หุ่นยนต์ควบคุมผ่านระบบเครือข่ายคอมพิวเตอร์ไร้สาย

WirelessLAN Robot

โดย

นายอัครพล ลิ่มอภิชาต

นายอาทิตย์ โพธิ์เย็น

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

WirelessLAN Robot

BY

Mr.AHKHARAPHOL LIMAPICHAT

Mr. ARTIT PHOYEN

A THESIS SUBMITTED IN PARTIAL FULFILMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF THE INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2006

หัวข้อปริญญานิพนธ์	หุ่นยนต์ควบคุมผ่านระบบเครือข่ายคอมพิวเตอร์ไร้สาย		
THESIS TITLE	WirelessLAN Robot		
ชื่อนักศึกษา	นายอัครพล ถิ่มอภิชาต	รหัสประจำตัว	46012213
	นายอาทิตย์ โพธิ์เย็น	รหัสประจำตัว	46012214
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.ปิติเขต สู้รักษา		
	ผศ. บุณย์ชนะ ภู่ระหงษ์		
ระดับการศึกษา	ปริญญาวิศวกรรมศาสตรบัณฑิต		
ภาควิชา	วิศวกรรมสารสนเทศ		
ปีการศึกษา	2549		

ปริญญานิพนธ์ฉบับนี้ได้รับการอนุมัติเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

.....

(รศ.ดร.ปิติเขต สู้รักษา)

อาจารย์ผู้ควบคุมวิทยานิพนธ์

.....

(ผศ. บุณย์ชนะ ภู่ระหงษ์)

อาจารย์ผู้ควบคุมวิทยานิพนธ์

หัวข้อปริญญานิพนธ์	**หุ่นยนต์ควบคุมผ่านระบบเครือข่ายคอมพิวเตอร์ไร้สาย**		
THESIS TITLE	WirelessLAN Robot		
ชื่อนักศึกษา	**นายอัครพล ลิ่มอภิชาต**	**รหัสประจำตัว**	**46012213**
	นายอาทิตย์ โพธิ์เย็น	**รหัสประจำตัว**	**46012214**
อาจารย์ผู้ควบคุมวิทยานิพนธ์	**รศ.ดร.ปิติเขต สู้รักษา**		
	ผศ. บุญยิ่งชนะ ภู่ระหงษ์		
ระดับการศึกษา	**ปริญญาวิศวกรรมศาสตร์บัณฑิต**		
ภาควิชา	**วิศวกรรมสารสนเทศ**		
ปีการศึกษา	**2549**		

บทคัดย่อ

โครงงานนี้เป็นการประดิษฐ์หุ่นยนต์ค้นหาผู้ประสบภัยในสถานที่ที่มนุษย์อาจเข้าไปถึงการทำงานจะแยกออกเป็น สามส่วนคือ หนึ่งหุ่นยนต์จะมีระบบการประมวลผลสัญญาณภาพจากกล้องโดยจะนำเอาสัญญาณมาประมวลผลเพื่อตรวจจับวัตถุที่มีการเคลื่อนไหวในส่วนที่สองเป็นการควบคุมความเร็วมอเตอร์ ด้วยทฤษฎี พีไอดี คอนโทรล ซึ่งจะใช้ควบคุมความเร็วล้อของหุ่นยนต์ ส่วนที่สามจะมีการสร้างแผนที่โดยใช้บอกพิกัดของหุ่นยนต์โดยใช้เข็มทิศดิจิตอล และ สัญญาณป้อนกลับจากมอเตอร์ เพื่อช่วยควบคุมหุ่นในระยะทางไกลผ่านระบบเครือข่ายคอมพิวเตอร์ไร้สายมาใช้ในการควบคุมหุ่นยนต์

THESIS TITLE	Wireless LAN Robot
STUDENT	Mr. Ahkharaphol Limapichat No. 46012213
	Mr. Artit Phoyen No. 46012214
ADVISOR	Assoc.Prof. Dr.Pitikhate Sooraksa
	Assoc.Prof. Boonchana Purahong
COURSE	Bachelor of Information Engineering
DEPARTMENT	Information Engineering
YEAR	2006

Abstract

In this project we build robot to help us find victim in the place human can't reach.The system of robot separated in three parts. In first part is system image processing, video signals are used in motion detection process . In second part we use PID Control theory to control speed of robot by control speed of motor. In third part is map generating to show where robot is using digital compass and return signal from motor. All of this are use to help control robot from long distance with wireless LAN system

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี ขอขอบพระคุณ อาจารย์ที่ปรึกษาที่ให้คำแนะนำและความช่วยเหลือที่ดีเสมอมาตลอดจนอีกทั้งยังได้ชี้แนะแนวทางในการแก้ปัญหาต่างๆในการทำงานวิจัยมาโดยตลอด

ขอขอบพระคุณพี่เหยี่ยว,พี่แมว,พี่ป๊อบ,และพี่ๆที่ Info-Dynamics Laboratory ที่คอยให้คำปรึกษาและชี้แนะแนวทางในการแก้ไขปัญหาต่างๆด้วยดีตลอดมา

ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ที่คอยห่วงใยและให้การสนับสนุนในการศึกษา รวมทั้งขอขอบคุณญาติสนิทและพี่ๆทุกคนที่เป็นกำลังใจพร้อมทั้งให้ความช่วยเหลือในด้านต่างๆมาโดยตลอด

สุดท้ายขอขอบพระคุณเพื่อนๆห้อง 4FS ที่คอยช่วยเหลือกันมาตลอด

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญภาพ

สารบัญภาพ(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบัน หุ่นยนต์ได้เข้ามามีบทบาทกับมนุษย์มากขึ้น ทั้งในการอำนวยความสะดวกในชีวิตประจำวัน และการเข้าทำหน้าที่ในภารกิจอันตรายต่างๆ ซึ่งหุ่นยนต์สามารถช่วยลดภาระและความเสี่ยงมนุษย์ได้เป็นอย่างดี จึงจำเป็นอย่างยิ่งที่ต้องมีการพัฒนาความสามารถของหุ่นยนต์ให้มีความสามารถมากขึ้น เพื่อจะตอบสนองความต้องการในการใช้งานของมนุษย์ การควบคุมผ่านระบบไร้สายเป็นอีกระบบหนึ่งที่นิยมนำเข้ามาใช้เพื่อช่วยในการควบคุมหุ่นยนต์ เพื่อความสะดวกในการควบคุมในระยะทางไกลๆ และ หรือเส้นทางที่มนุษย์ไม่สามารถเข้าถึงได้ ซึ่งจะทำให้สามารถปฏิบัติงานได้โดยเกิดประสิทธิภาพสูงสุด

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาและพัฒนาระบบการควบคุมหุ่นยนต์ผ่านระบบไร้สาย Wireless LAN

1.2.2 เพื่อศึกษาและวิเคราะห์ตำแหน่งเพื่อสร้างแผนที่โดยระบบ เข็มทิศ

1.2.3 เพื่อศึกษาและพัฒนาการประมวลผลทางด้านการเคลื่อนไหวของวัตถุ โดย Image Processing

1.2.4 เพื่อศึกษาและพัฒนาระบบการควบคุมความเร็วหุ่นยนต์ด้วยทฤษฎี PID Control

1.3 ขอบเขตของโครงงาน

1.3.1 สามารถควบคุมหุ่นยนต์ในระยะทางไกล ผ่านระบบ Wireless LAN อย่างมีประสิทธิภาพ

1.3.2 ในส่วนของการประมวลผลสัญญาณภาพ (Motion Detection) สามารถตรวจจับวัตถุที่กำลังเคลื่อนไหวได้โดยใช้ทฤษฎีการประมวลผลภาพ ด้วย (Image Processing)

1.3.3 สามารถสร้างแผนที่,สภาพแวดล้อมรอบๆตัวหุ่นยนต์ และพิกัดของหุ่นยนต์ได้อย่างแม่นยำ

1.4 ผลที่คาดว่าจะได้รับ

1.4.1 สามารถนำระบบควบคุมผ่าน Wireless LAN ไปประยุกต์ใช้กับอุปกรณ์ชนิดอื่นๆได้

1.4.2 ระบบของการประมวลผลและการตรวจจับวัตถุที่กำลังเคลื่อนไหวสามารถนำไปใช้กับ อุปกรณ์ชนิดอื่นๆได้

1.4.3 จากการวิเคราะห์เพื่อสร้างแผนที่โดยระบบ เข็มทิศ และการตรวจจับวัตถุที่กำลังเคลื่อนไหวโดยกล้องจะช่วยให้หุ่นยนต์ทำงานได้อย่างมีประสิทธิภาพมากขึ้น

1.4.4 ระบบการวัดอุณหภูมิสามารถวัดอุณหภูมิสิ่งแวดล้อมได้และนำไปใช้กับอุปกรณ์ชนิดอื่นๆได้

1.5 เครื่องมือที่ใช้ในการทดลอง

1.5.1ซอฟท์แวร์

1. คอมพิวเตอร์
2. Microsoft Visual Basic 6.0 Service Pack6
3. Keil uVision3 for ARM Microcontroller
4. RS232 - Program Burn Editor
5. PROTEL_99
6. Philips LPC2000 Flash Utility v2.2.3

1.5.2ฮาร์ดแวร์

1. หุ่นยนต์ 2 ล้อ
2. ไมโครคอนโทรลเลอร์ARM7 เบอร์ LPC2138
3. Ethernet IO Board
4. Access Point
5. สายสัญญาณ RF232
6. กล้อง IP Camera

บทที่ 2
ทฤษฎีและเอกสารของงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีไมโครคอนโทรลเลอร์

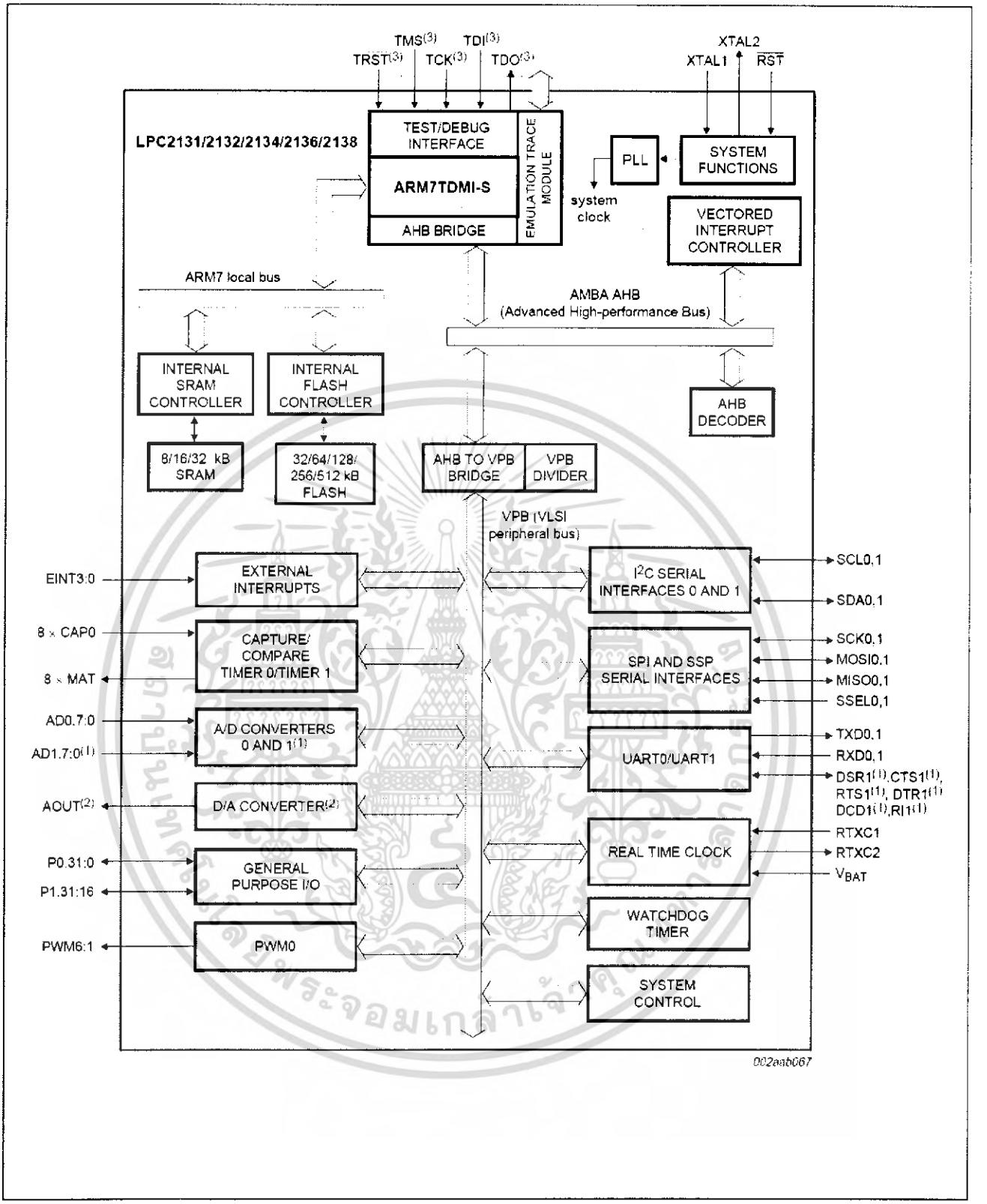
ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่รวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุท วงจรขับสัญญาณเอาท์พุท หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถนำไปใช้งานแทนวงจรอิเล็กทรอนิกส์ที่ซับซ้อนได้เป็นอย่างดี ช่วยลดจำนวนอุปกรณ์และขนาดของระบบในขณะที่มีความสามารถสูงขึ้น ภายใต้งบประมาณที่เหมาะสม

ไมโครคอนโทรลเลอร์มาจากคำ 2 คำรวมกันคือ "ไมโคร" (micro) ซึ่งหมายถึง ไมโครโปรเซสเซอร์ (microprocessor) ซึ่งเป็นอุปกรณ์ประมวลผลข้อมูลขนาดเล็ก ภายในประกอบด้วย หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit) หน่วยคำนวณทางคณิตศาสตร์และลอจิก (ALU : Arithmatic Logic Unitl) วงจรเชื่อมต่อหน่วยความจำ และวงจรสัญญาณนาฬิกา อีกคำหนึ่งคือคำว่า "คอนโทรลเลอร์" (controller) หมายถึง อุปกรณ์ควบคุม ดังนั้นไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม โดยที่สามารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างอิสระ

คุณสมบัติของไมโครคอนโทรเลอร์ตระกูล ARM7 Philips

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 16/32 บิต
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบ static RAM ขนาด 32 kb และ Flash program memory 512 kb
- รองรับการโปรแกรมแบบ IN-SYSTEM PROGRAMMING (ISP) ผ่านทาง ON-CHIP-BOOL-LOADER SOFTWARE UART 0 โดยต่อเข้ากับ PORT RS232 ของเครื่อง พีซี ได้โดยตรง
- 47 I/O PIN สามารถต่อกับระบบ I/O ที่เป็นระดับสัญญาณ 5V ได้
- Timer/Counter ขนาด 32 บิตจำนวน 2 ตัว
- UART แบบ FULL-DUPLEX จำนวน 2 ช่อง คือ UART 0 มาตรฐาน 4 PIN ETT เป็นสัญญาณระดับ RS232 และ UART1 เป็นสัญญาณระดับ TTL
- สามารถรองรับแหล่งกำเนิด interrupt ได้ 6 ประเภท

- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิพขนาด 60 MHz
- A TO D ขนาด 10 BIT จำนวน 8 ช่อง, D TO A ขนาด 10 BIT จำนวน 1 ช่อง
- PWM (Pulse width modulation) 6 เอาต์พุต, WATCHDOG TIMER, REAL TIME CLOCK ในตัว CPU
- SPI จำนวน 2 ช่อง, I2C จำนวน 2 ช่อง
- มีฟังก์ชั่นประหยัดพลังงาน
- สามารถรองรับอินเตอร์รัปต์จากภายนอกเพื่อกลับเข้าสู่โหมดพร้อมทำงาน



รูปที่ 2.1 บล็อกไดอะแกรมของไมโครคอนโทรลเลอร์ LPC2138

2.1.1 การต่อกับพอร์ตอนุกรม UART0

ในไมโครคอนโทรเลอร์ตระกูล LPC2000 มีวงจรพอร์ตสื่อสารอนุกรมอเนกประสงค์ (Universal Asynchronous Receiver Transmitter : UART) 2 วงจรที่เหมือนกัน คือ UART0 และ UART1 โดย UART1 มีขาเพิ่มเติมสำหรับติดต่อกับโมเด็ม ภายใน UART ทั้งสองตัวมีวงจรกำเนิดบอดเรตอัตโนมัติ (Baud Rate Generator) สำหรับสร้างสัญญาณนาฬิกาควบคุมจังหวะในการรับส่งข้อมูล และมีบัฟเฟอร์แบบ FIFO (First In First Out) ขนาด 14 ไบท์สำหรับหรือส่งข้อมูล

การกำหนดค่าเริ่มต้นสำหรับ UART0

ใน UART0 มีขาสัญญาณแค่สองขาคือขาสัญญาณ TxD0 ไว้สำหรับส่งข้อมูลอยู่ที่พอร์ด P0.0 และขา RxD0 ไว้สำหรับรับข้อมูลอยู่ที่พอร์ต P0.1

การใช้งาน UART0 เริ่มต้นด้วยการเขียนค่ายังรีจิสเตอร์ PINSEL0 เพื่อกำหนดให้ขา P0.0 และ P0.I มีการทำงานเป็นแบบ UART0 โดยต้องกำหนดบิตที่ 3 – 0 ของ PINSEL0 ให้มีค่าเป็น 0101 ซึ่งเขียนเป็นภาษาซีได้ดังนี้

```
PINSEL0 |= 0x000 ;
```

รีจิสเตอร์ที่เกี่ยวข้องกับ UART0 มีทั้งหมด 10 ตัวโดยแต่ละตัวมีขนาด 8 บิต

หลังจากที่กำหนดให้ขา P0.0 และ P0.1 ให้ทำงานเป็น UART0 แล้วถัดมาเป็นการกำหนดรูปแบบการติดต่อเช่น ติดต่อแบบ 8 บิต ใช้การตรวจสอบบิตที่ผิดพลาดแบบใด เช่น even parity จำนวนของ Stop bit โดยการกำหนดค่าผ่านทางรีจิสเตอร์ UART Lin Control Register : LCR ซึ่งมีรายละเอียดของรีจิสเตอร์ดังแสดงในตารางที่ 2.1

ตัวอย่างเช่น กำหนดรูปแบบการติดต่อเป็นแบบ 8 บิต ไม่ใช้พาริตี้บิต Stop bit 1บิต จะต้องเขียนค่า 0x83 ให้กับรีจิสเตอร์ U0LCR ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

```
U0LCR = 0x83 ;
```

ตารางที่ 2.1 ค่าประจำรีจิสเตอร์ UART0 Line Control Register (U0LCR)

Bit	Symbol	Value	Description	Reset Value
1:0	Word Length Select	00	5 bit character length	0
		01	6 bit character length	
		10	7 bit character length	
		11	8 bit character length	
2	Stop Bit Select	0	1 stop bit	0
		1	2 stop bit (1.5 if U0LCR[1:0] = 00)	
3	Parity Enable	0	Disable parity generation and checking	0
		1	Enable parity generation and checking	
5:4	Parity Select	00	Odd parity. Number of 1s in the transmitted Character and the attached parity bit will be odd	0
		01	Even parity. Number of 1s in the transmitted character and the attached parity bit will be even	
		10	Forced "1" stick parity	
		11	Forced "0" stick parity	
6	Break Control	0	Disable break transmission	0
		1	Enable break transmission. Output pin UART0 TXD is forced to logic 0 with U0CLR[6] is active high	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches	0
		1	Enable access to Divisor Latches	

ในรีจิสเตอร์ LCR มีบิตที่เรียกว่า DLAB (Divisor Latch Access Bit) ถ้าเราต้องการปรับค่าของวงจรกำเนิดบอดเรตต้องเซทบิตนี้ให้เป็น 1

ค่าของ Baud rate generator เป็นค่าของตัวหารขนาด 16 บิด เพื่อนำไปใช้หารค่าของ PCLK เพื่อให้ได้ความถี่ที่สูงกว่าค่าความเร็วบอดเรต 16 เท่า ทำให้ได้สมการค่าของตัวหารดังนี้

Divisor = PCLK / (16 * Baud)

ในกรณีของแผงวงจร JX – 2148 และ CP – JR ARM7 USB – LPC2148 EXP ใช้คริสตัลความถี่ 12.000 MHz สั่ง PLL คูณ 5 จะได้ CCLK = 60.0 MHz และสั่ง VPBDIV = 2 จะได้

Divisor = 30,000,000 / (16 * 9600) = 195.3125

ปัดเศษลงได้ = 195 หรือ 0xC3

ทดลองนำค่าที่ได้ไปคำนวณหาอัตราบอดจะได้

Baud = PCLK / (16 * Divisor) = 30,000,000 / (16 * 195)

= 9615 bps

ซึ่งผิดพลาดไป 0.156% สามารถใช้งานได้ เนื่องจากมาตรฐานของการสื่อสารแบบอนุกรมสามารถรับอัตราบอดที่ผิดพลาดได้ถึง 5%

เราต้องนำค่าตัวหารนี้ไปเก็บรีจิสเตอร์ขนาด 8 บิตสองตัว คือ Divisor Latch MSB (DLM) และ Divisor Latch LSB (DLL) ในขณะที่เขียนค่าเก็บในรีจิสเตอร์ DLM และ DLL นี้ค่าบิต DLAB ต้องมีค่าเป็น 1 เมื่อเขียนเสร็จแล้วต้องรีเซตค่าบิตนี้ให้กลับเป็น 0 ซึ่งเขียนเป็นคำสั่งภาษาซีได้ดังนี้

```
U0DLM = 0x00 ;
U0DLL = 0xC3 ;
U0LCR = 0x7F ;
```

หรือจะนำไปเขียนเป็นฟังก์ชั่นสำหรับกำหนดการทำงานของ UART0 ได้ดังนี้

```
void uart0_init( unsigned int baud rate)
{
    unsigned short u0dl ;
    u0dl        = 30000000 / ( 16 * baud rate ) ;
    PINSEL      |= 0x00000005 ;
    U0LCR       = 0x83 ;
    U0DLL       = u0dl & 0xFF ;
    U0DLM       = (u0dl >> 8) ;
    U0LCR       &= 0x7F ;
}
```

ในการเรียกใช้ฟังก์ชั่น uart0_init() จะต้องส่งค่าอัตราบอดที่ต้องการให้ฟังก์ชั่น ตัวอย่างเช่นต้องการอัตราบอดที่ 9600 bps จะต้องเรียกใช้ฟังก์ชั่นดังนี้ uart0_init(9600) ;

เมื่อกำหนดการทำงานให้กับ UART แล้ว จะสามารถรับส่งค่าผ่านพอร์ตอนุกรมได้ ในการส่งข้อมูลต้องเขียนข้อมูลไปยังรีจิสเตอร์ Transmit Holding Register (THR) ถ้าต้องการอ่านค่าจากข้อมูลที่รับจากพอร์ตอนุกรมต้องอ่านค่าจาดรีจิสเตอร์ Receiver Buffer Register (RBR) แสดงว่าการเขียนค่าให้กับ THR เป็นการเขียนค่าลงในรีจิสเดอร์แบบ FIFO ของ UART0 การอ่านค่าจากรีจิสเตอร์ RBR เป็นการอ่านค่าจากบัฟเฟอร์แบบ FIFO

ก่อนที่จะอ่านหรือเขียนค่าลงในรีจิสเตอร์ THR หรือ RBR จะต้องอ่านค่าสถานะของ UART ก่อนว่าผิดพลาดหรือไม่ โดยอ่านค่าจากรีจิสเตอร์ Line Status Register (LSR) ก่อน โดยค่าประจำบิตของรีจิสเตอร์แสดงได้ในตารางที่ 2.2

ตารางที่ 2.2 แสดงค่าประจำบิตของ UART0 Line Status Register (U0LSR)

Bit	Symbol	Value	Description	Reset Value
0	Receiver Data Ready (RDR)		U0LSR is set when the U0RBR holds an unread character is cleared when the UART0 RBR FIFO is empty	0
		0	U0RBR is empty	
		1	U0RBR contains valid data	
1	Overrun Error (OE)		The overrun error condition is set as soon as it occurs. An U0LSR read clears U0LSR1. U0LSR1 is set when UART0 RSR has a new character assembled and the UART0 RBR FIFO is full. In this case, the UART0 RBR FIFO will not be overwritten and the character in the UART0 RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	
2	Parity Error (PE)		When the parity bit of a received character is in the wrong state, a parity error occurs. An U0LSR read clears U0LSR[2]. Time of parity error detection is dependent on U0FCR[0]. Note: A parity error is associated with the character at the top of the UART0 RBR FIFO	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	

3	Framing Error (FE)		When the stop bit of a received character is a logic 0, a framing error occurs. An U0LSR read clears U0LSR[3]. The time of the framing error detection is dependent on U0FCR0. Upon detection of a framing error, the Rx will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error.	0
			Note: A framing error is associated with the character at the top of the UART0 RBR FIFO.	
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	Break Interrupt(BI)		When RXD0 is held in the spacing state (all 0's) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXD0 goes to marking state (all 1's). An U0LSR read clears this status bit. The time of break detection is dependent on U0FCR[0]. Note: The break interrupt is associated with the character at the top of the UART0 RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	Transmitter Holding Register Empty (THRE)		THRE is set immediately upon detection of an empty UART0 THR and is cleared on a U0THR write.	1
		0	U0THR and/or the U0TSR contains valid data.	
		1	U0THR and the U0TSR are empty.	
6	Transmitter Empty(TEMT)		TEMT is set when both U0THR and U0TSR are empty; TEMT is cleared when either the U0TSR or the U0THR contain valid data.	1
		0	U0THR and/or the U0TSR contains valid data.	
		1	U0THR and the U0TSR are empty.	

7	Error in RX FIFO(RXFE)		U0LSR[7] is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the U0RBR. This bit is cleared when the U0LSR register is read and there are no subsequent errors in the UART0 FIFO.	0
		0	U0RBR contains no UART0 RX errors or U0FCR[0]=0.	
		1	UART0 RBR contains at least one UART0 RX error.	

ก่อนที่จะเขียนข้อมูลให้ UART ต้องตรวจสอบดูที่บิต Transmitter Empty (TEMT) ของรีจิสเตอร์ LSR ก่อนว่าบัฟเฟอร์สำหรับส่งว่างหรือไม่ ถ้าว่างจะได้ค่าเป็น 1 จึงส่งข้อมูลได้ก่อนที่จะอ่านข้อมูลจาก UART จะต้องตรวจสอบบิต Receiver Data Ready (RDR) ก่อนถ้ามีข้อมูลพร้อมแล้วบิตนี้จะมีค่าเป็น 1 จึงอ่านค่าจาก UART ได้

เราสามารถนำมาเขียนเป็นฟังก์ชั่น putchar() ; สำหรับเขียนข้อมูลจำนวน 1 ไบท์ให้กับ UART และเขียนเป็นฟังก์ชั่น getchar() ; สำหรับอ่านค่าจาก UART ดังต่อไปนี้

```
int putchar (int ch)                    // Write character to Serial Port
{
        if ( ch == '\n' ){
                while (!(U0LSR & 0x20));
                U0THR = CR;             // output CR
        }
        while (!(U0LSR & 0x20));
        return (U0THR = ch);
}
int getchar (void)                      // Read character from Serial Port
{
        while (!(U0LSR & 0x01));
        return (U0RBR);
}
```

สำหรับการเขียนข้อมูลออกพอร์ทอนุกรม ในโปรแกรม Keil uVersion3 ได้จัดเตรียมฟังก์ชั่น printf() ; ไว้ให้แล้ว และถ้าต้องการรับข้อมูลจากพอร์ทอนุกรม สามารถใช้ฟังก์ชั่น scanf() ; โดยต้องสั่ง #include<stdio.h> ที่ส่วนพรีโปรเซสเซอร์ของโปรแกรมภาษาซีเมื่อศึกษาตัวฟังก์ชั่น printf() และ scanf() จะพบว่าตัวฟังก์ชั่นจะเรียกใช้ฟังก์ชั่น putchar() และ getchar()

ซึ่งผู้ใช้สามารถนำไปดัดแปลงได้เช่น ให้ฟังก์ชั่น printf() เป็นการเขียนค่าออกจากจอแสดงผล LCD ให้แก้ไขที่ตัวฟังก์ชั่น putchar() เท่านั้น

2.1.2. การต่อกับวงจรแปลงอนาล็อกเป็นดิจิตอล

ในไมโครคอนโทรลเลอร์ตระกูล LPC2000 จะมีวงจรแปลงอนาล็อกเป็นดิจิตอลซึ่งต่อไปจะใช้อักษรย่อว่า A/D ความละเอียด 10 บิต แบบ successive approximation อย่างน้อยหนึ่งวงจร โดยมีความเร็วสูงสุดในการแปลงสัญญาณถึง 2.44 µs หรือคิดเป็นความเร็ว 410 ksps

ภายใน LPC2148 มีวงจร A/D จำนวน 2 วงจรโดยแบ่งเป็น AD0 มีขาอินพุท 6 ขา คือ AD0.7 – AD0.6 และ AD0.4 – AD0.1 ส่วน AD1 มีขาอินพุท 8 ขาคือ AD1.7 – AD1.0 ทำให้มีขาต่อสำหรับวัดแรงดันอนาล็อกได้สูงสุดถึง 14 ขา ซึ่งต้องกำหนดการทำงานที่รีจิสเตอร์ PINSEL0 และ PINSEL1

การกำหนดค่าเริ่มต้นให้กับวงจรแปลงอนาล็อกเป็นดิจิตอล

ในการทำงานของวงจรแปลงอนาล็อกเป็นดิจิตอลจำเป็นต้องมีสัญญาณนาฬิกาควบคุมจังหวะการทำงาน โดยในไมโครคอนโทรลเลอร์ตระกูล LPC2000 จะนำค่าความถี่ของ PCLK มาหารค่าให้ได้ความถี่สูงสุดไม่เกิน 4.5 MHz ค่าตัวหารความถี่นี้จะเก็บค่าในฟิลด์ CLKDIV โดยนำค่ามาบวกหนึ่ง ดังนั้นจะได้สมการของค่า CLKDIV ดังนี้

CLKDIV = (PCLK / Adclk) – 1

ตัวอย่างในแผงวงจร JX-2148 หรือ CP-JR ARM7 USB-LPC2148 ใช้คริสตัลความถี่ 12.000 MHz ใช้วงจร PLL คูณค่าความถี่ได้ CCLK = 60.0 MHz กำหนดค่าหารความถี่ได้ PCLK = 30.0 MHz ดังนั้นจะได้ค่า

CLKDIV = (30.0 / 4.5) – 1 = 5.6667 ปัดเศษขึ้นได้ = 6

ในกรณีนี้จะได้ความถี่ของ

Adelk = 30.00 / (6 + 1) = 4.286 ?ผ

ในการแปลงสัญญาณจะใช้สัญญาณนาฬิกา Adclk จำนวน 11 ลูก ดังนั้นจะได้ความเร็วในการสุ่มสัญญาณสูงสุดเท่ากับ

4.286 MHz / 11 = 389 kSps

หลังจากคำนวณได้ค่า CLKDIV แล้วให้นำไปเขียนลงในรีจิสเตอร์ที่เกี่ยวข้องกับการแปลงอนาล็อกเป็นดิจิตอลซึ่งมีทั้งหมด 13 ตัวดังแสดงในตารางที่ 2.3

ในการใช้วงจร A/D อย่างง่ายสามารถใช้รีจิสเตอร์แค่ 2 ตัวคือ A/D Control Register (ADCR) สำหรับอ่านค่าการทำงานวงจร A/D และ A/D Global Data Register (ADGDR) ในการ

อ่านผลลัพธ์ของการแปลง A/D ซึ่งจะเป็นการใช้งานในโหมดเก่าทำให้สามารถนำโปรแกรมไปใช้กับไมโครคอนโทรลเลอร์ LPC2000 รุ่นเก่าได้หรือจะอ่านผลลัพธ์ของการเปลี่ยนแปลง A/D แต่ละแชนแนลได้จากรีจิสเตอร์ ADDR0 – ADDR7 ก็ได้

ตารางที่ 2.3 รีจิสเตอร์ที่เกี่ยวข้องกับการแปลง A/D

Generic Name	Description	Access	Reset Value	AD0 Address &Name	AD1 Address &Name
ADCR	A/D Control Register. The ADCR register must be written to select the operating mode before A/D conversion can occur.	R/W	0x0000 0001	0xE003 4004 AD0CR	0xE006 0000 AD1CR
ADGDR	A/D Global Data Register. This register contains the ADC's DONE bit and the result of the most recent A/D conversion.	R/W	NA	0xE003 4004 AD0GDR	0xE006 0004 AD1GDR
ADSTAT	A/D Status Register. This register contains DONE and OVERRUN flags for all of the A/D channels, as well as the A/D interrupt flag.	RO	0x0000 0000	0xE003 4030 AD0STAT	0xE006 000C AD1STAT
ADGSR	A/D Global Start Register. This address can be written (in the AD0 address range) to start conversions in both A/D converters simultaneously	WO	0x00	0x0E003 4008 ADGSR	
ADINTEN	A/D Interrupt Enable Register. This register contains enable bits that allow the DONE flag of each A/D channel to be included or excluded from contributing to the generation of an A/D interrupt.	R/W	0x0000 0100	0xE003 400C AD0INTEN	0xE006 000C AD1INTEN
ADDR0	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0.	RO	NA	0xE003 4010 AD0DR0	0xE006 0010 AD1DR1
ADDR1	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 1.	RO	NA	0xE003 4014 AD0DR1	0xE006 0014 AD1DR1
ADDR2	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 2.	RO	NA	0xE003 4018 AD0DR2	0xE006 0018 AD1DR2

ADDR3	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 3.	RO	NA	0xE003 401C AD0DR3	0xE006 001C AD1DR3
ADDR4	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 4.	RO	NA	0xE003 4020 AD0DR4	0xE006 0020 AD1DR4
ADDR5	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 5.	RO	NA	0xE003 4024 AD0DR5	0xE006 0024 AD1DR5
ADDR6	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 6.	RO	NA	0xE003 4028 AD0DR6	0xE006 0028 AD1DR6
ADDR7	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 7.	RO	NA	0xE003 402C AD0DR7	0xE006 002C AD1DR7

โดยจัดเรียงค่าแต่ละบิตของรีจิสเตอร์ ADCR อธิบายค่าแต่ละบิตของรีจิสเตอร์ ADCR ใน ตารางที่ 2.4

ตารางที่ 2.4 ความหมายตาละบิตของรีจิสเตอร์ ADCR

Bit	Symbol	Value	Description	Reset Value
7:0	SEL		Selects which of the AD0.7:0/AD1.7:0 pins is (are) to be sampled and converted. For AD0, bit 0 selects Pin AD0.0, and bit 7 selects pin AD0.7. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones. All zeroes is equivalent to 0x01.	0x01
15:8	CLKDIV		The VPB clock (PCLK) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 4.5 MHz. Typically, software should program the smallest value in this field that yields a clock of 4.5 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.	0

16	BURST	1	The AD converter does repeated conversions at the rate selected by the CLKS field, scanning (if necessary) through the pins selected by 1s in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1-bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that's in progress when this bit is cleared will be completed.	0
			Important: START bits must be 000 when BURST = 1 or conversions will not start.	
		0	Conversions are software controlled and require 11 clocks.	
19:17	CLKS		This field selects the number of clocks used for each conversion in Burst mode, and the number of bits of accuracy of the result in the RESULT bits of ADDR, between 11 clocks (10 bits) and 4 clocks (3 bits).	000
		000	11 clocks / 10 bits	
		001	10 clocks / 9bits	
		010	9 clocks / 8bits	
		011	8 clocks / 7bits	
		100	7 clocks / 6bits	
		101	6 clocks / 5bits	
		110	5 clocks / 4bits	
		111	4 clocks / 3bits	
20	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
21	PDN	1	The A/D converter is operational.	0
		0	The A/D converter is in power-down mode.	
23:22	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	START		When the BURST bit is 0, these bits control whether and when an A/D conversion is started:	0
		000	No start (this value should be used when clearing PDN to 0).	

		001	Start conversion now.	
		010	Start conversion when the edge selected by bit 27 occurs on P0.16/EINT0/MAT0.2/CAP0.2 pin.	
		011	Start conversion when the edge selected by bit 27 occurs on P0.22/TD3/CAP0.0/MAT0.0 pin.	
		100	Start conversion when the edge selected by bit 27 occurs on MAT0.1.	
		101	Start conversion when the edge selected by bit 27 occurs on MAT0.3.	
		110	Start conversion when the edge selected by bit 27 occurs on MAT1.0.	
		111	Start conversion when the edge selected by bit 27 occurs on MAT1.1.	
27	EDGE	1	This bit is significant only when the START field contains 010-111. In these cases: Start conversion on a falling edge on the selected CAP/MAT signal.	0
		0	Start conversion on a rising edge on the selected CAP/MAT signal.	
31:28	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

หลังจากได้ค่า CLKDIV จะต้องนำไปเขียนยังรีจิสเตอร์ ADCR ในบิตที่ 15:8 หลังจากรีเซตวงจรแปลงอนาล็อกเป็นดิจิตอลจะยังไม่ทำงานโดยจะอยู่ในโหมด Power Down โหมด เพื่อประหยัดพลังงาน และลดสัญญาณรบกวนตัวชิพภายในอันเนื่องมาจากวงจร A/D ดังนั้นก่อนจะใช้งานวงจร A/D จะต้องเปิดการทำงานของมันก่อนโดยสั่งเปิดที่บิต PDN (บิตที่ 21) ของรีจิสเตอร์ ADCR โดยต้องเซตค่าให้เป็น 1

เราสามารถกำหนดค่าความละเอียดในการแปลง A/D ได้ตั้งแต่ 4 บิตถึง 10 บิต โดยกำหนดที่ฟิลด์ CLKS ซึ่งอยู่ที่บิตที่ 19 -17 ของรีจิสเตอร์ ADCR โดยความละเอียดของการแปลง A/D มีค่าเท่ากับค่าของ CLKS -1 โดยจะกำหนดค่าได้ดังนี้ 000 = 11 clocks/ 10 บิต, 011 = 10 clocks/9 บิต,…….. , 111 = 4 clocks/4 บิต เมื่อกำหนดค่าตัวหารความถี่และกำหนดจำนวนคล็อกซึ่งก็คือความละเอียดของการแปลง A/D และเปิด Enable วงจร A/D แล้วก็พร้อมทำการแปลงอนาล็อกเป็นดิจิตอล

การทำงานของวงจร A/D มีสองโหมด ก็คือ ฮาร์ดแวร์(hardware) และซอฟท์แวร์(software)

ซึ่งกำหนดได้ที่บิต BURST ถ้าบิตนี้เป็น 0 จะทำงานในโหมดซอฟท์แวร์ ถ้าบิตนี้เป็น 1 จะทำงานในโหมดฮาร์ดแวร์

การทำงานในโหมดฮาร์ดแวร์ผู้ใช้จะต้องกำหนดจำนวนแชนแนลที่ต้องการทำการแปลง A/D และสั่งให้วงจรแปลง A/D ทำงาน วงจรจะทำการแปลง A/D ของแต่ละแชนแนลตามลำดับต่อเนื่องตลอดเวลา เมื่อทำการแปลงสมบูรณ์แล้วจะเขียนค่าที่ได้ลงในรีจิสเตอร์ ADDR และทำการอินเตอร์รัปต์แจ้งยังไมโครคอนโทรลเลอร์

การทำงานในโหมดซอฟท์แวร์ ผู้ใช้จะต้องกำหนดแชนแนลที่ต้องการทำการแปลงค่า A/D ที่บิต SEL และสั่งให้เริ่มทำการแปลงด้วยการเขียน 0x01 ไปยังบิตที่ 26 -24 (ฟิลด์ Start) ของรีจิสเตอร์ ADCR วงจร A/D จะทำการแปลงค่าเพียงแค่ครั้งเดียวและเก็บผลการแปลง A/D ลงในรีจิสเตอร์ ADDR เมื่อทำการแปลงเสร็จแล้วจะสามารถกำหนดให้วงจรทำการอินเตอรัปต์แจ้งไมโครคอนโทรลเลอร์เหมือนกับกรณีโหมดฮาร์ดแวร์หรือผู้ใช้จะวนอ่านค่าบิต DONE ในรีจิสเตอร์ ADDR เองก็ได้ การทำงานในโหมดซอฟท์แวร์ยังสามารถสั่งให้เริ่มทำการแปลง A/D เมื่อมีเหตุการณ์ของ Timer0 หรือ Timer1 เกิดขึ้นหรือสั่งให้เริ่มทำงานเมื่อพบขอบของสัญญาณที่อินพุตขา P0.16 หรือ P0.22 โดยสามารถเลือกได้ว่าจะเริ่มทำงานเมื่อพบขอบขาขึ้นหรือขอบขาลงได้ โดยกำหนดที่บิต EDGE ของรีจิสเตอร์ ADCR

เมื่อกำหนดค่าการทำงานของวงจร A/D เป็นที่เรียบร้อยแล้ว ขั้นตอนถัดมาให้อ่านค่าที่ได้จากการแปลงอนาล็อกเป็นดิจิตอล โดยอ่านค่าจากรีจิสเตอร์ ADGDR อธิบายความหมายของค่าแต่ละบิตของรีจิสเตอร์ ADGDR ได้ดังตารางที่ 2.5

ตารางที่ 2.5 ค่าแต่ละบิตของรีจิสเตอร์ ADGDR ซึ่งก็คือ AD0GDR และ AD1GDR

Bit	Symbol	Description	Reset Value
5:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:6	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the Ain pin selected by the SEL field, divided by the voltage on the VDDA pin (V/VREF). Zero in the field indicates that the voltage on the Ain pin was less than, equal to, or close to that on VSSA, while 0x3FF indicates that the voltage on Ain was close to, equal to, or greater than that on VREF.	NA
23:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
26:24	CHN	These bits contain the channel from which the RESULT bits were converted (e.g. 000 identifies channel 0, 001 channel 1...).	NA

29:27	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
30	OVERUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits. This bit is cleared by reading this register.	0
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read and when the ADCR is written. If the ADCR is written while a conversion is still in progress, this bit is set and a new conversion is started.	0

ก่อนที่จะอ่านค่าผลการแปลง A/D ต้องตรวจสอบที่บิตที่ 31 (DONE) ของรีจิสเตอร์ ADGDR ก่อนถ้าบิตนี้มีค่าเป็น 1 แสดงว่าการแปลง A/D สมบูรณ์แบบแล้ว ถัดมาให้อ่านค่าที่บิต 26 – 24 (CHN) เพื่ออ่านว่าค่าที่แปลง A/D สำเร็จนี้เป็นของแชนแนลใด ค่าข้อมูลดิจิตอลที่ได้จากวงจร A/D จะอยู่ในบิตที่ 15 – 6 ของรีจิสเตอร์ ADGDR ถ้าอ่านค่าข้อมูลดิจิตอลที่ได้จากการแปลงไม่ทันจนมีค่าใหม่มาเขียนทับ บิตที่ 30 (OVERRUN) จะมีค่าเป็น 1

ถ้าต้องการอ่านค่าผลการแปลง A/D ของแต่ละแชนแนลโดยตรงสามารทำได้โดยการอ่านค่าจากรีจิสเตอร์ ADDR (AD0DR0 – AD0R7 หรือ AD1DR0 – AD1DR7) ได้โดยตรงโดยค่าของรีจิสเตอร์เหล่านี้แสดงได้ในตารางที่ 2.6

ตารางที่ 2.6 ค่าแต่ละบิตของรีจิสเตอร์ ADDR ซึ่งก็คือ AD0DR0 – AD0RD7 หรือ AD1DR0 – AD1DR7

Bit	Symbol	Description	Reset Value
5:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
15:6	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the AIN pin, divided by the voltage on the VREF pin (V/VREF). Zero in the field indicates that the voltage on the AIN pin was less than, equal to, or close to that on VSSA, while 0x3FF indicates that the voltage on AIN was close to, equal to, or greater than that on VREF.	NA
29:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
30	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced	

		the result in the RESULT bits.This bit is cleared by reading this register.	
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read.	NA

2.1.3 การทดลองต่อกับอุปกรณ์ I^2C

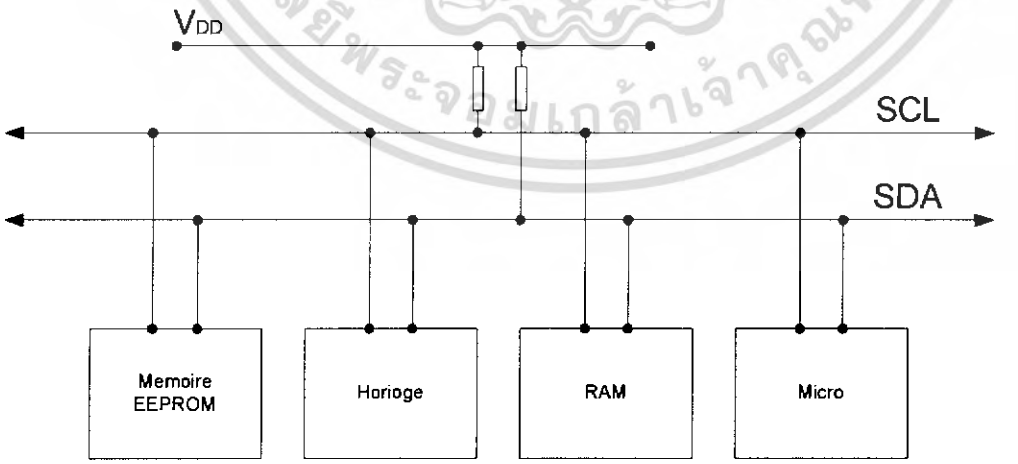
ในการรับส่งข้อมูลระหว่างอุปกรณ์ต่างๆ กับไมโครคอนโทรลเลอร์ จะมีการติดต่อได้ทั้งแบบขนาน และแบบอนุกรม โดยถ้าติดต่อแบบขนานจะต้องใช้ขาของพอร์ทจำนวนมาก ถ้าเป็นการติดต่อแบบอนุกรม จะมีข้อดีที่ใช้ขาของพอร์ทจำนวนไม่มาก แม้ว่าจะติดต่อช้ากว่าแบบขนาน

การรับส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์กับ อุปกรณ์รอบข้างแบบอนุกรมมีมาตรฐานหลายตัวจากหลายบริษัท เช่น I^2C (Inter Integrate Circuit) จากบริษัท Philips/Signatics, SPI (Serial Peripheral Interface) จากบริษัท Motorola, Microwire จากบริษัท National Semiconductors ในหัวข้อนี้จะทดลองต่อกับอุปกรณ์แบบ I^2C

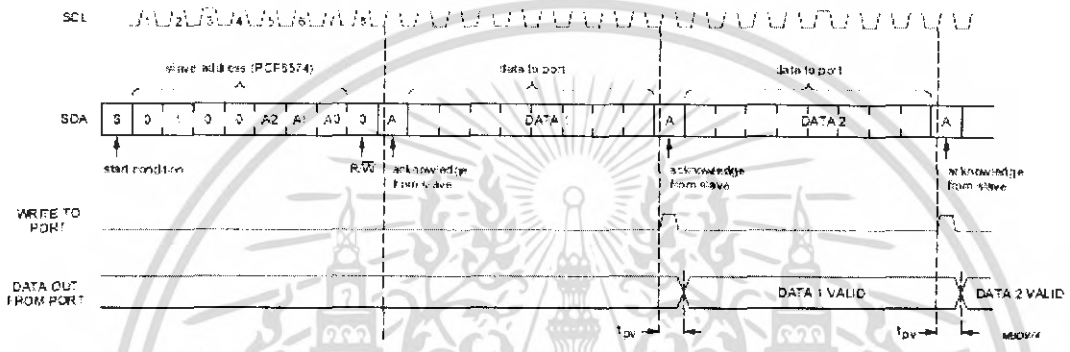
2.1.3.1 การทำงานของบัส I^2C

ระบบบัสของ I^2C จะใช้สายสัญญาณจำนวน 2 เส้น (ไม่นับสาย GND) คือ SDA (Serial Data Line) และ SCL (Serial Clock Line) โดยสัญญาณ SCL เป็นสัญญาณนาฬิกาโดยทำงานด้วยความถี่สูงสุด 400 kHz หรือต่ำกว่าขึ้นอยู่กับอุปกรณ์แต่ละตัวที่ใช้ อุปกรณ์ I^2C ทุกตัวจะต่อขนานกับสายสัญญาณทั้งสองเส้นดังแสดงระบบบัส I^2C ได้ดังรูปที่ 2.2

รูปที่ 2.2 แสดงการขัดคอนฟิกของระบบบัส I^2C



จากรูปที่ 2.2 ตัวต้านทาน R_P ที่ต่อพูลอัพกับไฟเลี้ยงจะมีค่าประมาณ 50 kΩ / จำนวนอุปกรณ์ I^2C ที่ต่อในบัส ในบัส I^2C อุปกรณ์แบ่งได้เป็น 2 ประเภท คือ (master) และสเลฟ (slave) อุปกรณ์ที่เป็นมาสเตอร์จะเป็นผู้ควบคุมบัส I^2C โดยส่งแอดเดรสเพื่อเลือกติดต่อกับอุปกรณ์ และส่งข้อมูลไปยังอุปกรณ์ตัวที่ต้องการติดต่อ อุปกรณ์ที่เป็นสเลฟจะคอยอ่านข้อมูลในบัสว่าแอดเดรสในบัสเป็นของตัวเองหรือไม่ ถ้าใช่ก็จะส่งสัญญาณตอบกลับ (ACK) กลับไปยังมาสเตอร์ และคอยรับส่งข้อมูลกับมาสเตอร์ ในไมโครคอนโทรลเลอร์ LPC2148 สามารถกำหนดให้เป็นได้ทั้งมาสเตอร์ หรือสเลฟ ในหัวข้อนี้จะทดลองใช้ LPC2148 ทำงานเป็นมาสเตอร์เท่านั้น



รูปที่ 2.3 สัญญาณของบัส I^2C และสถานะต่างๆ

ระบบบัส I^2C มีสถานะของบัสอยู่ 5 สถานะ ดังแสดงในรูปที่ 2.3

1. บัสว่าง (bus not busy) สัญญาณ SDA และ SCL มีค่าเป็นลอจิก 1 ทั้งคู่
2. เริ่มส่งข้อมูล (Start) เมื่อบัสว่างแล้ว มาสเตอร์ต้องการส่งข้อมูลจะเริ่มต้นด้วยการที่ให้ SDA เป็นลิจิก 0 ตามด้วยการสั่งให้ SCL เป็นลอจิก 0
3. หยุดการส่งข้อมูล (stop) เมื่อมาสเตอร์ต้องการหยุดการรับส่งข้อมูล สัญญาณ SCL มีค่าเป็น 1 แล้วสั่งให้ SDA เปลี่ยนค่าจาก 0 เป็น 1 เมื่อ SDA และ SCL เป็น 1 ทั้งคู่จะอยู่ในสถานะบัสว่าง
4. สถานะรับข้อมูล อยู่ระหว่างสถานะเริ่มส่งข้อมูล และหยุดการส่งข้อมูล สัญญาณ SCL จะเป็นพัลส์สี่เหลี่ยมเพื่อเป็นสัญญาณนาฬิกาของระบบ การรับส่งข้อมูลจะมีขนาด 8 บิต ข้อมูลบน SDA ต้องคงที่ขณะที่ SCL เป็นลอจิก 1 และบิตข้อมูล SDA เปลี่ยนได้ในขณะที่ SCL เป็นลอจิก 0
5. สถานะตอบกลับ (acknowledge) เมื่อมาสเตอร์ส่งข้อมูลครบ 8 บิต หรือ 8 ลูกแล้วช่วง SCL ลูกที่ 9 มาสเตอร์จะหยุดควบคุมขา SDA ตัวรับข้อมูลจะดึงสัญญาณ SDA เป็นลอจิก 0 (ขณะนี้ SCL ยังคงต้องเป็น 1) ถ้ามาสเตอร์ไม่ได้รับการตอบกลับจากสเลฟ จะหยุดการรับส่งข้อมูลทันที

ระหว่างสถานะเริ่มส่ง และสถานะหยุดรับส่งข้อมูลจะรับส่งข้อมูลได้ไม่จำกัด แต่เมื่อสเลฟได้รับแอดเดรสหรือข้อมูลแล้วต้องตอบกลับมาตลอดทุกไบท์

ภายในไมโครคอนโทรลเลอร์ จะมีวงจรสร้างสัญญาณต่างๆให้ ผู้ใช้เพียงแต่เขียนค่าไปยังรีจิสเตอร์ที่เกี่ยวข้องกับ I^2C แล้วสั่งให้ส่งข้อมูล ในการรับข้อมูลเพียงแค่อ่านข้อมูลจากรีจิสเตอร์ที่เกี่ยวข้องเท่านั้น

2.1.3.2 การกำหนดค่าเริ่มต้นสำหรับการติดต่อกับอุปกรณ์ I^2C

ภายในไมโครคอนโทรลเลอร์ตระกูล LPC2000 จะมีวงจรสำหรับติดต่อกับบัส I^2C จำนวนสองวงจรคือ I^2C0 และ I^2C1 โดยขา SCL0 อยู่ที่ขา P0.2, SDA0 อยู่ที่ขา P0.3 ในการกำหนดให้ขา P0.2 และ P0.3 ทำงานเป็น I^2C จะต้องกำหนดค่ารีจิสเตอร์ PINSEL0 โดยให้บิตที่ 7 ถึงบิตที่ 4 ของ PINSEL0 มีค่าเป็น 0101 ซึ่งเขียนคำสั่งภาษาซีได้ดังนี้

```
PINSEL0 |= 0x00000050 ;
```

ถ้าต้องการใช้งาน I^2C1 ขา SCL1 อยู่ที่ขา P0.11 และ SDA1 อยู่ที่ P0.14 โดยต้องกำหนดค่าบิตที่ 29 – 28 และ 23 -22 ของ PINSEL0 ให้เป็น 11 ทั้งคู่ ซึ่งเป็นภาษาซีได้ดังนี้

```
PINSEL0 |= 0x30C00000 ;
```

หลังจากที่สั่งรีจิสเตอร์ PINSEL0 ให้ขาที่เกี่ยวข้องการทำงานเป็น I^2C แล้วถัดมาต้องกำหนดค่าให้กับรีจิสเดอร์ที่เกี่ยวข้องกับ I^2C ซึ่งมีทั้งหมด 7 ตัวดังตาราง 2.7

ตารางที่ 2.7 รีจิสเตอร์ที่เกี่ยวข้องกับ I^2C ทั้ง 7 ตัว

Name	Description	Access	Reset Value	I^2C 0 Address And Name	I^2C 1 Address And Name
I2CONSET	I2C Control Set Register. When a one is written to a bit of this register, the corresponding bit in the I2C control register is set. Writing a zero has no effect on the corresponding bit in the I2C control register.	R/W	0x00	0xE001 c000 I20CONSET	0xE004 C000 I21CONSET
I2STAT	I2C Status Register. During I2C operation, this register provides detailed status codes that allow software to determine the next action needed.	RO	0xF8	0xE001 C004 I2C0STAT	0xE005 C004 I2C1STAT
I2DAT	I2C Data Register. During master or slave transmit mode, data to be transmitted is written to this register. During master or slave receive	R/W	0x00	0xE001 C008 I2C0DAT	0xE0015 C008 I2C1DAT

	mode, data that has been received may be read from this register.				
I2ADR	I2C Slave Address Register. Contains the 7-bit slave address for operation of the I2C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the general call address.	R/W	0x00	0xE001 C00C I2C0ADR	0xE005 C00C I2C1ADR
I2SCLH	SCH Duty Cycle Register High Half Word. Determines the high time of the I2C clock.	R/W	0x04	0xE001 C010 I2C0SCLH	0xE005 C010 I2C1SCLH
I2SCLL	SCL Duty Cycle Register Low Half Word. Determines the low time of the I2C clock. I2nSCLL and I2nSCLH together determine the clock frequency generated by an I2C master and certain times used in slave mode.	R/W	0x04	0xE001 C0014 I2C0SCLL	0xE005 C0014 I2C1SCLL
I2CONCLR	I2C Control Clear Register. When a one is written to a bit of this register, the corresponding bit in the I2C control register is cleared. Writing a zero has no effect on the corresponding bit in the I2C control register.	WO	NA	0xE001 C018 I2C0CONCLR	0xE005 C018 I2C1CONCLR

รีจิสเตอร์ที่สั่งควบคุมการทำงานจะแยกเป็น 2 ตัวคือ I2CONSET ไว้สำหรับเซตค่าบิตให้เป็น 1 และ I2ONCLR ไว้ล้างค่าบิตให้เป็น 0 รีจิสเตอร์ I2SCLH และ I2SCLL ใช้กำหนดค่าความถี่ของสัญญาณคล็อกของขา SCL รีจิสเตอร์ I2DAT เป็นข้อมูลที่ต้องการอ่านหรือเขียนให้กับอุปกรณ์ I^2C ส่วน I2STAT เป็นรีจิสเตอร์รายงานสถานะต่างๆของบัส I^2C

ในการทำงานของบัส I^2C สัญญาณ SCL ที่เป็นสัญญาณนาฬิกาของบัส สามารถกำหนดค่าความถี่ได้สูงสุดถึง 400 kHz แต่ต้องระวังให้ไม่เกินที่อุปกรณ์ I^2C จะรับได้ ตัวอย่างเช่นนำบัส I^2C ต่อกับไอซี PCF8574 ที่เป็นอุปกรณ์ขยายพอร์ทขนาด 8 บิตตัวไอซีจะได้รับสัญญาณคล็อก SCL ได้สูงสุดถึง 100 kHz

ในการกำหนดค่าความถี่ของ SCL วงจรภายในของไมโครคอนโทรลเลอร์ตระกูล LPC2000 จะนำค่าของ PCLK มาหารค่าให้ได้ความถี่ที่ต้องการ โดยกำหนดค่าตัวหารได้จากสมาการ

สัญญาณคล็อก = PCLK / (I2SCHL + I2SCHH)

จากตัวอย่างแผงวงจรของ JX – 2148 และ CP – JR ARM7 USB – LPC2148 กำหนดค่าตัวหาร VPBDIV = 2 ได้ PCLK = 30 MHz แล้วต้องกำหนดค่าตัวหารเป็น

100 kHz = 30,000 kHz / (I2SCHl + I2SCHH)

ได้ (I2SCHL + I2SCHH) = 300

กำหนดให้ I2SCHL = I2SSCHH = 150 หรือ 0x96

เขียนเป็นคำสั่งภาษาซีได้ดังนี้

I2C0SCLH = 0x96 ;

I2C0SCLL = 0x96 ;

ถัดมาเป็นการเปิดการทำงานของวงจร I^2C และกำหนดหน้าที่การทำงาน ซึ่งทำได้โดยเขียนคำสั่งที่รีจิสเตอร์ I2CONSET ซึ่งรีจิสเตอร์นี้ขนาด 8 บิต และมีค่าแต่ละบิตดังตารางที่ 2.8

ตารางที่ 2.8 ความหมายแต่ละบิตของรีจิสเตอร์ I2CONSET

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AA	Assert acknowledge flag.	
3	SI	I2C interrupt flag.	0
4	STO	STOP flag.	0
5	STA	START flag.	0
6	I2EN	I2C interface enable.	0
7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

ถ้าต้องการล้างค่าของบิตของรีจิสเตอร์ I2CONSET จะต้องสั่งที่รีจิสเตอร์ I2CONCLR ซึ่งแต่ละบิตมีความหมายดังนี้

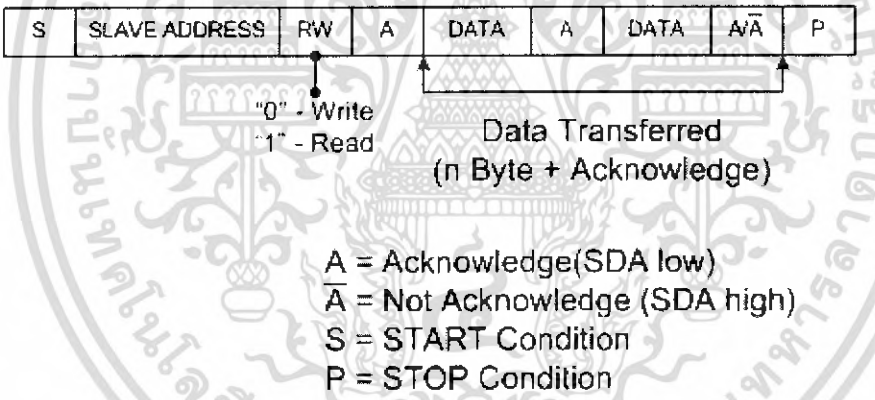
ตารางที่ 2.9 ความหมายแต่ละบิดของรีจิสเตอร์ I2CONCLR

Bit	Symbol	Description	Reset Value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

2	AA	Assert acknowledge Clear bit.	
3	SIC	I2C interrupt Clear bit.	0
4	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5	STAC	START flag Clear bit.	0
6	I2ENC	I2C interface Disable bit.	0
7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

ขั้นตอนการทำงานเมื่อเป็นมาสเตอร์ และส่งข้อมูลไปให้อุปกรณ์

โดยการทำงานของ I^2C บัสเมื่อให้ LPC2148 ทำงานเป็นมาสเตอร์ และส่งข้อมูลทำได้โดยการเซตบิต I2EN ให้เป็น 1 เพื่อเปิดการทำงานของวงจร I^2C และสั่งไม่ให้วงจรส่งสัญญาณสถานะตอบกลับ ซึ่งทำได้โดยค่าเซตบิต AAC ให้เป็น 1 ถัดมา LPC2148 จะต้องส่งสัญญาณสถานะต่างๆ ไปยังอุปกรณ์ I^2C ที่ต่อบนบัส ซึ่งมีขั้นตอนดังรูปที่ 2.5



รูปที่ 2.4 รูปแบบการทำงานในโหมดมาสเตอร์ส่งข้อมูล

เริ่มต้น LPC2148 จะสร้างสัญญาณสถานะเริ่มส่งข้อมูล (START) ด้วยการเซตบิต STA ของรีจิสเตอร์ I2CONSET ให้เป็น 1 ถัดมาส่งแอดเดรสขนาด 7 บิตของอุปกรณ์ที่ต้องการติดต่อ ส่วนบิตที่ 8 เป็นโหมดการติดต่อ ถ้าเป็นบิต 1 คืออ่านข้อมูล ถ้าเป็น 0 คือเขียนข้อมูล ต่อมาต้องรอการตอบกลับจากอุปกรณ์เมื่อตอบกลับแล้วจึงส่งข้อมูลขนาด 8 บิตไปให้อุปกรณ์บนบัสโดยเขียนข้อมูลยังรีจิสเตอร์ I2DAT เมื่อส่งข้อมูลครบ 8 บิตแล้วก็รออุปกรณ์ตอบกลับ โดยสามารถส่งข้อมูลไบท์เดียวหรือหลายไบท์ต่อเนื่องกันได้ เมื่อส่งข้อมูลครบแล้ว LPC2148 จะส่งสัญญาณสถานะหยุดการทำงาน (STOP) ไปยังบัส I^2C ด้วยการสั่งให้บิต STO ของรีจิสเตอร์ I2SONSET เป็น 1

ในการตรวจสอบว่าหลังจากสร้างสัญญาณ START อุปกรณ์บนบัสตอบกลับหรือยัง ให้ตรวจสอบค่าของรีจิสเตอร์ I2STAT ซึ่งมีค่าต่างๆดังแสดงในรูปที่ 2.5

0x08 เมื่อสั่งให้สร้างสัญญาณสถานะ START เสร็จสมบูรณ์แล้วจะอ่านค่าได้เป็น 0x08 ขั้นตอนต่อมาให้ส่งแอดเดรสของสเลฟขนาด 7 บิต และบิตสั่งอ่านหรือเขียน

0x18 เมื่อส่งแอดเดรสของสเลฟ และเลือกการติดต่อเป็นเขียนให้กับสเลฟ อุปกรณ์สเลฟตัวที่มีแอดเดรสตรงจะตอบกลับมาเป็น 0x18

0x20 เมื่อส่งแอดเดรสของสเลฟ และเลือกการติดต่อเป็นเขียน ให้กับสเลฟถ้าไม่มีการตอบกลับจากอุปกรณ์ที่มีแอดเดรสตามที่ส่งจะอ่านค่าได้เป็น 0x20 กรณีนี้อาจส่งแอดเตรสและบิต R/W ซ้ำอีกครั้งถ้ายังไม่ตอบกลับอีกแสดงว่าไม่มีอุปกรณ์ต่อในบัสให้หยุดการทำงานของ I^2C

0x28 เมื่อส่งข้อมูลไปยังสเลฟ และมันได้ข้อมูลถูกต้อง และตอบกลับแล้ว จะอ่านค่ารีจิสเตอร์ได้เป็น 0x28

0x30 เมื่อส่งข้อมูลไปยังสเลฟ ถ้าสเลฟรับค่าไม่ได้หรือรับค่าไม่สมบูรณ์แบบไม่มีการตอบกลับ จะอ่านค่าได้เป็น 0x30 ซึ่งกรณีต้องส่งข้อมูลซ้ำอีกครั้ง

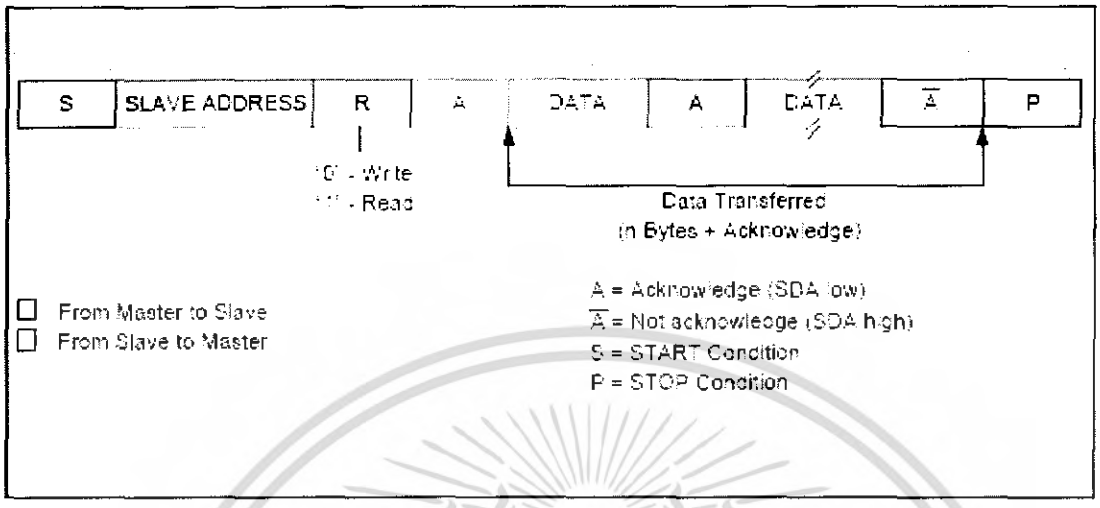
จากหลักการข้างต้นนำมาเขียนเป็นฟังก์ชั่น I2C_write() เพื่อเขียนข้อมูลให้กับ I^2C บัสได้ดังนี้ ในการเรียกใช้ฟังก์ชั่น I2C_write() จะต้องส่งค่าให้กับฟังก์ชั่นสองตัว ตัวแรกคือแอดเดรสของอุปกรณ์ I^2C ที่ต้องการติดต่อ ข้อมูลชุดที่สองเป็นข้อมูลที่ต้องการส่งให้อุปกรณ์ I^2C ถ้ามีอุปกรณ์ I^2C สามารถส่งข้อมูลได้ ฟังก์ชั่นจะคืนค่าเป็น 0 ถ้าไม่สามารถติดต่อกับอุปกรณ์ I^2C ได้ฟังก์ชั่นจะคืนค่าเป็น 1

```
void write_PCF8574(unsigned char Addr,unsigned char Data)
{
  I20CONCLR  = 0x6C;              // Reset all I2C Status
  I20CONSET |= 0x40;              // Enable I2C Interface
  I20CONSET |= 0x20;              // Send Start Condition
      // Wait I2C Status Return
  while((I20STAT)!= 0x08){;;} // Wait Start Condition Complete
  I20DAT    = Addr;               // Send PCF8574A+[Read/Write]
  I20CONCLR = 0x28;               // Clear Start Bit + Interrupt Flag
      // Wait I2C Status Return
  while((I20STAT)!= 0x18){;;} // Wait Slave Address+W, ACK
  I20DAT = Data;                  // Send Output Data to PCF8574A
  I20CONCLR = 0x0C;               // Wait I2C Status Return
  while((I20STAT)!= 0x28){;;}
      // Wait Slave Address+W, ACK
  I20CONSET |= 0x10;              // Send Stop Condition
  I20CONCLR = 0x0C;      // Clear Acknowledge Bit + Interrupt Flag
}
```

ขั้นตอนการทำงานเมื่อเป็นมาสเตอร์ และรับข้อมูลจากบัส I^2C

ในการทำงานของ I^2C บัสเมื่อให้ LPC2148 ทำงานเป็นมาสเตอร์ และรับข้อมูลทำได้ โดยการเซตค่าบิต I2EN ให้เป็น 1 เพื่อเปิดการทำงานของวงจร I^2C และสั่งไม่ให้วงจรส่งสัญญาณ

สถานะตอบกลับ ซึ่งทำได้โดยเซตค่าบิต AAC ให้เป็น 1 ถัดมา LPC2148 จะต้องส่งสัญญาณสถานะต่างๆไปยังอุปกรณ์ I^2C ที่ค่อบนบัส จะมีขั้นตอนดังรูปที่ 2.6



รูปที่ 2.5 รูปแบบของการรับส่งข้อมูลในโหมดมาสเตอร์รับข้อมูล

เริ่มต้นโดย LPC2148 สร้างสถานะเริ่มต้นให้กับบัส ถัดมาจะส่งแอดเดรสของสเลฟขนาด 7 บิต โดยบิตที่ 8 เป็น 1 เพื่อแสดงว่าต้องการอ่านข้อมูลจากอุปกรณ์ เมื่อสเลฟได้รับแอดเดรสของตัวเองจะตอบกลับโดนสร้างสถานะตอบกลับมายังบัส ถัดมาสเลฟจะส่งข้อมูลมาให้เมื่อมาสเตอรีได้รับข้อมูลจะตอบกลับโดยสร้างสถานะตอบกลับให้กับบัส(อาจจะไม่ตอบกลับไปยังอุปกรณ์ก็ได้ โดยที่เป็นสเลฟจะไม่มีการส่งข้อมูลซ้ำ) ในการรับข้อมูลสามารถรับเพียงแค่ไบท์เดียวหรือหลายไบท์ต่อเนื่องกันได้

ค่าสถานะที่อ่านได้จากรีจิสเตอร์ I2CSTAT ของการทำงานเป็นมาสเตอร์รับข้อมูล

0x08 สร้างสถานะ START และส่งไปยังบัส I^2C

0x40 ส่งแอดเดรสของสเลฟ และคำสั่งอ่านข้อมูลไปยังอุปกรณ์ และอุปกรณ์ตอบกลับแล้ว

0x48 ส่งแอดเดรสของสเลฟ และคำสั่งอ่านข้อมูลไปยังอุปกรณ์ แต่อุปกรณ์ไม่ตอบกลับ

0x50 ได้รับไบท์ข้อมูล และสร้างสถานะตอบกลับให้สเลฟแล้ว

0x58 ได้รับไบท์ข้อมูลแต่ไม่ได้ตอบกลับไปยังสเลฟ

จากหลักการข้างต้นนำมาเขียนฟังก์ชั่น I2C_read() เพื่ออ่านข้อมูลจากอุปกรณ์ที่อยู่บน I^2C บัสได้ดังนี้ ในการเรียกฟังก์ชั่น I2C_read() ค่าที่ส่งให้กับฟังก์ชั่นเป็นแอดเดรสของอุปกรณ์ที่ต้องการอ่านข้อมูล ค่าที่ฟังก์ชั่นคืนกลับมาจะเป็นข้อมูลที่อ่านได้จากอุปกรณ์I^2C

```
char I2C_raed (unsigned char Addr)
{
```

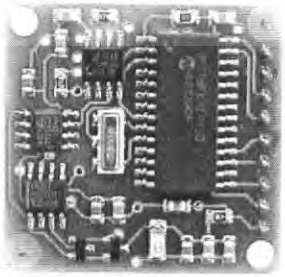
```
unsigned char i2cdata ;
I20CONCLR  = 0x6C ;                    // Reset All I2C Status
I20CONSET  |= 0x40 ;                   // Enable  I2C Interface
I20CONSET  |= 0x20 ;                   // Send Start Condition

while ((I20STAT) |= 0x08) ;            // Wait Start Condition Complete
I20DAT  =  Addr ;                      // Send PCF8574A+ [Read / Write]
I20CONCLR  =  0x28 ;                   // Clear Start Bit + Interrupt Flag
while ((I20STAT) != 0x40) ;            // Wait Slave ACK for Address + R
I20ONCLR  =  0x0C ;                    // Clear ACK Bit + Interrupt Flag

while ((I20STAT)  != 0x58) ;           // Wait Slave ACK for Data
i2cdata  = I20DAT ;
I20CONSET  |= 0x10 ;                   // Send Stop Condition
I20CONCLR  |= 0x0C ;                   // Clear ACK Bit + Interrupt Flag
return  i2cdata ;
}
```

2.2 การใช้ เข็มทิศแบบดิจิตอล CMPS03

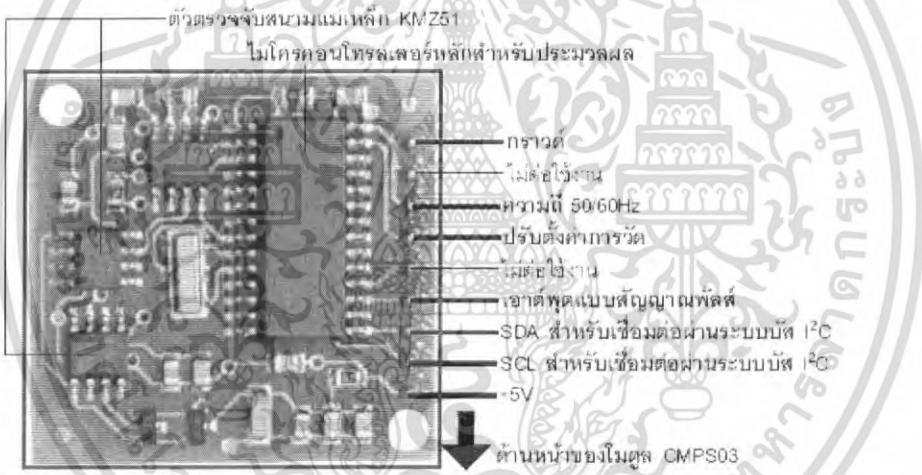
ในการทำโครงงานนี้ได้เลือกใช้ เข็มทิศแบบดิจิตอล CMPS03 เป็นตัวสร้างแผนที่ให้กับหุ่นยนต์และหัวใจสำคัญของ CMPS03 คือการตรวจจับสนามแม่เหล็กเบอร์ KMZ 51 Philips จำนาน 2 ตัวเพื่อให้มีความไวเพียงพอในการตรวจจับสนามแม่เหล็กโลกและไมโครคอนโทรลเลอร์เพื่อรับสัญญาณจากตัวตรวจจับมาประมวลผลเป็นข้อมูลดิจิตอลและสัญญาณ พัลส์สำหรับแสดงผลทิศทางรายละเอียดดูที่ Datasheet



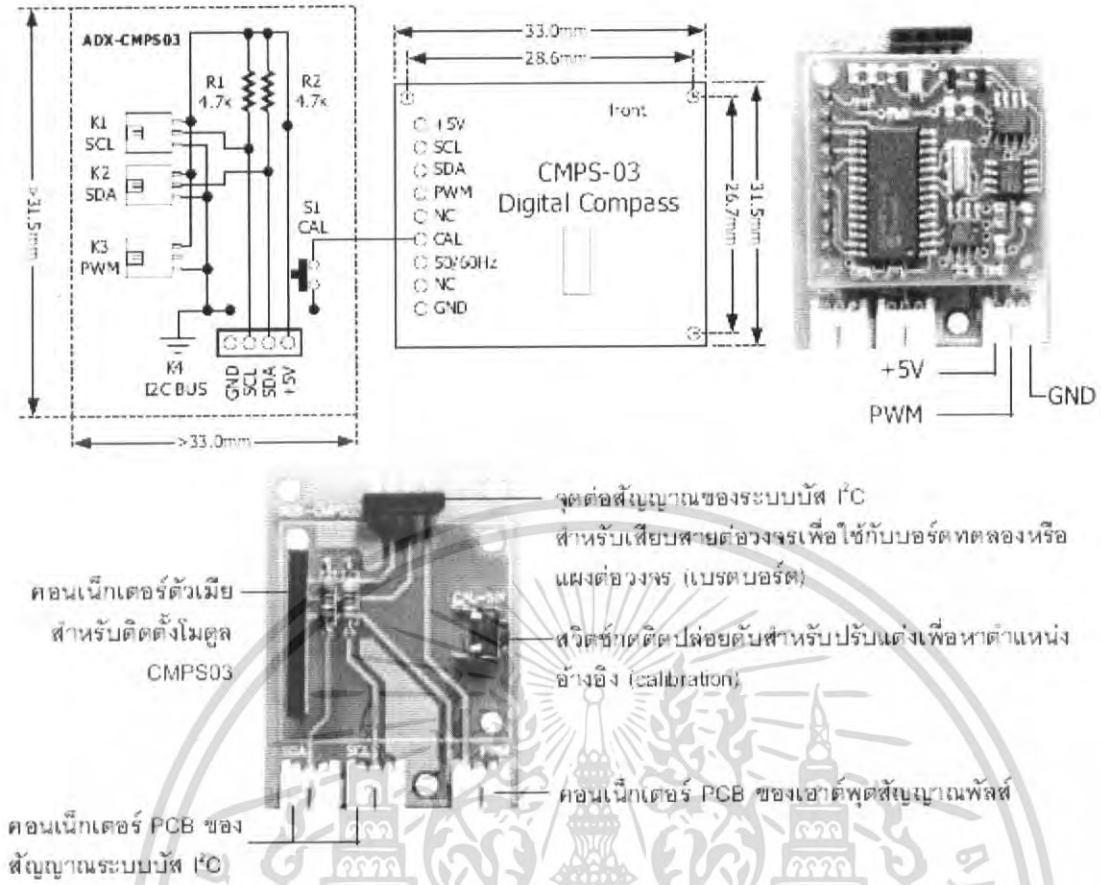
รูปที่ 2.6 ลักษณะภายนอกของ เข็มทิศแบบดิจิตอล CMPS03

คุณสมบัติ

- ใช้ไฟเลี้ยง +5V ต้องการกระแสไฟฟ้า 20 mA
- ใช้ตรวจจับสนามแม่เหล็กเบอร์ KMZ51 ของ Philips จำนวน 2 ตัว เพื่อให้ตรวจสนามแม่เหล็กโลกได้อย่างสมบูรณ์และมีความละเอียดมากเพียงพอ
- ความละเอียดของมุม 0.1 องศา
- ความผิดพลาด 3-4 องศาโดยประมาณ หลังจากการปรับแต่ง
- เอาต์พุตแบบสัญญาณพัลส์ ความกว้าง 1 ถึง 37 มิลลิวินาที โดยมีอัตราเพิ่มครั้งละ 0.1 มิลลิวินาที
- เอาต์พุตข้อมูลดิจิตอลผ่านการติดต่อระบบบัส I^2C รองรับสัญญาณนาฬิกาความถี่สูงถึง 1 MHz โดยให้ข้อมูล 2 รูปแบบ คือ 0-255 และ 0-3599
- ขนาดเล็กเพียง 32*35 มิลลิเมตร
- สื่อสารกับไมโครคอนโทรลเลอร์ยอดนิยมได้ทุกตระกูล อาทิ เบสิกแสตมป์ 2SX/2P, PIC, MCS- 51, PSoC, 68 HC11 ทั้งผ่านระบบบัส I^2C และด้วยการวัดสัญญาณพัลส์



รูปที่ 2.7 ขาที่ต่อใช้งาน เข็มทิศแบบดิจิตอล CMPS03



รูปที่ 2.8 การต่อเข้ากับอุปกรณ์เข็มทิศแบบดิจิตอล CMPS03

2.3 พื้นฐานการประมวลผลภาพ

การประมวลผลภาพดิจิตอล(Image Processing) เป็นกระบวนการที่นำข้อมูลภาพดิจิตอลเข้ามาทำการประมวลผล และให้ข้อมูลออกมาเป็นข้อมูลภาพตัวใหม่ ซึ่งการประมวลผลส่วนใหญ่จะเน้นในด้านการปรับปรุงภาพให้ดีขึ้น (Image Enhancement) การวิเคราะห์ภาพ (Image Analysis) การคืนสภาพเดิมของรูปภาพ (Image Restoration) การแบ่งภาพออกเป็นส่วนๆ (Image Segmentation) การแปลงภาพ (Image Transforms) และการบีบอัดรูปภาพ (Image Compression) ในหัวข้อนี้เรามีวัตถุประสงค์ คือ เพื่อสร้างพื้นฐานความเข้าใจเกี่ยวกับการประมวลผลภาพดิจิตอล ดังนั้นเนื้อหาส่วนใหญ่จึงเป็นเพียงพื้นฐานเบื้องต้น

2.3.1 ประเภทการกระทำการภาพ(Types of Image Operation)

การประมวลผลภาพดิจิตอลจะเป็น กระบวนการที่ทำการ(Operation) ภาพอย่างใดอย่างหนึ่งต่อภาพนำเข้า (Input Image) เพื่อให้ได้ภาพผลลัพธ์ (Output Image) มีลักษณะของภาพเป็นไปตามที่ต้องการ ซึ่งการกระทำการภาพที่ใช้ในการประมวลผลภาพดิจิตอลมีอยู่มากมายหลายแบบ

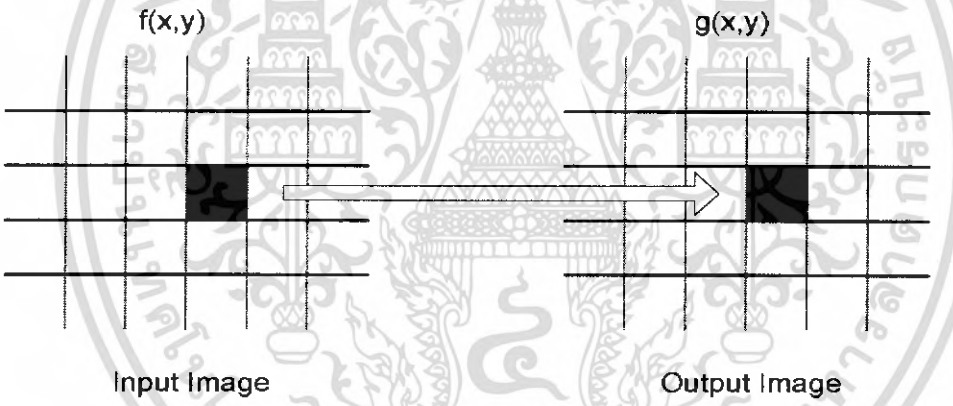
ความเข้าใจเกี่ยวกับคุณลักษณะและการแยกแยะประเภทของการกระทำการภาพ จะช่วยให้เราสามารถคาดคะเนภาพผลลัพธ์ที่จะได้จากการกระทำการแต่ละแบบ หรือการประมาณความซับซ้อนของการกระทำการภาพที่จะใช้

การกระทำการภาพในการประมวลผลภาพดิจิตอลสามารถแบ่งออกได้เป็นประเภทใหญ่ๆ 3 ประเภท คือ

1. การกระทำการจุดต่อจุด(Point Operation) การกระทำการแบบนี้ ค่าความเข้มแสดงในแต่ละพิกเซลของภาพผลลัพธ์ จะขึ้นกับค่าความเข้มแสงของพิกเซลในภาพนำเข้า ณ ตำแหน่งที่สมนัยกันดังรูปที่1 ลักษณะการกระทำการภาพประเภทนี้ได้แก่ การปรับความสว่าง หรือ ความคมชัดของภาพดิจิตอล การ บวก ลบ คูณ และ หาร ภาพดิจิตอล หรือ การกระทำการทางตรรกศาสตร์ต่างๆ เป็นต้น

ถ้า f(x,y) และ g(x,y) เป็นภาพนำเข้าและภาพผลลัพธ์ตามลำดับ ค่าของพิกเซล g(x,y) จะมีค่าดังนี้

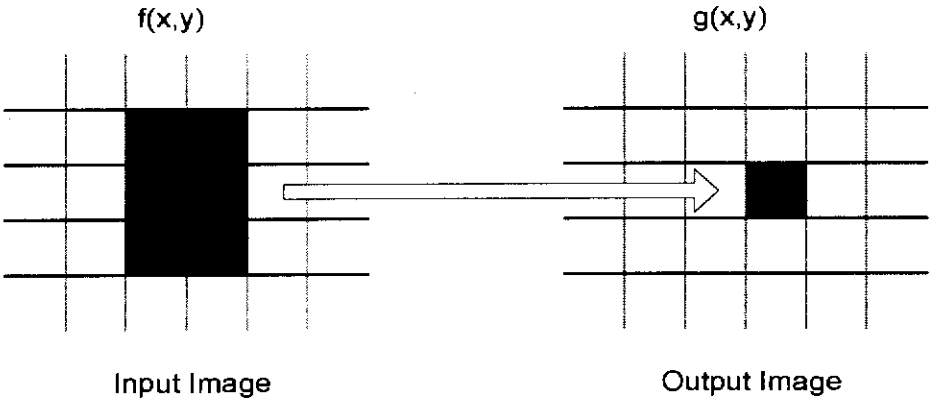
$$g(x_i,y_i) = \tau[f(x_i,y_i)] \text{ ; เมื่อ } \tau \text{ เป็นการกระทำการภาพใดๆ}$$



2. การทำการเฉพาะบริเวณ (Local Operation) สำหรับการกระทำการแบบนี้ค่าความเข้มแสงของพิกเซลแต่ละจุดในภาพผลลัพธ์จะขึ้นกับค่าความเข้มแสงของกลุ่มพิกเซลที่อยู่บริเวณเดียวกัน (Neighborhood Pixel) ในภาพนำเข้า ดังรูปที่ 2 ลักษณะการกระทำการภาพประเภทนี้ได้แก่ การหาขอบ (Edge Detection) การกรองสัญญาณในโดเมนระยะทาง (Spatial Filtering) เป็นต้น

ถ้า f(x,y) และ g(x,y) เป็นภาพนำเข้าและภาพผลลัพธ์ตามลำดับ ค่าของพิกเซล g(x,y) จะมีค่าดังนี้

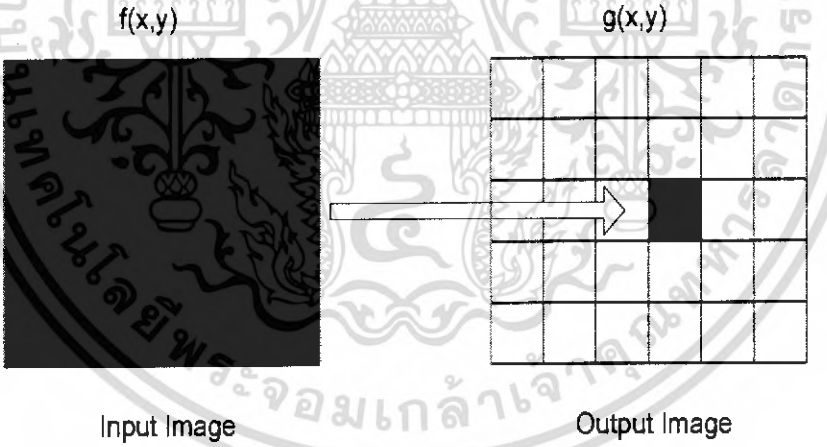
$$g(x_i,y_i) = \tau[neighborhood\ \ of\ \ f(x_i,y_i)]$$



3. การกระทำการทั้งหมด (Global Operation) การกระทำการแบบนี้ ค่าความเข้มแสดงในแต่ละพิกเซลของภาพผลลัพธ์ (Output Image) จะขึ้นกับค่าความเข้มแสงของพิกเซลทุกตัวในภาพนำเข้า ดังรูปที่ 3 ลักษณะการกระทำการภาพประเภทนี้ได้แก่ การเทรสโฮลดิ้ง (Thresholding) การทำฮีสโทรแกรม (Histrogram) เป็นต้น

ถ้า f(x,y) และ g(x,y) เป็นภาพนำเข้าและภาพผลลัพธ์ตามลำดับ ค่าของพิกเซล g(x,y) จะมีค่าดังนี้

$$g(x_i,y_i) = \tau[f(x_i,y_i) \quad for \quad all \ i]$$



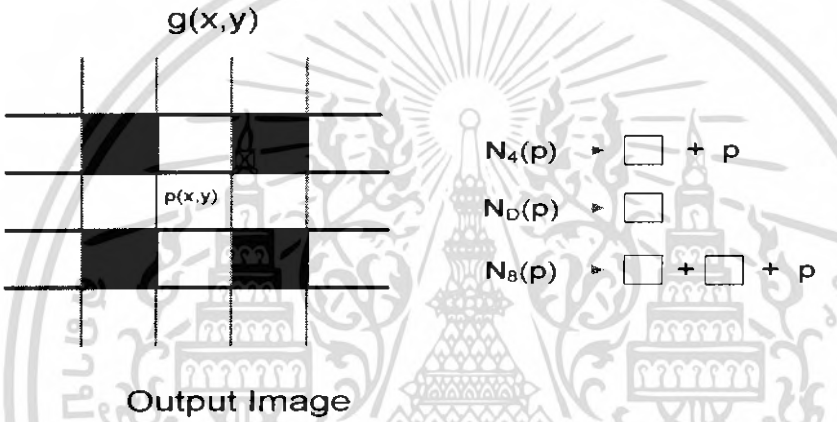
2.3.2 พื้นฐานความสัมพันธ์ระหว่างพิกเซล(Basic Relationships between Pixels)

รูปภาพดิจิตอลจะมีคุณสมบัติทั้งในด้านการวัดระยะทางและรูปร่างเชิงเรขาคณิตที่แตกต่างเล็กน้อยสำหรับฟังก์ชันทางระนาบสองมิติแบบต่อเนื่องที่ประกอบด้วยพิกเซลเรียงต่อกันเป็นระนาบสองมิติ การวัดระยะทางความห่างหรือรูปร่างเชิงเรขาคณิตของวัตถุต่างๆในรูปภาพ จะกำหนดด้วยระยะห่างระหว่างพิกเซลและรูปร่างการเชื่อมต่อของพิกเซล ดังนั้นความเข้าใจพื้นฐานในด้านความสัมพันธ์ระหว่างพิกเซล ดังนั้นความเข้าใจพื้นฐานในด้านความสัมพันธ์ระหว่างพิกเซลซึ่งเป็นส่วนประกอบของรูปภาพดิจิตอล จึงมีความจำเป็นที่ต้องกล่าวถึง

2.3.2.1 พิกเซลใกล้เคียง (Pixels Neighbors)

ในการประมวลผลภาพดิจิตอลนั้น มีการกระทำหลายแบบที่ใช้กลุ่มของพิกเซลใกล้เคียงกันเป็นหลัก นอกจากนี้ การเชื่อม (Connectivity) ระหว่างพิกเซลจะทำได้เฉพาะในส่วนของพิกเซลที่อยู่ใกล้เคียงกันเท่านั้น ดังนั้นการเข้าใจคำนิยามของพิกเซลใกล้เคียงสามารถกำหนดได้เป็นแบบใหญ่ๆ 2 แบบ คือ พิกเซลใกล้เคียงแบบ 4 ซึ่งพิกเซลประชิดจะเรียงตัวกันเป็นรูปกากบาท และพิกเซลใกล้เคียงแบบ 8 ซึ่งพิกเซลประชิดจะเรียงกันอยู่เป็นรูปสี่เหลี่ยมจัตุรัส ดังรูปที่ 3

ถ้ากำหนดให้พิกเซล p มีตำแหน่งอยู่ที่ (x,y) $N_4(p)$ และ $N_8(p)$ จะเป็นการประชิดพิกเซลแบบ 4 และ 8 ตามลำดับ และ $N_D(p)$ เป็นพิกเซลที่อยู่ในตำแหน่ง (x+1,y+1),(x+1,y-1),(x-1,y+1) และ (x-1,y-1) ในขณะที่สมาชิกของ $N_4(p)$ จะประกอบไปด้วย p และ พิกเซลที่ตำแหน่ง (x+1,y) , (x-1,y),(x,y+1) และ (x,y-1) และสมาชิกของ $N_8(p)$ จะประกอบด้วย p, $N_4(p)$ และ $N_D(p)$



2.3.3 สัญญาณรบกวนในรูปภาพดิจิตอล (Noise in Digital Image)

รูปภาพดิจิตอลที่ได้ในงานจริงๆมักจะมีคุณภาพลดลงเนื่องจากมีสัญญาณรบกวน (Noise) ปนอยู่ด้วยเสมอ การจำลองลักษณะของสัญญาณรบกวนสำหรับงานทางด้านการประมวลผลภาพดิจิตอลโดยมากจะอยู่ในรูปของฟังก์ชันการกระจาย (Distribution Functions) ทางสถิติต่างๆซึ่งในทางปฏิบัติสัญญาณรบกวนส่วนใหญ่จะเกิดขึ้นเนื่องจากปรากฏการณ์ธรรมชาติต่างๆในขั้นตอนการเปลี่ยนจากสัญญาณแสงมาเป็นสัญญาณไฟฟ้า ดังนั้นสมมติฐานของสัญญาณรบกวนที่นิยมใช้กันจะเป็นสัญญาณรบกวนแบบเกาส์เซียน (Gaussian) แบบเสมอ (Uniform) และแบบจุด (Impulse)

สัญญาณรบกวนแบบเกาส์เซียน (Gaussian Noise) จะเป็นสัญญาณรบกวนที่มีการกระจายในลักษณะปกติ (Normal) ซึ่งลักษณะการกระจายของสัญญาณรบกวนจะเป็นฟังก์ชันของค่าระดับเทา ดังสมการ

$$p(g) = \frac{1}{\sigma\sqrt{2\pi}} \; e^{\frac{-(g-m)^2}{2\sigma^2}}$$

เมื่อ g เป็นค่าระดับเทา

m เป็นค่าเฉลี่ย

σ^2 เป็นค่าความแปรปรวน

สัญญาณรบกวนแบบเสมอ (Uniform Noise) จะเป็นสัญญาณรบกวนที่มีการกระจายในลักษณะสม่ำเสมอ ซึ่งจะเป็นไปตามสมการ

$$p(g)\begin{cases}\dfrac{1}{b-a} & a \le g \le b \\ 0 & elsewhere\end{cases}$$

และสัญญาณรบกวนแบบจุด (Impulse Noise) จะเป็นสัญญาณรบกวนที่พิกเซลเป็นจุดๆ ไป โดยจะเปลี่ยนค่าความเข้มแสงของพิกเซล ณ ตำแหน่งหนึ่งๆ ให้มีค่าแตกต่างไปจากพิกเซลข้างเคียง ซึ่งค่าความเข้มแสงที่จำลองกันโดยมาก จะเป็นในลักษณะความเข้มแสงสีขาวและดำ (Salt and Pepper) นั่นคือรูปภาพดิจิตอลจะถูกทำลายโดยพิกเซลบางตำแหน่งจะถูกเปลี่ยนเป็นจุดดำ หรือ จุดขาว โดยพิกเซลรบกวน (Noise Pixels) สัญญาณรบกวนแบบนี้จะมีฟังก์ชันการกระจายตามสมการ

$$p(g)\begin{cases}A & g = a \\ B & g = b\end{cases}$$

เมื่อ a และ b เป็นค่าระดับเทาเดิม

A และ B เป็นค่าระดับเทาใหม่

2.3.4 การทำคอนโวลูชัน (Convolution)

การทำคอนโวลูชันในเชิงของการประมวลผลภาพดิจิตอล จะเป็นการกระทำการระหว่างภาพนำเข้า f(x,y) ที่มีขนาด N*N เมื่อ N เป็นเลขจำนวนเต็มใดๆกับ มาส์ค (Mask) m(x,y) ซึ่งจะเป็นภาพที่มีขนาด M*M เมื่อ M เป็นเลขจำนวนเต็มใดๆ (ปกติจะเป็นเลขคี่) และมีขนาดน้อยกว่า N มากๆ เช่น 3*3 ,5*5,7*7 ,9*9 หรือ 11*11 เป็นต้น ผลลัพธ์ความเข้มแสงใหม่ที่ได้จากการทำคอนโวลูชันจะถูกเก็บไว้ในภาพผลลัพธ์ g(x,y) ซึ่งขั้นตอนการทำคอนโวลูชันจะเป็นไปตามสมการ

$$g(x,y) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\sum_{i=0}^{M-1}\sum_{j=0}^{M-1} f(i,j)m(x-i,y-u)$$

จากสมการจะเห็นว่าการทำคอนโวลูชันจะมีวิธีการเป็นขั้นตอนดังนี้

1. กลับ (Flip) มาส์คเทียบกับตำแหน่งจุดตรงกลางของมาส์ค เมื่อใช้มาส์คมีขนาดเป็นเลข

จำนวนเต็มคี่ ซึ่งโดยปกติมาส์คที่ใช้ในการประมวลผลภาพดิจิตอลจะมีลักษณะสมนัยทั้งในแนว x และ y ทำให้มาส์คที่ได้จะมีลักษณะเหมือนเดิม ดังนั้นขั้นตอนนี้จึงมักจะถูกละไว้และการกำหนดมาส์คจะกำหนดในลักษณะที่เป็นมาส์คหลังจากการกลับแล้ว

2. เลื่อนมาส์คให้ไปทาบภาพนำเข้าโดยให้จุดตรงกลางของมาส์คตรงกับพิกเซล f(x,y)

3. ทำการคูณค่าความเข้มแสงระหว่างพิกเซลของมาส์คและพิกเซลของภาพนำเข้าที่ ณ ตำแหน่งเดียวกันแบบจุดต่อจุด

4. บวกผลคูณที่ได้ในขั้นตอนที่ 3 ทั้งหมด และนำผลลัพธ์ที่ได้ไปใส่เป็นค่าความเข้มสว่างของผลลัพธ์ ณ ตำแหน่งเดียวกับพิกเซลของภาพนำเข้า นั่นคือ g(x,y)

5. เลื่อนตำแหน่งไปที่พิกเซลถัดไปของภาพนำเข้าและทำซ้ำตั้งแต่ ขั้นตอนที่ 2 ถึง 4 จนกระทั่งครบพิกเซลทุกตัวในภาพนำเข้า

ตัวอย่างเช่น ถ้ามีมาส์คขนาด 3*3 ที่มีค่า ณ ตำแหน่งต่างๆหลังจากทำการกลับแล้วดังนี้

$$\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}$$

เมื่อนำมาส์คนี้ไปทาบพิกเซล p ณ. ตำแหน่ง (x,y) ของภาพนำเข้า f(x,y) ผลของการทำคอโวลูชัน จะได้ค่าความเข้มแสงของพิกเซล q ณ. ตำแหน่งเดียวกันของภาพผลลัพธ์ g(x,y) มีค่าดังนี้

$$q = g(x,y) = Af(x-1,y-1) + Bf(x-1,y) + Cf(x-1,y+1) + Df(x,y-1) + Ef(x,y) + Ff(x,y+1),$$
$$Gf(x+1,y-1) + Hf(x+1,y) + If(x+1,y+1)$$

จากนั้นมาส์คจะเลื่อนไปที่ตำแหน่งถัดไปทางขวามือ นั่นคือ f(x,y+1) โดยจะทำไปเรื่อยๆ จนครบทุกพิกเซลในภาพนำเข้า

จากขั้นตอนการคอนโวลูชันจะเห็นได้ว่า ถ้าเราต้องการให้มาส์คทุกตัวทาบลงบนภาพนำเข้าพอดี เราจะต้องข้ามไม่ทำการคอนโวลูชันพิกเซลแถวแรกๆหรือแถวท้ายๆและหลักแรกๆ หรือหลักท้ายๆซึ่งจำนวนที่หายไปจะขึ้นกับขนาดของมาส์ค ตัวอย่างเช่น มาส์คขนาด 3*3 การทำคอนโวลูชันจะไม่ทำกับพิกเซลแถวแรกและแถวสุดท้ายและในหลักแรกและหลักสุดท้าย การคอนโวลูชันเป็นพื้นฐานของการกระทำทางการประมวลผลดิจิตอลหลายประเภทในโดเมนเวลา โดยเฉพาะการกรองสัญญาณภาพ (Image Filtering) ซึ่งจะกล่าวถึงในหัวข้อถัดไป

2.3.5 การกรองสัญญาณภาพในโดเมนเวลา (Image Filtering in Time Domain)

การกรองสัญญาณภาพ (Image Filtering) ในโดเมนเวลาเป็นเทคนิคพื้นฐานที่ใช้งานกันอย่างแพร่หลายสำหรับงานด้านการประมวลผลสัญญาณภาพ การกรองสัญญาณภาพเป็นเทคนิคแบบหนึ่งของการปรับปรุงภาพ (Image Enhancement) ที่มีจุดมุ่งหมายในการกำจัดสัญญาณความถี่ต่ำ ความถี่สูงหรือความถี่ของสัญญาณในช่วงหนึ่งๆการกรองสัญญาณภาพทำได้ทั้งในโดเมนเวลาหรือระยะทางและในโดเมนความถี่ แต่การกรองสัญญาณภาพทำได้ทั้งในโดเมนเวลาจะทำได้ง่ายและรวดเร็วกว่าการกรองสัญญาณในโดเมนความถี่ หลักการของการกรองสัญญาณภาพในโดเมนเวลาเป็นการกระทำแบบเฉพาะบริเวณแบบหนึ่งที่มีพื้นฐานการทำงานเป็นกระบวนการคอนโวลูชันภาพต้นฉบับกับมาส์คแบบต่างๆโดยที่ลักษณะของมาส์คที่ใช้จะเป็นตัวกำหนดผลลัพธ์ที่จะได้ ซึ่งการกรองสัญญาณภาพในโดเมนเวลาจะทำได้ง่ายและรวดเร็วกว่าการกรองสัญญาณโดเมนเวลาเป็นการกระทำการแบบเฉพาะบริเวณแบบหนึ่งที่มีพื้นฐานการทำงานเป็นกระบวนการคอนโวลูชันภาพต้นฉบับกับมาส์คแบบต่างๆโดยที่ลักษณะของมาส์คที่ใช้จะเป็นตัวกำหนดผลลัพธ์ที่จะได้ ซึ่งการกรองสัญญาณภาพในโดเมนเวลาสามารถแบ่งออกได้เป็นประเภทใหญ่ 3 ประเภท คือ การทำภาพให้เรียบ (Image Smoothing) การทำภาพให้คมชัด (Image Sharpening) และตัวหาขอบ (Edge Detector)

การทำภาพให้เรียบมีวัตถุประสงค์ในการกำจัดสัญญาณรบกวนซึ่งเป็นสัญญาณประเภทความถี่สูง โดยพื้นฐานอาศัยหลักการทำการเฉลี่ยค่าความเข้มแสงเฉพาะบริเวณหรืออีกนัยหนึ่งคือเป็นการกรองสัญญาณความถี่ต่ำ (Low-pass Filtering) ผลกระทบของการทำภาพให้เรียบก็คือภาพต้นฉบับจะพร่ามัว (Blurring Effect) มีความคมชัดน้อยลง เนื่องจากขอบของวัตถุในรูปภาพจะเป็นบริเวณที่มีการเปลี่ยนแปลงระดับค่าความเข้มแสงและจัดว่าเป็นสัญญาณความถี่สูงจะถูกกรองออกไป ดังนั้นเทคนิคการทำภาพให้เรียบส่วนใหญ่จะเน้นที่การกำจัดสัญญาณรบกวนแต่จะไม่ทำลายของวัตถุในภาพมาส์คที่ใช้ในการทำภาพให้เรียบจะมีลักษณะที่พิกเซลทุกตำแหน่งจะมีค่าเป็นบวกหมด ตัวอย่างเช่น

$$\text{มาส์ค ค่าเฉลี่ย} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$$\text{มาส์ค แบบเกาส์เซียน} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \text{ หรือ } \frac{1}{32}\begin{bmatrix}1 & 4 & 1\\ 4 & 12 & 4\\ 1 & 4 & 1\end{bmatrix} \text{ เป็นต้น}$$

โดยทั่วไปมาส์คแบบเกาส์เซียนจะเป็นที่นิยมใช้มากกว่าแบบค่าเฉลี่ย เนื่องจากว่าจะมีผลกระทบต่อการพร่ามัว (Blur) ของภาพต้นฉบับน้อยกว่า และมีได้หลายแบบ โดยการกำหนดที่ค่า

เบี่ยงเบนมาตรฐานของค่าพิกเซลทั้งหมดในมาส์ค ซึ่งตัวกระทำเรียบแบบเกาส์เซียนสองมิติ G(x,y) จะมีค่าตามสมการ

$$G(x,y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

นอกจากนี้มาส์คแบบเกาส์เซียนจะเหมาะสำหรับการกำจัดสัญญาณรบกวนที่มีการกระจายแบบเกาส์เซียน ซึ่งเชื่อว่าเป็นสัญญาณที่เกิดกับปรากฏการณ์ตามธรรมชาติต่างๆ

การทำภาพเรียบหรือการกำจัดสัญญาณรบกวนยังสามารถใช้มาส์คที่มีความซับซ้อนมากขึ้นหรือมาส์คแบบไม่เชิงเส้นอีกหลายประเภท เช่น ค่ามัธยฐาน (Median) ค่าโหมด (Mode) หรือการหาค่าเฉลี่ยแบบไม่เชิงเส้น เป็นต้น

การทำภาพให้ชัด และ ตัวหาขอบจะเป็นการกรองสัญญาณในโดเมนเวลา ที่ใช้มาส์คที่เป็นประเภทการหาค่าอนุพันธ์ (Derivatives) ของพิกเซลเฉพาะบริเวณ โดยที่การทำให้ภาพชัดขึ้นจะเป็นการกรองสัญญาณในโดเมนเวลา ที่ทำให้บริเวณที่มีการเปลี่ยนแปลงค่าความเข้มแสงในรูปภาพเด่นชัดขึ้น นั่นคือทำงานในลักษณะเป็นการกรองสัญญาณความถี่สูง (High-pass Filtering) ส่วนตัวหาขอบจะเป็นตัวกระทำที่ต้องการหาตำแหน่งขอบของวัตถุ ที่อยู่ในภาพดิจิตอล มาส์คของการทำภาพให้ชัดและตัวหาขอบ จะมีลักษณะคล้ายกันคือค่าของพิกเซลในมาส์คจะมีทั้งบวกและลบ แต่จะมีลักษณะการวางตัวของค่าที่เป็นบวกและลบต่างกันกล่าวคือ มาส์คการทำภาพให้ชัดจะมีค่าเป็นบวกบริเวณใกล้ๆกับจุดตรงกลางมาส์ค และจะมีค่าเป็นลบอยู่รอบๆดังตัวอย่างต่อไปนี้

$$\text{มาส์ค ค่าเฉลี่ย} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \text{ หรือ } \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\text{มาส์ค แบบเกาส์เซียน} = \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix} \text{ หรือ } \begin{bmatrix} 1 & -2 & 1 \\ -1 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \text{ เป็นต้น}$$

มาส์คของตัวหาขอบจะมีวัตถุประสงค์ในการหาว่ามีการเปลี่ยนค่าความสว่างแบบฉับพลันในบริเวณใด ซึ่งถ้าพบก็หมายความว่าบริเวณนั้นๆอาจจะเป็นขอบของวัตถุ

ดังนั้นตัวอย่างของมาส์คจะเป็นดังต่อไปนี้

$$\text{โซเบล(Sobel)} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \text{ หรือ } \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$\text{พีวิธ(Prewitt)} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{หรือ} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

2.4 หลักการทำงานของดีซีมอเตอร์

ดีซีมอเตอร์เป็นทรานสดิวเซอร์แรงบิดซึ่งมีการออกแบบให้มีคุณลักษณะพิเศษคือแรงบิดของเพลาของดีซีมอเตอร์จะเป็นสัดส่วนโดยตรงกับกระแสอาร์มาเจอร์ แรงบิดของเพลาของดีซีมอเตอร์จะได้จากผลระหว่างสนามแม่เหล็กและขดลวดตัวนำ หลักการนี้แสดงได้ในรูปที่ 2.13 ในที่นี้กระแสที่ไหลในขดลวดตัวนำจะสร้างฟิลด์ที่ประกอบด้วยเส้นแรงแม่เหล็ก ϕ และขดลวดตัวนำเหล่านั้นอยู่ห่างจากศูนย์กลางการหมุนเท่ากับ r ความสัมพันธ์ระหว่างแรงบิดของเพลาและกระแสเท่ากับ

$$T = K \phi I \qquad (2\text{-}1)$$

เมื่อ T คือแรงบิดของเพลามีหน่วยเป็นนิวตัน-เมตร

ϕ คือเส้นแรงแม่เหล็กมีหน่วยเป็นเวเบอร์

I คือกระแสเป็นแอมแปร์

และ K คือตัวคงที่ ดังนั้นแรงบิดของเพลาจะเป็นสัดส่วนโดยตรงกับผลคูณของเส้นแรงแม่เหล็กและกระแสเมื่อขดลวดตัวนำเคลื่อนที่ในสนามแม่เหล็กก็จะทำให้เกิดโวลท์เตจตกคร่อมตัวมันเอง โวลท์เต็จนี้จะเป็นสัดส่วนกับความเร็วของเพลาของมอเตอร์และต้านการไหลของกระแส ความสัมพันธ์ระหว่างโวลท์เต็จย้อนกลับนี้และความเร็วของเพลามอเตอร์

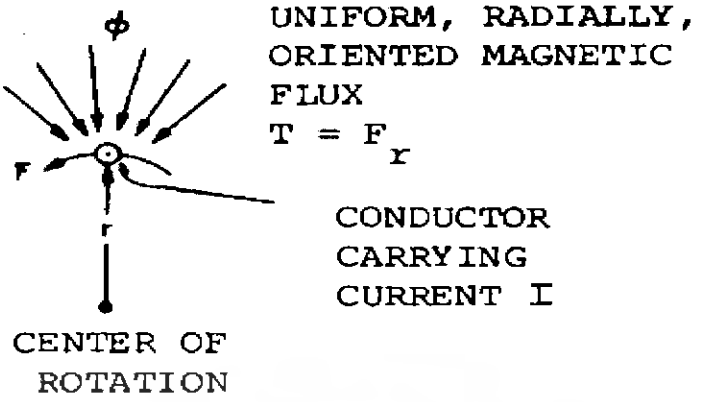
$$E = K \phi \omega \qquad (2\text{-}2)$$

เมื่อ E คือโวลท์เตจย้อนกลับ emf มีหน่วยเป็นโวลท์

ϕ คือเส้นแรงแม่เหล็กมีหน่วยเป็นเวเบอร์

ω คือความเร็วของมอเตอร์มีหน่วยเป็นเรเดียน/วินาที

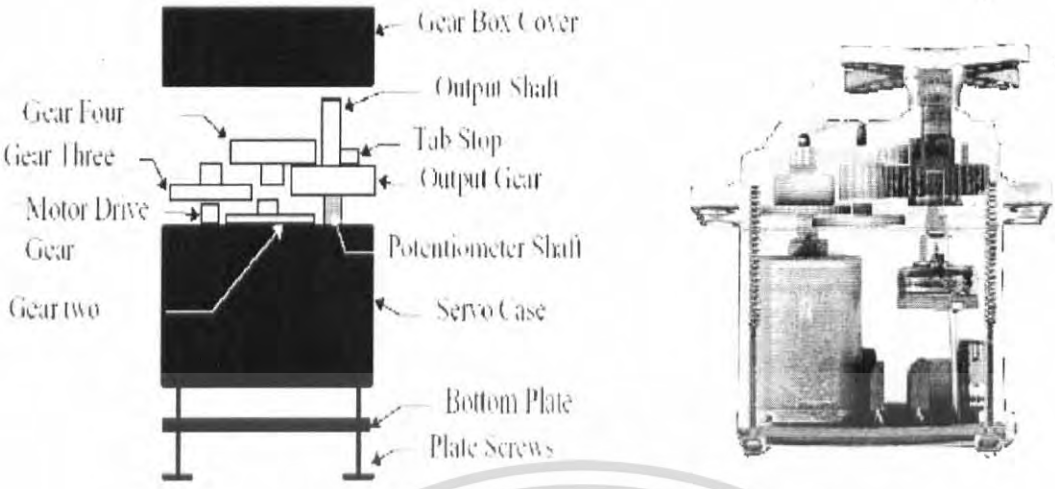
สมการ (2-1) และ (2-2) เป็นสมการที่แสดงถึงหลักการทำงานพื้นฐานของดีซีมอเตอร์



รูปที่ 2.9 การเกิดแรงบิดในตัวดีซีมอเตอร์

2.5 ทฤษฎี Servo motor

servo motor คือ มอเตอร์ไฟฟ้ากระแสตรง (DC motor) ที่ถูกประกอบรวมกับ ชุดเกียร์ และส่วนควบคุมต่างๆไว้ในโมดูลเดียวกัน โดยมอเตอร์ชนิดนี้จะมีสายต่อที่ใช้งานเพียง 3 เส้นเท่านั้น คือ VCC,GND และสายสัญญาณควบคุม(Control Line) ซึ่งสามารถควบคุมให้มอเตอร์หมุนซ้าย หรือ ขวา ได้จากสายสัญญาณเพียงเส้นเดียวโดยสัญญาณที่ใช้ควบคุมจะเป็นสัญญาณ พัลส์วิดมอด (PWM) แบบ TTL Level ระดับแรงดันที่จ่ายให้มอเตอร์นี้จะอยู่ในช่วงประมาณ 4 ถึง 6 โวลท์ ขึ้นอยู่กับคุณสมบัติของมอเตอร์แต่ละตัว ข้อดีของมอเตอร์ชนิดนี้ก็คือ จะมีขนาดเล็กน้ำหนักเบาให้แรงบิดสูง กินพลังงานน้อย และ สามารถควบคุมด้วยแรงดันลอจิกที่เป็น TTL ได้โดยตรงไม่จำเป็นต้องต่อวงจรขับ (Driver) อื่นๆ เพราะมอเตอร์ชนิดนี้จะมีวงจรควบคุมบรรจุไว้ภายในอยู่แล้ว ซึ่งมอเตอร์ชนิดนี้สามารถควบคุมให้หมุนไปในตำแหน่ง หรือ ทิศทางองศาที่ต้องการได้ โดยอาศัยสัญญาณความกว้างพัลส์ที่ป้อนให้มอเตอร์ แต่เซอร์โวมอเตอร์นี้จะหมุนได้แค่เพียงประมาณ 180 องศา หรือครึ่งรอบเท่านั้น หรือบางรุ่นอาจหมุนได้ถึง 210 องศา แต่จะไม่สามารถหมุนเป็นวงรอบได้เนื่องจากโครงสร้างภายในจะประกอบด้วย ตัวต้านทานชนิดปรับค่าได้ (VR) ที่ทำหน้าที่ตรวจสอบตำแหน่งการหมุนของมอเตอร์ และ ตัวต้านทานนี้จะถูกยึดติดกับแกนหมุนของมอเตอร์ ซึ่งจากการที่ตัวต้านทานปรับค่านี้ไม่สามารถหมุนเป็นวงรอบได้ ดังนั้น เซอร์โวมอเตอร์จึงถูกออกแบบให้หมุนได้ประมาณ 180 องศา หรือ ครึ่งรอบเท่านั้น เพื่อป้องกันความเสียหายที่เกิดกับตัวต้านทานปรับค่าได้ แต่ถ้าหากเราต้องการให้มอเตอร์หมุนเป็นวงรอบ (360^0) นั้นก็สามารถทำได้ โดยจะต้องทำการปรับแต่ง (Modify) ดัดแปลงชิ้นส่วนบางอย่างของมอเตอร์ ซึ่งวิธีการต่างๆจะได้กล่าวไว้ภายหลัง

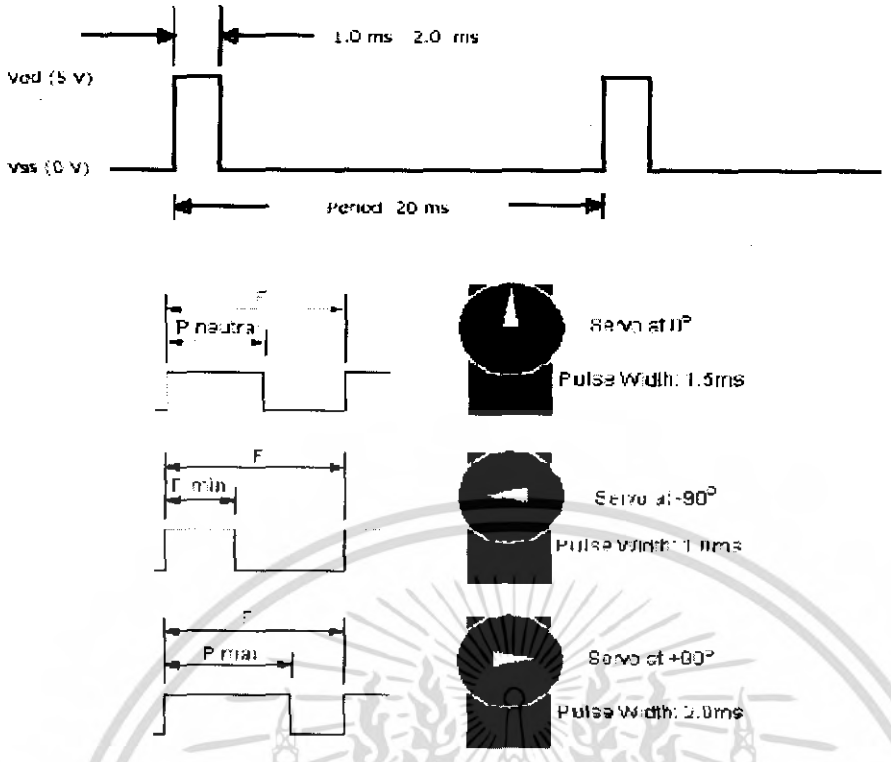


รูปที่2.10 ส่วนประกอบของ Servo Motor(1)

รูปที่2.11 ส่วนประกอบของ Servo Motor(2)

หลักการทำงานของ Servo Motor

การควบคุมการทำงานของเซอร์โวมอเตอร์ทำได้โดยการป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้จะขึ้นอยู่กับขนาดของความกว้างของพัลส์นั้นๆโดยทั่วไปแล้วความกว้างของสัญญาณพัลส์จะมีจุดให้อ้างอิง 3 จุด ดังรูป คือ



รูปที่2.12 ลักษณะ Pulse ที่จ่ายให้ Servo motor

จากรูปที่2.12 จะแสดงให้เห็นลักษณะของ Pulse ที่จ่ายให้ Servo motor จะทำให้ได้ผลดังนี้

- สัญญาณความกว้างพัลส์ขนาด 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือ จุดกึ่งกลางของมอเตอร์

- สัญญาณความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม -90 องศา หรือ ในทิศทางทวนเข็มนาฬิกา

- สัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม +90 องศา หรือ ในทิศทางทวนตามนาฬิกา

***หมายเหตุ** ค่าความกว้างพัลส์ และ ระยะองศาการหมุนของมอเตอร์ที่อธิบายด้านบน นั้นเป็นเพียงค่าประมาณเท่านั้นทั้งนี้ระยะการหมุนและขนาดของพัลส์ที่ควบคุมการทำงานของมอเตอร์ในแต่ละยี่ห้ออาจจะไม่เท่ากัน ดังนั้นในการใช้งานจึงควรศึกษารายละเอียดของมอเตอร์ในแต่ละรุ่นที่นำมาใช้ ซึ่งโดยปกติแล้วรายละเอียดต่างๆมักจะมีติดมากับตัวมอเตอร์นั้นๆอยู่แล้ว

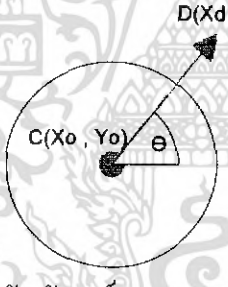
ส่วนการที่จะควบคุมให้มอเตอร์หมุนเป็นมุมอื่นๆ นั้นก็สามารถทำได้โดยการป้อนสัญญาณพัลส์เป็นระดับความกว้างต่างๆโดยอ้างอิงจากจุด ทั้ง 3 จุดที่กล่าวมานี้ ตัวอย่างเช่น ถ้าต้องการให้มอเตอร์หมุนไปที่มุม -45 องศา เราก็จะต้องป้อนสัญญาณพัลส์ที่มีความกว้าง 1.25 ms

เป็นต้น และ สัญญาณพัลส์นี้จะต้องจ่ายให้มอเตอร์ทุกๆ 20 ms (Period) เพื่อรักษาสภาพตำแหน่งของมอเตอร์ไว้

โดยหลักการก็คือ จะอาศัยการเปรียบเทียบช่วงเวลาของความกว้างพัลส์ที่จ่ายให้กับมอเตอร์ทางขาสัญญาณควบคุมกับค่าเวลาของวงจร RC ภายในบอร์ดควบคุมในตัวของมอเตอร์ ซึ่งค่าเวลาของวงจร RC นี้จะมีการเปลี่ยนแปลงตามการหมุนของมอเตอร์ เนื่องจากตัวต้านทานปรับค่าจะถูกยึดติดอยู่กับแกนหมุนของมอเตอร์ ซึ่งการหมุนของมอเตอร์จะทำให้ค่าความต้านทานของตัวต้านทานปรับค่าได้ (VR) เปลี่ยนแปลงไป เป็นผลทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงตามไปด้วย โดยในขณะที่เราป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ทางขาสัญญาณควบคุมสัญญาณนี้จะถูกนำไปเปรียบเทียบกับค่าเวลาของวงจร RC หากค่าทั้ง 2 ไม่เท่ากันมอเตอร์ก็จะหมุนทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงจนกระทั่งค่าเวลาความกว้างพัลส์ของวงจร RC เปลี่ยนแปลงจนเท่ากับสัญญาณพัลส์ทางขาควบคุม (Control line) มอเตอร์จึงจะหยุดหมุน

2.6 การเคลื่อนที่ของ Object

Object ที่นำมาแทนเป็นตัว Robot เพื่อจำลองการเคลื่อนที่ของ Robot ในโปรแกรมจะแทน Robot ด้วย Object วงกลมและเส้นตรงเพื่อใช้บอกทิศทางและแทนด้านหน้าของ Robot ดังรูป



รูปที่ 2.13 สัญลักษณ์ของ object ที่ใช้แทน Robot

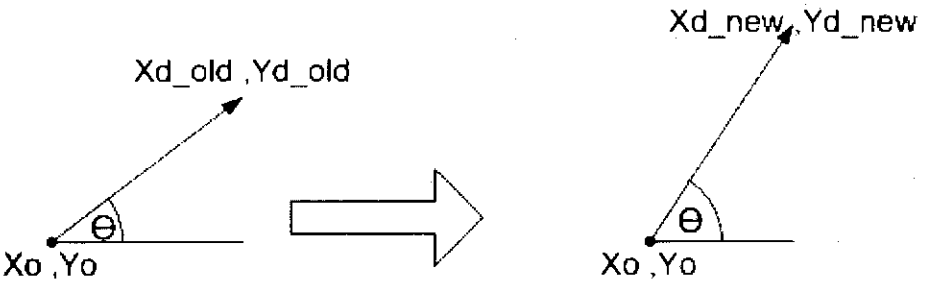
จากรูป วงกลมจะใช้แทนตัว Robot และบ่งบอกด้านหน้าของ Robot ด้วยเส้นตรง CD ในการเคลื่อนที่จะเคลื่อนที่ตามระยะทางที่กำหนดในทิศทางของเส้นตรง CD ในการควบคุมการเคลื่อนที่จะต้องใช้ parameter 2 ค่าคือ ระยะทางในการเคลื่อนที่(R) และมุมของเส้นตรง CD เทียบกับแกน x (θ) ที่ใช้ในการหาทิศทางในการเคลื่อนที่ ในการเคลื่อนที่นั้นจะพิจารณาเป็น 2 ส่วนคือ ส่วนของเส้นตรงและ ส่วนของวงกลม

การควบคุมการเคลื่อนที่ของเส้นตรงมี 2 ลักษณะคือ การเลื่อนเส้นตรงไปในทิศทางของลูกศร และการหมุนปลายลูกศรเพื่อเปลี่ยนทิศทาง

- การหมุนปลายลูกศรทำได้โดยใช้สมการ

$$Xd_{ใหม่} = L.cos(\theta_{ใหม่}) + Xo$$

$$Yd_{ใหม่} = L.sin(\theta_{ใหม่}) + Yo$$



รูปที่ 2.14 การหมุนทิศทางของเส้นตรง

โดยที่ L คือ ความยาวของเส้นตรง

Θ คือ มุมระหว่างเส้นตรงโดยวัดจากแกน x

Xdใหม่ คือ ค่าพิกัด x ใหม่ ที่ได้เมื่อมีการหมุนลูกศร(หมุนตัว Robot)

Ydใหม่ คือ ค่าพิกัด y ใหม่ ที่ได้เมื่อมีการหมุนลูกศร(หมุนตัว Robot)

Xo คือ ค่าพิกัด x ที่จุดกำเนิดของเส้นตรง

Yo คือ ค่าพิกัด y ที่จุดกำเนิดของเส้นตรง

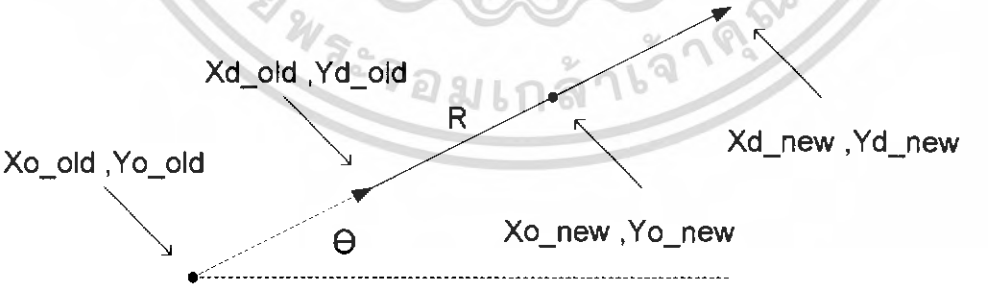
- การเลื่อนเส้นตรงไปในทิศทางของลูกศรสามารถทำได้โดยใช้สมการ

$$Xd_{ใหม่} = R.cos(\Theta) + Xd_{เก่า}$$

$$Yd_{ใหม่} = R.sin(\Theta) + Yd_{เก่า}$$

$$Xo_{ใหม่} = R.cos(\Theta) + Xo_{เก่า}$$

$$Yo_{ใหม่} = R.sin(\Theta) + Yo_{เก่า}$$

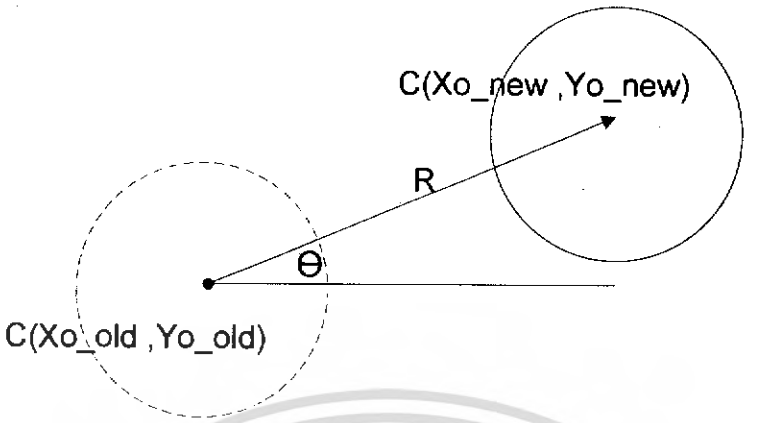


รูปที่ 2.15 การเคลื่อนที่ของเส้นตรงตามทิศทางของลูกศร

การเคลื่อนที่ของรูปวงกลมจะเคลื่อนที่ไปในทิศทางของเส้นตรง CD ที่กล่าวมาข้างต้น สามารถทำได้โดยการใช้สมการ

$$Xo_{ใหม่} = R.cos() + Xo_{เก่า}$$

$$Yo_{ใหม่} = R.sin() + Yo_{เก่า}$$



รูปที่ 2.16 การเคลื่อนที่ของวงกลมตามทิศทางของลูกศร

โดยที่ R คือ ระยะทางในการเคลื่อนที่

Θ คือ มุมระหว่างเส้นตรงโดยวัดจากแกน x

Xxใหม่ คือ ค่าพิกัด x ใหม่ ที่ได้เมื่อมีการเคลื่อนที่เป็นระยะ R

Yxใหม่ คือ ค่าพิกัด y ใหม่ ที่ได้เมื่อมีการเคลื่อนที่เป็นระยะ R

Xxเก่า คือ ค่าพิกัด x เก่าก่อนที่จะมีการเคลื่อนที่

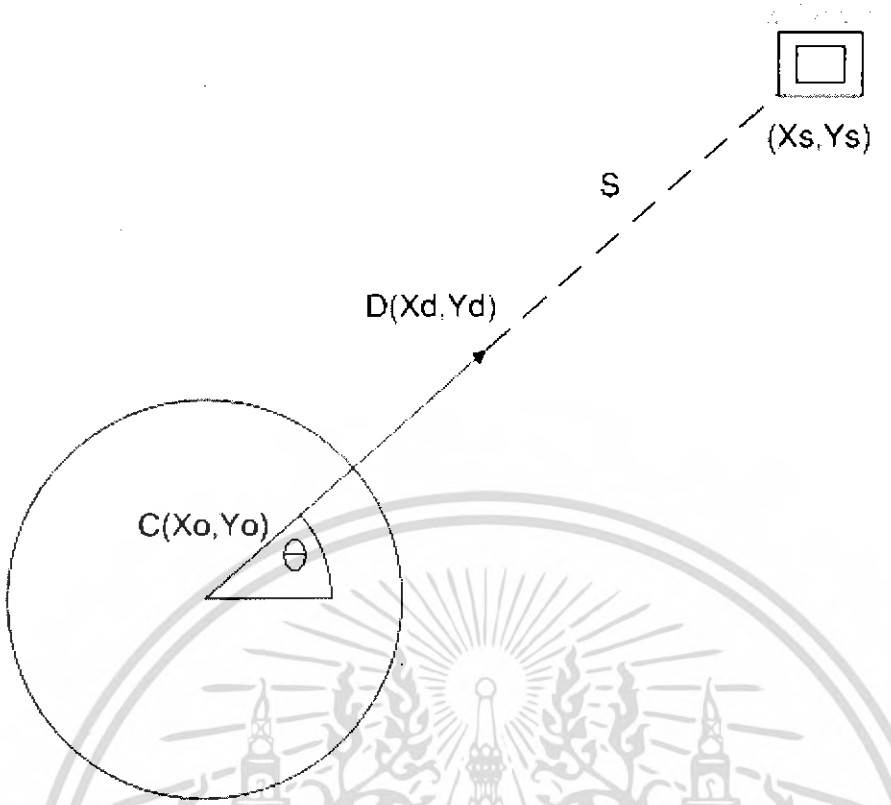
Yxเก่า คือ ค่าพิกัด y เก่าก่อนที่จะมีการเคลื่อนที่

การแสดงระยะห่างระหว่างวัตถุที่พบกับ Robot object

จากการใช้ SONAR ทำให้ได้ระยะที่ใกล้ที่สุดจากวัตถุกีดขวางกับตัว Robot ดังนั้นสามารถแสดงระยะห่างเป็นลักษณะของรูปภาพได้ ในที่นี้จะทำการหาตำแหน่งของวัตถุกีดขวางได้จากสมการดังนี้

$$Xs = S.cos(\Theta) + X0$$

$$Ys = S.sin(\Theta) + Y0$$



รูปที่ 2.17 การแสดงระยะของวัตถุกีดขวาง

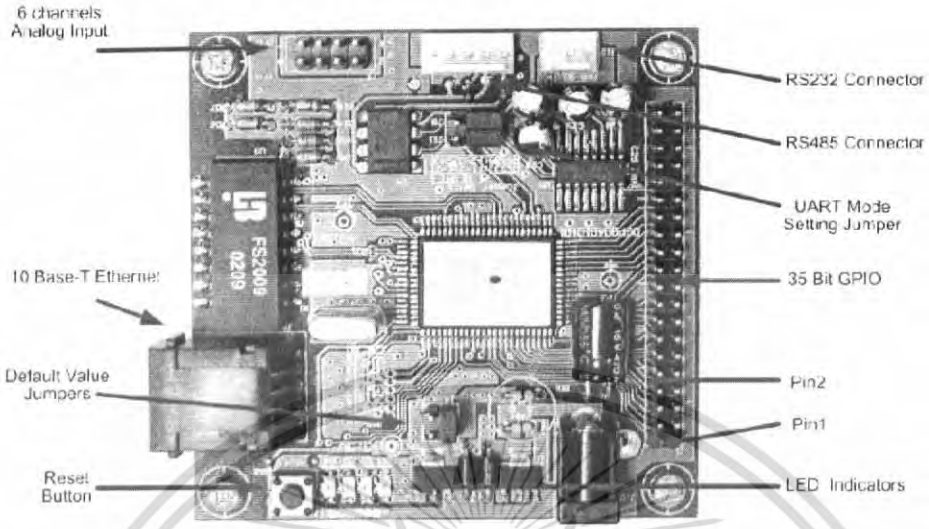
โดยที่ S คือ ระยะห่างระหว่างตัว Object กับวัตถุสิ่งกีดขวาง

θ คือ มุมระหว่างทิศทางของ object กับแกน x

Xs คือ ค่าพิกัด x ของวัตถุกีดขวางที่ระยะ S

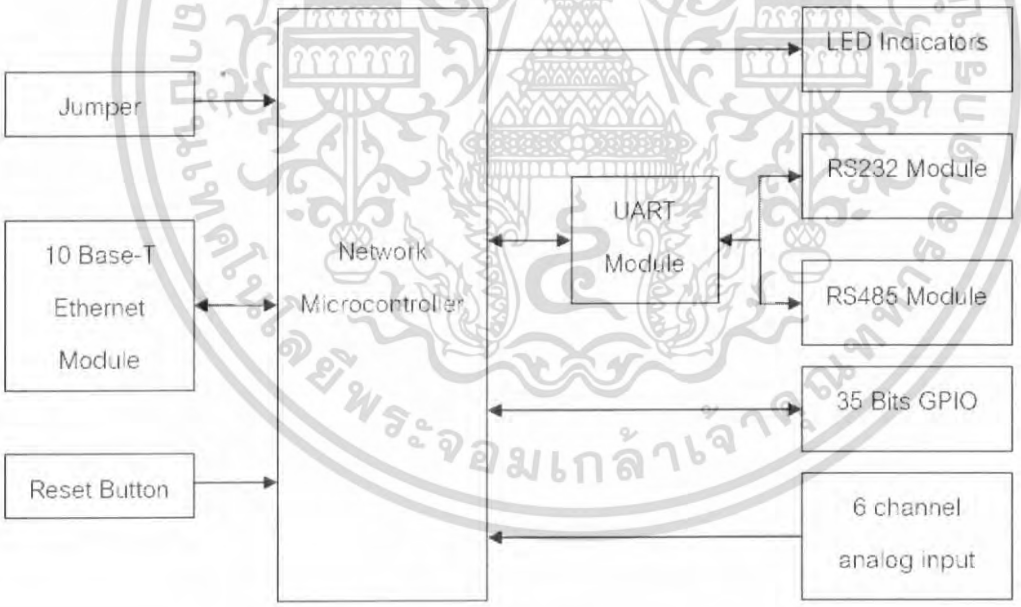
Ys คือ ค่าพิกัด y ของวัตถุกีดขวางที่ระยะ S

2.7 Ethernet IO Board



รูปที่ 2.18 Ethernet IO Board

2.7.1 ส่วนประกอบของ Ethernet IO Board



รูปที่ 2.19 Block Diagram ของ Ethernet IO Board

1. 10 Base-T Ethernet Module

   - ใช้ในการเชื่อมต่อ Ethernet IO Board กับระบบ Network ผ่านสายแลน โดยใช้ได้ทั้งสาย ตรง และ สายครอส

2. UART Module

- ใช้สำหรับเชื่อมต่อผ่านซีเรียล พอร์ท โดย UART Module มีความเร็วสูงสุดอยู่ที่ 115200 bps ซึ่งจะมีแยกได้อีก 2 โหมด คือ RS232 และ RS485 ซึ่งเราไม่สามารถ ใช้ได้พร้อมกันทั้ง 2 โหมด พร้อมกัน จะแยกการทำงานโดยเซท Jumper J8 และ J11

ตารางที่ 2.10 การเซท Jumper ของ RS232 และ RS485

UART MODE	J18		J11	
	1-2	2-3	1-2	2-3
RS232	Closed	Open	Closed	Open
RS485	Open	Closed	Open	Closed

3. RS232 Module

- ใช้สำหรับต่อ Ethernet IO Board เข้ากับคอมพิวเตอร์ หรือวงจรอื่นๆ

ตารางที่ 2.11 ขาของโมดูล RS232

J13 PIN#	Signal Name	Description
1	Tx	Transmitted data from Ethernet IO board
2	Rx	Received data to Ethernet IO board
3	GND	Ground

4. RS485 Module

- ใช้สำหรับเชื่อมต่อกับวงจรอื่นๆในระยะทางที่มากกว่า 1000 เมตร

ตารางที่ 2.12 ขาของโมดูล RS485

J13 PIN#	Signal Name	Description
1	TX+	Noninverting Driver Output
2	TX-	Inverting Driver Output

3	GND	Ground
4	RX+	Noninverting Receiver Input
5	RX-	Inverting Receiver Input

5. 6 Channels Analog Input

- Ethenet IO Board มีช่อง 6 ช่องสำหรับการแปลงสัญญาณอนาล็อกเป็นดิจิตอลขนาด 10 บิท โดยมีอัตราการแซมปลิ้งสูงสุดอยู่ที่ 48 kHz และมีค่าความต่างศักย์อ้างอิงอยู่ที่ (+2.5V) และค่าความต่างศักย์สูงสุดจะอ่านค่าได้เท่ากับ 0x3FF และค่าความต่างศักย์อ้างอิง (+2.5V) จะอ่านได้เท่ากับ 0X3FE

ตารางที่ 2.13 ค่าการแปลงอนาล็อก เป็น ดิจิตอล

Analog Input Voltage	ADCs Value
0	0x000
2.5V / 0x3FE	0x001
2.5V	0x3FE
2.5V + (2.5V / 0x3FE)	0x3FF

6.35 Bits GPIO

- Ethernet IO Board มีขา GPIO จำนวน 35 ขาที่ใช้ในการควบคุมวงจรอื่นๆ โดยในส่วนของระบบความจำจะมีการรองรับการต่อ โดยใช้พอร์ทขนาน Ethernet IO Board จะทำงานเป็นตัวมาสเตอร์ในการเขียน/อ่าน ข้อมูลแล้วส่งไปยังฮาร์ดแวร์ แล้วฮาร์ดแวร์ก็ทำการเขียน/อ่าน ข้อมูลในรูปของข้อมูลขนาด 16 Bits

ตารางที่ 2.14 ขาของ 35 Bits GPIO Connector

J1 Pin #	Signal Name	Port Name	Type	Current(mA)		Description
				Sink	Source	
1	+5V	-	-	-	-	Power + 5V
2	GPIO1	RA3	I/O	24	24	General Purpose Input/Output

3	A0	RB0	I/O	8	8	Address 0
4	A1	RB1	I/O	8	8	Address 1
5	GPIO2	RB2	I/O	8	8	General Purpose Input/Output
6	GPIO3	RB3	I/O	8	8	General Purpose Input/Output
7	Wr	RB4	I/O	8	8	Write Signal (active low)
8	Rd	RB5	I/O	8	8	Read Signal (active low)
9	GPIO4	RB6	I/O	8	8	General Purpose Input/Output
10	GPIO5	RB7	I/O	8	8	General Purpose Input/Output
11	D8	RC0	I/O	4	4	Data 8
12	D9	RC1	I/O	4	4	Data 9
13	D10	RC2	I/O	4	4	Data 10
14	D11	RC3	I/O	4	4	Data 11
15	D12	RC4	I/O	4	4	Data 12
16	D13	RC5	I/O	4	4	Data 13
17	D14	RC6	I/O	4	4	Data 14
18	D15	RC7	I/O	4	4	Data 15
19	Gnd	-	-	-	-	Ground
20	Gnd	-	-	-	-	Ground
21	D0	RD0	I/O	4	4	Data 0
22	D1	RD1	I/O	4	4	Data 1
23	D2	RD2	I/O	4	4	Data 2
24	D3	RD3	I/O	4	4	Data 3
25	D4	RD4	I/O	4	4	Data 4
26	D5	RD5	I/O	4	4	Data 5
27	D6	RD6	I/O	4	4	Data 6
28	D7	RD7	I/O	4	4	Data 7

29	GPIO6	RE0	I/O	8	8	General Purpose Input/Output
30	GPIO7	RE1	I/O	8	8	General Purpose Input/Output
31	GPIO8	RE2	I/O	8	8	General Purpose Input/Output
32	GPIO9	RE4	I/O	8	8	General Purpose Input/Output
33	GPIO10	RE6	I/O	24	24	General Purpose Input/Output
34	GPIO11	RE7	I/O	8	8	General Purpose Input/Output
35	GPIO12	RF4	I/O	8	8	General Purpose Input/Output
36	GPIO13	RF5	I/O	8	8	General Purpose Input/Output
37	GPIO14	RF6	I/O	8	8	General Purpose Input/Output
38	GPIO15	RF7	I/O	8	8	General Purpose Input/Output
39	Gnd	-	-	-	-	Ground
40	Gnd	-	-	-	-	Ground

7. Reset Button

- ใช้สำหรับรีเซทค่าของ Ethernet IO Board

8. Jumper

- ใช้ในการเลือกว่าจะใช้ค่าดีฟอลท์ หรือ หรือใช้ค่าที่เซทไว้ ถ้า Jumper J4 ปิดแสดงว่า Ethernet IO Board จะใช้ค่าดีฟอลท์ ทั้ง ไอพี แอดเดรส และ แมคแอดเดรส แต่ถ้า Jumper J4 เปิดจะใช้ค่า ไอพีแอดเดรส และแมค แอดเดรส ที่ทำการเซทไว้

Ethernet IO Board ได้มีการสำรอง Jumper J6 ไว้รองรับสำหรับให้ผู้ที่อาจต้องใช้ในส่วนของแอพพลิเคชั่นได้

ตารางที่ 2.15 การเซทค่าของ Jumper

Jumper	Port Name	Status	Logic	Description
JP4	RB2	Closed	Low	Use default IP Address and Mac Address
		Open	High	Use user setting IP Address and Mac Address
JP6	RB3	Closed	Low	User define
		Open	High	User define

9. LED Indicator

- หลอดแอลอีดี จะแสดงสถานะของ Network ของ Ethernet IO Board

ตารางที่ 2.16 ค่าของ LED Indicator

LED	Signal	Description
D1	Power	Bright when power on Ethernet IO board
D2	Link	Bright when connect Ethernet IO board to network
D3	ACK	Bright when send or receive data from network
D4	Col	Bright when data collision occur

2.7.2 หลักการเบื้องต้นทำงาน Ethernet IO Board

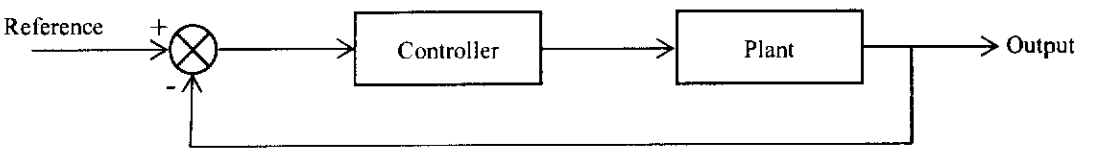
- Ethernet IO Board จะรับค่าข้อมูลมาจาก ไมโครคอนโทรลเลอร์ ผ่านซีเรียงพอร์ท แล้วจะทำการส่งผ่านสายแลนไปยัง คอมพิวเตอร์ โดยเราสามารถเขียนโปรแกรมเพิ่มเติมในการกำหนดวิธีการรับส่งข้อมูลได้

2.8 ทฤษฎี PID Control

ในการควบความเร็ว DC มอเตอร์ จะใช้การควบคุมโดยใช้สัญญาณ PWM ในการควบคุมความเร็ว และในการควบคุมสัญญาณ PWM นั้นจะมีทฤษฎีการควบคุมตัวหนึ่งที่เรียกว่า PID Control

2.8.1. ตัวควบคุมแบบ PID

ตัวควบคุมในตระกูลพีไอดี(PID)ที่ใช้กันอยู่ในปัจจุบันจะเป็นลักษณะตัวควบคุมแบบป้อนกลับ (Close Loop) ซึ่งมีโครงสร้างโดยทั่วไปดังรูปที่ 2.20



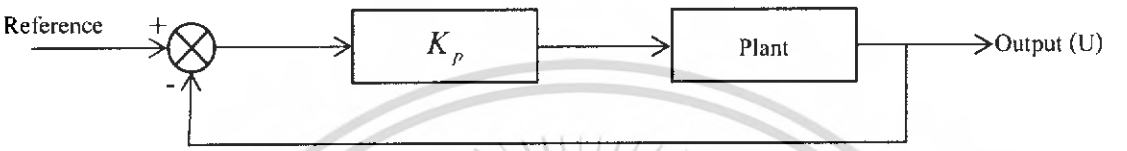
รูปที่ 2.20 โครงสร้างทั่วไปของตัวควบคุมตระกูล PID

2.8.2 ตัวควบคุมแบบพี (P : Proportional Controller)

ตัวควบคุมแบบสัดส่วนนี้เป็นตัวควบคุมที่สามารถสร้างได้ง่ายโดยอาศัยหลักการพื้นฐานของการขยายสัญญาณโดยมีความสัมพันธ์ระหว่างผลลัพธ์ของตัวควบคุมกับสัดส่วนความผิดพลาดดังสมการ

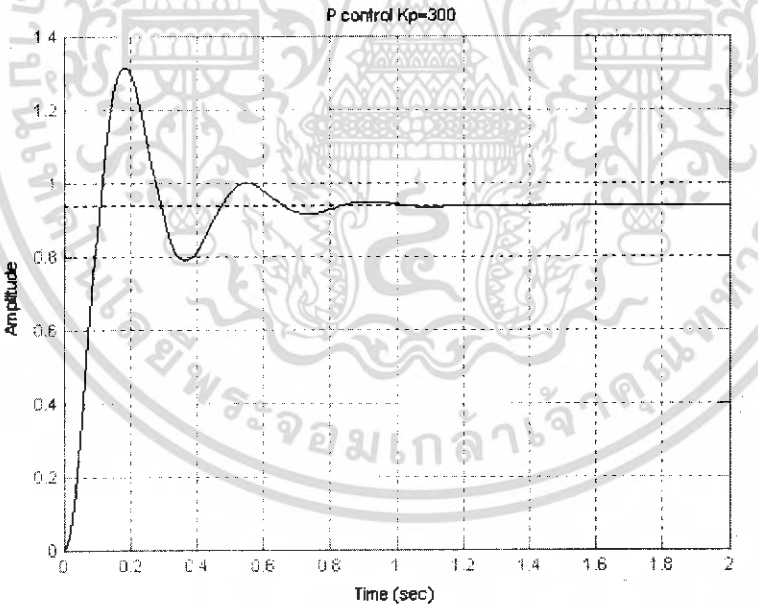
$$U = K_p e$$

ซึ่งตัวควบคุมมีโครงสร้างดังรูปที่ 2.21



รูปที่ 2.21 โครงสร้างของตัวควบคุมแบบสัดส่วน

ผลตอบสนองการทำงานแสดงได้ดังรูปที่ 2.22



รูปที่2.22ผลตอบสนองในโดเมนเวลาของตัวควบคุมแบบสัดส่วน

ข้อดีของตัวควบคุมแบบนี้คือสร้างได้ง่าย เนื่องจากมีสัมประสิทธิ์เพียงตัวเดียวที่จะต้องปรับค่า ส่วนข้อเสียก็คือตัวควบคุมแบบนี้จะทำให้เกิดการแกว่งของสัญญาณได้ และเกิดความคลาดเคลื่อนจากจุดหมายที่ต้องการที่เรียกว่าเกิดออฟเซ็ท (offset) ขึ้น

2.8.3 ตัวควบคุมแบบไอ (I : Integral controller)

ตัวควบคุมแบบอินทิกรัลนี้เป็นตัวควบคุมที่อาศัยหลักการอินทิกรัลในการสร้างสัญญาณควบคุมขึ้น

$$U = K_i \int e dt$$

ผลของการใช้งานตัวควบคุมแบบอินทิกรัลจะทำให้สามารถกำจัดความคลาดเคลื่อนของระบบออกได้จนหมดซึ่งสามารถแสดงได้ดังนี้

$$\frac{Y(s)}{R(s)} = \frac{G_p(s)}{1+G_p(s)}$$

$$Y(s) = E(s)G_p(s)$$

$$E(s) = \frac{R(s)}{1+G_p(s)}$$

$$e = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) = \lim_{s\to 0} \frac{sR(s)}{1+G_p(s)} = \lim_{s\to 0} \frac{1}{1+G_p(s)} = \frac{1}{1+\infty} = 0$$

ข้อดีของตัวควบคุมแบบนี้คือจะสามารถกำจัดความคลาดเคลื่อนที่เกิดจากการใช้ตัวควบคุมแบบสัดส่วน ส่วนข้อเสียคือจะทำให้ระบบทำงานได้ช้าลงซึ่งโดยปกติแล้วเราจะใช้งานตัวควบคุมแบบ I นี้ร่วมกับตัวควบคุมแบบสัดส่วน ซึ่งมีชื่อเรียกว่าตัวควบคุมแบบพีไอ (PI)

2.8.4 ตัวควบคุมแบบพีไอ (PI : Proportional Integral Controller)

ตัวควบคุมแบบนี้สร้างขึ้นเพื่อแก้ไขข้อเสียของตัวควบคุมแบบสัดส่วนโดยอาศัยการเพิ่มขึ้นของอันดับโดยสัญญาณที่ป้อนให้แก่ระบบจะแบ่งออกเป็นสองส่วนคือสัญญาณที่เป็นสัดส่วน และสัญญาณที่เป็นค่าอินทิกรัลของความคลาดเคลื่อนตามเวลาโดยที่อัตราขยายสัญญาณเป็นค่าคงที่ ซึ่งสามารถแสดงได้ตามสมการ

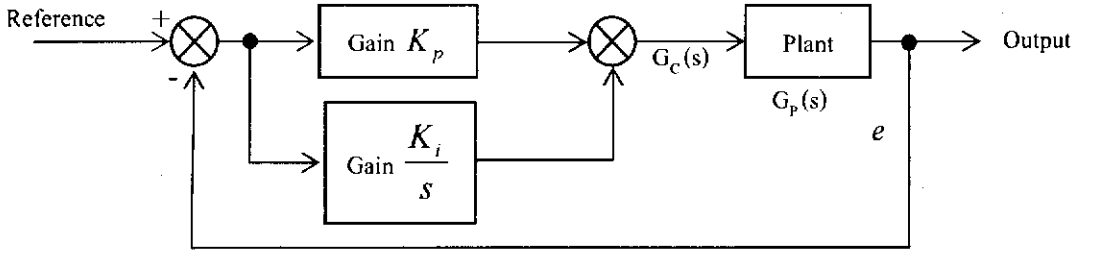
ฟังก์ชันถ่ายโอนคือ

$$G_c(s) = K_P + \frac{K_i}{s}$$

ผลตอบสนองแสดงได้ดังสมการ

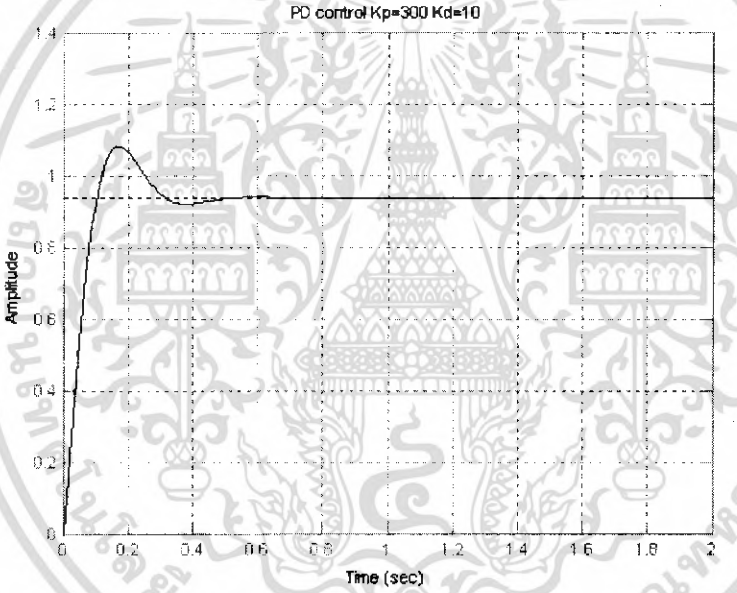
$$U = eG_C(s)G_P(s)$$

ตัวควบคุมพีไอ มีโครงสร้างดังรูป 2.23



รูปที่ 2.23 โครงสร้างของ Proportional Integral Controller

ผลตอบสนองการทำงานแสดงได้ดังรูปที่ 2.24



รูปที่ 2.24 ผลตอบสนองในโดเมนเวลาของตัวควบคุมแบบ PI

2.8.5 ตัวควบคุมแบบดี (D : Derivative Controller)

ตัวควบคุมแบบอันดับนี้จะช่วยขจัดข้อผิดพลาดของผลลัพธ์ที่ได้จากตัวควบคุมแบบสัดส่วน ผลลัพธ์จะถูกคำนวณจากอัตราการเปลี่ยนแปลงของความผิดพลาดต่อเวลา

$$U = K_p(\frac{de}{dt})$$

ตัวควบคุมแบบอันดับนี้จะไม่ถูกนำมาใช้เดี่ยว ๆ เนื่องจากการเปลี่ยนแปลงอย่างรวดเร็วของระบบที่ใช้งานตัวควบคุมแบบอันดับนี้จะจะชดเชยผลลัพธ์อย่างรวดเร็วผลในระยะยาวจะก่อให้เกิดความคลาดเคลื่อนที่สูง(Overshoot) ข้อดีของตัวควบคุมแบบนี้คือจะช่วยเพิ่มเสถียรภาพของระบบและลดovershoot และทำให้ผลตอบสนองชั่วคราวดีขึ้น

2.8.6 ตัวควบคุมแบบพีดี (PD : Proportional Derivative Controller)

ตัวควบคุมแบบพีดีนี้จะแก้ข้อผิดพลาดโดยจะสร้างสัญญาณควบคุมก่อนหน้าที่จะเกิดความผิดพลาดขึ้นดังนั้นจึงเหมาะกับระบบที่มีการเหลื่อมของเวลาค่อนข้างมากซึ่งสัญญาณควบคุมจะแปรตามอัตราการเปลี่ยนแปลงของสัญญาณผิดพลาด

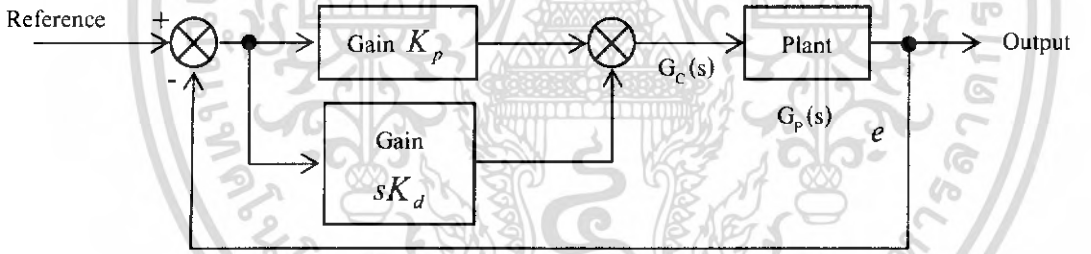
ฟังก์ชันถ่ายโอนแสดงได้ดังสมการ

$$G_c(s) = K_p + sK_d$$

ผลตอบสนองแสดงได้ดังสมการ

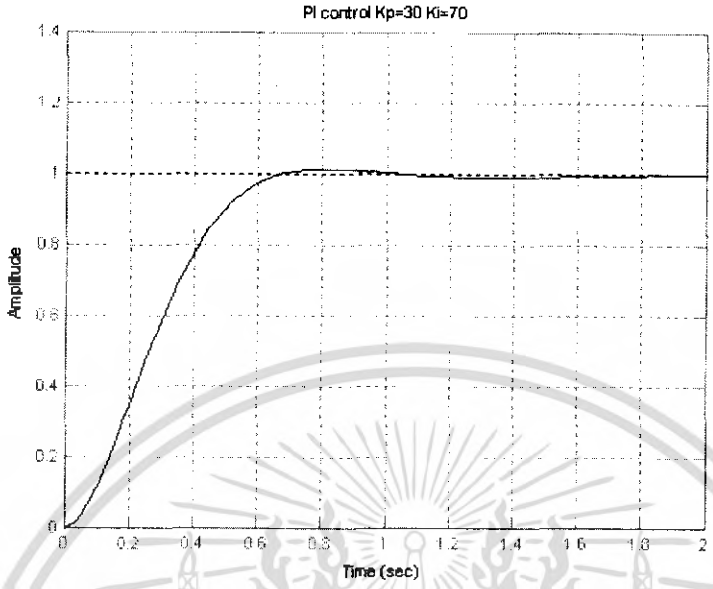
$$U = R(s)G_C(s)G_P(s)$$

ตัวควบคุมแบบ PD มีโครงสร้างดังรูปที่ 2.25



รูปที่ 2.25 โครงสร้างของ Proportional Derivative Controller

ผลตอบสนองการทำงานแสดงได้ดังรูปที่ 2.26



รูปที่ 2.26 ผลตอบสนองในโดเมนเวลาของตัวควบคุม PD

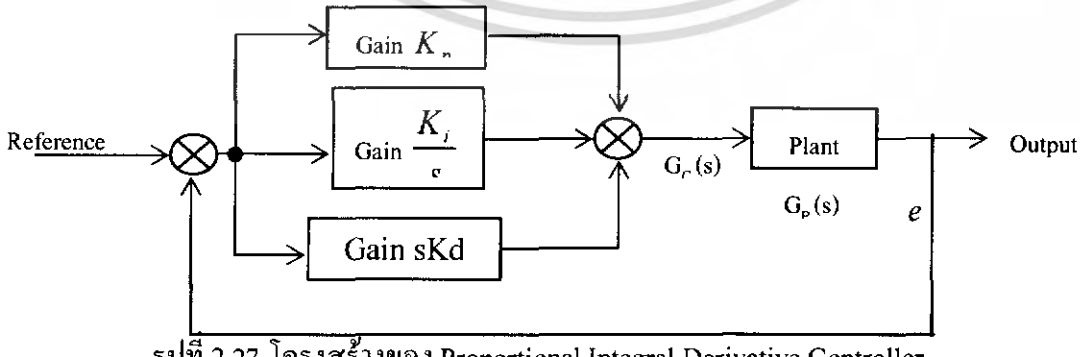
2.8.7 ตัวควบคุมแบบพีไอดี (PID : Proportional Integral Derivative Controller)

ตัวควบคุมชนิดนี้จะรวมเอาข้อดีของตัวควบคุมทั้งสามชนิดคือใช้อัตราขยายสัญญาณของตัวควบคุมแบบสัดส่วน ลดค่าความคลาดเคลื่อนโดยคุณสมบัติของตัวควบคุมแบบอินทริกัล และลดค่าความผิดพลาดด้วยคุณสมบัติของตัวควบคุมแบบเดริเวทีฟ

โดยในตัวควบคุมแบบ PID นี้จะประกอบด้วยส่วนของตัวควบคุม PI และ PD ซึ่งมีสมการของฟังก์ชันถ่ายโอนดังนี้

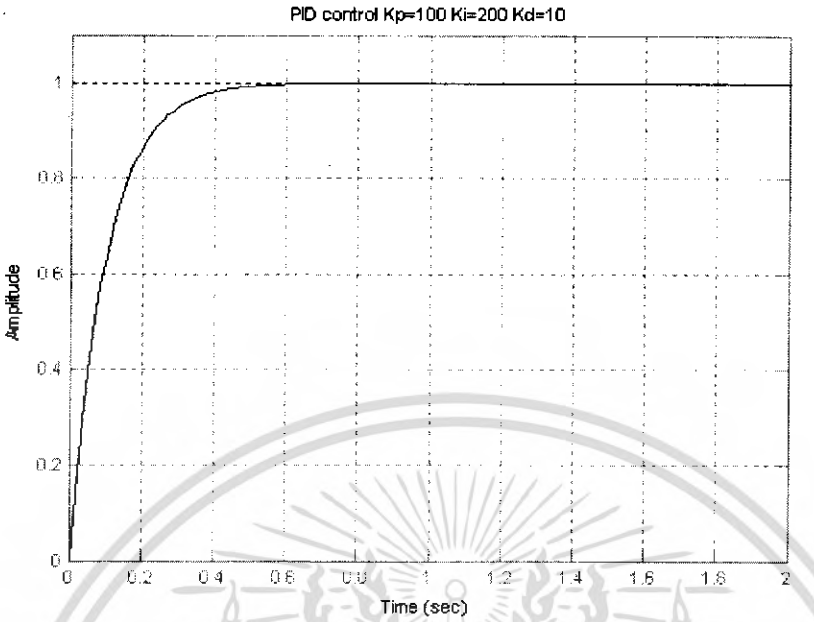
$$G_c(s) = K_P(1 + \frac{1}{T_i s} + T_d s)$$

ตัวควบคุมแบบพีไอดีมีโครงสร้างดังรูปที่ 2.27



รูปที่ 2.27 โครงสร้างของ Proportional Integral Derivative Controller

ผลตอบสนองการทำงานแสดงได้ดังรูปที่ 2.28



รูปที่ 2.28 ผลตอบสนองในโดเมนเวลาของตัวควบคุมแบบ PID

เมื่อได้ศึกษาทฤษฎีจากบทที่ 2 แล้วจะนำไปใช้กับขั้นตอนและวิธีการดำเนินงานและการออกแบบการทดลอง โดยจะแสดงในบทที่ 3

บทที่ 3
การออกแบบและวิธีการดำเนินงานวิจัย

3.1 การออกแบบการทำงานของระบบ

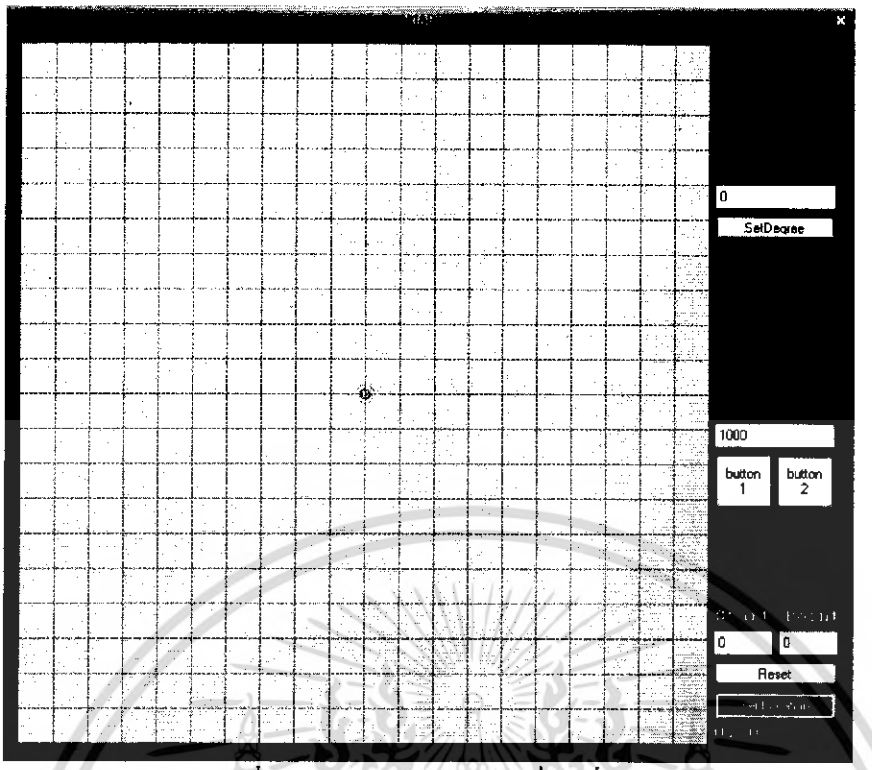
3.1.1 ส่วน User Control and Display

การทำงานในส่วนนี้จะประกอบด้วยส่วนหลัก 2 ส่วนคือ

- ส่วนโปรแกรมที่ใช้แสดงผลและควบคุมการทำงานติดต่อกับผู้ใช้

รูปที่ 3.1 Application User Control and Display

การออกแบบในส่วนนี้จะแบ่งออกเป็นส่วนๆ โดยส่วนที่เป็นพื้นสีขาว จะส่วนหน้าจอแสดงภาพ บริเวณสีเหลืองเป็นส่วนที่จะใช้ในการเชื่อมต่อระหว่างคอมพิวเตอร์ กับตัวหุ่นยนต์ โดยจะสามารถกำหนดไอพีได้ ว่าจะติดต่อกับหุ่นยนต์ไอพีอะไร บริเวณสีเขียวเป็นส่วนของ Joystick ที่ใช้ในการควบคุมหุ่นยนต์ บริเวณสีฟ้าเป็นส่วนที่ใช้สำหรับควบคุมกล้อง ส่วนที่เหลือทางด้านขวาของจอจะใช้สำหรับเปิดแอพพลิเคชั่นอื่นๆ คือ หน้าจอสร้างแผนที่ หน้าจอ Debug ตรงส่วนเข็มจะเป็นตัวบอกความเร็ว โดยเข็มสีฟ้าจะใช้สำหรับดูค่า Return Speed และสีเขียว ใช้สำหรับดูค่า Target Speed

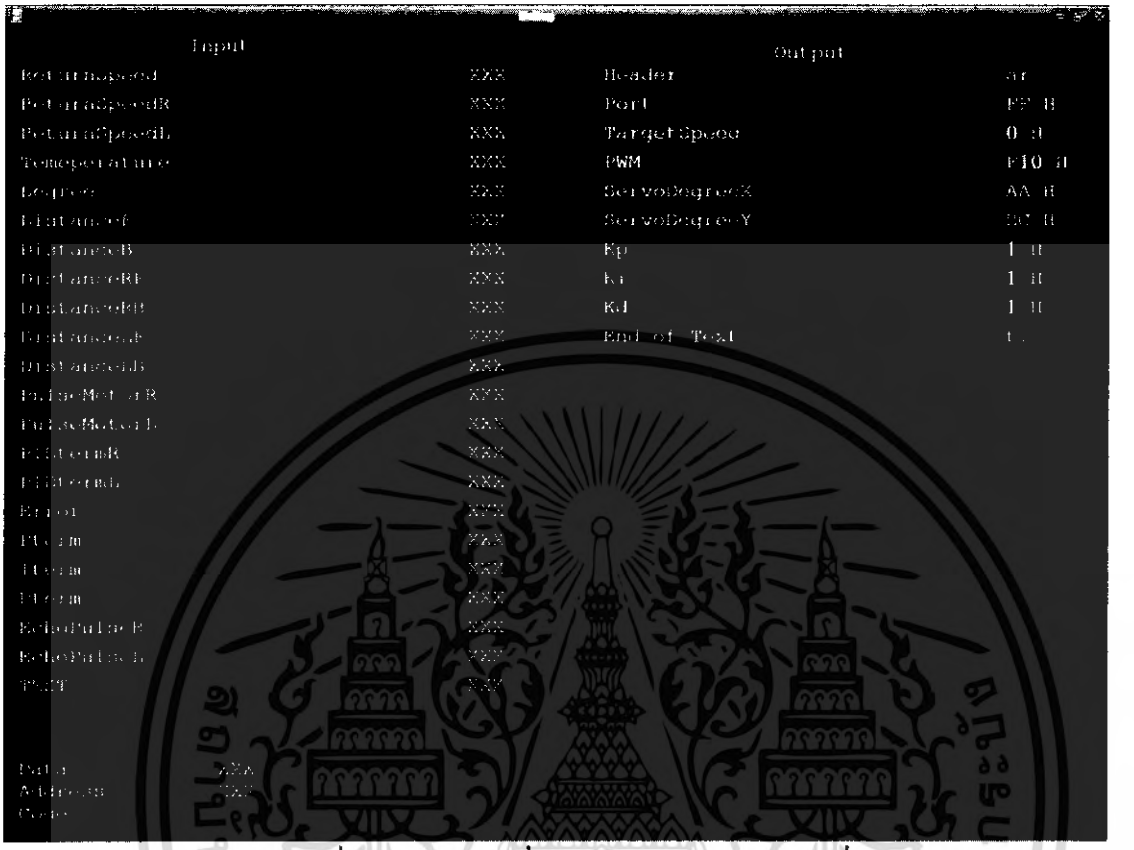


รูปที่ 3.2 หน้าจอแสดงการเคลื่อนที่ของหุ่นยนต์

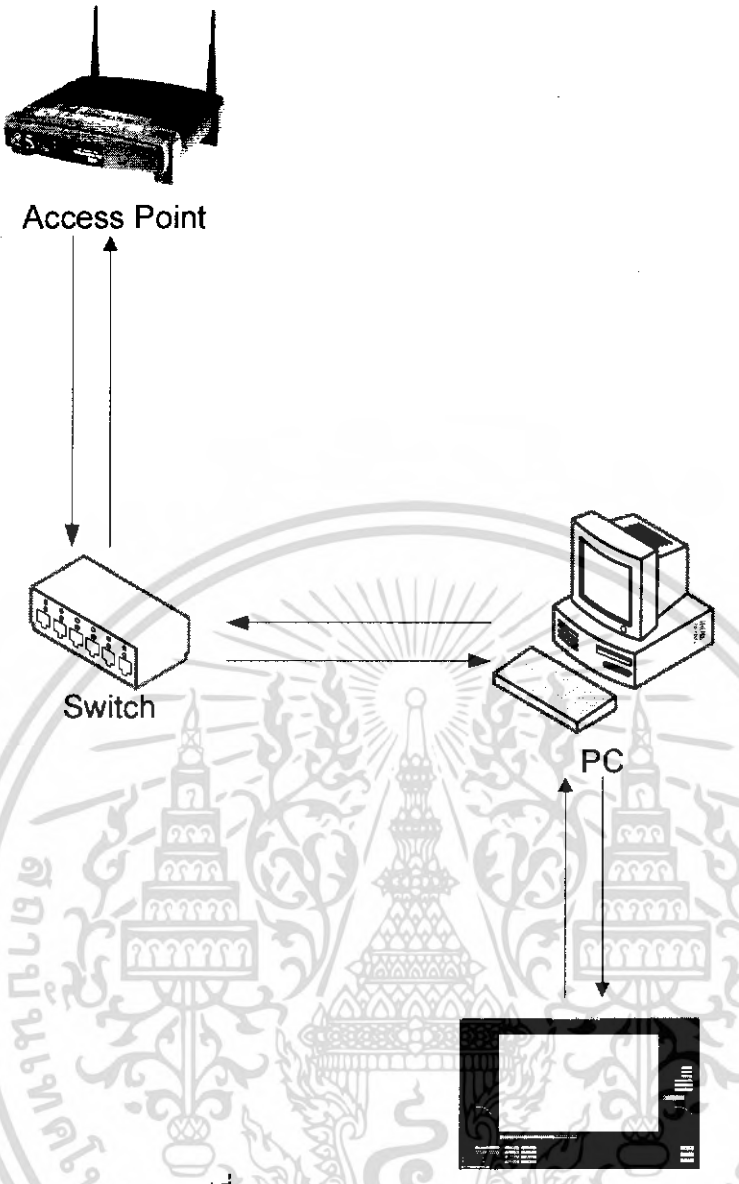
รูปที่ 3.3 หน้าจอ Debug แสดงค่าต่างๆที่รับจากหุ่นยนต์

- ส่วนของคอมพิวเตอร์และการรับส่งข้อมูล

การทำงาน คือ ส่วนของโปรแกรมจะทำการส่งข้อมูลออกจากคอมพิวเตอร์ผ่านการ์ด Wireless LAN



เข้ายัง Ethernet IO Board ที่ตัวหุ่นยนต์โดยที่หุ่นยนต์จะมี Access Point ที่ใช้สำหรับรับข้อมูล จากนั้น Ethernet IO Board จะทำการส่งข้อมูลเข้าไปยัง User Control & Display Unit เพื่อสั่งให้หุ่นยนต์ทำงานตามที่ต้องการ ในเวลาเดียวกัน Ethernet IO Board ก็จะรับค่าจากไมโครคอนโทรลเลอร์แล้วส่งกลับไปยังคอมพิวเตอร์เพื่อแสดงผล

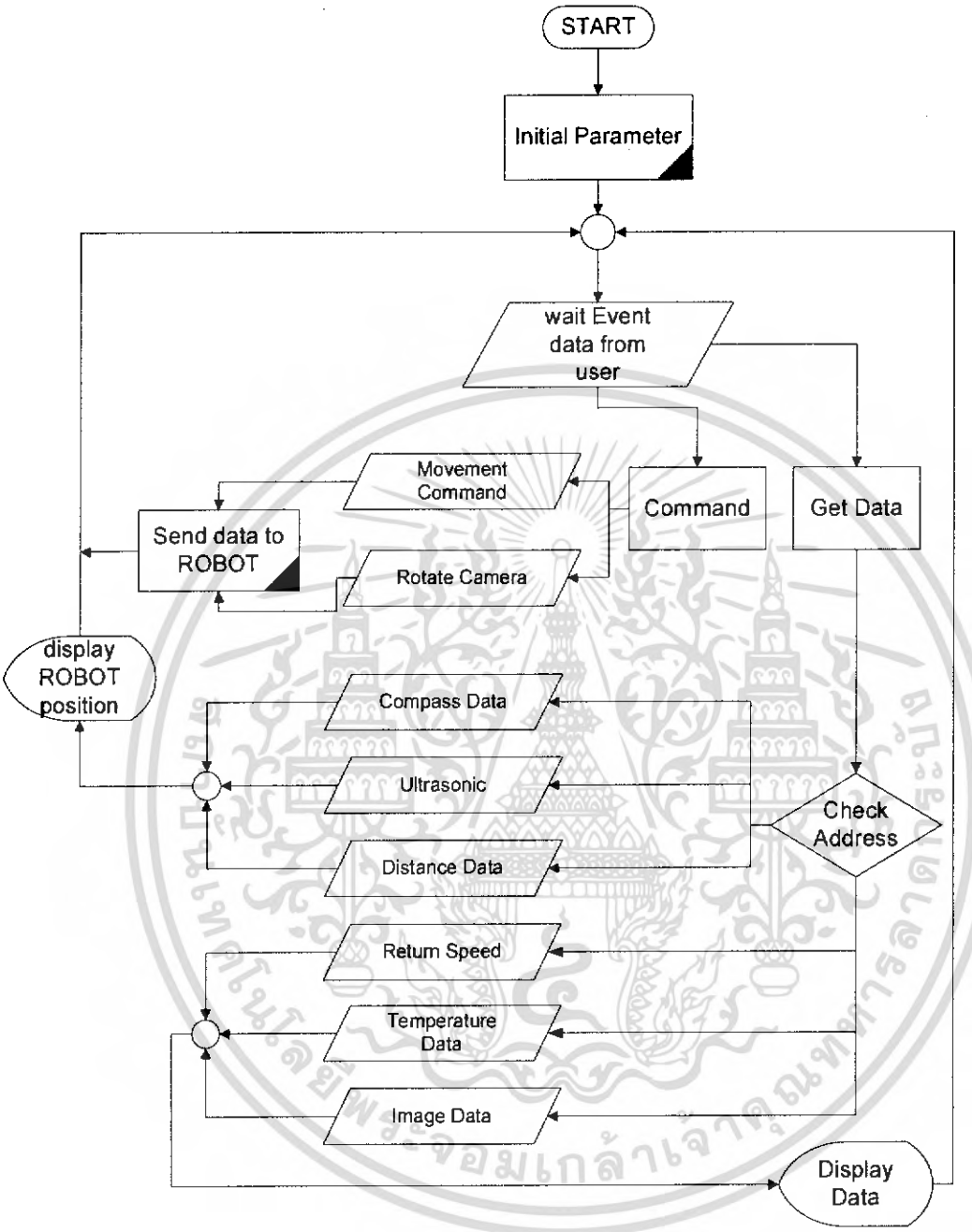


รูปที่ 3.4 Block User Control and Display

ในส่วนนี้การเขียนโปรแกรมการทำงานจะแสดงตามรายละเอียดใน flow chart โดยสรุปได้ดังนี้

- ทำการกำหนดค่าของพารามิเตอร์ต่างๆให้กับโปรแกรม
- รอรับ Input จาก User โดย User จะเป็นคนสั่งให้โปรแกรมทำอะไร
- ทำการตรวจสอบว่า Input ที่รับมาเข้าเงื่อนไขอะไร เช่น เป็นเงื่อนไขในการเคลื่อนที่ Robot หรือสั่ง Scan Sonar
- ทำการส่งข้อมูลออกไปที่ Ethernet IO Board ผ่าน การ์ด Wireless LAN
- รอรับข้อมูลจาก User Control & Display Unit และแสดงผล

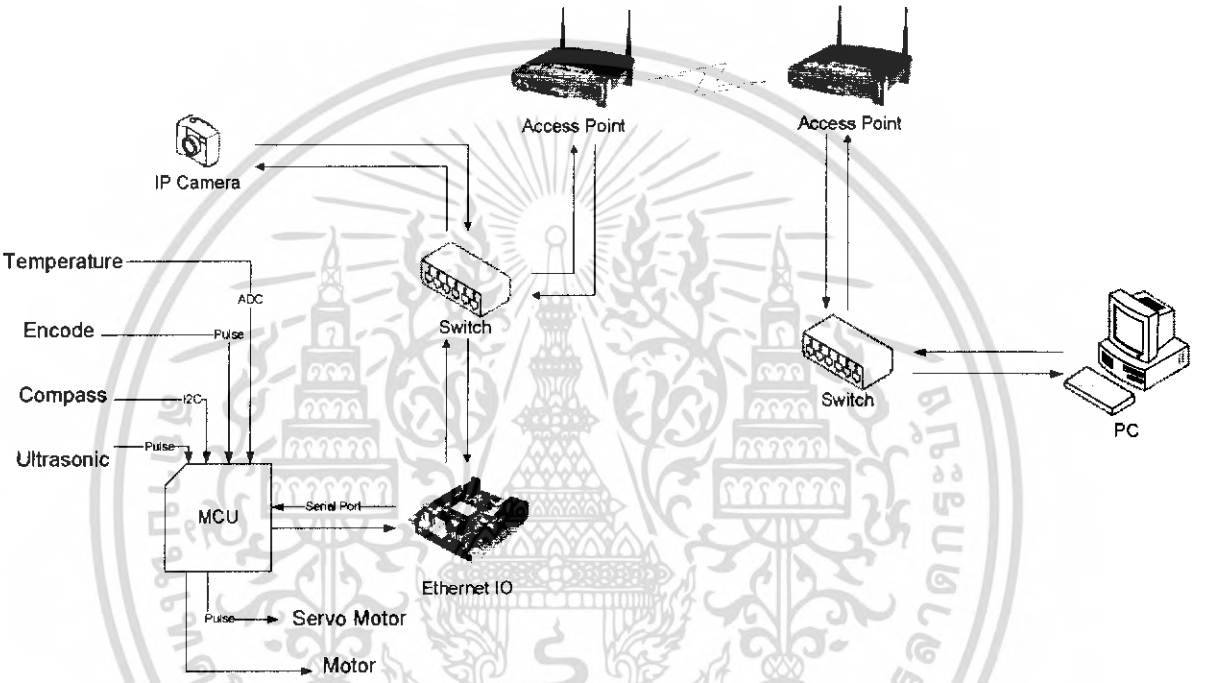
Flow Chart ของส่วน User Control & Display Unit แสดงได้ดังนี้



รูปที่ 3.5 Flow Chart User Control & Display Unit

3.1.2 ส่วน Device Control Unit

เนื่องจากส่วนนี้จะทำการควบคุมอุปกรณ์ต่างๆ โดยจะทำงานสัมพันธ์กับส่วน User Control and Display Unit ส่วนนี้จะทำการรับข้อมูลมา แล้วนำค่าที่รับได้มาทำการประมวลผลตามกระบวนการที่ได้โปรแกรมไว้ เพื่อใช้ควบคุมอุปกรณ์ต่างๆ และในขณะเดียวกันก็จะทำการส่งสัญญาณและข้อมูลต่างๆ กลับไปยังส่วน User Control and Display Unit โดยจะติดต่อได้ทั้งแบบเชื่อมต่อด้วยสายและแบบไร้สาย โดยใช้หลักของ TCP/IP ซึ่งจะส่งผ่านสายแลน หรือ ผ่าน Access Point ก็ได้



รูปที่ 3.6 Device Control Unit

ในส่วนนี้การเขียนโปรแกรมการทำงานจะแสดงตามรายละเอียดใน Flow chart โดยสรุปได้ดังนี้

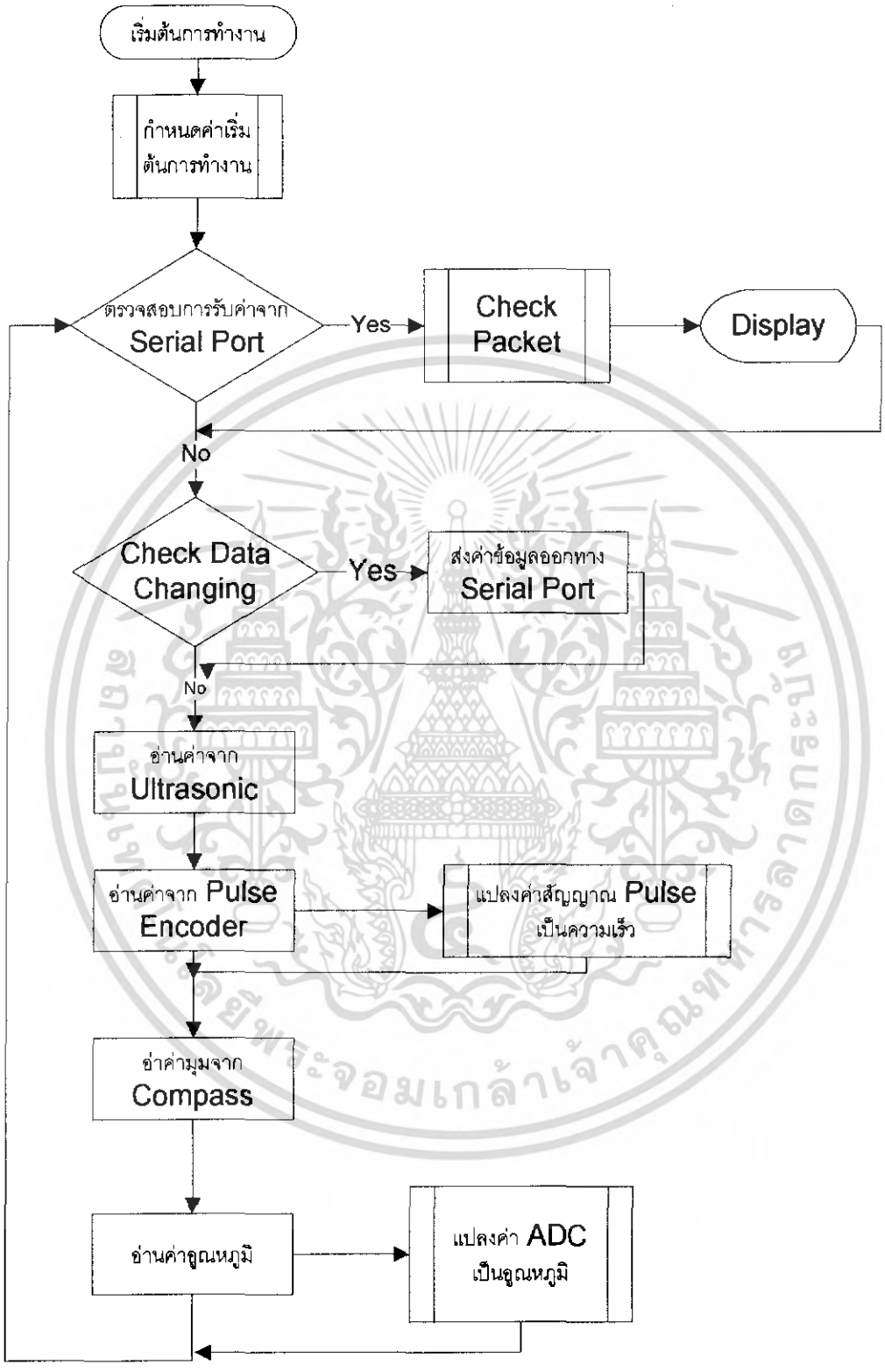
- ทำการกำหนดค่าของพารามิเตอร์ต่างๆให้กับโปรแกรม
- รอรับข้อมูลจากส่วน User Control and Display Unit
- เมื่อรับข้อมูลก็จะทำการตรวจสอบว่าตรงตามเงื่อนไขหรือไม่ เนื่องจากเงื่อนไขหลักๆมีอยู่ 3 เงื่อนไข ดังนี้

1. เงื่อนไขในการควบคุมการเคลื่อนที่หุ่นยนต์ เป็นฟังก์ชันที่ใช้ควบคุมให้หุ่นยนต์เคลื่อนที่ไปตามทิศทางต่างๆ โดยถ้าข้อมูลที่ส่งมาจากส่วน User Control and Display Unit ตรงกับเงื่อนไขที่ตั้งไว้มันก็จะทำงานตามกระบวนการ

2. เงื่อนไขในการควบคุมเซอร์โวมอเตอร์ จะเป็นฟังก์ชันที่ควบคุมให้เซอร์โวมอเตอร์หมุนไปทางซ้ายและขวา ครั้งละ 5 องศา

3. เงื่อนไขของการควบคุมการ Scan Sonar โดย Sonar จะทำงานได้นั้น เริ่มต้นเมื่อจ่ายไฟบวกและกราวด์แล้วเราต้องทำการสร้าง Trigger pulseให้ Sonar จากนั้นจะทำการสั่งให้ Timer ทำงาน แล้วสุดท้ายก็ทำการตรวจสอบว่ามีการรับ Echo pulse กลับมาหรือไม่ ถ้ามี Echo pulse กลับมาแสดงว่า Sonar ทำการตรวจพบวัตถุ ดังนั้นจะได้ค่าเวลา คือ Timer High:TH,Timer Low:TL ออกมาและในช่วงที่รอ Echo pulse ถ้าในระยะเวลา 36 ms ไม่มี Echo pulse กลับมาแสดงว่าตรวจไม่เจอวัตถุในระยะที่สามารถตรวจได้ (ระยะสูงสุดที่สามารถตรวจได้ขึ้นอยู่กับคุณสมบัติของโซนาร์) เมื่อได้ค่า (Timer hight:TH,Timer Low:TL) ก็จะทำการส่งกลับไปให้ User Control and Display เพื่อนำค่าไปคำนวณหาระยะทางและสร้างแผนที่ออกมา

Flow Chart ของส่วน Device Control Unit แสดงได้ดังนี้



รูปที่ 3.7 Flow chart Device Control Unit

3.2 หลักการของการเขียนโปรแกรมขับดีซีมอเตอร์

เมื่อเรารู้ว่าการทำงานของดีซีมอเตอร์คือต้องจ่ายแรงดันไฟฟ้ากระแสตรงให้ โดยเราจะอาศัยการจ่ายลอจิกจากไมโครคอนโทรเลอร์ไปที่วงจรขับดีซีมอเตอร์ โดยวงจรขับดีซีมอเตอร์ก็จะทำการขับเคลื่อนดีซีมอเตอร์ตามที่ได้สัญญาณมาจากไมโครคอนโทรเลอร์โดยเราจะทำการจ่ายลอจิกการทำงานโดยมีสองสถานะคือ 0 กับ 1 โดยถ้าจ่ายลอจิกกลับด้านกันก็จะทำให้มอเตอร์หมุนกลับทางนั่นเอง

ตัวอย่างโปรแกรม ARM7 LPC2138

```
void Timer1_PWM()                       ___irq
{
      unsigned long ChackInterrupt;
      T1TCR           = 0x00000000; // Timer1 Stop.
      ChackInterrupt  = T1IR;
      if (ChackInterrupt & 0x0001)  // Match1.0 for PWM.
      {
            switch(PID_TermR)       // Motor Right.
            {
            case 0x0000 :
                  PWMR  = 0x08;
                  P0    = ((P0_BUF | PWMR) & 0xFC) | (P0 & 0x03);
                  Port0(P0);
                  T1MR0 = T1TC + 0x00000FFF;
            break;
            case 0x0FFF :
                  PWMR  = 0x00;
                  P0    = (P0_BUF & 0xFC) | (P0 & 0x03);
                  Port0(P0);
                  T1MR0 = T1TC + 0x00000FFF;
            break;
            default :
                  if (PWMR == 0x08){
                        PWMR = 0x00;
                        P0 = (P0_BUF & 0xFC) | (P0 & 0x03);
                        Port0(P0);
                        T1MR0           = T1TC + PID_TermR;
                  }
                  else{
                        PWMR = 0x08;
                        P0 = ((P0_BUF | PWMR) & 0xFC) | (P0 & 0x03);
                        Port0(P0);
                        T1MR0 = T1TC + (0x00000FFF - PID_TermR);
                  }
            }
      }
      T1IR |= 0XFFFFFF00;  // Clear Timer Interrupt.
      VICVectAddr    &= 0x00000000;// Clear Vector Interrupt Address.
      T1TCR           = 0x00000001; // Timer1 Enable.
}
```

3.3 การควบคุมมอเตอร์เซอร์โว

ในการควบคุมมอเตอร์เซอร์โวและนำมาใช้งานนั้น นอกเหนือจากการจ่ายไฟและกราวด์แล้ว ต้องทำการจ่ายพัลส์ให้อีกขาหนึ่งเพื่อจะทำให้มอเตอร์หมุน และการหมุนนั้นสามารถหมุนได้ 2 ทิศทางและสามารถกำหนดองศาในการหมุนได้ (ภายในระยะ 0 – 180 องศาโดยประมาณ) ทั้งนี้ขึ้นอยู่กับขนาดของพัลส์ที่จ่ายให้กับมอเตอร์

ในโครงงานนี้ต้องการให้มอเตอร์เซอร์โวนั้นหมุนครั้งละ 5 องศา ได้ทั้งทวนเข็มและตามเข็มนาฬิกา จากคู่มือการใช้งานมอเตอร์เซอร์โวนั้นถ้าต้องการให้มอเตอร์อยู่ที่ตำแหน่ง 90 องศา ต้องจ่ายพัลส์บวกขนาด 1500 µs และพัลส์ช่วงลบอีก 20 ms

โปรแกรมในส่วนที่นับเวลาในการสร้างขนาดของพัลส์คือ

```
void Timer1_PWM()                                   __irq
{
      unsigned long ChackInterrupt;
      T1TCR           = 0x00000000;         // Timer1 Stop.
      ChackInterrupt  = T1IR;
      if (ChackInterrupt & 0x0004){         // Match 1.2 for Servo.
            switch (ServoSelect){
                  case 0x01 :
                  if (P1_BUF & 0x01){
                        P1_BUF          &= 0xFE; // CLR P1.0
                        Port1(P1_BUF);
                        T1MR2           = T1TC + ReturnServoX_L;
                        ServoSelect     = 0x02;
                  }
                  else{
                        P1_BUF          |= 0x01;   // SET P1.0
                        Port1(P1_BUF);
                        T1MR2           = T1TC + ReturnServoX_H;
                  }
                  break;
                  case 0x02 :
                  if (P1_BUF & 0x02){
                        P1_BUF          &= 0xFD;   // CLR P1.1
                        Port1(P1_BUF);
                        T1MR2           = T1TC + ReturnServoY_L;
                        ServoSelect     = 0x01;
                  }
                  else{
                        P1_BUF          |= 0x02;   // SET P1.0
                        Port1(P1_BUF);
                        T1MR2           = T1TC + ReturnServoY_H;
                  }
                  break;
            }
      }
      T1IR |= 0XFFFFFF00;           // Clear Timer Interrupt.
      VICVectAddr &= 0x00000000;    // Clear Vector Interrupt Address.
      T1TCR           = 0x00000001;// Timer1 Enable.
}
```

3.4 การควบคุมและใช้งานเข็มทิศ

โมดูลเข็มทิศดิจิตอล CMPS03 ออกแบบมาเพื่อช่วยในการกำหนดทิศทางการเคลื่อนที่ของหุ่นยนตร์อัตโนมัติ และนำมาใช้ในการสร้างเครื่องมือวัดและตรวจสอบทิศระบบอิเล็กทรอนิกส์โดย หัวใจสำคัญของโมดูลเข็มทิศคือตัวตรวจจับสนามแม่เหล็กโลก (Earth magnetic field) และไมโครคอนโทรลเลอร์เพื่อรับสัญญาณจากตัวตรวจจับมาประมวลผลเป็นข้อมูลดิจิตอลและสัญญาณพัลส์สำหรับแจ้งผลการวัด

การอ่านค่าทิศทางจากเอาต์พุตสัญญาณพัลส์ อ่านได้โดยขา PWM ของโมดูลโดยการนำค่าความกว้างของพัลส์ที่ได้จากเอาต์พุตสัญญาณพัลส์ของโมดูลมาระบุตำแหน่งองศาจาก 0 ถึง 359.9 องศาโดยมีย่านของค่าความกว้างสัญญาณพัลส์จาก 1มิลลิวินาทีไปจนถึง36.99มิลลิวินาทีมีความละเอียด0.1มิลิวินาทีต่อองศาในสัญญาณพัลส์แต่ละไซเกิลมีช่วงลอจิก 0 กว้าง 65 มิลลิวินาที ดังนั้นในการนำสัญญาณพัลส์มาประมวลผลเป็นค่ามุมจึงต้องใช้การนับค่าความกว้างของสัญญาณพัลส์เป็นหลักในการคำนวณค่าที่โมดูล

เมื่อทำการสร้างวงจร และตัวหุ่นยนต์เรียบร้อยแล้วขั้นตอนต่อไปเป็นการทดลองการควบคุมการเคลื่อนที่ของหุ่นยนต์ด้วยการทำงานแบบโดดเดี่ยว และการทำงานแบบมัลติเอเยนต์โดยแบล็คบอร์ด เพื่อทำการเปรียบเทียบประสิทธิภาพในสถานการณ์ต่าง ๆ

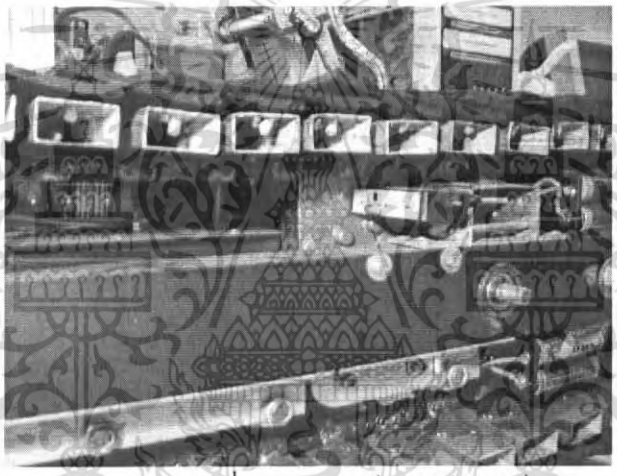
บทที่ 4

ผลการทดลอง

ในบทนี้กล่าวถึงขั้นตอนในการทำการทดลอง และ การทำงานของอุปกรณ์ต่างๆ ดังนี้

4.1 การวัดความเร็วของหุ่นยนต์โดยการวัดความกว้างของพัลส์ด้วยการเอ็นโค้ดเดอร์

การทดลองนี้เราจะนำตัวเอนโค้ดเดอร์มาตรวจจับความกว้างของพัลส์แล้วนำไปแสดงออกที่หน้าจอของโปรแกรม ซึ่งแสดงดังรูป



รูปที่ 4.1 อุปกรณ์เอนโค้ดเดอร์

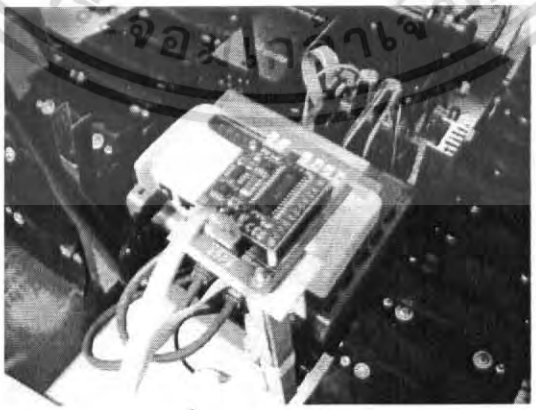
รูปที่ 4.2 พัลส์ที่ได้จาการทำงานของมอเตอร์


```
ReturnSpeed :          1248 D          4E0 H
ReturnSpeedR:          1056 D          420 H
ReturnSpeedL:          1280 D          500 H
```

รูปที่ 4.3 หน้าจอแสดงความเร็วรอบการทำงานของมอเตอร์

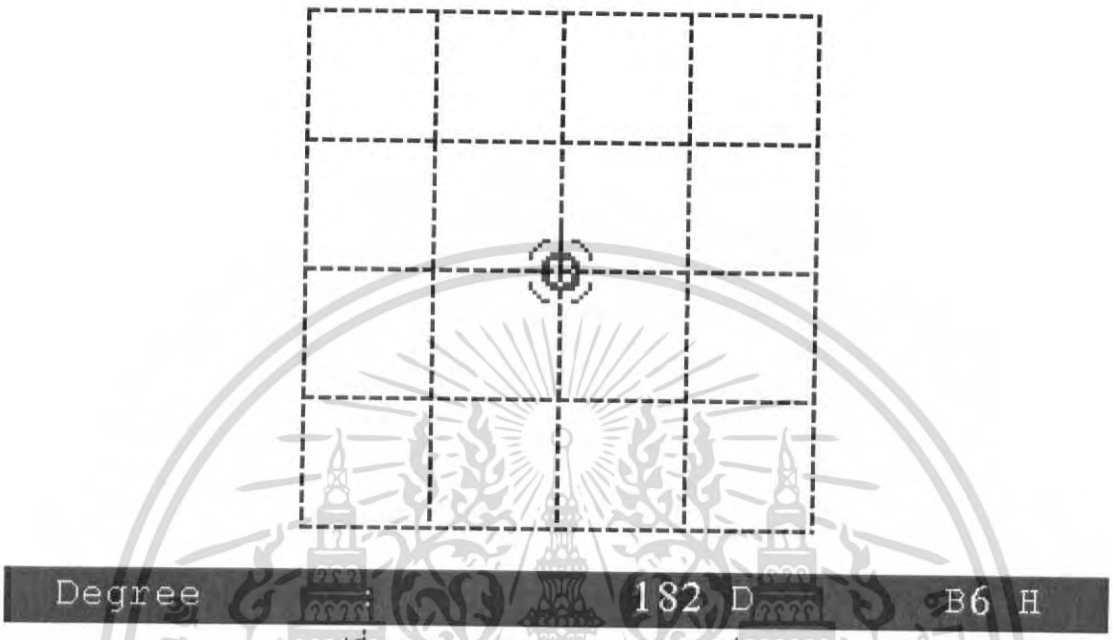
4.2 การทดลองสร้างแผนที่ด้วยโมดูลเข็มทิศ พัลส์ของมอเตอร์ และ โมดูลอัลตร้าโซนิค

ในการทดลองนี้เราจะนำโมดูลเข็มทิศ พัลส์ของมอเตอร์มาสร้างแผนที่โดยโมดูลเข็มทิศ จะเป็นตัวที่ใช้ในการกำหนดทิศทางของหุ่นยนต์ และจะใช้พัลส์ของมอเตอร์ในการวัดระยะทาง ส่วนโมดูลอัลตร้าโซนิคจะใช้ในการเก็บรายละเอียดรอบข้าง



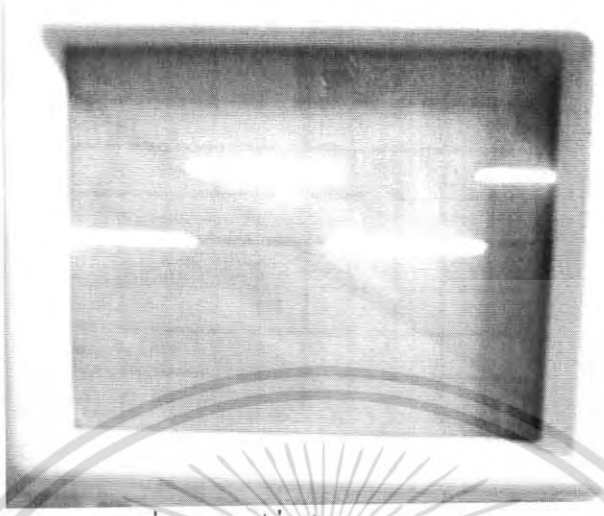
รูปที่ 4.4 โมดูลเข็มทิศ

โดยจะแสดงการทำงานดังรูปที่ 4.5 คือ วงกลมสีแดงจะแสดงตัวหุ่นยนต์และขีดน้ำเงิน คือ ตัวแสดงทิศทางของหุ่นยนต์ขณะนั้น



รูปที่ 4.5 แสดงการทำงานของโมดูลเข็มทิศ

รูปที่ 4.6 โมดูลอัลตร้าโซนิค

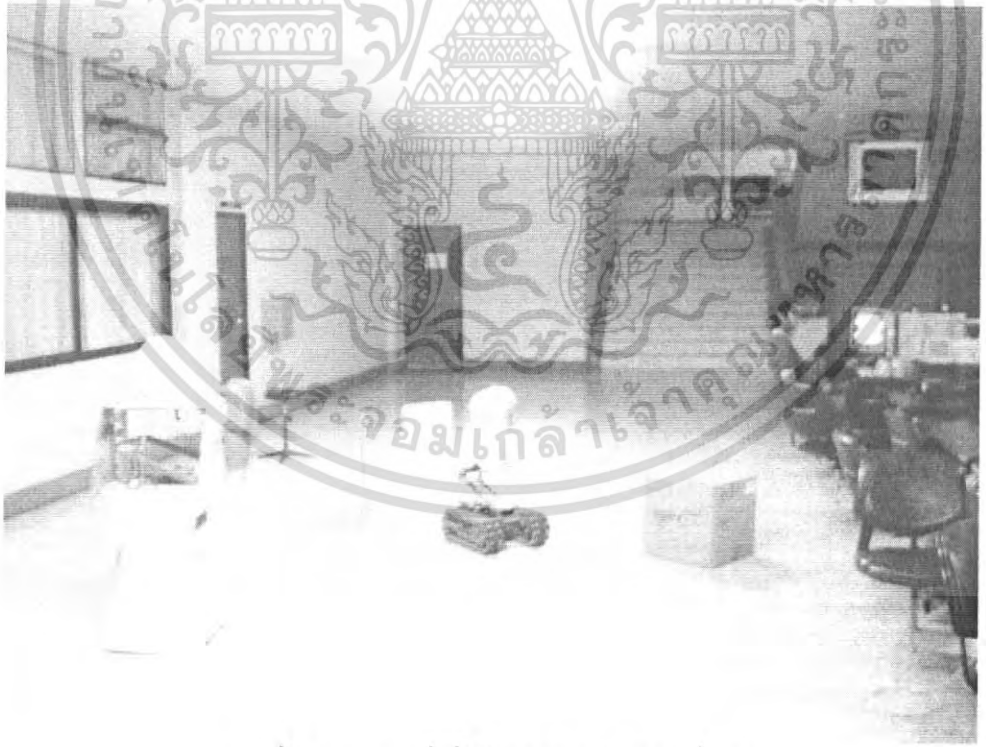


รูปที่ 4.7 พัลส์ที่ใช้ในการวัดระยะทาง

จากหัวข้อที่ 4.1 ที่เราใช้พัลส์ในการวัดความเร็วโดยดูจากความกว้างของพัลส์ และเราจะได้ระยะทาง คือ จำนวนพัลส์ที่เกิดขึ้น

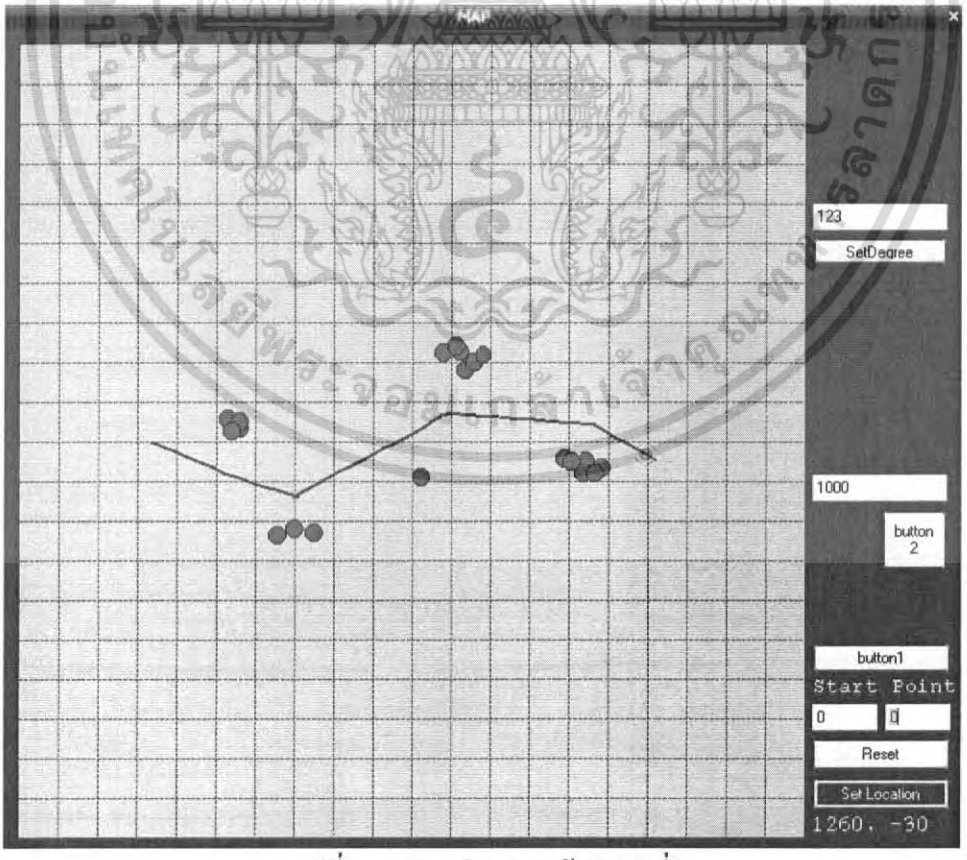
รูปที่ 4.8 สถานที่ ที่ใช้ทดลองสร้างแผนที่ (1)

รูปที่ 4.9 สถานที่ ที่ใช้ทดลองสร้างแผนที่ (2)



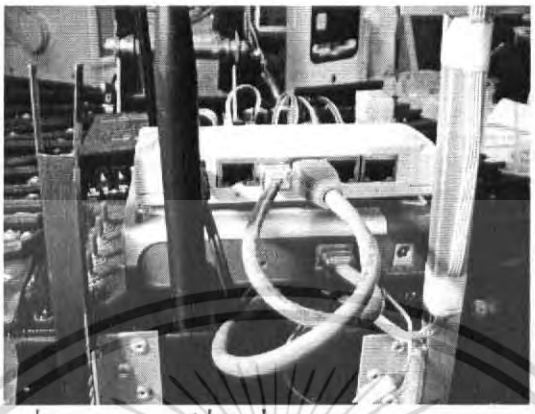
รูปที่ 4.10 สถานที่ ที่ใช้ทดลองสร้างแผนที่ (3)

รูปที่ 4.11 สถานที่ ที่ใช้ทดลองสร้างแผนที่ (4)



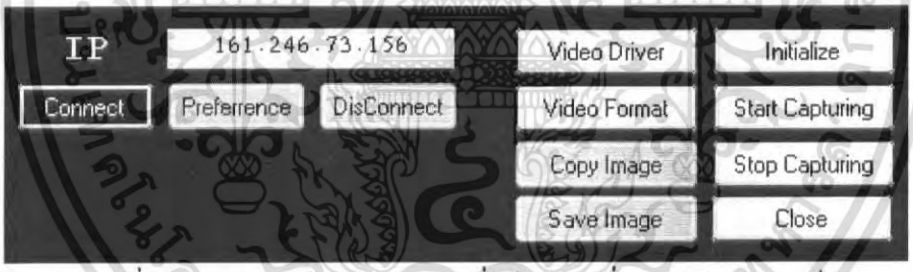
รูปที่ 4.12 ตัวอย่างการสร้างแผนที่

4.3 การทดลองเชื่อมต่อกับระบบ Wireless LAN



รูปที่ 4.13 อุปกรณ์ที่ใช้เชื่อมต่อกับ Wireless LAN

รูปที่ 4.7 แสดงอุปกรณ์ที่ใช้ในการติอต่อกับระบบ Wireless LAN คือ Access Point และ Hub ซึ่ง Hub ใช้ในการเชื่อมต่อ Access Point เข้ากับ Ethernet IO Board และสามารถรองรับอุปกรณ์อื่นๆ เพิ่มเติมได้อีก



รูปที่ 4.14 แสดงหน้าจอ Interface ที่ใช้ในการเชื่อมต่อกับ หุ่นยนต์

รูปที่ 4.8 แสดงหน้าจอ Interface การเชื่อมต่อคือ มีช่องสำหรับป้อน IP Address ซึ่งหากเชื่อมต่อได้ หรือมีปัญหาในการเชื่อมต่อก็จะมีข้อความแสดงสถานะการเชื่อมต่อแสดงขึ้นมา

บทที่ 5
สรุปผลการดำเนินงาน

5.1สรุปผลการดำเนินงาน

โครงงานนี้จัดทำขึ้นเพื่อศึกษาแนวทางในการพัฒนาหุ่นยนต์ให้มีประสิทธิภาพ โดยมีหัวข้อที่ได้ศึกษาและพัฒนาหลักๆอยู่ 3 หัวข้อ

1. การควบคุมหุ่นยนต์ผ่านระบบ Wireless LAN ซึ่งสามารถเขียนโปรแกรมและควบคุมหุ่นยนต์ผ่านระบบ Wireless LAN ได้ในระยะทางที่น่าพอใจ
2. การควบคุมความเร็วหุ่นยนต์โดยการนำทฤษฎี PID Motion Control มาควบคุมความเร็วของหุ่นยนต์ ซึ่งจากการทดลอง สามารถควบคุมความเร็วได้ตั้งแต่ 0-1500 รอบต่อวินาที
3. การสร้างแผนที่ จากการทดลองสามารถวัดทิศทาง ระยะทางที่เคลื่อนที่ไปได้ และสามารถระบุวัตถุ หรือสิ่งกีดขวางรอบข้างได้

5.2ปัญหาที่เกิดขึ้นกับโครงงาน

1. หากมีความถี่เดียวแต่ไม่มีการแบ่งช่องสัญญาณที่ดี ทำให้สัญญาณชนกัน อาจทำให้ Access Point เกิดการล่มได้

2. การสร้างแผนที่มีความคลาดเคลื่อนไปจากความจริงเนื่องจากองศาที่ได้จากโมดูลเข็มทิศมีความคลาดเคลื่อนเนื่องมาจากการรบกวนจากสนามแม่เหล็กรอบ ๆ หุ่นยนต์

3. เมื่อหุ่นยนต์เกิดแรงต้านการเคลื่อนที่ ตัวควบคุม PID ทำการจ่ายกระแสเพิ่มขึ้นจนถึงจุดสูงสุดแต่ไม่สามารถควบคุมให้ถึง Target Speed ได้จะเกิดการสะสมของ I-Term ขึ้นไปเรื่อย ๆ ทำให้เมื่อปรับ Target Speed ลงมาจะไม่สามารถควบคุมความเร็วได้

5.3 แนวทางในการพัฒนาต่อ

1. สามารถพัฒนาต่อในส่วนของ Wireless LAN ให้มีประสิทธิภาพในการส่งสัญญาณให้ดีขึ้นสามารถ และแก้ไขปัญหาเรื่องสัญญาณรบกวน

2. สามารถนำการควบคุมฮาร์ดแวร์ผ่านระบบ Wireless LAN ไปประยุกต์ใช้กับอุปกรณ์อื่นๆได้

3. โดยการพัฒนาสามารถพัฒนาได้ทั้งการเลือกใช้งานอุปกรณ์ที่ดีขึ้น การปรับปรุงระบบการควบคุมมอเตอร์ และเลือกใช้กล้องที่มีคุณภาพสูงขึ้น

บรรณานุกรม

[1] ธีรวัฒน์ ประกอบผล, **"การประยุกต์ใช้งานไมโครคอนโทรลเลอร์"**, สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2543

[2] ฉัททวุฒิ พืชผล, พิชิต สันติกุลานนท์, "**คู่มือเรียน Visual Basic 6**" , บริษัท โปรวิชั่น จำกัด, 2547

[3] สุธี พงศาสกุลชัย, หทัยชนก งามอินทร์, "**คัมภีร์ Visual C# 2005**" , บริษัท เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด, 2542

[4] โอภาส ศิริครรชิตถาวร, "**เรียนรู้และพัฒนาไมโครคอนโทรเลอร์ ARM7 LPC2148 ด้วยภาษาซี**", โรงพิมพ์วชิรินทร์สาสน์ รัชดา, 2549