หุ่นยนต์ควบคุมด้วยคอมพิวเตอร์

COMPUTER CONTROLLED ROBOT

โดย

นาย จตุรวิทย์ จันไพบูลย์

นาย จิรวัฒก์ ม่วงดิษฐ์

อาจารย์ที่ปรึกษา

รศ.ดร. จงกล งามวิวิทย์

อาจารย์ ถาวร เบญจนราสุทธิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ ปีการศึกษา 2543

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง หุ่นยนต์ควบคุมด้วยคอมพิวเตอร์

COMPUTER CONTROLLED ROBOT

ผู้จัดทำ 1. นายจตุรวิทย์ จันไพบูลย์

2. นายจิรวัฒก์ ม่วงดิษฐ์

_____ อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. จงกล งามวิวิทย์)

_____ อาจารย์ที่ปรึกษา

(อาจารย์ ถาวร เบญจนราสุทธิ์)

หุ่นยนต์ควบคุมด้วยคอมพิวเตอร์

นาย จตุรวิทย์ จันไพบูลย์
นาย จิรวัฒก์ ม่วงดิษฐ์
รศ.ดร. จงกล งามวิวิทย์ อาจารย์ที่ปรึกษา
อ. ถาวร เบญจนราสุทธิ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการออกแบบและสร้างระบบควบคุมหุ่นยนต์ที่สามารถหลบสิ่งกีดขวางและเคลื่อนที่ไปยังตำแหน่งที่ต้องการ โดยใช้ระยะทางที่สั้นที่สุด กล้องวิดีโอจะคอยจับภาพหุ่นยนต์และสิ่งกีดขวาง และส่งให้เครื่องคอมพิวเตอร์ส่วนบุคคลแสดงผล แล้วคอมพิวเตอร์จะทำการประมวลผลภาพและส่งสัญญาณควบคุมไร้สายให้กับไมโครคอนโทรลเลอร์ 89C51 ควบคุมการเคลื่อนที่ของหุ่นยนต์

COMPUTER CONTROLLED ROBOT

Jaturawit Janpaiboon

Jirawat Muangdit

Assoc. Prof. Dr. Jongol Ngramwiwit Advisor

Taworn Benjanarasuth Advisor

ABSTRACT

This project presents the design and construction of robot control system in order to avoid the obstruction and to move the robot to the target position within the shortest distance. A video camera captures the pictures of robot and obstruction and sends to the personal computer for displaying the captured picture. Then personal computer will process the captured pictures and sends the wireless control signal to the microcontroller 89C51 to the movement of the robot.

สารบัญ

สารบัญภาพ

สารบัญตาราง

บทที่ 1
บทนำ

ปัจจุบันวิวัฒนาการทางด้านเทคโนโลยีได้มีการนำหุ่นยนต์มาใช้ในลักษณะงานต่าง ๆ กันอย่างแพร่หลายซึ่งสามารถแบ่ง หุ่นยนต์ได้เป็น 2 ประเภทใหญ่ ๆ คือ

1. Fixed Robot
2. Mobile Robot

Fixed Robot คือ หุ่นยนต์ที่ฐานถูกยึดอยู่กับที่ ไม่สามารถเคลื่อนที่ได้อิสระ เช่น แขนกล ซึ่งทำหน้าที่ในการประกอบชิ้นส่วนอุปกรณ์ต่าง ๆ ในโรงงานอุตสาหกรรมเป็นต้น

Mobile Robot คือ หุ่นยนต์ที่ถูกตรึงกับฐานที่สามารถเคลื่อนที่ได้โดยอิสระ โดยฐานอาจมีล้อหรือระบบการเคลื่อนที่แบบอื่นก็ได้ โดยงานที่สำคัญอย่างหนึ่งของหุ่นยนต์ชนิดนี้ก็คืองานสำรวจ (Observation)โดยการที่นำ Mobile Robot มาใช้งานสำรวจหรือปฏิบัติงาน เช่น งานที่เสี่ยงอันตรายต่อชีวิตมนุษย์ ตัวอย่าง เช่น งานกู้วัตถุระเบิด หรือ งานสำรวจที่มนุษย์ไม่สามารถเข้าไปสำรวจได้ด้วยตัวเอง โดยอาจมีปัญหาทางด้านมลภาวะด้านสิ่งแวดล้อมการทำงานต่าง ๆ อันได้แก่ อุณหภูมิ รังสี สารเคมี ฝุ่นละออง เป็นต้น ยกตัวอย่าง เช่น ขณะที่เกิดไฟไหม้ในแหล่งที่มีการเก็บสารเคมีอันตรายแทนที่จะส่งมนุษย์เข้าไปตรวจสอบ สำรวจพื้นที่ซึ่งเป็นการเสี่ยงต่อการสูญเสียชีวิตมากเราสามารถที่จะส่ง Mobile Robot เข้าไปสำรวจแทนได้โดยข้อมูลที่ได้จากการสำรวจจะขึ้นอยู่กับชนิดของเซนเซอร์ (Sensor) หรืออุปกรณ์ที่ใช้ในการสำรวจ

1.1 วัตถุประสงค์

1. เพื่อให้หุ่นยนต์สามารถเคลื่อนที่หลบหลีกสิ่งกีดขวางได้
2. เพื่อศึกษาการประยุกต์ใช้งานอินฟราเรด
3. เพื่อศึกษาค้นคว้าทดลองเพื่อพัฒนาการ ประยุกต์ใช้งานไมโครคอนโทรลเลอร์เพื่อควบคุมการทำงานของหุ่นยนต์
4. เพื่อศึกษาการรับส่งข้อมูลผ่านพอร์ตอนุกรม

1.2 ขอบข่ายของโครงงาน

โครงงานนี้ถูกแบ่งออกได้เป็น 2 ส่วน คือ

1. ส่วนของฮาร์ดแวร์ (Hardware)

   ประกอบด้วย

   - ตัวรถ
   - ตัวกล้องวิดีโอ (Video Camera)
   - วงจรอิเล็กทรอนิกส์
   - เครื่องคอมพิวเตอร์

2. ส่วนของซอฟแวร์ (Software)

   ประกอบด้วย

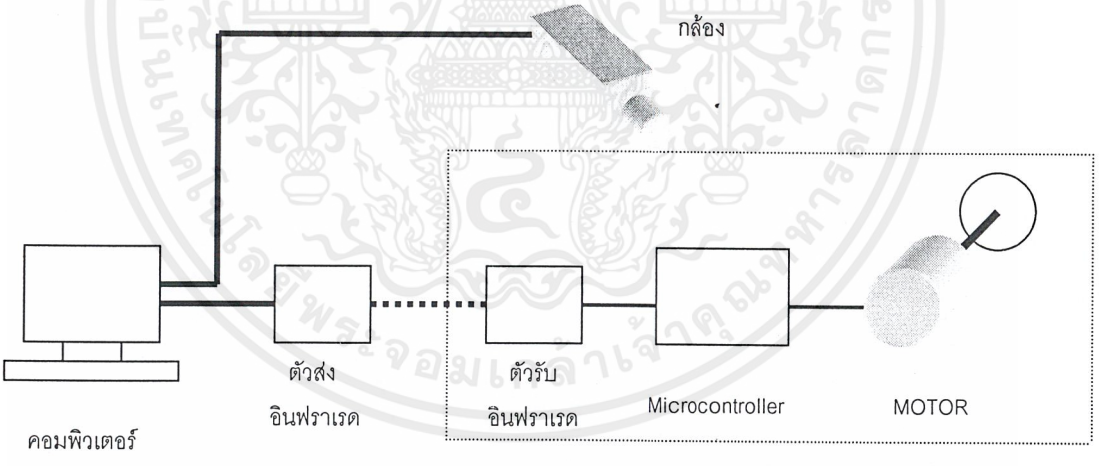
   - โปรแกรมการควบคุมมอเตอร์โดยใช้โปรแกรมภาษาแอสเซมบลี (Assembly)
   - โปรแกรมการรับส่งค่าและประมวลผลด้วยโปรแกรม Visual Basic 6

บทที่ 2
ทฤษฎีและหลักการ

2.1 ขั้นตอนการทำงาน

เริ่มต้นกล้องวิดีโอจะจับภาพมาแล้วส่งให้คอมพิวเตอร์ทำการจำลองภาพให้เป็นรูปภาพแบบง่ายๆ เมื่อทำการระบุตำแหน่งที่ต้องการให้หุ่นยนต์ไป เครื่องคอมพิวเตอร์จะทำการคำนวณหาเส้นทางที่หุ่นยนต์สามารถเดินทางไปได้และคำนวณหาเส้นทางที่สั้นที่สุดด้วย หลังจากนั้นคอมพิวเตอร์จะป้อนคำสั่งที่ใช้ในการควบคุมหุ่นยนต์โดยส่งข้อมูลออกทางพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ผ่านตัวรับส่งอินฟราเรด เมื่อไมโครคอนโทรลเลอร์ในหุ่นยนต์ได้รับคำสั่งแล้วก็จะทำการเดินตามคำสั่งที่ได้รับมา จากนั้นกล้องต้องคอยจับภาพเพื่อคอยดูว่าหุ่นยนต์เดินไปยังตำแหน่งที่ต้องการหรือไม่ ถ้ายังไปไม่ถึงก็จะยังคงส่งคำสั่งไปให้หุ่นยนต์จนกว่าหุ่นยนต์จะอยู่ที่ตำแหน่งที่ต้องการ

2.2 บล็อกไดอะแกรมของระบบ



รูปที่ 2.1 แผนภาพบล็อกไดอะแกรมแสดงการทำงานของหุ่นยนต์ควบคุมด้วยคอมพิวเตอร์

ระบบสามารถถูกแบ่งออกเป็น 4 ส่วนดังนี้

ส่วนคอมพิวเตอร์ จะเป็นตัวรับคำสั่งจากผู้ใช้งานและรับภาพที่ส่งมาจากกล้อง เพื่อนำมาคำนวณ แล้วคอมพิวเตอร์จะทำการแปลงข้อมูลที่ใช้ในการควบคุมให้อยู่ในรูปแบบที่จะใช้ส่งผ่านพอร์ตอนุกรมเพื่อที่จะส่งให้ตัวส่งอินฟราเรด

ส่วนรับภาพ ใช้กล้องดิจิตอลเป็นตัวจับภาพจากทางด้านบนโดยครอบคลุมพื้นที่สำรวจ แล้วส่งสัญญาณภาพมาให้คอมพิวเตอร์ผ่านทางพอร์ต USB ตัวกล้องจะถูกยึดติดกับเสาเพื่อให้กล้องสามารถมองจากมุมบนได้

ส่วนการขับเคลื่อนเซอร์โวมอเตอร์ เป็นส่วนที่ทำให้หุ่นยนต์สามารถเคลื่อนที่ได้ โดยจะใช้ เซอร์โวมอเตอร์ ทั้งหมด 2 ตัว ซึ่งสามารถควบคุมความเร็วและทิศทางของมอเตอร์ได้จากพัลส์ (pulse) ที่ส่งไปให้มอเตอร์

ส่วนไมโครคอนโทรลเลอร์ จะทำหน้าที่ในการควบคุมการทำงานของหุ่นยนต์ทั้งหมด โดยจะรับคำสั่งการควบคุมจากคอมพิวเตอร์ที่ส่งมาทางพอร์ตอนุกรม ผ่านตัวรับส่งอินฟราเรด และทำการควบคุมการหมุนของมอเตอร์ให้เป็นไปตามคำสั่งโดยการส่งพัลส์ไปให้

ส่วนการติดต่อสื่อสาร ทำหน้าที่เป็นตัวเชื่อมต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์ ส่วนนี้ประกอบด้วย การเชื่อมต่อผ่านพอร์ตอนุกรมและส่งข้อมูลต่อด้วยอินฟราเรด

2.3 ไมโครคอนโทรลเลอร์ตระกูล 8051

2.3.1 คุณสมบัติทั่วไปของไมโครคอนโทรลเลอร์ MCS-51

คุณสมบัติทั่วไปที่สำคัญของไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีดังนี้

- เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต
- มีวงจรออสซิลเลเตอร์และวงจรผลิตสัญญาณนาฬิกาภายในไอซี
- มีขาสัญญาณอินพุตเอาต์พุตจำนวน 32 บิต
- สามารถเชื่อมต่อหน่วยความจำข้อมูลภายนอก (external data memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 k
- สามารถเชื่อมต่อหน่วยความจำโปรแกรมภายนอก (external program memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 k
- มีหน่วยความจำโปรแกรมภายในตัว (on-chip program memory) ขนาด 4 k โดยเฉพาะเบอร์ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 8 k สำหรับเบอร์ 8031 และ M8032 จะไม่มีหน่วยความจำในส่วนนี้
- มีหน่วยความจำข้อมูลภายในตัว (on-chip data memory) ขนาด 128 ไบต์ โดยเฉพาะเบอร์ 8032 และ 8052 จะมีหน่วยความจำในส่วนนี้ถึง 256 ไบต์

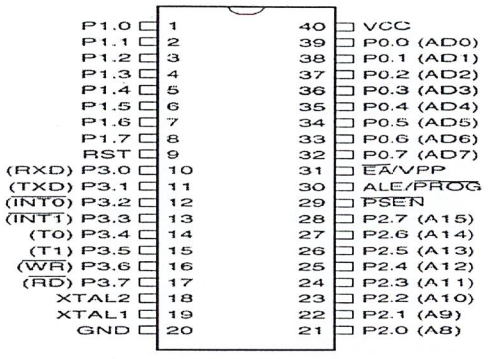
- หน่วยความจำข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย ทำให้การควบคุมหรือการตรวจสอบสถานะบิตทำได้ง่าย ส่งผลให้การเขียนโปรแกรมทำได้ง่ายมากขึ้น
- มีไทเมอร์/เคาน์เตอร์ (timer/counters) ขนาด 16 บิต จำนวน 2 ตัว โดยเฉพาะเบอร์ 8032 และ 8052 จะมีไทเมอร์/เคาน์เตอร์จำนวน 3 ตัว
- การอินเตอร์รัปต์สามารถทำได้จาก 5 แหล่งกำเนิดโดยเฉพาะเบอร์ 8032 และ 8052 จะทำการอินเตอร์รัปต์ได้จาก 6 แหล่งกำเนิด โดยการอินเตอร์รัปต์ยังสามารถจัดระดับความสำคัญได้เป็น 2 ระดับ
- มีพอร์ตสื่อสารอนุกรมภายในตัวเอง ซึ่งทำงานเป็นแบบฟูลดูเพล็กซ์ (full duplex)
- มีคำสั่งในการคำนวณทางคณิตศาสตร์และทางตรรกศาสตร์
- คำสั่งโดยส่วนใหญ่ใช้เวลาการทำงานเพียง 1 ไมโครวินาที เมื่อใช้คริสตอลความถี่ 12 เมกะเฮิรตซ์
- ต้องการแหล่งจ่ายไฟ 5 โวลต์ เพียงชุดเดียว

ตารางที่ 2.1 แสดงคุณสมบัติของไมโครคอนโทรลเลอร์แต่ละเบอร์ในตระกูล MCS-51

ชื่อเบอร์	หน่วยความจำภายใน		จำนวนไทเมอร์/เคาน์เตอร์	จำนวนอินเตอร์รัปต์
	เก็บโปรแกรม	เก็บข้อมูล		
8052AH	8k*8 ROM	256*8 RAM	3*16-Bit	6
8051AH	4k*8 ROM	128*8 RAM	2*16-Bit	5
8051	4k*8 ROM	128*8 RAM	2*16-Bit	5
8032AH	ไม่มี	256*8 RAM	3*16-Bit	6
8031AH	ไม่มี	128*8 RAM	2*16-Bit	5
8031	ไม่มี	128*8 RAM	2*16-Bit	5
8751H	4k*8 EPROM	128*8 RAM	2*16-Bit	5
8751H-12	4k*8 EPROM	128*8 RAM	2*16-Bit	5

2.3.2 โครงสร้างภายนอกของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกัน ดังแสดงในรูปที่ 2.2 สำหรับหน้าที่การใช้งานของแต่ละขามีดังนี้



รูปที่ 2.2 แสดงการจัดตำแหน่งขาต่างๆของไมโครคอนโทรลเลอร์ตระกูล MCS-51

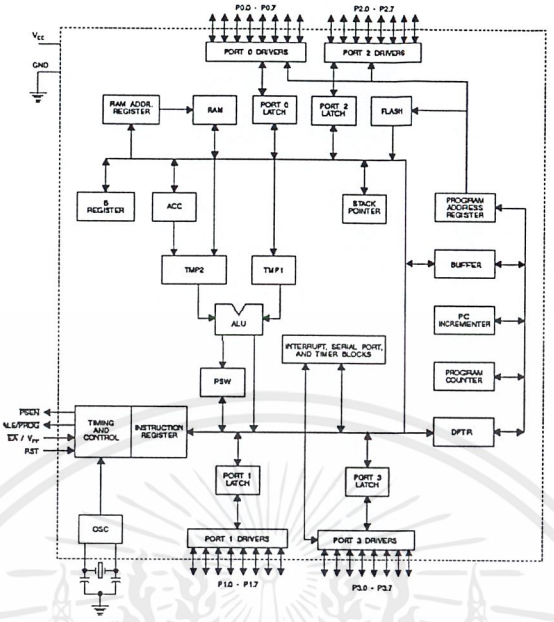
- ขา V_{CC} เป็นขาป้อนแรงดันไฟเลี้ยง +5 โวลต์
- ขา V_{SS} เป็นขากราวด์
- ขาพอร์ต 0 (Port 0) มี 8 ขา ได้แก่ขา $P_{0.0}$-$P_{0.7}$ เป็นขาพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้ขาพอร์ตเหล่านั้นอยู่ในสถานะปล่อยลอย ซึ่งในสถานะนี้เองที่สามารถนำมาใช้เป็นพอร์ตอินพุตอิมพีแดนซ์สูงได้ นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้าที่ในการกำหนดตำแหน่งแอดเดรสไบต์ต่ำ (A_0-A_7) ซึ่งจะใช้งานเป็นแบบมัลติเพล็กซ์กับการรับส่งข้อมูลขนาด 8 บิต (D_0-D_7)
- ขาพอร์ต 1 (Port 1) มี 8 ขา ได้แก่ขา $P_{1.0}$-$P_{1.7}$ เป็นขาพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ตเพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากนี้สำหรับเบอร์ 8032 และ 8052 ขาพอร์ต $P_{1.0}$ และ $P_{1.1}$ จะถูกนำมาใช้งานเป็นขา T2 และ T2EX ตามลำดับด้วย
- ขาพอร์ต 2 (Port 2) มี 8 ขาได้แก่ขา $P_{2.0}$-$P_{2.7}$ เป็นขาพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วย โดยทำหน้าที่ในการกำหนดตำแหน่งแอดเดรสไบต์สูง (A_8-A_{15})
- ขาพอร์ต 3 (Port 3) มี 8 ขาได้แก่ขา $P_{3.0}$-$P_{3.7}$ เป็นขาพอร์ตอินพุตเอาต์พุตแบบ 2 ทิศทางสำหรับใช้งานทั่วไปโดยถ้าใช้งานเป็นอินพุตพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้เป็นพอร์ตอินพุต นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุตเอาต์พุตแล้วมันยังถูกใช้งานในหน้าที่พิเศษต่าง ๆ ดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 แสดงหน้าที่พิเศษของแต่ละขาของพอร์ต P3

ขาพอร์ต	หน้าที่พิเศษ
$P_{3.0}$	RXD (serial input port)
$P_{3.1}$	TXD (serial output port)
$P_{3.2}$	$\overline{\text{INT0}}$ (external interrupt 0)
$P_{3.3}$	$\overline{\text{INT1}}$ (external interrupt 1)
$P_{3.4}$	T0 (Timer 0 external input)
$P_{3.5}$	T1 (Timer 1 external input)
$P_{3.6}$	$\overline{\text{WR}}$ (external data memory write strobe)
$P_{3.7}$	$\overline{\text{RD}}$ (external data memory read strobe)

- ขารีเซต (RST) ใช้สำหรับการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตต้องคงสถานะเป็น 1 อย่างน้อยนาน 2 แมชีนไซเกิล ในขณะที่ออสซิเลเตอร์ยังทำงานอยู่
- ขา ALE/$\overline{\text{PROG}}$ เป็นขาสัญญาณเพื่อทำหน้าที่ควบคุมการแลตช์ (latch) ค่าตำแหน่งแอดเดรสไบต์ต่ำ (Address Latch Enable) เมื่อต้องการติดต่อกับหน่วยความจำภายนอก นอกจากนี้ขานี้ยังทำหน้าที่เป็นอินพุตรับพัลซ์ในการโปรแกรม (program pulse input) ในส่วนของหน่วยความจำ EPROM สำหรับไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ที่มีหน่วยความจำโปรแกรมภายในเป็น EPROM
- ขา $\overline{\text{PSEN}}$ (Program Store Enable) ทำหน้าที่เป็นสัญญาณสโตรบเพื่ออ่านคำสั่งจากหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำสั่งจากหน่วยความจำภายนอก ขานี้จะส่งสัญญาณสโตรบจำนวน 2 ครั้งในแต่ละแมชีนไซเกิล แต่ในขณะที่ติดต่อกับหน่วยความจำข้อมูลภายนอกจะไม่มีการส่งสัญญาณสโตรบแต่อย่างใด
- ขา $\overline{\text{EA}}$/VPP (External Access enable/VPP) ขาสำหรับการเลือกใช้หน่วยความจำโปรแกรมจากภายในหรือจากภายนอก โดยถ้ามีสถานะเป็น 0 จะหมายถึงให้ไมโครคอนโทรลเลอร์รับคำสั่งจากหน่วยความจำภายนอกที่ตำแหน่งแอดเดรส 0-0FFFH (0-1FFFH ถ้าเป็นเบอร์ 8052) อย่างไรก็ตามถ้าบิตป้องกัน (security bit) ในหน่วยความจำ EPROM ถูกโปรแกรมไว้ไมโครคอนโทรลเลอร์จะไม่รับคำสั่งจากหน่วยความจำภายนอกเลย นอกจากนี้ขานี้ยังทำหน้าที่รับแรงดันไฟสำหรับการโปรแกรม (V_{PP}) ขนาด 21 โวลต์ เพื่อใช้ในระหว่างการโปรแกรม EPROM
- ขา $XTAL_1$ และขา $XTAL_2$ เป็นขาอินพุตและเอาต์พุตของวงจรอินเวอร์ติ้งออสซิลเลเตอร์แอมปลิไฟเออร์ (inverting oscillator amplifier) สำหรับใช้ต่อร่วมกับคริสตอลภายนอก

2.3.3 โครงสร้างภายในของ MCS-51



รูปที่ 2.3 แสดงโครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51

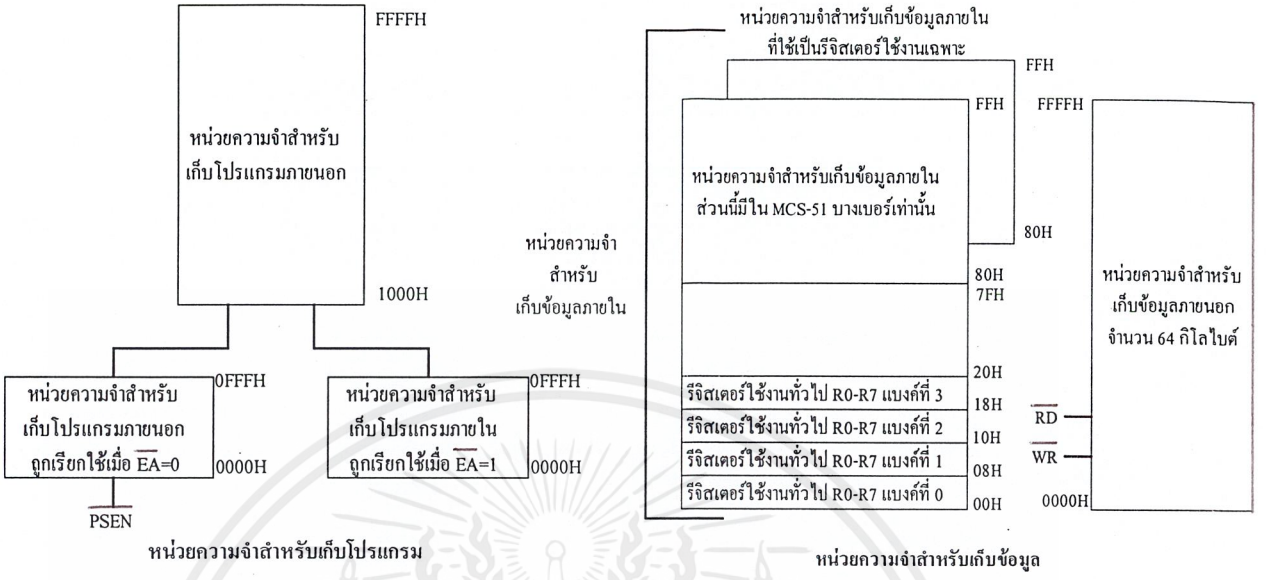
โครงสร้างภายในของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แสดงดังในรูปที่ 2.3 โดยส่วนที่มีเครื่องหมายดอกจันจะมีเฉพาะในเบอร์ 8032 และ 8052 เท่านั้น

2.3.4 การจัดหน่วยความจำ

ในไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบ่งชนิดหรือหน้าที่ของหน่วยความจำออกเป็น 2 ส่วนคือหน่วยความจำโปรแกรม (Program Memory) และหน่วยความจำข้อมูล (Data Memory)

หน่วยความจำโปรแกรมจะใช้สำหรับเก็บโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์ซึ่งบางเบอร์จะมีหน่วยความจำในส่วนนี้อยู่ภายในตัว โดยอาจจะมีขนาดไม่เท่ากันหรือเป็นหน่วยความจำต่างชนิดกัน เช่น บางเบอร์เป็น ROM และบางเบอร์อาจเป็น EPROM และบางเบอร์อาจไม่มีหน่วยความจำในส่วนนี้เลย โปรแกรมการทำงานจะถูกเก็บไว้ยังหน่วยความจำโปรแกรมภายนอกทั้งหมด

สำหรับหน่วยความจำข้อมูลจะใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่าง ๆ จากการทำงานของโปรแกรม ซึ่งใน MCS-51 ทุกเบอร์จะมีหน่วยความจำในส่วนนี้อยู่จำนวนหนึ่งแต่อาจมีขนาดมากน้อยต่างกันไปในแต่ละเบอร์สำหรับการจัดโครงสร้างของหน่วยความจำทั้งในส่วนของหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลแสดงไว้ดังรูปที่ 2.4



รูปที่ 2.4 แสดงการจัดโครงสร้างของหน่วยความจำ

2.3.4.1 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมสามารถแบ่งออกได้เป็น 2 ส่วนคือ หน่วยความจำโปรแกรมภายในและหน่วยความจำโปรแกรมภายนอก หน่วยความจำโปรแกรมภายในจะถูกเลือกใช้งานถ้าขาสัญญาณ $\overline{EA}$ มีค่าเป็น 1 โดยจะถูกใช้งานในช่วงแอดเดรส 0-0FFFH (หรือช่วงแอดเดรส 0-0FFFH ในเบอร์ 8052) นอกเหนือจากช่วงแอดเดรสนี้จะใช้หน่วยความจำโปรแกรมภายนอกทั้งหมด ในกรณีตรงกันข้ามถ้าขาสัญญาณ $\overline{EA}$ มีค่าเป็น 0 ในช่วงแอดเดรส 0-0FFFH (หรือช่วงแอดเดรส 0-1FFFH ในเบอร์ 8052) จะถูกใช้จากหน่วยความจำภายนอก หรือกล่าวได้ว่าถ้าขาสัญญาณ $\overline{EA}$ มีค่าเป็น 0 จะเป็นการเลือกใช้หน่วยความจำโปรแกรมภายนอกทั้งหมดตลอดช่วงแอดเดรส

2.3.4.2 หน่วยความจำข้อมูล

หน่วยความจำข้อมูลสามารถแบ่งออกได้เป็น 2 ส่วน คือ หน่วยความจำข้อมูลภายในและหน่วยความจำข้อมูลภายนอก สำหรับหน่วยความจำข้อมูลภายในยังแบ่งออกได้เป็น 2 ส่วนย่อย คือ ส่วนที่ใช้เก็บข้อมูลทั่วไปและส่วนที่ใช้เป็นรีจิสเตอร์หน้าที่พิเศษหรือ SFR (Special Function Register) โดยส่วนที่ใช้เก็บข้อมูลทั่วไปจะถูกใช้สำหรับเก็บข้อมูลหรือค่าตัวแปรต่างๆ จากการทำงานของโปรแกรม ส่วนรีจิสเตอร์หน้าที่พิเศษจะถูกใช้งานเป็นรีจิสเตอร์ควบคุมการทำงานและบอกสถานะการทำงานของไมโครคอนโทรลเลอร์

ตำแหน่งแอดเดรส	(MSB)			บิตแอดเดรส				(LSB)	รีจิสเตอร์หน้าที่พิเศษ
	WDT	T32	SERR	IZC	P3HZ	P2HZ	P1HZ	ALF	
0F8H	FF	FE	FD	FC	FB	FA	F9	F8	IOCON
0F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
0E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
	CY	AC	F0	RS1	RS0	0V	F1	P	
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
0CDH	ไม่สามารถเข้าถึงได้ระดับบิต								TH2
0CCH	ไม่สามารถเข้าถึงได้ระดับบิต								TL2
0CBH	ไม่สามารถเข้าถึงได้ระดับบิต								RCAP2H
0CAH	ไม่สามารถเข้าถึงได้ระดับบิต								RCAP2L
	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$	
0C8H	CF	CE	CD	CC	CB	CA	C9	C8	T2CON
	PCT		PT2	PS	PT1	PX1	PT0	PX0	
0B8H	BF	-	BD	BC	BB	BA	B9	B8	IP
0B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
	EA		ET2	ES	ET1	EX1	ET0	EX0	
0A8H	AF	-	AD	AC	AB	AA	A9	A8	IE
0A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
99H	ไม่สามารถเข้าถึงได้ระดับบิต								SBUF
	SM0	SM1	SM2	REN	TB8	RB8	T1	R1	
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
8DH	ไม่สามารถเข้าถึงได้ระดับบิต								TH1
8CH	ไม่สามารถเข้าถึงได้ระดับบิต								TH0
8BH	ไม่สามารถเข้าถึงได้ระดับบิต								TL1
8AH	ไม่สามารถเข้าถึงได้ระดับบิต								TL0
89H	ไม่สามารถเข้าถึงได้ระดับบิต								TMOD
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
87H	ไม่สามารถเข้าถึงได้ระดับบิต								PCON
83H	ไม่สามารถเข้าถึงได้ระดับบิต								DPH
82H	ไม่สามารถเข้าถึงได้ระดับบิต								DPL
81H	ไม่สามารถเข้าถึงได้ระดับบิต								SP
80H	87	86	85	84	83	82	81	80	P0

รูปที่ 2.5 แสดงการจัดหน่วยความจำและตำแหน่งของรีจิสเจอร์หน้าที่พิเศษต่างๆ

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์เป็นอย่างน้อย และบางเบอร์อาจมีถึงขนาด 256 ไบต์

รีจิสเตอร์หน้าที่พิเศษ (SFR)

รีจิสเตอร์หน้าที่พิเศษมีบทบาทอย่างมากในการควบคุมการทำงานของไมโครคอนโทรลเลอร์และทำให้การเขียนโปรแกรมสามารถทำได้สะดวกมากขึ้น รีจิสเตอร์หน้าที่พิเศษทำหน้าที่สำคัญคือควบคุมการทำงานในส่วนต่าง ๆ ภายในไมโครคอนโทรลเลอร์และทำหน้าที่แสดงสถานะการทำงาน ซึ่งในรีจิสเตอร์หน้าที่พิเศษบางตัวยังสามารถเข้าถึงได้ในระดับบิต (bit addressable) ด้วย ดังแสดงรูปการจัดหน่วยความจำและตำแหน่งของรีจิสเตอร์หน้าที่พิเศษต่าง ๆ ในรูปที่ 2.5

รีจิสเตอร์ใช้งานทั่วไป

รีจิสเตอร์ใช้งานทั่วไปมีไว้สำหรับให้ผู้เขียนโปรแกรมสามารถนำข้อมูลไปพักไว้ชั่วคราวหรือใช้งานทั่วไปได้ตามต้องการ ซึ่งรีจิสเตอร์ใช้งานทั่วไปนี้มีอยู่ด้วยกัน 8 ตัว คือรีจิสเตอร์ R_0-R_7 โดยรีจิสเตอร์ทั้ง 8 ตัวถูกจัดให้อยู่รวมกันและมีให้เลือกใช้ถึง 4 แบงค์ (bank) นั่นคือมีรีจิสเตอร์ใช้งานทั่วไปถึง 32 ตัวให้ใช้งาน เพียงแต่การเลือกใช้รีจิสเตอร์ R_0-R_7 ในแบงค์ใดแบงค์หนึ่งจะถูกกำหนดจากบิต RS0, RS1 ในรีจิสเตอร์หน้าที่พิเศษ PSW ดังนั้นการเลือกใช้จึงเลือกได้เพียงแบงค์เดียวในขณะใดขณะหนึ่ง อย่างไรก็ตามค่าข้อมูลที่เก็บได้ไว้ในรีจิสเตอร์แบงค์ใดก็ตามที่มีชื่อเดียวกันแต่อยู่คนละแบงค์จะไม่มีผลซึ่งกันและกันเลย ทำให้ผู้เขียนโปรแกรมใช้งานรีจิสเตอร์ทั่วไปนี้ได้ทั้ง 32 ตัวอย่างเต็มที่และไม่ยุ่งยากในการเขียนโปรแกรม

2.4 การรับส่งข้อมูลผ่าน Serial port

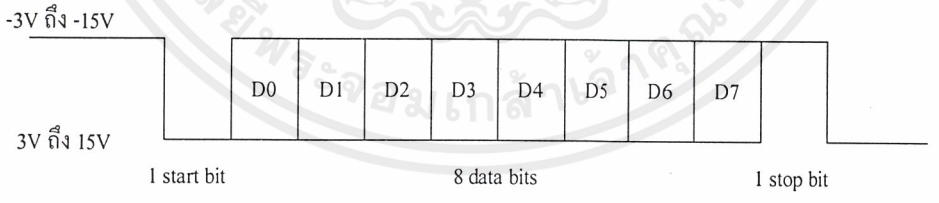
การสื่อสารข้อมูลอนุกรมนิยมนำมาใช้ในการโอนถ่ายข้อมูล ระหว่างระบบควบคุมด้วยไมโครคอนโทรลเลอร์หรืออุปกรณ์คอมพิวเตอร์ต่างๆ เพราะใช้เส้นสัญญาณเพียง 2 หรือ 3 เส้นสำหรับการรับส่งข้อมูลเท่านั้น โดยจะเป็นการรับส่งข้อมูลทีละบิต จนครบหนึ่งไบต์ คือ D_0-D_7 โดยจะส่ง D_0 ออกไปก่อนแล้วตามด้วย D_1 ไปเรื่อย ๆ จนถึง D_7 การส่งข้อมูลแบบอนุกรมนี้จะช้ากว่าการส่งแบบขนาน แต่มีข้อดีคือถ้าเป็นการส่งแบบขนานที่ต้องส่งข้อมูลระยะไกลจะทำให้สิ้นเปลืองสายสัญญาณมาก แต่เนื่องจากการประมวลข้อมูลในระบบคอมพิวเตอร์เป็นลักษณะแบบขนาน ทำให้ต้องมีการแปลงผันข้อมูลให้เหมาะสมก่อนการส่งข้อมูล รวมทั้งภายหลังการรับข้อมูลแบบอนุกรมด้วย

ภายใน MCS-51 จะมีตัวรับส่งข้อมูลอนุกรมอยู่ภายใน โดยตัวส่งข้อมูลจะทำหน้าที่แหล่งข้อมูลจากการประมวลผลที่เป็นแบบขนานให้เป็นข้อมูลแบบอนุกรมก่อน (Parallel-to-serial conversion) แล้วส่งข้อมูลออกมาโดยมีสัญญาณนาฬิกาเป็นตัวกระตุ้น ตัวแปลงข้อมูลนี้อาจพิจารณาง่ายๆ ว่าเป็นชิฟรีจิสเตอร์ (shift register) ที่ทำหน้าที่เลื่อนข้อมูลออกมา ถ้าสัญญาณนาฬิกามีความถี่สูงจะทำให้ส่งข้อมูลได้เร็ว ส่วนตัวรับข้อมูลจะเป็นการรับข้อมูลแบบอนุกรมและแปลงให้เป็นข้อมูลแบบขนานเพื่อใช้ในการประมวลผล (Serial-to-parallel conversion) ใน MCS-51 ตัวแปลงข้อมูลทั้งสองประเภทนี้จะเป็นรีจิสเตอร์ภายใน

การส่งข้อมูลอนุกรมด้วยชุดคำสั่งของ MCS-51 นั้น ทำได้ง่ายเพียงการเขียนโปรแกรมโอนถ่ายข้อมูลแบบขนานให้กับรีจิสเตอร์ที่จัดการข้อมูลอนุกรมเท่านั้น หรือการรับข้อมูลก็อ่านจากรีจิสเตอร์นี้เช่นกัน โดยไม่ต้องเขียนโปรแกรมเพื่อแปลงผันข้อมูลระหว่างอนุกรมและขนาน รวมทั้งการเลื่อนบิต (shift bits) จากขาสัญญาณแต่อย่างใด จึงนับว่าอำนวยความสะดวกให้กับผู้ใช้งานมาก

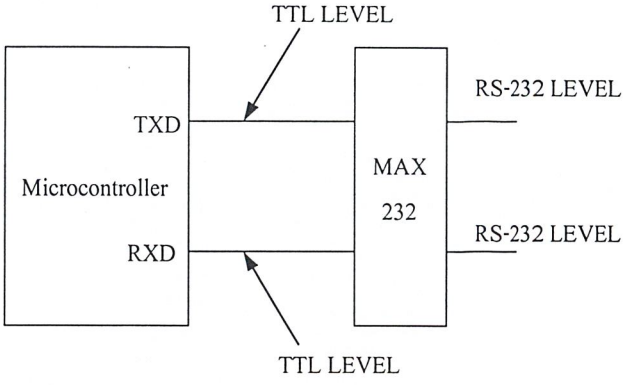
2.4.1 พอร์ตสื่อสารข้อมูลอนุกรม RS-232C

รูปแบบของข้อมูลอนุกรมที่ใช้งานกันโดยแพร่หลายเป็นลักษณะ การสื่อสารแบบอะซิงโครนัส (Asynchronous communication) และเป็นไปตามมาตรฐาน RS-232C รูปแบบของข้อมูลภายในเส้นสัญญาณจะมีบิตข้อมูลเพิ่มขึ้นจากบิตข้อมูลสื่อสารจริง เช่น บิตเริ่มต้น (start bit) และบิตสิ้นสุด (stop bit) เพื่อนำมาใช้ในการบอกจังหวะการเริ่มต้นและสิ้นสุดข้อมูลแต่ละไบต์



รูปที่ 2.6 รูปแบบสัญญาณข้อมูลอนุกรมแบบอะซินโครนัส โดยมีระดับสัญญาณแบบ RS-232C

ตามมาตรฐาน RS-232C ระดับสัญญาณข้อมูลของวงจร (TTL voltage levels) ซึ่งมีค่าอยู่ในช่วง 0 ถึง 5 โวลต์จะต้องนำมาปรับให้เป็นระดับแบบ RS-232C (RS-232 voltage levels) ซึ่งมีค่าสูงขึ้นในช่วงจาก +15 ถึง –15 โวลต์ รูปที่ 2.7 แสดงการเชื่อมต่อ RS-232C ของไมโครคอนโทรลเลอร์โดยมี IC MAX232 เป็นตัวปรับระดับสัญญาณ (RS-232C Line driver and receiver)



รูปที่ 2.7 การเชื่อมต่อสัญญาณอนุกรมของ Microcontroller

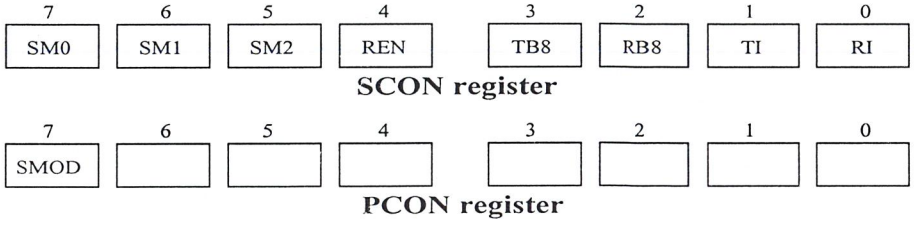
สำหรับความเร็วของการส่งข้อมูลอนุกรมมักจะเรียกกันว่า อัตราบอต (Baud rate) ซึ่งอธิบายโดยง่ายสำหรับรูปแบบข้อมูลอนุกรมแบบอะซิงโครนัสว่า เป็นจำนวนของบิตข้อมูลภายในหนึ่งช่วงเวลา อัตราบอตที่ใช้งานกันโดยทั่วไป เช่น 1200, 2400, 4800 และ 9600 เป็นต้น

2.4.2 การใช้งานพอร์ตอนุกรมของ 89C51

รีจิสเตอร์ที่เกี่ยวข้องกับการสื่อสารอนุกรมของ 89C51 ประกอบด้วย รีจิสเตอร์ SBUF สำหรับเก็บข้อมูลสื่อสารรีจิสเตอร์ SCON ควบคุมการสื่อสาร รีจิสเตอร์ PCON ควบคุมความเร็วการสื่อสาร และขาสัญญาณ RXD ($P_{3.1}$) เป็นขาสัญญาณที่ต่อเข้ากับอุปกรณ์หรือวงจรภายนอก

เนื่องจากการส่งข้อมูลหนึ่งไบต์ออกไปทางพอร์ตอนุกรมในคราวหนึ่งๆ จะใช้เวลามาก ดังนั้นเพื่อไม่ให้เสียเวลาในการประมวลผลไป 89C51 จึงนำแฟล็ก TI และ RI ซึ่งเป็นตำแหน่งบิตอยู่ภายในรีจิสเตอร์ SCON เพื่อแจ้งให้กับผู้ใช้งานทราบ โดยเมื่อใดก็ตามที่มีการส่งข้อมูลแบบอนุกรมเสร็จสิ้น บิตแฟลก TI จะมีค่าเป็น 1 หรือเมื่อได้รับข้อมูลอนุกรมเข้ามาแฟลก RI ก็จะมีค่าเป็น 1 เช่นกัน

การทำงานของพอร์ตอนุกรมของ MCS-51 มีหลายโหมด โดยเราสามารถโปรแกรมการทำงานได้ในรีจิสเตอร์ SCON โดยความหมายของแต่ละบิตเป็นดังนี้



รูปที่ 2.8 ชื่อและตำแหน่งของบิตต่างๆภายในรีจิสเตอร์ SCON และ PCON (เฉพาะเกี่ยวกับการส่งข้อมูลอนุกรม)

ตารางที่ 2.3 ความหมายบิตต่างๆของรีจิสเตอร์ SCON

ชื่อบิต	ตำแหน่งบิต	ความหมาย
SM0	9FH	บิตเลือกโหมดการทำงานบิต 0
SM1	9EH	บิตเลือกโหมดการทำงานบิต 1
SM2	9DH	บิตเลือกโหมดการทำงานบิต 2
REN	9CH	บิตแฟลกกำหนดการยอมให้มีการรับข้อมูล
TB8	9BH	ค่าของบิต 9 สำหรับการส่งข้อมูลในโหมด 2 และ 3 สามารถ set และ clear ได้โดยโปรแกรม
RB8	9AH	ค่าของบิต 9 เมื่อรับข้อมูลเข้ามา
TI	99H	บิตแฟลกแสดงการอินเตอร์รัปต์ภายหลังการส่งข้อมูลออกไปโดยจะ set เมื่อส่งข้อมูลออกไปหมดแล้ว และจะสามารถ clear ได้โดยโปรแกรม
RI	98H	แฟลกแสดงการอินเตอร์รัปต์ภายหลังรับข้อมูลเข้ามา สามารถ clear ได้โดยโปรแกรม

ตารางที่ 2.4 โหมดต่างๆของการรับส่งแบบอนุกรม

SM0	SM1	โหมด	ความหมาย	อัตราบอด
0	0	0	ชิพรีจิสเตอร์	เปลี่ยนแปลงไม่ได้
0	1	1	8 บิต UART	เปลี่ยนแปลงได้โดยการกำหนดจาก Timer
1	0	2	9 บิต UART	เปลี่ยนแปลงไม่ได้
1	1	3	9 บิต UART	เปลี่ยนแปลงได้โดยการกำหนดจาก Timer

ในการใช้พอร์ตอนุกรมจะต้องโปรแกรมให้กับรีจิสเตอร์ SCON เสียก่อนเพื่อกำหนดโหมดการทำงานและลักษณะต่างๆเช่น

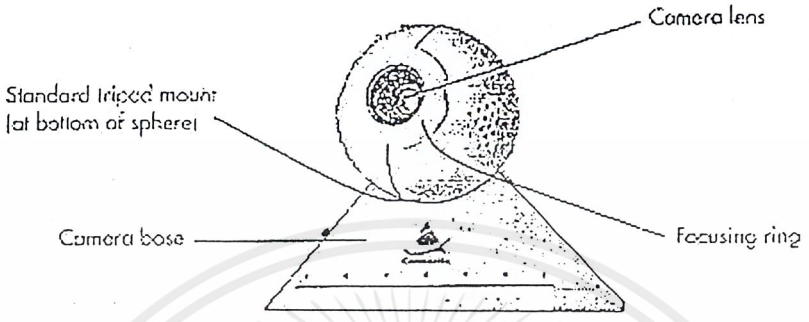
MOV SCON,#01010010B

เป็นการกำหนดให้พอร์ตอนุกรมทำงานในโหมด 1 และอีนาเบิลให้มีการรับข้อมูล พร้อมกับกำหนดให้ TI เป็นลอจิก 1

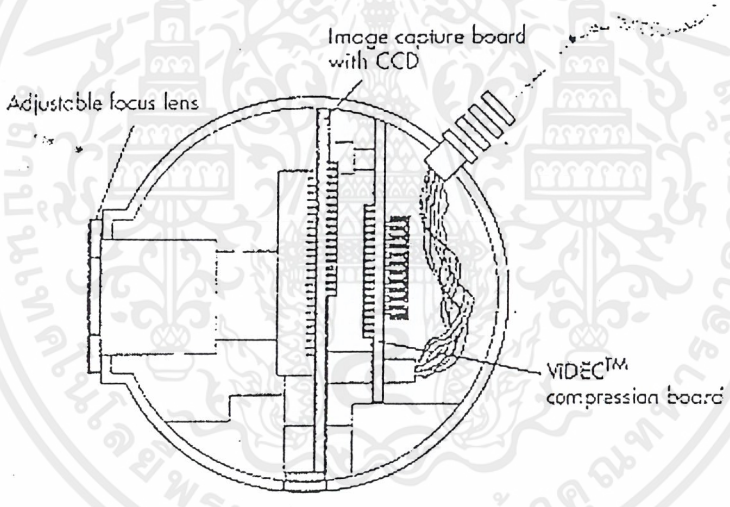
ในการส่งข้อมูลทุกโหมดสามารถทำได้โดยเขียนข้อมูลไปยัง SBUF เมื่อข้อมูลถูกส่งไปแล้วบิต TI จะถูกเซ็ตเป็น 1 ในการส่งข้อมูลจะต้องคอยตรวจสอบบิต TI เพราะว่าถ้า TI ยังไม่เป็น 1 แสดงว่าข้อมูลยังส่งไม่หมด ถ้าหากมีการเขียนข้อมูลไบต์ต่อไปไปยังริสเตอร์ SBUF จะทำให้เกิดข้อ

ผิดพลาดขึ้นสำหรับในการรับข้อมูลบิต REN จะต้องเซ็ตให้เป็น 1 ยกเว้นโหมด 0 เพื่ออนุญาตให้รอรับข้อมูลได้ เมื่อข้อมูลรับเข้ามาเรียบร้อยแล้วบิต RI จะถูกเซ็ตให้เป็น 1

2.5.กล้อง QuickCam



รูปที่ 2.9 แสดงลักษณะภายนอกของกล้อง QuickCam



รูปที่ 2.10 แสดงลักษณะภายในของกล้อง QuickCam

2.5.1 สิ่งที่จำเป็นต้องมีในการใช้งานกล้อง QuickCam ได้แก่

- เครื่องคอมพิวเตอร์ตั้งแต่รุ่น Pentium 200 ขึ้นไป
- Microsoft Windows 98
- หน่วยความจำ 16 MB
- Video: 16-bit color display
- CD-ROM drive
- พอร์ต USB 1 พอร์ต

- ไมโครโฟนสำหรับการบันทึกเสียงในภาพแบบ Movie
- Sound Card ซึ่ง Compatible กับ Windows

2.5.2 คุณสมบัติพิเศษของกล้อง QuickCam

- ให้ภาพที่มีความละเอียดสูงสุดที่ 352x288 pexels 24 bit color
- Frame Rate 15fps (เปลี่ยนแปลงได้ตามขนาดของ Windows และไมโครคอมพิวเตอร์)
- ได้รับไฟเลี้ยงจากพอร์ต USB
- ปรับโฟกัสได้จาก 1 นิ้ว ถึง Infinity (Maximum Len Speed = f/1.6)

2.5.3 ความสามารถของกล้อง QuickCam

1) เก็บภาพเคลื่อนไหวในคอมพิวเตอร์
การใช้งานกล้อง QuickCam นั้นเป็นวิธีที่ง่ายและประหยัดวิธีหนึ่งในการเก็บภาพเคลื่อนไหวไว้ในไมโครคอมพิวเตอร์ ทำให้สามารถที่จะติดต่อสื่อสารกัน โดยแสดงภาพเคลื่อนไหวได้และสามารถหลีกเลี่ยงปัญหาในขั้นตอนการติดตั้งฮาร์ดแวร์ การซื้อ Video Board ที่แพงหรือการศึกษาถึงวิธีการใช้งาน การใช้กล้อง QuickCam นั้นจะช่วยให้เราสามารถที่จะ Capture ภาพได้ง่ายและทำการดัดแปลงแก้ไขได้ด้วย

2) เก็บภาพนิ่งในไมโครคอมพิวเตอร์
กล้อง QuickCam มีซอฟแวร์ที่ทำให้สามารถบันทึกภาพนิ่งให้อยู่ในรูปแบบ BMP,TIFF or JPEG ซึ่งสามารถนำมาประมวลผลใน Word Processing, Page Layout, การ Present งาน หรือ โปรแกรม Graphics ซึ่งภาพนิ่งเหล่านั้นสามารถนำไปใช้ในการบอกตำแหน่ง การใช้ในงานพิมพ์ Graphics ที่ต้องการคุณภาพสูง เช่น จดหมาย หรือเอกสารสำคัญต่าง ๆ

2.6 เซอร์โวมอเตอร์

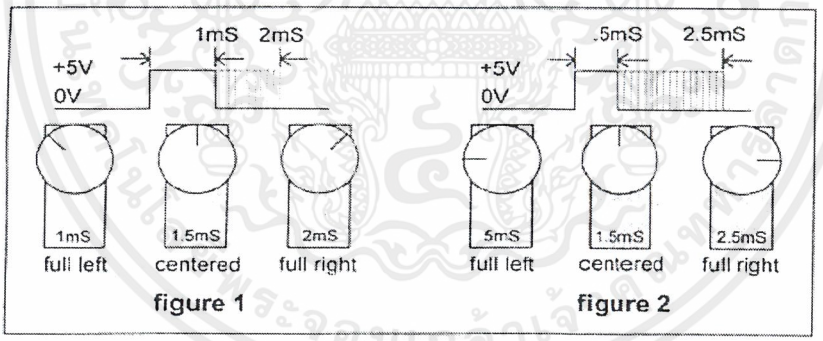
2.6.1 คุณสมบัติ

- Signal Pulse = Positive pulse 1520 uS. neutral
- ต้องการแหล่งจ่ายไฟที่จ่ายแรงดันไม่น้อยกว่า 4.8 โวล์ต
- มีความเร็วในการหมุน 0.19 วินาทีต่อ 60 องศา
- ให้แรงบิด 42 oz.-in. (3.0 kg.-cm.)
- มีขนาด 1.6" x 0.8" x 1.4" (41 mm. x 20 mm. x 36 mm.)
- มีน้ำหนัก 1.75 oz. (49 g.)

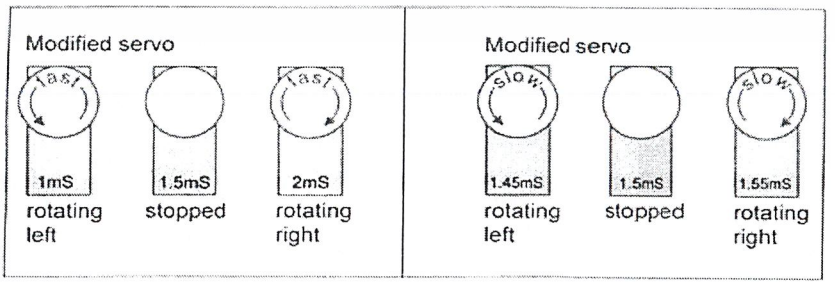
- Current Draw (Idle) = 9.7 mA.
- Current Draw (Moving) = 130 mA.

2.6.2 รายละเอียดทั่วไป

RC (remote control) Servo คืออุปกรณ์สำหรับการขับเคลื่อนหุ่นยนต์หรืองานที่ต้องการการกำหนดการหมุนเป็นช่วงๆ หรือตามองศาที่ต้องการ โดยกำหนดการหมุนในลักษณะครึ่งวงกลม โดยจะรับสัญญาณพัลซ์ (Pulse) ที่ตำแหน่งซ้ายสุด 0.5 mS , เซ็นเตอร์ 1.5 mS ,และขวาสุด 2.5 mS ภายในตัวเซอร์โวมอเตอร์ประกอบด้วยแผงควบคุมซึ่งทำหน้าที่รับสัญญาณจากไมโครคอนโทรลเลอร์มาควบคุมการหมุนหรือเคลื่อนที่ไปยังตำแหน่งที่ต้องการ ชุดเกียร์ (Gear) ที่ติดตั้งไว้ภายในประกอบด้วยเฟืองพลาสติก ซึ่งทำหน้าที่เพิ่มกำลังหรือแรงบิด ให้กับตัวเซอร์โวมอเตอร์ การรับสัญญาณพัลซ์จากไมโครคอนโทรลเลอร์จะรับสัญญาณเพียง 1 I/O เท่านั้น จึงประหยัดขา I/O ได้มากกว่าการใช้สเตปมอเตอร์ (Stepmotor) ติดตั้งง่ายเพราะมีลักษณะเป็นสี่เหลี่ยมและมีขนาด 1.6 x 0.8 x 1.4 นิ้ว น้ำหนัก 1.75 oz. (49 g) รับแรงดันไฟตรงได้ตั้งแต่ 4.8 - 6 โวล์ต กินกระแส 9.7 mA. (ขณะหยุดนิ่ง) ,130 mA. (ขณะหมุน) มอเตอร์หมุนได้ทั้งซ้ายและขวา เซอร์โวมอเตอร์สามารถนำไปใช้ร่วมกับบอร์ด รับสัญญาณได้โดยตรงทั้งจากพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ หรือรับสัญญาณจากไมโครคอนโทรลเลอร์



รูปที่ 2.11 แสดงการกำหนดช่วงพัลซ์ ของเซอร์โวที่ยังไม่ได้ถูกแก้ไข



รูปที่ 2.12 แสดงการกำหนดช่วงพัลซ์ของเซอร์โวที่ถูกแก้ไขแล้ว

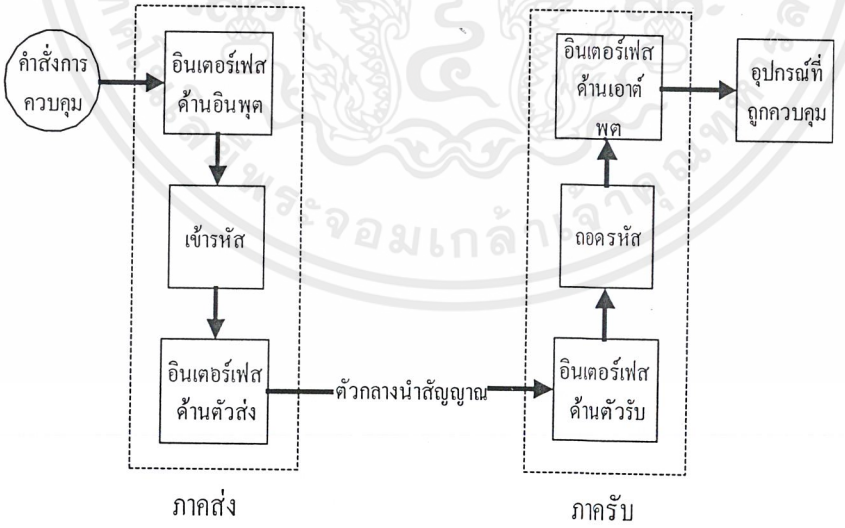
2.6.3 การทำงานของ เซอร์โวมอเตอร์

เซอร์โวมอเตอร์จะรับสัญญาณพัลซ์ เพื่อใช้ในการกำหนดการหมุน สัญญาณพัลซ์ที่ได้มีความกว้าง 1.5 mS จะทำให้เซอร์โวมอเตอร์กลับมาอยู่ที่ตำแหน่งกึ่งกลาง เมื่อสัญญาณพัลซ์มีความกว้างเพิ่มมากขึ้น เช่นถ้าต้องการให้เซอร์โวมอเตอร์หมุนมาทางขวา ก็ต้องเพิ่มความกว้างของช่วงพัลซ์จาก 1.5 mS เพิ่มมาเป็น 1.8 , 1.9 , 2.0 , 2.1 เซอร์โวมอเตอร์ก็จะหมุนตามมาทางขวาเรื่อยๆ เมื่อเพิ่มความกว้างของช่วงพัลซ์ ไปจนถึง 2.5 mS ซึ่งเป็นตำแหน่งที่เซอร์โวมอเตอร์จะหมุนไปที่ตำแหน่งขวาสุด ในทำนองกลับกันถ้าต้องการให้เซอร์โวมอเตอร์หมุนไปทางซ้าย ก็ต้องลดช่วงพัลซ์ให้แคบลงไปเรื่อยๆ จนไปถึง 0.5 mS เซอร์โวมอเตอร์ก็จะหมุนมาทางซ้ายสุด

หมายเหตุ : แรงดันไฟตรงที่ใช้กับเซอร์โวมอเตอร์ใช้ได้ตั้งแต่ 4.8 โวล์ต ไปจนถึง 6.0 โวล์ต ห้ามจ่ายแรงดันไฟมากไปกว่านี้เพราะจะมีผลทำให้แผงวงจรที่อยู่ภายในเซอร์โวมอเตอร์เกิดความเสียหายได้

2.7 หลักการของรีโมตคอนโทรล

บล็อกไดอะแกรมในรูปที่ 2.15 แสดงโครงสร้างและหลักการทำงานของระบบควบคุมระยะไกลโดยทั่วไป ในลักษณะของการควบคุมแบบทางเดียว เริ่มจากตัวกำหนดคำสั่งที่ใช้สำหรับการควบคุมว่ามีคำสั่งอะไรบ้าง ชุดคำสั่งทั้งหมดมีกี่คำสั่งเป็นต้น เมื่อมีการกำหนดรูปแบบของคำสั่งแล้ว รูปแบบของคำสั่งที่ถูกเลือก จะถูกส่งไปยังภาคส่งสัญญาณที่ทำหน้าที่แปลงสัญญาณ หรือรวมสัญญาณควบคุมให้มีรูปแบบที่เหมาะสมกับวงจร



รูปที่ 2.13 บล็อกไดอะแกรมแสดงโครงสร้างและหลักการทำงานของระบบรีโมตคอนโทรล

โดยอาจทำการเข้ารหัสสัญญาณให้แต่ละคำสั่งมีรหัสเฉพาะของตัวมันเองให้เป็นสัญญาณทางไฟฟ้าก่อนที่จะถูกส่งออกไปยังภาครับ โดยตัวอินเตอร์เฟสด้านตัวส่งเพื่อทำหน้าที่ส่งสัญญาณที่

โดยอาจทำการเข้ารหัสสัญญาณให้แต่ละคำสั่งมีรหัสเฉพาะของตัวมันเองให้เป็นสัญญาณทางไฟฟ้าก่อนที่จะถูกส่งออกไปยังภาครับโดยตัวอินเตอร์เฟสด้านตัวส่งเพื่อทำหน้าที่ส่งสัญญาณที่ภาครับต้องเข้าใจได้ นั่นก็คือต้องเป็นระบบเดียวกัน สัญญาณไฟฟ้า สัญญาณแสง หรือสัญญาณเสียงความถี่สูง สัญญาณนี้สามารถเดินทางผ่านตัวกลางที่เป็นสายนำสัญญาณ หรือผ่านตัวกลางอากาศ ขึ้นอยู่กับระบบที่ถูกออกแบบ

หากใช้สายสัญญาณเป็นตัวนำสัญญาณจะเรียกว่า ระบบใช้สาย ซึ่งถ้าใช้สัญญาณไฟฟ้าเป็นสัญญาณควบคุม (ที่มีการจัดรูปแบบหรือเข้ารหัสแล้ว) ก็จะใช้สายไฟฟ้าเป็นตัวนำสัญญาณ แต่ถ้าหากใช้สัญญาณแสงเป็นตัวควบคุม ตัวนำสัญญาณจะเป็นเส้นใยแก้วนำแสงหรือไฟเบอร์ออปติก ในกรณีที่สัญญาณควบคุมถูกส่งไปในอากาศ เพื่อเดินทางไปยังเครื่องรับ เช่น การใช้สัญญาณไฟฟ้าในรูปของคลื่นวิทยุ หรือการใช้สัญญาณแสงอินฟาเรดโดยตรง ระบบจะมีชื่อว่า ระบบไร้สาย ระบบนี้เองที่กำลังเป็นที่นิยมใช้กันอยู่มากในปัจจุบัน

สัญญาณที่เข้ามายังเครื่องรับหรือภาครับ จะถูกตัวอินเตอร์เฟสทำหน้าที่แปลงสัญญาณ ให้อยู่ในรูปของสัญญาณไฟฟ้าที่เข้ากับระบบของตัวรับ ก่อนถูกถอดรหัสเพื่อทราบวัตถุประสงค์ของคำสั่ง จากนั้นส่วนของวงจรอินเตอร์เฟสด้านเอาต์พุตจะทำหน้าที่ควบคุมการทำงานของอุปกรณ์ที่ต้องการ ตามลักษณะคำสั่งที่ได้รับ

ระบบที่กล่าวถึงมานั้น เป็นระบบรีโมตคอนโทรลหรือการควบคุมระยะไกลแบบทางเดียว ที่มีการสั่งงานจากจุดหนึ่งแล้วเกิดการทำงานขึ้นอีกจุดหนึ่ง หากจุดที่ถูกสั่งให้ทำงาน มีความสามารถในการสั่งการกลับมายังจุดเริ่มต้นให้ทำงานได้ด้วยแล้ว แสดงว่าการทำงานของจุดหรือตำแหน่งทั้งสองมีความเสมอภาคกัน อย่างนี้ถือเป็นระบบควบคุมระยะไกลแบบสองทางซึ่งมักปรากฏให้เห็นอย่างชัดเจนในระบบการสื่อสารทั่วไป

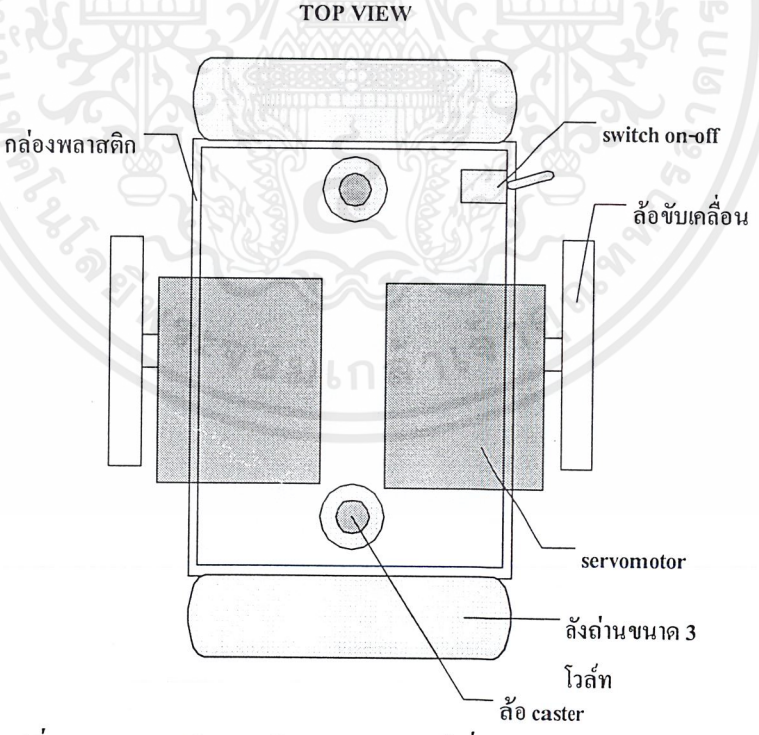
บทที่ 3
การออกแบบและส่วนประกอบของโครงงาน

3.1. หุ่นยนต์

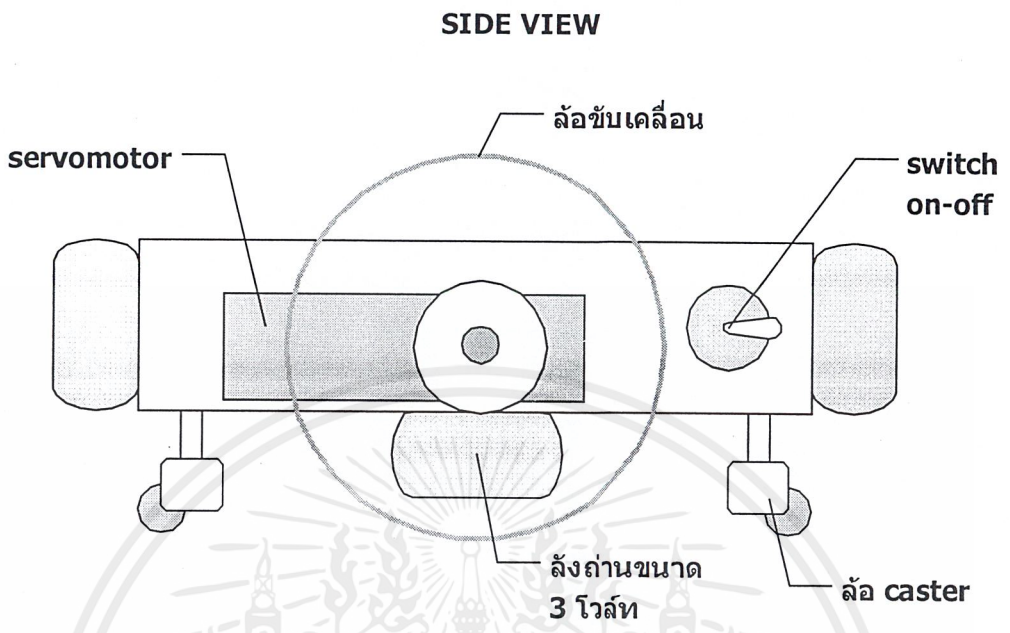
โครงหุ่นยนต์ทำจากกล่องพลาสติกใสขนาดเล็กซึ่งเจาะรูให้มอเตอร์ยื่นออกมาได้ หุ่นยนต์จะประกอบได้ด้วยล้อขับเคลื่อน 2 ล้อและล้ออิสระอีก 2 ล้อ ล้อขับเคลื่อนจะวางตัวอยู่ตรงแนวกลางของหุ่นยนต์เพื่อให้หุ่นยนต์หมุนเป็นวงกลมได้ ส่วนล้ออิสระจะวางในแนวตั้งฉากกับล้อขับเคลื่อนเพื่อให้หุ่นยนต์สมดุล

ใช้ถ่านไฟฉาย Alkaline ขนาด AA 6 ก้อนเป็นแหล่งจ่ายไฟ จ่ายไฟรวมทั้งหมด 9 โวล์ต โดยเราจะติดลังถ่าน 2 อันไว้ที่ด้านหน้าและด้านหลังของหุ่นยนต์ ส่วนอีกอันติดที่ข้างใต้ เหตุผลที่เราติดลังถ่านไว้แบบนี้ เพื่อให้หุ่นยนต์อยู่ในสภาพสมดุล ลดภาระของล้อไม่ให้ล้อใดล้อหนึ่งรับน้ำหนักมากเกินไป

รูปข้างล่างแสดงโครงสร้างของหุ่นยนต์ที่ออกแบบไว้ โดยจะแสดงเฉพาะมุมมองข้างบนและด้านข้างเท่านั้น

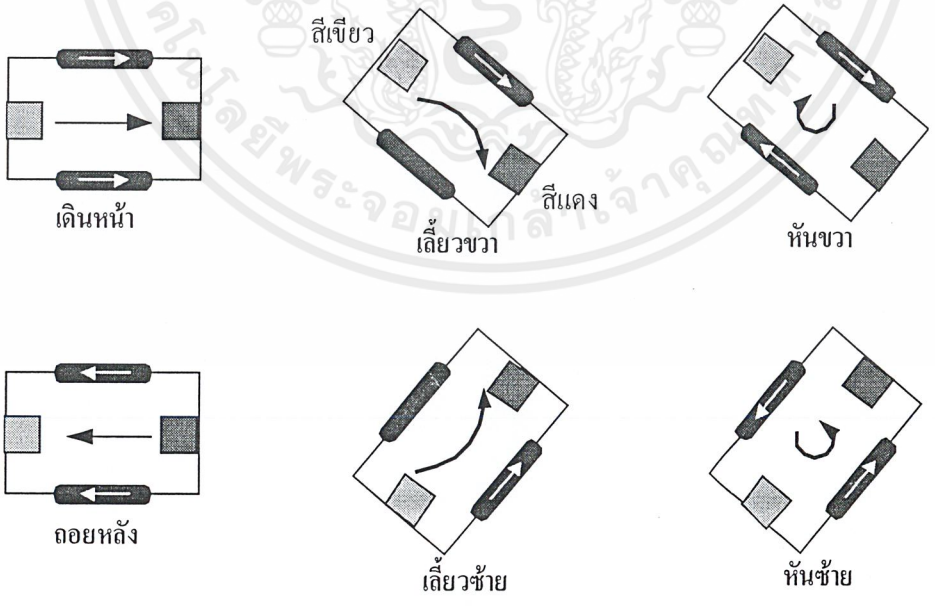


รูปที่ 3.1 แสดงรูปโครงสร้างของหุ่นยนต์เมื่อมองจากข้างบน



รูปที่ 3.2 แสดงรูปโครงสร้างของหุ่นยนต์เมื่อมองจากด้านข้าง

การออกแบบการวางล้อแบบนี้ก็เพื่อให้ง่ายต่อการควบคุมทิศทางของหุ่นยนต์ พิจารณาได้จากรูปข้างล่าง



รูปที่ 3.3 แสดงการเคลื่อนที่ในรูปแบบต่างๆของหุ่นยนต์

จากรูปที่ผ่านมาเราสามารถแบ่งการเคลื่อนที่ของหุ่นยนต์ได้เป็น 6 แบบคือ

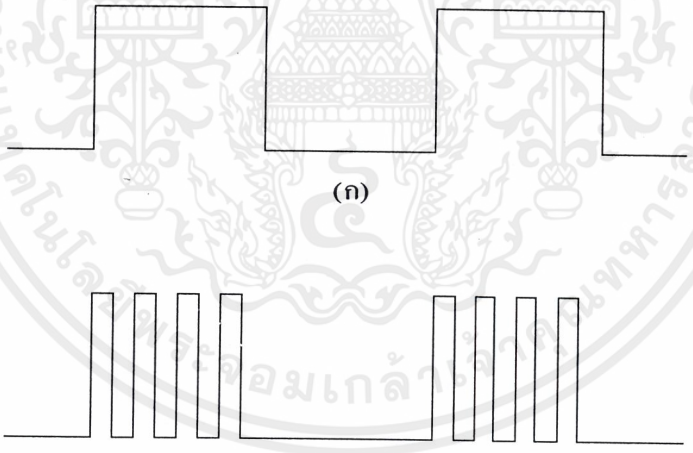
1. **เดินหน้า** สามารถทำได้โดยการสั่งให้มอเตอร์ทั้งคู่หมุนไปข้างหน้า
2. **ถอยหลัง** ทำได้โดยการสั่งให้มอเตอร์ทั้งคู่หมุนไปข้างหลัง
3. **เลี้ยวขวา** ทำได้โดยการสั่งให้มอเตอร์ทางซ้ายหมุนไปหน้า ส่วนมอเตอร์ทางขวาอยู่กับที่
4. **เลี้ยวซ้าย** ทำได้โดยการสั่งให้มอเตอร์ทางขวาหมุนไปหน้า ส่วนมอเตอร์ทางซ้ายอยู่กับที่
5. **หันขวา** ทำได้โดยการสั่งให้มอเตอร์ทางซ้ายหมุนไปหน้า ส่วนมอเตอร์ทางขวาหมุนกลับหลัง
6. **หันซ้าย** ทำได้โดยการสั่งให้มอเตอร์ทางซ้ายหมุนกลับหลัง ส่วนมอเตอร์ทางขวาหมุนไปหน้า

ในโครงงานนี้เราจะใช้เฉพาะรูปแบบที่ 1,2,5 และ6 เท่านั้น เนื่องจากจะทำให้การหมุนของหุ่นยนต์ไม่มีผลกระทบต่อการเปลี่ยนตำแหน่งจุดศูนย์กลางของหุ่นยนต์

3.2. วงจรรับส่งอินฟราเรด

3.2.1 วงจรส่งอินฟราเรด

การออกแบบวงจรโดยใช้การส่งสัญญาณแบบโทนเบิรสต์ (tone burst) ซึ่งประกอบด้วยพัลส์ความถี่สูงแบบต่อเนื่องตลอดช่วงความกว้างของบิตข้อมูลที่เป็น “1” ในขณะที่บิตข้อมูลอยู่ในสภาวะต่ำ สัญญาณจะคงเดิมไม่มีการเปลี่ยนแปลงแต่อย่างใด



(ก)

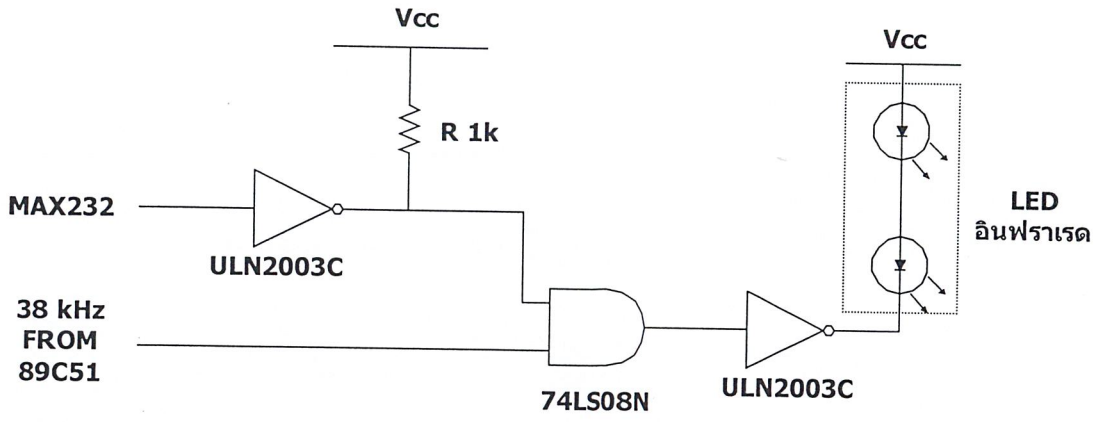
(ข)

รูปที่ 3.4 ลักษณะสัญญาณโทนเบิรสต์

(ก) สัญญาณพัลส์ข้อมูล

(ข) สัญญาณโทนเบิรสต์

วงจรตัวส่งอินฟราเรดที่ใช้มีรูปดังนี้



รูปที่ 3.5 รูปวงจรตัวส่งอินฟราเรด

รูปข้างบนจะเห็นได้ว่า สัญญาณที่ออกมาจาก IC MAX232 จะเป็นสัญญาณพัลซ์ข้อมูล เราจะนำมารวมกับสัญญาณความถี่สูงที่เกิดจากไมโครคอนโทรลเลอร์ 89C51 กำเนิดพัลซ์ความถี่ 38 kHz การรวมกันของ 2 สัญญาณนี้จะถูกรวมด้วย AND Gate จะได้สัญญาณโทนเบิรสต์ออกมา แล้วส่งต่อให้ภาคขับ LED อินฟราเรดต่อไป ตัว NOT Gate ตัวแรกมีไว้เพื่อกลับสถานะของสัญญาณที่ออกมาจาก MAX232 เพราะเนื่องจากว่าที่ภาครับอินฟราเรดมีการกลับรูปสัญญาณเกิดขึ้น จึงต้องใช้ NOT Gate เพื่อให้ค่าที่ถูกต้อง ส่วนตัวต้านทาน 1 k เป็นตัวพูลอัพ (pull-up) เพื่อให้สัญญาณมีการชดเชยกันอย่างเหมาะสม

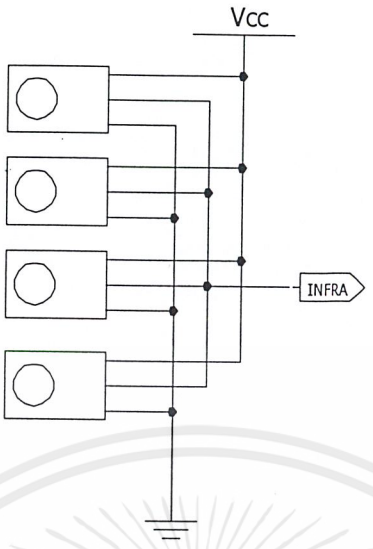
ภาคขับ LED อินฟราเรดเราใช้ NOT Gate เป็นตัวขับ โดยการต่อแบบนี้จะเสมือนกับต่อเป็นแบบอีมิตเตอร์ร่วมของทรานซิสเตอร์ ซึ่งสามารถจ่ายกระแสให้ LED อินฟราเรดได้ 0.5 A

3.2.2 วงจรรับอินฟราเรด

วงจรรับอินฟราเรดถูกออกแบบโดยใช้โมดูลอินฟราเรดเบอร์ TSOP1838 โมดูลตัวนี้จะตอบสนองเฉพาะสัญญาณความถี่ที่ 38 kHz เท่านั้น โดยจะทำการกรองความถี่พาหะ 38 kHz ออกแล้วเอาเฉพาะสัญญาณข้อมูลส่งให้ภาคขยายสัญญาณเพื่อส่งเป็นเอาท์พุตออกไป แต่เอาท์พุตที่ได้จะมีสถานะกลับกันกับสัญญาณข้อมูลจริง

เนื่องจากต้องการให้ตัวรับอินฟราเรดสามารถรับอินฟราเรดได้ทุกทิศทาง เราจึงออกแบบติดตั้งโมดูลอินฟราเรดให้หันหน้าออกจากกัน โมดูลอินฟราเรด 1 ตัวมีรัศมีการรับ 90 องศา ฉะนั้นเราจึงใช้โมดูลทั้งหมด 4 ตัวในการรับข้อมูลจากทุกทิศทาง

รูปข้างล่างเป็นรูปการต่อวงจรตัวรับอินฟราเรด

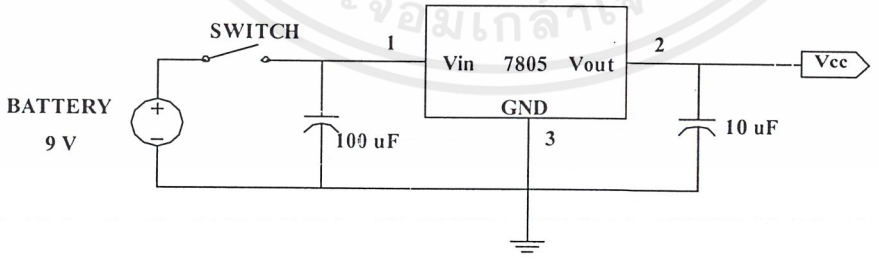


รูปที่ 3.6 รูปวงจรตัวรับอินฟราเรด

3.3. วงจรควบคุมไมโครคอนโทรลเลอร์

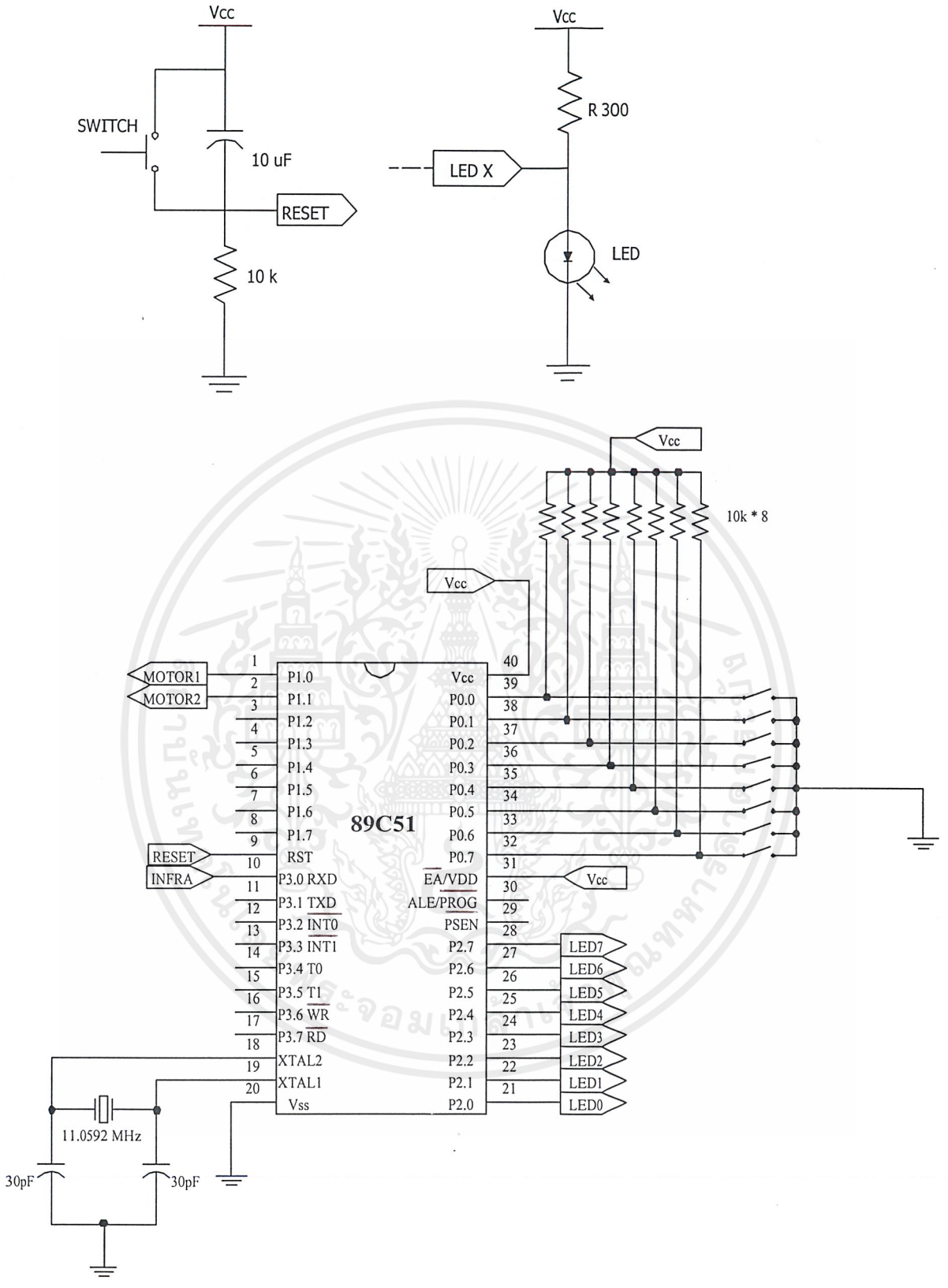
ไมโครคอนโทรลเลอร์ที่ใช้เป็นเบอร์ 89C51 ของบริษัท Atmel ซึ่งอยู่ในตระกูล MCS-51 ไมโครคอนโทรลเลอร์ตัวนี้มี ROM ภายใน4k ซึ่งมีมากพอในการโปรแกรม ที่ พอร์ต 1ขา 0 เราจะต่อกับ มอเตอร์ ตัวที่ 1 ส่วน พอร์ต 1 ขา 1 เราจะต่อเข้ากับ มอเตอร์ตัวที่ 2 โดยสองขานี้จะเป็นขาส่งพัลซ์ ไปให้ มอเตอร์ ส่วน พอร์ต 0 จะต่อกับ DIP Switch เผื่อไว้สำหรับปรับโหมดการทำงานที่รองรับในอนาคต พอร์ต 2 จะต่อกับวงจร LED มีไว้สำหรับตรวจสอบค่าที่เราส่งมาให้หุ่นผ่านพอร์ตอนุกรม

รูปวงจรต่อไปนี้แสดงวงจรจ่ายไฟของหุ่นยนต์



รูปที่ 3.7 รูปวงจรแหล่งจ่ายไฟของหุ่นยนต์

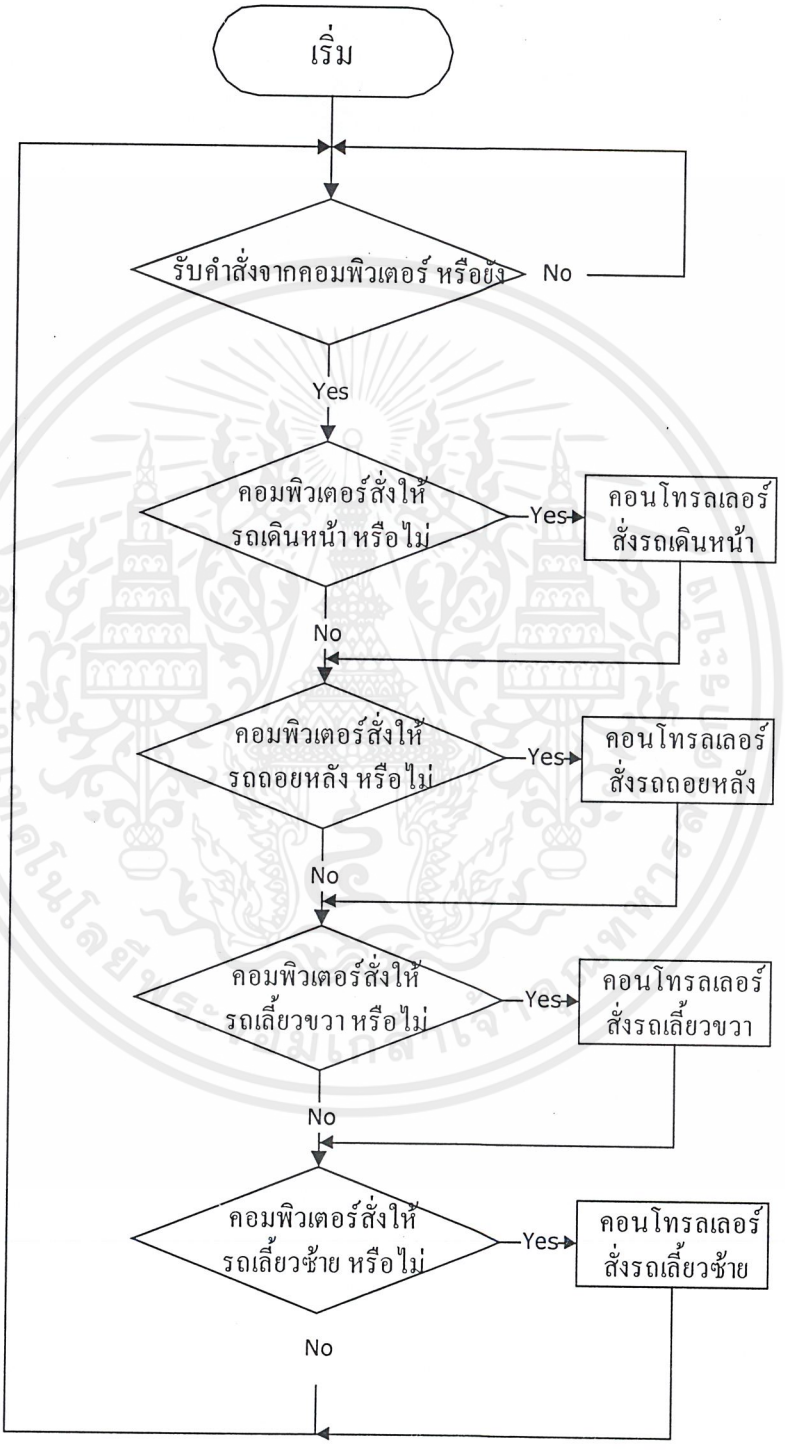
รูปที่ 3.8 แสดงวงจรควบคุมไมโครคอนโทรลเลอร์



รูปที่ 3.8 รูปวงจรควบคุมหุ่นยนต์

3.4. การควบคุมตัวรถ

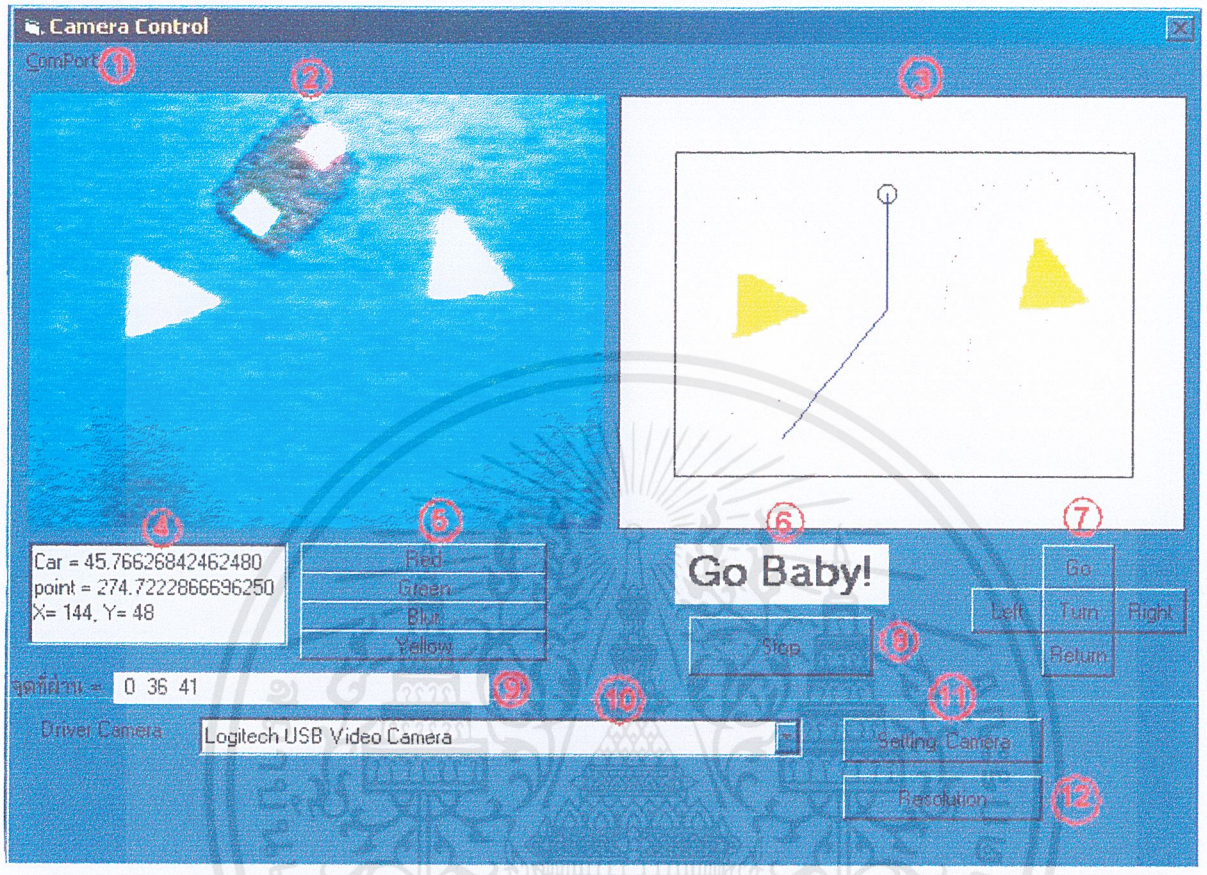
ทำการเขียนโปรแกรมควบคุมมอเตอร์ด้วยภาษาแอสเซมบลี โดยโปรแกรมนี้จะถูกเก็บไว้ในชิปไอซี 89C51 ซึ่งแสดงที่ภาคผนวก ส่วนโฟลว์ชาร์ทแสดงการทำงานของโปรแกรมจะแสดงได้ดังนี้



รูปที่ 3.9 โฟลว์ชาร์ตแสดงการทำงานของส่วนการควบคุมรถ

3.5.ส่วนการติดต่อผู้ใช้

ใช้โปรแกรม Microsoft Visual Basic สร้างโปรแกรมทั้งในส่วนติดต่อกับผู้ใช้และในส่วนการประมวลผล รูปข้างล่างเป็นรูปแบบฟอร์มที่สร้างขึ้น



รูปที่ 3.10 แสดงส่วนประกอบของโปรแกรมที่เขียนด้วย Visual Basic 6

จากรูปฟอร์มข้างบน เราสามารถแบ่งส่วนประกอบออกได้เป็นส่วนย่อยๆดังนี้

1. MnuPort : เป็นออบเจ็คชนิด Menu เป็นส่วนให้ผู้เลือกว่าจะส่งคำสั่งควบคุมหุ่นยนต์โดยผ่านพอร์ตอนุกรมพอร์ตไหน ซึ่งจะมีให้เลือกใช้ 2 พอร์ตคือ พอร์ต com1 และพอร์ต com2 ดังรูป

รูปที่ 3.11 แสดงรูป Comport

2. ezVidCap1 : เป็นออบเจ็คชนิด ezVidCap สามารถดึงภาพจากกล้องมาได้ ตามปกติใน VB6 จะไม่มีออบเจ็คนี้ ต้อง Add Components เข้ามาเอง ออบเจ็คนี้สามารถ Download มาได้จาก Website http//i.am/shrinkwrapvb
3. Picture1 : เป็นออบเจ็คชนิด Picture Box เป็นออบเจ็คที่สามารถแสดงภาพในรูปแบบของบิตแมพ (Bitmap) ,ไอคอน หรือเมตาไฟล์ (Metafile เป็นไฟล์ที่เก็บคำอธิบายว่าจะสร้างภาพในลักษณะใด ณ ตำแหน่งใด) ใช้สำหรับวาดรูปภาพจำลองเส้นทางการเดินของหุ่นยนต์
4. Text1 : เป็นออบเจ็คชนิด Text Box ใช้สำหรับแสดงรายละเอียดตำแหน่งของหุ่นยนต์ โดยจะแสดงค่าทิศทางของรถ ทิศทางของจุดที่หุ่นยนต์จะต้องไป และโคออร์ดิเนตที่หุ่นยนต์อยู่
5. ComRed,ComGreen,ComBlue,ComYellow : เป็นออบเจ็คชนิด CommandButton ใช้เป็นปุ่มในการสั่งทำงาน โดยถ้ากดปุ่ม ComRed จะเป็นการสำเนาภาพจาก ezVidCap1 มาลงใน Picture 1 โดยจะไม่นำภาพสีแดงมาด้วย ส่วน ComGreen, ComBlue, ComYellow จะมีลักษณะการทำงานคล้ายกัน ต่างกันเพียงเวลาในการทำสำเนาภาพจะไม่นำภาพสีเขียว สีน้ำเงิน สีเหลืองมาด้วย
6. Label4 : เป็นออบเจ็คชนิด Label ใช้แสดงข้อความบ่งบอกสถานะการสั่งงานหุ่นยนต์ โดยถ้าข้อความแสดงว่า Ready! จะแสดงว่าตอนนี้หุ่นยนต์อยู่กับที่ พร้อมรับคำสั่งที่จะสั่งให้หุ่นยนต์ แต่ถ้าข้อความแสดงว่า Go Baby! จะแสดงว่าตอนนี้หุ่นยนต์กำลังเคลื่อนที่อยู่ แต่ผู้ใช้ก็สามารถสั่งคำสั่งใหม่ให้หุ่นยนต์ได้
7. Command 2-Command 6 : เป็นออบเจ็คชนิด CommandButton ใช้เป็นปุ่มคำสั่งบังคับหุ่นยนต์โดยตรงโดยจะมีปุ่มเดินหน้า ถอยหลัง หันซ้าย หันขวา
8. Commamd7 : เป็นออบเจ็คชนิด CommandButton ใช้เป็นปุ่มคำสั่งให้หุ่นยนต์หยุดเดิน
9. Label 3: เป็นออบเจ็คชนิด Label ใช้แสดงข้อความบอกจุดที่หุ่นยนต์ต้องเดินผ่าน
10. CbDriver: เนื่องจากภายในคอมพิวเตอร์มี driver สำหรับกล้องวิดีโออยู่หลายชนิด ดังนั้นจึงสร้าง cbDriver ซึ่งเป็นออบเจ็คชนิด ComboBox ขึ้น เพื่อช่วยผู้ใช้โปรแกรมในการเลือกไดรเวอร์ที่เหมาะสำหรับการใช้งานนั้นๆ
11. CmdSetting: เป็นออบเจ็คชนิด CommandButton สร้างขึ้นเพื่อช่วยในการปรับแสง สี รวมถึงความชัดเจนของภาพโดยเรียกจากไดรเวอร์ Logitech USB Video Camera จากนั้นผู้ใช้งานสามารถปรับแต่งภาพได้ตามต้องการ

12. CmdSol: เป็นออบเจ็คชนิด CommandButton สร้างขึ้นเพื่อใช้ปรับความละเอียดของภาพ (Resolution) โดยมีหน่วยเป็น Pixel ซึ่งจะแสดงค่าความละเอียดของภาพตามที่เราต้องการ

3.6. ส่วนการประมวลผล

ในโปรแกรม VB6 สามารถแบ่งโปรแกรมออกได้เป็น 3 ส่วนคือ

1. ส่วนฟังก์ชัน
2. ส่วนโปรแกรมย่อย
3. ส่วนตอบสนองเหตุการณ์

ส่วนฟังก์ชัน มีดังต่อไปนี้

1. **Point2Point** :เป็นฟังก์ชั่นที่ถูกสร้างขึ้นมาเพื่อตรวจสอบความเป็นไปได้ที่ตัวหุ่นยนต์จะเดินทางจากจุดหนึ่งไปยังอีกจุดหนึ่งโดยมีกระบวนการดังนี้

รูปที่ 3.12 รูปประกอบการอธิบายการหาฟังก์ชัน Point2Point

สมมติว่าต้องการให้หุ่นยนต์เดินทางจากจุด P1 ไปยัง P2 เราจะใช้ฟังก์ชันนี้ตรวจสอบดูว่าหุ่นยนต์สามารถเดินทางไปได้หรือไม่ โดยจะแบ่งเส้นตรงระหว่างจุดสองจุดออกเป็นส่วนย่อย ๆ แล้วพิจารณา ณ จุดที่ถูกแบ่ง (คือจุด 1,2,3,4,5,6) แล้วเรียกฟังก์ชัน CarArea เพื่อตรวจดูว่าจุดที่ถูกแบ่งนั้นรถสามารถอยู่ได้หรือไม่ ถ้าทุกจุดหุ่นยนต์สามารถอยู่ได้ฟังก์ชัน Point2Point นี้จะให้ค่า True หมายความว่า หุ่นยนต์สามารถเดินไปได้ แต่ถ้าผลออกมาเป็น False หมายความว่ามีสิ่งกีดขวางอยู่ทำให้หุ่นยนต์ไม่สามารถเดินทางเป็นเส้นตรงตามทางที่ต้องการได้

2. **Object** : เป็นฟังก์ชันสำหรับตรวจสอบว่า ณ จุดนั้นเป็นสิ่งกีดขวางหรือไม่ (ในที่นี้สมมติให้จุดสีเหลืองเป็นสิ่งกีดขวาง) ถ้าจุดนั้นเป็นสิ่งกีดขวางฟังก์ชันนี้จะแสดงค่า True แต่ถ้าไม่ใช่ก็จะให้ค่า False ออกมา
3. **CarArea** : เป็นฟังก์ชั่นสำหรับตรวจสอบว่า ณ จุดที่พิจารณาหุ่นยนต์สามารถเคลื่อนที่ไปได้หรือไม่ สามารถอธิบายได้ดังรูปต่อไปนี้

รูปที่ 3.13 แสดงจุดที่ใช้ในการตรวจสอบของฟังก์ชัน CarArea

ทำการสร้างจุด 16 จุดรอบจุดที่พิจารณาแล้วตรวจดูว่าทั้ง 16 จุดนี้โดนสิ่งกีดขวางหรือไม่โดยเรียกฟังก์ชัน Object ขึ้นมาตรวจสอบ ถ้าทั้ง 16 จุดนี้ไม่โดนสิ่งกีดขวางฟังก์ชัน CarArea จะให้ค่า True ออกมา แต่ถ้ามีจุดใดจุดหนึ่งโดนสิ่งกีดขวางฟังก์ชันดังกล่าวจะให้ผลเป็น False

4. **ColorRGB** : เป็นฟังก์ชันให้ค่าแม่สีหลัก แดง เขียว น้ำเงิน โดยค่าที่ให้จะเป็นค่ารหัสสีของจุดที่พิจารณา

ส่วนโปรแกรมย่อย มีดังนี้

1. **CarGo** : จะทำการส่งคำสั่งให้หุ่นยนต์เดินหน้า
2. **CarReturn** : จะทำการส่งคำสั่งให้หุ่นยนต์หันซ้าย หันขวา
3. **CarTurnLeft** : จะทำการส่งคำสั่งให้หุ่นยนต์เลี้ยวซ้าย
4. **CarTurnRight** : จะทำการส่งคำสั่งให้หุ่นยนต์เลี้ยวขวา
5. **CarStop** : จะทำการส่งคำสั่งให้หุ่นยนต์หยุด
6. **PathA** : เป็นกระบวนการหาเส้นทางที่สั้นที่สุดเพื่อให้หุ่นยนต์ไปถึงจุดหมายได้เร็วที่สุด

ส่วนตอบสนองเหตุการณ์ มีดังนี้

1. Picture1_MouseDown

เมื่อเราทำการคลิ๊กเมาส์บน Picture1 โปรแกรมจะทำการตรวจดูก่อนว่า ณ จุดที่คลิ๊กนั้น หุ่นยนต์สามารถอยู่ได้หรือไม่ ถ้าได้ก็จะเริ่มต้นคำนวณหาจุดที่หุ่นยนต์จะต้องเดินผ่าน โดยคอมพิวเตอร์จะมีขั้นตอนการคำนวณดังนี้

- สแกนภาพจากกล้องที่ละจุดถ้าพบสีแดงจะทำการเก็บค่าตำแหน่งสีแดงไว้ ถ้าเจอสีเขียวก็จะเก็บค่าตำแหน่งสีเขียวไว้ แต่ถ้าเจอสีเหลืองก็จะทำการตรวจสอบว่า ณ จุดนั้น

เป็นจุดมุมของสิ่งกีดขวางหรือไม่ โดยพิจารณาได้จากจุดรอบข้างของจุดนั้น ๆ ถ้าจุดรอบข้างทั้ง 8 จุด มีสีเหลืองน้อยกว่า 4 จุด ก็แสดงว่าจุดพิจารณานั้นเป็นจุดมุมของสิ่งกีดขวาง

2	1	8
3	X	7
4	5	6

รูปที่ 3.14 แสดงการพิจารณาจุดรอบข้าง

ถ้าจุดพิจารณาเป็นจุดมุมของสิ่งกีดขวาง ก็จะทำการสร้างจุดสีแดงขึ้นรอบจุดนั้น 8 จุด และทำการพิจารณาทีละจุดว่าจุดนั้น ๆ หุ่นยนต์สามารถอยู่ได้หรือไม่ ถ้าอยู่ได้ก็จะคงจุดนั้นไว้ แต่ถ้าอยู่ไม่ได้ก็จะลบจุดนั้นทิ้ง สุดท้ายจะได้จุดสีแดงรอบสิ่งกีดขวาง

จุดสีแดงที่เราสร้างขึ้นนี้จะถูกเก็บค่าตำแหน่งไว้เพื่อไว้ใช้ในการคำนวนหาระยะที่สั้นที่สุดต่อไป

- หาจุดศูนย์กลางของหุ่นยนต์ โดยการหาจุดศูนย์กลางของสีแดง หาได้จาก

$$red\_center.X = (\sum Xred)/n$$

$$red\_center.Y = (\sum Yred)/n$$

โดยที่ n คือ จำนวนจุดสีแดงทั้งหมด

จุดศูนย์กลางของสีเขียวหาได้จาก

$$green\_center.X = (\sum Xgreen)/m$$

$$green\_center.Y = (\sum Ygreen)/m$$

โดยที่ m คือ จำนวนจุดสีเขียวทั้งหมด

หาจุดศูนย์กลางของหุ่นยนต์ได้จาก

$$X = (red\_center.X + green\_center.X)/2$$

$$Y = (red\_center.Y + green\_center.Y)/2$$

เมื่อหาตำแหน่งของหุ่นยนต์ได้ ก็จะทำการวาดวงกลมลงใน Picture1 เพื่อบ่งบอกตำแหน่งเริ่มต้นเดินของหุ่นยนต์ แล้วจะทำการหาเส้นทางที่สั้นที่สุดโดยเรียกใช้โปรแกรมย่อย PathA ในการคำนวณ ถ้าจุดเป้าหมายที่จะให้หุ่นยนต์เดินไปเป็นจุดอับหรือโดนสิ่งกีดขวางบังทางหมด ก็จะให้โปรแกรมแจ้งบอกผู้ใช้ว่าไม่สามารถไปได้แล้วจะรออรับคำสั่งใหม่ แต่ถ้าหาเส้นทางที่สั้นที่สุดได้ก็จะทำการวาดเส้นทางที่หุ่นยนต์จะต้องเดินโดยจะวาดบน Picture1

หลังจากนั้นก็จะเรียกใช้ Timer2 เพื่อให้เกิดการตอบสนองเหตุการณ์เป็นคาบเวลา โดยเราจะตั้งเวลาให้เกิดเหตุการณ์ Timer2_Timer ทุก ๆ 300 mS

2. Timer2_Timer

เมื่อเกิดเหตุการณ์นี้ จะเกิดขั้นตอนการทำงานดังนี้

- ทำการสแกนภาพขนาด 60*60 โดยมีจุดศูนย์กลางของภาพเป็นจุดเดียวกับจุดศูนย์กลางของหุ่นยนต์ครั้งล่าสุด สาเหตุที่สแกนภาพขนาดนี้ก็เพื่อให้โปรแกรมทำงานเร็วขึ้นการสแกนภาพนี้ถ้าเจอจุดสีแดงและสีเขียวก็จะเก็บค่าตำแหน่งไว้
- เมื่อสแกนจนครบแล้วก็จะทำการคำนวนหาจุดศูนย์กลางของสีแดงและสีเขียวแล้วคำนวนหาจุดศูนย์กลางของหุ่นยนต์
- ทำการหาทิศทางการหันของหุ่นยนต์โดยพิจารณาจากรูปต่อไปนี้

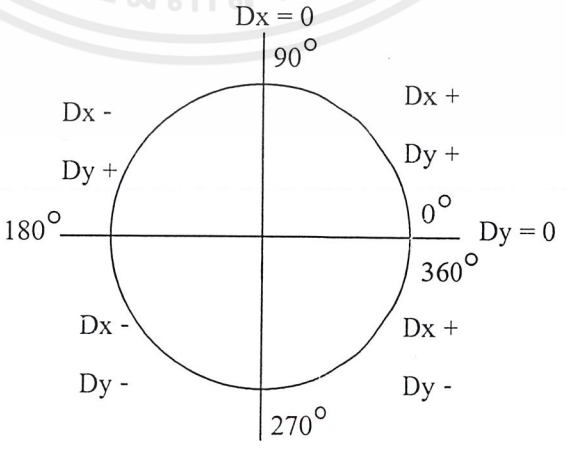
รูปที่ 3.15 แสดงการจำลองจุดที่ใช้แทนหัวและท้ายของหุ่นยนต์

สมมติให้ P1 เป็นด้านหน้าของหุ่นยนต์ และกำหนดให้

Dx = X1-X2

Dy = Y2-Y1

เมื่อพิจารณาร่วมกับวงกลม 1 หน่วย จะสามารถแบ่งเงื่อนไขในการคำนวณได้เป็น



รูปที่ 3.16 แสดงเครื่องหมายของ Dx และ Dy ที่มุมต่างๆกัน

— ถ้า Dx มีเครื่องหมายเป็น - ทิศทางของหุ่นยนต์สามารถหาได้จาก

DeCar = 180 + 180/π*arctan(Dy/Dx) องศา

— ถ้า Dx = 0 แสดงว่าหุ่นยนต์วางตัวอยู่ตามแนวแกน Y กรณีนี้ arctan(Dy/Dx) จะหาค่าไม่ได้ จึงต้องใช้วิธีพิจารณาเครื่องหมายของ Dy แทน ซึ่งจะได้ว่า

ถ้า Dy มีเครื่องหมายเป็น + จะได้ทิศทางของหุ่นยนต์เท่ากับ 90 องศา

ถ้า Dy มีเครื่องหมายเป็น - จะได้ทิศทางของหุ่นยนต์เท่ากับ 270 องศา

— ถ้า Dx มีเครื่องหมายเป็น + การหาทิศทางของหุ่นยนต์แบ่งได้เป็น 2 กรณีคือ

ถ้า Dy มีเครื่องหมายเป็น – จะสามารถคำนวณหาทิศทางได้คือ

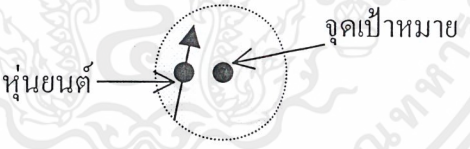
DeCar = 360 + 180/π*arctan(Dy/Dx) องศา

ถ้า Dy มีเครื่องหมายเป็น + หรือมีค่าเป็น 0 จะสามารถคำนวณหาทิศทางได้คือ

DeCar = 180/π*arctan(Dy/Dx) องศา

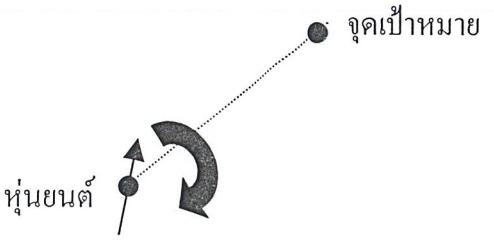
- ทำการหาทิศทางของจุดที่เราจะต้องไปโดยใช้จุดศูนย์กลางของหุ่นยนต์กับจุดเป้าหมายนำมาคำนวณเหมือนกับการหาทิศทางการหันของหุ่นยนต์
- เลือกคำสั่งที่จะสั่งให้กับหุ่นยนต์โดยพิจารณาจาก

1. ถ้าจุดศูนย์กลางของรถอยู่ใกล้จุดเป้าหมายในรัศมีที่กำหนด ก็จะสั่งให้หุ่นยนต์หยุดเดินแล้วพร้อมที่จะเดินไปยังจุดเป้าหมายอันต่อไป



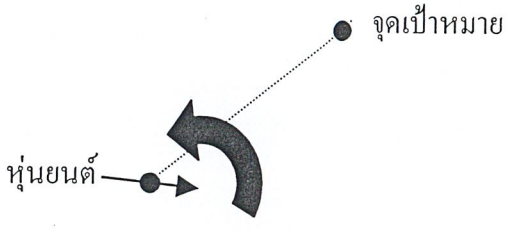
รูปที่ 3.17 แสดงตำแหน่งเมื่อหุ่นยนต์หยุดเดิน

2. ถ้ามุมของหุ่นยนต์มากกว่ามุมของจุดเป้าหมายก็จะสั่งให้หุ่นยนต์หันขวา



รูปที่ 3.18 แสดงการหันขวาของหุ่นยนต์

3. ถ้ามุมของหุ่นยนต์น้อยกว่ามุมของจุดเป้าหมายก็จะสั่งให้หุ่นยนต์หันซ้าย

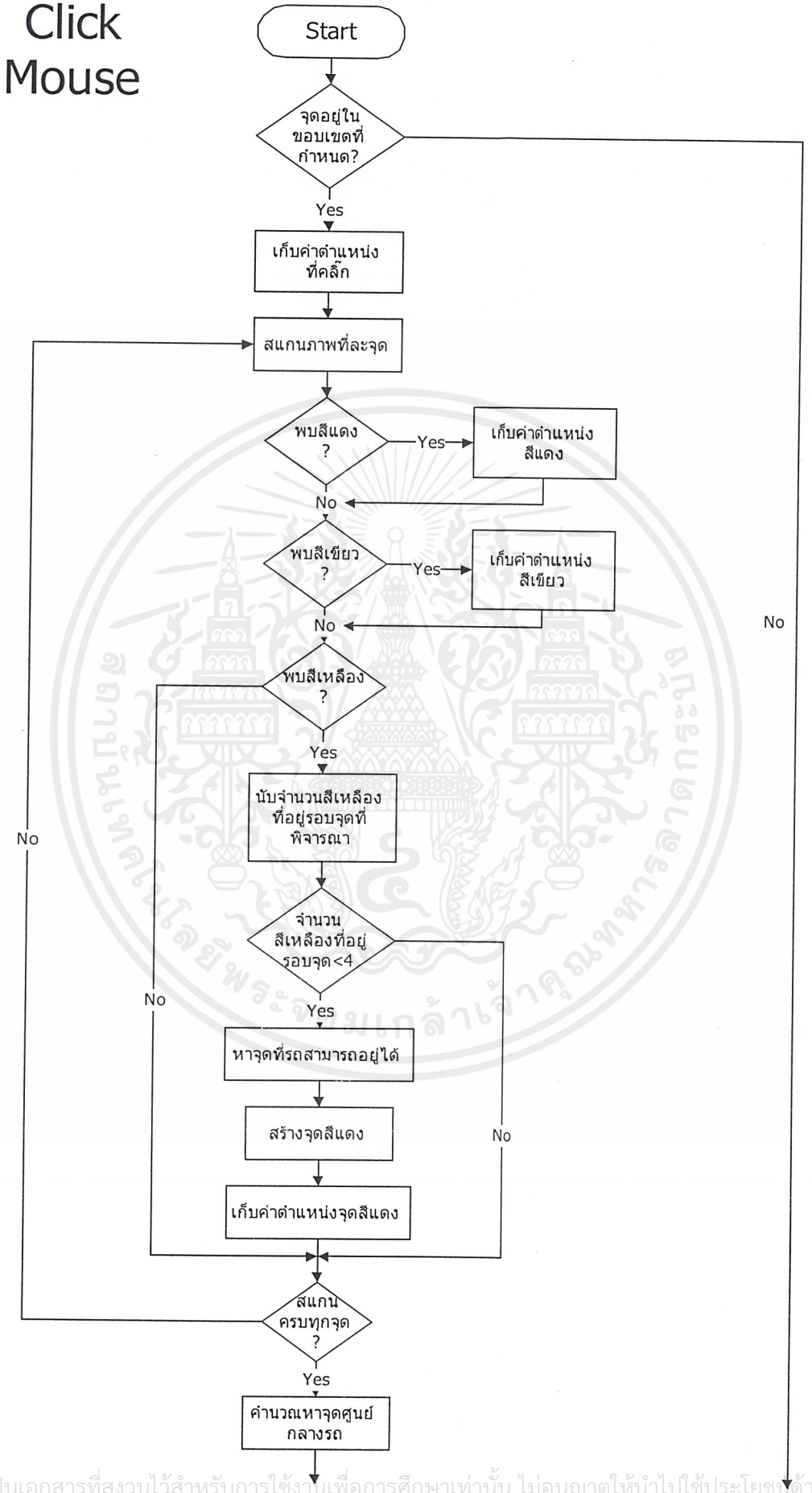


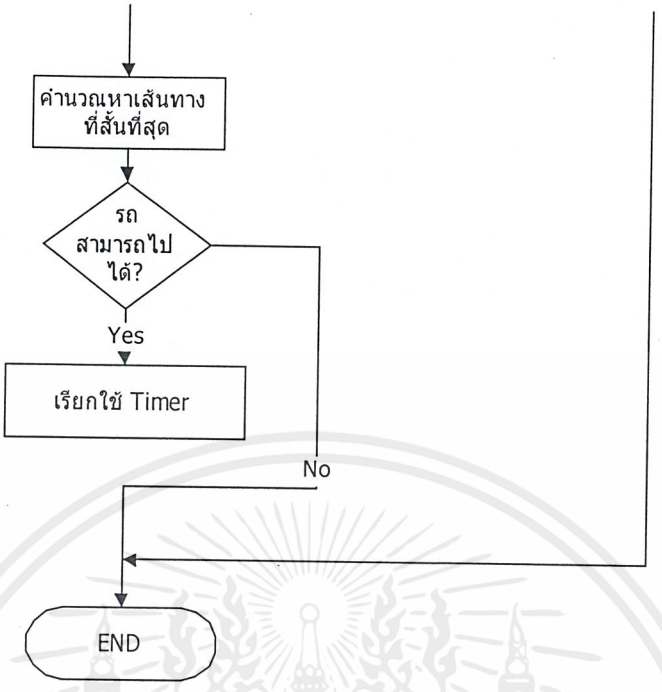
รูปที่ 3.19 แสดงการหันซ้ายของหุ่นยนต์

4. ถ้าหุ่นยนต์อยู่ที่จุดเป้าหมายอันสุดท้ายแล้ว (จุดคลิ๊กเมาส์) ก็สั่งให้หุ่นยนต์หยุดเดิน (หยุดการตอบสนองต่อเหตุการณ์ Timer2_Timer) แล้วรอรับคำสั่งใหม่อีกที

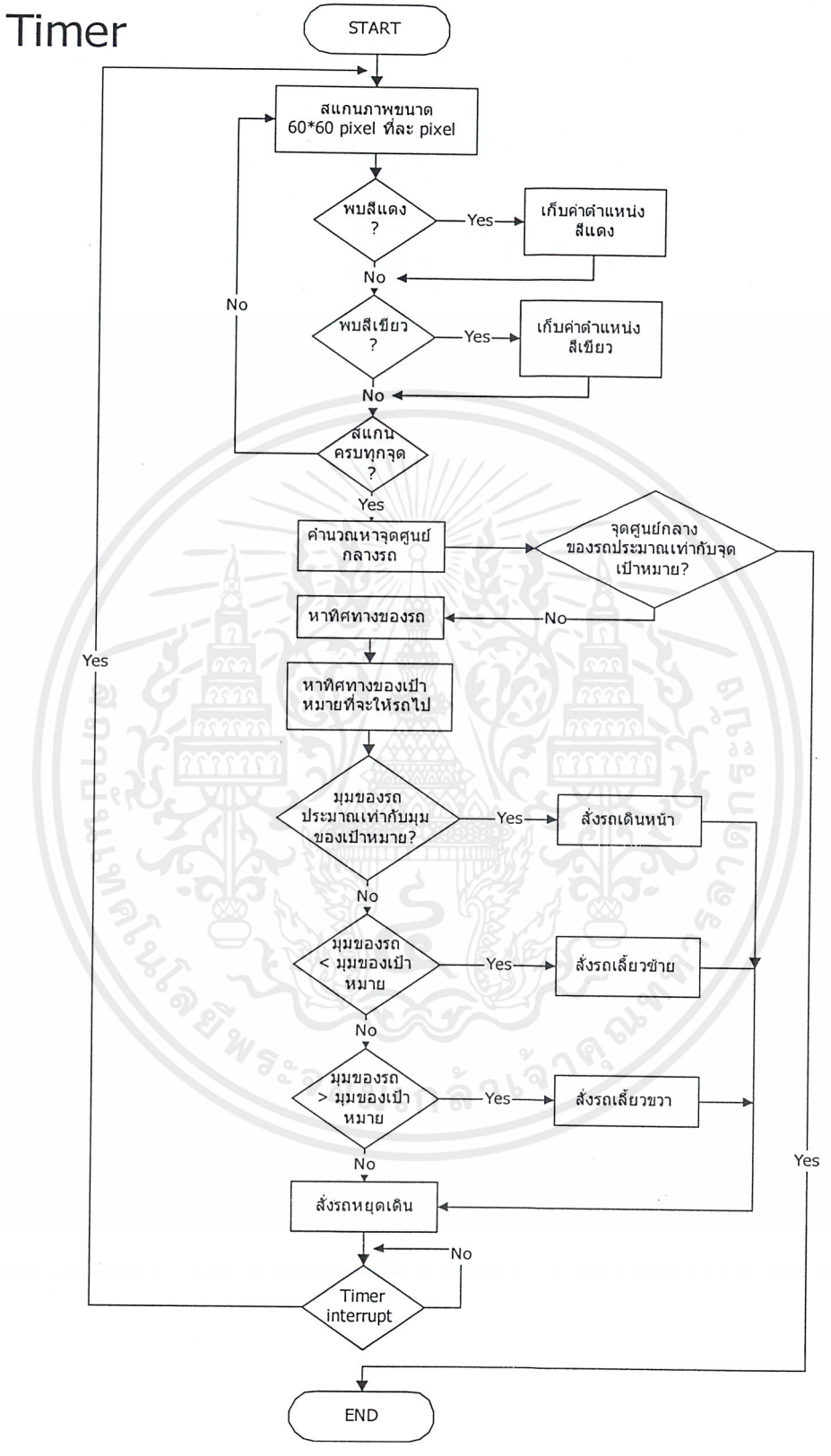
โฟลว์ชาร์ตต่อไปนี้แสดงการทำงานของส่วนประมวลผล

Click Mouse





รูปที่ 3.20 โฟล์วชาร์ต ของเหตุการณ์ Picture1_MouseDown



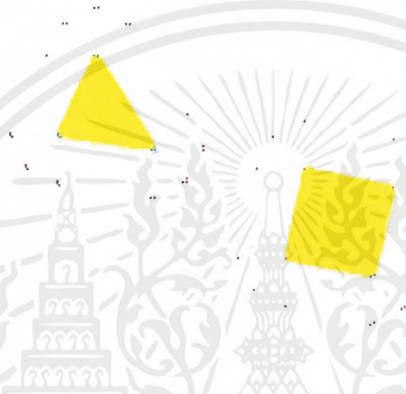
รูปที่ 3.21 โฟล์วชาร์ต ของเหตุการณ์ Timer2_Timer

บทที่ 4

ผลการทดลอง

4.1 การทดลองการคำนวณเส้นทาง

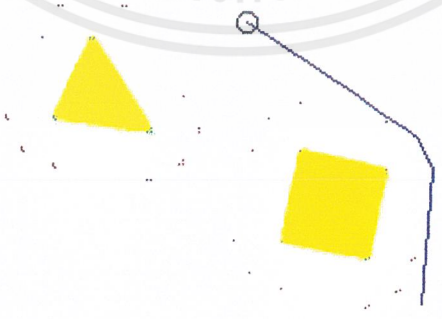
ทดลองการคำนวณเส้นทางของพื้นที่ที่มีสิ่งกีดขวางวางอยู่ 2 อันดังรูปข้างล่าง



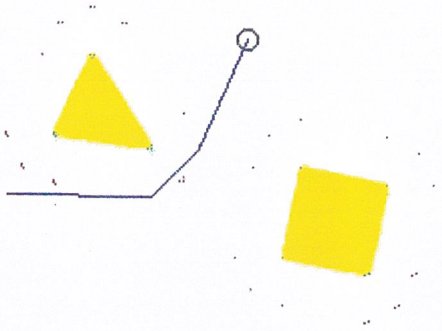
รูปที่ 4.1 แสดงรูปสิ่งกีดขวาง

จากรูปข้างบนจะเห็นว่าตรงมุมของสิ่งกีดขวางจะมีจุดสีแดงอยู่โดยรอบ จุดนี้เราจะเอาไว้ใช้ในการคำนวณหาเส้นทางที่สั้นที่สุด

ต่อมาทำการคลิ๊กเมาส์สั่งให้โปรแกรมหาเส้นทางที่สั้นที่สุด รูปข้างล่างจะเป็นรูปการคลิ๊กครั้งที่ 1 และ 2 ตามลำดับ



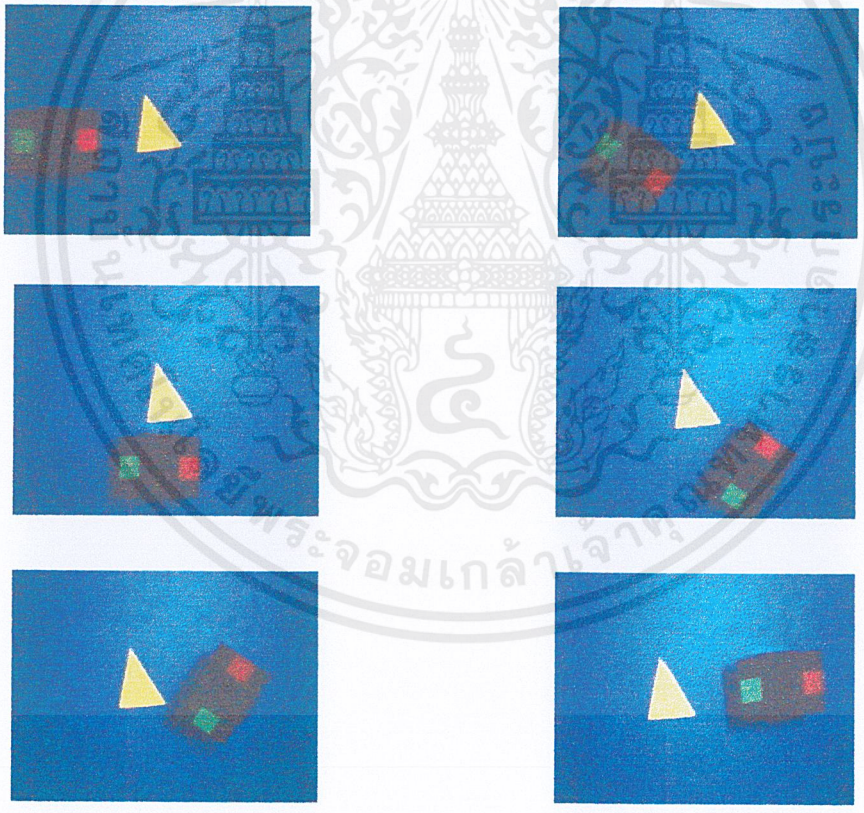
รูปที่ 4.2 แสดงการทดลองคลิ๊กเมาส์ครั้งที่ 1



รูปที่ 4.3 แสดงการทดลองคลิ๊กเมาส์ครั้งที่ 2

4.2 การทดลองการทำงานของหุ่นยนต์

การทดลองการเดินของหุ่นยนต์ใช้วิธีการดูเส้นทางเดินของหุ่นยนต์ว่าสามารถหลบสิ่งกีดขวางได้หรือไม่



รูปที่ 4.4 แสดงการทดลองการหลบสิ่งกีดขวางของหุ่นยนต์

บทที่ 5
บทสรุปและวิจารณ์

5.1 สรุปผลการดำเนินงาน

โครงงานนี้เกี่ยวกับการสร้างหุ่นยนต์ที่ควบคุมด้วยคอมพิวเตอร์โดยใช้รีโมตคอนโทรลแบบอินฟราเรดเป็นตัวส่งข้อมูลระหว่างหุ่นยนต์กับคอมพิวเตอร์ การควบคุมหุ่นยนต์จะควบคุมผ่านคอมพิวเตอร์โดยจะมีกล้องวิดีโอจับการเคลื่อนไหว ตำแหน่งของหุ่นยนต์และสิ่งกีดขวางเพื่อนำไปใช้ในการออกคำสั่งให้กับหุ่นยนต์

จากการดำเนินงาน หุ่นยนต์สามารถเดินทางตามเส้นทางที่กำหนดโดยมีความสามารถพื้นฐานคือเลี้ยวซ้าย เลี้ยวขวา หันหน้า หันหลัง พร้อมทั้งมีความสามารถสำคัญคือหลบหลีกสิ่งกีดขวางได้ตามที่ต้องการ และสามารถหาเส้นทางที่สั้นที่สุดในการที่จะไปยังจุดหมายได้ด้วย

5.2 ปัญหาที่เกิดและแนวทางแก้ไข

ปัญหา : การคำนวณหาเส้นทางการเดินอาจจะไม่ใช่เส้นทางที่สั้นที่สุด

แนวทางแก้ไข : เขียนโปรแกรมที่มีการคำนวณหาเส้นทางที่ดีกว่านี้

ปัญหา : ใช้เวลานานในการออกคำสั่งให้รถเดิน

แนวทางแก้ไข : เขียนโปรแกรมให้มีการทำงานสั้นขึ้นหรือไม่ก็เปลี่ยนภาษาโปรแกรมที่ใช้เขียน เช่น อาจจะใช้โปรแกรม Visual C++ ซึ่งจะทำงานเร็วขึ้นได้

ปัญหา : ภาพที่ได้จากกล้องวิดีโอไม่คมชัดเท่าที่ควร ทำให้การคำนวณหาเส้นทางมีการผิดพลาดเกิดขึ้น อีกทั้งภาพที่ส่งมามีการกระตุกพอสมควร

แนวทางแก้ไข : ใช้กล้องวิดีโอที่ให้ภาพคมชัดกว่ากล้อง QuickCam ที่ใช้ในการทดลองในโครงงานนี้

5.3 แนวทางพัฒนาในอนาคต

1. นำระบบการตรวจจับต่างๆมาติดตั้งบนหุ่นยนต์ เพื่อที่จะได้เพิ่มความสามารถของหุ่นยนต์ เช่น ตัวตรวจจับกลิ่น ตัวตรวจจับความร้อน
2. ใช้หน่วยประมวลผล (CPU) ที่มีความสามารถสูงติดตั้งบนหุ่นยนต์ เพื่อที่หุ่นยนต์จะสามารถเคลื่อนที่ไปได้เองโดยไม่ต้องมีคนคอยควบคุมอยู่บนหน้าจอคอมพิวเตอร์

3. ทำระบบการสื่อสารด้วยคลื่นวิทยุ เพื่อลดข้อจำกัดการส่งข้อมูลให้เป็นเส้นตรง และเพื่อให้สามารถส่งผ่านสิ่งกีดขวางได้

ภาคผนวก

Code ของโปรแกรม Visual Basic 6

```
**********Code in Forms**********
Option Explicit
Private Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long
Private Declare Function ReleaseDC Lib "user32" (ByVal hwnd As Long, ByVal Hdc As Long) As Long
Dim CameraH As Long
Dim PicX%, PicY%, i%, AA As Byte, target As Byte
Dim PicM As Boolean
Dim P() As Integer
Dim XX() As Integer
Private Type Point
    X As Integer
    Y As Integer
End Type
Dim Car As Point
Dim clientP As POINTAPI
Dim AAA(0 To 100) As Point
Dim lpPoint As POINTAPI
Private Const Rr = 40, Rrr = 42, pi = 22 / 7
---------------------------------------------------------------------------------------------------
Private Sub CmdSetting_Click()
  ezVidCap1.ShowDlgVideoSource
End Sub
---------------------------------------------------------------------------------------------------
Private Sub CmdSol_Click()
  ezVidCap1.ShowDlgVideoFormat
End Sub
---------------------------------------------------------------------------------------------------
Private Sub Command2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Call CarTurnLeft
End Sub
---------------------------------------------------------------------------------------------------
Private Sub Command2_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Call CarStop
```

```
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub Command3_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Call CarTurnRight
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub Command3_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Call CarStop
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub Command4_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Call CarReturn
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub Command4_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Call CarStop
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub Command6_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Call CarGo
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub Command6_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Call CarStop
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub Command7_Click()
   Timer2.Interval = 0
   Label4.Caption = "Ready!"
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub ComRed_Click()
Dim Gx, Gy, Drg%, Drb%, Dgb%, R%, G%, B As Integer
Dim Gcolor As Long
```

```
    For Gx = 0 To 320
        For Gy = 0 To 240
          Gcolor = GetPixel(CameraH, Gx, Gy)
          Call ColorRGB(Gcolor, R, G, B)
          Drg = R - G
          Drb = R - B
          Dgb = G - B
          If Drg > 30 And Drb > 30 Then Gcolor = 16777215
          SetPixel Picture1.Hdc, Gx, Gy, Gcolor
        Next Gy
    Next Gx
End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub ComGreen_Click()
Dim Gx, Gy, Drg%, Drb%, Dgb%, R%, G%, B As Integer
Dim Gcolor As Long
    For Gx = 0 To 320
        For Gy = 0 To 240
          Gcolor = GetPixel(CameraH, Gx, Gy)
          Call ColorRGB(Gcolor, R, G, B)
          Drg = R - G
          Drb = R - B
          Dgb = G - B
          If Drg < -30 And Dgb > 30 Then Gcolor = 16777215
          SetPixel Picture1.Hdc, Gx, Gy, Gcolor
        Next Gy
    Next Gx
End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub ComBlur_Click()
Dim Gx, Gy, Drg%, Drb%, Dgb%, R%, G%, B As Integer
Dim Gcolor As Long
    For Gx = 0 To 320
        For Gy = 0 To 240
```

```
            Gcolor = GetPixel(CameraH, Gx, Gy)
            Call ColorRGB(Gcolor, R, G, B)
            Drg = R - G
            Drb = R - B
            Dgb = G - B
            If Dgb < -30 And Drb < -30 Then Gcolor = 16777215
            SetPixel Picture1.Hdc, Gx, Gy, Gcolor
        Next Gy
    Next Gx
End Sub
```

```
Private Sub ComYellow_Click()
Dim Gx, Gy, Drg%, Drb%, Dgb%, R%, G%, B As Integer
Dim Gcolor As Long
    For Gx = 0 To 320
        For Gy = 0 To 240
            Gcolor = GetPixel(CameraH, Gx, Gy)
            Call ColorRGB(Gcolor, R, G, B)
            If Abs(R - G) < 30 And R > 220 Then
                Gcolor = 16777215
            End If
            SetPixel Picture1.Hdc, Gx, Gy, Gcolor
        Next Gy
    Next Gx
End Sub
```

```
Private Sub Form_Load()
Dim i As Integer
  CameraH = GetDC(ezVidCap1.hwnd)
  If 0 < ezVidCap1.NumCapDevs Then
    For i = 0 To ezVidCap1.NumCapDevs - 1
      cbDriver.AddItem (ezVidCap1.GetDriverName(i))
    Next i
    cbDriver.ListIndex = ezVidCap1.DriverIndex
```

```
    Else
        cbDriver.AddItem ("<none>")
        cbDriver.ListIndex = 0
        MsgBox "No Video Capture Device!", vbInformation, App.Title
    End If
    If MSComm1.PortOpen = False Then MSComm1.PortOpen = True
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub Form_Unload(Cancel As Integer)
    ReleaseDC ezVidCap1.hwnd, CameraH
    MSComm1.PortOpen = False
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub cbDriver_Click()
    Dim oldDriver As Long
    oldDriver = ezVidCap1.DriverIndex
    On Error Resume Next
    ezVidCap1.DriverIndex = cbDriver.ListIndex
    If Err Then
        ezVidCap1.DriverIndex = oldDriver
        cbDriver.ListIndex = oldDriver
    End If
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub MnuCom1_Click()
    MSComm1.PortOpen = False
    MSComm1.CommPort = 1
    MnuCom2.Checked = False
    MnuCom1.Checked = True
    MSComm1.PortOpen = True
End Sub
---------------------------------------------------------------------------------------------------------------
Private Sub MnuCom2_Click()
    MSComm1.PortOpen = False
```

```
    MSComm1.CommPort = 2
    MnuCom2.Checked = True
    MnuCom1.Checked = False
    MSComm1.PortOpen = True
End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub CarGo()
    MSComm1.Output = Chr(&HFD)
End Sub
Private Sub CarReturn()
    MSComm1.Output = Chr(&HF7)
End Sub
Private Sub CarTurnLeft()
    MSComm1.Output = Chr(&HFE)
End Sub
Private Sub CarTurnRight()
    MSComm1.Output = Chr(&HFB)
End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub CarStop()
    MSComm1.Output = Chr(&HFF)
End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, XXX As Single, YYY As Single)
    Dim X%, Y%, Carry%, Jr%
    Dim A%, B%, C%, h%, k%
    Dim ColorR%, ColorG%, ColorB%, Drg%, Drb%, Dgb%, red_x&, red_y&, green_x&, green_y&
    Dim Gcolor As Long
    Dim D As Single
    Dim red_center As Point, green_center As Point
        Timer2.Interval = 0
        en = 0
        target = 1
        h = 0
```

```
k = 0
green_x = 0
green_y = 0
Picture1.Cls
If CarArea(XXX, YYY, Rr) = False Then
   MsgBox "กรุณา click ใหม่", vbExclamation + vbOKOnly, "Error!"
   Exit Sub
End If
If Abs(AAA(0).X - XXX) <= 20 And Abs(AAA(0).Y - YYY) <= 20 Then
   Picture1.Line (AAA(0).X, AAA(0).Y)-(XXX, YYY), vbBlue
   Exit Sub
End If
D = 3.14 / 8
A = Fix(Rrr * Cos(D))
B = Fix(Rrr * Cos(2 * D))
C = Fix(Rrr * Sin(D))
Carry = 0
i = 1
For Y = 0 To 240
  For X = 0 To 320
     Gcolor = GetPixel(CameraH, X, Y)
     Call ColorRGB(Gcolor, ColorR, ColorG, ColorB)
     Drg = ColorR - ColorG
     Drb = ColorR - ColorB
     Dgb = ColorG - ColorB
     If Drg > 30 And Drb > 30 Then
        red_x = red_x + X
        red_y = red_y + Y
        h = h + 1
     End If
     If Drg < -30 And Dgb > 30 Then
        green_x = green_x + X
        green_y = green_y + Y
        k = k + 1
```

```
End If
If Object(X, Y) = True Then
   SetPixel Picture1.Hdc, X, Y, vbYellow
   If Object(X, Y - 1) = True Then Carry = Carry + 1
   If Object(X - 1, Y - 1) = True Then Carry = Carry + 1
   If Object(X - 1, Y) = True Then Carry = Carry + 1
   If Object(X - 1, Y + 1) = True Then Carry = Carry + 1
   If Object(X, Y + 1) = True Then Carry = Carry + 1
   If Object(X + 1, Y + 1) = True Then Carry = Carry + 1
   If Object(X + 1, Y) = True Then Carry = Carry + 1
   If Object(X + 1, Y - 1) = True Then Carry = Carry + 1
   If Carry < 4 Then
      SetPixel Picture1.Hdc, X, Y, vbGreen
         If CarArea(X, Y - Rrr, Rr) = True Then
            AAA(i).X = X
            AAA(i).Y = Y - Rrr
            SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
            i = i + 1
         End If
         If CarArea(X + B, Y - B, Rr) = True Then
            AAA(i).X = X + B
            AAA(i).Y = Y - B
            SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
            i = i + 1
         End If
         If CarArea(X + Rrr, Y, Rr) = True Then
            AAA(i).X = X + Rrr
            AAA(i).Y = Y
            SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
            i = i + 1
         End If
         If CarArea(X + B, Y + B, Rr) = True Then
            AAA(i).X = X + B
            AAA(i).Y = Y + B
```

```
                SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
                i = i + 1
            End If
            If CarArea(X, Y + Rrr, Rr) = True Then
                AAA(i).X = X
                AAA(i).Y = Y + Rrr
                SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
                i = i + 1
            End If
            If CarArea(X - B, Y + B, Rr) = True Then
                AAA(i).X = X - B
                AAA(i).Y = Y + B
                SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
                i = i + 1
            End If
            If CarArea(X - Rrr, Y, Rr) = True Then
                AAA(i).X = X - Rrr
                AAA(i).Y = Y
                SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
                i = i + 1
            End If
            If CarArea(X - B, Y - B, Rr) = True Then
                AAA(i).X = X - B
                AAA(i).Y = Y - B
                SetPixel Picture1.Hdc, AAA(i).X, AAA(i).Y, vbRed
                i = i + 1
            End If
        End If
        Carry = 0
    End If
  Next X
Next Y
Text1.Text = i - 1
AAA(i).X = XXX
```

```
AAA(i).Y = YYY
If h = 0 Or k = 0 Then Exit Sub
red_center.X = Fix(red_x / h)
red_center.Y = Fix(red_y / h)
green_center.X = Fix(green_x / k)
green_center.Y = Fix(green_y / k)
Car.X = Fix((red_center.X + green_center.X) / 2)
Car.Y = Fix((green_center.Y + red_center.Y) / 2)
Picture1.Circle (Car.X, Car.Y), 5
AAA(0).X = Car.X
AAA(0).Y = Car.Y
ReDim P(i, i)
For Y = 0 To i
  For X = 0 To i
    If X = Y Then
      P(X, Y) = M
    Else
      If Abs(AAA(X).X - AAA(Y).X) < 5 And Abs(AAA(X).Y - AAA(Y).Y) < 5 Then
        P(X, Y) = M
      Else
        If Point2Point(X, Y) Then
          P(X, Y) = Fix(((AAA(X).X - AAA(Y).X) ^ 2 + (AAA(X).Y - AAA(Y).Y) ^ 2) ^ (1 / 2))
        Else
          P(X, Y) = M
        End If
      End If
    End If
  Next X
Next Y
PathA i, P(), XX(), AA
If en = 1 Then
  Exit Sub
End If
For X = 1 To AA
```

```
        Picture1.Line (AAA(XX(X - 1)).X, AAA(XX(X - 1)).Y)-(AAA(XX(X)).X, AAA(XX(X)).Y), vbBlue
    Next X
    Label4.Caption = "Go Baby!"
    Timer2.Interval = 300
End Sub
---------------------------------------------------------------------------------------------------------
Private Function Point2Point(ByVal P1%, ByVal P2%) As Boolean
  Dim Cx%, Cy%, Dx%, Dy%, Ex%, Ey%, j%
  Point2Point = True
  Cy = AAA(P2).Y
  Cx = AAA(P2).X
  Dx = AAA(P1).X - AAA(P2).X
  Dy = AAA(P1).Y - AAA(P2).Y
  If Abs(Dx) > Abs(Dy) Then
    If Dx > 0 Then
      For j = 0 To Dx Step 20
        Ey = (Dy / Dx) * j + Cy
        Ex = j + Cx
        If CarArea(Ex, Ey, Rr) = False Then
          Point2Point = False
          Exit Function
        End If
      Next j
    Else
      For j = Dx To 0 Step 20
        Ey = (Dy / Dx) * j + Cy
        Ex = j + Cx
        If CarArea(Ex, Ey, Rr) = False Then
          Point2Point = False
          Exit Function
        End If
      Next j
    End If
  Else
```

```
    If Dy > 0 Then
      For j = 0 To Dy Step 20
        Ex = (Dx / Dy) * j + Cx
        Ey = j + Cy
        If CarArea(Ex, Ey, Rr) = False Then
          Point2Point = False
          Exit Function
        End If
      Next j
    Else
      For j = Dy To 0 Step 20
        Ex = (Dx / Dy) * j + Cx
        Ey = j + Cy
        If CarArea(Ex, Ey, Rr) = False Then
          Point2Point = False
          Exit Function
        End If
      Next j
    End If
  End If
End Function
```

```
Private Function Object(ByVal X%, ByVal Y%) As Boolean
Dim Gcolor As Long
Dim ColorR%, ColorG%, ColorB%, Drg%, Drb%, Dgb%
  Object = False
  Gcolor = GetPixel(CameraH, X, Y)
  Call ColorRGB(Gcolor, ColorR, ColorG, ColorB)
  Drg = ColorR - ColorG
  Drb = ColorR - ColorB
  Dgb = ColorG - ColorB
  If Abs(ColorR - ColorG) < 30 And ColorR > 220 Then Object = True
End Function
```

```
Private Function CarArea(ByVal X%, ByVal Y%, ByVal Rc%) As Boolean
    Dim A%, B%, C%
    Dim D As Single
    D = 3.14 / 8
    A = Fix(Rc * Cos(D))
    B = Fix(Rc * Cos(2 * D))
    C = Fix(Rc * Sin(D))
    CarArea = True
    If X - Rc < 0 Or X + Rc > 320 Or Y - Rc < 0 Or Y + Rc > 240 Then
       CarArea = False
    ElseIf Object(X, Y - Rc) = True Then
       CarArea = False
    ElseIf Object(X + C, Y - A) = True Then
       CarArea = False
    ElseIf Object(X + B, Y - B) = True Then
       CarArea = False
    ElseIf Object(X + A, Y - C) = True Then
       CarArea = False
    ElseIf Object(X + Rc, Y) = True Then
       CarArea = False
    ElseIf Object(X + A, Y + C) = True Then
       CarArea = False
    ElseIf Object(X + B, Y + B) = True Then
       CarArea = False
    ElseIf Object(X + C, Y + A) = True Then
       CarArea = False
    ElseIf Object(X, Y + Rc) = True Then
       CarArea = False
    ElseIf Object(X - C, Y + A) = True Then
       CarArea = False
    ElseIf Object(X - B, Y + B) = True Then
       CarArea = False
    ElseIf Object(X - A, Y + C) = True Then
       CarArea = False
```

```
    ElseIf Object(X - Rc, Y) = True Then
       CarArea = False
    ElseIf Object(X - A, Y - C) = True Then
       CarArea = False
    ElseIf Object(X - B, Y - B) = True Then
       CarArea = False
    ElseIf Object(X - C, Y - A) = True Then
       CarArea = False
    End If
End Function
-----------------------------------------------------------------------------------------------------------
Private Function ColorRGB(Gcolor As Long, R As Integer, G As Integer, B As Integer)
   R = Gcolor Mod 256
   B = Gcolor \ 65536
   G = (Gcolor \ 256) Mod 256
End Function
-----------------------------------------------------------------------------------------------------------
Private Sub Timer1_Timer()
   Timer1.Interval = 0
   Call CarStop
End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub Timer2_Timer()
Dim Drg%, Drb%, Dgb%, R%, G%, B%, Dx%, Dy%, X%, Y%, red_x&, red_y&, green_x&
Dim green_y&, h%, k%
Dim Gcolor As Long
Dim red_center As Point, green_center As Point
Dim DeCar As Double, DePoint As Double
red_x = 0
   red_y = 0
   h = 0
   k = 0
   green_x = 0
   green_y = 0
```

```
    For Y = Car.Y - 30 To Car.Y + 30
      For X = Car.X - 30 To Car.X + 30
        If X < 0 Or X > 320 Or Y < 0 Or Y > 240 Then
          Gcolor = vbBlack
        Else
          Gcolor = GetPixel(CameraH, X, Y)
        End If
        Call ColorRGB(Gcolor, R, G, B)
        Drg = R - G
        Drb = R - B
        Dgb = G - B
        If Drg > 30 And Drb > 30 Then
          red_x = red_x + X
          red_y = red_y + Y
          h = h + 1
        End If
        If Drg < -30 And Dgb > 30 Then
          green_x = green_x + X
          green_y = green_y + Y
          k = k + 1
        End If
      Next X
    Next Y
If h <> 0 And k <> 0 And red_x <> 0 And red_y <> 0 And green_x <> 0 And green_y <> 0 Then
    red_center.X = Fix(red_x / h)
    red_center.Y = Fix(red_y / h)
    green_center.X = Fix(green_x / k)
    green_center.Y = Fix(green_y / k)
End If
    Car.X = Fix((red_center.X + green_center.X) / 2)
    Car.Y = Fix((green_center.Y + red_center.Y) / 2)

    ******Find Car's angle******
    Dx = red_center.X - green_center.X
```

```
Dy = green_center.Y - red_center.Y
Select Case Sgn(Dx)
   Case -1
      DeCar = 180 + Fix(180 / pi * Atn(Dy / Dx))
   Case 0
      If Sgn(Dy) = 1 Then
         DeCar = 90
      Else: DeCar = 270
      End If
   Case 1
      If Sgn(Dy) = -1 Then
         DeCar = 360 + Fix(180 / pi * Atn(Dy / Dx))
      Else
         DeCar = Fix(180 / pi * Atn(Dy / Dx))
      End If
End Select

******Find Point's angle******
Dx = AAA(XX(target)).X - Car.X
Dy = Car.Y - AAA(XX(target)).Y
Select Case Sgn(Dx)
   Case -1
      DePoint = 180 + Fix(180 / pi * Atn(Dy / Dx))
   Case 0
      If Sgn(Dy) = 1 Then
         DePoint = 90
      Else: DePoint = 270
      End If
   Case 1
      If Sgn(Dy) = -1 Then
         DePoint = 360 + Fix(180 / pi * Atn(Dy / Dx))
      Else
         DePoint = Fix(180 / pi * Atn(Dy / Dx))
      End If
```

```
    End Select

    *******Select Command*******
    If Abs(Car.X - AAA(XX(target)).X) < 10 And Abs(Car.Y - AAA(XX(target)).Y) < 10 Then
       target = target + 1
    ElseIf DeCar - DePoint > 15 Then
       Call CarTurnRight
    ElseIf DeCar - DePoint < -15 Then
       Call CarTurnLeft
    Else
       Call CarGo
    End If
    If target > AA Then
       Timer2.Interval = 0
       Label4.Caption = "Ready!"
    End If
    Timer1.Interval = 1
    Text1.Text = "Car = " & DeCar & vbTab & "point = " & DePoint _
    & vbTab & " X= " & Car.X & ", Y= " & Car.Y
End Sub
```

```
*************Code in Module***************
Option Explicit
Public Declare Function GetPixel Lib "gdi32" (ByVal Hdc As Long, ByVal X As Long, ByVal Y As Long) As Long
Public Declare Function SetPixel Lib "gdi32" (ByVal Hdc As Long, ByVal X As Long, ByVal Y As Long,
                ByVal crColor As Long) As Long
Public Const M = 9999
Public en As Byte
Public Declare Function ClientToScreen Lib "user32" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long
Public Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal X As Long, ByVal Y As Long,
        ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long,
        ByVal ySrc As Long, ByVal dwRop As Long) As Long
Public Type POINTAPI
```

```
    X As Long
    Y As Long
End Type
Public Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long

Public Sub PathA(N As Integer, P() As Integer, ByRef A() As Integer, ByRef AA As Byte)
  Dim S As Byte, t As Byte, C As Byte, j As Byte, i As Byte, II As Byte, k As Byte
  Dim dc As Single, smalldist As Single, newdist As Single
  Dim distance() As Single
  Dim precede() As Byte, repath() As Byte
  Dim TT As String
  ReDim distance(N)
  ReDim precede(N)
  ReDim repath(N)
  Dim perm() As Integer
  ReDim perm(2 * N)
    S = 0
    t = N
    i = 0
    perm(0) = 0
    For II = 0 To N
      distance(II) = M
    Next II
    distance(S) = 0
    C = S
    Do While (C <> t)
      smalldist = M
      dc = distance(C)
      For II = 0 To N
        If Not find(II, perm(), i) Then
          newdist = dc + P(C, II)
          If newdist > M Then
            newdist = M
          End If
```

```
            If newdist < distance(II) Then
               distance(II) = newdist
               precede(II) = C
            End If
            If distance(II) < smalldist Then
               smalldist = distance(II)
               k = II
            End If
         End If
      Next II
      C = k
      i = i + 1            'count perm
      If i > N Then
         MsgBox "ไม่มีทางไป", vbExclamation + vbOKOnly, "Error"
         en = 1
         Exit Sub
      End If
      perm(i) = C
   Loop
   II = 0
   repath(II) = t
   Do
      II = II + 1
      repath(II) = precede(k)
      k = precede(k)
   Loop Until k = S
   ReDim A(II)
   For j = 0 To II
      A(j) = (repath(II - j))
      TT = TT + "  " + CStr(A(j))
   Next j
   AA = II
   FrmCamera.Label3.Caption = TT
End Sub
```

```
Private Function find(ByVal U As Byte, ByRef V() As Integer, ByVal j1 As Byte) As Boolean
Dim i1 As Byte
find = False
For i1 = 0 To j1
   If V(i1) = U Then
      find = True
   End If
Next i1
End Function
```

โปรแกรมการทำงานของหุ่นยนต์

```
; FILENAME          PROJECT_CAR.ASM
; DESCRIPTION       ROBOT  PROGRAM
; HARDWARE          COMPUTER  CONTROL  ROBOT
; ASSEMBLER         SXA51
; START-DATE        26/03/2001
; SOFTWARE ENG.     JATURAWIT J AND JIRAWAT M.
; COMPANY           KMITL

D_LED       EQU   20H       ;ตั้งค่าตัวแปร Delay ของ LED
COMMAND     EQU   21H       ;ตั้งค่าตัวแปร คำสั่งการควบคุมหุ่นยนต์
TIME        EQU   22H       ;ตั้งค่าตัวแปร Time เพื่อควบคุม Servo Motor
SPEED1      EQU   23H       ;ตั้งค่าตัวแปร Speed1 เพื่อควบคุมความเร็วเดินหน้า
SPEED2      EQU   24H       ;ตั้งค่าตัวแปร Speed2 เพื่อควบคุมความเร็วถอยหลัง

        ORG   0000H         ;ตั้งค่า Origin ที่ ROM ADDRESS 0000H
        SJMP  TEST_LED      ;กระโดดไปที่ TEST LED

        ORG   000BH         ;ROM ADDRESS INTERRUPT TIMER0
        JMP   TIME0         ;กระโดดไปที่ การควบคุม Servo Motor

        ORG   0023H         ;ROM ADDRESS INTERUPT SERIAL
        JMP   SERIAL        ;กระโดดไปที่ รับข้อมูล Serial

;******************************** TEST LED ******************************
TEST_LED:
        MOV   D_LED,#7FH    ;ตั้งค่า Delay การวิ่งของ LED
        MOV   R0,#08H       ;ตั้งค่าจำนวนการวน LOOP ไฟวิ่ง (มีทั้งหมด 8 ดวง)
        MOV   A,#01111111B  ;ใส่ค่าเพื่อจะเปิด LED ดวงซ้ายสุด
LED_R:                      ;LOOP ไฟวิ่งทางขวา
        MOV   P2,A          ;แสดงค่าที่ LED
        LCALL DELAY         ;หน่วงเวลา
```

```
        RR      A               ;หมุนค่าใน Reg A ทางขวา
        DJNZ    R0,LED_R        ;วน LOOP ไฟวิ่งทางขวา
        MOV     R0,#08H         ;ตั้งค่าจำนวนการวน LOOP ไฟวิ่ง (มีทั้งหมด 8 ดวง)
        MOV     A,#11111110B    ;ใส่ค่าเพื่อจะเปิด LED ดวงขวาสุด
LED_L:                          ;LOOP  ไฟวิ่งทางซ้าย
        MOV     P2,A            ;แสดงค่าที่ LED
        LCALL DELAY             ;หน่วงเวลา
        RL      A               ;หมุนค่าใน Reg A ทางซ้าย
        DJNZ    R0,LED_L        ;วน LOOP ไฟวิ่งทางซ้าย
        MOV     A,#0FFH         ;ใส่ค่าเพื่อจะปิด LED ทั้ง 8 ดวง
        MOV     P2,A            ; แสดงค่าที่ LED
;****************************** START   PROGRAM ***************************
START:
        MOV     COMMAND,#00H
        MOV     TMOD,#21H       ;SET TIMER1 MODE2(AUTO RE-LOAD)
                                ;SET TIMER0 MODE1(16 BIT)
        MOV     SCON,#50H       ;SET SERIAL MODE1
        MOV     TH1,#0D0H       ;SET BAUD RATE=600 (1200=E8,600=D0,300=A0)
        MOV     TH0,#0FEH       ;0.5 msec Timer0 #0FE33H
        MOV     TL0,#033H       ;0.5 msec Timer0 #0FE33H
        MOV     IE,#10010010B   ;SET INTERRUPT ENABLE SERIAL AND TIMER0
        MOV     IP,#00000010B   ;HIGHER PRIORITY TIMER0
        MOV     TIME,#0AH       ;SET TIME LOOP  5 msec
        SETB    TR1             ;START TIMER1
        SETB    TR0             ;START TIMER0
        CLR     P1.0            ;P1.0=RIGHT WHEEL
        CLR     P1.1            ;P1.1=LEFT WHEEL
        MOV     SPEED1,#01H     ;ตั้งค่าเพื่อควบคุมความเร็วเดินหน้า
        MOV     SPEED2,#05H     ;ตั้งค่าเพื่อควบคุมความเร็วถอยหลัง
        JMP     $               ;รอการ Interrupt จากทั้ง  Serial และ Timer0
;****************************** END   PROGRAM ****************************
```

```
;********************** ควบคุม Servo Motor โดยใช้ Timer0 **********************
TIME0:
        CLR   TR0                       ;Stop Timer0
        CLR   TF0                       ;Clear Timer0 Interrupt
        MOV   TH0,#0FEH                 ;0.5 msec Timer0 #0FE33H
        MOV   TL0,#033H                 ;0.5 msec Timer0 #0FE33H
        DJNZ  TIME,TTT                  ;Check ค่า Time Loop
        MOV   TIME,#0AH                 ;SET TIME LOOP 5 msec
        MOV   P1,#0FFH                  ;P1.0=RIGHT WHEEL , P1.1=LEFT WHEEL
TTT:    MOV   31H,A                     ;เก็บค่า Reg A
        JNB   COMMAND.2,STOP_R ;Check ว่า Motor ขวา ถูกสั่งให้หยุดหรือไม่
R_B:    JB    COMMAND.0,R_F     ;Check ว่า Motor ขวา ถูกสั่งให้ถอยหลังหรือไม่
        MOV   A,SPEED1                  ;สั่งให้ Motor ขวาถอยหลัง
        SJMP  RUN_R
R_F:    MOV   A,SPEED2                  ;สั่งให้ Motor ขวาเดินหน้า
RUN_R:
        CJNE  A,TIME,STOP_R             ;ให้ Motor ขวารับคำสั่ง
        SETB  P1.0
STOP_R:
        JNB   COMMAND.3,STOP_L ;Check ว่า Motor ซ้าย ถูกสั่งให้หยุดหรือไม่
L_B:    JB    COMMAND.1,L_F     ;Check ว่า Motor ซ้าย ถูกสั่งให้ถอยหลังหรือไม่
        MOV   A,SPEED2                  ;สั่งให้ Motor ซ้ายถอยหลัง
        SJMP  RUN_L
L_F:    MOV   A,SPEED1                  ;สั่งให้ Motor ซ้ายเดินหน้า
RUN_L:
        CJNE  A,TIME,STOP_L             ;ให้ Motor ซ้ายรับคำสั่ง
        SETB  P1.1
STOP_L:
        MOV   A,31H                     ;คืนค่า Reg A
        SETB  TR0                       ;Start Timer0
        RETI                            ;Return Interrupt
;************************** Return Interrupt Timer0 **************************
```

```
;************************ รับข้อมูล Serial จาก Computer ************************
SERIAL:
        CLR   RI                    ;Clear ค่า Receive Interrupt
        MOV   30H,A                 ;เก็บค่า Reg A
        MOV   A,SBUF                ;รับค่าคำสั่ง Serial จาก Computer
        MOV   COMMAND,#00H          ;Clear คำสั่ง
        JB    ACC.3,SS1             ;Check คำสั่งว่าเดินหน้าหรือไม่
        MOV   COMMAND,#00001111B    ;Set ค่าคำสั่งเดินหน้า
SS1:    JB    ACC.2,SS2             ;Check คำสั่งว่าเลี้ยวขวาหรือไม่
        MOV   COMMAND,#00001110B    ;Set ค่าคำสั่งเลี้ยวขวา
SS2:    JB    ACC.1,SS3             ;Check คำสั่งว่าถอยหลังหรือไม่
        MOV   COMMAND,#00001100B    ;Set ค่าคำสั่งถอยหลัง
SS3:    JB    ACC.0,SS4             ;Check คำสั่งว่าเลี้ยวซ้ายหรือไม่
        MOV   COMMAND,#00001101B    ;Set ค่าคำสั่งเลี้ยวซ้าย
SS4:    CPL   A                     ;กลับค่าใน Reg A เพื่อเตรียมแสดงผล
        MOV   P2,A                  ;แสดงค่า Serial ที่รับมา แสดงที่ LED
        MOV   A,30H                 ;คืนค่า Reg A
        RETI                        ;Return Interrupt
;************************** Return Interrupt Serial **************************

;******************************** Delay Loop ********************************
DELAY:
        MOV   R1,D_LED
D1:     MOV   R2,#0FFH
        DJNZ  R2,S
        DJNZ  R1,D1
        RET
;****************************** End Delay Loop ******************************

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

โปรแกรมการทำงานของตัวส่งสัญญาณ

```
; FILENAME     PROJECT_TRA.ASM
; DESCRIPTION  ROBOT PROGRAM
; HARDWARE     ROBOT REMOTE
; ASSEMBLER    SXA51
; START-DATE   26/03/2001
; SOFTWARE ENG. JATURAWIT J AND JIRAWAT M.
; COMPANY      KMITL

        ORG    0000H                ;ตั้งค่า Origin ที่ ROM ADDRESS 0000H
;****************************** START  PROGRAM ***************************
;****************************** สร้างคลื่นพาหะ 38 kHz ***********************
START:
        MOV    R0,#01H              ;ตั้งค่า T_ON 26 μSec ( Duty Circle ประมาณ 7% )
        SETB   P3.7                 ; T_ON
LOOP1:
        DJNZ   R0,LOOP1
        CLR    P3.7                 ; T_OFF
        MOV    R0,#08H              ;ตั้งค่า T_OFF 208 μSec
LOOP2:
        DJNZ   R0,LOOP2
        SJMP   START
;********************************** End ********************************
        END
```

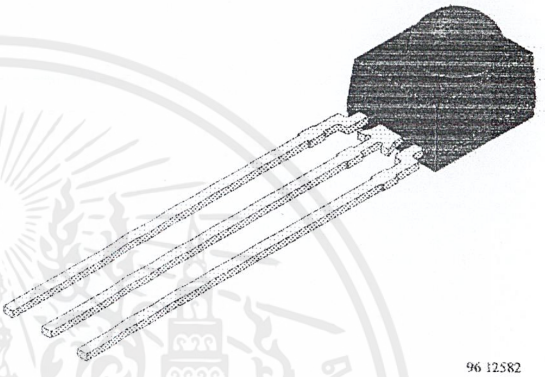
Photo Modules for PCM Remote Control Systems

Available types for different carrier frequencies

Type	fo	Type	fo
TSOP1830	30 kHz	TSOP1833	33 kHz
TSOP1836	36 kHz	TSOP1837	36.7 kHz
TSOP1838	38 kHz	TSOP1840	40 kHz
TSOP1856	56 kHz		

Description

The TSOP18.. – series are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter.
The demodulated output signal can directly be decoded by a microprocessor. The main benefit is the reliable function even in disturbed ambient and the protection against uncontrolled output pulses.



96 12582

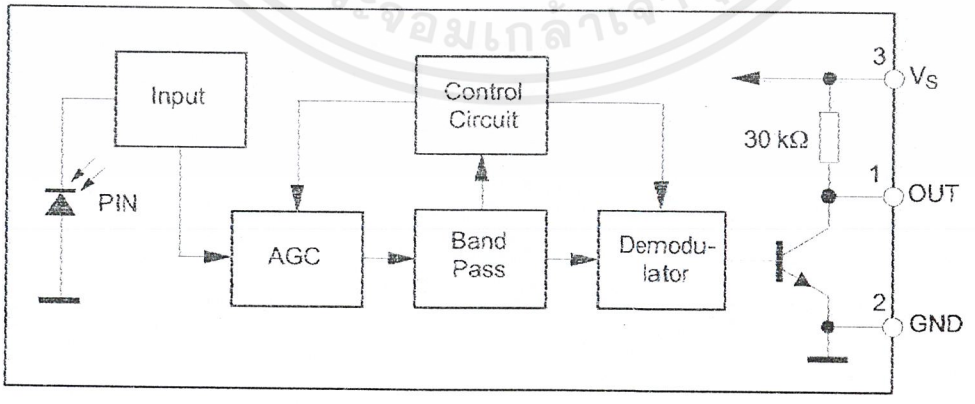
Features

- Photo detector and preamplifier in one package
- Internal filter for PCM frequency
- TTL and CMOS compatibility
- Output active low
- Improved shielding against electrical field disturbance
- Suitable burst length ≥6 cycles/burst

Special Features

- Small size package
- Enhanced immunity against all kinds of disturbance light
- No occurrence of disturbance pulses at the output
- Short settling time after power on (<200µs)

Block Diagram



9612226

Absolute Maximum Ratings

T_{amb} = 25°C

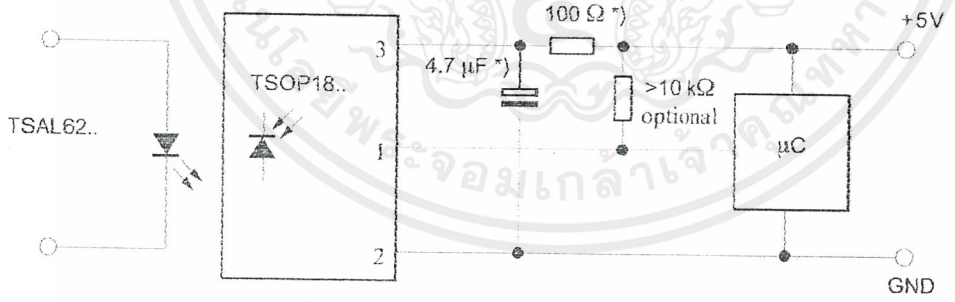
Parameter	Test Conditions	Symbol	Value	Unit
Supply Voltage	(Pin 3)	V_S	–0.3...6.0	V
Supply Current	(Pin 3)	I_S	5	mA
Output Voltage	(Pin 1)	V_O	–0.3...6.0	V
Output Current	(Pin 1)	I_O	5	mA
Junction Temperature		T_j	100	°C
Storage Temperature Range		T_{stg}	–25...+85	°C
Operating Temperature Range		T_{amb}	–25...+85	°C
Power Consumption	(T_{amb} ≦ 85 °C)	P_{tot}	50	mW
Soldering Temperature	t ≦ 10 s, 1 mm from case	T_{sd}	260	°C

Basic Characteristics

T_{amb} = 25°C

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Supply Current (Pin 3)	V_S = 5 V, E_v = 0	I_{SD}	0.9	1.2	1.5	mA
	V_S = 5 V, E_v = 40 klx, sunlight	I_{SH}		1.3		mA
Supply Voltage (Pin 3)		V_S	4.5		5.5	V
Transmission Distance	E_v = 0, test signal see fig.6, IR diode TSAL6200, I_F = 300 mA	d		35		m
Output Voltage Low (Pin 1)	I_{OSL} = 0.5 mA, E_e = 0.7 mW/m², f = f_o	V_{OSL}			250	mV
Irradiance (30 – 40 kHz)	Pulse width tolerance: $t_{pi} - 4/f_o < t_{po} < t_{pi} + 6/f_o$, test signal see fig.6	$E_{e\,min}$		0.3	0.5	mW/m²
Irradiance (56 kHz)		$E_{e\,min}$		0.4	0.7	mW/m²
Irradiance		$E_{e\,max}$	30			W/m²
Directivity	Angle of half transmission distance	$\varphi_{1/2}$		±45		deg

Application Circuit

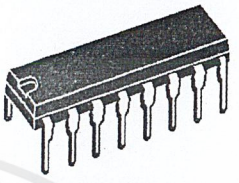


*) recommended to suppress power supply disturbances

ULN2001A-ULN2002A
ULN2003A-ULN2004A

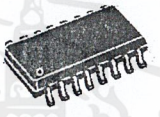
SEVEN DARLINGTON ARRAYS

- SEVEN DARLINGTONS PER PACKAGE
- OUTPUT CURRENT 500mA PER DRIVER (600mA PEAK)
- OUTPUT VOLTAGE 50V
- INTEGRATED SUPPRESSION DIODES FOR INDUCTIVE LOADS
- OUTPUTS CAN BE PARALLELED FOR HIGHER CURRENT
- TTL/CMOS/PMOS/DTL COMPATIBLE INPUTS
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY LAYOUT



DIP16

ORDERING NUMBERS: ULN2001A/2A/3A/4A



SO16

ORDERING NUMBERS: ULN2001D/2D/3D/4D

DESCRIPTION

The ULN2001A, ULN2002A, ULN2003 and ULN2004A are high voltage, high current darlington arrays each containing seven open collector darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

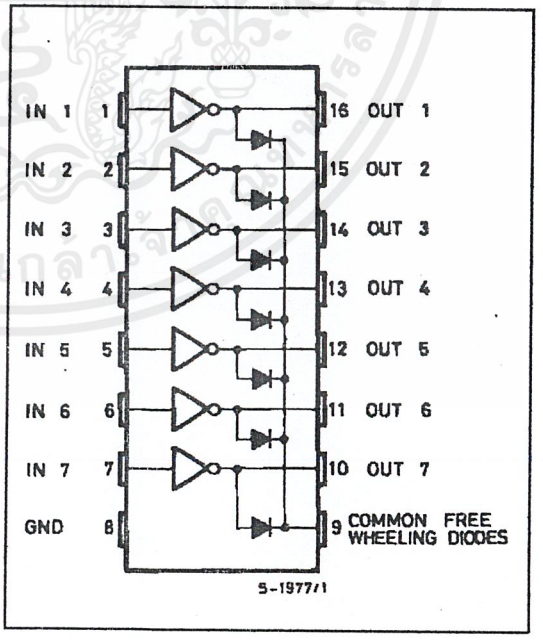
The four versions interface to all common logic families :

ULN2001A	General Purpose, DTL, TTL, PMOS, CMOS
ULN2002A	14-25V PMOS
ULN2003A	5V TTL, CMOS
ULN2004A	6–15V CMOS, PMOS

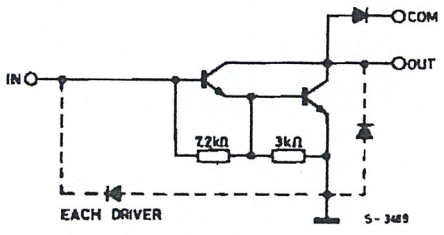
These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal printheads and high power buffers.

The ULN2001A/2002A/2003Aand 2004A are supplied in 16 pin plastic DIP packages with a copper leadframe to reduce thermal resistance. They are available also in small outline package (SO-16) as ULN2001D/2002D/2003D/2004D.

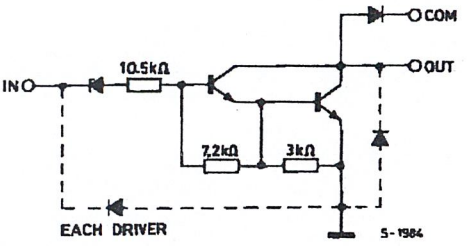
PIN CONNECTION



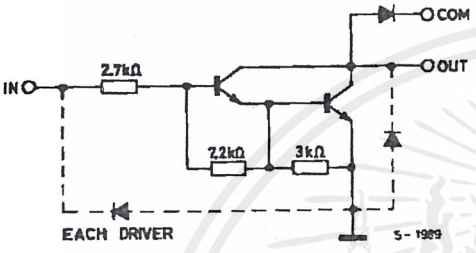
SCHEMATIC DIAGRAM



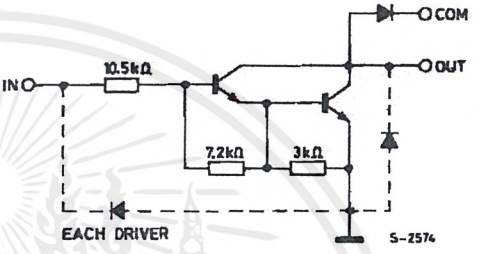
Series ULN-2001A
(each driver)



Series ULN-2002A
(each driver)



Series ULN-2003A
(each driver)



Series ULN-2004A
(each driver)

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_o	Output Voltage	50	V
V_{in}	Input Voltage (for ULN2002A/D - 2003A/D - 2004A/D)	30	V
I_c	Continuous Collector Current	500	mA
I_b	Continuous Base Current	25	mA
T_{amb}	Operating Ambient Temperature Range	– 20 to 85	°C
T_{stg}	Storage Temperature Range	– 55 to 150	°C
T_j	Junction Temperature	150	°C

THERMAL DATA

Symbol	Parameter		DIP16	SO16	Unit
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	70	100	°C/W

ELECTRICAL CHARACTERISTICS (T_{amb} = 25°C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit	Fig.
I_{CEX}	Output Leakage Current	V_{CE} = 50V			50	µA	1a
		T_{amb} = 70°C, V_{CE} = 50V			100	µA	1a
		T_{amb} = 70°C for ULN2002A V_{CE} = 50V, V_i = 6V			500	µA	1b
		for ULN2004A V_{CE} = 50V, V_i = 1V			500	µA	1b
$V_{CE(sat)}$	Collector-emitter Saturation Voltage	I_C = 100mA, I_B = 250µA		0.9	1.1	V	2
		I_C = 200 mA, I_B = 350µA		1.1	1.3	V	2
		I_C = 350mA, I_B = 500µA		1.3	1.6	V	2
$I_{i(on)}$	Input Current	for ULN2002A, V_i = 17V		0.82	1.25	mA	3
		for ULN2003A, V_i = 3.85V		0.93	1.35	mA	3
		for ULN2004A, V_i = 5V		0.35	0.5	mA	3
		V_i = 12V		1	1.45	mA	3
$I_{i(off)}$	Input Current	T_{amb} = 70°C, I_C = 500µA	50	65		µA	4
$V_{i(on)}$	Input Voltage	V_{CE} = 2V				V	5
		for ULN2002A I_C = 300mA			13		
		for ULN2003A I_C = 200mA			2.4		
		I_C = 250mA			2.7		
		I_C = 300mA			3		
		for ULN2004A I_C = 125mA			5		
		I_C = 200mA			6		
		I_C = 275mA			7		
		I_C = 350mA			8		
h_{FE}	DC Forward Current Gain	for ULN2001A V_{CE} = 2V, I_C = 350mA	1000				2
C_i	Input Capacitance			15	25	pF	
t_{PLH}	Turn-on Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	
t_{PHL}	Turn-off Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	
I_R	Clamp Diode Leakage Current	V_R = 50V			50	µA	6
		T_{amb} = 70°C, V_R = 50V			100	µA	6
V_F	Clamp Diode Forward Voltage	I_F = 350mA		1.7	2	V	7

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ รศ.ดร. จงกล งามวิวิทย์ และอาจารย์ ถาวร เบญจนราสุทธิ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอ กราบขอบพระคุณมา ณ ที่นี้

จตุรวิทย์ จันไพบูลย์
จิรวัฒก์ ม่วงดิษฐ์

บรรณานุกรม

[1] หนังสือ รีโมต เครื่องควบคุมไร้สาย บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) พิมพ์ครั้งแรก พ.ศ.2538

[2] รศ.สมยศ จุณณะปิยะ การประยุกต์ใช้ไมโครคอนโทรลเลอร์ ตระกูล MCS-51 พิมพ์ครั้งที่ 2 พ.ศ.2541

[3] วันสุระ ศรีใสดี ไมโครคอนโทรลเลอร์ภาคปฏิบัติ MCS-51 พิมพ์ครั้งแรก พ.ศ.2542

[4] อาจารย์สุนทร วิทูสุรพจน์ และอาจารย์ธีรวัฒน์ ประกอบผล ปฏิบัติการทดลองไมโครคอนโทรลเลอร์ด้วย JAZZ-31 บริษัท ศิลารีเสิร์ช จำกัด

[5] AARON M. TENENBAUM and MOSHE J. AUGENSTEIN, DATA STRUCTURES USING PASCAL, PRENTICE-HELL INC, 2nd edition, 1981

[6] ฉัททวุฒิ พืชผล และ พิชิต สันติกุลานนท์, คู่มือเรียน Visual Basic 6, บริษัท โปรวิชั่น จำกัด, พิมพ์ครั้งที่1 สิงหาคม 2542