หุ่นยนต์เตะฟุตบอล

SOCCER ROBOT

โดย

นาย ธนัฐ ระวังวงศ์

นาย พีระศักดิ์ ตั้งจิตอารีย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

หุ่นยนต์เตะฟุตบอล

SOCCER ROBOT

โดย

นาย ธนัฐ ระวังวงศ์ เลขประจำตัว 46015221

นาย พีระศักดิ์ ตั้งจิตอารีย์ เลขประจำตัว 46015234

อาจารย์ที่ปรึกษา

ผศ.ดร. ชูชาติ ปิณฑวิรุจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์	ปีการศึกษา 2548
ภาควิชา	อิเล็กทรอนิกส์
คณะ	วิศวกรรมศาสตร์
สถาบัน	เทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง (ภาษาไทย)	หุ่นยนต์เตะฟุตบอล
(ภาษาอังกฤษ)	SOCCER ROBOT

จัดทำโดย	นายธนัฐ	ระวังวงศ์	รหัส	46015221
	นายพีระศักดิ์	ตั้งจิตอารีย์	รหัส	46015234
อาจารย์ที่ปรึกษา	ผศ.ดร. ชูชาติ	ปิณฑวิรุจน์		

ปริญญานิพนธ์ฉบับนี้ได้ผ่านการตรวจสอบจากอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ .....อาจารย์ที่ปรึกษา

(ผศ.ดร. ชูชาติ ปิณฑวิรุจน์)

วันที่ 17/มี.ค./49

หุ่นยนต์เตะฟุตบอล

นาย ธนัฐ ระวังวงศ์ รหัส 46015221
นาย พีระศักดิ์ ตั้งจิตอารีย์ รหัส 46015234
ผศ.ดร.ชูชาติ ปิณฑวิรุจน์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2548

บทคัดย่อ

รายงานฉบับนี้เกี่ยวกับการออกแบบ และ การสร้างหุ่นยนต์เตะบอลซึ่งประกอบด้วย 3 ส่วนหลักคือ หุ่นยนต์ , การติดต่อแบบไร้สายและการประมวลผลทางภาพ โดยติดกล้องไว้เหนือสนามฟุตบอลและส่งสัญญาณภาพของหุ่นยนต์ไปให้เครื่องคอมพิวเตอร์ ซึ่งคอมพิวเตอร์จะประมวลหาตำแหน่งของหุ่นยนต์และลูกฟุตบอลและจากนั้นคอมพิวเตอร์จะส่งสัญญาณไปควบคุมให้หุ่นยนต์เคลื่อนไปเตะลูกบอล

SOCCER ROBOT

Mr. Tanat Rawangwong ID. 46015221

Mr. Peerasak Thangjitarec ID. 46015234

Assoc.Prof. Chuchat Pintawirut Advisor

Educational Year 2005

Abstract

This report concerns about design and construction of robot soccer. Which consists of three main units. The robot, the wireless telecommunication and the image processing unit . A camera installed over the foot ball field sends the image of the soccer robot to the PC. The image processing in a PC the position of robot and foot ball. The PC then sends command via wireless telecommunication to the robot to move and kick the ball.

กิตติกรรมประกาศ

รายงานฉบับนี้สำเร็จลุล่วงได้ด้วยดีด้วยความช่วยเหลือจากบุคคลหลายฝ่าย ทางคณะผู้จัดทำขอกราบพระคุณเป็นอย่างสูงไว้ ณ ที่นี้ โดยเฉพาะอย่างยิ่ง ผศ.ดร.ชูชาติ ปิณฑวิรุฬน์ ที่ให้คำปรึกษาและคำแนะนำในโครงการชิ้นนี้ ขอขอบคุณรุ่นพี่นักศึกษาปริญญาโททุกท่านที่ให้ความช่วยเหลือ ทำให้รายงานฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี

นาย ธนัฐ ระวังวงศ์

นาย พีระศักดิ์ ตั้งจิตอารีย์

สารบัญ

เรื่อง	หน้า
บทที่ 1 บทนำ	1
1.1 กล่าวนำ	1
1.2 ขอบเขตการทำงาน	1
บทที่ 2 ทฤษฎีการสื่อสาร	2
2.1 การมอดูเลต	2
2.2 หลักการระบบสื่อสาร	2
2.3 ทำไมต้องมีการมอดูเลชั่น	3
2.4 RF Transmission	4
2.5 HT12E IC Encoder ของภาคส่ง	10
2.5.1 Address/data programming (preset)	11
2.5.2 Transmission enable	12
2.6 HT12D IC Decoder ของภาครับ	12
2.6.1 การทำงานของ HT12D	13
บทที่ 3 ความรู้เบื้องต้นเกี่ยวกับเดลไฟ	17
3 .1 รู้จักกับ Delphi7	17
3 .2 หลักการเขียนโปรแกรมด้วย Delphi7	17
3 .2.1ตัวแปลและชนิดของข้อมูล	17
3.2.2 การประกาศตัวแปร	18
3 .2.3 การใช้งานค่าคงที่	19
3.2.4 ตัวดำเนินการเปรียบเทียบ	19
3.2.5 การควบคุมทิศทางการทำงานของโปรแกรม	20
3.2.6 สร้างและใช้งาน Procedure	21
3.2.7 สร้างและใช้งาน Function	22
บทที่ 4 การใช้งานพอร์ตอนุกรม	23

4.1 การสื่อสารแบบอนุกรม 23
4.1.1 ประโยชน์ของพอร์ตอนุกรม 23
4.1.2 การเพิ่มระยะทางของพอร์ตอนุกรม 23
4.1.3 ลักษณะสัญญาณของพอร์ตอนุกรม 23
4.1.4 ปัญหาด้านรับไม่สามารถรับข้อมูลไม่ได้ 24
4.1.5 การTest ว่าพอร์ตอนุกรมของ PC เสียหรือไม่ 26
4.2 การติดต่อและควบคุมSerial Port 26
4.2.1 พื้นฐานการสื่อสารแบบอนุกรม 26
4.2.2รู้จักกับมาตรฐานRS-232 C 27
4.2.3 องค์ประกอบของการรับส่งข้อมูลแบบอนุกรม 28
4.2.4 อัตราเร็วในการรับส่งข้อมูลแบบอนุกรม 29
บทที่ 5 การขับมอเตอร์ (Drive Motor) 30
5.1 การควบคุมทิศทางการหมุนของDC Motor
หลักการทำงานของวงจร H-Bridge Switching 30
5.2 การควบคุมทิศทางการหมุนของ DC Motor
สร้างวงจร H-Bridge Switching จาก Transistor 32
บทที่ 6 อิมเมจและการประยุกต์ใช้งาน 36
6.1 การเปลี่ยนแปลงรูปแบบทางเลขาคณิต 36
6.2 การทำเทรสโชว์ (Image thresholding) 43
6.3 การหักลบภาพและตรวจการเปลี่ยนแปลง 43
(Image Subtraction and Change Detection)
6.4 การแยก RGB 44
บทที่ 7 แนวทางการออกแบบ 45
7.1 การทำงานโดยสังเขป 45
7.2 แนวการออกแบบ 46
7.3 การส่งสัญญาณไปควบคุมการทำงานของ HARD WARE 46
7.4 การเขียนโปรแกรมควบคุมการทำงานของกล้อง 48
7.5 การเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทลเลอร์ 60

บทที่ 8 การทดลอง และ ผลการทดลอง 62
บทที่ 9 บทสรุปและวิจารณ์ 77
ภาคผนวก 78
กิตติกรรมประกาศ 88
เอกสารอ้างอิง 89

บทที่ 1
บทนำ

1.1 กล่าวนำ

ปัจจุบันนี้เรื่องราวของการเชื่อมต่อระหว่างอุปกรณ์ภายนอกกับเครื่องคอมพิวเตอร์โดยผ่านทางพอร์ตของคอมพิวเตอร์ เป็นสิ่งที่จำเป็นที่ต้องศึกษาทั้งทางด้านฮาร์ดแวร์และซอฟแวร์เพื่อที่จะสามารถนำไปประยุกต์ใช้งานได้อย่างมีประสิทธิภาพ

ด้วยเหตุนี้ผู้จัดทำจึงได้ทำการศึกษาถึงการเชื่อมต่ออุปกรณ์ภายนอกผ่านทางพอร์ตอนุกรมของคอมพิวเตอร์โดยใช้โปรแกรม เดลไฟ ในการเขียนซอฟแวร์เพื่อควบคุมฮาร์ดแวร์ โดยฮาร์ดแวร์ก็คือหุ่นยนต์เตะฟุตบอลที่ได้นำเอารถบังคับวิทยุมาประยุกต์ให้มีความสามารถตรงตามวัตถุประสงค์

1.2 ขอบเขตการทำงาน

ในการจัดทำโครงงานนี้มีขอบเขตของโครงงานคือ

ศึกษาความรู้เบื้องต้นเกี่ยวกับการเชื่อมต่ออุปกรณ์ภายนอกผ่านทางพอร์ตอนุกรมของคอมพิวเตอร์

ศึกษาการทำงานของเครื่องรับ-ส่งวิทยุของ TLP315 กับ RLP315

ศึกษาการใช้งานโปรแกรม เดลไฟ เบื้องต้น

ศึกษาการประมวลผลทางอิมเมจ โดยใช้กล้องจับภาพเพื่อหาตำแหน่งของหุ่นยนต์

สามารถควบคุมการทำงานของหุ่นยนต์โดยอัตโนมัติ

บทที่ 2
ทฤษฎีการสื่อสาร

2.1 การมอดูเลต

ในขบวนการมอดูเลต เราใช้คลื่นรูปซายน์ที่มีความถี่สูงเป็นพาหะ แล้วเปลี่ยนแปลงคุณสมบัติบางอย่างของพาหะด้วยสัญญาณข่าวสาร โดยทั่วไปสัญญาณข่าวสารได้แก่ สัญญาณออดิโอ(หรือเสียงพูด) สัญญาณภาพ หรือข่าวสารอื่นๆ การเปลี่ยนแปลงคุณสมบัติของคลื่นพาหะนี้เรียกว่า การมอดูเลต

การมอดูเลตให้กับคลื่นพาหะแบ่งออกได้เป็น 3 แบบ

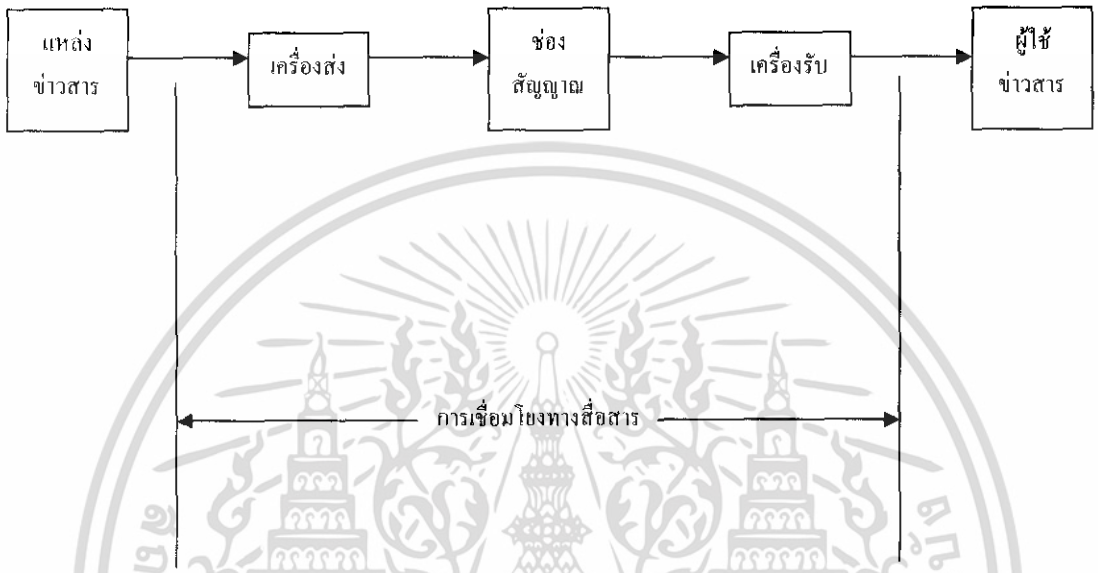
1. มอดูเลตทางแอมพลิจูด (amplitude modulation หรือ AM)
2. มอดูเลตทางความถี่ (frequency modulation หรือ FM)
3. มอดูเลตทางเฟส (phase modulation หรือ PM)

ในทางปฏิบัติสัญญาณ FM กับสัญญาณ PM จะคล้ายคลึงกันมาก บางทีเราเรียกรวมๆทั้ง FM และ PM ว่า การมอดูเลตทางมุม (angle modulation) กล่าวโดยสรุป การมอดูเลตแบ่งออกเป็น 2 แบบใหญ่ๆ คือ AM และ FM (หรือ PM)

2.2 หลักการระบบสื่อสาร

ระบบการสื่อสารสามารถอธิบายการทำงานได้ตามรูปที่ 2.1 ดังนี้ โดยข่าวสารจะถูกแปลงให้อยู่ในรูปของสัญญาณไฟฟ้าที่เรียกว่า สัญญาณมอดูเลตติ้ง (molulating signal) สัญญาณนี้จะแปลงให้อยู่ในรูปของรหัสตามวิธีการสื่อสารแบบดิจิตอลหรือส่งตรงเข้าเครื่องส่งตามวิธีการสื่อสารแบบอนาล็อกก็ได้ ในเครื่องส่งจะมีตัวมอดูเลตที่ทำหน้าที่รวมสัญญาณมอดูเลตติ้งกับตัวพาห์ (carrier) ตัวพาห์นี้มีกำลังส่งสูงพอที่จะพาสัญญาณมอดูเลตติ้งไปที่ไกลๆได้ด้วยความถี่สูงตามกระบวนการมอดูเลชั่น (modulation) จานั้นสัญญาณจะถูกคัปปลิ้งออกอากาศหรือส่งตามสายส่งก็ได้สัญญาณที่ผ่านช่องสัญญาณ (channel) ซึ่งไม่ว่าจะสายส่งหรืออากาศก็ตามจะถูกรบกวนจากสัญญาณรบกวนภายนอก (noise) เมื่อสัญญาณไปถึงเครื่องรับของผู้ใช้ปลายทาง เสาอากาศของเครื่องรับจะแปลงสัญญาณที่เป็นคลื่นแม่เหล็กไฟฟ้า แต่ถ้าส่งตามสายเครื่องรับจะรับสัญญาณในรูปของกระแสไฟฟ้าหรือแรงดันไฟฟ้าได้ทันที จากนั้นตัวดีมอดูเลเตอร์ (demodulator) ในเครื่องรับจะแปลงสัญญาณที่มีความถี่สูงให้มีความถี่ต่ำลงและแยกสัญญาณ

มอดูเลตติ้งออกจากตัวพาห์ตามกระบวนการดีมอดูเลชั่น (demodulation) และถูกถอดรหัส (decode) กลับเป็นสัญญาณอนาล็อกเหมือนเดิม ตามวิธีการสื่อสารแบบดิจิตอล



รูปที่ 2.1 แผนภาพของระบบสื่อสาร

2.3 ทำไมต้องมีการมอดูเลชั่น

1 ทำให้สัญญาณมีกำลังสูงสามารถเดินไปที่ไกลๆได้

2 ทำให้สัญญาณมีความถี่สูงขึ้น ซึ่งเหมาะสำหรับการรับส่งสัญญาณมากขึ้นเพราะใช้เสาอากาศที่สั้นลงได้

3 สามารถแบ่งความถี่ให้หลายๆสัญญาณส่งพร้อมกันภายใต้ตัวพาห์ตัวเดียวกันได้ เรียกว่า การมัลติเพล็กซิง

4 ทำให้สัญญาณมีความต้านทานการรบกวนของสัญญาณรบกวนได้ดีขึ้น

2.4 RF Transmission

ในการติดต่อกันระหว่างคอมพิวเตอร์กับหุ่นยนต์นั้นจะใช้ RF เบอร์ TLP315 และRLP315 ซึ่งเป็นโมดุลในการรับ-ส่ง RF โดย TLP315 เป็นโมดุลตัวส่งที่มี 4 ขา ประกอบไปด้วย

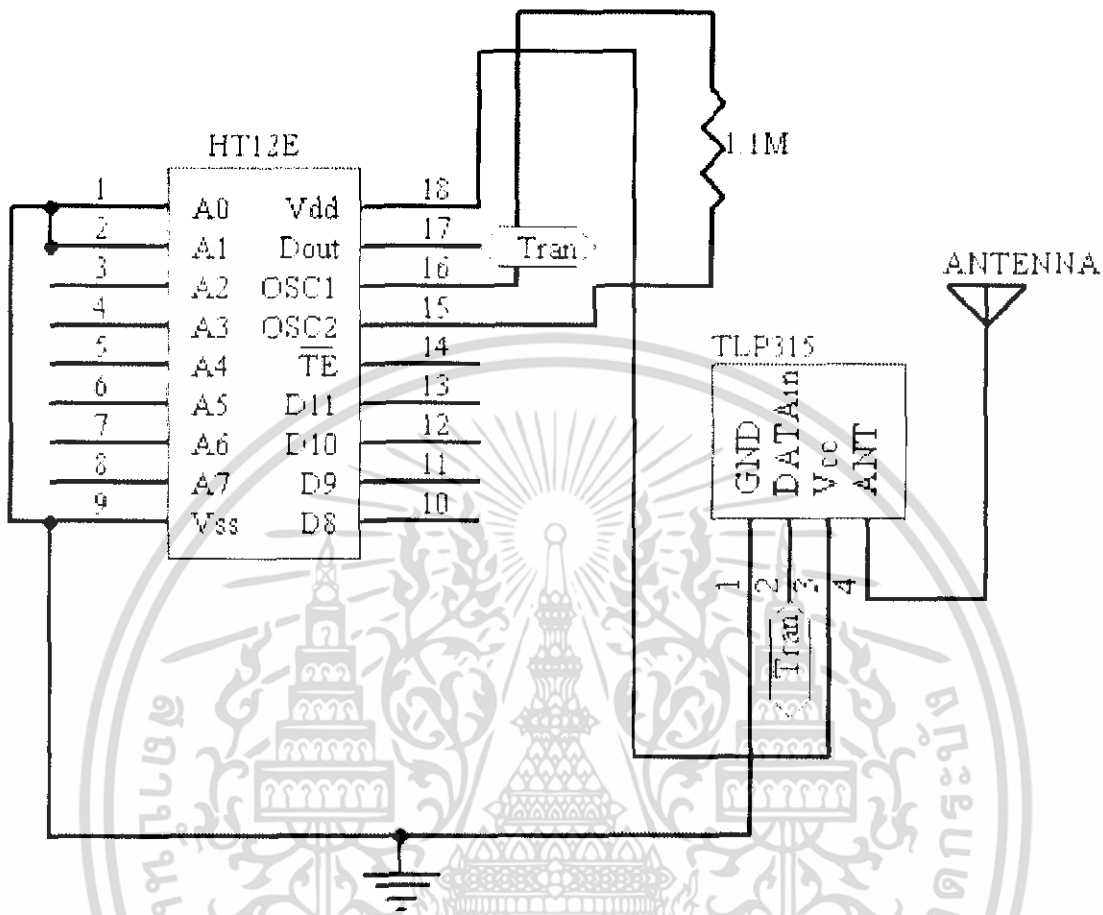
1. GND
2. Data In
3. Vcc
4. Antenna (RF Output)

ดังรูปที่ 2.1 แสดงขา 1 , 2 , 3 และขา 4 เรียงจากซ้ายไปขวาตามลำดับ

รูปที่ 2.2 แสดง TLP315 ภาคส่ง RF

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating Supply Voltage	-	2.0	-	12.0	V
Icc 1	Peak Current (2V)	-	-	-	1.64	mA
Icc2	Peak Current (2V)	-	-	-	19.4	mA
Vh	Input High Voltage	Idata = 100μA(High)	Vcc-0.5	Vcc	Vcc+0.5	V
Vl	Input Low Voltage	Idata = 0μA(Low)	-	-	0.3	V
FO	Absolute Frequency	315 MHz Module	314.8	315	315.2	MHz
PO	RF Output Power = 50 Ω	Vcc = 9-12V	-	16	-	dBm
DR	Data Rate	External Encoding	512	4.8K	200K	bps

ตารางที่ 2-1 แสดงข้อมูลของ TLP315



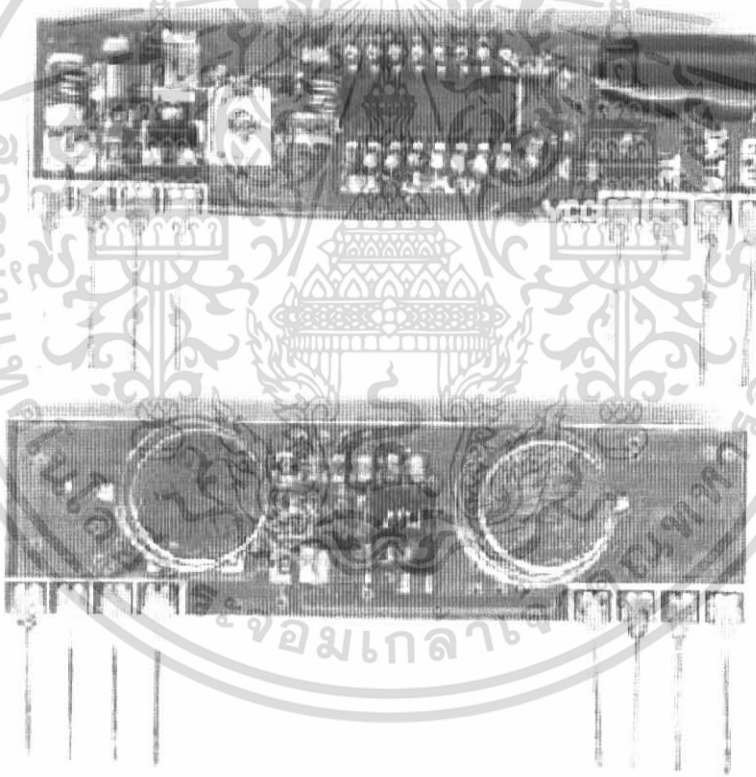
รูปที่ 2.3 แสดงการประยุกต์ใช้งาน TLP315

เมื่อทำการ Set ค่า A_0 และ A_1 แล้วป้อนข้อมูลเข้าทาง D_8-D_{11} และทำการ Set ขา TE ของ HT12E ให้เป็น 0 แล้วข้อมูลจะถูก Encoder มาที่ขา D_{OUT} ของ HT12E และเมื่อเรานำขานี้ไปต่อกับขา Data In ของ TPL315 ก็จะกลายเป็นการส่งข้อมูลออกไปทาง Antenna ของ TLP315

และ RLP315 ซึ่งเป็นโมดุลที่มี 8ขา ประกอบด้วย

1. GND
2. Digital Data Output
3. Linear Output/Test
4. Vcc
5. Vcc
6. GND
7. GND
8. Antenna

ดังรูปที่ 2.3 เรียงตามขาจากซ้ายไปขวา 1, 2, 3, 4, 5, 6, 7 และ 8 ตามลำดับ



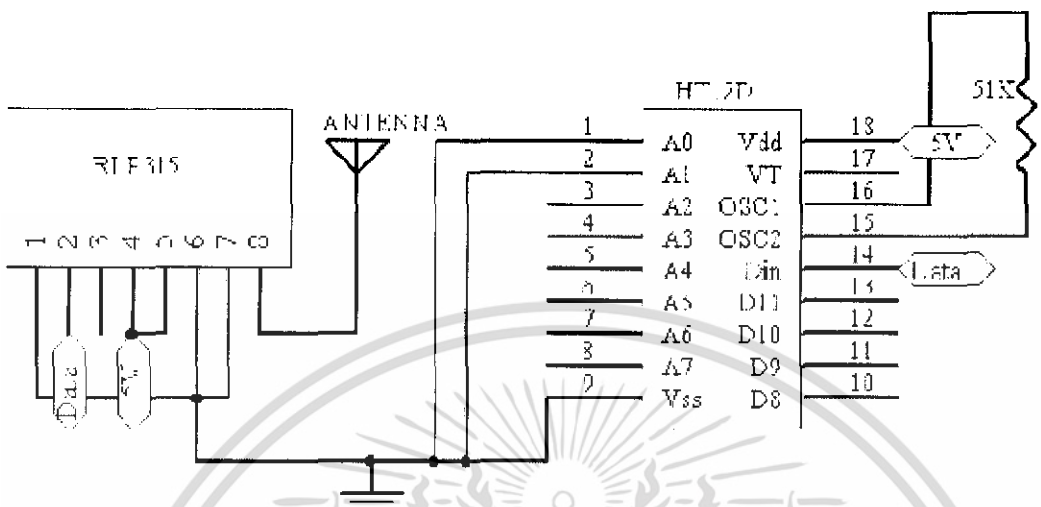
รูปที่ 2.4 แสดง RLP315 ภาครับ RF

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating Data Voltage	-	3.3	5.0	6.0	V
Itot	Operating Current	-	-	4.5	-	mA
Vdata	Data Out	Idata = +200 μA (High)	Vcc-0.5	-	Vcc	V
		Idata = -10 μA (Low)	-	-	0.3	V

Electrical Characteristics					
Characteristics	**SYM**	**Min**	**Typ**	**Max**	**Unit**
Operation Radio Frequency	FC	315,418 and 433.29			MHz
Sensitivity	Perf	-	-110	-	dBm
Channel Width	-	-	+/- 500	-	kHz
Noise Equivalent BW	-	-	4	-	kHz
Receiver Turn On Time	-	-	5	-	Ms
Operation Temperature	TOP	-20	-	80	C
Baseboard Data Rate	-	-	4.8	-	Hz

ตารางที่ 2-2 แสดงข้อมูลของ RLP315

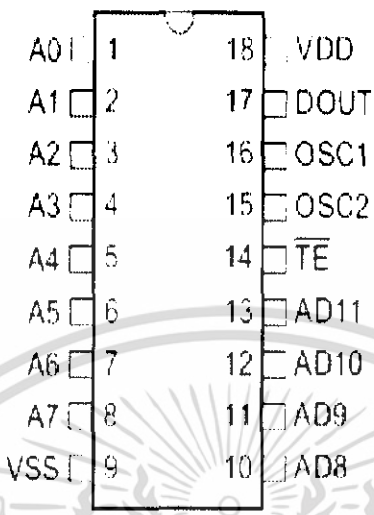
การประยุกต์ใช่งานเราจะใช่งานร่วมกับ IC เบอร์ HT12E และ HT12D ซึ่งเป็นตัว Encoder และ Decoder เพื่อที่จะทำให้สามารถเพิ่มช่องสัญญาณในการรับ-ส่งได้



รูปที่ 2.5 แสดงการประยุกต์ใช้งาน RLP315

เมื่อข้อมูลไปถึงภาครับ โดยรับจาก Antenna ของ RLP315 ซึ่งข้อมูลนี้ก็คือข้อมูลที่ถูก Encoder มาแล้วเราก็นำข้อมูลนี้ผ่านเข้าไปที่ HT12D (ที่ถูก Set ขา A_0 และ A_1 ให้ตรงกับตัวส่งแล้ว) ทางขา Data In ของ HT12D ข้อมูลที่ถูก Decode แล้วก็จะปรากฏอยู่ที่ D_8–D_{11} ของ HT12D ในขณะที่ขา VT ของ HT12D ก็จะเป็น Logic 1 เพื่อแสดงว่า HT12D สามารถ Decode สัญญาณได้

2.5 HT12E IC Encoder ของภาคส่ง



รูปที่ 2.6 แสดงภาพ IC HT12E

Pin Name	I/O	Internal Connection	Description
A0-A7	I	NMOS Transmission Gate Protection Diode	Input pins address A0-A7 setting. These pins can be externally set to VSS or left open
AD8-AD11	I	NMOS Transmission Gate Protection Diode	Input pins for address/data AD8-AD11 setting. These pins can be externally set to VSS or left open
DOUT	O	CMOS OUT	Encoder data serial transmission output
TE	I	CMOS IN Pull – high	Transmission enable, active low
OSC 1	I	OSCILLATOR 1	Oscillator input pin
OSC 2	O	OSCILLATOR 2	Oscillator input pin
VSS	I	-	Negative power supply, grounds
VDD	I	-	Positive power supply

ตารางที่ 2-3 แสดงคำบรรยายขาแต่ขาของ HT12E

Symbol	Parameter	Test Conditions		Min	Typ.	Max.	Unit
		Vdd	Conditions				
V_{DD}	Operating Voltage	-	-	2.4	5	12	V
I_{STB}	Operating Current	3V	Oscillator Stops	-	0.1	1	μA
		12V		-	2	4	μA
I_{DD}	Operating Current	3V	No load f_{OSC}= 3kHz	-	40	80	μA
		12V		-	150	300	μA
I_{DOUT}	Output drive current	5V	V_{OH}= 0.9 V_{DD}(Source)	-1	-16	-	mA
			V_{OL}= 0.1 V_{DD}(Sink)	1	16	-	mA
V_{IH}	"H" Input Voltage	-	-	0.8V_{DD}	-	V_{DD}	V
V_{IL}	"L" Input Voltage	-	-	0	-	0.8V_{DD}	V
f_{OSC}	Oscillator Frequency	5V	R_{OSC}= 1.1 MΩ	-	3	-	kHz
R_{TE}	TE pull – high resistance	5V	V_{TE}= 0V	-	1.5	3	MΩ

ตารางที่ 2-4 แสดงข้อมูลของ HT12E

2.5.1 Address/data programming (preset)

สถานะของแต่ละขาของ address/data สามารถกำหนดค่าจำเพาะได้โดยให้ preset เป็นลกจิก Low หรือ High ถ้าสัญญาณ transmission-enable นำมาใช่ encoder สแกนและส่งผ่านสถานะของ 12 บิต ของ address/data แบบอนุกรมผ่าน A0 ถึง AD11 สำหรับ HT12E encoder

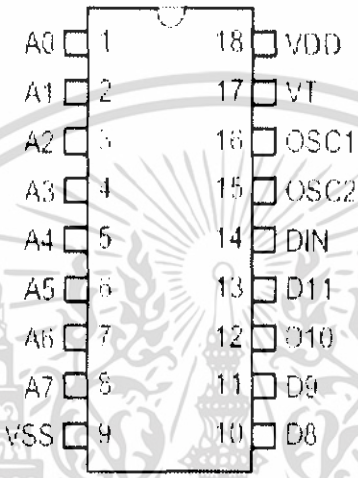
เนื่องจากการส่งผ่านข้อมูล บิตเหล่านี้จะถูกส่งด้วยกระบวนการซิงโครนัส ถ้า trigger signal ไม่ทำงาน IC จะเข้าสู่โหมด standby คือไม่ทำงานและกระแสจะลดลงน้อยกว่า 1uA สำหรับแหล่งจ่ายแรงดัน 5 โวลต์

โดยปรกติการประยุกต์ใช้ preset ขา address ด้วยรหัสจำเพาะโดยใช้ DIP หรือ PCB ขณะที่ข้อมูลถูกเลือกโดยการกดสวิตซ์

2.5.2 Transmission enable

สำหรับ Encoder HT12E การส่งผ่านจะถูกกำหนดโดย ใช้สัญญาณ Low ให้กับขา TE

2.6 HT12D IC Decoder ของภาครับ



รูปที่ 2.7 แสดงภาพ IC HT12D

Pin Name	I/O	Internal Connection	Description
A0-A7	I	NMOS Transmission Gate	Input pins for address A0-A7 setting. These pins can be externally set to VSS or VDD
AD8-AD11	O	CMOS OUT	Output data pins
DIN	I	CMOS IN	Serial data pin
VT	O	CMOS OUT	Valid Transmission, active high
OSC 1	I	OSCILLATOR 1	Oscillator input pin
OSC 2	O	OSCILLATOR 2	Oscillator input pin
VSS	I	-	Negative power supply, grounds
VDD	I	-	Positive power supply

ตารางที่ 2-5 แสดงคำบรรยายขาแต่ละขาของ HT12D

Symbol	Parameter	Test Conditions		Min	Typ.	Max.	Unit
		Vdd	Conditions				
V_{DD}	Operating Voltage	-	-	2.4	5	12	V
I_{STB}	Operating Current	3V	Oscillator Stops	-	0.1	1	μA
		12V		-	2	4	μA
I_{DD}	Operating Current	5V	No load f_{OSC}= 150kHz	-	200	400	μA
I_O	Data output source current (D8-D11)	5V	V_{OH}= 4.5V	-1	-16	-	mA
	Data output silk current (D8-D11)		V_{OL}= 0.5V	1	16	-	mA
I_{VT}	VT output source current	5V	V_{OH}= 4.5V	-1	-16	-	mA
	VT output silk current		V_{OL}= 0.5V	1	16	-	
V_{IH}	"H" Input Voltage	5V	-	3.5	-	5	V
V_{IL}	"L" Input Voltage	5V	-	0	-	1	V
F_{OSC}	Oscillator Frequency	5V	R_{OSC}= 51kHz	-	150	-	kHz

ตารางที่ 2-6 แสดงข้อมูลของ HT12D

2.6.1 การทำงานของ HT12D

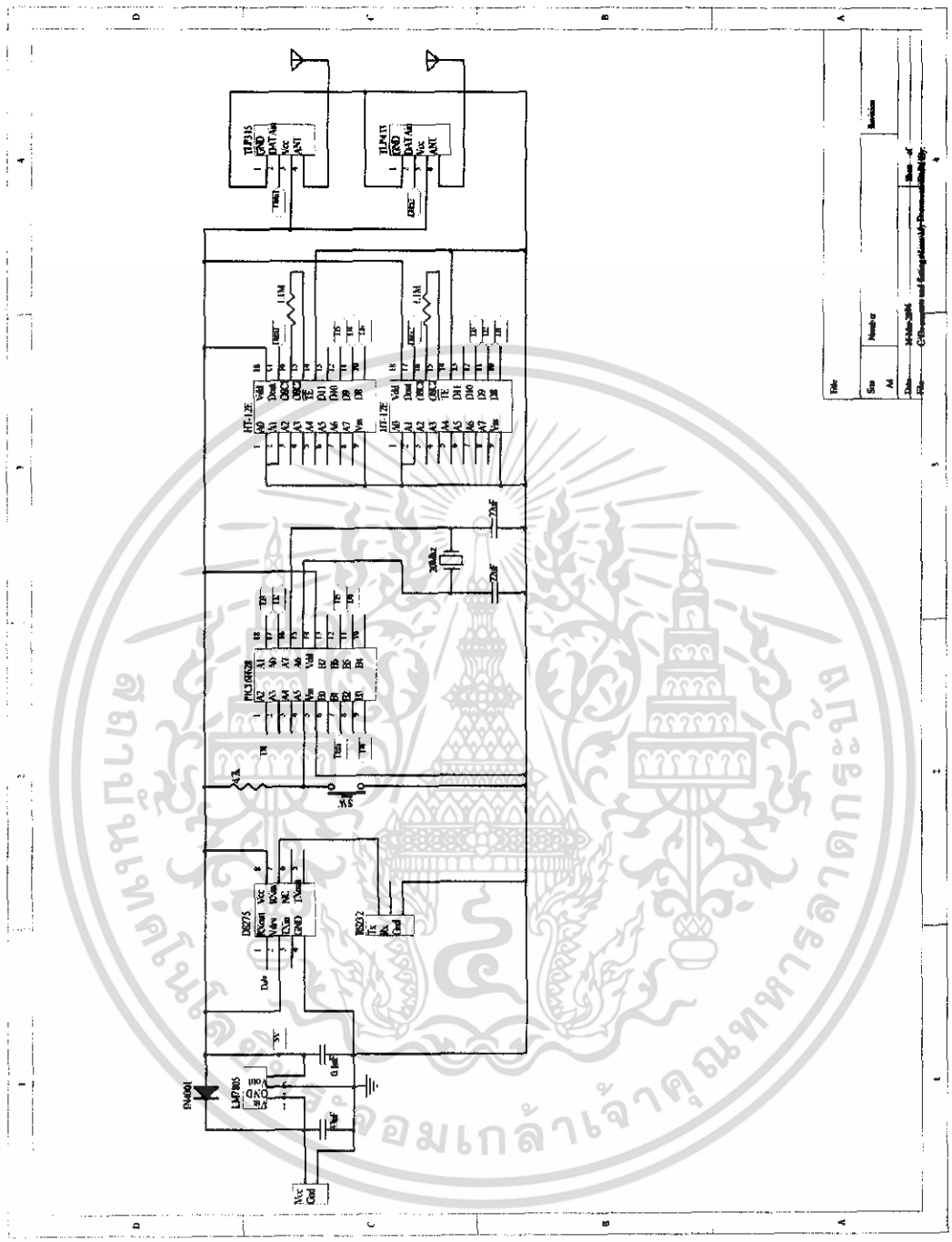
12 บิต อนุกรมของ decoder สามารถแบ่งได้หลายกรณีโดยการรวม ขา Address และขา Data โดยการจับคู่กับ 12 บิต อนุกรมของ encoder

Decoder รับข้อมูลที่ส่งมาจาก encoder แปล N บิตแรกของรหัสเป็น Address และ 12-N บิตสุดท้ายเป็น Data เมื่อ N เป็นรหัส Address สัญญาณที่ขา DIN active จึงจะทำการ decode ข้อมูล

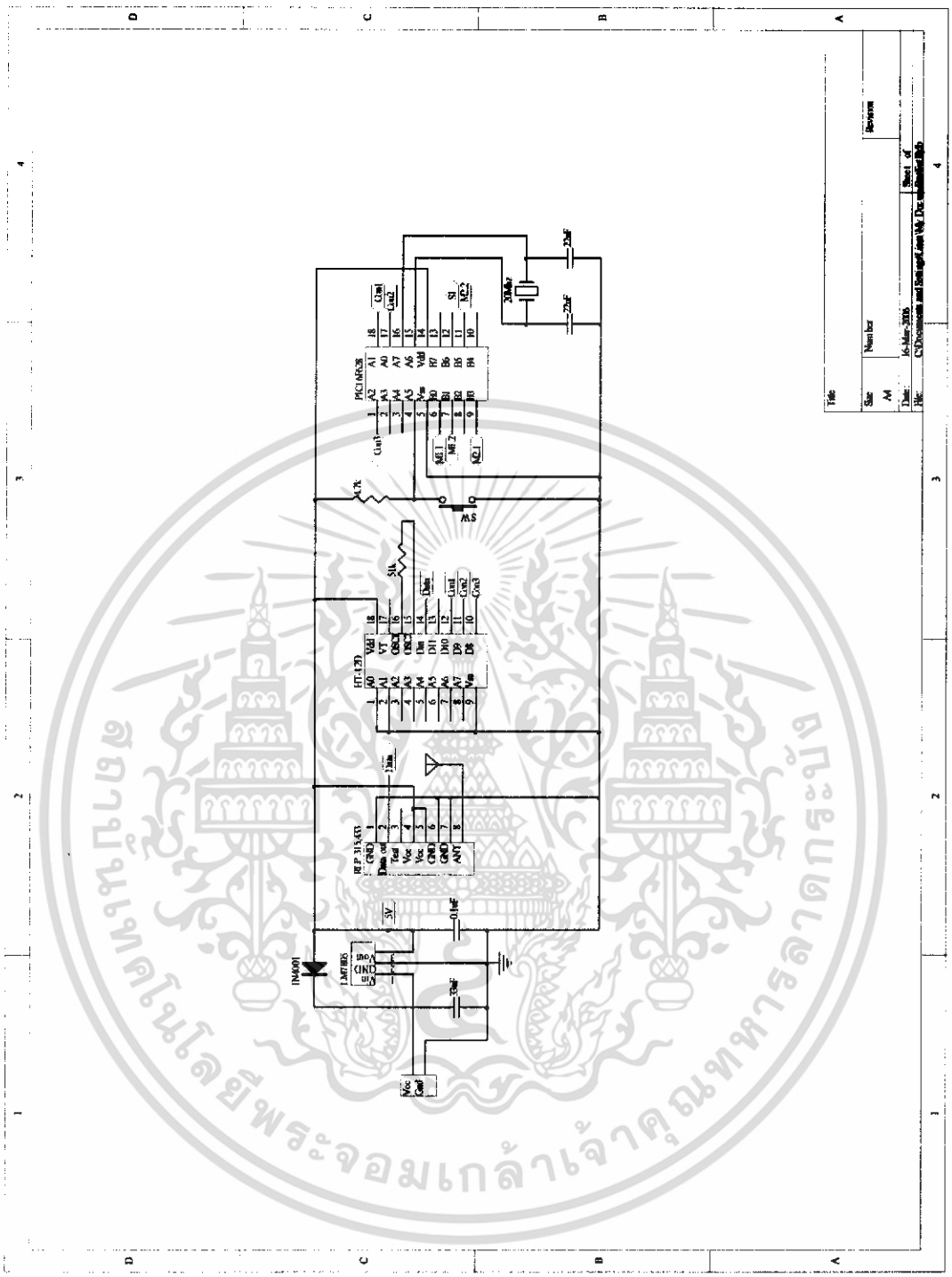
ข้อมูล

Decoder จะ check ข้อมูลจาก Address ที่ได้รับมาอย่างต่อเนื่อง 3 ครั้ง ถ้า Address Code ตรงกับ Address ของ decoder 12-N บิตของข้อมูลจะถูก decode ไปที่ขา Output และขา VT จะเป็น High เพื่อบอกว่าข้อมูลถูกต้อง

ดังนั้นขา VT จะเป็น High เมื่อข้อมูลถูกต้องเท่านั้น โดยปรกติจะเป็น Low



รูปที่ 2.8 วงจรภาคส่ง



รูปที่ 2.9 วงจรภาครับ

บทที่ 3
ความรู้เบื้องต้นเกี่ยวกับเดลไฟ

3.1 รู้จักกับ Delphi7

Delph7 คือ ซอฟแวร์ที่นำมาใช้เขียนโปรแกรมเพื่อสร้างแอพพลิเคชั่น หรือซอฟแวร์อีกที โดยมันจะประกอบไปด้วยเครื่องมือชนิดต่างๆ ที่ใช้ในการเขียนโปรแกรมที่ทำงานได้สะดวก

Delphi7 จัดเป็นเครื่องมือเขียนโปรแกรมชนิด Visual Programming เช่นเดียวกับ Visual Basic หรือ Visual C++ โดยมีข้อดีคือ สามารถเขียนโปรแกรมได้ง่าย และให้ผลงานออกมารวดเร็ว ซึ่งจะแตกต่างจากเครื่องมือเขียนโปรแกรมรุ่นเดิมๆ ที่มีความยุ่งยากซับซ้อนดังนั้น Delphi7 จึงใช้งานได้อย่างง่านและรวดเร็วกว่า

Delphi7 ได้รับการพัฒนาให้สามารถใช้แอพพลิเคชั่นแบบข้ามแพล็ตฟอร์มได้นั่นคือ สามรถพัฒนาแอพพลิเคชั่นที่ทำงานได้ทั้งบน Windows และลีนุกซ์นั่นเอง

Delphi7 นั้นมีความสามารถมากมาย ได้รับการต้อนรับเป็นอย่างดีจากนักพัฒนาแอพพลิเคชั่นทั่วโลกรวมทั้งเมืองไทยด้วย สามรถสร้างแอพพลิเคชั่นบน Windows ได้หลายเวอร์ชั่น ซึ่งแอพพลิเคชั่นที่สร้างจาก Delphi ได้รับการยกย่องเป็นอย่างมากในด้านขนาดโปรแกรมที่กะทัดรัดและทำงานได้อย่างรวดเร็ว เมื่อเทียบกับแอพพลิเคชั่นที่สร้างจากเครื่องมือตัวอื่นๆ ทำให้ Delphi แต่ละเวอร์ชั่นได้รับรางวัลจากสถาบันต่างๆ ทั่วโลก

สำหรับในประเทศไทย Delphi ถูกนำไปใช้พัฒนาระบบงานด้านฐานข้อมูลโดย Delphi มีจุดเด่นในการสร้างแอพพลิเคชั่นที่ติดต่อกับฐานข้อมูลได้หลายรูปแบบ

3.2 หลักการเขียนโปรแกรมด้วย Delphi7

3.2.1 ตัวแปลและชนิดของข้อมูล

ก่อนการเขียนโปรแกรมสิ่งแรกที่ต้องรู้จักคือ ข้อมูล เพราะข้อมูลคือ สิ่งที่เราใช้ในการควบคุมและแลกเปลี่ยนกันระหว่างผู้ใช้งานกับตัวโปรแกรม

เมื่อมีข้อมูลแล้วเราจะต้องมีสิ่งที่เรียกว่า ตัวแปร ขึ้นมาเก็บข้อมูลนั้นไว้ซึ่ง Delphi7 นั้นสามารถทำงานกับข้อมูลได้หลายชนิด แบ่งข้อมูลได้เป็นกลุ่มๆ ดังนี้

1. ข้อมูลเบื้องต้น(Simple Type) เป็นข้อมูลชนิดเบื้องต้นที่Delphiใช้งานพื้นฐานเช่น ตัวอักษร, เลขจำนวนเต็ม, เลขทศนิยม เป็นต้น
2. ข้อมูลชนิดข้อความ (String) เป็นการนำข้อมูลเบื้องต้นชนิดตัวอักษรมาเรียงต่อกัน

3. ข้อมูลชนิดโครงสร้าง(Structure Type)เป็นชนิดข้อมูลเกิดจากการนำเอาชนิดข้อมูลเบื้องต้นมาประกอบกันเป็นโครงสร้างเพื่อเก็บข้อมูลเป็นชุดๆ เช่น อาร์เรย์
4. ข้อมูลชนิดอ้างอิง(Pointer)เป็นข้อมูลที่เราใช้ในการอ้างอิงกับหน่วยความจำในคอมพิวเตอร์ซึ่งจะใช้บอกตำแหน่งข้อมูล

3.2.2 การประกาศตัวแปร

ในการใช้งานข้อมูลนั้นต้องมีตัวแปลมารองรับ ตัวแปรจะถูกใช้งานได้ก็ต่อเมื่อ มีการประกาศตัวแปรเสียก่อน ซึ่งก็เหมือนกับต้องมาลงทะเบียนให้ Delphi รู้จักเสียก่อนจึงจะใช้งานได้ ซึ่งมีรูปแบบของการประกาศตัวแปรดังนี้

VAR ชื่อตัวแปร : ชนิดตัวแปร;

สำหรับการประกาศตัวแปร เรามักจะประกาศไว้ต้นโปรแกรม หรือตอนต้นของโปรแกรมย่อย ซึ่งจะประกาศเป็นตัวแปรชนิดเดียวหรือหลายชนิดพร้อมๆ กันก็ได้ โดยมีกฎเกณฑ์ดังนี้

1. ตัวแปรต้องขึ้นต้นด้วยตัวอักษรหรือเส้นขีดล่างเท่านั้น(_)แล้วตามด้วยตัวอักษรตัวเลขเครื่องหมาย
2. ห้ามเว้นวรรคในชื่อตัวแปร และในตัวแปลห้ามมีเครื่องหมายต่างๆ เช่น $, %, *, @, +, - เป็นต้น
3. ตัวแปรหนึ่งตัวความยาวไม่เกิน 63 ตัวอักษร
4. ตัวอักษรตัวพิมพ์ใหญ่ พิมพ์เล็กมีความต่างกัน เพราะฉะนั้นตัวแปร Name, name, NAME จึงต่างกัน
5. ตัวแปรที่ตั้งห้ามตรงกับคำที่สงวนไว้ เช่น begin, end, if เป็นต้น

ข้อมูลชนิด Array

อาร์เรย์ เป็นรูปแบบการจัดเก็บข้อมูลแบบเป็นชุด ซึ่งข้อมูลแต่ละตัวในอาร์เรย์ต้องเป็นข้อมูลชนิดเดียวกัน โดยจะใช้ Index ในการอ้างอิงถึงข้อมูลแต่ละตัวในอาร์เรย์ รูปแบบมีดังนี้

VAR ชื่อตัวแปรชนิดอาร์เรย์ : array [indexเริ่มต้น.....indexสุดท้าย] of ชนิดข้อมูลที่เก็บในอาร์เรย์;

Type ชนิดของอาร์เรย์ = array [indexเริ่มต้น.....indexสุดท้าย] of ชนิดข้อมูลที่เก็บในอาร์เรย์;

รู้จักกับ Dynamic Array

ในบางครั้งการทำงานกับอาร์เรย์เราอาจพบปัญหาต่อไปนี้

1. ยากที่จะกำหนดขนาดของอาร์เรย์ เช่น การสร้างอาร์เรย์ที่เก็บรายชื่อพนักงานบริษัท แต่ไม่ทราบว่ามีพนักงานกี่คน แถมมีโอกาสที่จะเพิ่มหรือลดได้ตลอดเวลา
2. การประกาศอาร์เรย์ขนาดใหญ่จากข้อหนึ่ง อาจทำให้เราแก้ปัญหาโดยประกาศอาร์เรย์ที่ใหญ่ๆไปเลย ซึ่งทำให้มีการใช้พื้นที่ไม่คุ้มค่า เพราะมีหน่วยความจำที่จองไว้มากแต่ไม่ได้ใช้
3. ประกาศอาร์เรย์ขนาดใหญ่เอาไว้ แต่ไม่ได้ถูกใช้งานทันที แม้บางครั้งต้องใช้อาร์เรย์ขนาดใหญ่ก็จริงแต่ว่ากว่าจะได้ใช้งานก็อาจจะเป็นการทำงานในช่วงท้ายๆของโปรแกรม ทำให้หน่วยความจำที่จองไว้ตั้งแต่ต้นไม่ได้นำมาใช้งานทันที เป็นการสิ้นเปลืองทรัพยากรณ์ของระบบโดยไม่จำเป็น

จากปัญหาข้างต้นทำให้มีการคิดไดนามิกอาร์เรย์ขึ้นมา โดยจะเป็นอาร์เรย์ที่มีคุณสมบัติพิเศษคือไม่จำเป็นต้องระบุขนาดในตอนประกาศ แต่ไม่สามารถเปลี่ยนขนาดอาร์เรย์ได้ในขณะที่เรายังให้แอพพลิเคชั่นทำงาน ซึ่งถ้าจะเปลี่ยนแปลงขนาด เราจะต้องเขียนโค๊ดเอง

ในการใช้งานเราจะประกาศตัวแปรแบบอาร์เรย์เอาไว้โดยไม่ระบุขนาด และจะใช้การระบุขนาดของอาร์เรย์ด้วยฟังก์ชัน SetLength ส่วน Index ของไดนามิกอาร์เรย์จะเริ่มนับตั้งแต่ 0 ขึ้นไป

3.2.3 การใช้งานค่าคงที่

ในบางครั้งเราอาจจะต้องใช้ข้อมูลค่าหนึ่งซึ่งมีค่าเหมือนเดิมไปตลอดโดยไม่เปลี่ยนแปลง เราเรียกข้อมูลตัวนั้นว่าค่าคงที่(Constant)

ค่าคงที่นั้นสามารถกำหนดให้เป็นข้อมูลชนิดใดก็ได้ ไม่จำเป็นต้องเป็นตัวเลขหรือตัวอักษรเท่านั้นซึ่งมีรูปแบบการกำหนดค่าดังนี้

Const ชื่อค่าคงที่ = ค่าคงที่ที่กำหนด;

3.2.4 ตัวดำเนินการเปรียบเทียบ

ตัวดำเนินการเปรียบเทียบหรือ Comparison Operator นี้ทำหน้าที่เปรียบเทียบค่าของตัวแปรสองค่า ในลักษณะมากกว่าหรือน้อยกว่าหรือเท่ากับ โดยจะให้ค่าจริง (True) หรือเท็จ (False) โดยใช้ตัวดำเนินการ <, >, =, >=, <= และยังมีตัวดำเนินการเปรียบเทียบทางตรรกะ และ (and), หรือ (or) เป็นตัวช่วยกำหนดการเปรียบเทียบระหว่างสองตรรกะหรือมากกว่าของโปรแกรม

3.2.5 การควบคุมทิศทางการทำงานของโปรแกรม

การควบคุมทิศทางของโปรแกรมเป็นการเรียนแบบการทำงานของมนุษย์ เพราะในสถานการณ์ต่างๆสมองของเราจะต้องมีการคิดเพื่อตัดสินทำงานในแบบต่างๆ

ในการเขียนโปรแกรมก็คือ การจำลองตัวเราลงไปเพื่อจัดการกับสถานการณ์ต่างๆ ซึ่งเราต้องมีข้อมูลประกอบการพิจารณา ถ้าข้อมูลบอกเราอย่างหนึ่งเราก็ต้องทำงานแบบหนึ่ง แต่ถ้าบอกข้อมูลอีกอย่างเราก็ต้องทำงานอีกแบบหนึ่งที่แตกต่างกัน ซึ่งการที่ต้องจัดการกับสถานการณ์ต่างๆนี้เรามักเรียกมันว่า การควบคุมทิศทางของโปรแกรม(Control Flow)

การควบคุมทิศทางของโปรแกรม มีอยู่ด้วยกัน 2 รูปแบบได้แก่

1. การตัดสินใจ คือการเลือกจากตัวเลือกที่มีอยู่
2. การทำงานแบบวนซ้ำ คือวนซ้ำจนกว่าจะครบจำนวนหรือถูกกำหนดให้หยุดการวนซ้ำ

การใช้งาน If…Then…Else

จะเป็นการทำงานที่มีการเลือกหนึ่งตัวจากสองตัวเลือก ซึ่งเราจะต้องมีตัวแปรตัวหนึ่งถูกตรวจสอบว่า มีค่าเป็นจริงหรือเป็นเท็จ ถ้าเป็นจริง(True) ก็จะต้องเลือกตัวเลือกแรกซึ่งอยู่หลังคำว่า Then แต่ถ้าตรวจสอบว่าเป็นเท็จ ก็จะต้องเลือกตัวเลือกอีกตัวที่อยู่หลังคำว่า Else รูปแบบการใช้งานมีดังนี้

If……then

ชุดคำสั่งเมื่อเงื่อนนั้นตรวจสอบแล้วเป็น True

Else

ชุดคำสั่งเมื่อเงื่อนไขนั้นตรวจสอบแล้วว่าเป็น Flse

การใช้งาน Case…of

เราจะใช้ Case…of ในการเลือกหนึ่งตัวจากตัวเลือกที่มีหลายตัว โดยเราจะใช้ตัวแปลหนึ่งตัวตรวจสอบว่า ตรงกับตัวเลือกใด โดยที่ตัวเลือกแต่ละตัวเราก็กำหนดให้มีการทำงานที่ต่างด้วย รูปแบบมีดังนี้

Case……..of

ชุดคำสั่งเมื่อตรวจสอบค่าตัวแปรตรงกับค่าที่ 1

.

.

.

Else

ชุดคำสั่งเมื่อค่าตัวแปรตรวจสอบไม่ตรงกับค่าใดๆ

End;

การใช้งาน While...do

While...do เป็นการทำงานที่เราต้องการให้มีการวนซ้ำไปเรื่อยๆ โดยทุกๆ ครั้งที่มีการวนซ้ำใหม่ ให้มีการตรวจสอบเงื่อนไขการวนซ้ำก่อนทุกครั้ง ถ้าเงื่อนไขเป็นจริง ก็จะวนซ้ำต่อไป แต่ถ้าเป็นเท็จก็ให้หยุดการวนซ้ำ มีรูปแบบการทำงานดังนี้

While.....do

.....

การใช้งาน repeat...until

การทำงานของ repeat...until นั้นจะแตกต่างจาก while...do ตรงที่การทำงานหยุดการวนซ้ำจะหยุดก็ต่อเมื่อตรวจสอบแล้วว่าเป็นจริงจึงจะหยุด มีรูปแบบการใช้งานดังนี้

Repeat

.....

Until.....

การใช้งาน for...to...do

เมื่อเราจะใช้จะใช้แอพพลิเคชั่นทำงานซ้ำด้วยจำนวนรอบที่แน่นอน ซึ่งจะใช้ตัวแปรหนึ่งตัวทำหน้าที่ในการนับว่าทำซ้ำครบรอบหรือไม่ โดยแต่ละครั้งของการวนรอบตัวแปรที่นับรอบนั้นจะเพิ่มค่าทีละหนึ่ง และเพิ่มไปเรื่อยๆ จนกว่าจะมากกว่าค่าสุดท้าย มีรูปแบบการใช้งานดังนี้

For [ตัวแปรนับรอบ] := [ค่าเริ่มต้น] to [ค่าสุดท้าย] do

.....

การใช้งาน for...downto...do

สามารถลดค่าตัวแปรในการนับลงทีละ 1 โดยใช้งานดังนี้

For[ตัวแปรนับรอบ] := [ค่าเริ่มต้น] downto [ค่าสุดท้าย] do

.....

3.2.6 สร้างและใช้งาน Procedure

โพรซีเจอร์ คือ โปรแกรมย่อยที่ทำงานเสร็จแล้วจะไม่มีการส่งคืนค่าใดๆกลับมายังผู้ที่เรียกใช้งานโพรซีเจอร์นั้น เราจึงใช้โพรซีเจอร์ในงานที่ไม่สนใจ หรือไม่ต้องการผลลัพธ์ที่ได้จากการทำงานของโพรซีเจอร์ เพียงแค่สั่งให้โพรซีเจอร์ทำงานเท่านั้นก็พอแล้ว

รูปแบบการใช้งาน

Procedure [ชื่อโพรซีเจอร์];

Begin

.....

End;

3.2.7 สร้างและใช้งาน Function

ฟังก์ชัน คือโปรแกรมย่อยที่ถูกเรียกใช้งาน และภายหลังที่ทำงานเสร็จแล้วจะคืนค่ากลับมาให้กับผู้ที่เรียกฟังก์ชันนั้นได้นำไปใช้งานต่อไป ดังนั้นการใช้งานฟังก์ชันจึงมักจะสนใจที่ผลการทำงานที่ได้รับกลับมา ซึ่งหมายถึงการนำค่านั้นไปใช้ในคำสั่งถัดไปหลังจากเรียกใช้ฟังก์ชันนั้น

Function [ชื่อฟังก์ชัน]:[ชนิดของตัวแปร];

Begin

[ชุดคำสั่งในโปรแกรมย่อยแบบ Function]

[ชื่อฟังก์ชัน] := [ค่าที่คืนกลับมา]

End;

บทที่ 4
การใช้งานพอร์ตอนุกรม

4.1 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมก็คือการที่อุปกรณ์หรือ Devices สองตัวทำการติดต่อกันโดยสัญญาณออกไปหรือรับเข้ามามีลักษณะของขบวนสัญญาณที่เปลี่ยนแปลงสลับไปมาอย่างต่อเนื่อง ณ เวลาใดเวลาหนึ่งโดยมีรูปแบบการส่งที่แน่นอนเช่นการส่งรหัสมอร์ส การสื่อสารผ่าน RS232, RS485 ฯลฯ

4.1.1 ประโยชน์ของพอร์ตอนุกรม

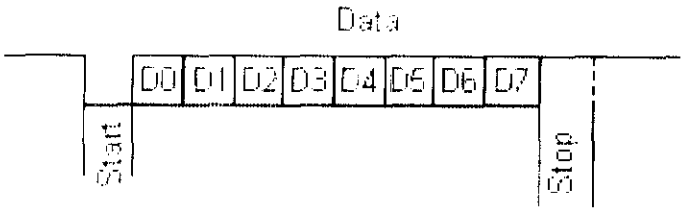
พอร์ตอนุกรมนั้นนอกจากใช่สื่อสารกับอุปกรณ์ภายนอกด้วยกันแล้วยังใช่เป็นอุปกรณ์สำหรับการดีบักโปรแกรม เช่น ใช่ดูค่าตัวแปรของโปรแกรม ดูสถานะของพอร์ตฯลฯ

4.1.2 การเพิ่มระยะทางของพอร์ตอนุกรม

สัญญาณอนุกรมที่ออกจากตัวไมโครคอนโทรลเลอร์จะเป็นสัญญาณ TTL ที่มีขนาด 0-5V ถ้าการสื่อสารไม่ไกลกันเช่นอยู่บนบอร์ดเดียวกันก็ไม่ต้องใช่วงจรใดๆช่วยแต่ถ้าตัวบอร์ดหรือวงจรอยู่ไกลกันก็ใช้ไอซีช่วยเช่น DS232 ซึ่งก็จะกลายเป็นการสื่อสารแบบ RS232 แต่ ก็มีข้อจำกัดด้านระยะทาง คือประมาณ 15-20 เมตรที่บอร์ดเร็ต 9600 และบอร์ดเร็ตจะแปลผกผันกับระยะทางด้วยและยังขึ้นกับไอซีไดร์เวอร์และชนิดของสายด้วย

4.1.3 ลักษณะสัญญาณของพอร์ตอนุกรม

สัญญาณของพอร์ตอนุกรมที่เป็นแบบที่นิยมใช้ทั่วไปคือ 1 บิต start, ข้อมูล 8 บิตและstop 1 บิตไม่มี parity บิต รวมทั้งหมดก็ 10 บิต จะมี สัญญาณตามรูปข้างล่าง

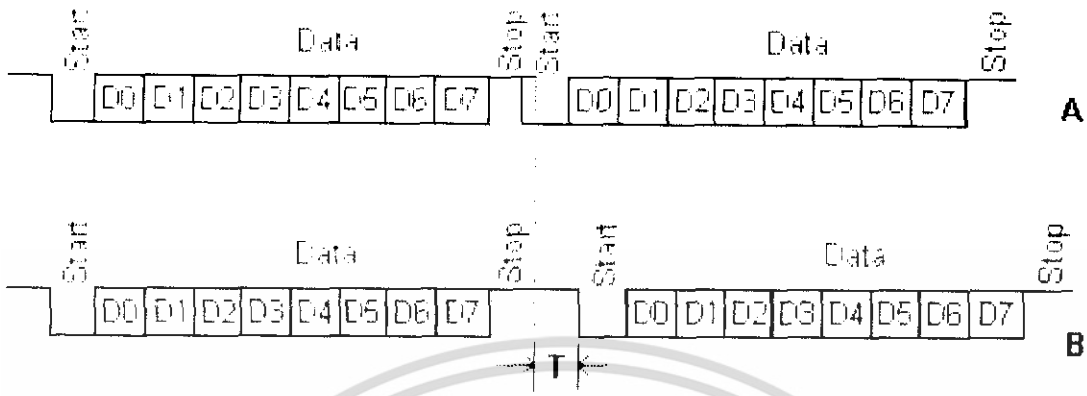


รูปที่ 4.1 แสดงรูปสัญญาณการส่งข้อมูลแบบ Serial Port

โดยบิตแรกจะเป็น start bit ซึ่งจะ active low ถัดไปอีก 8 บิตจะเป็นข้อมูลศูนย์หรือหนึ่งและบิตสุดท้ายจะเป็น stop บิตซึ่งจะ active high สำหรับความกว้างของแต่ละบิตก็คำนวณได้โดยอินเวิร์สค่าบอร์ดเร็ตหรือ 1/บอร์ดเร็ต เช่นบอร์ดเร็ต 9600 จะมีความกว้างของสัญญาณ 1 บิตเท่ากับ 104 uS เพราะฉะนั้น 10 บิตก็เท่ากับ 1.04 mS

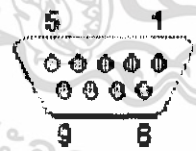
4.1.4 ปัญหาด้านรับไม่สามารถรับข้อมูลไม่ได้

จากค่าของเวลาอันนี้ทำให้เรารู้ว่าถ้าเราส่งข้อมูลติดๆกันโดยไม่คำนึงถึงเวลาของแต่ละบิตจะทำให้ด้านรับซึ่งอาจเป็นไมโครคอนโทรลเลอร์หรือ PC แยกข้อมูลไม่ออกเช่นหลังจากส่งบิต stop ออกไป แล้วทำการส่งข้อมูลชุดใหม่ออกไปทันทีโดยที่ เวลาของ stop บิตยังไม่ครบทำให้ด้านรับๆ ข้อมูลผิดพลาดได้ ดูรูปข้างล่างประกอบ



รูปที่ 4.2A ข้อมูลที่รับ-ส่งผิดพลาด รูปที่ 4.2B ข้อมูลที่รับ-ส่งถูกต้อง

จากรูปที่ 4.2 ถ้าการส่งข้อมูลเป็นแบบรูป A จะทำให้ด้านรับเกิดความผิดพลาดได้เนื่องจาก บิต stop ของข้อมูลชุดแรกยังไม่ครบเวลาแล้วส่ง start บิตของข้อมูลชุดถัดไปทับ การส่งที่ใช้ได้และมีเสถียรภาพจะเป็นตามรูป B โดยหน่วงเวลา มากกว่าหรือเท่ากับเวลาของแต่ละบิต



รูปที่ 4.3 แสดงขาของพอร์ตอนุกรมแบบ 9 ขา

Signal Name	DB-9 Pin
FG (Frame Ground)	-
TD (Transmit Data)	3
RD (Receive Data)	2
RTS (Request To Send)	7
CTS (Clear To Send)	8
SG (Signal Ground)	5
DSR (Data Set Ready)	6
CD (Carrier Detect)	1
RI(Ring Indicator)	9
DTR (Data Terminal Ready)	4

ตารางที่ 4.1 แสดงหมายเลขประจำขาของพอร์ตอนุกรมแบบ 9 ขา

4.1.5 การ Test ว่าพอร์ตอนุกรมของ PC เสียหรือไม่

การ Test ทำได้โดยช็อตขา 2 กับขา 3 เข้าด้วยกันแล้วเปิดโปรแกรม Hyper terminal จากนั้นลองพิมพ์อะไรลงไปก็ได้ตัวที่พิมพ์จะถูกส่งออกไปที่ขา Td แล้วรับเข้ามาที่ขา Rd ปรากฏให้เห็นที่หน้าจอ

* Rd กับ Td จะเป็นชื่อที่ใช้เรียกขารับและส่งข้อมูลของด้าน PC ส่วน Rx และ Tx หมายถึงขารับและส่งด้านของอุปกรณ์ภายนอกที่นำมาต่อกับ PC

4.2 การติดต่อและควบคุม Serial Port

4.2.1 พื้นฐานการสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมในคอมพิวเตอร์นั้นจะมีความเร็วในการสื่อสารช้ากว่าแบบขนาน ทั้งนี้เพราะว่าการเคลื่อนย้ายข้อมูลอนุกรมนั้นเป็นการส่งข้อมูลทีละ 1 บิต แต่พอร์ตขนานนั้นสามารถส่งข้อมูลได้ทีละหลายๆบิตพร้อมกันส่งผลให้การสื่อสารข้อมูลแบบอนุกรมมีความเร็วต่ำกว่าแบบขนาน แต่ว่าการส่งข้อมูลแบบอนุกรมนั้นมีข้อที่เหนือกว่าการส่งข้อมูลแบบขนานคือ การที่สามารถส่งข้อมูล

ได้ในระยะทางที่ไกลกว่าแบบขนาน อีกทั้งสายสัญญาณที่ใช้ยังมีน้อยกว่าการส่งข้อมูลแบบขนานอีกด้วย การสื่อสารแบบอนุกรมสามารถแบ่งออกได้เป็น 3 รูปแบบดังนี้

1. Simplex สามารถส่งข้อมูลได้อย่างเดียว เป็นการสื่อสารแบบทางเดียว

2. Half-Duplex สามารถส่งข้อมูลไปยังปลายทางและสามารถรับข้อมูลจากปลายทางได้แต่ไม่สามารถทำการส่งและรับข้อมูลได้ในเวลาเดียวกัน

3. Full-Duplex สามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน

นอกจากนี้แล้วยังสามารถแบ่งประเภทของการสื่อสารแบบอนุกรมตามลักษณะสัญญาณในการส่งได้อีก 2 แบบคือ

1. การสื่อสารแบบซิงโครนัส (Synchronous) สำหรับการสื่อสารแบบซิงโครนัสนี้จะใช้สัญญาณนาฬิกาควบคุมการส่งสัญญาณ เช่น สายคีย์บอร์ดคอมพิวเตอร์ โดยจะมีสายสัญญาณเส้นหนึ่งเป็นสายสัญญาณนาฬิกา ส่วนอีกเส้นหนึ่งเป็นสายของข้อมูล (และจะมีสายกราวด์ด้วย)

สำหรับการสื่อสารแบบซิงโครนัสนี้เหมาะสำหรับการทำงานในระยะใกล้ ข้อมูลที่จะส่งมีไม่มากนัก เพราะถ้าระยะทางไกลขึ้นจะทำให้สัญญาณนาฬิกามีปัญหา อีกทั้งต้องมีสายหลายเส้นทำให้สิ้นเปลืองมาก

2. การสื่อสารแบบอะซิงโครนัส (Asynchronous) สำหรับการสื่อสารแบบอะซิงโครนัสนั้นจะใช้สายข้อมูลเพียงตัวเดียว แต่จะใช้รูปแบบการส่งข้อมูล หรือ Bit Pattern เป็นตัวกำหนดว่าส่วนไหนเป็นส่วนเริ่มต้นข้อมูล, ส่วนไหนเป็นตัวข้อมูล, ส่วนไหนจะเป็นส่วนตรวจสอบความถูกต้องของข้อมูล และส่วนไหนเป็นส่วนปิดท้ายของข้อมูล โดยต้องกำหนดให้สัญญาณนาฬิกาเท่ากันทั้งภาคส่ง และภาครับ ซึ่งจะมีอุปกรณ์พิเศษที่ชื่อว่า UART หรือ Universal Asynchronous Receiver / Transmitter คอยควบคุมการรับ และส่งข้อมูล

UART ที่นิยมใช้งานในคอมพิวเตอร์คือ ชิป 5250 ซึ่งให้ความเร็วสูงสุด 57.6 Kbps (หรือ 56 K) ขณะที่ปัจจุบันเริ่มใช้งานกับชิป 16450 ซึ่งสนับสนุน 8250

4.2.2 รู้จักกับมาตรฐาน RS-232 C

มาตรฐาน RS-232 C เป็นมาตรฐานที่ได้รับการออกแบบมาเพื่อที่จะทำให้อุปกรณ์ต่อพ่วงจากผู้ผลิตต่างกันสามารถทำงานร่วมกันได้ มาตรฐานหลายชนิดได้รับการออกแบบขึ้นมา แต่มาตรฐานที่ได้รับความนิยมและใช้กันกว้างขวางมากที่สุดคือ มาตรฐาน RS-232C ซึ่งถูกประกาศใช้ในปี 1969 โดย

สมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association : EIA) ในยุคแรกๆการอินเตอร์เฟสแบบ RS-232C ถูกออกแบบสำหรับเชื่อมต่อเทอร์มินอล (DTE: Data Terminal Equipment) กับโมเด็ม (DCE: Data Communication Equipment) ทั้งนี้ก็เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลบนสายเดียวกัน

มาตรฐาน RS-232 C ได้แบ่งอุปกรณ์ออกเป็น 2 ประเภท ซึ่งอุปกรณ์ทั้งสองประเภทนี้คือ

1. อุปกรณ์ DTE (Data Terminal Equipment) เป็นอุปกรณ์สำหรับส่งข้อมูล (Output)

2. อุปกรณ์ DCE (Data Communication Equipment) เป็นอุปกรณ์สำหรับรับข้อมูล (Input)

ตามมาตรฐาน RS-232C แล้วคอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งคอนเน็กเตอร์ที่นิยมใช้กันอยู่จะเป็นชนิด D-Type แบบ 9 ขา และแบบ 25 ขา โดยจะติดตั้งอยู่หลังเครื่องคอมพิวเตอร์ ระดับแรงดันจะมีค่าระหว่าง -3V ถึง -15 V

สำหรับลอจิก High และลอจิก Low จะมีระดับแรงดันระหว่าง +3 V ถึง +15V สามารถรับส่งข้อมูลได้ที่ความยาวของสัญญาณสูงสุด 50 ฟุต หรือ 150 เมตร

4.2.3 องค์ประกอบของการับส่งข้อมูลแบบอนุกรม

การสื่อสารแบบอนุกรมที่นิยมใช้กับคอมพิวเตอร์นั้น เป็นการสื่อสารข้อมูลแบบอะซิงโครนัส นั่นคือ ต้องใช้สายสัญญาณเส้นเดียวทำหน้าที่ทั้งส่งส่วนที่เป็นข้อมูล และส่วนที่ใช้ควบคุมการส่งข้อมูล ดังนั้นข้อมูลที่อ่านได้แต่ละบิตจากการส่งแบบอนุกรม จึงต้องแยกแยะว่าใช้สำหรับวัตถุประสงค์ใด โดยเราสามารถแบ่งได้เป็น 4 ส่วน คือ

1. Start Bit ขนาด 1 บิต
2. บิตข้อมูล (Data Character) ขนาด 7 หรือ 8 บิต
3. Parity Bit ขนาด 1 บิต
4. Stop Bit ขนาด 1 หรือ 2 บิต

แต่ละตัวอักษรที่ถูกส่งออกไปเป็นกลุ่มจะประกอบไปด้วยบิตเริ่มต้น บิตข้อมูล บิตพาริตี้ (จะมีหรือไม่มีก็ได้) และบิตจบ โดยเราพอจะสรุปหน้าที่ของแต่ละส่วนได้ดังนี้

Start Bit หรือบิตเริ่มต้น จะใส่ที่จุดเริ่มต้นเสมอ เพื่อเตือนอุปกรณ์ฝ่ายรับว่าข้อมูลกำลังจะมาถึง

Data Character หรือบิตข้อมูล การส่งบิตข้อมูลจะส่งเป็นกลุ่ม ๆ โดยทั่วไปจะส่งเป็น 7 บิต หรือ 8 บิต ซึ่งเพียงพอสำหรับการส่ง Ascii Word

Parity Bit หรือบิตพาริตี้ ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่ง เราจะใส่บิตพาริตี้เข้าไป แต่ทั้งตัวรับและตัวส่งจะต้องรู้กันว่าใช้พาริตี้แบบไหนในการส่งข้อมูลซึ่งหลักการในการกำหนดบิตพาริตี้มีหลายแบบดังนี้

พาริตี้คู่ (Even Parity) ค่าของบิตพาริตี้เมื่อรวมกับทุก ๆ บิต ของข้อมูลแล้ว จะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคู่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด 3 ตัว ดังนั้นบิตพาริตี้จะเป็น 0

พาริตี้คู่ (Odd Patity) ค่าของบิตพาริตี้นี้เมื่อรวมกับทุก ๆ บิตของข้อมูลแล้ว จะต้องมีจำนวนบิตที่เป็นเลข 1 เป็นเลขคี่ ตัวอย่างเช่น ข้อมูล 1000101 มีเลข 1 ทั้งหมด ดังนั้นบิตพาริตี้จึงเป็น 1

ไม่มีพาริตี้ (None) ถ้าตั้งบิตพาริตี้เป็น None ทั้งภาครับและภาคส่งจะไม่มีการตรวจสอบบิตพาริตี้ Stop Bit หรือบิตจบ เป็นบิตที่ส่งมาปิดท้ายข้อมูล

4.2.4 อัตราเร็วในการรับส่งข้อมูลแบบอนุกรม

การที่อุปกรณ์ 2 อย่างจะติดต่อสื่อสารกันได้นั้น จะต้องทำงานด้วยอัตราเร็วเท่ากัน ซึ่งอัตราเร็วในการสื่อสารแบบอะซิงโครนัสคือ ค่าบอดเรต (Baud Rate) มีหน่วยเป็นบิตต่อวินาที ซึ่งค่าอัตราเร็วในการสื่อสารแบบอนุกรมสำหรับมาตรฐาน RS-232C นั้นมีใช้ดังนี้ 110, 150, 300, 600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที

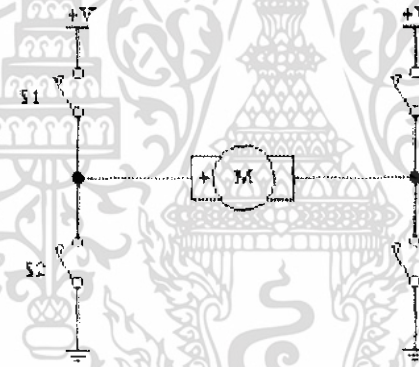
บทที่ 5

การขับมอเตอร์ (Drive Motor)

DC motor

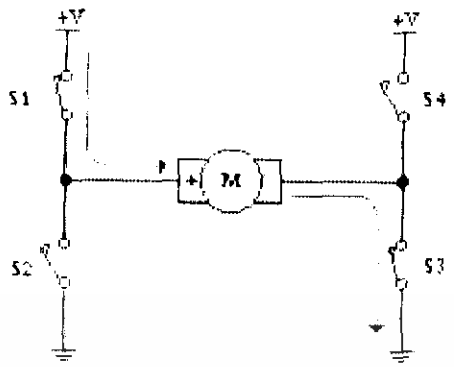
จะมีการทำงานเพียง 3 Step เท่านั้น คือ หมุนตามเข็มนาฬิกา หมุนทวนเข็มนาฬิกา และหยุดนิ่ง เท่านั้น ซึ่งอัตราความเร็วในการหมุนและแรงบิดจะขึ้นอยู่กับกระแสและแรงดันก็จะจ่ายให้มันถ้าหาก แรงบิดของมอเตอร์ไม่เพียงพอต่อการทำงาน เราก็สามารถที่จะทำการทดรอบของการหมุนลงก็จะได้ แรงบิดเพิ่มมากขึ้นหรือเราสามารถควบคุมความเร็วของมอเตอร์ได้โดยการจ่ายพลังงานในลักษณะ Pulse Width Modulation (PWM)

5.1 การควบคุมทิศทางการหมุนของ DC Motor หลักการทำหลักการทำงานของวงจร H-Bridge Switching



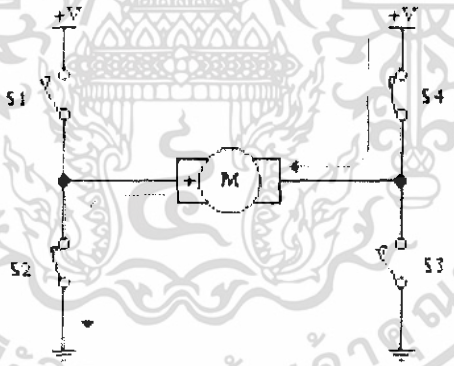
รูปที่ 5.1 แสดงสภาวะเริ่มต้นการทำงานของวงจร

หลักการของวงจรนั้นจะประกอบไปด้วยสวิทช์ 4 ตัว คือ S1, S2, S3, S4 ซึ่งในรูปตัวอย่างจะใช้ DC Motor เป็น Load ในสภาวะเริ่มต้น สวิทช์ทุกตัวOff อยู่ ก็จะไม่มีอะไรเกิดขึ้น เพราะไม่มีกระไหล เข้าสู่มอเตอร์ (รูปที่ 5.1)



รูปที่ 5.2 แสดงสภาวะเมื่อ on S1 กับ S3 พร้อมกัน

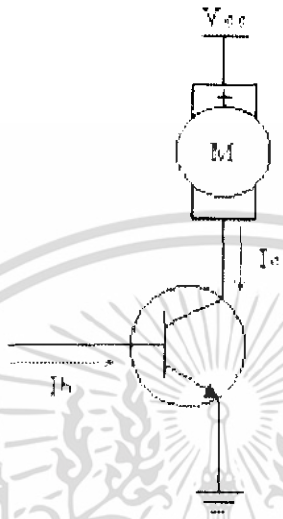
และเมื่อ ON สวิทช์ S1 และ S3 พร้อมกัน(รูปที่ 5.2) จะเป็นการเชื่อมวงจร ทำให้มีกระแสไฟฟ้า ไหลผ่านมอเตอร์ จากขั้วบวกของมอเตอร์ ไปยังขั้วลบของมอเตอร์ จึงทำให้มอเตอร์สามารถหมุนได้ ในทิศทาง Forward (จะหมุนแบบตามเข็มนาฬิกา หรือทวนเข็มนาฬิกา ขึ้นอยู่กับลักษณะของการพันขดลวดภายในมอเตอร์)



รูปที่ 5.3 แสดงสภาวะเมื่อ on S2 กับ S4 พร้อมกัน

และเมื่อ ON สวิทช์ S2 และ S4 พร้อมกัน (รูปที่ 5.3) ก็จะเป็นการเชื่อมวงจรและทำให้เกิดกระแสไฟฟ้าไหลผ่านมอเตอร์ จากขั้วลบของมอเตอร์ ไปยังขั้วบวกของมอเตอร์จึงทำให้มอเตอร์หมุนได้ และเป็นการหมุนในทิศทางReverse (กลับทิศทางกับกรณีแรก)

5.2 การควบคุมทิศทางการหมุนของ DC Motor สร้างวงจร H-Bridge Switching จาก Transistor



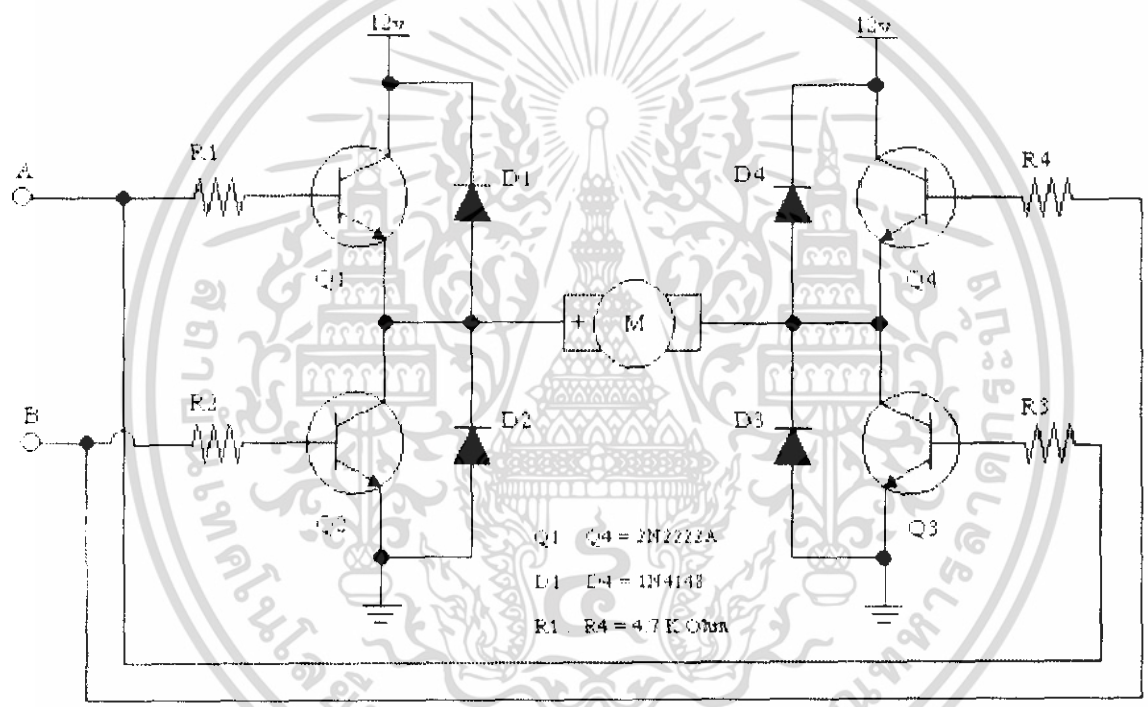
รูปที่ 5.4 แสดงวงจรใช้ทรานซิสเตอร์เป็นตัวสวิทช์ควบคุมมอเตอร์

ทรานซิสเตอร์เป็นอุปกรณ์สารกึ่งตัวนำ (Semiconductor device) ที่เราสามารถนำคุณสมบัติของการ Cutoff และการ Saturation มาประยุกต์ใช้เป็นสวิทช์ได้และที่สำคัญมันเป็นสวิทช์อิเล็กทรอนิกส์ ที่เราสามารถควบการปิด/เปิด ได้ จากรูปที่ 5.4 การนำทรานซิสเตอร์มาเป็นสวิทช์ควบคุมมอเตอร์ เมื่อเราป้อนกระแส I_B ด้วยปริมาณที่มากพอ ก็จะทำให้ทรานซิสเตอร์ทำงานจะทำให้กระแส I_C ไหล ผ่านมอเตอร์ได้ (กระแส I_B มากเพียงพอที่จะทำให้ทรานซิสเตอร์อยู่ในสภาวะ"อิ่มตัว" ได้)

ในสภาวะอิ่มตัว (Saturation mode) ทรานซิสเตอร์เหมือนกับสวิทช์ปิดวงจร ค่าความต้านทานระหว่างขา C และขา E จะมีค่าเข้าใกล้ศูนย์ กระแส I_C ที่จะไหลมีค่าเข้าใกล้ Ic (max)

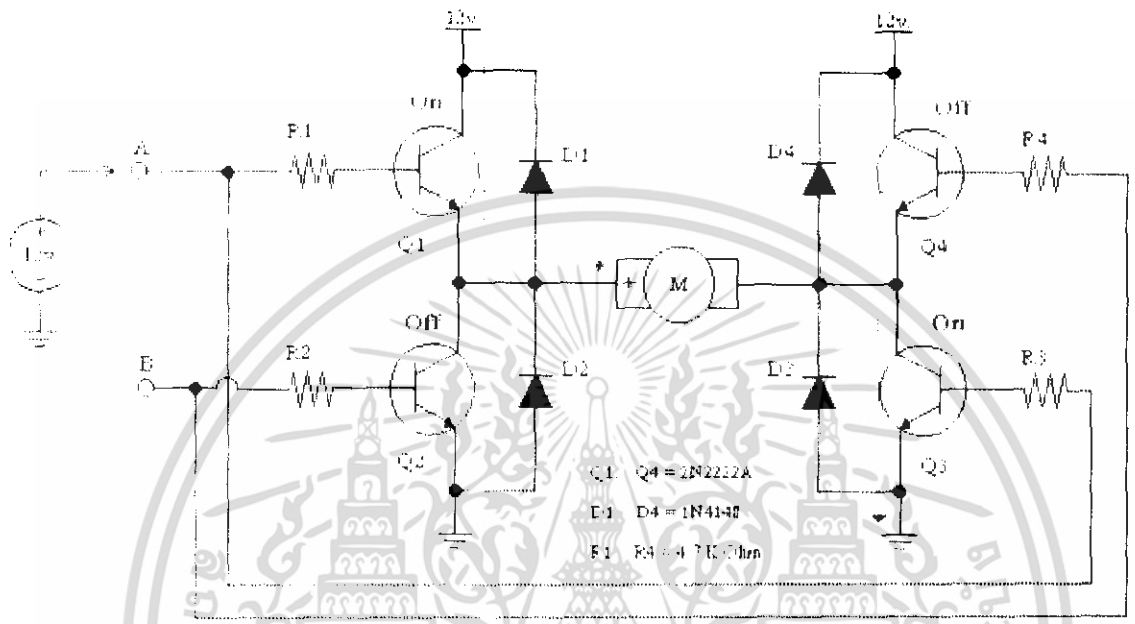
ในสภาวะคัทออฟ (Cutoff mode) นี้จะเกิดขึ้นเมื่อหยุดจ่ายกระแส I_B (I_B=0) ทรานซิสเตอร์จะทำงานเหมือนกับสวิทช์เปิดวงจร ค่าความตานทานระหว่างขา C และขา E จะมีค่าเป็นอนันต์ กระแส Ic จะมีค่าเข้าใกล้ศูนย์

ข้อดีของการนำทรานซิสเตอร์มาประยุกต์ใช้งานเป็นสวิทช์คือการเปิด/ปิดสวิทช์ได้นับล้านครั้งต่อวินาที (ความเร็วในการตอบสนองมีหน่วยเป็น ns) และไม่ทำให้เกิดปัญหารบกวนจากสนามแม่เหล็ก



รูปที่ 5.5 แสดงรูปวงจรในสภาวะเริ่มต้นการทำงานของวงจร

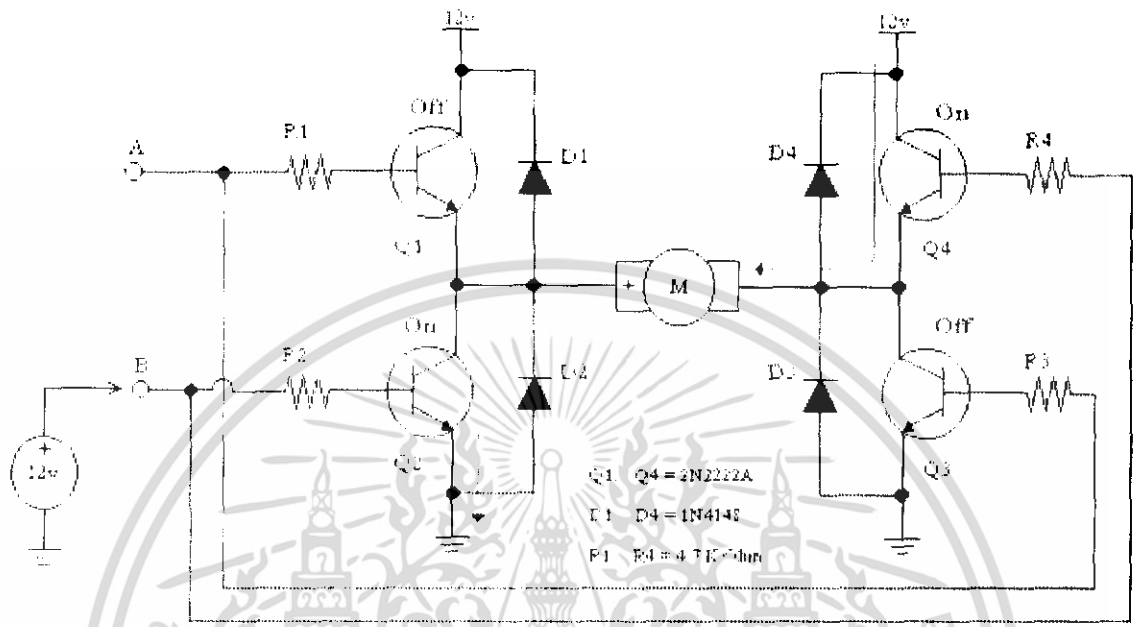
กรณีที่ Q1 และ Q3 ทำงาน



รูปที่ 5.6 แสดงสภาวะของวงจรขณะที่ Q1 และ Q3 ทำงาน

เมื่อมีการจ่ายแรงดัน 12 V.เข้าที่จุด A ทำให้มีกระแสไหลผ่าน R1 เข้าสู่ขา base ของ Q1 และมีกระแสไหลผ่าน R13 เข้าสู่ขา base ของ Q3 ทำให้ Q1 และ Q3 ทำงาน (On) เปรียบเสมือน สวิทช์วงจร ส่งผลให้มีกระแสไหลจากแหล่งจ่าย (12 v.) ผ่านขา Collector และ Emitter ของ Q1 ผ่านเข้าสู่ขั้วบวก (+) ของมอเตอร์ ผ่านไปยังขา Collector และ Emitter ของ Q3 ทำให้มีกระแสไหลผ่านมอเตอร์ในทิศทางบวก และครบวงจร จึงทำให้มอเตอร์สามารถหมุน ในทิศทาง Forward ได้

กรณีที่ Q2 และ Q4 ทำงาน



รูปที่ 5.7 แสดงสภาวะของวงจรขณะที่ Q2 และ Q4 ทำงาน

เมื่อมีการจ่ายแรงดัน 12 V.เข้าที่จุด B ทำให้กระแสไหลผ่าน R2 เข้าสู่ขา base ของ Q2 และมีกระแสไหลผ่าน R4 เข้าสู่ขา base ของ Q4 ทำให้ Q2 และ Q4 ทำงาน (On) เปรียบเสมือนสวิทช์ปิดวงจร ส่งผลให้มีกระแสไหลจากแหล่งจ่าย (12v.) ผ่านขา Collector และ Emitter ของ Q4 ผ่านเข้าสู่ขั้วลบ (-) ของมอเตอร์ ผ่านไปยังขา Collector และ Emitter ของ Q2 ทำให้มีกระแสไหลผ่านมอเตอร์ ในทิศทาง กท และครบวงจร จึงทำให้มอเตอร์สามารถหมุนในทิศทาง Reward ได้

บทที่ 6
อิมเมจและการประยุกต์ใช้งาน

6.1 การเปลี่ยนแปลงรูปแบบทางเลขาคณิต

การเปลี่ยนแปลงรูปแบบทางเลขาคณิต คือ กระบวนการเปลี่ยนช่องว่างท่ามกลางวัตถุในรูป หรือเปลี่ยนรูปทรงทางเลขาคณิตของวัตถุในรูปภาพ

ขั้นตอนการคิด

เป็นการจำกัดความของรูปภาพที่เป็น f(x,y) ที่ (x,y) คือจุด (0,0) และ f(x,y) คือระดับสีเทาที่จุด (0,0) เพื่อที่จะแสดง การการเปลี่ยนแปลงรูปแบบทางเลขาคณิต 2 ข้อกำหนดที่จำเป็นคือ

1. การเปลี่ยนแปลงช่องว่าง
2. การสอดแทรกระดับสีเทา

การเปลี่ยนแปลงเชิงเส้นแบบมีช่องว่าง

คำจำกัดความ

การแมปปิ้ง F: E2 - > E2 คือ การแปลงเชิงเส้นของ E2 ในตัวของมัน สำหรับทุกเวคเตอร์ X และ Y และสำหรับทุกสเกลล่า C และ D

F(cX−dY) = cF(x)+dF(Y)

ถ้า F และ G คือ เส้นตรง 2 เส้นที่ได้รับการเปลี่ยนแปลง

GF(x) = G(F(x))

ชั้นย่อยๆ

การหมุนในแบบตั้งฉากและการแปรความยาว มุม และ เส้นขนาน ที่ถูกกำหนดไว้ ความคล้ายคลึงของการหมุน การสเกล และการแปรความ มุมและเส้นขนานที่ถูกกำหนดไว้ การเปลี่ยนแปลงแบบเลียนแบบ เช่น การเฉือน การหมุน การสเกล และ การแปรความเส้นขนานที่ถูกกำหนดไว้

การแปรความ คือ ย้ายตำแหน่งของวัตถุจากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่งโดยมีลักษณะการเคลื่อนที่เป็นแนวเส้นตรง และไม่ทำให้รูปร่างของวัตถุมีการเปลี่ยนแปลงไปโดยจุด (x,y) เมื่อต้องการย้ายไปยัง (x',y') ทำได้โดยการบวกค่าระยะทางที่ต้องการย้ายไปตามสมการ

$$x' = x + Tx$$

$$y' = y + Ty$$

เมื่อนำมาเขียนให้อยู่ในรูปของเมตริกซ์จะได้ดังนี้

การสเกลลิ่ง คือการเปลี่ยนขนาดของวัตถุ ทำได้โดยการคูณ scaling factor เข้าไปในสมการ

$$x' = sx \times x$$

$$y' = sy \times y$$

นำมาเขียนในรูปเมตริกซ์จะได้ดังนี้

โดยเมื่อ Sx หรือ Sy มีค่ามากกว่า 1 วัตถุจะมีขนาดใหญ่ขึ้น
Sx หรือ Sy มีค่าน้อยกว่า 1 แต่มากกว่า 0 วัตถุจะมีขนาดเล็กลง
Sx หรือ Sy มีค่าน้อยกว่า 0 วัตถุจะเกิดการสะท้อน

การหมุน

การหมุน คือ การย้ายตำแหน่งของจุด โดยมีเส้นทางของการย้ายเป็นแนววงกลมรอบจุดกำเนิดตามสมการ

$$x' = x\cos A - y\sin A$$

$$y' = x\sin A + y\cos A$$

นำมาเขียนในรูปเมตริกซ์จะได้ดังนี้

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos A & \sin A \\ \sin A & \cos A \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

การเฉือน

การเฉือน คือ การทำให้วัตถุมีรูปร่างที่ผิดเพี้ยนไปจากเดิม ในการคำนวณหาตำแหน่งใหม่ของจุดทำได้ดังสมการ

1) เมื่อทำการ shear ในทิศทาง x

$$x' = x + Shx * y$$

$$y' = y$$

2) เมื่อทำการ shear ในทิศทาง y

$$y' = y + Shy * x$$

$$x' = x$$

เมื่อนำมาเขียนให้อยู่ในรูปของเมตริกซ์จะได้ดังนี้

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & Shx \\ Shy & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

เมื่อต้องการทำ transformation หลายๆรูปแบบ เราสามารถคำนวณหาตำแหน่งของจุดใหม่ได้ดังนี้

$$\begin{bmatrix} x' & \cos\theta & \sin\theta \\ y' & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} Sx & 0 & 1 & Shx \\ 0 & Sy & Shy & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} Zx \\ Zy \end{bmatrix}$$

พิจาณาจากสมการจะเห็นว่า translation matrix ไม่สามารถนำเข้าไปรวมกับเมตริกซ์อื่นได้ จึงได้มีการใช้ homogeneous coordinates โดย homogeneous coordinates จะต้องใช้ค่า 3 ค่า คือ (xh,yh,z) โดย xh = wh และ yh = wy โดยในระนาบ 2 มิตินั้นให้ w=1 และสามารถเขียน transformation matrix ต่างๆ ได้ดังนี้

$$P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad T = \begin{bmatrix} 1 & 0 & Zx \\ 0 & 1 & Zy \\ 0 & 0 & 1 \end{bmatrix} \qquad S = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Point Matrix Translation Matrix Scaling

Shear Matrix

$$Sh = \begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation Matrix

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Forward Mapping

Forward Mapping คือการคำนวณหาพิกัดของจุดในภาพใหม่โดยคำนวณจากพิกัดของจุดในภาพเดิม

Backward Mapping

Backward Mapping คือการคำนวณหาพิกัดของจุดในภาพเก่าจากพิกัดของจุดในภาพใหม่ที่พิกัดที่ต้องการ โดยในการคำนวณจะใช้ Inverse ของ Transformation Matrix Forward Mapping จะทำให้เกิดช่องว่างขึ้นในภาพใหม่เนื่องจากค่าพิกัดที่คำนวณได้ไม่ไปตกยังจุดนั้นๆ วิธีแก้คือใช้ Backward Mapping ซึ่งจะทำการกำหนดจุดในภาพใหม่ลงไปก่อนจากนั้นจึงทำการคำนวณหาจุดที่กำหนดจากภาพเก่า

2D Transformation Matrix

ภาพรวมของการ Transformation คือ การนำค่าแต่ละพิกัด (x,y) ไปคูณกับ Matrix ที่ใช้ในการ Transformation แล้วนำค่าพิกัด (x,y) ที่ได้จากการคูณไป mapping

1. Translation

เป็นการย้ายรูปภาพไปยังจุดต่างๆบนระนาบ x,y โดยใช้ Translation Matrix คูณ กับ แต่ละพิกัด (x,y) ของรูปภาพ

Translation Matrix

$$Trd\_d) = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \qquad T^{-1}(d\_d) = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix}$$

โดย dx คือค่าที่ต้องการย้ายในแนวแกน x

dy คือค่าที่ต้องการย้ายในแนวแกน y

2. Scaling

$$S(s\_s) = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad S^{-1}(s\_s) = \begin{bmatrix} 1/Sx & 0 & 0 \\ 0 & 1/Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

เป็นการขยายภาพ โดยการ Translate รูปภาพไปยัง Origin ก่อนแล้วทำการ Scaling โดยใช้ Scaling Matrix แล้ว Translate กลับมาที่เดิม

Scaling Matrix

โดย Sx คือ ค่าที่ต้องการขยายในแนวแกน x

Sy คือ ค่าที่ต้องการขยายในแนวแกน y

3. Rotation

เป็นการหมุนภาพ โดยการ Translate รูปไปยัง Origin แล้วทำการ Rotate โดยใช้ Rotation Matrix แล้ว Translate กลับมาที่เดิม

Rotation Matrix

$$R(q) = \begin{bmatrix} \cos A & \sin A & 0 \\ \sin A & \cos A & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R'(q) = \begin{bmatrix} \cos A & \sin A & 0 \\ \sin A & \cos A & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

โดยที่ A คือมุมที่ต้องการจะหมุน

4. Shear

เป็นการแปลงภาพแบบเฉียง คือ การทำภาพในแนวนอน หรือแนวตั้ง หรือทั้ง 2 แนว โดย Translate รูปภาพไปยัง Origin แล้วทำการ Shear โดยใช้ Shear Matrix แล้ว Translate กลับมาที่เดิม

Shear Matrix

$$Sh(q) = \begin{bmatrix} 1 & Shx & 0 \\ Shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Sh'(q) = \frac{1}{1 - Shx * Shy} \begin{bmatrix} 1 & Shx & 0 \\ Shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

โดย Shx คือ ค่าที่ต้องการให้ภาพเฉียงในแนวแกน x

Shy คือ ค่าที่ต้องการให้ภาพเฉียงในแนวแกน y

ในแต่ละ Matrix ที่ใช้ในการ Transformation ที่กำหนดให้นี้จะใช้กระบวนการคูณทั้งหมดอย่างเช่น ถ้าอยากทำ Rotation จะได้ Matrix ที่ใช้ในการ Rotation ดังนี้

$$T = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix T เป็น Matrix ที่ใช้ในการ Rotation กระบวนการจะดูจากหลังมาหน้า โดย Translate รูปภาพไปยัง Origin ทำการหมุนภาพ แล้ว Translate กลับมาที่เดิม การ Inverse Transformation จะได้ T-1 โดย T-1 เกิดจากการนำ Matrix Inverse ของ Matrix Transformation มาคูณกันนั่นเอง

6.2 การทำเทรสโชว์ (Image thresholding)

การทำเทรสโชว์จะได้เอาท์พุทของภาพออกมาเป็นไบนารี่ Image thresholding เป็นการเปลี่ยนภาพจาก Intensity ให้กลายเป็น Binary Image โดยกระเลือกค่า Threshold มาค่าหนึ่งหรือเป็นช่วงซึ่งเป็นตัวแบ่งว่า Pixel ใดมีค่าเป็น '1' หรือ '0' ดังนั้นหากเราใช้จุดสีสามสีคือ แดง เขียว น เงิน เราจะสามารถทำการแบ่งภาพ RGB Image ออกเป็น Intensity Image ได้สามภาพ คือ Red, Green, Blue Channel Intensity Image แล้วนำมาทำการ Thresholding เพื่อแยกแยะวัตถุ

6.3 การหักลบภาพและตรวจการเปลี่ยนแปลง (Image Subtraction and Change Detection)

ในการประยุกต์ใช้งานในการประมวลผลภาพหลายอย่าง เราอาจต้องการที่จะเปรียบเทียบภาพที่มีความสลับซับซ้อน 2 ภาพ วิธีที่ง่ายแต่มีประสิทธิภาพคือ การเรียง(Align) ทั้งสองเข้าด้วยกันแล้วทำการหารหาผลต่างจากนั้นก็ทำการปรับคุณภาพผลต่าง ตัวอย่างเช่น อุปกรณ์ที่หายไปจากบอร์ดวงจรสามารถตรวจหาได้โดยภาพของบอร์ดดังกล่าวกับภาพบอร์ดที่มีอุปกรณ์วางถูกต้อง การประยุกต์ใช้งานอีกอย่างพบใช้กรณีที่เราต้องการทำภาพของหลอดเลือดและเส้นเลือดในร่างกาย ซึ่งทำได้โดยการฉีดสารทึบแสงเข้าไปในหลอดเลือดแล้วทำการถ่ายภาพเอ็กเรย์ จากนั้นทำการลบภาพออกจากภาพเอ็กเรย์ของหลอดเลือดก่อนฉีดสารทึบแสงผลต่างของภาพทั้งสองจะแสดงทางเดินของเลือดอย่างชัดเจน การประยุกต์ใช้งานอื่นๆคือการตรวจจับความแตกต่างในระบบรักษาความปลอดภัย หรือการตรวจแผ่นปรินท์วงจรโดยอัตโนมัติ และอื่นๆ

6.4 การแยก RGB

ใน 1 Pixel จะประกอบไปด้วยค่าสี 3 สี รวมกันเป็น 1 Pixel ซึ่งเราสามารถแยกค่าแต่ละสีออกมาจาก Pixel ได้ดังนี้

R = ค่าสี Mod 256

G = (ค่าสี div 256) Mod 256

B = 0 div 65536

บทที่ 7
แนวทางการออกแบบ

7.1 การทำงานโดยสังเขป

การเขียนโปรแกรมควบคุมการทำงานของหุ่นยนต์เตะฟุตบอลในโปรเจ็คนี้นั้น จะใช้การทำงานของโปรแกรมที่เขียนจากเดลไฟล์ ประมวลผลร่วมกับการทำงานของไมโครคอนโทลเลอร์ PIC โดยโปรแกรมที่เขียนจากเดลไฟล์จะประมวลผลผ่านไปสั่งการทำงานควบคุม Hard Ware ผ่านทางพอร์ท RS232 ของคอมพิวเตอร์ หลังจากนั้นข้อมูลที่ส่งมาจะนำไปประมวลผลผ่านไมโครคอนโทลเลอร์ PIC ซึ่งหลังจากนั้นจะผ่านข้อมูลที่มวลผลจาก PIC ซึ่งเป็นสัญญาณที่จะส่งไปขับมอเตอร์ให้ไปทำการผ่านการ Encoder ซึ่งจะเป็นการเพิ่มความเสถียรของสัญญาณ และอีกทั้งยังเป็นการเพิ่มย่านการทำงานของการคอนโทลสัญญาณที่อาจจะเพิ่มไว้ใช้ในอนาคตได้อีกด้วย เพราะสัญญาณที่เราได้นี้เราจะต้องนำไปผ่านการส่งสัญญาณไร้สายทางความถี่ RF เพื่อที่จะไปควบคุมการทำงานของ Hard Ware ทางภาครับสัญญาณอีกครั้งหนึ่ง ดังนั้นการ Encoder จะเป็นการทำให้สัญญาณที่ส่งไปยังภาครับจะมีความถูกต้อง และแม่นยำเพิ่มขึ้น

จากนั้นเมื่อทางภาครับได้รับสัญญาณที่ส่งมาจากทางภาคส่งแล้ว สัญญาณที่ได้จะถูกนำไปผ่านการ Decoder เพื่อให้ได้เป็นสัญญาณที่เหมือนกับสัญญาณที่ออกมาจากไมโครคอนโทรเลอร์ PIC สัญญาณที่ได้ออกมานี้จะนำไปประมวลผลโดยไมโครคอนโทรเลอร์เพื่อควบคุมการทำงานของมอเตอร์ โดยนำไปควบคุมวงจรขับมอเตอร์โดยในที่นี้ใช้ทรานซิสเตอร์เป็นตัวจ่ายกระแสไปควบคุมการทำงานของมอเตอร์ การทำงานของมอเตอร์นั้นจะเป็นการทำงานควบคุมการหมุนในสองทิศทางโดยมอเตอร์ 2 ตัว ถ้าหากจะให้รถวิ่งตรงก็ให้มอเตอร์ทั้งสองตัวหมุนในทิศทางเดียวกัน หากต้องการจะให้ถอยหลังก็ให้มอเตอร์ทั้งสองตัวหมุนในทิศทางที่ตรงกันข้ามกับในกรณีแรก ต่อมาหากจะให้มอเตอร์เลี้ยวซ้ายหรือขวาก็ให้มอเตอร์ทั้ง 2 ตัวหมุนตรงกันข้ามกัน โดยการจะเลี้ยวซ้ายหรือเลี้ยวขวานั้นขึ้นอยู่กับทิศทางการหมุนของมอเตอร์ทั้งสองตัวในการหมุนกรณีตรงและถอยหลังในตอนแรกเป็นตัวกำหนด

การควบคุมการทำงานของหุ่นยนต์นั้นจะควบคุมให้ทำงานโดยผ่านทางกล้อง CCD เป็นตัวตรวจจับวัตถุ ในที่นี้ก็คือตัวหุ่นยนต์แต่ในการตรวจจับนั้นเราจะใช้สีเป็นตัวประมวลผลในการส่งสัญญาณไปสั่งการทำงานส่วน Hard Ware ภายนอก โดยสีที่กล้องจับได้นั้นจะแบ่งเป็นสีส่วนที่หัวรถ และสีส่วนที่ท้ายรถ ซึ่งทั้งสองสีนั้นจะต้องเป็นสีที่ไม่เหมือนกันอย่างเด่นชัด และจะต้องตัดกับพื้นหลังอย่างเห็นได้ชัดอีกด้วย ทั้งนี้เพื่อป้องกันการตรวจจับสีที่ผิดพลาดและคลาดเคลื่อน จุดที่เป็นกึ่งกลางของสีที่เป็นของหัวรถและท้ายรถนั้น จะถูกนำไปประมวลผลผ่านทางโปรแกรม

เปรียบเทียบกับจุดที่เราคลิกเพื่อให้รถไปถึง โดยผ่านการโปรแกรมควบคุมซึ่งในที่นี้ใช้โปรแกรมเดลไฟล์เขียนขึ้นมา โดยใช้การเปรียบเทียบระยะห่างและทิศทางของจุดที่จะไปเทียบกับ จุดที่หัวท้ายของหุ่นยนต์

7.2 แนวการออกแบบ

ในการออกแบบในช่วงแรกนั้น ให้ทำการออกแบบให้ควบคุมการทำงานของหุ่นยนต์โดยในขั้นแรกนี้จะควบคุมในกรณีที่ยังไม่ผ่านสัญญาณไร้สายทาง RF และยังไม่ใช้กล้อง CCD เป็นตัวตรวจจับและควบคุมการทำงาน แต่การทำงานในตอนแรกนั้นควบคุมโดยให้คนเป็นผู้ควบคุม โดยให้เขียนโปรแกรมที่มีปุ่ม ตรง ถอย ซ้าย ขวา ยิง เป็นตัวควบคุมการทำงานโดยการคลิกที่ปุ่มนั้นๆ ให้หุ่นยนต์เคลื่อนที่ตามที่เราคลิก

เมื่อส่วนที่ควบคุมการทำงานโดยคนแบบมีสายใช้การได้แล้วต่อมาก็ทำการควบคุมแบบไร้สายโดยส่งผ่าน RF แต่ยังใช้คนเป็นคนควบคุมเหมือนเดิม เมื่อการควบคุมโดยคนใช้งานได้อย่างสมบูรณ์แล้ว ต่อมาจึงทำการควบคุมผ่านทางกล้อง CCD ให้ทำการเคลื่อนที่อัตโนมัติ โดยเขียนโปรแกรมควบคุมการทำงานเพิ่มเติมให้หุ่นยนต์สามารถเคลื่อนที่ไปตามจุดที่เราคลิกผ่านภาพที่กล้องได้

7.3 การส่งสัญญาณไปควบคุมการทำงานของ HARD WARE

การส่งสัญญาณไปควบคุมการทำงานจะใช้ไอซีไมโครคอนโทรเลอร์ 2 ชุดช่วยกันประมวลผล โดยชุดแรกนั้นจะอยู่ที่ชุดภาคส่งซึ่งจะทำหน้าที่นำสัญญาณที่ประมวลผลได้จากโปรแกรมควบคุมการทำงานซึ่งประมวลผลผ่านคอมพิวเตอร์ โดยสัญญาณที่ประมวลผลออกมาได้นั้นจะมีลักษณะเป็นโค๊ดอักขระแทนการทำงานของหุ่นทั้งสองตัวผ่านทาง Serial Port ระบบ RS 232 ซึ่งมีดังนี้

อักขระ	หุ่นตัวที่ 1	หุ่นตัวที่ 2
A	ตรง	ตรง
B	ซ้าย	ตรง
C	หยุด	ตรง
D	ขวา	ตรง
E	ยิง	ตรง
F	ตรง	ซ้าย
G	ซ้าย	ซ้าย
H	หยุด	ซ้าย

อักขระ	หุ่นตัวที่ 1	หุ่นตัวที่ 2
I	ขวา	ซ้าย
J	ยิง	ซ้าย
K	ตรง	หยุด
L	ซ้าย	หยุด
M	หยุด	หยุด
N	ขวา	หยุด
O	ยิง	หยุด
P	ตรง	ขวา
Q	ซ้าย	ขวา
R	หยุด	ขวา
S	ขวา	ขวา
T	ยิง	ขวา
U	ตรง	ยิง
V	ซ้าย	ยิง
W	หยุด	ยิง
X	ขวา	ยิง
Y	ยิง	ยิง

หลังจากที่คอนโครเลอร์ภาคส่งได้รับสัญญาณอักขระที่ได้ Serial Port ก็จะทำการประมวลผลสัญญาณที่ได้เป็น โค๊ดเลขฐานสอง 3 บิตเพื่อที่จะนำไปควบคุมหุ่นทั้ง 2 ตัว ซึ่งในที่นี้หุ่นหนึ่งตัวจะใช้สัญญาณควบคุม 3 บิต โดยที่แต่ละบิตก็จะส่งไปตามขา Port Output แต่ละขาแยกกันไป โดยที่บิต '1' ขานั้นจะ Active High และ บิต '0' ขานั้นจะActive Low ดังนั้นจะใช้ขา Output Port ในการควบคุมหุ่นทั้งหมด 6 ขา โดย 3 ขาแรกใช้ควบคุมหุ่นตัวที่ 1 ส่วนอีก 3 ขาที่เหลือใช้ควบคุมหุ่นตัวที่ 2 โดยโค๊ดการทำงานของหุ่นทั้ง 2 จะเหมือนกันแต่สามารถแยกกันทำงานได้โดย 3 ขาแรกที่ใช้ควบคุมหุ่นตัวที่ 1 จะส่งไป Encode และส่งผ่าน RF ในความถี่หนึ่ง ส่วนอีก 3 ขาที่ใช้ควบคุมหุ่นตัวที่ 2 ก็จะส่งไป Encode อีกตัวและส่งผ่าน RF ในอีกความถี่หนึ่ง โดยโค๊ดควบคุมหุ่นแต่ละตัวเป็นดังนี้

ขา Output1	ขา Output2	ขา Output3	การทำงานของหุ่น
0	0	0	หยุด
0	0	1	ตรง
0	1	0	ซ้าย
0	1	1	ขวา
1	0	0	ยิง

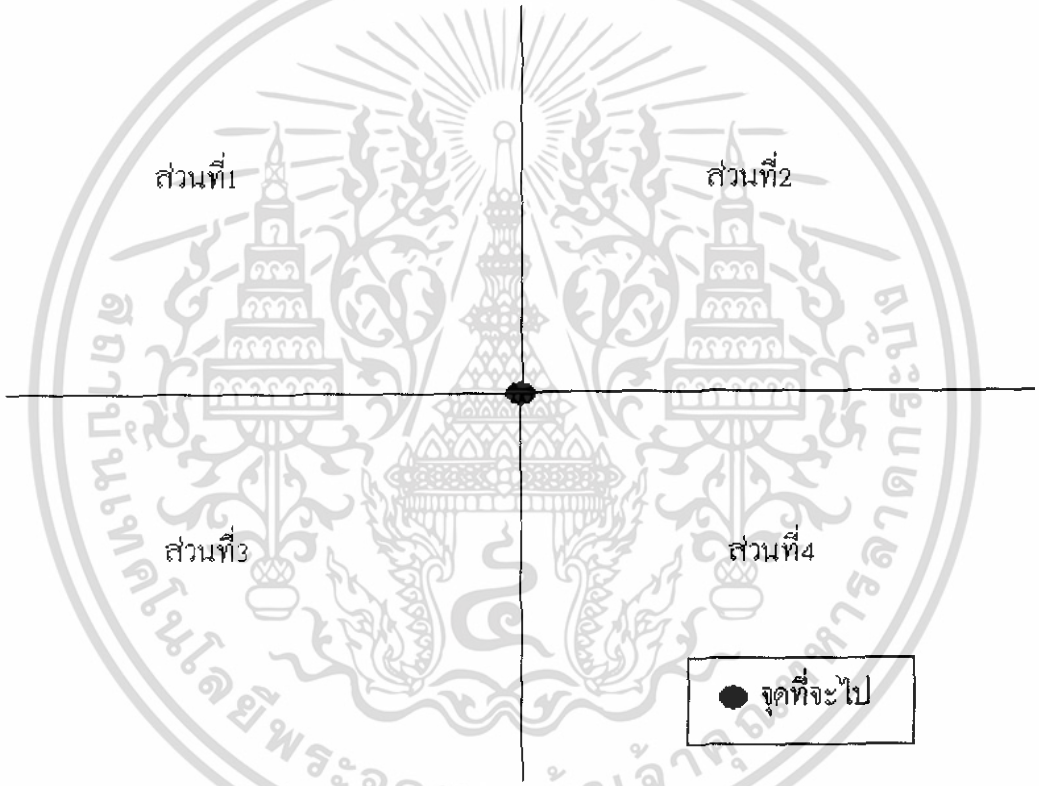
ต่อมาไอซีคอนโทรเลอร์ชุดที่ 2 ซึ่งในที่นี้อยู่ที่ภาครับ หรือบนตัวหุ่นนั่นเอง แต่ในที่นี่มีหุ่น 2 ตัวดังนั้นจึงมีภาครับ 2 ชุด ใช้คอนโทรเลอร์ 2 ตัวแยกกันประมวลผลโดยสัญญาณ RF ที่ส่งมาจากภาคส่ง ในที่นี้มี 2 ความถี่แยกกันประมวลผลตามหุ่นแต่ละตัวโดยหุ่นตัวที่ 1 ก็จะรับสัญญาณความถี่หนึ่ง ส่วนตัวที่ 2 ก็จะรับอีกความถี่หนึ่งตามที่กล่าวไปแล้วข้างต้น หลังจากนั้นสัญญาณของหุ่นแต่ละตัวที่รับได้ก็จะนำมาผ่านการ Decode จากนั้นนำสัญญาณซึ่งผ่านการ Decode มาแล้ว ซึ่งในที่นี้จะให้ Output เหมือนกับ Input ที่เข้าการ Encode ในภาคส่ง ต่อมาจึงนำ Output ที่ออกจาก Decoder มาผ่านคอนโทรเลอร์เพื่อที่จะทำการประมวลผลแล้วนำสัญญาณดังกล่าวในควบคุมมอเตอร์ต่อไป

7.4 การเขียนโปรแกรมควบคุมการทำงานของกล้อง

การเขียนโปรแกรมควบคุมการทำงานโดยการประมวลผลของกล้อง CCD นั้นเราจะใช้การกำหนดจุดหัวของหุ่นและส่วนท้ายของหุ่นทั้งสองตัว จุดพิกัดของโกล์ทั้งสองฝั่งและพิกัดของบอลเพื่อนำไปประมวลผลผ่านโปรแกรมเดลไฟล์ โดยในที่นี้จะใช้ฟังก์ชั่นต่างๆ ช่วยในการประมวลผลอันประกอบด้วยฟังก์ชั่น Gotopoint ซึ่งทำหน้าที่ให้หุ่นยนต์เคลื่อนที่ไปยังพิกัดที่ต้องการ ฟังก์ชั่น Break ทำหน้าที่ป้องกันไม่ให้หุ่นทั้งสองตัววิ่งชนกัน ฟังก์ชั่น Defen ทำหน้าที่ให้หุ่นป้องกันประตูของฝ่ายตนเอง ฟังก์ชั่น Shoot ทำหน้าที่ยิงประตู

7.4.1 ฟังก์ชั่น Gotopoint

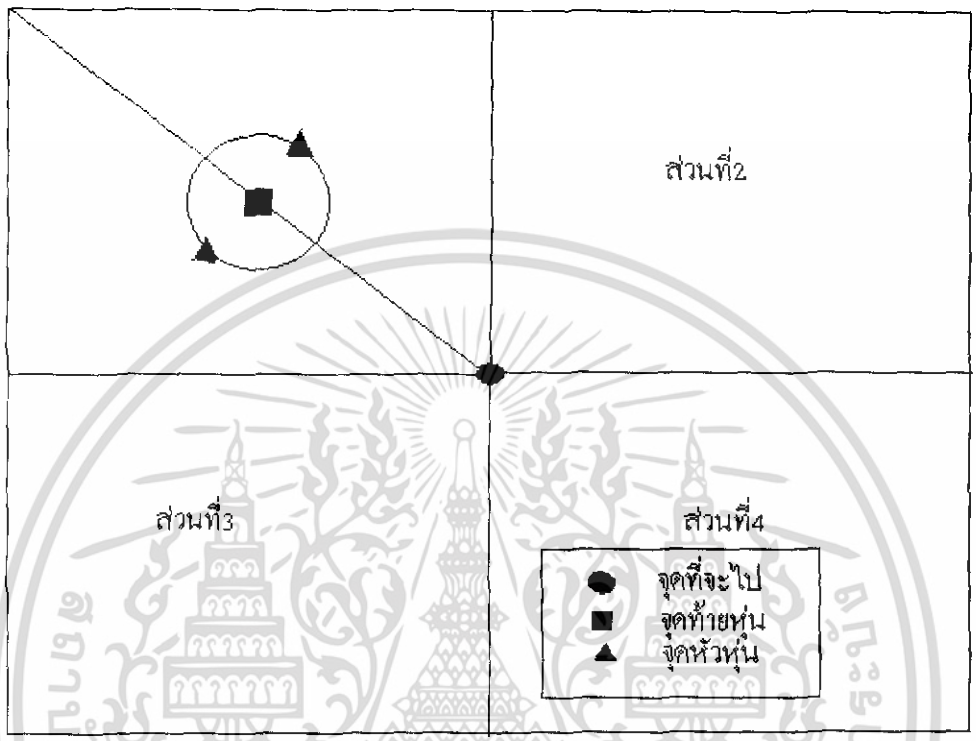
ฟังก์ชั่นนี้จะทำหน้าที่ให้หุ่นตัวที่กำหนดวิ่งไปยังจุดที่ต้องการ โดยการนำพิกัดจุดหัวหุ่น จุดท้ายหุ่น และจุดที่หุ่นจะไปเป็นตัวประมวลผล โดยในการประมวลผลจะแบ่งหน้าจอออกเป็น 4 ส่วน ซึ่งจะใช้ จุดที่จะไปเป็นจุดอ้างอิง หากจุดหัวหุ่นและจุดท้ายหุ่นอยู่ในส่วนใด ก็ให้ทำการประมวลผลตามที่กำหนดดังนี้



รูปที่ 7.1 แสดงส่วนอ้างอิงต่างๆ เทียบกับจุดที่จะไป

7.4.1.1 อยู่ในส่วนที่ 1

ใช้รูปอธิบายประกอบดังนี้

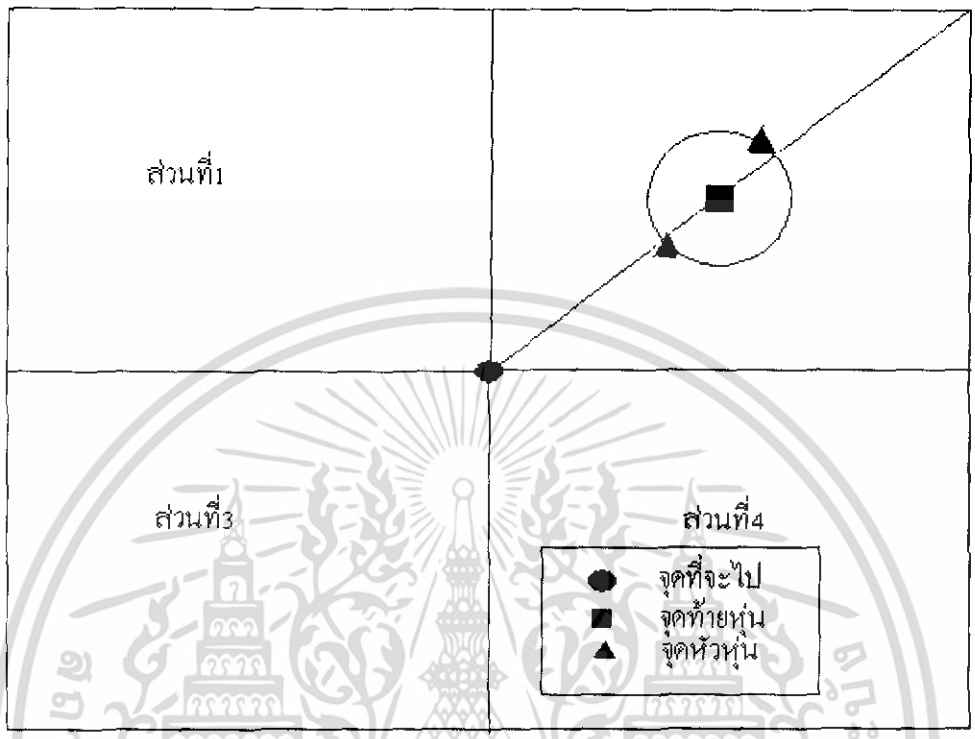


รูปที่ 7.2 แสดงส่วนที่ 1

จากรูปหากจุดหัวหุ่นอยู่ในส่วนครึ่งวงกลมบนแต่ไม่อยู่ในเส้นทแยงมุม ให้หุ่นยนต์เลี้ยวขวาจนกว่าจุดหัวหุ่นจะอยู่บนเส้นทแยงมุมจึงให้หุ่นวิ่งตรง ในทางตรงข้ามหากจุดหัวหุ่นอยู่ในส่วนครึ่งวงกลมล่าง ก็ให้หุ่นเลียวซ้ายจนกว่าจุดหัวหุ่นจะอยู่ในเส้นทแยงมุม

7.4.1.2 อยู่ในส่วนที่ 2

อธิบายตามดังรูป

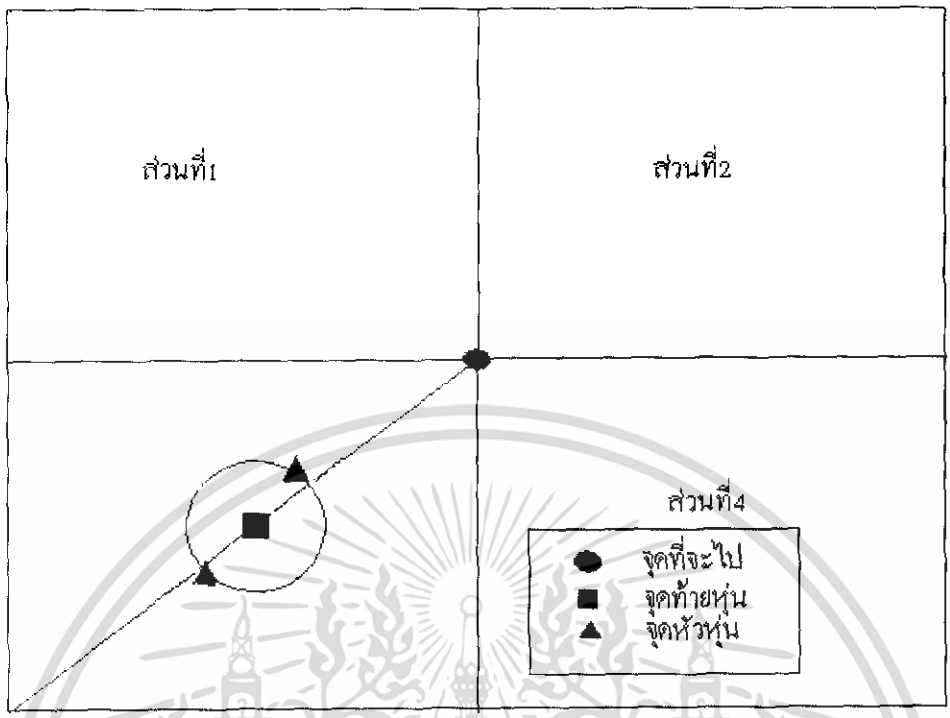


รูปที่ 7.3 แสดงส่วนที่ 2

จากรูปหากจุดหัวหุ่นอยู่ในส่วนครึ่งวงกลมบนแต่ไม่อยู่ในเส้นทแยงมุม ให้หุ่นยนต์เลี้ยวซ้ายจนกว่าจุดหัวหุ่นจะอยู่บนเส้นทแยงมุมจึงให้หุ่นวิ่งตรง ในทางตรงข้ามหากจุดหัวหุ่นอยู่ในส่วนครึ่งวงกลมล่าง ก็ให้หุ่นเลียวขวาจนกว่าจุดหัวหุ่นจะอยู่ในเส้นทแยงมุม

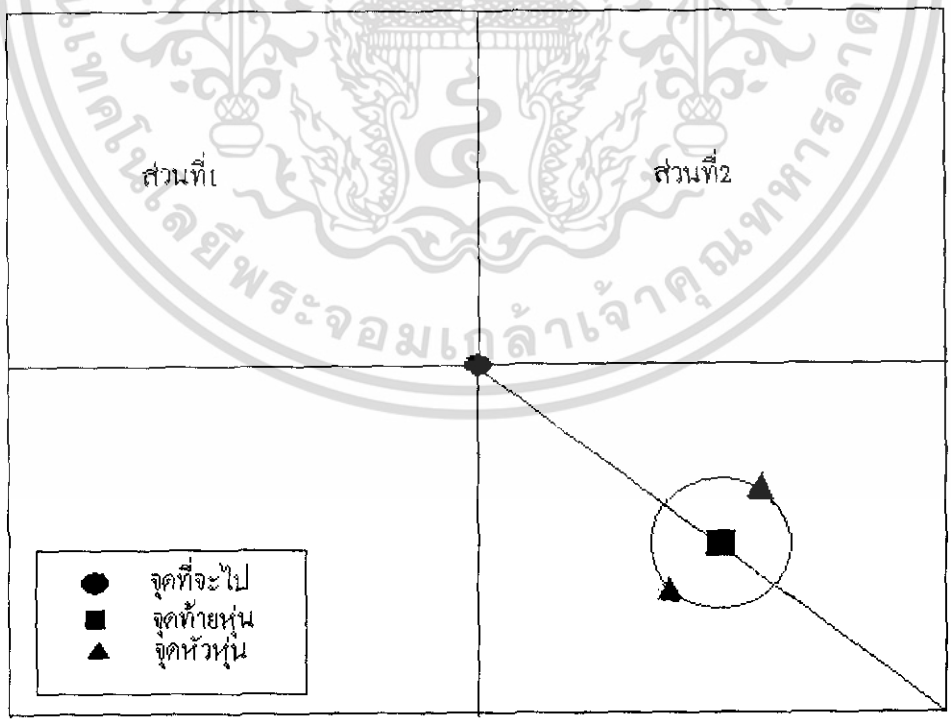
7.4.1.3 อยู่ในส่วนที่ 3

จากรูปหากจุดที่หัวหุ่นอยู่ในครึ่งวงกลมบนแต่ไม่อยู่ในเส้นทแยงมุม ให้หุ่นเลี้ยวขวาจนกว่าหุ่นจะเลี้ยวไปจนถึงจุดทับเส้นทแยงมุมจึงให้หุ่นวิ่งตรง แต่หากจุดหัวหุ่นอยู่ในส่วนครึ่งวงกลมล่าง ก็ให้หุ่นเลี้ยวซ้ายแทนจนกว่าจุดหัวหุ่นจะอยู่ทับเส้นทแยงมุมจึงให้ตรง ตามรูป



รูปที่ 7.4 แสดงส่วนที่ 3

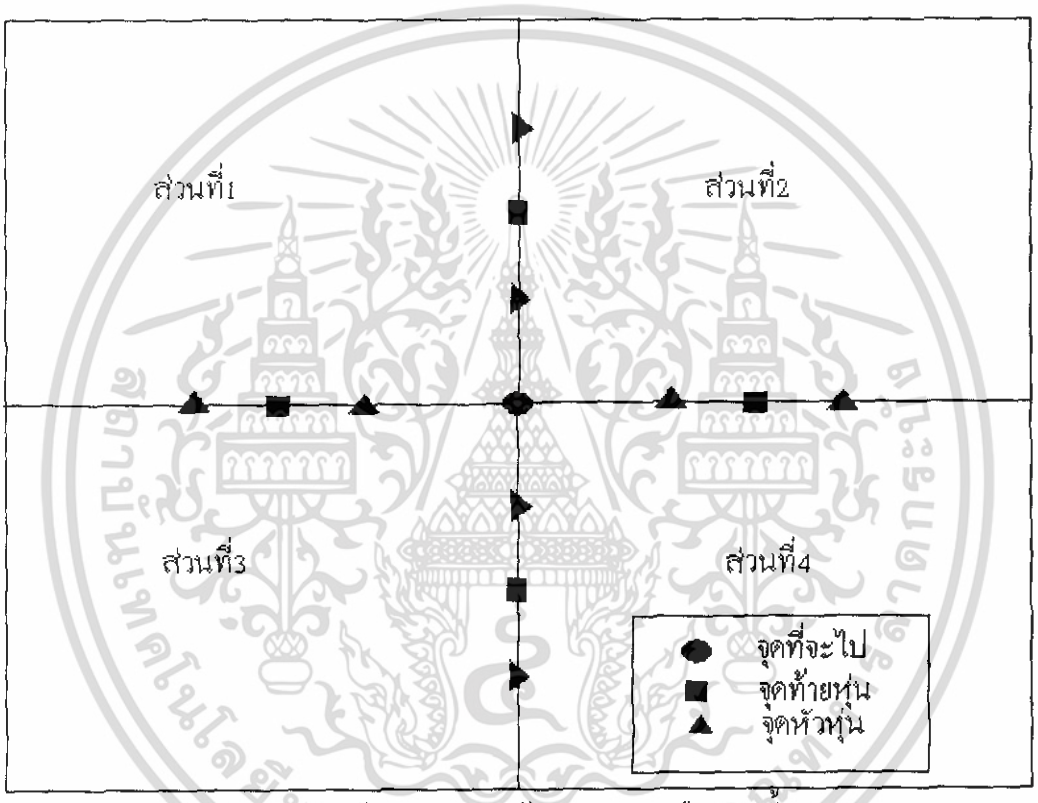
7.4.1.4 อยู่ในส่วนที่ 4



รูปที่ 7.5 แสดงส่วนที่4

หากจุดที่หัวรถอยู่ในครึ่งวงกลมบน ให้หุ่นยนต์เลี้ยวซ้ายจนกว่าจะทับกับเส้นทแยงมุม จึงให้หุ่นวิ่งตรงต่อไป แต่ในทางตรงกันข้ามหากจุดหัวหุ่นอยู่ในครึ่งวงกลมล่างก็ให้หุ่นเลี้ยวขวาจนกว่าจนกว่าจุดหัวหุ่นจะทับเส้นทแยงมุมจึงให้หุ่นวิ่งตรง

7.4.1.5 จุดหัวหุ่นและจุดท้ายหุ่นอยู่ทับเส้นแนวนอนหรือแนวตั้ง



จุดหัวท้ายหุ่นอยู่บนเส้นแนวนอนหรือแนวตั้ง

รูปที่ 7.6

หากจุดหัวและท้ายหุ่นอยู่บนเส้นแนวนอนหรือแนวตั้ง กรณีหากจุดหัวหุ่นอยู่ระหว่างจุดท้ายหุ่นและจุดที่จะไป ให้หุ่นวิ่งตรง แต่หากจุดท้ายหุ่นอยู่ระหว่างจุดหัวหุ่นและจุดที่จะไปให้หุ่นเลี้ยวซ้าย

จากทุกกรณีที่กล่าวมาทั้งหมดนี้ตั้งแต่ 7.4.1.1 – 7.4.1.5 หากจุดหัวหุ่นอยู่ระหว่างจุดท้ายหุ่นและจุดที่จะไป ให้หุ่นวิ่งตรง แต่หากจุดท้ายหุ่นอยู่ระหว่างจุดหัวหุ่นและจุดที่จะไปให้หุ่นเลี้ยวซ้าย

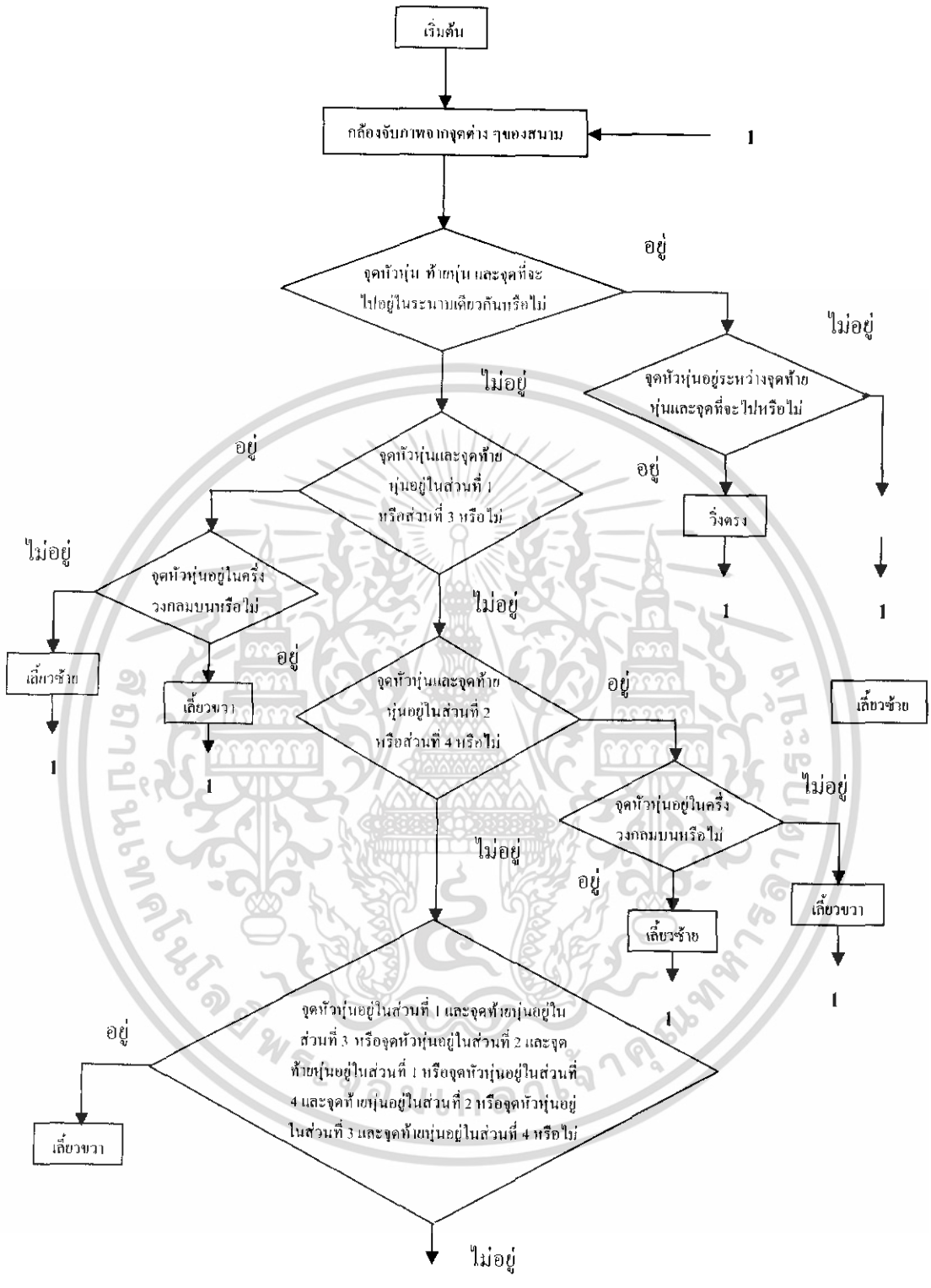
นอกจากที่กล่าวมาแล้วยังมีอีก 2 กรณีที่ต้องเขียนโปรแกรมควบคุมคือ

- **จุดหัวหุ่นอยู่ในส่วนที่ 1 จุดท้ายหุ่นอยู่ในส่วนที่ 3 หรือ จุดหัวหุ่นอยู่ในส่วนที่ 3 จุดท้ายหุ่นอยู่ในส่วนที่ 4 หรือ จุดหัวหุ่นอยู่ในส่วนที่ 4 จุดท้ายหุ่นอยู่ในส่วนที่ 2 หรือ จุดหัวหุ่นอยู่ในส่วนที่ 2 จุดท้ายหุ่นอยู่ในส่วนที่ 1**

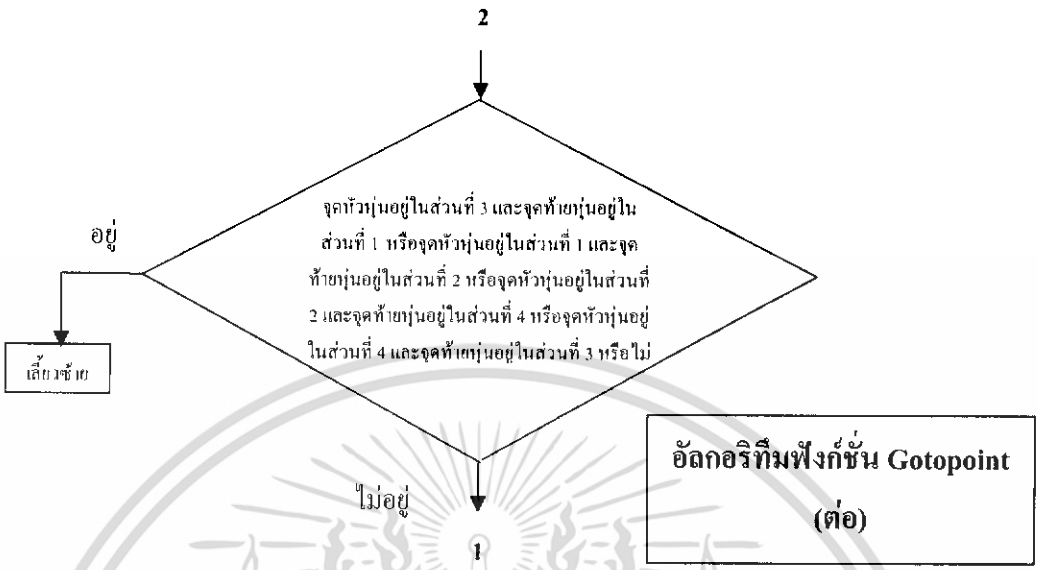
ในกรณีนี้ให้หุ่นเลี้ยวขวาจนกว่าจะไปตรงในกรณีที่ 7.4.1.5 จึงให้ทำงานตามเงื่อนไขกรณีที่ 7.4.1.5

- **จุดหัวหุ่นอยู่ในส่วนที่ 1 จุดท้ายหุ่นอยู่ในส่วนที่ 2 หรือ จุดหัวหุ่นอยู่ในส่วนที่ 2 จุดท้ายหุ่นอยู่ในส่วนที่ 4 หรือ จุดหัวหุ่นอยู่ในส่วนที่ 4 จุดท้ายหุ่นอยู่ในส่วนที่ 3 หรือ จุดหัวหุ่นอยู่ในส่วนที่ 3 จุดท้ายหุ่นอยู่ในส่วนที่ 1**

ในกรณีนี้ให้หุ่นเลี้ยวซ้ายจนกว่าจะไปตรงในกรณีที่ 7.4.1.5 จึงให้ทำงานตามเงื่อนไขกรณีที่ 7.4.1.5



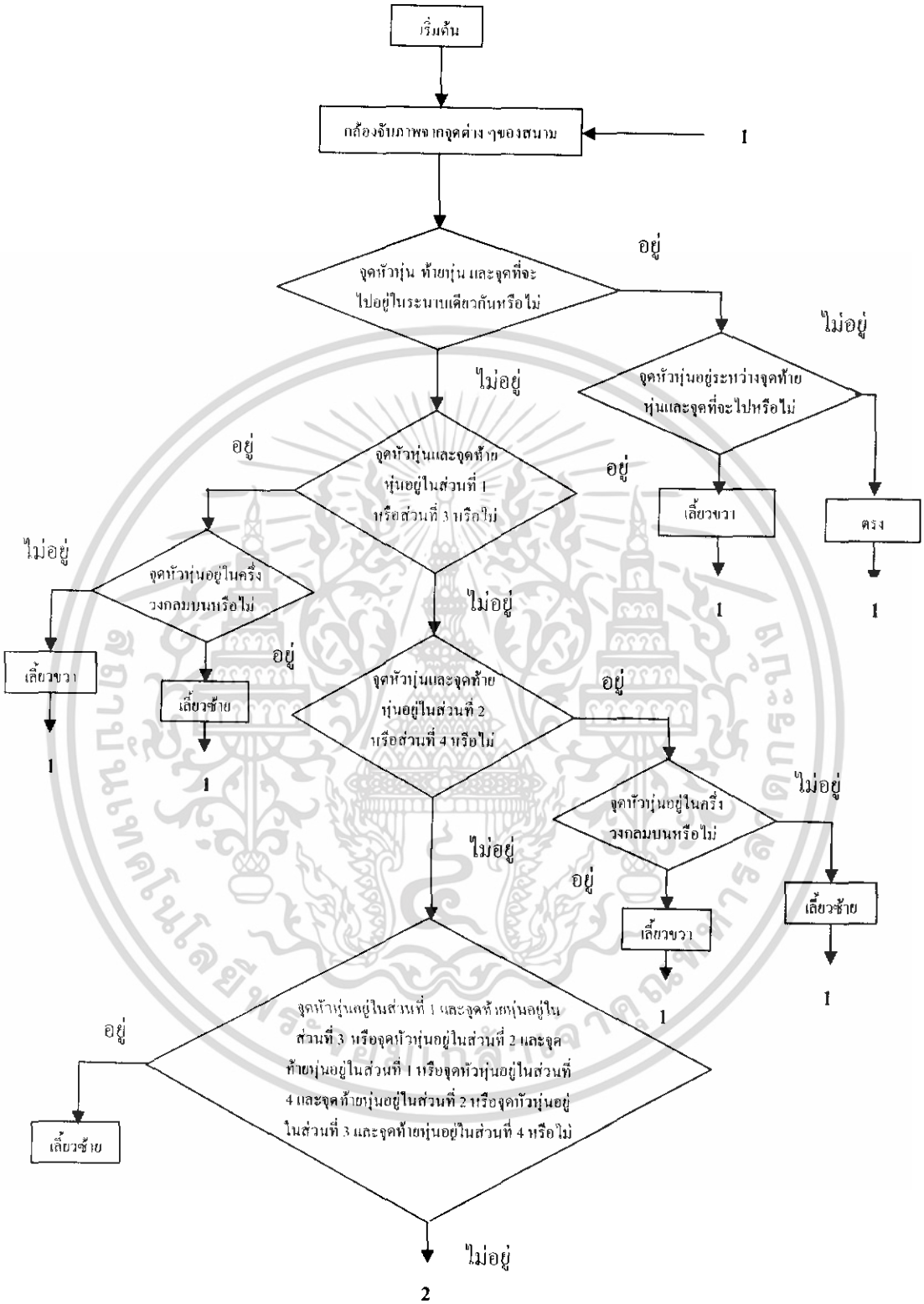
อัลกอริทึมฟังก์ชั่น Gotopoint



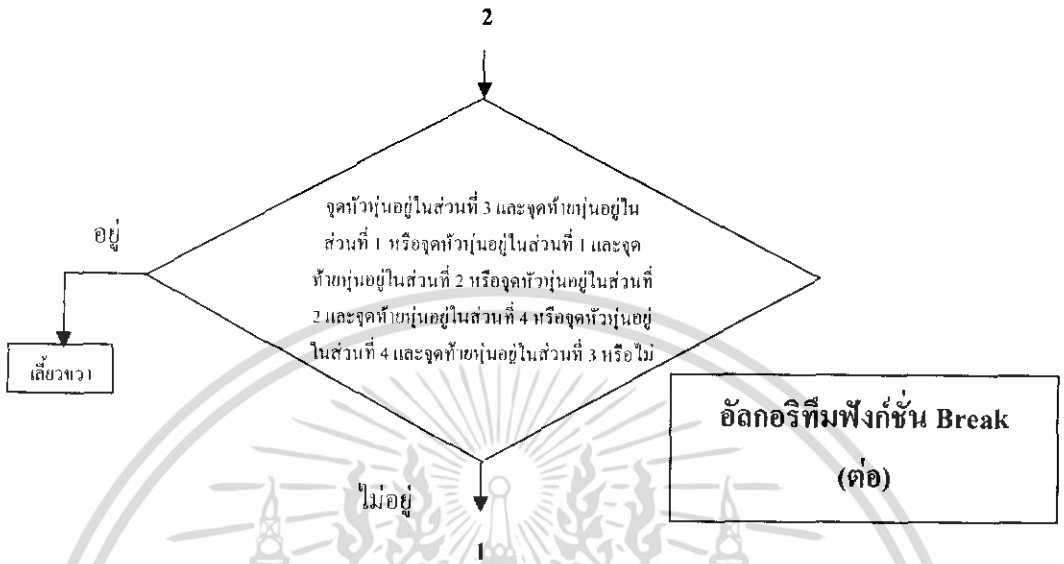
รูปที่ 7.7 อัลกอริทึมฟังก์ชั่น Gotopoint

7.4.2 ฟังก์ชั่น Break

ฟังก์ชั่น Break จะทำหน้าที่ป้องกันไม่ให้หุ่นวิ่งไปชนหุ่นอิกตัวอันจะทำให้เกิดการติดขัดได้ โดยในฟังก์ชั่นนี้จะใช้จุดหัวหุ่น จุดท้ายหุ่น และพิกัดจุดกึ่งกลางของตัวหุ่นฝ่ายตรงข้ามเป็นตัวประมวลผล ซึ่งการทำงานจะตรงกันข้ามกับกับฟังก์ชั่น Gotopoint ทุกประการ โดยเปลี่ยนจุดที่จะเคลื่อนที่ไปเป็นพิกัดของหุ่นฝ่ายตรงข้ามแทน และเหตุการณ์ไหนที่ฟังก์ชั่น Gotopoint สั่งให้หุ่นเลี้ยวซ้าย ฟังก์ชั่นนี่จะให้เลี้ยวขวา และเหตุการณ์ไหนที่ฟังก์ชั่น Gotopoint สั่งให้หุ่นเลี้ยวขวา ฟังก์ชั่นนี่จะให้เลี้ยวซ้ายแทน ถ้าตรงก็ให้เปลี่ยนเป็นเลี้ยวว้ายหรือขวาก็ได้ ซึ่งจะมีอัลกอริทึมดังนี้



อัลกอริทึมฟังก์ชั่น Break



รูปที่ 7.7 อัลกอริทึมฟังก์ชั่น Break

7.4.3 ฟังก์ชั่น Defen

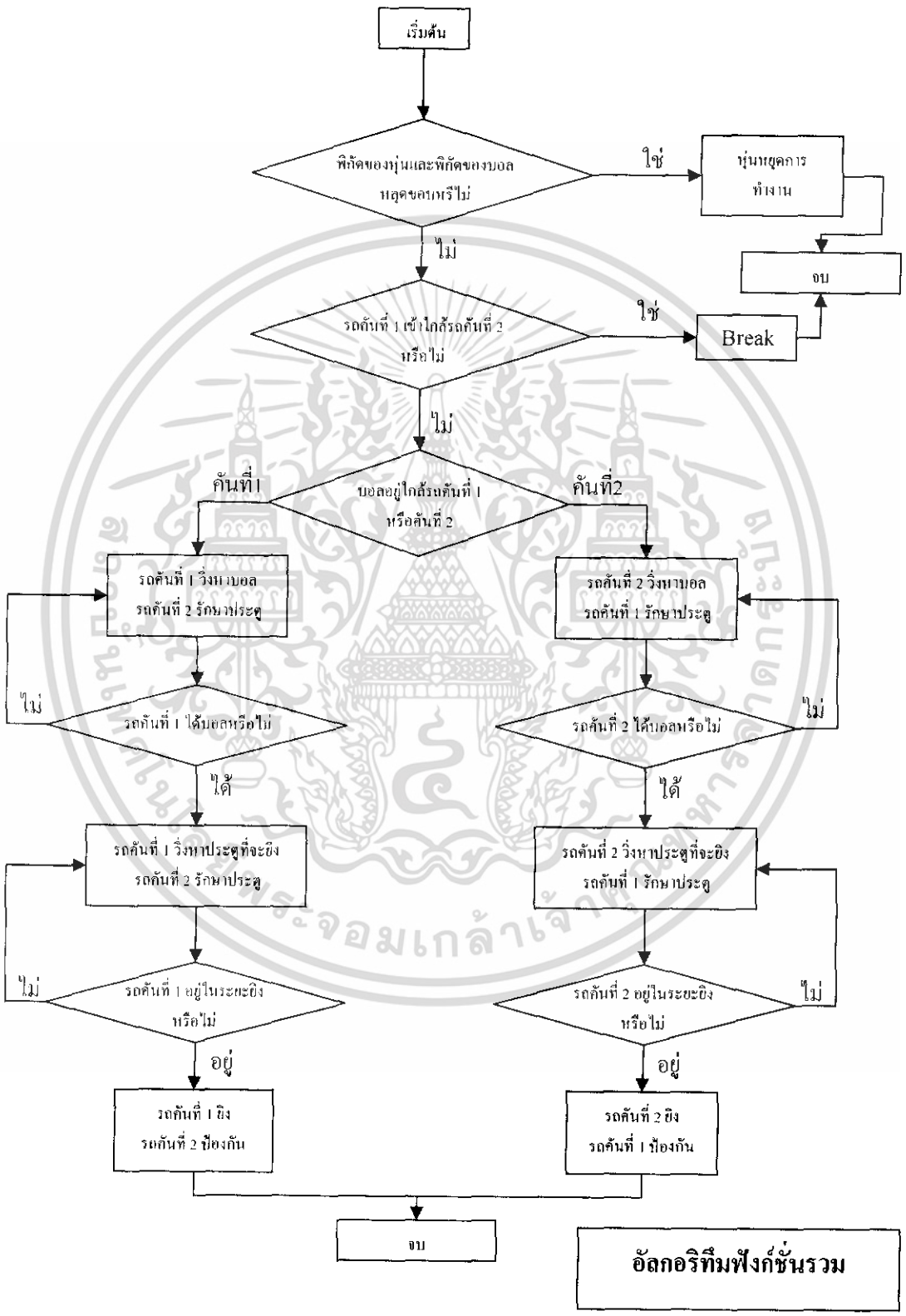
ฟังก์ชั่นนี่จะทำหน้าที่ในการป้องกันประตูของฝ่ายตนเองโดยการนำจุดหัวหุ่นจุดท้ายหุ่นและพิกัดของบอลนำมาประมวลผล บวกกับการนำฟังก์ชั่น Gotopoint มาช่วยโดยการให้หุ่นวิ่งไปบังระหว่างพิกัดของโกล์ฝ่ายตนเองกับพิกัดของบอล ซึ่งจะใช้ฟังก์ชั่น Gotopoint ช่วยในการให้หุ่นวิ่งไปในพิกัดนั้น

7.4.4 ฟังก์ชั่น Shoot

เป็นฟังก์ชั่นซึ่งทำหน้าที่ยิงประตูฝ่ายตรงข้าม โดยจะนำจุดหัวหุ่น จุดท้ายหุ่น และพิกัดของโกล์ฝ่ายตรงข้ามมาประมวลผลการทำงานจะเหมือนกับฟังก์ชั่น Gotopoint เพียงแต่เปลี่ยนพิกัดจุที่จะไปเป็นพิกัดของโกล์ฝ่ายตรงข้าม และเปลี่ยนจากการวิ่งตรงเป็นยิงแทน

7.4.5 ฟังก์ชั่นรวม

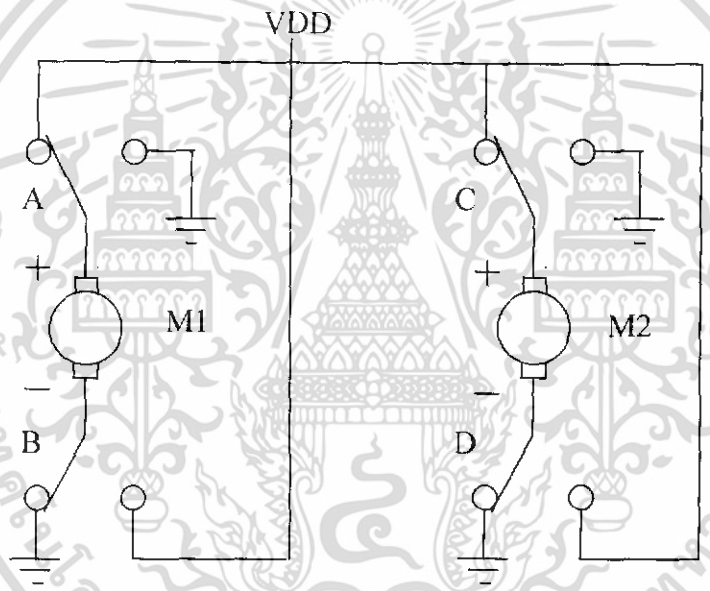
เป็นการนำเอาฟังก์ชั่นที่กล่าวมาแล้วทั้งหมดมาประมวลผลควบคุมการทำงานของหุ่นยนต์ทั้ง 2 ตัวซึ่งมีอัลกอริทึมดังนี้



7.5 การเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทลเลอร์

การเขียนโปรแกรมควบคุมหุ่นยนต์ในที่นี้ใช้ไมโครคอนโทลเลอร์ PIC เป็นตัวควบคุม ในที่นี้จะใช้ Compiler (โปรแกรมที่ใช้สำหรับเขียนภาษาควบคุมการทำงานให้แก่ไมโครคอนโทลเลอร์) โปรแกรมอะไรก็ได้ ซึ่งจะสามารถใช้อัลกอริทึมเดียวกันได้หมด

แนวการเขียนโปรแกรมแกรมในที่นี้จะใช้เพื่อควบคุมการทำงานของมอเตอร์ 2 ตัวให้หมุนซ้าย หมุนขวา หรือหยุด แล้วแต่การประมวลผลของโปรแกรมหลักในคอมพิวเตอร์จะส่งข้อมูลที่ต้องการให้ทำงานมาแบบไหนซึ่งได้กล่าวมาแล้วในหัวข้อข้างต้น การหมุนของมอเตอร์จะสามารถกำหนดได้โดยการจ่ายสัญญาณการทำงานที่ขาควบคุมการทำงาน 4 ขาโดยสมมติให้เป็นขา A, B, C, D ซึ่งลักษณะวงจรสมมูลของการควบคุมมอเตอร์จะเป็นดังนี้



รูปที่ 7.8 แสดงวงจรสมมูลการทำงานของขาไมโครคอนโทลเลอร์แทนสวิตซ์

จากรูปให้สวิตซ์ตัวใดต่อวงจรไปที่ VDD ก็ต่อเมื่อมีสัญญาณ Active High จ่ายออกมาจากขาของไมโครคอนโทลเลอร์ที่แทนด้วยขานั้น ๆ เช่นในรูปสวิตซ์ A ต่อไปที่ VDD และ สวิตซ์ B ต่อไปที่กราวด์ แสดงว่าในขณะนี้ขา A ของไมโครคอนโทลเลอร์ Active High แต่ขา B ของไมโครคอนโทลเลอร์ Active Low ดังนั้นสรุปการทำงานดังนี้ถ้าให้การทำงานดังรูปมอเตอร์หมุนไปในทางทิศเดียวกันโดยให้เป็นทิศตรง

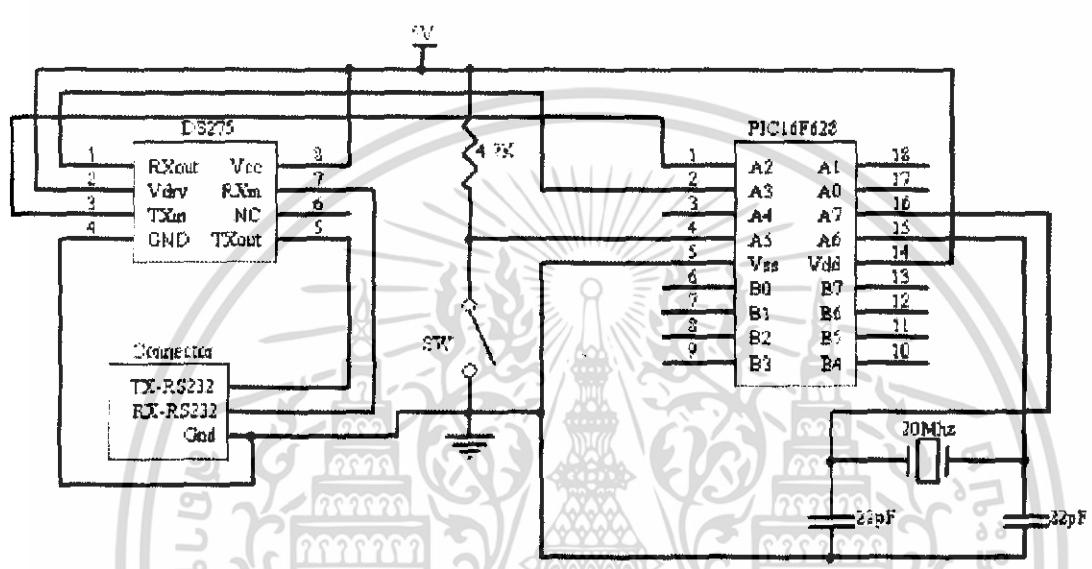
การทำงาน	ขาที่ Active High
วิ่งตรง	A, C
ถอย	B, D
ซ้าย	B, C
ขวา	A, D
หยุด	ไม่มี

จะเห็นว่าการจะทำให้หุ่นยนต์เลี้ยวนั้นจะต้องทำให้มอเตอร์ทั้ง 2 ตัวหมุนไปในคนละทิศทางกัน ซึ่งก็แล้วแต่ว่าเป็นไปในทิศทางไหน และถ้าตรงมอเตอร์ทั้ง 2 ตัวจะหมุนไปในทิศทางเดียวกัน แต่มอเตอร์ทั้ง 2 ตัวจะหมุนในทิศเดียวกันแต่กลับข้างกับในกรณีตรงถ้าให้หุ่นถอยหลัง

บทที่ 8

การทดลอง และ ผลการทดลอง

การทดลองที่ 1 ทดลองการติดต่อระหว่างพอร์ตอนุกรม กับ PIC16F628



รูปที่ 8.1 รูปการทดลองที่ 1

วัตถุประสงค์

เพื่อทดสอบการส่งข้อมูลผ่านพอร์ตอนุกรมเพื่อติดต่อ กับ PIC16F628

ขั้นตอนการทดลอง

1. เขียนโปรแกรมควบคุม PIC16F628 โดยใช้โปรแกรม CCS C Compiler โดยมี Source code ดังนี้

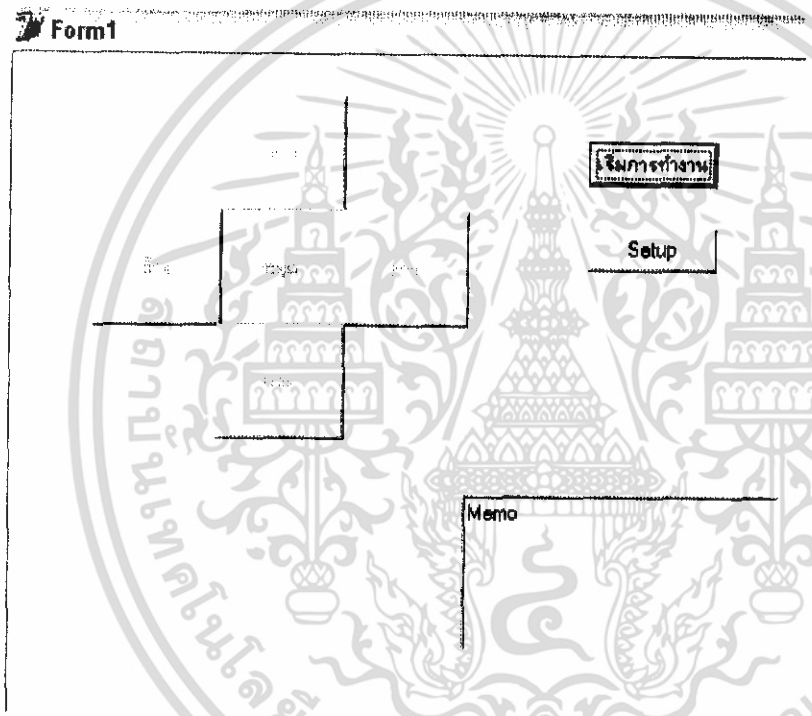
```
#include <16F628.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT,nomclr
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_B2, rcv=PIN_B1)
#define BUFFER_SIZE 32
byte buffer[BUFFER_SIZE];
```

```
 byte next_in = 0;
 byte next_out = 0;
#int_rda
 void serial_isr() {
 int t;
 buffer[next_in]=getc();
 t=next_in;
 next_in=(next_in+1) % BUFFER_SIZE;
 if(next_in==next_out)
 next_in=t;
}
#define bkbhit (next_in!=next_out)
byte bgetc() {
 byte c;
 while(!bkbhit) ;
 c=buffer[next_out];
 next_out=(next_out+1) % BUFFER_SIZE;
 return(c);
 }
 main() {
 char ch;
 enable_interrupts(global);
 enable_interrupts(int_rda);
 do {
while(bkbhit)
ch =  bgetc() ;
switch(ch) {
case'1': printf("\r\nBuffered data => %c",ch);
```

```
output_high (PIN_A1);
 output_high (PIN_B5);
 output_low (PIN_A0);
 output_low (PIN_B4)
 break;
 case'2': printf("\r\nBuffered data => %c",ch);
 output_high (PIN_A1);
 output_low (PIN_B5);
 output_low (PIN_A0);
 output_low (PIN_B4);
 break;
 case'3': printf("\r\nBuffered data => %c",ch);
 output_low (PIN_A1);
 output_low (PIN_B5);
 output_low (PIN_A0);
 output_low (PIN_B4);
 break;
 case'4': printf("\r\nBuffered data => %c",ch);
 output_low (PIN_A1);
 output_high (PIN_B5);
 output_low (PIN_A0);
 output_low (PIN_B4);
 break;
 case'5': printf("\r\nBuffered data => %c",ch);
 output_low (PIN_A1);
 output_low (PIN_B5);
 output_high (PIN_A0);
 output_high (PIN_B4);
```

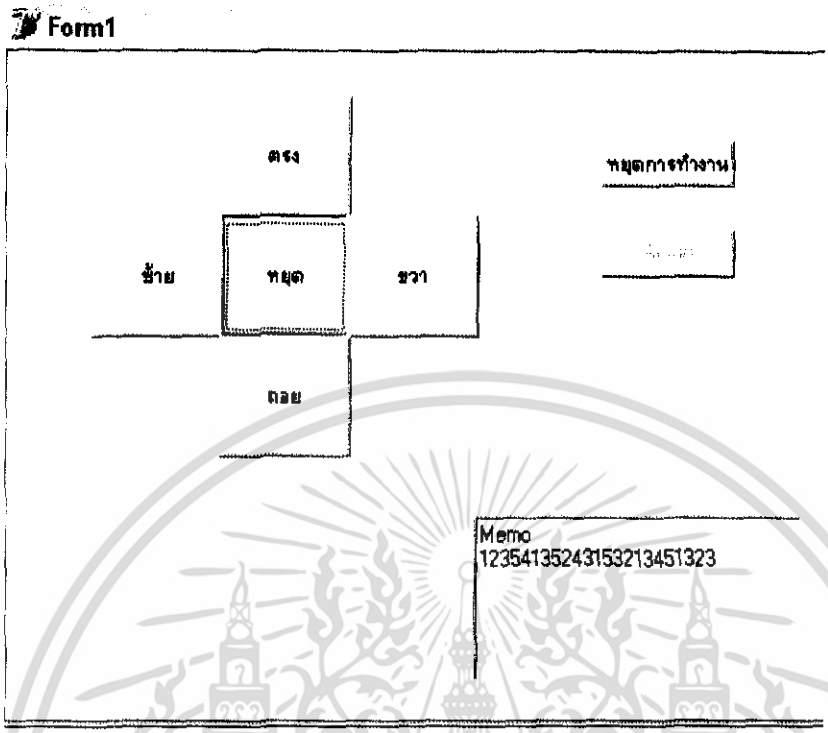
```
break;
}
ch=0;
} while (TRUE);
}
```

2. เขียนโปรแกรม Del Phi เพื่อติดต่อกับ พอร์ตอนุกรม



รูปที่ 8.2 ภาพโปรแกรม Del Phi

1. ต่อวงจรดังรูปที่ 8.1
2. สังเกตที่หน้าจอโปรแกรมจะมีการส่งค่ากลับมาเหมือนกับค่าที่ได้ส่งไป

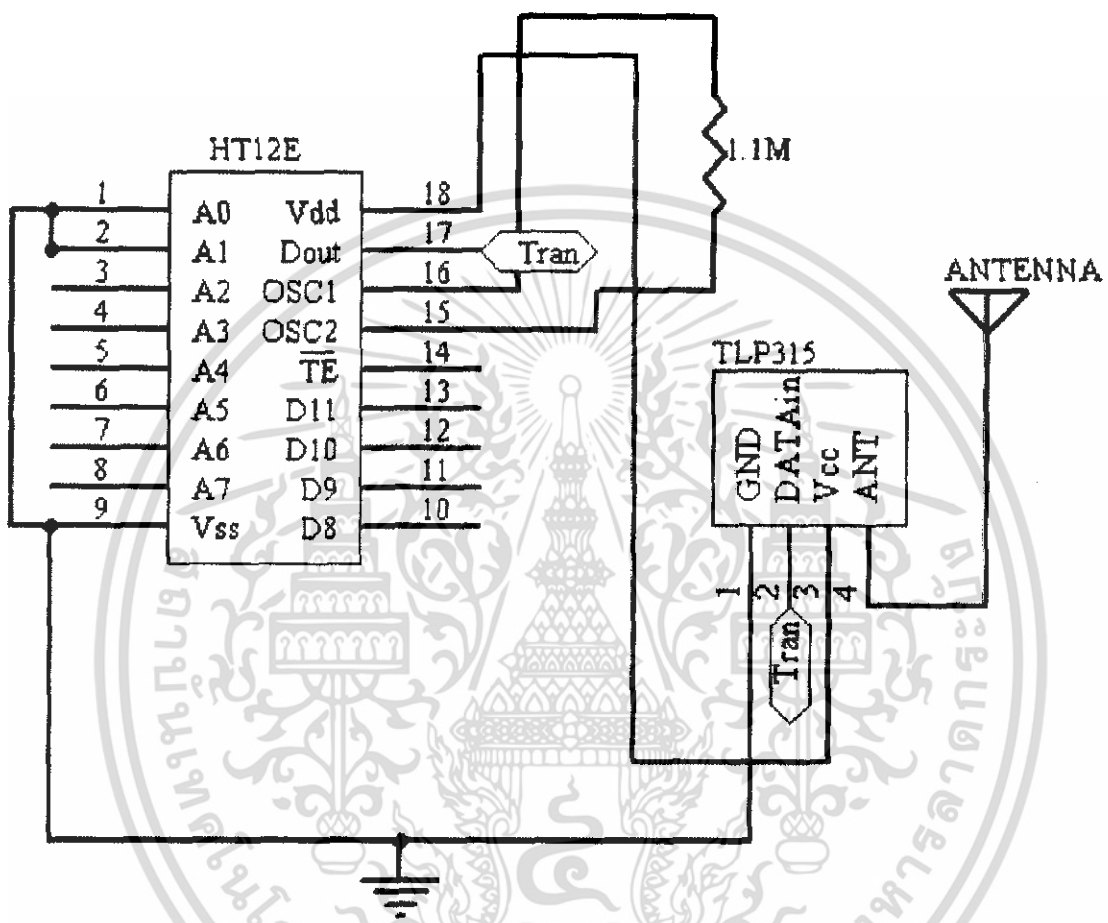


รูปที่ 8.3 ผลที่ได้จากการทดลอง

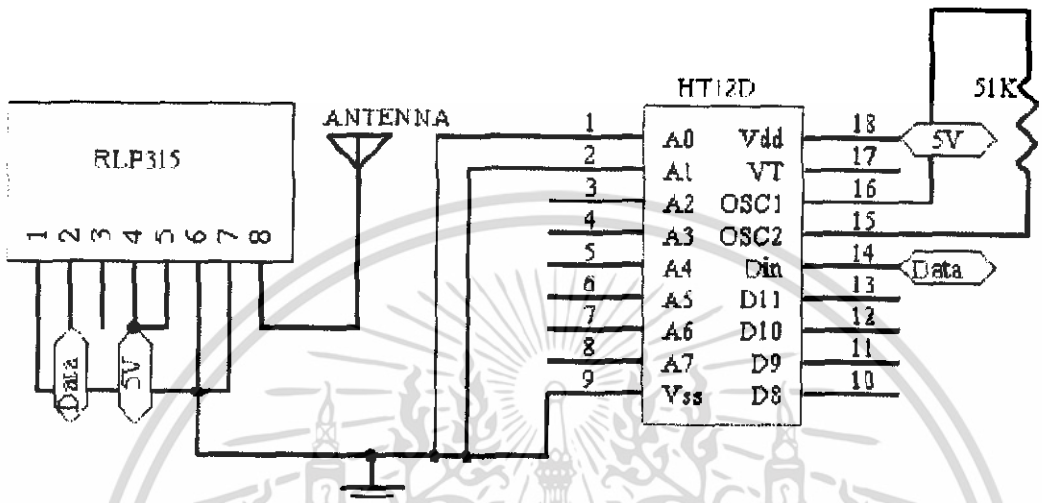
สรุปผลการทดลอง

สามารถทำการติดต่อพอร์ตอนุกรม กับ PIC16F628 ได้

การทดลองที่ 2 การทดลองการรับ-ส่งข้อมูลด้วย TLP315 กับ RLP315



รูปที่ 8.4 ภาพวงจรตัวส่ง



รูปที่ 8.5 ภาพวงจรตัวรับ

วัตถุประสงค์

เพื่อทดสอบการส่งข้อมูลของตัวรับและตัวส่ง

ขั้นตอนการทดลอง

1. ต่อวงจรตามรูปที่ 8.4 และรูปที่ 8.5
2. ป้อนข้อมูลเข้าที่ขา D8-D11 ของ HT12E โดย Set ขา TE เป็น GND โดยข้อมูลจะส่งไปที่ตัวรับและจะแสดงข้อมูลที่ขา D8-D11 ของ HT12D

ผลการทดลอง

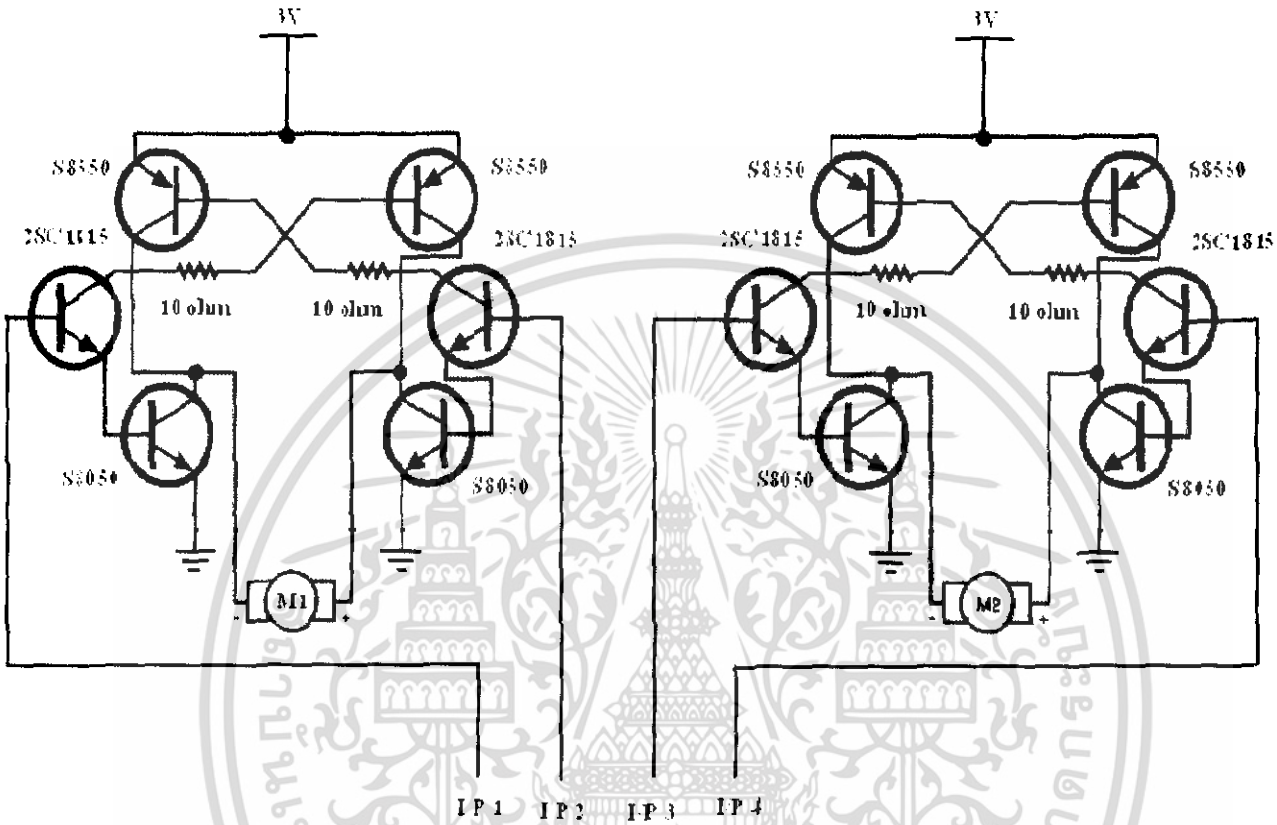
ส่ง				รับ			
D0	D1	D2	D3	D0	D1	D2	D3
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1
0	0	0	0	1	0	0	0
1	0	0	1	1	0	0	1
1	0	1	0	1	0	1	0
1	1	1	1	1	1	1	1

ตารางที่ 8.1 แสดงผลการรับ-ส่ง

สรุปผลการทดลอง

ข้อมูลที่ตัวรับตรงกับข้อมูลที่ได้ส่งไปแสดงว่าสามารถรับ-ส่งทาง RF ได้

การทดลองที่ 3 การควบคุม Motor ด้วยวงจร Transistor



รูปที่ 8.6 วงจรควบคุม Motor

วัตถุประสงค์

เพื่อทดสอบการควบคุมการทำงานของ Motor

ขั้นตอนการทดลอง

1. ต่อวงจรตามรูปที่ 8.6
2. ทำการป้อน Input 1-4 ด้วยข้อมูล 4 บิต
3. จ่าย Supply ให้กับวงจร

ผลการทดลอง

I/P				Motor 1(I/P1,I/P2)	Motor 2(I/P3,I/P4)
I/P1	I/P2	I/P3	I/P4		
0	1	0	1	หมุนตามเข็ม	หมุนตามเข็ม
1	0	1	0	หมุนทวนเข็ม	หมุนทวนเข็ม
0	1	1	0	หมุนตามเข็ม	หมุนทวนเข็ม
1	0	0	1	หมุนทวนเข็ม	หมุนตามเข็ม

ตารางที่ 8.2 แสดงผลการควบคุม Motor

สรุปผลการทดลอง

สามารถควบคุม Motor ได้ด้วยการป้อน I/P ให้เก่งจร

การทดลองที่ 4 ความคลาดเคลื่อนของพิกัดในการหยุดของหุ่นยนต์

วัตถุประสงค์

เพื่อหาความคลาดเคลื่อนในการให้หุ่นวิ่งไปหาและจอดในพิกัดที่ต้องการเพื่อทำการปรับเทียบ

ขั้นตอนการทดลอง

1.เปิดโปรแกรม Delphai7 แล้วเปิดโปรแกรมการทดลอง(ในที่นี้จะทดลองเฉพาะฟังก์ชั่น Gotopoint เท่านั้น)

2.นำหุ่นไปวางในสนามในจุดที่ต้องการ

3.จับสีหัวหุ่น และสีท้ายหุ่น หลังจากนั้นกำหนดจุดที่ต้องการให้หุ่นไป

4.กดปุ่มเริ่มการทำงานหลังจากนั้นรอให้หุ่นยนต์จอด จากนั้นจึงวัดระยะคลาดเคลื่อนระหว่างจุดที่กำหนดและจุดที่หุ่นจอด บันทึกค่า

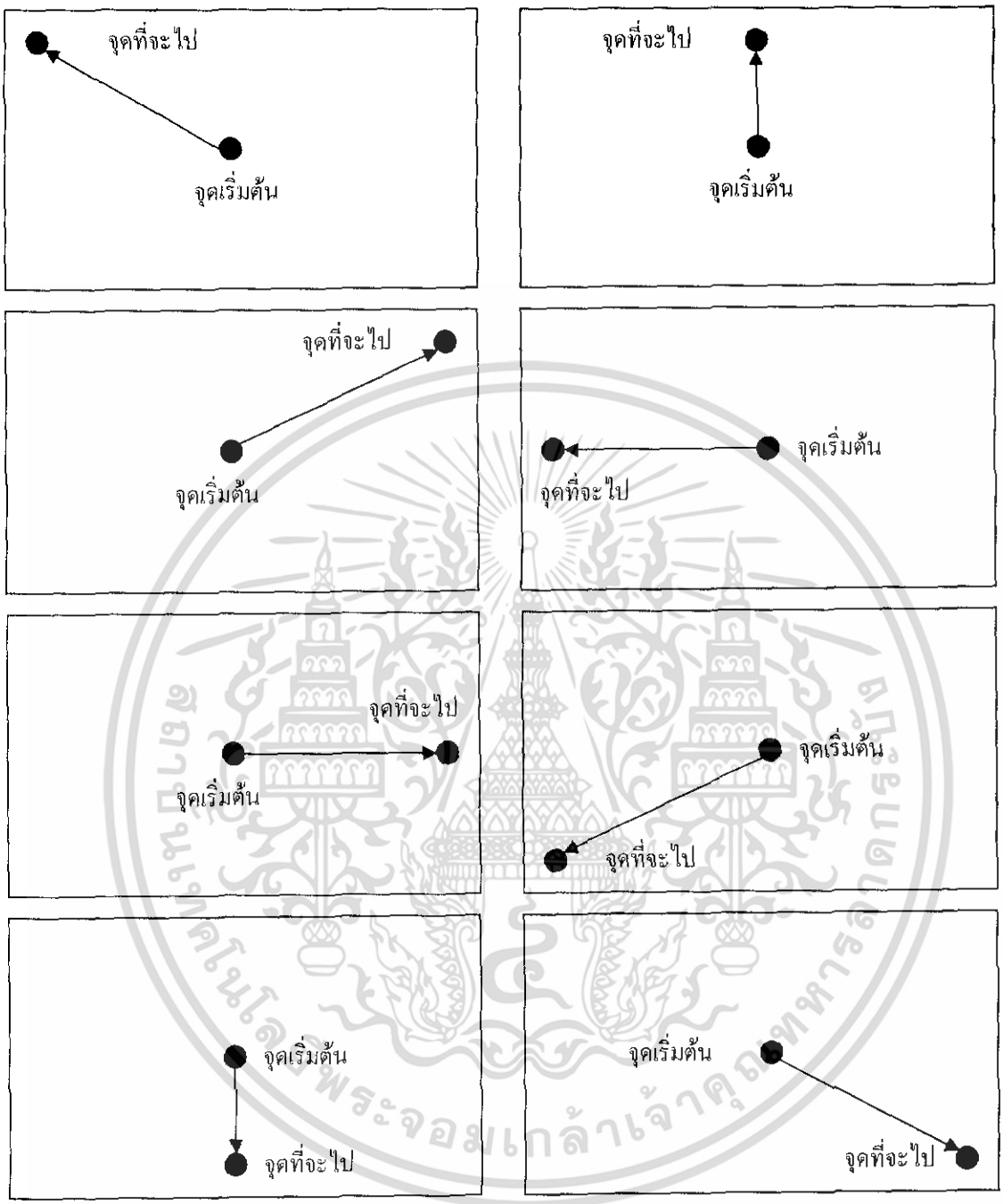
5.ทำตามขั้นตอนที่ 1-4 อีก 10 ครั้ง

6.เปลี่ยนจุดพิกัดที่จะไปตามรูป

ผลการทดลอง

ครั้งที่		1	2	3	4	5	6	7	8	9	10
ระยะคลาดเคลื่อน (cm)	บนซ้าย	5	3.8	8	7.5	0	2.5	1	4.5	4.4	5.3
	บนกลาง	10	9.5	7.4	6	8	12	10	7	5.5	9.5
	บนขวา	3	5.3	4	2.5	6	3.5	5	0	4.4	2
	ซ้าย	1	2.2	3	4.5	5	3	1	1	0	1.5
	ขวา	3	1	2.2	3.4	1	0	0.5	2	3	3
	ล่างซ้าย	7.5	8	8.5	6	4.4	3.2	7	10	12.6	7.7
	ล่างขวา	12.7	11.5	6.5	4	12	13.6	7	8.8	9.5	10

ตารางที่ 8.3 ผลการทดลองความคลาดเคลื่อนในการจอดหุ่น



รูปที่ 8.7 แสดงจุดเริ่มต้นและจุดที่จะไปตามการทดลอง

สรุปผลการทดลอง

จากการทดลองจะพบว่าระยะในการจอดของหุ่นในจุดที่ต้องการนั้นจะมีความคลาดเคลื่อนอยู่ในระยะหนึ่งแต่ว่าเป็นระยะที่สามารถยอมรับได้หากเรากำหนดจุดที่หุ่นจะไปเป็นพิกัดของบอลแทนและออกแบบระบบจับบอลที่สามารถครอบพิกัดของบอลได้หมดก็จะทำให้หุ่นวิ่งไปหาบอลและเลี้ยงบอลได้

การทดลองที่ 5 ความแม่นยำในการยิงประตู

วัตถุประสงค์

เพื่อหาระยะยิงที่เหมาะสมในการกำหนดเป็นจุดยิงที่แม่นยำ

ขั้นตอนการทดลอง

1.เปิดโปรแกรม Delphai7 แล้วเปิดโปรแกรมการทดลอง(ในที่นี้จะทดลองเฉพาะฟังก์ชั่น Shoot เท่านั้น)

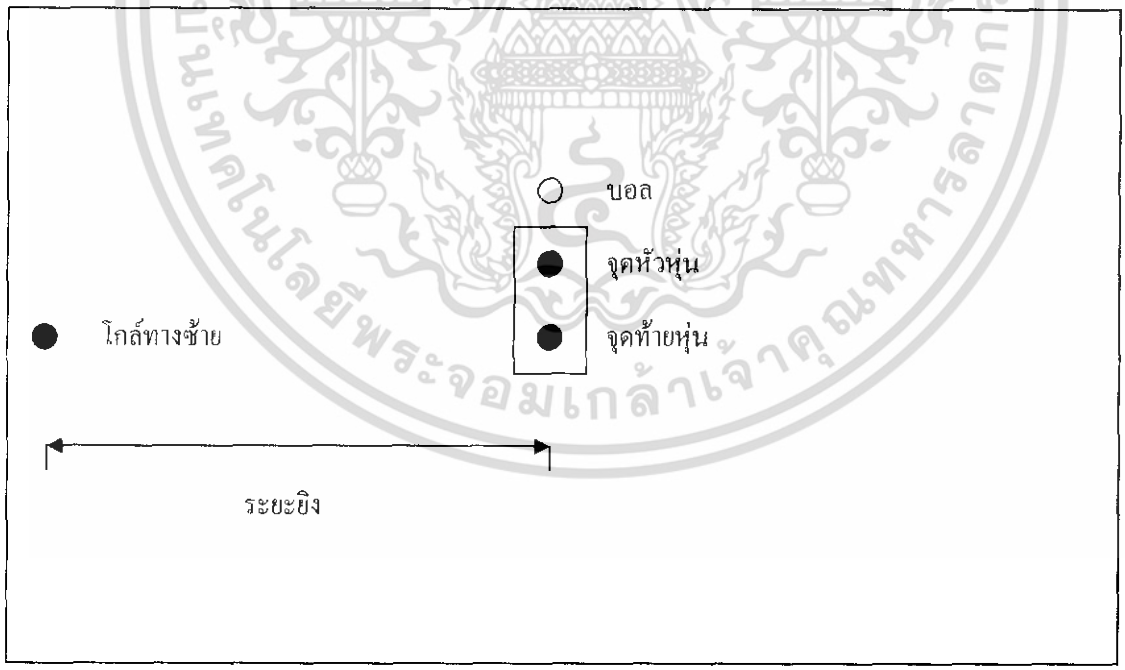
2.นำหุ่นไปวางในสนามในจุดที่ต้องการตามระยะห่างจากบอลที่กำหนด

3.จับสีหัวหุ่น สีท้ายหุ่น และสีของบอลหลังจากนั้นกำหนดจุดที่ต้องการให้หุ่นยิง(โกล์)

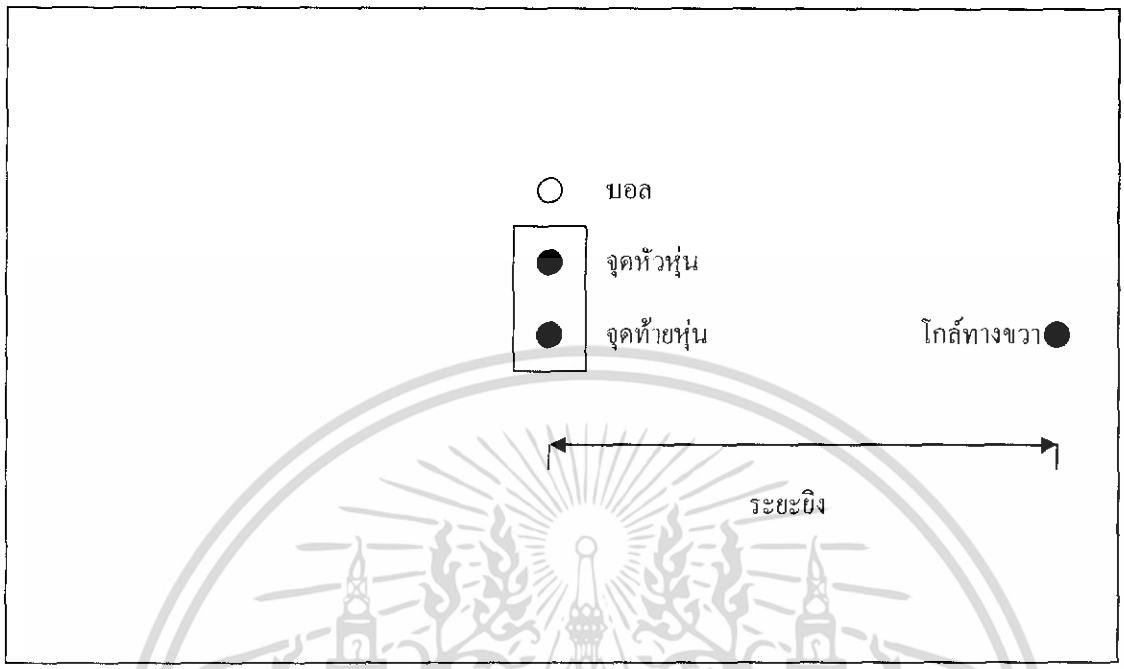
4.กดปุ่มเริ่มการทำงานหลังจากนั้นรอให้หุ่นยนต์ยิงประตู จากนั้นจึงวัดระยะคลาดเคลื่อนระหว่างจุดที่กำหนดและจุดที่บอลผ่านในระนาบเดียวกับ บันทึกค่า

5.ทำตามขั้นตอนที่ 1-4 อีก โดยเปลี่ยนเป็นหุ่นอีกตัวหนึ่ง

6.ทำตามขั้นตอนที่ 1-5 อีก โดยเปลี่ยนจุดไปอยู่ในอีกข้างของสนาม (เนื่องจากมีการผิดเพี้ยนของภาพที่เกิดจากกล้องจึงต้องมีการทดลองทั้งสองฝั่งสนามเพื่อให้ได้ระยะยิงที่เหมาะสม



รูปที่ 8.8 แสดงการทดลองยิงลูกกับโกล์ทางซ้าย



รูปที่ 8.9 แสดงการทดลองยิงลูกกับโกล์ทางขวา

ผลการทดลอง

ระยะยิง (cm)	ความคลาดเคลื่อนหุ่นตัวที่ 1 (cm)		ความคลาดเคลื่อนหุ่นตัวที่ 2 (cm)	
	โกล์ทางซ้าย	โกล์ทางขวา	โกล์ทางซ้าย	โกล์ทางขวา
50	20	29	35	30
45	16	28.5	32.5	27
40	14	25	25.5	24
35	14	20	20	24
30	14	20	17.5	20
25	12	15	14	20
20	12	11	13	18
15	4	5	0	0

ตารางที่ 8.4 ผลการทดลองความแม่นยำของการยิง

สรุปผลการทดลอง

จากการทดลองเราจะพบว่าจุดพิกัดของโกล์ทั้งสองฝั่งนั้นให้ความแม่นยำในระยะการยิงที่ไม่เท่ากัน ทั้งนี้เนื่องมาจากความผิดเพี้ยนของภาพอันเกิดจากเลนส์กล้อง โดยเห็นได้ชัดจากภาพที่ปรากฏบนจอมอนิเตอร์ ซึ่งมีลักษณะบวมตรงกลางนั้นเอง

หุ่นตัวที่ 2 จะมีความแม่นยำใกล้เคียงกับหุ่นตัวที่ 1 แต่สิ่งที่มีผลการทบต่อความแม่นยำอีกอย่างนอกจากความผิดเพี้ยนของภาพคือ ความเร็วในการหมุนตัวโดยหุ่นตัวที่มีความเร็วในการหมุนตัวยิงน้อยจะมีความแม่นยำสูงกว่าหุ่นตัวที่มีความเร็วในการหมุนมาก แต่ทั้งนี้ต้องแลกมาด้วยสปีดที่ลดลงเช่นกัน ส่วนระยะห่างระหว่างหุ่นกับพิกัดที่จะยิงก็มีผลกระทบอย่างมากเช่นกันกับความแม่นยำ โดยในการทดลองเห็นได้ว่ายิ่งระยะห่างระหว่างหุ่นกับพิกัดที่จะยิงมีค่ายิ่งน้อยความแม่นยำจะยิ่งมากขึ้น โดยการทดลองในครั้งนี้ได้ระยะยิงที่มีค่ายอมรับได้คือตั้งแต่ระยะห่าง 15 cm ลงไป ซึ่งเราจะนำไปกำหนดเป็นระยะยิงต่อไป

บทที่ 9
บทสรุปและวิจารณ์

หุ่นยนต์เตะฟุตบอลนี้จะเป็นการนำเอาการประมวลผลสีภาพโดยใช้โปรแกรมซึ่งเขียนโดย Delphi 7 เป็นโปรแกรมประมวลผลโดยในที่นี้เราจะต้องมีการตรวจจับสีที่ส่วนหัวและส่วนท้ายของหุ่น เพื่อที่จะให้หุ่นวิ่งไปยังจุดที่เราต้องการ โดยในที่นี้จะใช้กล้อง CCD เป็นตัวตรวจจับภาพที่ได้แล้วส่งผลที่ได้จากการประมวลผลโดยคอมพิวเตอร์ไปให้ไมโครคอนโทลเลอร์เป็นตัวสั่งการทำงานของส่วน Hard Ware อีกต่อหนึ่ง ซึ่งจะสั่งงานในรูปแบบไร้สาย ดังนั้นเพื่อให้สัญญาณที่ส่งแบบไร้สายนั้นมีความถูกต้องมากขึ้นจึงต้องผ่านการ Encode และต่อมาจึงทำการ Decode สัญญาณที่ส่งมาจากภาคส่งที่ภาครับเพื่อทำเพื่อนำสัญญาณไปควบคุมการทำงานของหุ่นต่อไปซึ่งในที่นี้คือใช้ควบคุมมอเตอร์ให้วิ่งตรง เดินหน้า เลี้ยวซ้าย เลี้ยวขวา ถอยหลัง หรือหยุดการเคลื่อนที่

การควบคุมและประมวลผลแบบนี้อาจจะนำไปประยุกต์ในงานด้านอื่นๆ ได้อีก เช่น ใช้ในกรณีควบคุมการทำงานของหุ่นยนต์กู้ภัย หรือการควบคุมอุปกรณ์ที่ต้องมีการเคลื่อนที่ แต่จุดที่อุปกรณ์ทำงานอยู่นั้นมนุษย์ไม่สามารถเข้าไปด้วยตัวเอง หรืออาจจะใช้ในกรณีการควบคุมที่มีการเคลื่อนที่แบบอัตโนมัติแบบต่างๆ

ในการควบคุมหุ่นยนต์เตะฟุตบอลเราจะใช้กล้อง CCD เป็นตัวตรวจจับวัตถุโดยอาศัยการตรวจจับสีที่ส่วนหัวและส่วนท้ายของหุ่น ดังนั้นถ้าหากสีที่เราใช้ในการตรวจจับมีความกลมกลืนกับพื้นสนามหรือขอบสนามภายนอกที่กล้องจับได้ ก็อาจจะทำให้กล้องจับสีเพื่อประมวลผลหาส่วนหัวและส่วนท้ายของหุ่นผิดพลาดได้ อีกประเด็นหนึ่งที่อาจทำให้เกิดความผิดพลาดในการจับสีก็คือแสง โดยหากพื้นที่ในส่วนที่กล้องจับภาพได้มีความเข้มของแสงไม่เท่ากัน ก็จะทำให้สีในแต่ละส่วนที่แสงไม่เท่ากันมีความเข้มไม่เท่ากัน ดังนั้นเมื่อนำภาพไปประมวลผลก็อาจจะเกิดความผิดพลาดได้เช่นเดียวกัน แต่ทางแก้ที่ช่วยลดผลกระทบของทั้งสองกรณีก็ยังพอมี คือในการเขียนโปรแกรมประมวลผลสีที่ส่วนหัวและส่วนท้ายของหุ่น ให้เราทำการโปรแกรมเผื่อค่าผิดพลาดที่ความเข้มของสีอาจจะคลาดเคลื่อนไป ก็จะพอช่วยแก้ไขในปัญหาทั้งสองได้ ปัญหาอีกอย่างหนึ่งก็คือความเร็วในการเคลื่อนที่ของหุ่นถ้าหากหุ่นมีความเร็วในการเคลื่อนที่มากจนเกินไป ก็อาจจะทำให้เวลาหุ่นเปลี่ยนรูปแบบการเคลื่อนที่ไม่ทันเช่น เมื่อหุ่นเลี้ยวขวาอยู่แต่เมื่อเลี้ยวถึงจุดที่ควรจะเปลี่ยนเป็นวิ่งตรง หุ่นอาจเปลี่ยนการทำงานตามการประมวลผลไม่ทันจึงอาจจะหมุนเลยจากขอบเขตการวิ่งตรงกลายเป็นเลี้ยวว้ายได้

เอกสารอ้างอิง

1. อภิชาติ ภู่พลับ, สัจจะ จรัสรุ่งรวีวง, "เริ่มต้นเขียนโปรแกรมติดต่อและควบคุมฮาดแวร์ด้วย Visual Basic"
2. สัจจะ จรัสรุ่งรวีวร, จักรพงษ์ สุขประเสริฐ, "เริ่มต้นอย่างมืออาชีพด้วย Delphi 7 "
3. ประกิจ ตังติสานนท์ , "วิศวกรรมการสื่อสาร (ไฟฟ้า-อิเล็กทรอนิกส์)" , คณะวิศวกรรมศาสตร สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พิมพ์ครั้งที่ 4 กรุงเทพ 2539
4. ประจิน พลังสันติกุล , "เรียนรู้และใช้งาน CCS C Compiler เขียนโปรแกรมภาษา C ควบคุมไมโครคอนโทรลเลอร์"
5. ผศ.ดร. ชูชาติ ปิณฑวิรุจน์ , "เอกสารประกอบการสอน วิชา Digital Image Processing"

ภาคผนวก

1. โปรแกรมควบคุม PIC16F628 โดยใช้โปรแกรม CCS C Compiler โดยมี Source code ดังนี้

```
#include <16F628.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT,nomclr
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_B2, rcv=PIN_B1)
#define BUFFER_SIZE 32
byte buffer[BUFFER_SIZE];
byte next_in = 0;
byte next_out = 0;
#int_rda
void serial_isr() {
int t;
buffer[next_in]=getc();
t=next_in;
next_in=(next_in+1) % BUFFER_SIZE;
if(next_in==next_out)
next_in=t;
}
#define bkbhit (next_in!=next_out)
byte bgetc() {
byte c;
while(!bkbhit) ;
c=buffer[next_out];
next_out=(next_out+1) % BUFFER_SIZE;
return(c);
}
```

```
main() {
char ch;
enable_interrupts(global);
enable_interrupts(int_rda);
do {
while(bkbhit)
ch =  bgetc() ;
switch(ch) {
case'1': printf("\r\nBuffered data => %c",ch);
output_high (PIN_A1);
output_high (PIN_B5);
output_low (PIN_A0);
output_low (PIN_B4)
break;
case'2': printf("\r\nBuffered data => %c",ch);
output_high (PIN_A1);
output_low (PIN_B5);
output_low (PIN_A0);
output_low (PIN_B4);
break;
case'3': printf("\r\nBuffered data => %c",ch);
output_low (PIN_A1);
output_low (PIN_B5);
output_low (PIN_A0);
output_low (PIN_B4);
break;
case'4': printf("\r\nBuffered data => %c",ch);
output_low (PIN_A1);
```

```
output_high (PIN_B5);
output_low (PIN_A0);
output_low (PIN_B4);
break;
case'5': printf("\r\nBuffered data => %c",ch);
output_low (PIN_A1);
output_low (PIN_B5);
output_high (PIN_A0);
output_high (PIN_B4);
break;
}
ch=0;
} while (TRUE);
}
```

2. โปรแกรม Del Phi เพื่อติดต่อกับ พอร์ตอนุกรม

```
unit Robot;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, CPort, StdCtrls, CPortCtl;
type
    TForm1 = class(TForm)
     Up: TButton;
     Stop: TButton;
     Right: TButton;
     Left: TButton;
     Down: TButton;
     ComPort: TComPort;
     Start: TButton;
     Setup: TButton;
     LedUp: TComLed;
     LedLeft: TComLed;
     LedStop: TComLed;
     LedRight: TComLed;
     LedDown: TComLed;
     Memo: TMemo;
procedure StartClick(Sender: TObject);
procedure SetupClick(Sender: TObject);
procedure AfterOpen(Sender: TObject);
procedure AfterClose(Sender: TObject);
procedure UpClick(Sender: TObject);
procedure LeftClick(Sender: TObject);
```

```
procedure StopClick(Sender: TObject);
procedure RightClick(Sender: TObject);
procedure DownClick(Sender: TObject);
procedure OnRxChar(Sender: TObject; Count: Integer);
    private
      { Private declarations }
    public
      { Public declarations }
    end;
var
    Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.StartClick(Sender: TObject);
begin
 if ComPort.Connected then
   ComPort.Close
 else
   ComPort.Open;
end;
procedure TForm1.SetupClick(Sender: TObject);
begin
 ComPort.ShowSetupDialog;
end;
procedure TForm1.AfterOpen(Sender: TObject);
begin
 Start.Caption := 'ËÂØ´¡ÒÃ·Ó§Ò';
 Up.Enabled := true;
```

```
  Left.Enabled := true;
  Stop.Enabled := true;
  Right.Enabled := true;
  Down.Enabled := true;
  Setup.Enabled :=false;
end;
procedure TForm1.AfterClose(Sender: TObject);
begin
  Start.Caption := 'àÃÔèÁ¡ÒÃ·Ó§Ò¹';
  Up.Enabled := false;
  Left.Enabled := false;
  Stop.Enabled := false;
  Right.Enabled := false;
  Down.Enabled := false;
  Setup.Enabled :=true;
  LedUp.State := (lsOff);
  Led1Left.State := (lsOff);
  LedStop.State := (lsOff);
  LedRight.State := (lsOff);
  LedDown.State := (lsOff);
end;
procedure TForm1.UpClick(Sender: TObject);
  var
  Str: String;
begin
  Str := '1';
  ComPort.WriteStr(Str);
  LedUp.State := (lsOn);
```

```
  LedLeft.State := (lsOff);
  LedStop.State := (lsOff);
  LedRight.State := (lsOff);
  LedDown.State := (lsOff);
end;
procedure TForm1.LeftClick(Sender: TObject);
  var
  Str: String;
begin
  Str := '2';
  ComPort.WriteStr(Str);
  LedUp.State := (lsOff);
  LedLeft.State := (lsOn);
  LedStop.State := (lsOff);
  LedRight.State := (lsOff);
  LedDown.State := (lsOff);
end;
procedure TForm1.StopClick(Sender: TObject);
 var
  Str: String;
begin
  Str := '3';
  ComPort.WriteStr(Str);
  LedUp.State := (lsOff);
  LedLeft.State := (lsOff);
  LedStop.State := (lsOn);
  LedRight.State := (lsOff);
  LedDown.State := (lsOff);
```

```
end;
procedure TForm1.RightClick(Sender: TObject);
 var
  Str: String;
begin
  Str := '4';
  ComPort.WriteStr(Str);
  LedUp.State := (lsOff);
  LedLeft.State := (lsOff);
  LedStop.State := (lsOff);
  LedRight.State := (lsOn);
  LedDown.State := (lsOff);
end;
procedure TForm1.DownClick(Sender: TObject);
 var
  Str: String;
begin
  Str := '5';
  ComPort.WriteStr(Str);
  LedUp.State := (lsOff);
  LedLeft.State := (lsOff);
  LedStop.State := (lsOff);
  LedRight.State := (lsOff);
  LedDown.State := (lsOn);
end;
procedure TForm1.OnRxChar(Sender: TObject; Count: Integer);
var
 Str: String;
```

```
begin
  ComPort.ReadStr(Str, Count);
  Memo.Text := Memo.Text + Str;
end;
end.
```