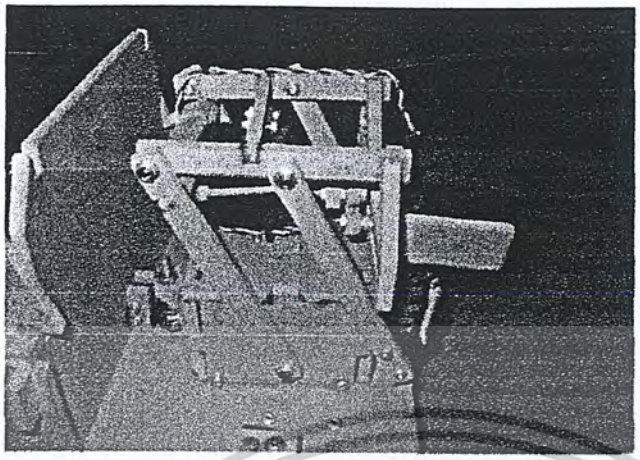
3.1.2.2 แกนที่ 2 เป็นแขนที่หมุนในทิศแกน Y เพื่อที่จะให้ส่วนของมือกลสามารถลงไปด้านหน้าเพื่อหยิบของ และยกขึ้นมาด้านบน โดยใช้เซอร์โวมอเตอร์เป็นตัวขับเคลื่อนให้หมุน และยึดไว้กับแขน โดยแขนยึดเชื่อมกันทั้ง 4 มุมของฐาน



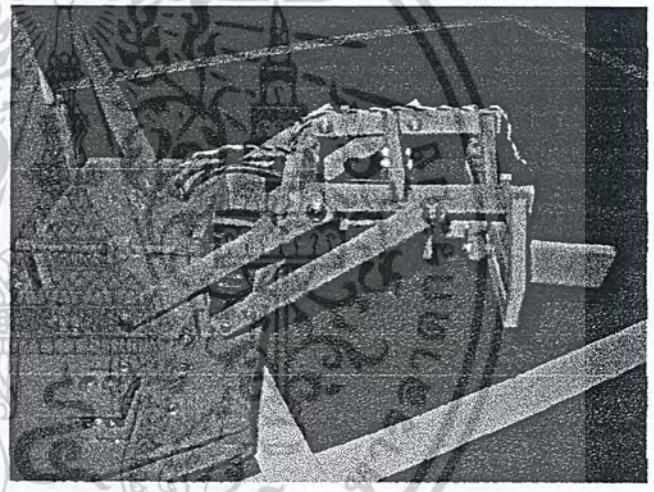
รูปที่ 3-6 รูปแสดงภาพด้านข้างของแขนกลในตำแหน่งที่เริ่มต้น



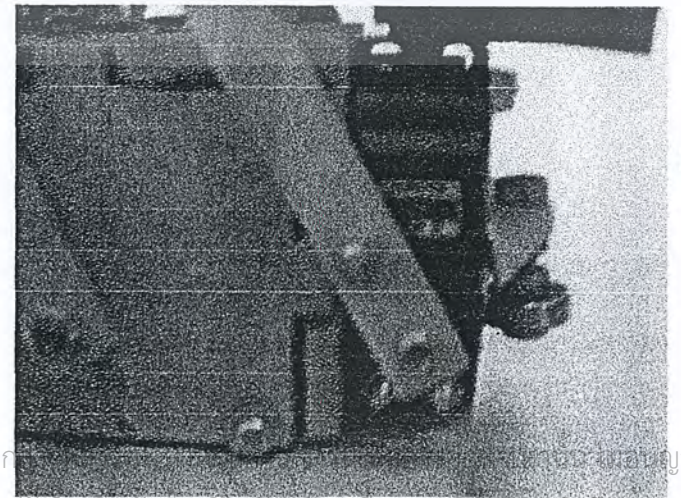
รูปที่ 3-7 รูปแสดงภาพด้านข้างของแขนกลในตำแหน่งที่หมุนลงมาด้านหน้า



รูปที่ 3-8 รูปแสดงแขนหุ่นขณะหมุนยกขึ้นมาข้างบน

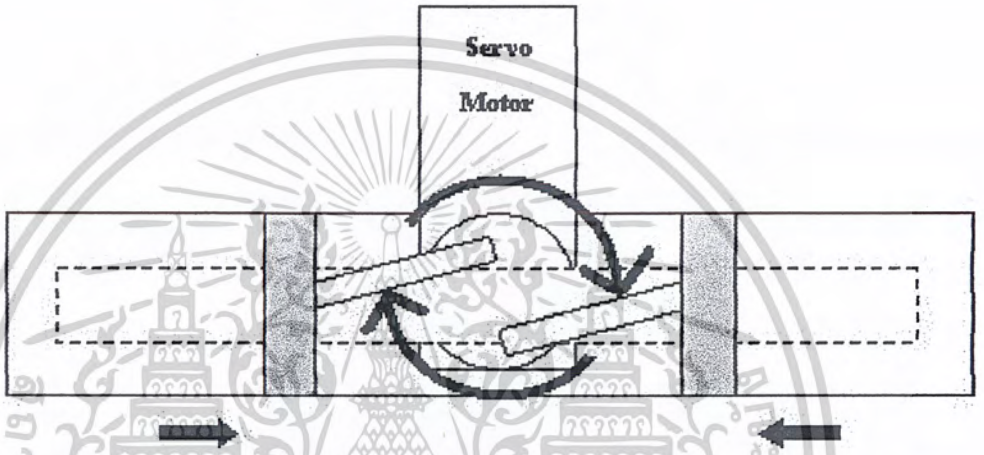


รูปที่ 3-9 รูปแสดงแขนหุ่นขณะหมุนลงมาหยิบของ

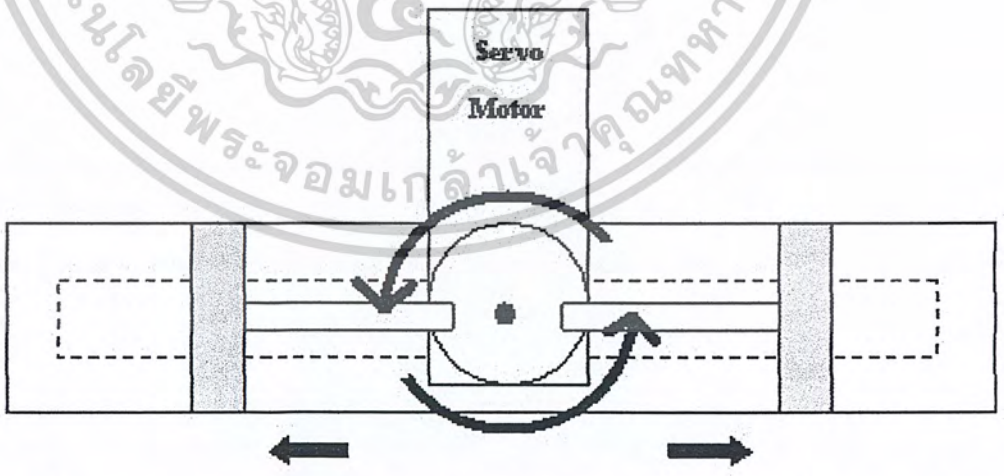


รูปที่ 3-10 รูปแสดงแขนกลยึดกับเซอร์โวมอเตอร์

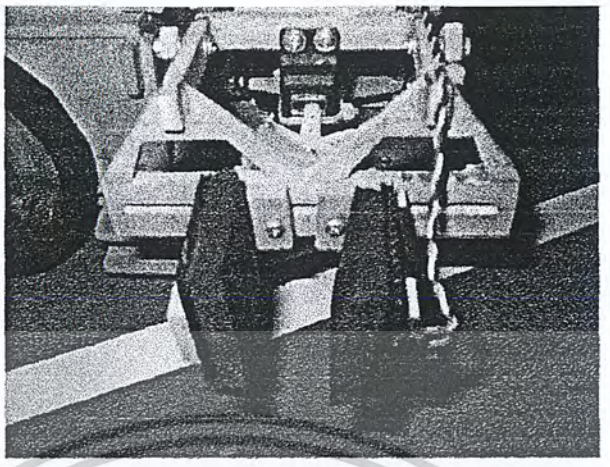
3.1.2.3 แกนที่ 3 แกนนี้เป็นส่วนสำคัญมากเพราะเป็นมือจับนั่นเอง เปรียบได้กับมือของหุ่นยนต์ทางด้านกลไกนั้นค่อนข้างจะซับซ้อน ขับเคลื่อนโดยใช้เซอร์โวมอเตอร์ (Servo Motor) และหมุนในแนวแกน X และยังมีลิมิตสวิตช์ติดอยู่ที่มือจับเพื่อเป็นตัวตรวจจับด้วยว่าได้บีบโดนวัตถุหรือไม่



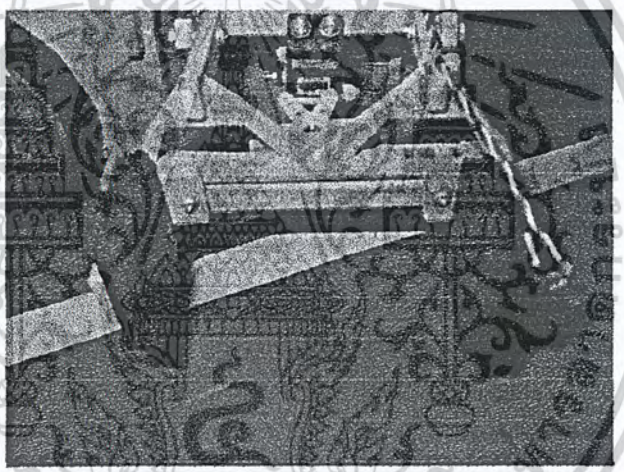
รูปที่ 3-11 รูปแสดงเซอร์โวมอเตอร์หมุนตามเข็มนาฬิกา ทำให้มือจับสไลด์บีบเข้ามา



รูปที่ 3-12 รูปแสดงเซอร์โวมอเตอร์หมุนทวนเข็มนาฬิกา ทำให้มือจับสไลด์อ้าออกไป



รูปที่ 3-13 รูปแสดงมือจับของแขนกลบีบเข้ามาสุด



รูปที่ 3-14 รูปแสดงมือจับของแขนกลอ้าออกไปสุด

3.2 ส่วนของการควบคุมของหุ่นยนต์

ในการออกแบบในส่วนของการควบคุมหุ่นยนต์นั้นในโปรเจกท์นี้ได้สร้างเป็นคอนโทรลบ็อกขึ้นมา ซึ่งในการออกแบบคอนโทรลบ็อกนี้มีอุปกรณ์ฮาร์ดแวร์ดังนี้

3.2.1ไมโครคอนโทรลเลอร์ P89C51RD2

- เข้าระบบ ISP (In System Programming) ในการดาวน์โหลดโปรแกรมเข้าสู่ตัวไมโครคอนโทรลเลอร์โดยผ่านพอร์ตอนุกรมซึ่งมาจากคอมพิวเตอร์
- พอร์ต 0.0 ถึง 0.4 รับค่าเซนเซอร์จากตัวหุ่นยนต์
- พอร์ต 0.5 ถึง0.7 รับค่ารหัส 3บิตจากพอร์ต 1.5 ถึง 1.7ไมโครคอนโทรลเลอร์89C51 ซึ่งเป็นรหัสที่ทำให้ได้รู้ว่าได้กดคีย์แพดปุ่มไหนไป

- พอร์ต 1 ทั้งหมด ส่งข้อมูลขนานไปสู่พอร์ต 3 ของไมโครคอนโทรลเลอร์ 89C51
- พอร์ต 2.0 ถึง 2.6 ส่วนนี้เป็นการส่งข้อมูลไปสู่ไมโครคอนโทรลเลอร์ 89C2051 ทั้ง 2 ตัว
- พอร์ต 3.0 และ 3.1 ส่งข้อมูลไปยัง Drive รีเลย์ เพื่อควบคุม ดีซี มอเตอร์ ของฐานแขนกล
- พอร์ต 3.2 ส่งสัญญาณไปควบคุมเซอร์โวมอเตอร์ตัวที่ควบคุมส่วนแขน
- พอร์ต 3.3 ส่งสัญญาณไปควบคุมเซอร์โวมอเตอร์ตัวที่ควบคุมส่วนของมือจับ
- พอร์ต 3.4 ถึง 3.7 รับค่าลิมิตสวิตช์ทั้ง 4 ตัวจากแขนกลเพื่อนำไปประมวลผล

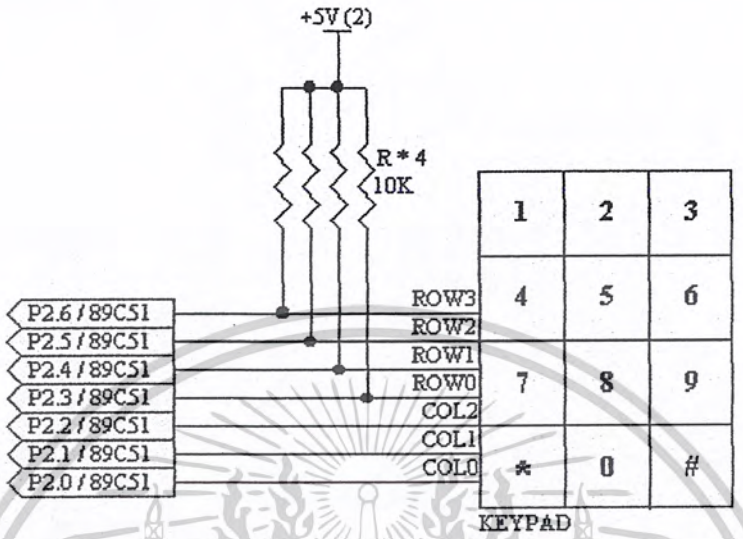
3.2.2 ไมโครคอนโทรลเลอร์ 89C51

- พอร์ต 0.0 ถึง 0.7 ส่งข้อมูลเพื่อไปแสดงผลบนจอ LCD
- พอร์ต 1.0 ติดต่อกับขา RS ของจอLCD
- พอร์ต 1.1 ติดต่อกับขา E ของจอ LCD
- พอร์ต1.2 ถึง1.3 ส่งสัญญาณไปควบคุมรีเลย์เพื่อปรับโหมดของไมโครคอนโทรลเลอร์ P89C51RD2 ว่าให้เป็นโหมดปกติหรือโหมดISP (In System Programming)
- พอร์ต 0.5 ถึง 0.7 ส่งข้อมูล 3 บิต ไปสู่ไมโครคอนโทรลเลอร์ P89C51RD2 ซึ่งเป็นรหัสที่จะทำให้รู้ว่ากดคีย์แพดปุ่มไหนไป
- พอร์ต 2.0 ถึง 2.7 เชื่อมต่อกับคีย์แพด
- พอร์ต 3.0 ถึง 3.7 รับข้อมูลจากไมโครคอนโทรลเลอร์ 89C51RD2 ซึ่งข้อมูลนี้จะนำไปแสดงผลบนจอ LCD เกี่ยวกับสถานะของเซนเซอร์

3.2.3 ไมโครคอนโทรลเลอร์ 89C512051(มี 2 ตัว ซ้ายและขวา)

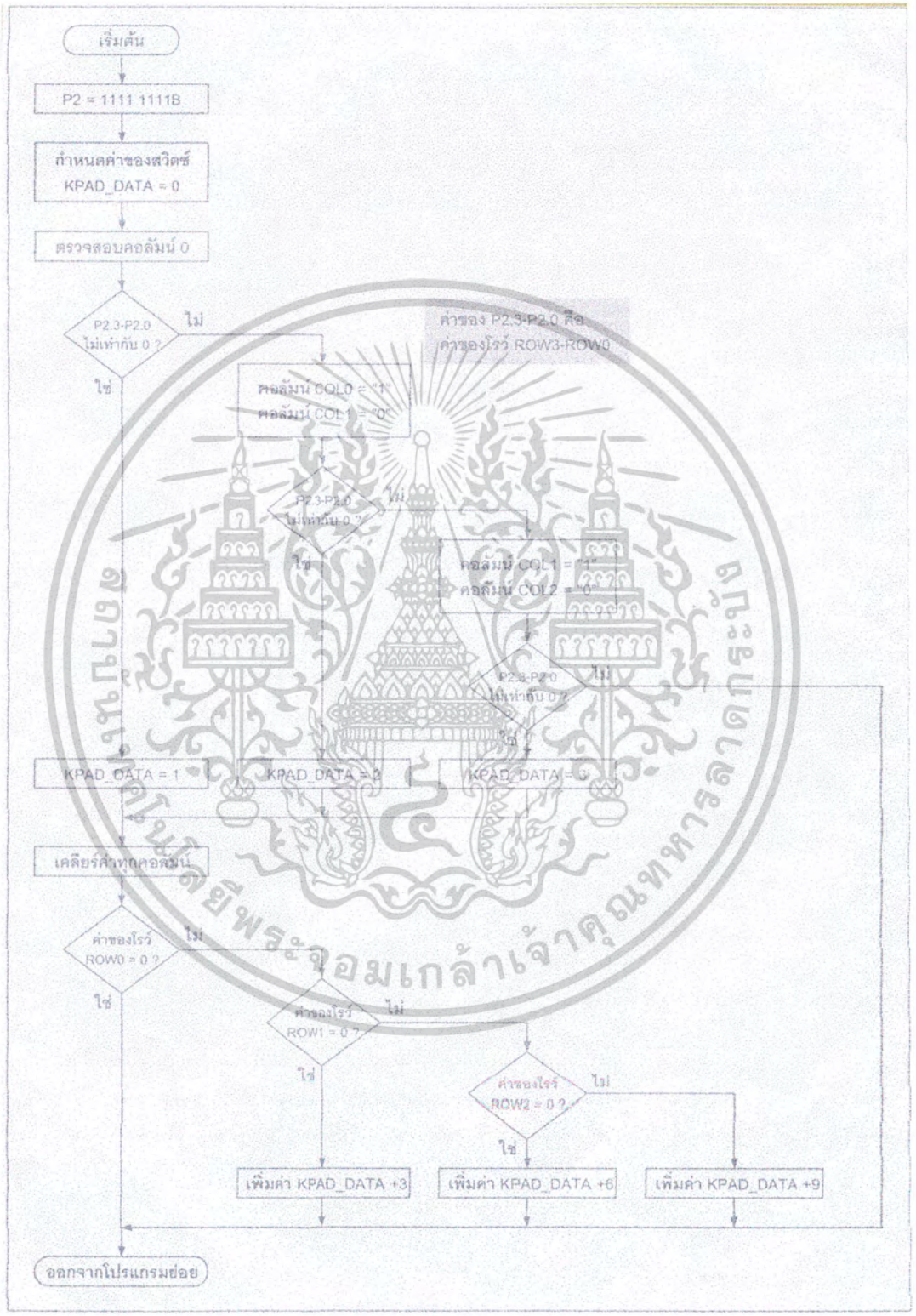
- พอร์ต 1.0 ถึง 1.6 รับข้อมูลทางด้านระดับความเร็วมอเตอร์ ,การเดินหน้าของมอเตอร์และการเลือกไมโครคอนโทรลเลอร์ซ้ายหรือขวาในการรับข้อมูลนี้จากไมโครคอนโทรลเลอร์ P89C51RD2
- พอร์ต 3.3 ถึง 3.4 ของไมโครคอนโทรลเลอร์ตัวซ้าย เพื่อส่งข้อมูลไปยัง IC L298 เพื่อไปขับมอเตอร์ตัวซ้าย
- พอร์ต 3.3 ถึง 3.4 ของไมโครคอนโทรลเลอร์ตัวขวา เพื่อส่งข้อมูลไปยัง IC L298 เพื่อไปขับมอเตอร์ตัวขวา

3.2.4 คีย์แพด (Keypad)



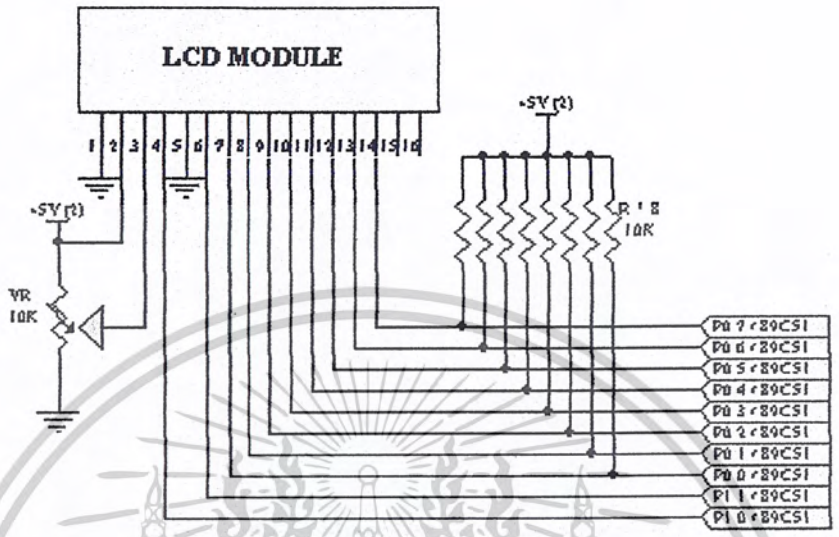
รูปที่ 3-15 รูปแสดงวงจร Keypad

จากการต่อวงจรดังรูปที่3-15 ใช้พอร์ต 2 ของไมโครคอนโทรลเลอร์ 89C51 เชื่อมต่อเข้ากับแป้นสวิตช์เมตริกทั้ง 7 เส้น คือสายของหลัก 3 สาย C0-C2 และสายทางแถวอีก 4 สาย R0-R3 โดยที่สายแถวนี้จะต่อความต้านทานพูลอัพไว้เพื่อกำหนดสถานะเริ่มต้นที่ไม่มีการกดสวิตช์โดยไมโครคอนโทรลเลอร์จะทำการส่งข้อมูล "0" ไปยังสายหลักทั้งสามตามลำดับ ในทุกครั้งที่มีการส่งข้อมูลไปยังสายหลักของแป้นสวิตช์เมตริก ไมโครคอนโทรลเลอร์จะทำการอ่านค่าที่สายแถวทุกสายเข้ามาด้วย หากไม่มีการกดสวิตช์ ค่าของสายแถวก็จะยังเป็น "1" ทั้งหมดถ้าหากมีการกดสวิตช์ ค่าของสายแถวจะไม่เป็น "1111"อีกต่อไปเป็นการบอกให้ทราบว่ามีการกดแป้นสวิตช์เมตริกขึ้นแล้ว จากนั้นไมโครคอนโทรลเลอร์ก็จะทำการค้นหาตำแหน่งสิ่งที่ได้มาจะเป็นค่าของตำแหน่งของแป้นนั้น และนำค่าที่ได้ไปใช้ในกระบวนการต่อไป



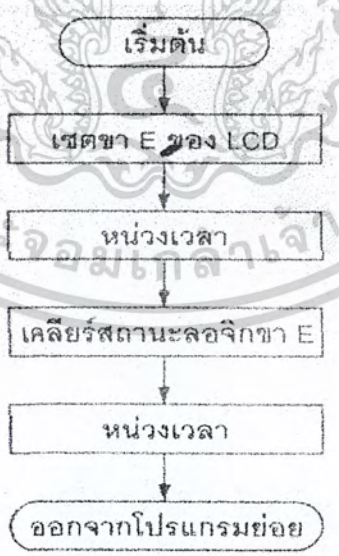
รูปที่ 3-16 โฟลว์ชาร์ตโปรแกรมย่อยการอ่านค่าคีย์แพด

3.2.5 จอแสดงผล LCD

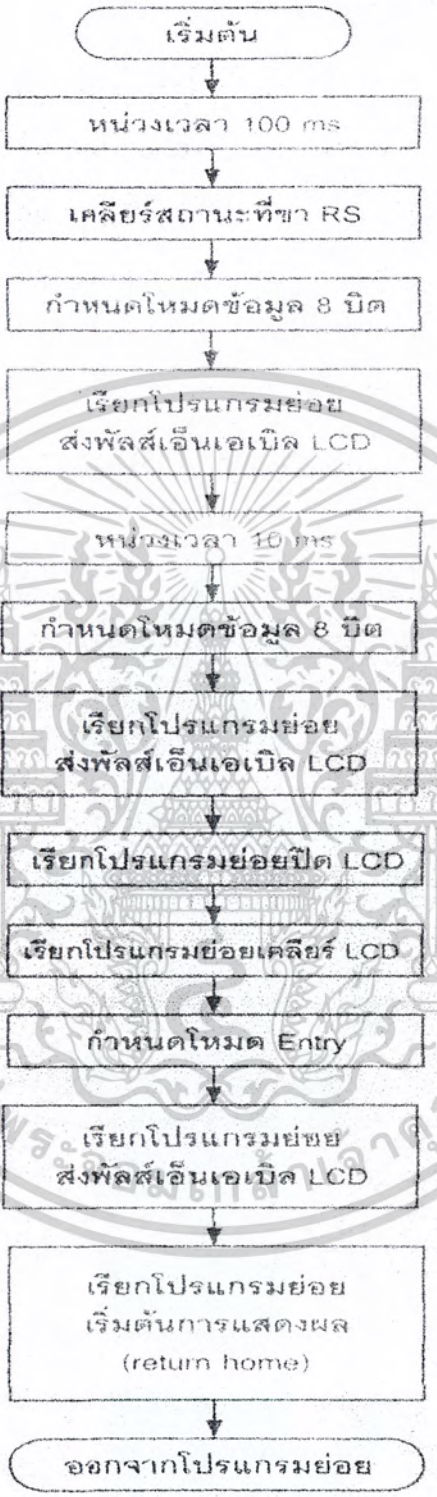


รูปที่ 3-17 รูปแสดงวงจรจอ LCD MODULE

จากการต่อวงจรดังรูปที่3-17 ใช้พอร์ต 0 ของไมโครคอนโทรลเลอร์ 89C51 เชื่อมต่อเข้ากับขา D0-D7 ของจอ LCD และให้ขา P1.0 ต่อเข้ากับขา RS และ P1.1 ต่อเข้ากับ ขา E ของจอ LCD ซึ่ง 2 ขานี้เป็นส่วนนี้จะควบคุมในการรับข้อมูลของขา D0-D7 ซึ่งได้กล่าวไว้แล้วในบทที่ 2

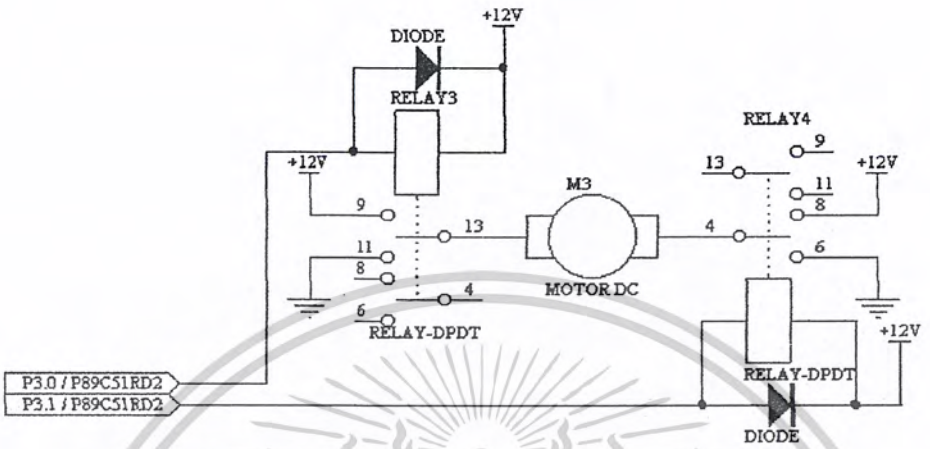


รูปที่ 3-18 โฟลวชาร์ตการส่งพัลส์เอ็นเอเบิลให้แก่โมดูล LCD



รูปที่ 3-19 โฟลวชาร์ตของการอินิเชียลโมดูล

3.2.6 วงจรขับมอเตอร์ด้วยรีเลย์



รูปที่ 3-20 รูปแสดงวงจรขับมอเตอร์ด้วยรีเลย์

จากรูปที่ 3-20 เป็นการต่อวงจรโดยใช้รีเลย์ 12V 8 ขา 2 ตัว ต่อเข้ากับมอเตอร์ตัวที่ขับเคลื่อนให้ฐานแขนกลหมุน โดยต่อเป็นวงจร H Bridge ซึ่งควบคุมด้วย 2 บิต โดยใช้ P3.0 และ P3.1 ของไมโครคอนโทรลเลอร์ P89C51RD2 ส่งสัญญาณ 2 บิตดังนี้คือ

- P 3.0 = "1" และ P3.1 = "0" มอเตอร์จะหมุนตามเข็มนาฬิกา
- P 3.0 = "0" และ P3.1 = "1" มอเตอร์จะหมุนตามทวนนาฬิกา
- P 3.0 = "1" และ P3.1 = "1" หรือ P 3.0 = "0" และ P3.1 = "0" มอเตอร์จะหยุดหมุน

3.2.7 วงจรขับมอเตอร์ด้วย IC L298

โปรเจ็กต์นี้ได้เลือกใช้ IC L298 ในการขับมอเตอร์ขับเคลื่อนหุ่นยนต์ทั้ง 2 ตัว เหตุที่เลือกใช้เพราะว่ามีความสะดวกในการใช้ มีราคาที่ไม่สูงเกินไป และยังสามารถกระแสได้สูงสุดได้ถึง 4 A เลยทีเดียว ซึ่งมีรายละเอียดการต่อแต่ละขาดังนี้คือ

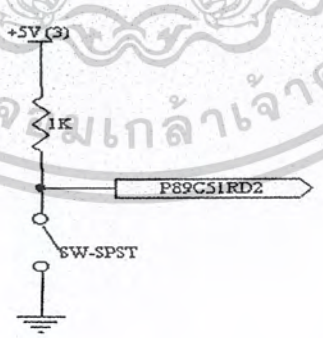
- ขา1 เป็นขา Current Sensing A ซึ่งในโปรเจ็กต์นี้ไม่ได้ใช้จึงต่อลง GND
- ขา2 เป็นขา Output 1 ต่อเข้ากับขั้วข้างหนึ่งของมอเตอร์ฝั่งซ้าย
- ขา3 เป็นขา Output 2 ต่อเข้ากับขั้วอีกข้างหนึ่งของมอเตอร์ฝั่งซ้าย
- ขา4 เป็นขา Supply Voltage Vs ซึ่งต่อกับแหล่งจ่ายไฟ 12V
- ขา5 เป็นขา Input 1 ต่อเข้ากับ P 3.4 ของไมโครคอนโทรลเลอร์ 89C2051 ตัวซ้าย

- ขา6 เป็นขา Enable A ซึ่งต้องต่อเข้ากับแหล่งจ่ายไฟ 5 V
- ขา7 เป็นขา Input 2 ต่อเข้ากับ P 3.3 ของไมโครคอนโทรลเลอร์ 89C2051 ตัวซ้าย
- ขา8 เป็นขา GND ต่อกับขา GND ของแหล่งจ่ายไฟ
- ขา9 เป็นขา Logic Supply Voltage Vss ซึ่งต้องต่อกับแหล่งจ่ายไฟ 5V
- ขา10 เป็นขา Input 3 ต่อเข้ากับ P 3.4 ของไมโครคอนโทรลเลอร์ 89C2051 ตัวขวา
- ขา11 เป็นขา Enable B ซึ่งต้องต่อเข้ากับแหล่งจ่ายไฟ 5 V
- ขา12 เป็นขา Input 4 ต่อเข้ากับ P 3.3 ของไมโครคอนโทรลเลอร์ 89C2051 ตัวขวา
- ขา13 เป็นขา Output 3 ต่อเข้ากับขั้วข้างหนึ่งของมอเตอร์ฝั่งขวา
- ขา14 เป็นขา Output 4 ต่อเข้ากับขั้วอีกข้างหนึ่งของมอเตอร์ฝั่งขวา
- ขา15 เป็นขา Current Sensing B ซึ่งในโปรเจ็กต์นี้ไม่ได้ใช้จึงต่อลง GND

โดยวิธีควบคุมมอเตอร์ทำได้ดังนี้คือ

- P 3.3 = "1" และ P3.4 = "0" มอเตอร์จะหมุนตามเข็มนาฬิกา
- P 3.3 = "0" และ P3.4 = "1" มอเตอร์จะหมุนตามทวนนาฬิกา
- P 3.3 = "1" และ P3.4 = "1" หรือ P 3.0 = "0" และ P3.1 = "0" มอเตอร์จะหยุดหมุน

3.2.8 ลิมิตสวิตช์



รูปที่ 3-21 รูปแสดงวงจรลิมิตสวิตช์

โปรเจ็กต์นี้ได้ออกแบบให้ใช้ลิมิตสวิตช์ในการจับตำแหน่งของแขนกลและเช็คการจับสิ่งของ ของมือจับ มีทั้งหมด4 ตัวซึ่งได้ผ่าน IC 74HC573 ก่อนที่จะเข้าพอร์ตของไมโครคอนโทรลเลอร์ซึ่งลิมิตสวิตช์แต่ละตัวต่อกับพอร์ตดังนี้

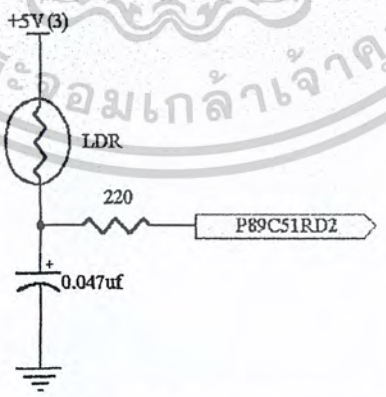
- ลิมิตสวิตช์ตัวมือจับ ต่อกับ พอร์ต 3.4 ของไมโครคอนโทรลเลอร์ P89C51RD2 เป็นลิมิตที่เช็คว่าจับโดนวัตถุหรือยัง
- ลิมิตสวิตช์ตัวขวา ต่อกับ พอร์ต 3.5 ของไมโครคอนโทรลเลอร์ P89C51RD2 เป็นลิมิตที่เช็กว่าแขนกลหมุนไปถึงด้านขวามือหรือยัง
- ลิมิตสวิตช์ตัวซ้าย ต่อกับ พอร์ต 3.6 ของไมโครคอนโทรลเลอร์ P89C51RD2 เป็นลิมิตที่เช็กว่าแขนกลหมุนไปถึงด้านซ้ายมือหรือยัง
- ลิมิตสวิตช์ตัวกลาง ต่อกับ พอร์ต 3.7 ของไมโครคอนโทรลเลอร์ P89C51RD2 เป็นลิมิตที่เช็กว่าแขนกลหมุนไปตรงกลางหรือยัง

3.2.9 เซอร์โวมอเตอร์

ในโปรเจกต์นี้ได้ใช้เซอร์โวมอเตอร์ 2 ตัว ในการควบคุมแขนกลและมือจับเนื่องจากเป็นเซอร์โวมอเตอร์สำเร็จรูปมีวงจรขับมอเตอร์และวงจรป้อนกลับอยู่ในตัวดังนั้นเราจึงสามารถต่อสายสัญญาณเข้ากับไมโครคอนโทรลเลอร์ได้เลย ซึ่งเราส่งสัญญาณพัลส์ 1 ms ถึง 3 ms จะได้องศาของมอเตอร์จาก 0 องศา ถึง 210 องศา มอเตอร์แต่ละตัวต่อกับพอร์ตดังนี้

- มอเตอร์ตัวควบคุมแขนกล ต่อกับ พอร์ต 3.2 ของไมโครคอนโทรลเลอร์ P89C51RD2
- มอเตอร์ตัวควบคุมมือจับ ต่อกับ พอร์ต 3.3 ของไมโครคอนโทรลเลอร์ P89C51RD2

3.2.10 เซนเซอร์

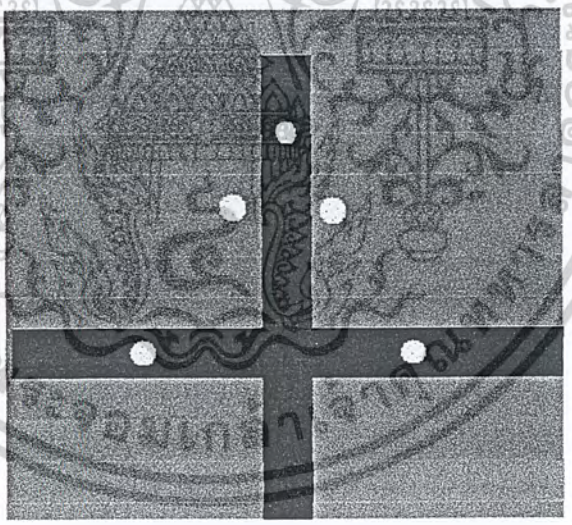


รูปที่ 3-22 รูปแสดงวงจรเซนเซอร์

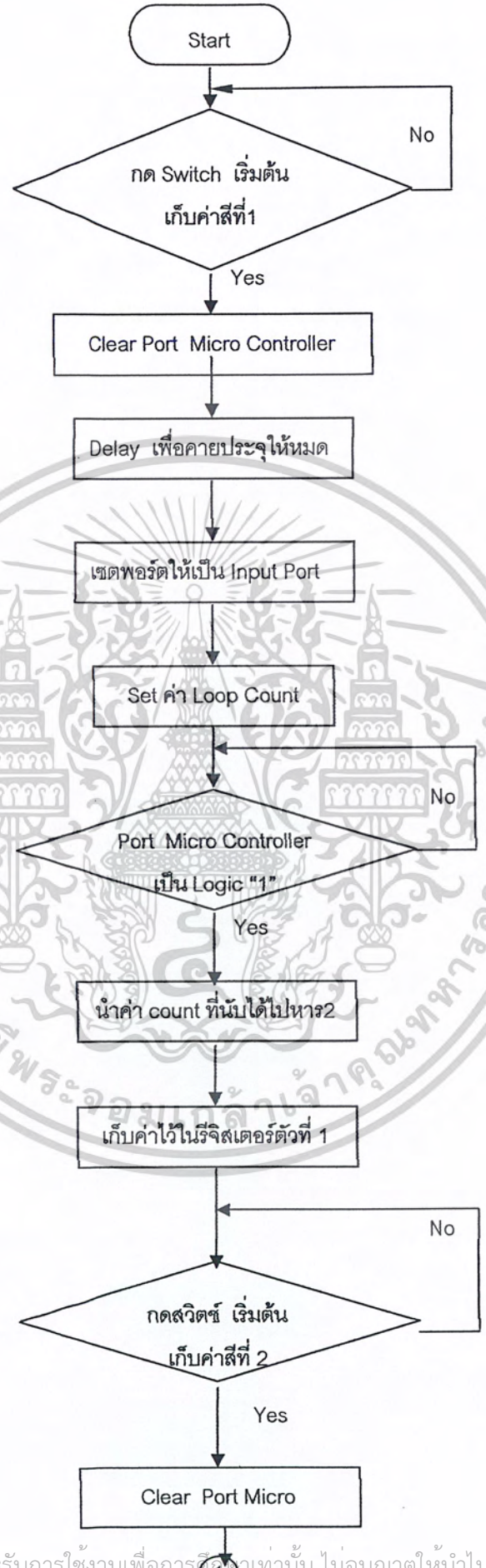
ในโปรเจกต์นี้ได้ใช้เซนเซอร์ 5 ตัวในการจับเส้นทางเดินซึ่งได้ใช้เซนเซอร์แบบ ชาร์จ และคายประจุ C ข้อดีของเซนเซอร์ชนิดนี้คือ ถ้ามีสนามหลายสนาม สีหลายสี เราไม่จำเป็นต้องปรับค่าอ้างอิงของสี จากตัวต้านทานปรับค่าอย่างเช่น เซนเซอร์ที่ใช้กันทั่วไป แต่เราสามารถที่จะจดจำ

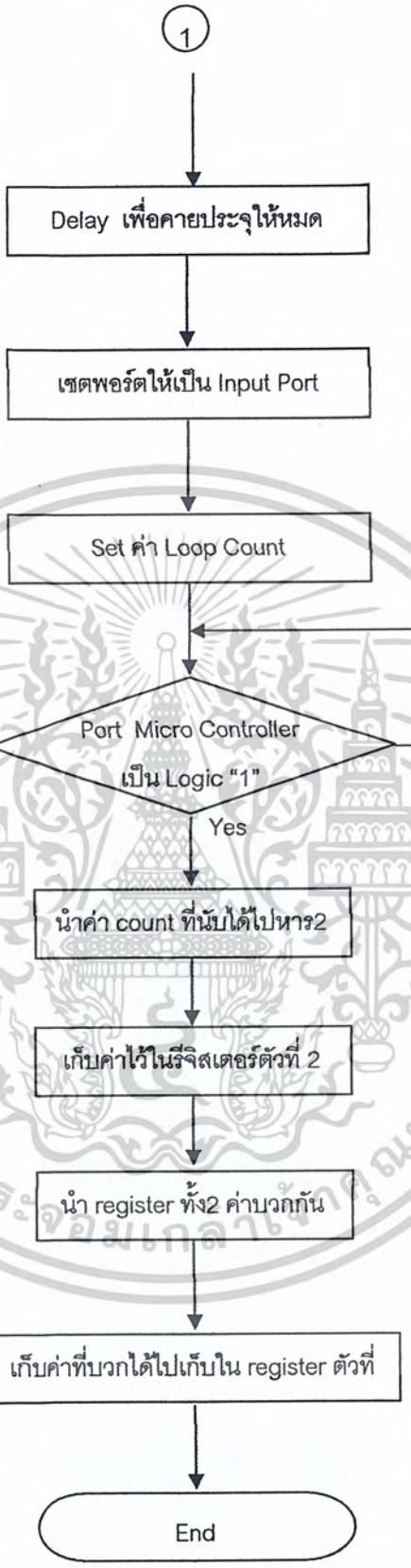
ค่าอ้างอิงของสีโดยการกดปุ่มบนคีย์แพดโดยปุ่มละสี มันจะเก็บค่าสีที่ได้ไปคำนวณในไมโครคอนโทรลเลอร์ ก็จะได้ค่าอ้างอิงของสีขึ้นมาได้ โดยเซนเซอร์แต่ละตัวต่อกับพอร์ตดังนี้

- เซนเซอร์ตัวขวาสุด ต่อกับ P0.0 ของไมโครคอนโทรลเลอร์ 89C51RD2ซึ่งมีหน้าที่ในการเช็คว่าหุ่นยนต์เจอแยกขวาหรือยัง
- เซนเซอร์ตัวขวา ต่อกับ P0.1 ของไมโครคอนโทรลเลอร์ 89C51RD2ซึ่งมีหน้าที่ในการเดินตามเส้น
- เซนเซอร์ตัวกลาง ต่อกับ P0.2 ของไมโครคอนโทรลเลอร์ 89C51RD2ซึ่งมีหน้าที่ในการเช็คว่าหุ่นยนต์เจอแยกแล้วยังมีทางอยู่ด้านหน้าหรือไม่
- เซนเซอร์ตัวซ้าย ต่อกับ P0.3 ของไมโครคอนโทรลเลอร์ 89C51RD2ซึ่งมีหน้าที่ในการเดินตามเส้น
- เซนเซอร์ตัวซ้ายสุด ต่อกับ P0.4 ของไมโครคอนโทรลเลอร์ 89C51RD2ซึ่งมีหน้าที่ในการเช็คว่าหุ่นยนต์เจอแยกซ้ายหรือยัง

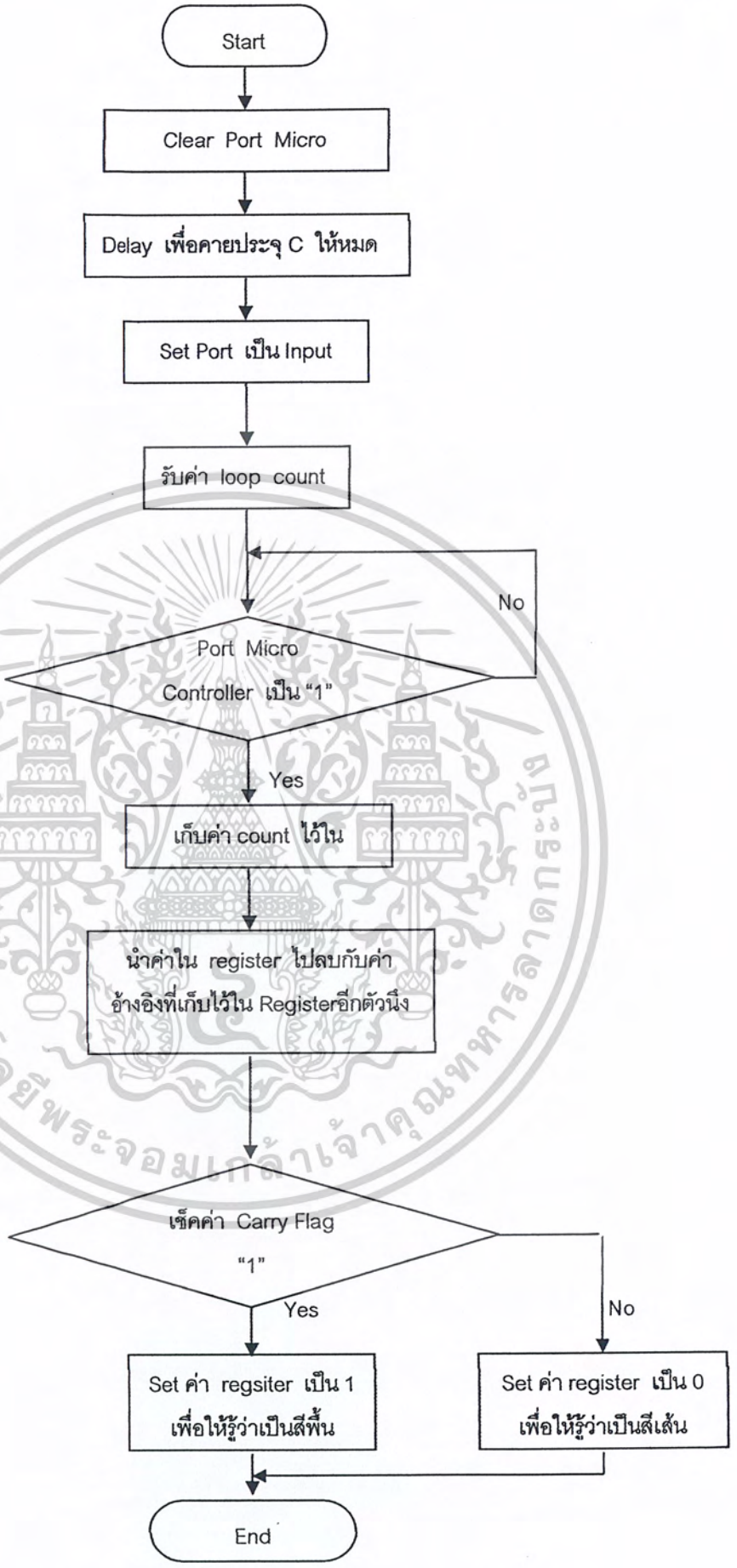


รูปที่ 3-23 รูปแสดงการวางตำแหน่งของเซนเซอร์บนหุ่นยนต์และแสดงถึงกรณีที่หุ่นยนต์วิ่งไปเจอสี่แยกแล้ว

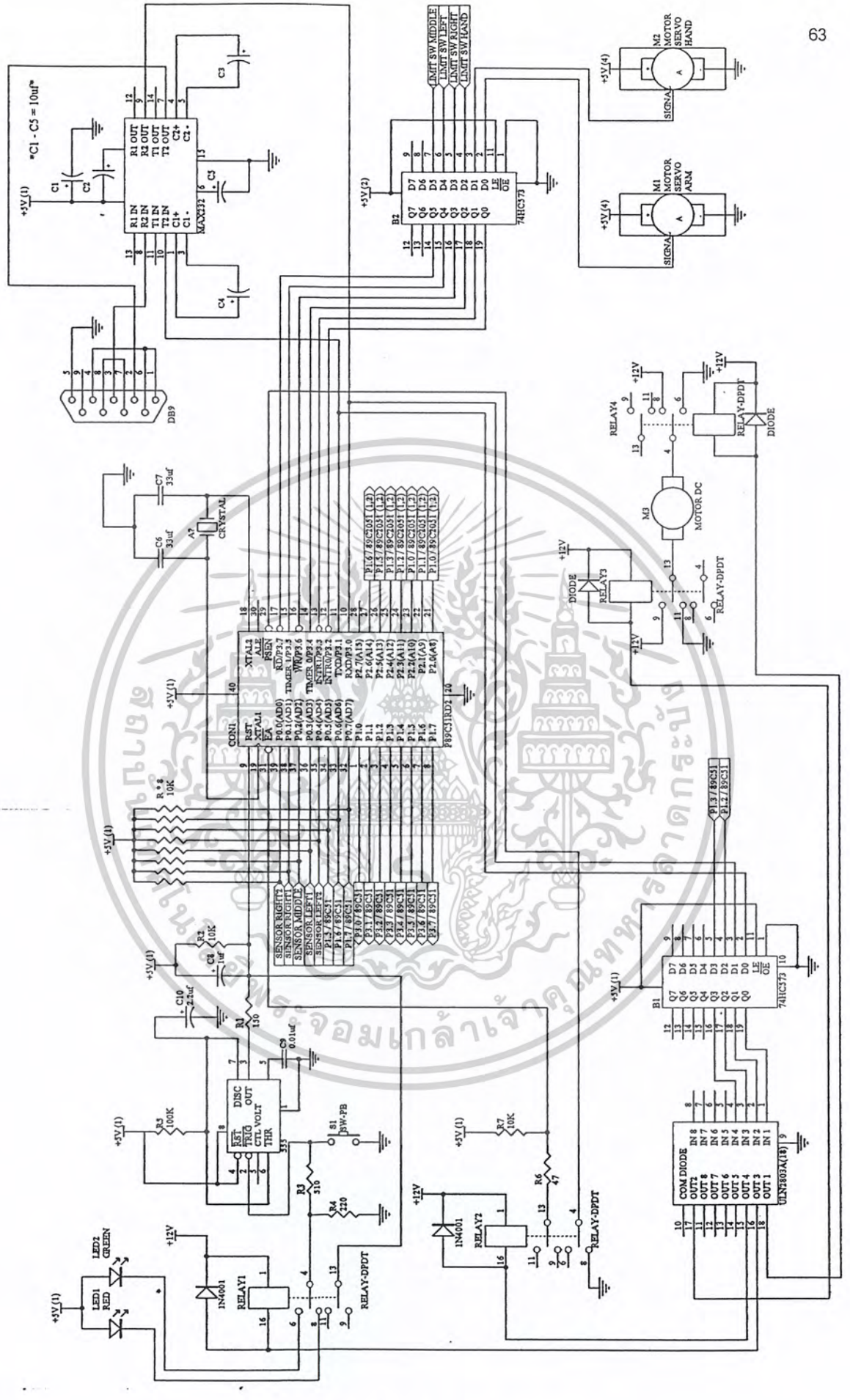




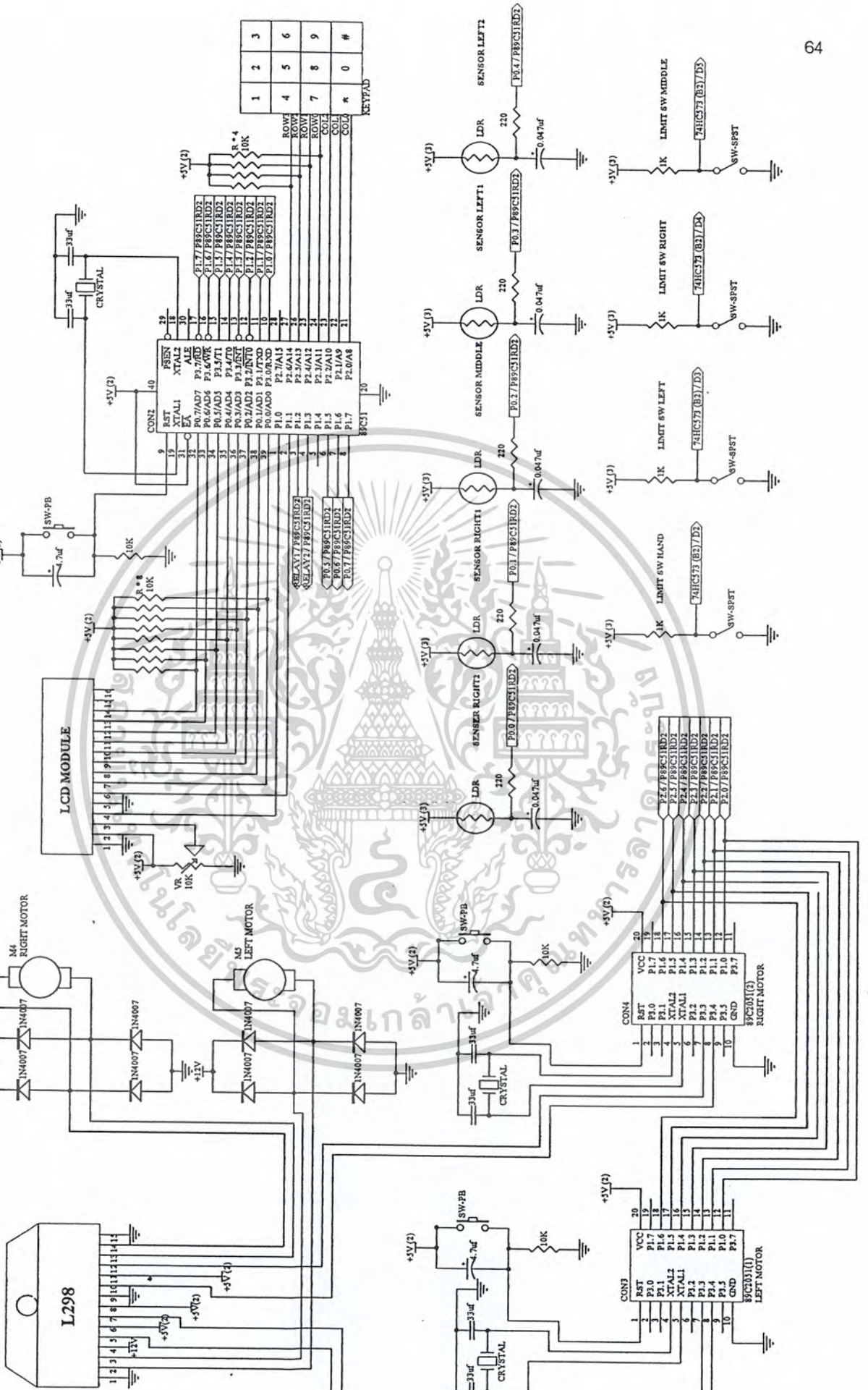
รูปที่ 3-24 โฟลว์ชาร์ตการเก็บค่าอ้างอิงระหว่างสีพื้นกับสีเส้นของเซนเซอร์



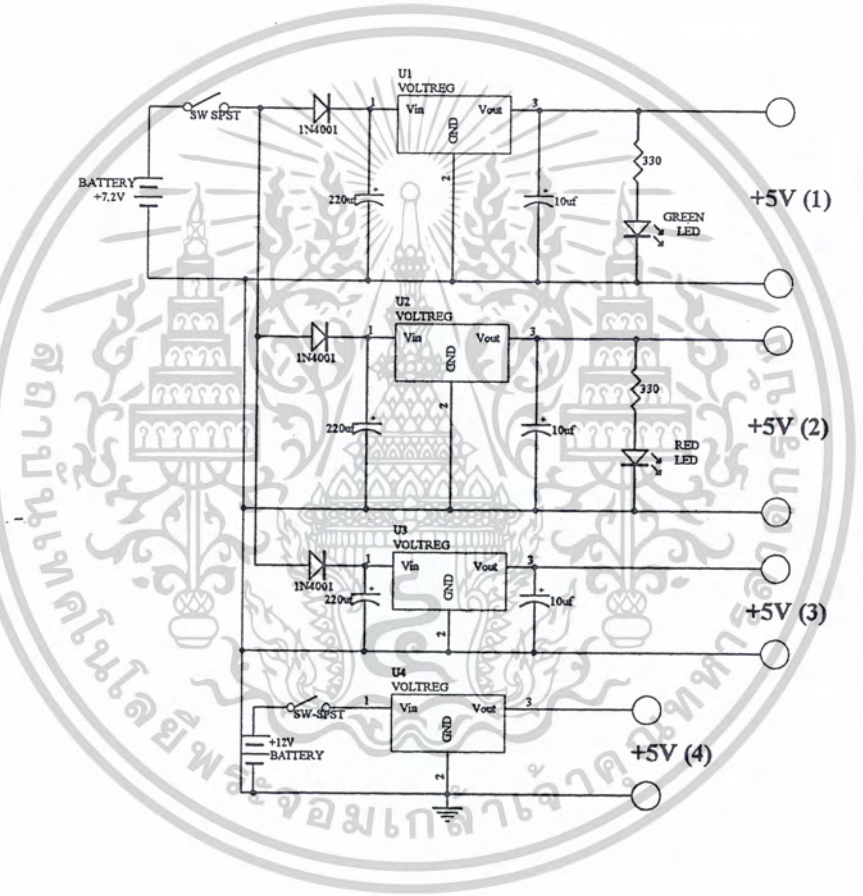
รูปที่ 3-25 โฟลว์ชาร์ตแสดงการทำงานเพื่อเช็คว่าเป็นสีพื้นหรือสีเส้น



รูปที่ 3-26 วงจรส่วนประมวลผลกลาง 1



รูปที่ 3-27 วงจรส่วนประมวลผลกลาง 2

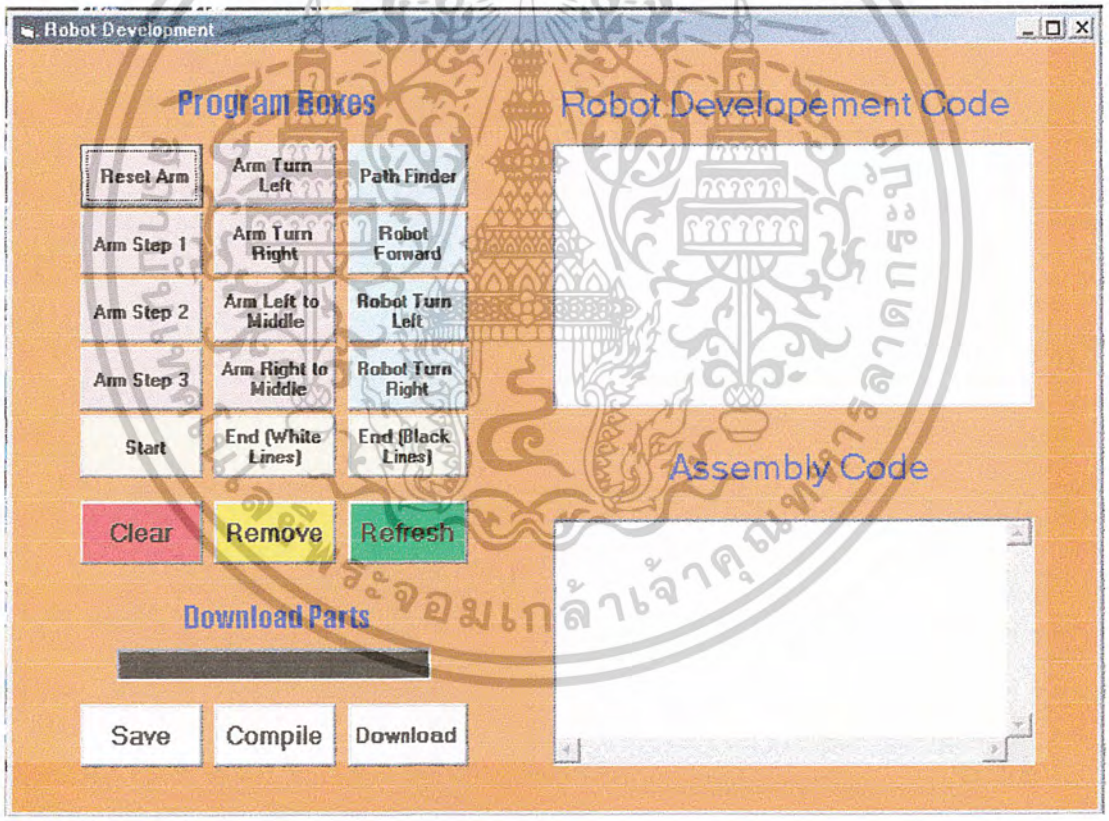


รูปที่ 3-28 วงจรแหล่งจ่ายไฟ 5 V.

3.3 ส่วนของโปรแกรมควบคุมหุ่นยนต์

ในส่วนนี้เป็นส่วนที่สำคัญที่สุดของโปรเจ็กต์นี้เนื่องจากเป็นส่วนที่ออกแบบขึ้นมาเพื่อให้คนอื่นใช้ได้โดยง่าย และหุ่นยนต์ต้องทำงานถูกต้องตามที่เขียนด้วย จึงมีความซับซ้อนในโปรแกรมพอสมควร ซึ่งมีส่วนประกอบดังนี้คือ

- Program Boxes
- Robot Development Code
- Assembly Code
- Download Parts



รูปที่ 3-29 หน้าต่างหลักของโปรแกรม Robot Development

3.3.1 Program Boxes

เป็นส่วนของคำสั่งต่างๆที่ จะนำไปใส่ใน Robot Development Code เพื่อให้หุ่นยนต์ทำงานตามโปรแกรม ซึ่งกล่องแต่ละกล่องใน Program Boxes สามารถที่จะใช้ เม้าส์คลิกได้ เมื่อคลิกแล้วคำสั่งจะไปปรากฏบนหน้าจอในส่วนของ Robot Development Code ถ้าวางเม้าส์ค้างไว้ที่กล่องแต่ละกล่องจะมีรายละเอียดของคำสั่ง สร้างมาจากCommand Button ซึ่งคำสั่งแต่ละคำสั่งมีรายละเอียดดังนี้

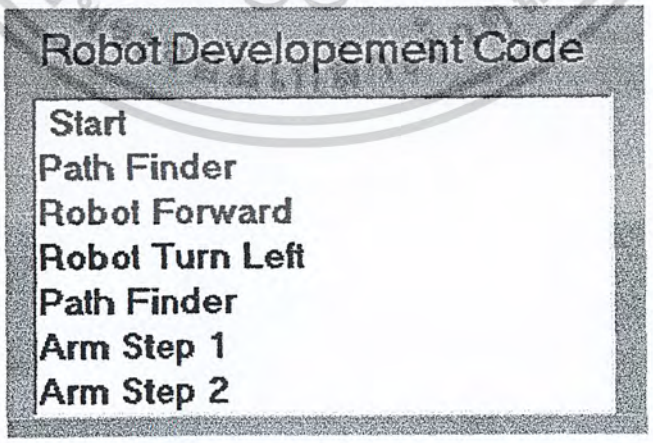
- Reset Arm เป็นคำสั่งที่จะสั่งให้แขนกลกลับเข้าไปสู่ตำแหน่งเริ่มต้น
- Arm Step 1 เป็นคำสั่งที่สั่งให้แขนกลยกขึ้นมาและอ้ามือจับออกมาสุด
- Arm Step 2 เป็นคำสั่งที่สั่งให้แขนกลเคลื่อนที่ลงไปหยิบวัตถุแล้วยกขึ้นมา
- Arm Step 3 เป็นคำสั่งที่สั่งให้แขนกลเคลื่อนที่ลงไปวางวัตถุหลังจากนั้นเก็บแขนกลเข้าตำแหน่งเริ่มต้น
- Arm Turn Left เป็นคำสั่งที่สั่งให้แขนกลหมุนไปทางตำแหน่งด้านซ้าย
- Arm Turn Right เป็นคำสั่งที่สั่งให้แขนกลหมุนไปทางตำแหน่งด้านขวา
- Arm Left to Middle เป็นคำสั่งที่สั่งให้แขนกลซึ่งอยู่ตำแหน่งด้านซ้ายหมุนกลับไปที่ตำแหน่งตรงกลาง
- Arm Right to Middle เป็นคำสั่งที่สั่งให้แขนกลซึ่งอยู่ตำแหน่งด้านขวาหมุนกลับไปที่ตำแหน่งตรงกลาง
- Path Finder เป็นคำสั่งไว้ใช้ให้หุ่นยนต์เดินตามเส้นจนกว่าจะเจอแยกก็จะหยุดรอคำสั่งต่อไป
- Robot Forward เป็นคำสั่งที่ทำหลังจากคำสั่ง Path Finder เจอทางแยกแล้วจะสั่งให้หุ่นยนต์เดินหน้า
- Robot Turn Left เป็นคำสั่งที่ทำหลังจากคำสั่ง Path Finder เจอทางแยกแล้วจะสั่งให้หุ่นยนต์เลี้ยวซ้าย
- Robot Turn Right เป็นคำสั่งที่ทำหลังจากคำสั่ง Path Finder เจอทางแยกแล้วจะสั่งให้หุ่นยนต์เลี้ยวขวา
- Start เป็นคำสั่งที่ต้องใส่ก่อนทุกครั้งเมื่อจะเริ่มโปรแกรม
- End(White Lines) เป็นคำสั่งจบโปรแกรมถ้าสนามมีเส้นทางเป็นสีขาว
- End(Black Lines) เป็นคำสั่งจบโปรแกรมถ้าสนามมีเส้นทางเป็นสีดำ

- Clear เป็นปุ่มที่ไว้ใช้เมื่อต้องการลบโปรแกรมทั้งหมดทิ้ง
- Remove เป็นปุ่มที่ไว้ใช้เมื่อต้องการลบบรรทัดที่ต้องการ
- Refresh เป็นปุ่มที่ไว้ใช้เมื่อต้องการรีเฟลช

รูปที่ 3-30 รูปแสดงส่วนของ Program Boxes

3.3.2 Robot Development Code

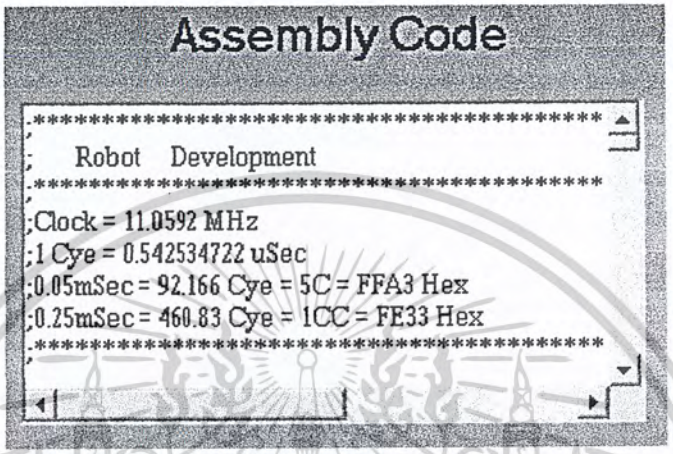
เป็นส่วนที่โชว์ลำดับการป้อนคำสั่งจาก Program Boxes สร้างมาจาก List Box



รูปที่ 3-31 รูปแสดงหน้าต่าง Robot Development
ซึ่งมีลำดับของคำสั่งจาก Program Boxes

3.3.3 Assembly Code

เป็นส่วนหน้าต่างที่แสดงถึง โค้ด Assembly ที่แปลงมาจาก ลำดับคำสั่งบนหน้าต่างของ Robot Development Code สร้างมาจาก Text Box

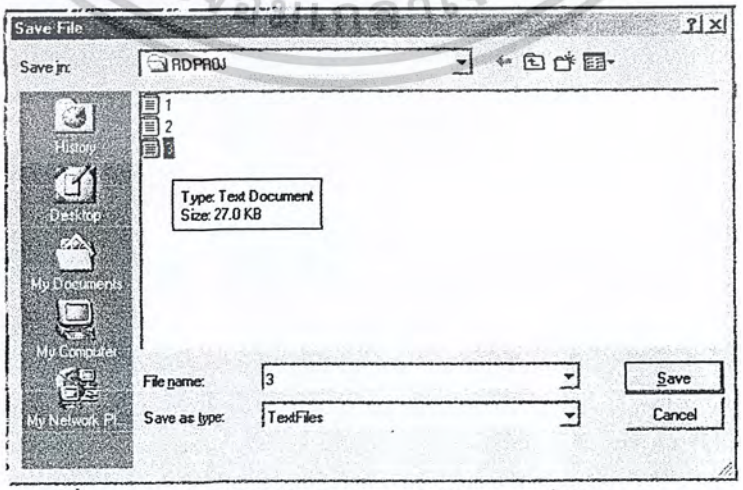


รูปที่ 3-32 รูปแสดงหน้าต่าง Assembly Code ซึ่งแปลงโค้ดมาจาก หน้าต่าง Robot Development Code

3.3.4 Download Part

เป็นส่วนที่จะนำโปรแกรมที่สร้างขึ้นมาไปเซฟในไฟล์หลังจากนั้นทำการประมวลผลและนำไป ดาวน์โหลดลงหุ่น ซึ่งมีรายละเอียดดังนี้

- Save เป็นคำสั่งที่เซฟโปรแกรมที่สร้างขึ้นมาจาก Program Boxes

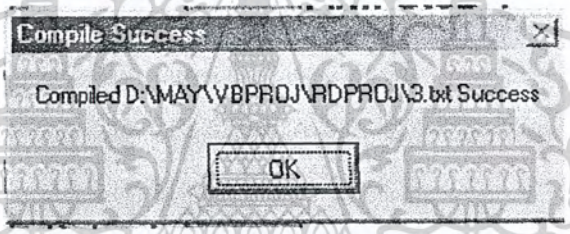


รูปที่ 3-33 รูปแสดงหน้าต่างการเซฟโปรแกรมไว้ใน Text File

หลังจากเซฟโปรแกรมแล้วไฟล์ที่เซฟไว้จะปรากฏบนหน้าต่างในส่วนของ Download Parts ซึ่งหน้าต่างสร้างมาจาก Text Box ดังรูป

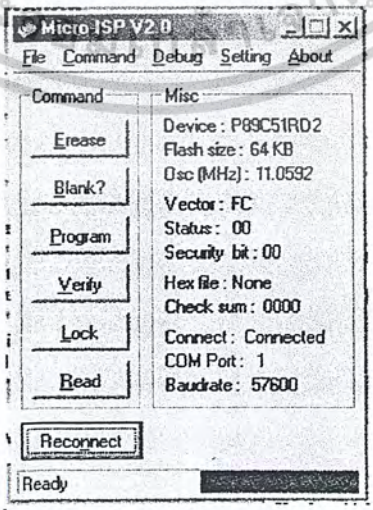
รูปที่ 3-34 รูปแสดงหน้าต่างของ Download Parts

- Compile เป็นปุ่มคำสั่งที่เมื่อกดแล้วจะนำ Text file ที่เซฟไว้ก่อนหน้านี้มาแปลงเป็น Hex file ซึ่งโปรแกรมที่ใช้แปลงคือโปรแกรม ASEM.EXE ซึ่งใช้คำสั่ง shell ของ VB 6 ในการเรียกขึ้นมาซึ่งเมื่อแปลงเสร็จจะมีหน้าต่างแสดงผลว่าแปลงเสร็จแล้ว



รูปที่ 3-35 รูปแสดงหน้าต่างแสดงผลของการแปลง Hex file

- Download เป็นปุ่มคำสั่งที่เมื่อกดปุ่มโปรแกรมจะเรียก โปรแกรมMicro ISP V2.0 ซึ่งใช้คำสั่ง shell ของ VB 6 ในการเรียกขึ้นมาซึ่งโปรแกรมดาวน์โหลดมีรายละเอียดดังนี้



รูปที่ 3-36 รูปแสดงโปรแกรม Micro ISP

รายละเอียดคำสั่งใช้งานของโปรแกรม Micro-ISP V2.0

เมนู File

ใช้ในการเรียกหรือโหลดไฟล์นามสกุล .HEX ที่ต้องการโปรแกรมลงในตัวไมโครคอนโทรลเลอร์มาเก็บไว้ในบัฟเฟอร์ของโปรแกรมรวมถึงการบันทึกข้อมูลที่อยู่ในบัฟเฟอร์ในรูปของไฟล์นามสกุล .HEX และใช้เลือกปิดโปรแกรมเมื่อต้องการใช้งาน รายละเอียดคำสั่งย่อยในเมนูนี้มีดังนี้

Load Buffer (F2) ใช้เรียกหรือโหลดไฟล์นามสกุล .HEX ที่ต้องการโปรแกรมลงในตัวไมโครคอนโทรลเลอร์มาเก็บไว้ในบัฟเฟอร์ของโปรแกรม สามารถใช้การกดปุ่ม F2 บนคีย์บอร์ดแทนการเรียกผ่านเมนูคำสั่งได้

Save Buffer As (F3) ใช้บันทึกข้อมูลที่อยู่ในบัฟเฟอร์ในรูปของไฟล์นามสกุล .HEX สามารถใช้การกดปุ่ม F2 บนคีย์บอร์ดแทนการเรียกผ่านเมนูคำสั่งได้

Exitt (Alt+F4) ออกจากโปรแกรม

เมนู Command

เป็นเมนูคำสั่งสำหรับติดต่อกับหน่วยความจำโปรแกรมภายในไมโครคอนโทรลเลอร์ มีรายละเอียดของคำสั่งย่อย ดังนี้

Erase (F5) ลบข้อมูลของหน่วยความจำโปรแกรมภายในไมโครคอนโทรลเลอร์

Blank check ตรวจสอบข้อมูลว่างของหน่วยความจำโปรแกรม

Program chip (F6) โปรแกรมหรือเขียนข้อมูลลงในหน่วยความจำโปรแกรม

Verify chip (F7) ตรวจสอบข้อมูลในหน่วยความจำโปรแกรมกับบัฟเฟอร์ ใช้ตรวจสอบความถูกต้องของการโปรแกรม

Read chip (F8) อ่านข้อมูลจากหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ในกรณีที่ไม่มีการป้องกันมาเก็บไว้ในหน่วยความจำบัฟเฟอร์

Lock chip (F9) ใช้ป้องกันการอ่านข้อมูลในหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ เมื่อทำการคลิ๊กจะปรากฏหน้าต่างเลือกระดับการป้องกันดังในรูปที่.....

Reconnect สั่งให้โปรแกรมทำการติดต่อกับไมโครคอนโทรลเลอร์อีกครั้ง (ไม่มีในเมนู Command แต่สามารถเลือกใช้ได้จากกรอบ Command บนหน้าต่างหลักของโปรแกรม อยู่ในตำแหน่งมุมล่างซ้ายมือ)

เมนู Setting

ใช้ในการกำหนดค่าพารามิเตอร์ก่อนการโปรแกรม ไม่ว่าจะเป็นเลือกเบอร์ที่ต้องการโปรแกรมเลือกพอร์ดอนุกรมที่ทำการติดต่อ, เลือกอัตราเร็วในการถ่ายทอดข้อมูล, เลือกความถี่ของสัญญาณนาฬิกาหลัก มีรายละเอียดของคำสั่งย่อยดังนี้

Preference กำหนดค่าพารามิเตอร์ที่ต้องใช้ในการติดต่อระหว่างโปรแกรม Micro- ISP กับตัวไมโครคอนโทรลเลอร์ ประกอบด้วย

Device เลือกเบอร์ของไมโครคอนโทรลเลอร์ มีด้วยกัน 8 เบอร์ คือ P89C51RA+,P89C51RB+, P89C51RC+, P89C51RD+, P89C51RA2, P89C51RB2, P89C51RC2และ P89C51RD2

Oscillator เลือกความถี่ของสัญญาณนาฬิกาหลัก ซึ่งก็คือความถี่ของคริสตอลที่ใช้นั่นเองสามารถเลือกได้จากค่าที่กำหนดไว้แล้วคือ 7.3728MHz,11.0592MHz,14.7456MHz,18.432MHz,22.1184MHz หรือค่าอื่นที่ต้องการ โดยคลิ๊กไปที่ other จะปรากฏหน้าต่างสำหรับป้อนค่าความถี่สัญญาณนาฬิกาหลัก สามารถกำหนดได้ตั้งแต่ 1-40MHz แต่ไม่ควรป้อนมากกว่า 20MHz สำหรับ P89C51RB2/RC2 และ RD2

Full chip operation เป็นการเลือกให้ไมโครคอนโทรลเลอร์ P89C51Rx2 ทำงานอย่างสมบูรณ์ในทุกๆขั้นตอนของการติดต่อกับหน่วยความจำโปรแกรม ไม่ว่าจะเป็นการอ่าน,เขียน,ลบ หรือตรวจสอบข้อมูลกับบัฟเฟอร์ ในกรณีลบ จะลบทั้งข้อมูลและบิตป้องกันด้วย ในกรณีเขียนหรือโปรแกรม ก็จะโปรแกรมข้อมูลลงในหน่วยความจำทั้งหมด โดยไม่สนใจว่าขนาดของไฟล์ที่นำมาโปรแกรมนั้นจะมีความยาวแค่ไหน และเมื่อโปรแกรมแล้วก็จะทำการตรวจสอบทุกๆ แอดเดรส ทุกๆไบต์ในทันที ทำให้การโปรแกรมด้วยฟังก์ชั่นนี้จะใช้เวลาเท่ากันหมด ไม่ว่าโปรแกรมจะยาวหรือสั้น ดังนั้นหากต้องการลดเวลาในการโปรแกรมควรไม่เลือกความสามารถนี้ได้

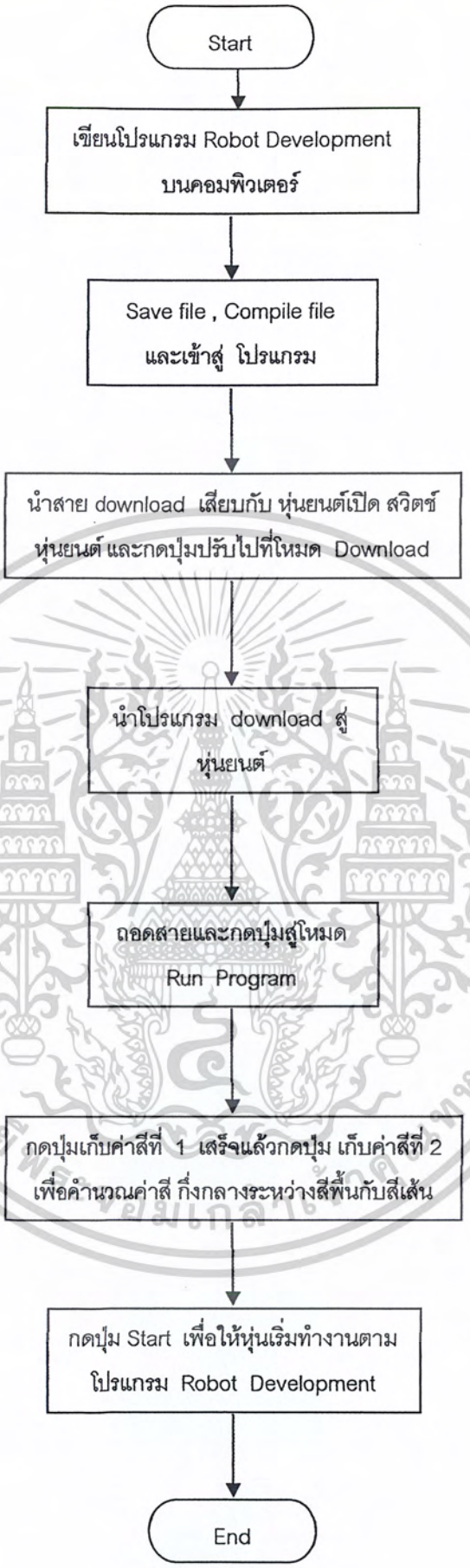
Serial port ใช้เลือกพอร์ตอนุกรมที่ทำการติดต่อกับไมโครคอนโทรลเลอร์ในการโปรแกรมแบบ ISP มีให้เลือกทั้งสิ้น 4 พอร์ต คือ Com1-Com4 โดยโปรแกรมจะมีการตรวจสอบว่าเครื่องคอมพิวเตอร์มีพอร์ตอนุกรมที่สามารถใช้งานได้ที่พอร์ตใดบ้างอย่างอัตโนมัติ

Baud rate (bps) ใช้เลือกอัตราเร็วในการถ่ายทอดข้อมูล มีหน่วยเป็นบิตต่อวินาที (bit per second : bps) สามารถเลือกได้ 6 ระดับคือ 9600,14400,19200,28800,38400,57600 และ 115200 บิตต่อวินาที

Always on top ใช้ในการกำหนดให้หน้าต่างหลักของโปรแกรม Micro-ISP ได้อย่างสะดวกมากขึ้น

ในกรณีต้องการเขียนข้อมูลหรือโปรแกรมลงในหน่วยความจำ

1. เรียกไฟล์นามสกุล .HEX ที่ต้องการโปรแกรมขึ้นมา
2. ทำการลบข้อมูลก่อนการโปรแกรมใหม่ทุกครั้ง โดยการกดปุ่ม Erase สังเกตสถานะการทำงานที่แถบแสดงสถานะทางด้านล่างหน้าต่างหลักของโปรแกรม Micro-ISP
3. จากนั้นทำการโปรแกรมข้อมูล โดยกดปุ่ม Program สังเกตสถานะการทำงานที่แถบแสดงสถานะทางด้านล่างหน้าต่างหลักของโปรแกรม Micro-ISP

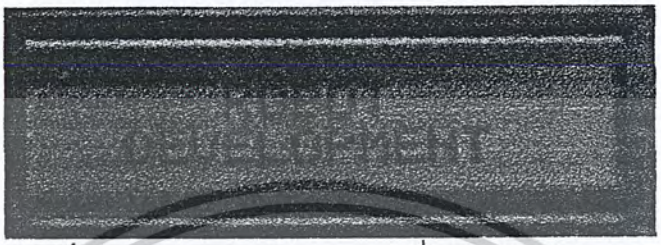


รูปที่ 3-37 โฟลว์ชาร์ตขั้นตอนการทำงานระบบรวมของ Robot Development

3.4 วิธีการใช้งานของหุ่นยนต์

ก่อนจะเริ่มใช้หุ่นยนต์จำเป็นต้องดาวน์โหลดโปรแกรมก่อนซึ่งมีขั้นตอนดังนี้

ขั้นตอนที่ 1 เปิดสวิตช์ Power ซึ่งมี สองตัว ตัวบนเป็นสวิตช์แหล่งจ่าย 7.2 V. ตัวล่างเป็นสวิตช์ 12 V. หน้าจอ LCD จะขึ้นหน้าจอว่า"Robot Development"



รูปที่3-38 รูปแสดงหน้าจอ LCD เมื่อเปิดสวิตช์ Power

ขั้นตอนที่ 2 กดปุ่ม PGM บนคีย์แพดซึ่งเป็นปุ่มเปลี่ยนโหมดเป็นโหมดดาวน์โหลด ซึ่งเมื่อกดแล้วจะมีข้อความว่า"Program Mode Please Download" เสร็จแล้วเรานำสายดาวน์โหลดมาเสียบกับหุ่นยนต์แล้วนำโปรแกรมที่กล่าวข้างต้นมาดาวน์โหลดลงหุ่นยนต์

รูปที่3-39 รูปแสดงหน้าจอ LCD เมื่อกดปุ่ม PGM

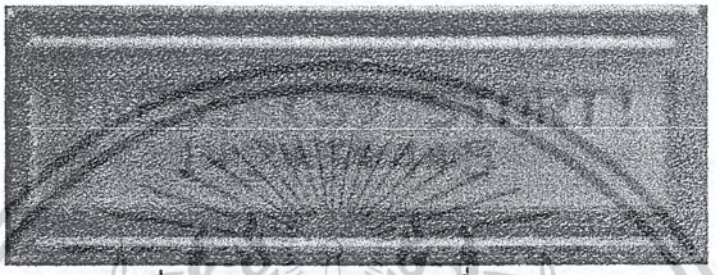
จากนี้จะพูดถึง หลังจากเขียนโปรแกรมเสร็จแล้วนำโปรแกรมดาวน์โหลดลงหุ่นแล้วจะเซตหุ่นให้เริ่มทำงานได้ ซึ่งมีรายละเอียดดังนี้

ขั้นตอนที่ 1 กดปุ่มRun เพื่อปรับโหมดจากโหมดดาวน์โหลดให้เป็นโหมดรันโปรแกรม ซึ่งจะปรากฏข้อความว่า"Running Mode"

รูปที่3-40 รูปแสดงจอLCD เมื่อกดปุ่ม RUN

ขั้นตอนที่ 2 นำหุ่นไปวางบนพื้นที่เป็นสีเส้น หลังจากนั้นกดปุ่ม C1 เพื่อให้เซนเซอร์เก็บค่าและคำนวณค่าสีเส้น

ขั้นตอนที่ 3 นำหุ่นไปวางบนพื้นที่เป็นสีพื้น หลังจากนั้นกดปุ่ม C2 เพื่อให้เซนเซอร์เก็บค่าและคำนวณค่าสีพื้น หลังจากนั้น จอ LCD จะขึ้นข้อความว่า"Press (S) Start !" และบรรทัดที่ 2 จะแสดงสถานะเซนเซอร์1-5 นับจากตัวซ้ายไปขวาถ้ามีเครื่องหมาย " * " หมายความว่าเซนเซอร์ตัวนั้นเช็คเจอเส้น

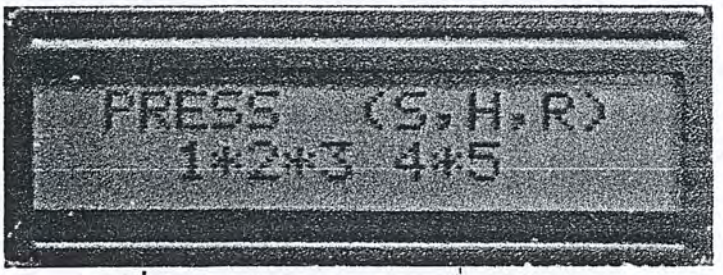


รูปที่ 3-41 รูปแสดงจอLCD เมื่อกดปุ่ม C2

ขั้นตอนที่ 4 สามารถเลือกกดได้ 3 ปุ่ม คือ Reset ,H.R.,Str. ถ้ากดปุ่มReset จะเป็นการรีเซตโปรแกรม ถ้ากดปุ่ม H.R. จะเป็นการรีเซตตำแหน่งของแขนกล ถ้ากดปุ่ม Str จะเป็นการเริ่มทำงานโปรแกรมของหุ่นยนต์

รูปที่ 3-42 รูปแสดงจอLCD เมื่อกดปุ่ม Str

ขั้นตอนที่ 5 เมื่อหุ่นยนต์ทำโปรแกรมจนจบต้องกดปุ่ม CCL. เพื่อให้หุ่นรู้ว่าหยุดทำโปรแกรม และหุ่นก็จะรอกดปุ่ม Str. อีกครั้ง ซึ่งหน้าจอ LCD จะแสดงว่า "Press (S,H,R)"



รูปที่ 3-43 รูปแสดงจอ LCD เมื่อกดปุ่ม CCL.

บทที่ 4
การทดลองและผลการทดลอง

จากการจำลองสนามซึ่งได้ออกแบบมาซึ่งจะได้ทางเดินครบทุกกรณีดังภาพ

รูปที่ 4-1 ผังจำลองสนามในการใช้ของหุ่นยนต์

การทดลองเราจำลองเส้นทางเดินให้ได้ดังรูปต่อไปนี้

รูปที่ 4-2 รูปแสดงเส้นทางเดินที่ให้หุ่นยนต์เดินไป

เราทำการทดลองให้หุ่นยนต์เริ่มที่จุด Start วิ่งตามเส้นทางที่กำหนดไว้ไปจนถึงจุดที่สิ่งของวางอยู่แล้วสั่งให้หุ่นยนต์หยิบวัตถุจากทางขวาของหุ่นมาวางไว้ทางซ้ายของหุ่นโดยเราต้องเขียน Robot Development Code ดังนี้

Start
Path Finder
Robot Turn Left
Path Finder
Robot Turn Right
Path Finder
Robot Forward
Path Finder
Arm Turn Right
Arm Step 1
Arm Step 2
Arm Turn Left
Arm Step 3
Arm Left To Middle
End (Black Lines)

ผลการทดลอง

หุ่นยนต์สามารถทำงานตามเป้าหมายได้ถูกต้องโดยสามารถเดินตามเส้นทางที่ต้องการได้ และเมื่อเราลองให้เดินเส้นทางแบบอื่นหุ่นยนต์ก็สามารถทำงานได้ถูกต้องเช่นกัน

บทที่ 5
บทวิจารณ์และสรุป

ปริญญานิพนธ์ฉบับนี้มีวัตถุประสงค์ในการที่จะพัฒนาโปรแกรมในการควบคุมหุ่นยนต์ให้เป็นภาษาที่ใช้งานง่ายบุคคลทั่วไปก็สามารถใช้งานได้ซึ่งหุ่นยนต์ต้นแบบที่สร้างขึ้นมานั้น ทางผู้จัดทำพยายามที่จะสร้างให้หุ่นยนต์มีฟังก์ชันในการควบคุมฮาร์ดแวร์ให้ได้อุปกรณ์มากชิ้นที่สุดเท่าที่งบทางคณะจะจัดให้ได้ซึ่งข้อนี้ยังเป็นข้อจำกัดในการออกแบบหุ่นยนต์ โปรเจกต์นี้ยังสามารถพัฒนาให้มีประสิทธิภาพสูงขึ้นได้ถ้าปัจจัยทางด้านงบประมาณและเวลามีมากขึ้น

สรุปผล

จากผลการทดลองโดยรวมถือว่าได้ผลตามเป้าหมายที่ได้วางไว้ กล่าวคือ โปรแกรมที่สร้างไว้สามารถเขียนเพื่อควบคุมหุ่นยนต์ให้หุ่นยนต์ทำงานได้อย่างมีประสิทธิภาพ หุ่นยนต์สามารถที่จะวิ่งตามเส้นทางที่กำหนดไว้ได้ แขนกลสามารถขยับตามที่สั่งไว้ได้ คุณภาพในการเช็กสีของเซนเซอร์อยู่ในขั้นที่น่าพอใจ

ในสิ่งที่ทางผู้จัดทำเห็นว่าควรมีการพัฒนาขึ้นคือส่วนของโปรแกรมควรที่จะมีรายละเอียดในการควบคุมมากกว่านี้ ระบบขับเคลื่อนของหุ่นยนต์ควรจะขับเคลื่อนให้ได้ตำแหน่งที่แม่นยำกว่านี้ก็จะสามารถทำให้หุ่นยนต์มีประสิทธิภาพมากขึ้นได้

ภาคผนวก
แสดงโปรแกรมการทำงาน

1. ไมโครคอนโทรลเลอร์ 89C2051 ตัวขวา

```
;*************************************************************************
;                Robot    Development
;*************************************************************************
;Filename       projrunr
;Description    Robot  Program
;Assemble       ASEM
;Software Eng.  Metee Nilparak
;*************************************************************************
;Clock = 11.0592MHz
;1 Cye = 1.085 uSec
;0.1mSec = 92.166 Cye = 5C = FFA3 Hex
;0.5mSec = 460.83 Cye = 1CC = FE33 Hex
;*************************************************************************
;                อ้างตัวแปรต่างๆ
;*************************************************************************
        VLEVEL          EQU   15D
        RSPEED          EQU   2BH
        TIME    EQU     24H

;*************************************************************************
;                INITIAL  STATUS
;*************************************************************************
        ORG     0000H
        JMP     START
```

```
        ORG   000BH             ;INTERRUPT TIMER0
        JMP   TIME0

START:  MOV TMOD,#21H                   ;SET TIMER1 MODE2(AUTO RE-LOAD)
                                        ;SET TIMER0 MODE1(16 BIT),300=A0)
        MOV   TH0,#0FEH
        MOV   TL0,#033H         ;0.5 msec Timer #0FE33H

        MOV   TIME,#VLEVEL
        SETB  TR0               ;START TIMER0
        CLR   P3.3
        CLR   P3.4
        SETB  EA
        SETB  ET0
        MOV   P1,#0FFH
        MOV   RSPEED,#00011111B

;*************************************************************************
;               START  MIAN  PROGRAM
;*************************************************************************
MAIN:   MOV   A,P1
        JNB   ACC.5,RUN
        JMP   MAIN

RUN:    MOV   RSPEED,A
        JMP   MAIN
```

```
;*************************************************************************
;       PLUSE WIDTH MODULATION (TIMER 0)
;*************************************************************************
TIME0: CLR   TR0
       MOV   31H,A          ;PUSH A
       MOV   TH0,#0FEH
       MOV   TL0,#033H
       DJNZ  TIME,TZ
       MOV   TIME,#VLEVEL
       CLR   P3.4

TZ:    MOV   A,RSPEED
       ANL   A,#00011111B

R_B:   JB    ACC.4,R_F
       CLR   P3.3
       CJNE  A,TIME,STOP_L      ;stop 14  back 1
       SETB  P3.4
       SJMP  STOP_L

R_F:   SETB  P3.3
       CLR   ACC.4
       CJNE  A,TIME,STOP_L      ;run 4  stop 11
       SETB  P3.4

STOP_L:      MOV   A,31H        ;POP A
       SETB  TR0
       RETI
```

```
;*********************************************************************
;       Dummy Delay time 1m,10ms,1s
;*********************************************************************
DELAY_1ms:   MOV   R6,#0E6H
DELAY_1ms_1:       NOP
             NOP
             DJNZ  R6,DELAY_1ms_1
             RET
DELAY_10ms:  MOV   R7,#010
DELAY_10ms_1:      MOV   R6,#0E6H
DELAY_10ms_2:      NOP
             NOP
             DJNZ  R6,DELAY_10ms_2
             DJNZ  R7,DELAY_10ms_1
             RET
DELAY_1s:    MOV   R5,#100
DELAY_1s_1:  ACALL DELAY_10ms
             DJNZ  R5,DELAY_1s_1
             RET

;*********************************************************************
END
```

2. ไมโครคอนโทรลเลอร์ 89C2051 ตัวซ้าย

```
;*************************************************************************
;               Robot    Development
;*************************************************************************
;Filename   projrunl
;Description          Robot  Program
;Assemble ASEM
;Software Eng.        Metee Nilparak
;*************************************************************************
;Clock = 11.0592MHz
;1 Cye = 1.085 uSec
;0.1mSec = 92.166 Cye = 5C = FFA3 Hex
;0.5mSec = 460.83 Cye = 1CC = FE33 Hex
;*************************************************************************
;           อ้างตัวแปรต่างๆ
;*************************************************************************
    VLEVEL      EQU   15D
    LSPEED      EQU   2CH
    TIME   EQU   24H

;*************************************************************************
;               INITIAL  STATUS
;*************************************************************************
    ORG    0000H                ;START MAIN PROGRAM
    JMP    START                ;กระโดดไป main program

    ORG    000BH                ;INTERRUPT TIMER0
           JMP    TIME0
```

```
START:    MOV   TMOD,#21H                ;SET TIMER1 MODE2(AUTO RE-
LOAD)
                                         ;SET TIMER0 MODE1(16 BIT),300=A0)
    MOV   TH0,#0FEH
    MOV   TL0,#033H                      ;0.5 msec Timer #0FE33H

          MOV   TIME,#VLEVEL
    SETB  TR0                            ;START TIMER0
    CLR   P3.3
    CLR   P3.4
    SETB  EA
    SETB  ET0
    MOV   P1,#0FFH
    MOV   LSPEED,#00111111B

;*************************************************************************
;         START  MIAN  PROGRAM
;*************************************************************************
MAIN:     MOV   A,P1
    JB    ACC.5,RUN
    JMP   MAIN

RUN:      MOV   LSPEED,A
    JMP   MAIN

;*************************************************************************
;   PLUSE WIDTH MODULATION (TIMER 0)
;*************************************************************************
TIME0:    CLR   TR0
```

```
        MOV   31H,A           ;PUSH A
        MOV   TH0,#0FEH
        MOV   TL0,#033H
        DJNZ  TIME,TZ
        MOV   TIME,#VLEVEL
        CLR   P3.4

TZ:     MOV   A,LSPEED
        ANL   A,#00011111B

R_B:    JB    ACC.4,R_F
        CLR   P3.3
        CJNE  A,TIME,STOP_L     ;stop 14  back 1
        SETB  P3.4
        SJMP  STOP_L

R_F:    SETB  P3.3
        CLR   ACC.4
        CJNE  A,TIME,STOP_L     ;run 4  stop 11
        SETB  P3.4

STOP_L: MOV   A,31H           ;POP A
        SETB  TR0
        RETI

;*********************************************************************
;   Dummy Delay time 1m,10ms,1s
;*********************************************************************
DELAY_1ms:      MOV   R6,#0E6H
```

```
DELAY_1ms_1:    NOP
                NOP
                DJNZ  R6,DELAY_1ms_1
                RET
DELAY_10ms:     MOV   R7,#010
DELAY_10ms_1:   MOV   R6,#0E6H
DELAY_10ms_2:   NOP
                NOP
                DJNZ  R6,DELAY_10ms_2
                DJNZ  R7,DELAY_10ms_1
                RET
DELAY_1s: MOV   R5,#100
DELAY_1s_1:     ACALL DELAY_10ms
                DJNZ  R5,DELAY_1s_1
                RET

;*****************************************************************
END
```

3. ไมโครคอนโทรลเลอร์ 89C51

```
;*****************************************************************************
;               Robot    Development
;*****************************************************************************
;Filename   projkey
;Description        Robot  Program
;Assemble ASEM
;Software Eng.      Metee Nilparak
;*****************************************************************************
;Clock = 11.0592MHz
;1 Cye = 1.085 uSec
;0.1mSec = 92.166 Cye = 5C = FFA3 Hex
;0.5mSec = 460.83 Cye = 1CC = FE33 Hex
;*****************************************************************************
;               อ้างตัวแปรต่างๆ
;*****************************************************************************
        KPAD_ROW0   BIT     P2.3
        KPAD_ROW1   BIT     P2.4
        KPAD_ROW2   BIT     P2.5
        KPAD_ROW3   BIT     P2.6
        KPAD_COL2   BIT     P2.2
        KPAD_COL1   BIT     P2.1
        KPAD_COL0   BIT     P2.0
        LCD_EN              BIT     P1.1
        LCD_RS              BIT     P1.0
        FLAG        EQU     02FH
        KEYPRESSED  BIT     FLAG.0
        KPAD_DATA   EQU     032H
        LCD_ADDR    EQU     030H
```

```
LCD_DATA     EQU   031H

;*************************************************************************
;            INITIAL  STATUS
;*************************************************************************
             ORG   0000H
             MOV   P0,#00000000B
             MOV   P1,#00000000B
             MOV   P2,#11111111B
             MOV   P3,#11111111B

;*************************************************************************
;            START  MIAN  PROGRAM
;*************************************************************************
MAIN:        MOV   FLAG,#0
             MOV   R4,#00H
             CALL  INIT_LCD

             MOV   LCD_ADDR,#000H
             CALL  SET_ADDR_LCD
             MOV   DPTR,#TITLE_1
             CALL  WRLINE_LCD

             MOV   LCD_ADDR,#040H
             CALL  SET_ADDR_LCD
             MOV   DPTR,#TITLE_2
             CALL  WRLINE_LCD

             CALL  DELAY_1s
```

```
        CALL   DELAY_1s

        MOV    LCD_ADDR,#000H
        CALL   SET_ADDR_LCD
        MOV    DPTR,#TITLE_5
        CALL   WRLINE_LCD

        MOV    LCD_ADDR,#040H
        CALL   SET_ADDR_LCD
        MOV    DPTR,#TITLE_6
        CALL   WRLINE_LCD

LOOP:   ACALL GET_KPAD
        MOV    A,KPAD_DATA
        JZ     NEXT
        JB     KEYPRESSED,SHOW
        SETB   KEYPRESSED
        AJMP   SHOW

NEXT:   CLR    KEYPRESSED
SHOW:   ACALL SEND_B
        CALL   REC
        AJMP   LOOP

;*************************************************************************
;   RECIEVE STATUS OF SENSORS
;*************************************************************************
REC:            ;CJNE  R4,#0FFH,EXX
        MOV    LCD_ADDR,#040H
```

```
        CALL   SET_ADDR_LCD
        MOV    A,P3

SSEN0:  CJNE   A,#00000000B,SSEN1
        MOV    DPTR,#SEN_0
        JMP    SENS

EXX:    JMP    EXX1

SSEN1:  CJNE   A,#00000001B,SSEN2
        MOV    DPTR,#SEN_1
        JMP    SENS

SSEN2:  CJNE   A,#00000010B,SSEN3
        MOV    DPTR,#SEN_2
        JMP    SENS

SSEN3:  CJNE   A,#0000011B,SSEN4
        MOV    DPTR,#SEN_3
        JMP    SENS

SSEN4:  CJNE   A,#00000100B,SSEN5
        MOV    DPTR,#SEN_4
        JMP    SENS

SSEN5:  CJNE   A,#00000101B,SSEN6
        MOV    DPTR,#SEN_5
        JMP    SENS
```

```
SSEN6:      CJNE  A,#00000110B,SSEN7
            MOV   DPTR,#SEN_6
            JMP   SENS

SSEN7:      CJNE  A,#00000111B,SSEN8
            MOV   DPTR,#SEN_7
            JMP   SENS

SSEN8:      CJNE  A,#00001000B,SSEN9
            MOV   DPTR,#SEN_8
            JMP   SENS

SSEN9:      CJNE  A,#00001001B,SSEN10
            MOV   DPTR,#SEN_9
            JMP   SENS

SSEN10:     CJNE  A,#00001010B,SSEN11
            MOV   DPTR,#SEN_10
            JMP   SENS

SSEN11:     CJNE  A,#00001011B,SSEN12
            MOV   DPTR,#SEN_11
            JMP   SENS

SSEN12:     CJNE  A,#00001100B,SSEN13
            MOV   DPTR,#SEN_12
            JMP   SENS

SSEN13:     CJNE  A,#00001101B,SSEN14
```

```
        MOV   DPTR,#SEN_13
        JMP   SENS

SSEN14:     CJNE  A,#00001110B,SSEN15
        MOV   DPTR,#SEN_14
        JMP   SENS

SSEN15:     CJNE  A,#00001111B,SSEN16
        MOV   DPTR,#SEN_15
        JMP   SENS

SSEN16:     CJNE  A,#00010000B,SSEN17
        MOV   DPTR,#SEN_16
        JMP   SENS

SSEN17:     CJNE  A,#00010001B,SSEN18
        MOV   DPTR,#SEN_17
        JMP   SENS

SSEN18:     CJNE  A,#00010010B,SSEN19
        MOV   DPTR,#SEN_18
        JMP   SENS

SSEN19:     CJNE  A,#00010011B,SSEN20
        MOV   DPTR,#SEN_19
        JMP   SENS

SSEN20:     CJNE  A,#00010100B,SSEN21
        MOV   DPTR,#SEN_20
```

```
        JMP   SENS

SSEN21:     CJNE  A,#00010101B,SSEN22
        MOV   DPTR,#SEN_21
        JMP   SENS

SSEN22:     CJNE  A,#00010110B,SSEN23
        MOV   DPTR,#SEN_22
        JMP   SENS

SSEN23:     CJNE  A,#00010111B,SSEN24
        MOV   DPTR,#SEN_23
        JMP   SENS

SSEN24:     CJNE  A,#00011000B,SSEN25
        MOV   DPTR,#SEN_24
        JMP   SENS

SSEN25:     CJNE  A,#00011001B,SSEN26
        MOV   DPTR,#SEN_25
        JMP   SENS

SSEN26:     CJNE  A,#00011010B,SSEN27
        MOV   DPTR,#SEN_26
        JMP   SENS

SSEN27:     CJNE  A,#00011011B,SSEN28
        MOV   DPTR,#SEN_27
        JMP   SENS
```

```
SSEN28:         CJNE  A,#00011100B,SSEN29
                MOV   DPTR,#SEN_28
                JMP   SENS

SSEN29:         CJNE  A,#00011101B,SSEN30
                MOV   DPTR,#SEN_29
                JMP   SENS

SSEN30:         CJNE  A,#00011110B,SSEN31
                MOV   DPTR,#SEN_30
                JMP   SENS

SSEN31:         CJNE  A,#00011111B,EXX1
                MOV   DPTR,#SEN_31
                JMP   SENS

SENS:           CALL  WRLINE_LCD
EXX1:           MOV   A,#00H

                RET

;*************************************************************************
;   Keypad Scan key Subroutine
;*************************************************************************
GET_KPAD:       MOV   P2,#0FFH
                MOV   KPAD_DATA,#0

CHK_COL0:       CLR   KPAD_COL0
```

```
                MOV   A,P2
                ANL   A,#01111000B
                CJNE  A,#01111000B,COL0_DETECT
                AJMP  CHK_COL1

COL0_DETECT:    MOV   KPAD_DATA,#01
                AJMP  GET_ROW

CHK_COL1:       SETB  KPAD_COL0
                CLR   KPAD_COL1
                MOV   A,P2
                ANL   A,#01111000B
                CJNE  A,#01111000B,COL1_DETECT
                AJMP  CHK_COL2

COL1_DETECT:    MOV   KPAD_DATA,#02
                AJMP  GET_ROW

CHK_COL2:       SETB  KPAD_COL1
                CLR   KPAD_COL2
                MOV   A,P2
                ANL   A,#01111000B
                CJNE  A,#01111000B,COL2_DETECT
                RET

COL2_DETECT:    MOV   KPAD_DATA,#03

GET_ROW:CLR     KPAD_COL0
                CLR   KPAD_COL1
```

```
        CLR     KPAD_COL2
        JB      KPAD_ROW0,CHK_ROW1
        RET

CHK_ROW1:       JB      KPAD_ROW1,CHK_ROW2
        MOV     A,KPAD_DATA
        ADD     A,#3
        MOV     KPAD_DATA,A
        RET

CHK_ROW2:       JB      KPAD_ROW2,CHK_ROW3
        MOV     A,KPAD_DATA
        ADD     A,#6
        MOV     KPAD_DATA,A
        RET

CHK_ROW3:       MOV     A,KPAD_DATA
        ADD     A,#9
        MOV     KPAD_DATA,A
        RET

;*************************************************************************
;    Show DSP Subroutine
;*************************************************************************
SEND_B:         CJNE    A,#10,SEND_B1
        RET

SEND_B1: CJNE   A,#7,SEND_B2
        RET
```

```
SEND_B2: CJNE   A,#4,SEND_B3
         SETB   P1.2
         SETB   P1.3

         MOV    LCD_ADDR,#000H
         CALL   SET_ADDR_LCD
         MOV    DPTR,#TITLE_3
         CALL   WRLINE_LCD

         MOV    LCD_ADDR,#040H
         CALL   SET_ADDR_LCD
         MOV    DPTR,#TITLE_4
         CALL   WRLINE_LCD

         RET

SEND_B3: CJNE   A,#1,SEND_B4
         CLR    P1.2
         CLR    P1.3

         MOV    LCD_ADDR,#000H
         CALL   SET_ADDR_LCD
         MOV    DPTR,#TITLE_5
         CALL   WRLINE_LCD

         MOV    LCD_ADDR,#040H
         CALL   SET_ADDR_LCD
         MOV    DPTR,#TITLE_6
```

```
        CALL   WRLINE_LCD

        MOV    R4,#00H
        RET

SEND_B4: CJNE  A,#2,SEND_B5
        CLR    P1.7
        SETB   P1.6
        SETB   P1.5

        MOV    LCD_ADDR,#000H
        CALL   SET_ADDR_LCD
        MOV    DPTR,#TITLE_8
        CALL   WRLINE_LCD

        MOV    R4,#0FFH
        RET

SEND_B5: CJNE  A,#5,SEND_B6
        CLR    P1.7
        SETB   P1.6
        CLR    P1.5

        MOV    LCD_ADDR,#040H
        CALL   SET_ADDR_LCD
        MOV    DPTR,#TITLE_7
        CALL   WRLINE_LCD

        RET
```

```
SEND_B6: CJNE   A,#8,SEND_B7
         CLR    P1.7
         CLR    P1.6
         SETB   P1.5
         RET

SEND_B7: CJNE   A,#11,SEND_B8
         RET

SEND_B8: CJNE   A,#3,SEND_B9
         SETB   P1.7
         SETB   P1.6
         SETB   P1.5

         MOV    LCD_ADDR,#000H
         CALL   SET_ADDR_LCD
         MOV    DPTR,#TITLE_9
         CALL   WRLINE_LCD

         RET

SEND_B9: CJNE   A,#6,SEND_B10
         SETB   P1.7
         SETB   P1.6
         CLR    P1.5

         MOV    LCD_ADDR,#000H
         CALL   SET_ADDR_LCD
```

```
        MOV   DPTR,#TITLE_12
        CALL  WRLINE_LCD

        RET

SEND_B10:   CJNE  A,#9,SEND_B11
        SETB  P1.7
        CLR   P1.6
        SETB  P1.5

        MOV   LCD_ADDR,#000H
        CALL  SET_ADDR_LCD
        MOV   DPTR,#TITLE_13
        CALL  WRLINE_LCD

        MOV   LCD_ADDR,#040H
        CALL  SET_ADDR_LCD
        MOV   DPTR,#TITLE_10
        CALL  WRLINE_LCD

        RET

SEND_B11:   CJNE  A,#12,SEND_B12
        SETB  P1.7
        CLR   P1.6
        CLR   P1.5

        MOV   LCD_ADDR,#000H
        CALL  SET_ADDR_LCD
```

```
        MOV   DPTR,#TITLE_5
        CALL  WRLINE_LCD

        MOV   LCD_ADDR,#040H
        CALL  SET_ADDR_LCD
        MOV   DPTR,#TITLE_6
        CALL  WRLINE_LCD

        MOV   R4,#00H
        RET

SEND_B12:       CJNE  A,#00H,SEND_B13
        CLR   P1.7
        CLR   P1.6
        CLR   P1.5

SEND_B13:       RET

;*************************************************************************
;   LCD Initialize
;*************************************************************************
INIT_LCD: ACALL DELAY_100ms
        CLR   LCD_RS

        MOV   P0,#00111000B
        ACALL LCD_CLK
        ACALL DELAY_10ms

        MOV   P0,#00111000B
```

```
        ACALL LCD_CLK

        ACALL LCD_OFF

        ACALL LCD_CLR

        MOV   P0,#00000110B
        ACALL LCD_CLK

        ACALL LCD_HOME
        RET

;*************************************************************************
;   LCD Clear Display
;*************************************************************************
LCD_CLR: CLR   LCD_RS
        MOV   P0,#00000001B
        ACALL LCD_CLK
        RET

;*************************************************************************
;   LCD Return Home
;*************************************************************************
LCD_HOME:       CLR   LCD_RS
        MOV   P0,#00000010B
        ACALL LCD_CLK
        RET
```

```
;*************************************************************************
;    LCD Display Off
;*************************************************************************
LCD_OFF: CLR    LCD_RS
         MOV    P0,#00001000B
         ACALL LCD_CLK
         RET

;*************************************************************************
;    LCD Clk
;*************************************************************************
LCD_CLK: SETB   LCD_EN
         ACALL LCD_DELAY
         CLR    LCD_EN
         ACALL LCD_DELAY
         RET

;*************************************************************************
;    LCD Display On
;*************************************************************************
LCD_ON:          CLR    LCD_RS
         MOV    P0,#00001100B
         ACALL LCD_CLK
         RET

;*************************************************************************
;    LCD Cursor On
;*************************************************************************
LCD_BLINK:       CLR    LCD_RS
```

```
            MOV   P0,#00001111B
            ACALL LCD_CLK
            RET

;*************************************************************************
;   LCD Left Shift Display
;*************************************************************************
LCD_LSHF:       CLR   LCD_RS
            MOV   P0,#00011000B
            ACALL LCD_CLK
            RET

;*************************************************************************
;   LCD Right Shift Display
;*************************************************************************
LCD_RSHF:       CLR   LCD_RS
            MOV   P0,#00011100B
            ACALL LCD_CLK
            RET

;*************************************************************************
;   Set LCD Address
;   I/P:            LCD_ADDR
;*************************************************************************
SET_ADDR_LCD: CLR   LCD_RS
            MOV   A,LCD_ADDR
            SETB  ACC.7
            MOV   P0,A
            ACALL LCD_CLK
```

```
        RET

;*************************************************************************
;   Write Character to show LCD
;   I/P:            LCD_DATA
;*************************************************************************
WRCHAR_LCD:  SETB  LCD_RS
        MOV   P0,LCD_DATA
        ACALL LCD_CLK
        ACALL LCD_ON
        RET

;*************************************************************************
;   Write Line of 16 Character From ROM
;   I/P         DPTR : Locate ROM Address
;*************************************************************************
WRLINE_LCD:     MOV   R0,#0
WRLINE_LCD_1:  SETB  LCD_RS
        CLR   A
        MOVC A,@A+DPTR
        MOV   P0,A
        ACALL LCD_CLK
        INC   DPTR
        INC   R0
        CJNE  R0,#16,WRLINE_LCD_1
        ACALL LCD_ON
        RET
```

```
;*************************************************************************
;    Dummy Delay time LCD_DELAY, 10m, 100m, 1s
;*************************************************************************
LCD_DELAY:      MOV   R7,#002
LCD_DELAY_1:    MOV   R6,#0E6H
LCD_DELAY_2:    NOP
                NOP
                DJNZ  R6,LCD_DELAY_2
                DJNZ  R7,LCD_DELAY_1
                RET
DELAY_10ms:     MOV   R7,#010
DELAY_10ms_1:   MOV   R6,#0E6H
DELAY_10ms_2:   NOP
                NOP
                DJNZ  R6,DELAY_10ms_2
                DJNZ  R7,DELAY_10ms_1
                RET
DELAY_100ms:    MOV   R7,#100
DELAY_100ms_1:  MOV   R6,#0E6H
DELAY_100ms_2:  NOP
                NOP
                DJNZ  R6,DELAY_100ms_2
                DJNZ  R7,DELAY_100ms_1
                RET
DELAY_1s:MOV    R5,#100
DELAY_1s_1:     ACALL DELAY_10ms
                DJNZ  R5,DELAY_1s_1
                RET
```

```
;*************************************************************************
;    Define  Constant < Store in Flash EEPROM Program Memory >
;*************************************************************************
;                    0123456789ABCDEF
TITLE_1:            DB      '   ROBOT    '
TITLE_2:            DB      ' DEVELOPMENT  '
TITLE_3:            DB      ' PROGRAM MODE '
TITLE_4:            DB      'PLEASE  DOWNLOAD'
TITLE_5:            DB      ' RUNNING MODE '
TITLE_6:            DB      'PLEASE PRESS (1)'
TITLE_7:            DB      'PLEASE PRESS (2)'
TITLE_8:            DB      'PRESS (S) START!'
TITLE_9:            DB      ' PRESS  (S,H,R) '
TITLE_10:  DB       ' ARM HOME POS.. '
TITLE_12:  DB       ' PRESS  (C,H,R) '
TITLE_13:  DB       ' PRESS (S,R) '
SEN_0:              DB      ' 1*2*3*4*5*  '
SEN_1:              DB      ' 1*2*3*4*5   '
SEN_2:              DB      ' 1*2*3*4 5*  '
SEN_3:              DB      ' 1*2*3*4 5   '
SEN_4:              DB      ' 1*2*3 4*5*  '
SEN_5:              DB      ' 1*2*3 4*5   '
SEN_6:              DB      ' 1*2*3 4 5*  '
SEN_7:              DB      ' 1*2*3 4 5   '
SEN_8:              DB      ' 1*2 3*4*5*  '
SEN_9:              DB      ' 1*2 3*4*5   '
SEN_10:             DB      ' 1*2 3*4 5*  '
SEN_11:             DB      ' 1*2 3*4 5   '
SEN_12:             DB      ' 1*2 3 4*5*  '
```

```
SEN_13:         DB    ' 1*2 3 4*5  '
SEN_14:         DB    ' 1*2 3 4 5* '
SEN_15:         DB    ' 1*2 3 4 5  '
SEN_16:         DB    ' 1 2*3*4*5* '
SEN_17:         DB    ' 1 2*3*4*5  '
SEN_18:         DB    ' 1 2*3*4 5* '
SEN_19:         DB    ' 1 2*3*4 5  '
SEN_20:         DB    ' 1 2*3 4*5* '
SEN_21:         DB    ' 1 2*3 4*5  '
SEN_22:         DB    ' 1 2*3 4 5* '
SEN_23:         DB    ' 1 2*3 4 5  '
SEN_24:         DB    ' 1 2 3*4*5* '
SEN_25:         DB    ' 1 2 3*4*5  '
SEN_26:         DB    ' 1 2 3*4 5* '
SEN_27:         DB    ' 1 2 3*4 5  '
SEN_28:         DB    ' 1 2 3 4*5* '
SEN_29:         DB    ' 1 2 3 4*5  '
SEN_30:         DB    ' 1 2 3 4 5* '
SEN_31:         DB    ' 1 2 3 4 5  '

;*************************************************************************
END
```

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ในครั้งนี้ ต้องขอกราบพระคุณอย่างสูงแด่ อาจารย์สุมิตร พนาอุดมทรัพย์ ที่ให้โอกาสพวกกระผมได้ทำหัวข้อโปรเจ็กต์นี้ รวมทั้งยังให้คำปรึกษาและสนับสนุนในด้านต่างๆตลอดมาจนสำเร็จลุล่วงไปด้วยดี

ขอขอบคุณ นายจตุรวิทย์ จันไพบูลย์ อดีตพี่ชุมนุมโรบอทที่ให้คำปรึกษาเพิ่มเติมให้แก่โปรเจ็กต์นี้ และชุมนุมโรบอทที่เอื้อเฟื้อสถานที่ในการสร้างหุ่นยนต์

ขอขอบคุณ นายมารุต ยิ้มแย้ม นักศึกษาภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้คำปรึกษาเพิ่มเติมในด้านโปรแกรมคอมพิวเตอร์

สุดท้ายนี้ขอขอบพระคุณ ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้โอกาสได้ทำโครงงานวิจัย เพื่อเสริมสร้างทักษะความรู้ ให้กับผู้จัดทำการวิจัยและให้การสนับสนุนจนทำให้โครงการวิจัยและปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

นายเมธี นิลภารักษ์

นายวันลาภ พฤทธ์กิตติวงศ์

เอกสารอ้างอิง

1. วรพจน์ กรแก้ววัฒนกุล และ ชัยวัฒน์ ลิ้มพรจิตรวิไล, "เรียนรู้ปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช ฉบับ AT89C5x ของ Atmel", อินโนเวทีฟ เอ็กเพอริเมนต์, 399 หน้า, 2542
2. ทีมงาน อินโนเวทีฟ เอ็กเพอริเมนต์, "คู่มือสำหรับอ้างอิงและใช้งานบอร์ดทดลอง NX-51 V2.0", อินโนเวทีฟ เอ็กเพอริเมนต์, หน้า 1-37, 2544
3. วรพจน์ กรแก้ววัฒนกุล และ ชัยวัฒน์ ลิ้มพรจิตรวิไล, "ชุดเรียนรู้การสร้างหุ่นยนต์อัตโนมัติขนาดเล็ก", อินโนเวทีฟ เอ็กเพอริเมนต์, หน้า 31-50, 2543
4. วรพจน์ กรแก้ววัฒนกุล และ ชัยวัฒน์ ลิ้มพรจิตรวิไล และ อรรถพล บุญยะโภคา, "เรียนรู้และสร้างหุ่นยนต์อย่างง่าย", อินโนเวทีฟ เอ็กเพอริเมนต์, หน้า 7-13, 2544
5. ทีมงาน อินโนเวทีฟ เอ็กเพอริเมนต์, "คู่มือการใช้งาน EX-8951 board", อินโนเวทีฟ เอ็กเพอริเมนต์, หน้า 8-12, 2544
6. บุญชัย กิ่งรุ้งเพชร, "คู่มือ Protel 99", ซีเอ็ดยูเคชั่นจำกัด, หน้า 9-77, 2543
7. ธาริน สิทธิธรรมชารี, "คู่มือการเขียนโปรแกรม Microsoft Visual Basic Version 6.0 ฉบับเพื่อการใช้งานจริง", ซีเอ็ดยูเคชั่นจำกัด, หน้า 8-53, 2544