การพัฒนาหุ่นยนต์สำรวจแบบ ROCKER-BOGIE

THE DEVELOPMENT OF ALL TERRAIN ROCKER-BOGIE ROBOT

นายเกรียงไกร แจ่มขุนเทียน
นายคณิต ธรรมกิตติคุณ
นายสุรพร พวงพันธ์บุตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

THE DEVELOPMENT OF ALL TERRAIN ROCKER-BOGIE ROBOT

KREINGKRAI JAMKHUNTEIN

KANIT THAMKITTIKUN

SURAPORN PUANGPANBUTE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2005

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

.....

หัวข้อปริญญานิพนธ์	การพัฒนาหุ่นยนต์สำรวจแบบ ROCKER-BOGIE THE DEVELOPMENT OF ALL TERRAIN ROCKER-BOGIE ROBOT
นักศึกษาผู้จัดทำ	นายเกรียงไกร แจ่มขุนเทียน **รหัสประจำตัว** 45010063 นายคณิต ธรรมกิตติคุณ **รหัสประจำตัว** 45010080 นายสุรพร พวงพันธ์บุตร **รหัสประจำตัว** 45010872
ปริญญา	วิศวกรรมศาสตรบัณฑิต
สาขาวิชา	วิศวกรรมการวัดคุม
ปีการศึกษา	2548

อาจารย์ผู้ควบคุมปริญญานิพนธ์	**ลายมือชื่อ**
ดร. พงษ์ชัย นิลาศ	

ภาควิชารับรองแล้ว

(รศ. ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์	การพัฒนาหุ่นยนต์สำรวจแบบ ROCKER-BOGIE THE DEVELOPMENT OF ALL TERRAIN ROCKER-BOGIE ROBOT			
นักศึกษาผู้จัดทำ	นายเกรียงไกร	แจ่มขุนเทียน	รหัสประจำตัว	45010063
	นายคณิต	ธรรมกิตติคุณ	รหัสประจำตัว	45010080
	นายสุรพร	พวงพันธ์บุตร	รหัสประจำตัว	45010872
อาจารย์ที่ปรึกษา	ดร. พงษ์ชัย	นิลาศ		
ปีการศึกษา	2548			

บทคัดย่อ

ปริญญานิพนธ์นี้ได้นำเสนอการออกแบบและทดลองสร้างหุ่นยนต์ 6 ล้อ ที่มีคุณลักษณะสามารถเคลื่อนที่ข้ามสิ่งกีดขวางที่มีขนาดใหญ่กว่าขนาดของเส้นผ่านศูนย์กลางของล้อ โดยใช้ระบบกลไกช่วงล่างแบบ Rocker-Bogie ซึ่งสามารถทำให้ล้อทั้งหมดทุกล้อสัมผัสอยู่กับผิวของวัตถุที่หุ่นยนต์กำลังเคลื่อนที่ผ่านตลอดเวลา ถึงแม้ว่าจะมีการเคลื่อนที่ผ่านในพื้นผิวที่มีความต่างระดับกันมาก ซึ่งทำให้หุ่นยนต์สามารถส่งถ่ายพลังงานขับไปยังล้อ เพื่อใช้ในการปีนป่ายได้เป็นอย่างดี โดยคณะผู้วิจัยนี้มีความประสงค์ที่จะทำให้หุ่นยนต์สำรวจตัวนี้สามารถเคลื่อนที่ผ่านสิ่งกีดขวางที่วางอยู่ในลักษณะตั้งฉากกับพื้นได้

Thesis Title	The Development of all Terrain Rocker-Bogie Robot
Author	Mr. Kreingkrai Jamkhuntein
	Mr. Kanit Thumkittikun
	Mr. Suraporn Puangpanbute
Thesis Advisor	Dr. Phongchai Nilas
Year	2005

ABSTRACT

This Thesis presents the design and development of a six-wheel Rocker-Bogie robot. The unique of this robot is able to claim over obstacles that have the height larger than the robot wheel's diameter. The Rocker-Bogie system automatically adjusts the wheels to touch on the surface of the obstacles at all time , which allows the robot to apply its claiming force more efficiently. The main purpose of this project is to study the Rocker-Bogie system and construct a robot that can overcome the perpendicular obstacle.

กิตติกรรมประกาศ

ปริญญานิพนธ์เรื่อง การพัฒนาหุ่นยนต์สำรวจแบบ Rocker-Bogie สำเร็จลุล่วงไปได้ด้วยดี เพราะได้รับความเมตตาจาก ดร.พงษ์ชัย นิลาศ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน และขอขอบพระคุณ รศ.ประภาษ อุดคกิมาพันธุ์ ผศ.ไสว พงศ์สวัสดิ์ และดร.อำนาจ คณะรัฐ ที่คอยให้คำแนะนำอันมีประโยชน์อย่างยิ่ง รวมไปถึง ดร.รัชทิน จันทร์เจริญ ภาควิศวกรรมเครื่องกล จุฬาลงกรณ์มหาวิทยาลัย สำหรับคำแนะนำและแนวคิดดีๆ ที่ได้กรุณาแนะนำมา

ขอขอบพระคุณภาควิชาวิศวกรรมการวัดคุม และวิศวกรรมเครื่องกล สำหรับอุปกรณ์และเครื่องมืออำนวยความสะดวกในการทำโครงงาน

และที่ลืมเสียไม่ได้คือ ขอกราบขอบพระคุณคุณพ่อคุณแม่อันเป็นที่รักยิ่งของเรา ที่คอยให้ความช่วยเหลือ ทั้งด้านกำลังทรัพย์และกำลังใจในการทำงานครั้งนี้

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

ในปัจจุบันนี้หุ่นยนต์เข้ามามีบทบาทมากขึ้นในภาคอุตสาหกรรม หุ่นยนต์ถูกใช้ทั้งในการสำรวจ หรือลำเลียงชิ้นส่วนอุปกรณ์ต่างๆภายในโรงงาน แต่ในระยะหลังได้มีการพัฒนาหุ่นยนต์เคลื่อนที่ได้ (Mobile Robots) เพื่อใช้งานในพื้นที่เสี่ยงอันตรายสำหรับมนุษย์ ซึ่งหุ่นยนต์ประเภทนี้โดยมากจะถูกสร้างให้มีความสามารถในการเคลื่อนที่บนพื้นผิวที่มีความขรุขระมากได้ดี โดยกลไกที่ช่วยสนับสนุนในการเคลื่อนที่บนพื้นผิวเช่นนี้มีอยู่หลายประเภท มีทั้งที่เป็นแบบใช้ล้อ เป็นล้อที่มีสายพาน หรือแบบที่ใช้ขาในการก้าวเดินคล้ายกับแมลง โดยที่กลไกแต่ละแบบแต่ละชนิดก็จะมีข้อดีและข้อเสียเปรียบที่ต่างกันออกไป

ในการศึกษาครั้งนี้ผู้ทำโครงงานได้เลือกใช้การขับเคลื่อนแบบใช้ล้อเพราะมีระบบกลไกแบบง่าย โดยศึกษาถึงระบบกลไกที่เหมาะสมในการ เคลื่อนที่บนพื้นผิวที่ขรุขระได้ดี ซึ่งจะเลือกใช้ระบบกลไกช่วงล่างที่เรียกว่า Rocker-Bogie ซึ่งระบบช่วงล่างแบบ Rocker-Bogie จะไม่อาศัยสปริงหรือชิ้นส่วนที่มีความยืดหยุ่น ส่งผลให้มีแรงในการขับเคลื่อนดีขึ้น และระบบช่วงล่างดังกล่าวนี้ยังช่วยให้ล้อทุกล้อสัมผัสอยู่กับพื้นตลอดเวลา แม้ว่าจะเคลื่อนที่บนพื้นผิวที่ขรุขระ โดยทำให้คุณลักษณะภายในมีการกระจายน้ำหนักลงสู่ล้อทั้งหกอย่างสม่ำเสมอ ทำให้มีแรงในการขับเคลื่อนดีขึ้น จากคุณสมบัติดังกล่าวระบบช่วงล่างแบบ Rocker-Bogie จึงสามารถช่วยให้หุ่นยนต์เคลื่อนที่ข้ามสิ่งกีดขวางที่มีขนาดใหญ่มากเมื่อเทียบกับขนาดเส้นผ่านศูนย์กลางของล้อได้

1.1 วัตถุประสงค์ในการทำปริญญานิพนธ์

ปริญญานิพนธ์นี้เป็นการศึกษาและออกแบบสร้างหุ่นยนต์ขับเคลื่อน 6 ล้อที่มีกลไกช่วงล่างแบบ Rocker-Bogie เพื่อเป็นการศึกษาถึงความสามารถของหุ่นยนต์ที่มีระบบช่วงล่างประเภทนี้ว่ามีความสามารถในการปีนไต่สิ่งกีดขวางที่มีความยากต่อการข้ามมากๆ เช่น วัตถุที่มีความสูงมากกว่าความสูงของล้อของหุ่นยนต์หรือวัตถุที่มีลักษณะตั้งฉากกับพื้นผิว

1.2 ขั้นตอนการศึกษา

การทำปริญญานิพนธ์ฉบับนี้มีขั้นตอนการศึกษาเริ่มจากการศึกษาถึง ทฤษฎี โครงสร้างของระบบช่วงล่างแบบ Rocker-Bogie ซึ่งมีสมการทางคณิตศาสตร์รองรับจากนั้นจึงทำการออกแบบโดยใช้โปรแกรม Solid Work และทำการกัดชิ้นส่วนของหุ่นยนต์เพื่อนำมาประกอบเป็นตัวหุ่นยนต์จากนั้นก็ทำการออกแบบและสร้างระบบควบคุมทั้งระบบการเคลื่อนที่และระบบบังคับเลี้ยวโดย

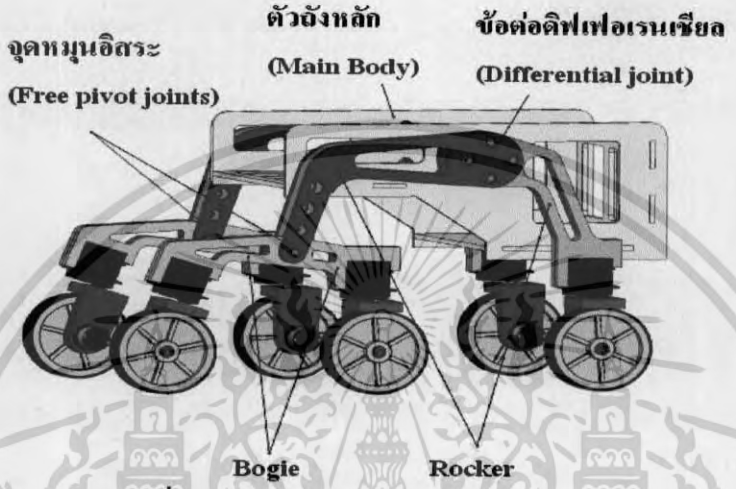
ระบบการเคลื่อนที่ใช้เป็นแบบการควบคุมแรงบิดของแต่ละล้อ (Torque Control) และมีการควบคุมแบบป้อนกลับ(Feedback Control) ส่วนระบบบังคับเลี้ยวใช้มอเตอร์ Servo บังคับเลี้ยว จากนั้นทำการศึกษาและเขียนโปรแกรมที่ใช้ควบคุมโดยใช้โปรแกรม Visual Basic ผู้ควบคุมสามารถควบคุมการเคลื่อนที่ของหุ่นยนต์ผ่านพอร์ตอนุกรมของคอมพิวเตอร์(RS 232)ได้

1.3 ประโยชน์ที่คาดว่าจะได้รับ

จากการทำปริญญานิพนธ์ทำให้สามารถสร้างหุ่นยนต์ที่มีระบบช่วงล่างแบบ Rocker-Bogie ซึ่งมีความสามารถในการข้ามสิ่งกีดขวางได้เป็นอย่างดี เราสามาถนำไปใช้เป็นหุ่นยนต์สำรวจหรือหุ่นยนต์ปฏิบัติการแทนมนุษย์ในที่ๆมีความเสี่ยงอันตรายหรือที่ที่มีสภาพภูมิประเทศที่ยากต่อการเข้าถึงโดยมนุษย์ และยังสามารถพัฒนาต่อให้มีประสิทธิภาพในหลายๆรูปแบบ ทั้งการควบคุมจากระยะไกลๆ การติดอุปกรณ์ต่างๆ เช่น กล้อง อุปกรณ์ตรวจจับชนิดต่างๆ เพื่อให้เป็นหุ่นยนต์เอนกประสงค์ต่อไป

บทที่ 2
ทฤษฎี และหลักการ

2.1 หลักการและทฤษฎีที่เกี่ยวข้องที่นำมาใช้งาน



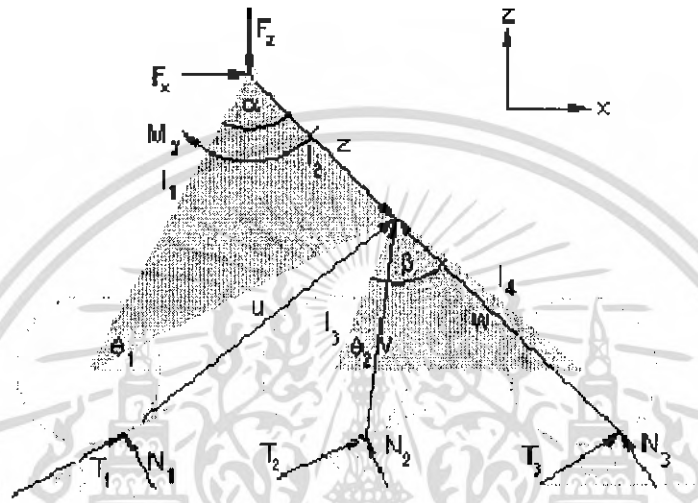
รูปที่ 2.1 ระบบช่วงล่างแบบ Rocker-Bogie

ระบบช่วงล่างแบบ Rocker-Bogie เป็นระบบช่วงล่างที่ทำงานได้ดีที่ความเร็วต่ำ และเป็นแบบ passive โดยล้อขับเคลื่อนทั้ง 6 ล้อจะเป็นอิสระต่อกัน ติดตั้งอยู่บนฐานล้อแบบหมุน (articulated frame) ซึ่งฐานล้อดังกล่าวประกอบขึ้นจากแขนกระดก (rocker arm) จำนวน 2 แขน ประกอบติดกับลำตัว แขนกระดกแต่ละอันจะติดตั้งล้อหลังไว้ที่ปลายด้านหนึ่ง ส่วนปลายอีกด้านหนึ่งจะต่อกับแขนกระดกชุดที่สอง หรือที่เรียกว่า "Bogie" ผ่านจุดหมุนอิสระ (free pivoting joint) โดยที่ปลายทั้งสองของ bogie ติดตั้งด้วยล้อขับเคลื่อน และแขนกระดก (rocker) ติดตั้งกับลำตัวโดยใช้กลไกดิฟเฟอเรนเชียล (differential joint) ซึ่งจะทำให้มุมก้มเงย (pitch angle) ของลำตัวเป็นค่าเฉลี่ยระหว่างมุมของแขนกระดกทั้งสองข้าง แสดงดังรูปที่ 2.1 ระบบช่วงล่างแบบ Rocker-Bogie จะทำให้ล้อทุกล้อสัมผัสพื้นอยู่ตลอดเวลา แม้ในขณะที่เคลื่อนที่บนพื้นผิวที่ขรุขระ นอกจากนี้ยังมีการกระจายน้ำหนักลงสู่ล้อทั้งหกอย่างสม่ำเสมอ ทำให้มีแรงขับเคลื่อนที่ดีขึ้น

เมื่อเปรียบเทียบระบบช่วงล่างของการขับเคลื่อนด้วยล้อแบบ 4 ล้อของรถยนต์ และแบบ Rocker-Bogie แล้ว แบบรถยนต์จะใช้ตัวหน่วงหรือที่เรียกว่า แหนบรถยนต์ในการรับน้ำหนักซึ่งมีความยืดหยุ่น ทำให้แรงกดที่พื้นมีค่ามากพอ ซึ่งแรงที่กดนี้จะขึ้นอยู่กับสภาพความขรุขระของพื้นสัมผัส แต่ระบบช่วงล่างแบบ Rocker-Bogie จะใช้กลไกที่มีความเป็นอิสระ ทำให้ล้อสามารถสัมผัสพื้นได้ทุกสภาพความขรุขระของพื้น ทำให้สมรรถนะในการปีนไต่สิ่งกีดขวางดีขึ้น แรงที่กดพื้นของล้อจะไม่ขึ้นอยู่กับความขรุขระของพื้น แต่จะขึ้นอยู่กับมุมที่ล้อสัมผัสพื้น

ด้วยเหตุผลที่กล่าวมา จะเห็นได้ว่าระบบกลไกช่วงล่างแบบ Rocker-Bogie นั้นมีลักษณะโครงสร้างที่ไม่ซับซ้อน และมีศักยภาพในการเคลื่อนที่บนพื้นผิวที่มีความขรุขระมากได้ดี ทำให้เหมาะสมกับการใช้เป็นระบบช่วงล่างของยานพาหนะที่ต้องเคลื่อนที่บนสภาพแวดล้อมที่เราไม่สามารถควบคุมได้

2.1.1 รูปแบบและสมการทางคณิตศาสตร์ของระบบช่วงล่างแบบ Rocker-Bogie



รูปที่ 2.2 ผังวัตถุอิสระของตัวรถ

F_x : แรงในแนวแกน X

F_z : แรงในแนวแกน Z

Z : ระยะจากจุดหมุนอิสระถึงข้อต่อดิฟเฟอเรนเชียล

M_y: โมเมนต์รอบจุดดิฟเฟอเรนเชียล

I_1 : ระยะจากจุดศูนย์กลางล้อที่ 1ถึงข้อต่อดิฟเฟอเรนเชียล

I_2 : ระยะจากจุดศูนย์กลางล้อที่ 2ถึงข้อต่อดิฟเฟอเรนเชียล

I_3 : ระยะจากจุดศูนย์กลางล้อที่ 3ถึงข้อต่อดิฟเฟอเรนเชียล

U : ระยะจากจุดสัมผัสล้อที่ 1ถึงจุดหมุนอิสระ

V : ระยะจากจุดสัมผัสล้อที่ 2ถึงจุดหมุนอิสระ

W : ระยะจากจุดสัมผัสล้อที่ 3ถึงจุดหมุนอิสระ

α_1 : มุมระหว่างพื้นสัมผัสกับล้อที่1

α_2 : มุมระหว่างพื้นสัมผัสกับล้อที่2

α_3 : มุมระหว่างพื้นสัมผัสกับล้อที่3

T_1 : แรงเสียดทานที่เกิดที่ล้อที่1

T_2 : แรงเสียดทานที่เกิดที่ล้อที่2

T_3 : แรงเสียดทานที่เกิดที่ล้อที่3

N_1 : แรงกดที่พื้นกระทำกับล้อที่1

N_2 : แรงกดที่พื้นกระทำกับล้อที่2

N_3 : แรงกดที่พื้นกระทำกับล้อที่3

จากรูปที่ 2.2 เขียนสมการสมดุลของผังวัตถุอิสระได้ดังนี้

$$\sum F_x ; \quad T_1c_1 + T_2c_2 + T_3c_3 - N_1s_1 - N_2s_2 - N_3s_3 + F_x = ma \tag{2-1}$$

$$\sum F_y ; \quad T_1c_1 + T_2c_2 + T_3c_3 + N_1s_1 + N_2s_2 + N_3s_3 + F_z = 0 \tag{2-2}$$

$$\sum M_{yr} ; \quad T_1c_1u_z - T_1s_1u_x - N_1c_1u_z - N_1s_1u_z - F_xz_z + F_zz_x - M_y = 0 \tag{2-3}$$

$$\sum M_{yb} ; \quad T_2c_2v_z - T_2s_2v_x - N_2c_2v_x - N_2s_2v_z + T_3c_3w_z + T_3s_3w_x + N_3s_3w_x + N_3c_3w_x - N_3s_3w_z = 0 \tag{2-4}$$

โดยที่ $c_i = \cos(\alpha_i)$และ $s_i = \sin(\alpha_i)$;i =1,2,3 กำหนด $T \leq \mu N$ และ $N > 0$

จาก 4 สมการหลักข้างต้นนำมาแทนค่าและทำการจัดเรียงใหม่

จากสมการ (2-1) ได้

$$\mu N_1\cos(\alpha_1)+\mu N_2\cos(\alpha_2)+\mu N_3+\cos(\alpha_3)-N_1\sin(\alpha_1)-N_2\sin(\alpha_2)-N_3\sin(\alpha_3)+F_x = ma \tag{2-5}$$

$$(\mu\cos(\alpha_1)-\sin(\alpha_1))N_1+(\mu\cos(\alpha_2)-\sin(\alpha_2))N_2+(\mu\cos(\alpha_3)-\sin(\alpha_3))N_3+F_x = ma \tag{2-5a}$$

จากสมการ (2-2) ได้

$$\mu N_1\sin(\alpha_1)+\mu N_2\sin(\alpha_2)+\mu N_3\sin(\alpha_3)+ N_1\cos(\alpha_1)+N_2\cos(\alpha_2)+ N_3\cos(\alpha_3)-F_z = 0 \tag{2-6}$$

$$(\mu\sin(\alpha_1)+\cos(\alpha_1))N_1+(\mu\sin(\alpha_2)+\cos(\alpha_2))N_2+ (\mu\sin(\alpha_3) + \cos(\alpha_3))\, N_3-F_z = 0 \tag{2-6a}$$

จากสมการ (2-3) ได้

$$\mu N_1\cos(\alpha_1)u_z -\mu N_1\sin(\alpha_1)u_x-\cos(\alpha_1)N_1u_x -\sin(\alpha_1)\, N_1u_z- F_xz_z + F_zz_x- M_y = 0 \tag{2-7}$$

$$(\mu\cos(\alpha_1)u_z- \mu\sin(\alpha_1)u_x- \cos(\alpha_1)u_x- \sin(\alpha_1)u_z)N_1- F_xz_z+ F_zz_x- M_y = 0 \tag{2-7a}$$

จากสมการ (2-4) ได้

$$\mu N_2\cos(\alpha_2)v_z-\mu N_2\sin(\alpha_2)v_x-\cos(\alpha_2)N_2v_x-\sin(\alpha_1)N_2v_z+\cos(\alpha_3)\mu N_3w_z+\mu N_3\sin(\alpha_3)w_z$$
$$+N_3\cos(\alpha_3)w_x-N_3\sin(\alpha_3)w_z = 0 \tag{2-8}$$

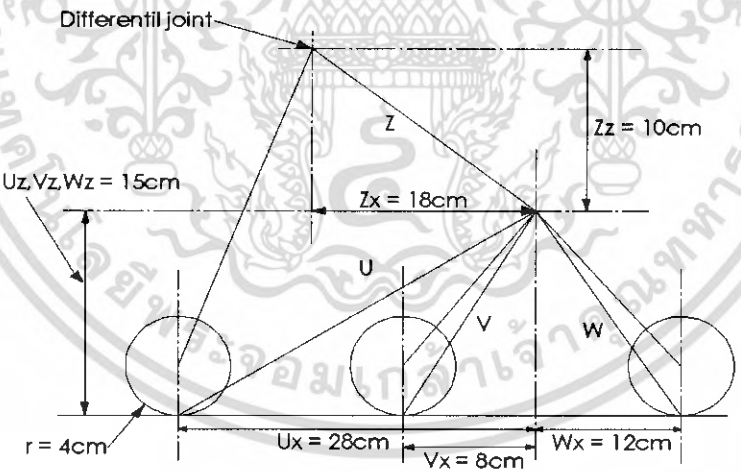
$$(\mu\cos(\alpha_2)v_z- \mu\sin(\alpha_2)v_x- \cos(\alpha_2)v_x-\sin(\alpha_1)v_z)N_2+(\mu(\cos(\alpha_3)w_z+\mu\sin(\alpha_3)w_z+\cos(\alpha_3)w$$
$$- \sin(\alpha_3)w_z)\, N_3 = 0 \tag{2-8a}$$

จากสมการที่ (2.51) , (2.61) , (2.71)และ(2.81) นำมาเขียนเป็นแบบจำลองทางคณิตศาสตร์ของระบบนี้ ได้เป็น

$$\begin{bmatrix} (\mu\cos\alpha_1 - \sin\alpha_1) & (\mu\cos\alpha_2 - \sin\alpha_2) \\ (\mu\sin\alpha_1 - \cos\alpha_1) & (\mu\sin\alpha_2 - \cos\alpha_2) \\ (\mu\cos\alpha_1 u_z - \mu\sin\alpha_1 u_x & 0 \\ -\cos\alpha_1\mu_x - \sin\alpha_1 u_z) & \\ 0 & (\mu\cos\alpha_2 v_z - \mu\sin\alpha_2 v_x \\ & -\cos\alpha_2 v_x - \sin\alpha_2 v_z) \end{bmatrix}$$

$$\begin{bmatrix} (\mu\cos\alpha_3 - \sin\alpha_3) & 1 & 0 & 0 \\ (\mu\sin\alpha_3 - \cos\alpha_3) & 0 & -1 & 0 \\ 0 & -Z_z & Z_x & -1 \\ (\mu\cos\alpha_3 w_z + \mu\sin\alpha_3 w_z & 0 & 0 & 0 \\ +\cos\alpha_3 w_x - \sin\alpha_3 w_z) & & & \end{bmatrix} \cdot \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ F_x \\ F_z \\ M_y \end{bmatrix} = \begin{bmatrix} ma \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2\text{-}9)$$

จากโมเดลทางคณิตศาสตร์นี้ทำให้หาค่าพารามิเตอร์ต่างๆที่เหมาะสมได้ โดยได้ทำการออกแบบโครงสร้างออกมา ดังรูปที่ 2.3



รูปที่ 2.3 การกำหนดขนาดจริงให้กับผังวัดถุอิสระของตัวรถ

เมื่อคิดแรงในสภาวะปกติที่รถเคลื่อนที่ไปในแนวแกน X ที่มุม α_i=0, μ=0.5 แล้วแทนค่าพารามิเตอร์ต่างๆ ลงใน 4 สมการหลัก จะได้ค่า ดังนี้

$$0.5N_1 + 0.5N_2 + 0.5N_3 + F_x = 0 \quad (2\text{-}10)$$

$$N_1 + N_2 + N_3 = 73.575 \quad (2\text{-}11)$$

$$-0.205N_1 - 0.10F_x - M_y = -13.2435 \quad (2\text{-}12)$$

$$-0.005N_2 + 0.195\,N_3 = 0 \quad (2\text{-}13)$$

ซึ่งที่สภาวะพื้นเรียบกำหนดให้ $N_1 = N_2 = N_3$ จะได้

$$1.5N_1 + F_x = 0 \quad (2\text{-}14)$$

$$3N_1 = 73.575 \quad (2\text{-}15)$$

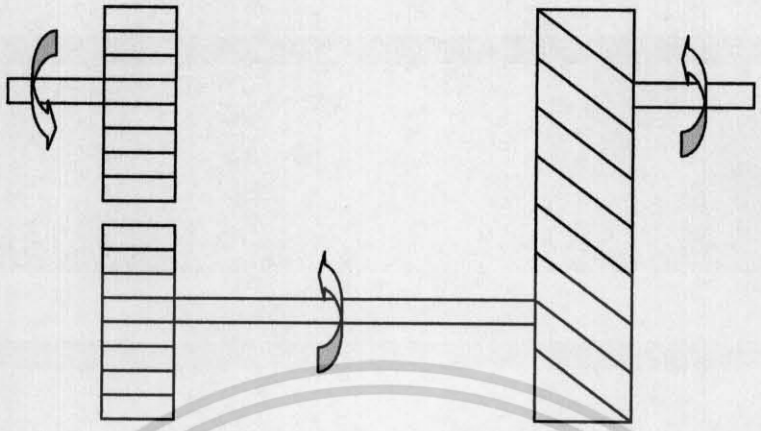
$$-0.205N_1 - 0.10F_x - M_y = -13.2435 \quad (2\text{-}12)$$

$$0.19\,N_1 = 0 \quad (2\text{-}16)$$

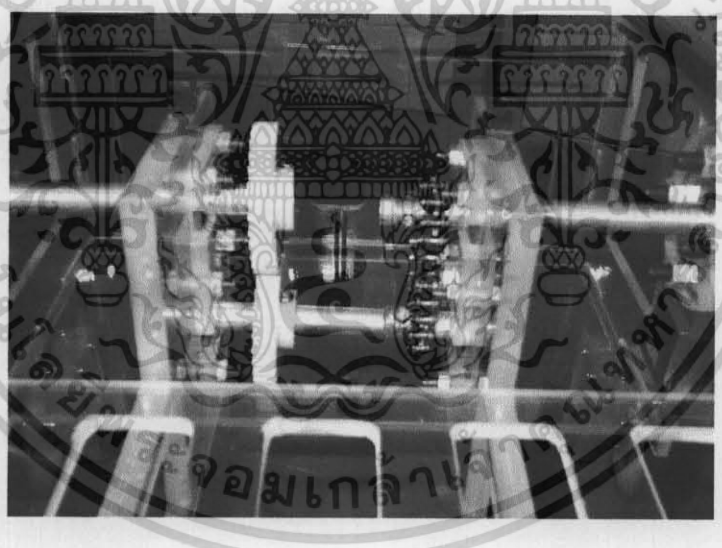
เมื่อทำการแก้สมการทั้ง 4 สมการจะได้ค่าออกมาดังนี้ $N_1 = N_2 = N_3 = 24.525$, $F_x = -36.78$, $M_y = 11.8946$ พบว่า เราต้องทำการวางน้ำหนัก ให้ได้ โมเมนต์รอบจุด differential joint เท่ากับ 11.8946 N.m

2.1.2 กลไกดิฟเฟอเรนเชียล (Differential joint)

กลไกดิฟเฟอเรนเชียล (Differential joint) เป็นกลไกที่สำคัญในหุ่นยนต์ที่ใช้ระบบช่วงล่างแบบ Rocker-Bogie ซึ่งจะช่วยทำให้ระบบช่วงล่างของ (Differential joint) ล้อมีการสัมผัสพื้นทุกล้อเมื่อมีการยกตัวขึ้นของล้อใดล้อหนึ่งหรือมากกว่าหนึ่งล้อ โดยล้อที่เหลือจะสัมผัสกับพื้นได้ทุกล้อ กลไกของหุ่นยนต์ตัวนี้ใช้ ชุดเฟืองขบ และชุดเฟืองสายพาน เป็นตัวควบคุมความสัมพันธ์ระหว่างขาทั้งสองข้างของตัวหุ่นยนต์ โดยเมื่อขาอีก ข้างของตัวหุ่นยนต์มีการขยับตัวขึ้นหรือลง ขาอีกข้างของหุ่นยนต์ก็จะเคลื่อนที่ในทิศทางตรงกันข้าม ดังรูปที่ 2.7



รูปที่ 2.4 ทิศทางการเคลื่อนที่ของชุดกลไกดิฟเฟอเรนเชียล (Differential joint)



รูปที่ 2.5 กลไกดิฟเฟอเรนเชียล

บทที่ 3
การออกแบบระบบควบคุม วงจรขับมอเตอร์ และโปรแกรมที่เกี่ยวข้อง

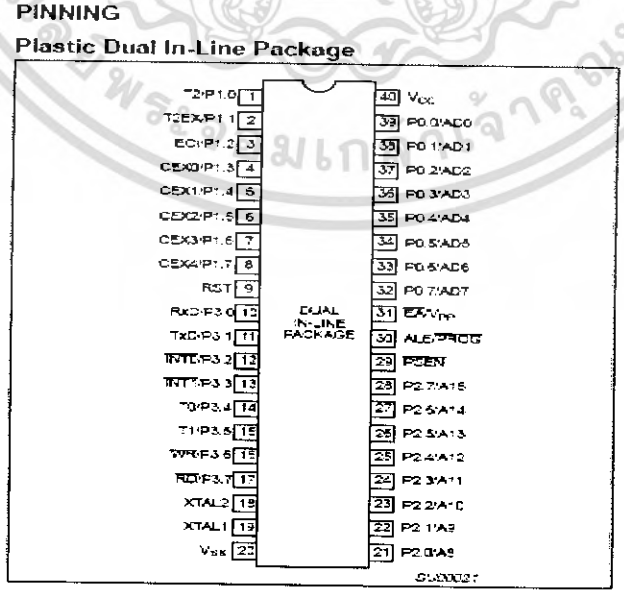
หลักการทำงานของหุ่นยนต์

หลักการทำงานของหุ่นยนต์นั้นจะเริ่มจากการรอรับคำสั่งต่างๆจากโปรแกรม Rocker-Bogie Robot Control ที่ถูกพัฒนาขึ้นด้วยโปรแกรม Visual Basic ซึ่งจะติดต่อกับหุ่นยนต์ผ่านสายสัญญาณอนุกรม RS-232 เพื่อส่งคำสั่งต่างๆไปประมวลผลที่ไมโครคอนโทรลเลอร์ซึ่งถูกติดอยู่ที่หุ่นยนต์แล้วส่งค่าสัญญาณสังเกตต่างๆกลับไปแสดงผลที่โปรแกรม Rocker-Bogie Robot Control โดยที่สามารถแบ่งออกเป็นหัวข้อต่างๆ ได้ดังนี้

3.1 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ถือเป็นหัวใจหลักในการควบคุมการทำงานของหุ่นยนต์ โดยจะเป็นตัวติดต่อกับโปรแกรม Rocker-Bogie Robot Control เพื่อรับคำสั่งในการทำงานผ่านสายสัญญาณอนุกรม RS-232 และควบคุมความเร็วและทิศทางของหุ่นยนต์ อีกทั้งยังส่งค่าสัญญาณสังเกตกลับไปแสดงผลที่โปรแกรม Rocker-Bogie Robot Control ด้วย โดยคณะผู้จัดทำจะใช้ไมโครคอนโทรลเลอร์ Phillips เบอร์ P89C51RD2BN

3.1.1 คุณสมบัติทางเทคนิคที่สำคัญของ P89C51RD2B



รูปที่ 3.1 แสดงขาต่างๆของไมโครคอนโทรลเลอร์ P89C51RD2BN

ไมโครครอลโทรลเลอร์ P89C51RD2HBP มีคุณสมบัติพื้นฐาน ดังนี้

- เป็นไมโครคอนโทรลเลอร์ 8 บิต ที่เข้ากันได้กับไมโครคอนโทรลเลอร์ MCS-51 พื้นฐาน

- หน่วยความจำโปรแกรมภายในตัวไมโครคอนโทลเลอร์เป็นแบบแฟลช ทำให้สามารถลบและเขียนใหม่ได้ถึง 10,000 ครั้ง จึงสามารถใช้งานในรูปแบบไมโครคอนโทรลเลอร์ชิปเดี่ยวไม่ต้องใช้หน่วยความจำภายนอก ส่งผลให้สามารถใช้งานพอร์ตอินพุตเอาต์พุตของไมโครคอนโทรลเลอร์ได้อย่างมีประสิทธิภาพ ขนาดของหน่วยความจำโปรแกรมสูงถึง 64 กิโลไบต์

- หน่วยความจำข้อมูลแรมภายในมีขนาด 1 กิโลไบต์

- สามารถโปรแกรมข้อมูลลงในหน่วยความจำโปรแกรมแบบในวงจรหรือในระบบ (ISP : In-System Programming)

- ความถี่สัญญาณนาฬิกาสูงสุด 33MHz ในกรณีทำงานด้วยสัญญาณนาฬิกาภายใน 12 ลูกต่อแมชชีนไซเกิล และ 20 MHz กรณีทำงานด้วยสัญญาณนาฬิกาภายใน 6 ลูกต่อแมชชีนไซเกิล

- P89C51RD2BN ได้รับการกำหนดให้ทำงานเบื้องต้นในโหมดสัญญาณนาฬิกา 12 ลูกต่อแมชชีนไซเกิล และ สามารถเลือกเปลี่ยนเป็นสัญญาณนาฬิกา 6 ลูกต่อแมชชีนไซเกิลได้ โดยสามารถเปลี่ยนกลับไปกลับมาได้ซึ่งแตกต่างจาก P89C51RD2HBP ที่ทำงานเบื้องต้นในโหมดสัญญาณนาฬิกา 6 ลูกต่อแมชชีนไซเกิล ซึ่งเมื่อเปลี่ยนเป็นสัญญาณนาฬิกา 12 ลูกต่อแมชชีนไซเกิลแล้วจะเปลี่ยนกลับมาไม่ได้อีก

- ขาพอร์ต 8 บิต 4 พอร์ต แบบกึ่งสองทิศทาง (Quasi-Bidirectional) เป็นได้ทั้งพอร์ตอินพุตและเอาต์พุต

- อุปกรณ์เพอริเฟอรัลภายในไมโครคอนโทลเลอร์สามารถทำงานด้วยความเร็ว 12 สัญญาณนาฬิกาต่อแมชชีนไซเกิล แม้ว่าซีพียูจะทำงานด้วยความเร็ว 6 สัญญาณนาฬิกาต่อแมชชีนไซเกิล เป็นคุณสมบัติพิเศษของไมโครครอลโทรลเลอร์ P89C51RD2BN

- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์

- ไทเมอร์/เคาน์เตอร์ขนาด 16 บิต 3 ตัว (ไทเมอร์ 0 , 1 และ 2)

- มีรีจิสเตอร์ตัวชี้ตำแหน่งข้อมูลหรือ DPTR 2 ตัว

- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 7 ประเภท

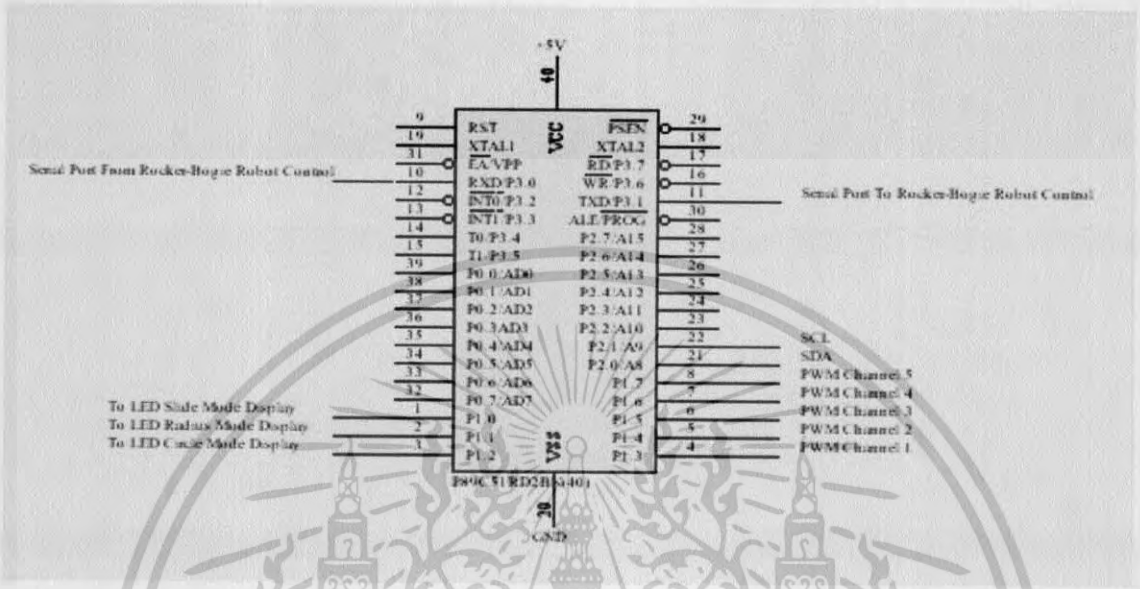
- กำหนดนัยสำคัญของการตอบสนองอินเตอร์รัปต์ได้ 4 ระดับ

- สามารถติดต่อหน่วยความจำภายนอกได้สูงสุด 64 กิโลไบต์

- มีโมดูลวงจรนับโปรแกรมได้ (PCA : Programmable Counter Array) 5 ช่องทางอิสระต่อกันซึ่งจะบรรจุความสามารถพิเศษอยู่ 5 อย่าง คือ วงจรตรวจจับสัญญาณ (PCA Capture Mode) , วงจรเปรียบเทียบสัญญาณ (PCA Compare Mode) , วงจรสัญญาณเอาต์พุตความเร็วสูง (PCA Hi-Speed Output Mode) , วงจรมอดูเลชั่นตามความกว้างพัลส์ (PCA Pulse-Width Modulation) และวงจรวอตช์ด็อกไทเมอร์ (PCA Watchdog Timer Mode)

3.1.2 การจัดการขาของไมโครคอนโทรลเลอร์

ในการสร้างหุ่นยนต์ Rocker-Bogie นี้ ทางคณะผู้จัดทำได้มีการจัดการขาอินพุตเอาต์พุตต่างๆของไมโครคอนโทรลเลอร์ ดังรูปที่ 3.2



รูปที่ 3.2 ขาของไมโครคอนโทรลเลอร์

- RxD และ TxD ใช้ในการติดต่อสื่อสารกับโปรแกรม Rocker-Bogie Robot Control ผ่านทางสายสัญญาณอนุกรม RS-232

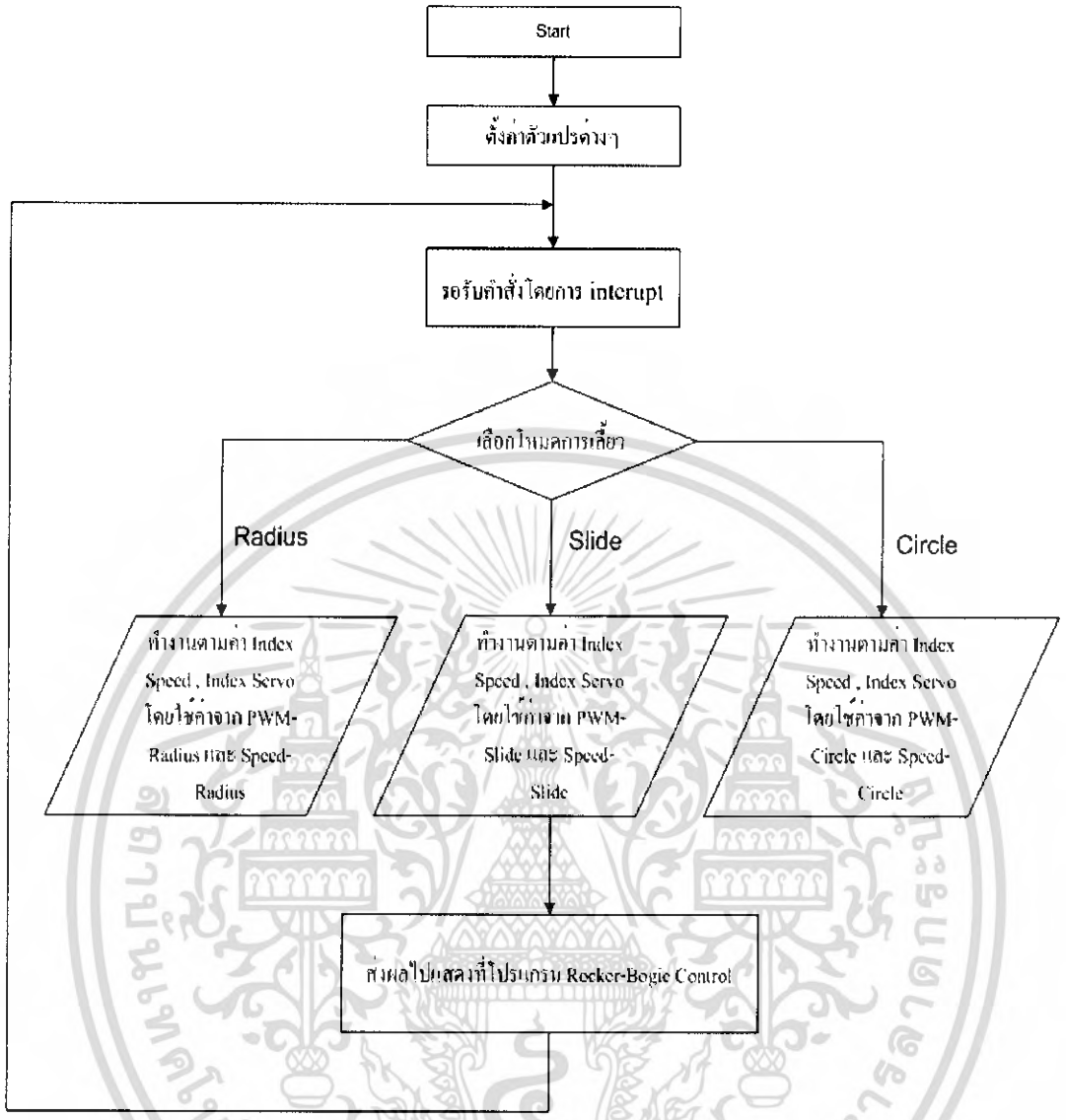
- P2.0 และ P2.1 ใช้เป็นขาสัญญาณ SDA และ SCL ตามลำดับ เพื่อติดต่อกับไอซี PCF8591 ซึ่งเป็น 4 Channel A/D และ 1 Channel D/A ซึ่งจะติดต่อกันด้วยโปรโตคอล I2C (I2C Protocol , I2C Bus) เพื่อแปลงสัญญาณดิจิตอลไปเป็นสัญญาณอะนาลอกซึ่งใช้ในการควบคุมความเร็วของมอเตอร์กระแสตรงทั้ง 6 ตัว

- P1.4 ถึง P1.7 (CEX0-CEX4) จะใช้โมดูลวงจรนับโปรแกรมได้ (PCA : Programmable Counter Array) เพื่อสร้างสัญญาณตามความกว้างพัลส์ (Pulse-Width Modulation) 5 ช่องทางเพื่อใช้ใควบคุมการทำงานของเซอร์โวมอเตอร์ทั้ง 6 ตัว

- P1.0 ถึง P1.2 ใช้ควบคุม LED ที่ใช้แสดงผลสถานะการเลี้ยวของหุ่นยนต์ โดยที่ P1.0 เป็นการเลี้ยวแบบกะทันหัน , P1.1 เป็นการเลี้ยวแบบรัศมี และ P1.2 เป็นการเลี้ยวแบบวงกลม

3.1.3 การเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์

คณะผู้จัดทำได้เลือกใช้ภาษาซีในการเขียนโปรแกรมควบคุมการทำงานของ ไมโครคอนโทรลเลอรและใช้คอมไพเลอร์ของ RIDE เพื่อความสะดวกในการเขียนโปรแกรม ซึ่งการเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์นั้นมี Flowchart ดังรูปที่ 3.3



รูปที่ 3.3 Flowchart ควบคุมการทำงานของไมโครคอนโทรลเลอร์

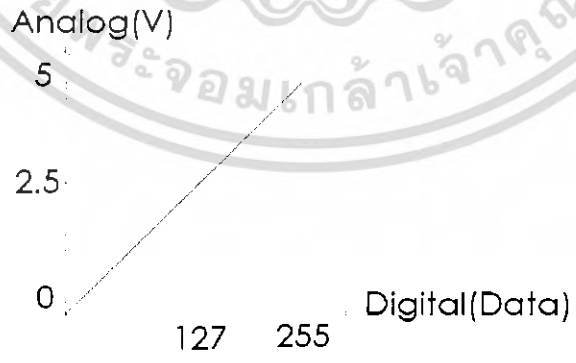
โดยที่จะเริ่มจากการตั้งค่าของตัวแปรต่างๆ (Initial) เช่น ตั้งค่ารูปแบบการเลี้ยวเป็นแบบกะทันหัน , ตั้งค่าความเร็วเป็นศูนย์ , ตั้งค่าการเลี้ยวเป็นศูนย์ จากนั้นจะรอรับคำสั่งจากโปรแกรม Rocker-Bogie Robot Control คือ คำสั่งเดินหน้า , คำสั่งถอยหลัง , คำสั่งเลี้ยวซ้าย , คำสั่งเลี้ยวขวา , คำสั่งหยุด และคำสั่งเลือกรูปแบบการเลี้ยว ผ่านทางสายสัญญาณอนุกรม RS-232 ที่ต่ออยู่กับ RxD และ TxD ด้วยการอินเตอร์รัปต์ของสัญญาณอนุกรม แล้วจึงตรวจสอบคำสั่งที่รับมาว่าคืออะไร แล้วทำการประมวลผลตามคำสั่งนั้นๆ แล้วก็กลับไปรอรับคำสั่งอีกครั้งซึ่งจะวนทำงานอย่างนี้ตลอดไป ซึ่งการเขียนโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์นั้นสามารถแบ่งออกเป็น 2 หัวข้อหลักๆ ได้ดังนี้

3.1.4 การเขียนโปรแกรมติดต่อกับ PCF8591 เพื่อควบคุมความเร็วของหุ่นยนต์

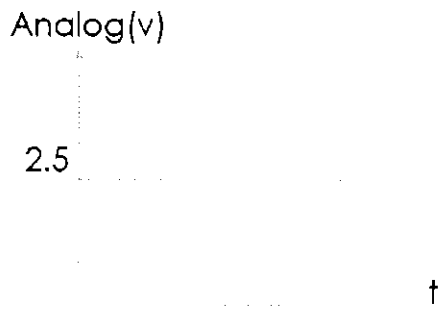
การติดต่อกับไอซี PCF8591 นั้นจะกระทำเพื่อแปลงค่าสัญญาณควบคุมความเร็วจากรูปแบบของสัญญาณดิจิตอลไปเป็นสัญญาณอะนาลอก เพื่อควบคุมการทำงานของวงจร Half-Bridge ที่ควบคุมความเร็วของมอเตอร์กระแสตรง ซึ่งก็คือความเร็วของหุ่นยนต์นั่นเอง เนื่องจากการควบคุมให้หุ่นยนต์เดินหน้าหรือถอยหลังได้นั้น วงจร Half-Bridge ต้องการแรงดันที่มีค่าเป็นบวกที่อยู่ในช่วง 0 ถึง +12 Vdc เพื่อทำให้หุ่นยนต์เดินหน้า และต้องการแรงดันไฟลบที่อยู่ในช่วง 0 ถึง -12 Vdc เพื่อทำให้หุ่นยนต์ถอยหลัง จะพบว่าวงจร Half-Bridge นั้นต้องการสัญญาณอะนาลอกที่มีแรงดันในช่วง -12 ถึง +12 Vdc แต่สัญญาณที่ออกจากไมโครคอนโทรลเลอร์นั้นเป็นสัญญาณดิจิตอล 0 ถึง +5 Vdc ซึ่งจะพบว่าไม่สามารถใช้ร่วมกันได้ โดยจะมีปัญหาที่เกิดขึ้นคือ รูปแบบของสัญญาณไม่ตรงกันและระดับแรงดันไม่ตรงกัน

ทางคณะผู้จัดทำจึงได้ใช้ไอซี PCF8591 เพื่อแปลงรูปแบบของสัญญาณจากสัญญาณดิจิตอลไปเป็นสัญญาณอะนาลอก เพื่อแก้ปัญหารูปแบบของสัญญาณที่ไม่ตรงกัน โดยที่ไอซี PCF8591 จะทำการแปลงค่าสัญญาณดิจิตอลขนาด 8 บิตไปเป็นสัญญาณอะนาลอก 0 ถึง +5 Vdc ตามรูปที่ 3.4 จากนั้นนำสัญญาณอะนาลอกที่ได้ไปลบกับค่าสัญญาณคงที่(Constant Voltage Signal) +2.5 Vdc เพื่อสร้างสัญญาณแรงดันบวกและลบ โดยจะมีค่าอยู่ในช่วง -2.5 ถึง +2.5 Vdc คือเมื่อสัญญาณอะนาลอก +5 Vdc มาลบกับสัญญาณคงที่ +2.5 Vdc จะได้แรงดัน +2.5 Vdc แต่เมื่อสัญญาณอะนาลอก 0 Vdc มาลบกับสัญญาณคงที่ +2.5 Vdc จะได้แรงดัน -2.5 Vdc

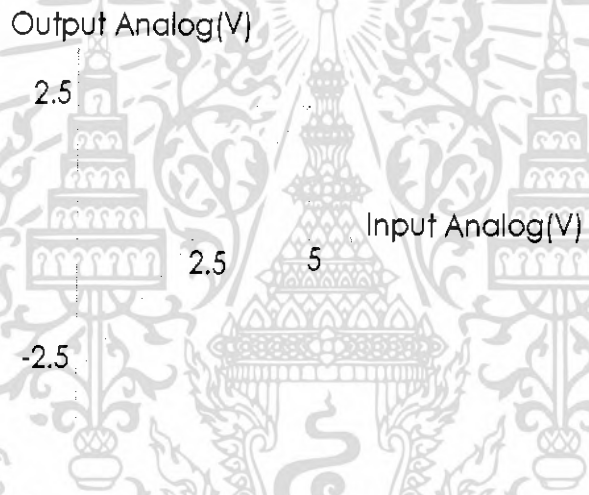
จากนั้นนำสัญญาณอะนาลอกในช่วง -2.5 ถึง +2.5 Vdc มาขยายสัญญาณโดยใช้ Operational Amplifier ด้วยอัตราขยาย G1 เพื่อให้มีแรงดันอยู่ในช่วง -12 ถึง +12 Vdc เพื่อใช้กับวงจร Half-Bridge



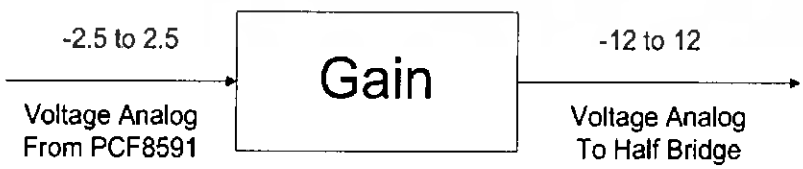
รูปที่ 3.4 สัญญาณดิจิตอลเทียบกับสัญญาณอะนาลอก



รูปที่ 3.5 สัญญาณอะนาลอกคงที่ที่นำมาลบ

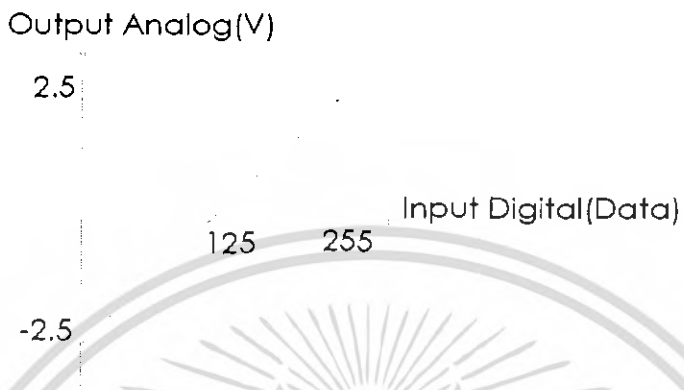


รูปที่ 3.6 ช่วงของแรงดันเมื่อนำมาลบกับสัญญาณคงที่ +2.5 Vdc



รูปที่ 3.7 แรงดันเมื่อผ่านตัว Amplifier

ซึ่งในการกำหนดค่าของสัญญาณควบคุมความเร็วนั้นซึ่งเป็นรูปแบบของสัญญาณดิจิตอล ขนาด 8 บิตนั้นจะกำหนดโดย 0x7F ถึง 0x00 เป็นการสัญญาณควบคุมความเร็วถอยหลัง และ 0x7F ถึง 0xFF เป็นการสัญญาณควบคุมความเร็วเดินหน้า ดังรูปที่ 3.8



รูปที่ 3.8 สัญญาณอินพุทที่เป็นดิจิตอลเทียบกับสัญญาณเอาท์พุทที่เป็นอะนาลอก

ไอซี PCF8591 นั้นเป็นไอซีที่ทำงานบนระบบบัส I2C (Inter-IC Communication) ดังนั้นจึงควรศึกษาการทำงานของระบบบัส I2C เสียก่อน

3.1.5 หลักการทำงานและคุณสมบัติของระบบบัส I2C

I2C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซีโดยบัส I2C ได้รับการพัฒนาขึ้นโดยบริษัท Phillips ด้วยจุดมุ่งหมายหลักคือ ต้องการให้ไอซีหรือโมดูลสามารถติดต่อ สั่งงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือสายข้อมูล และอีกเส้นหนึ่งคือ สายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I2C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือพ่วงกันไป ส่วนการกำหนดแอดเดรสหรือตำแหน่งสำหรับอุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบนบัส I2C มีชื่อเรียกอย่างเป็นทางการว่า สายข้อมูลอนุกรม หรือ SDA (Serial Data Line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรม หรือ SCL (Serial Clock Line) ซึ่งทั้ง SDA และ SCL นั้นต่างเป็นสายสัญญาณ 2 ทิศทาง (Bi-Directional Line) ต้องมีการต่อตัวต้านทานพูลอัพกับแรงดัน +5 Vdc ไว้ตลอดเวลา เพื่อให้สายมีสภานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุตของอุปกรณ์ที่ต่ออยู่บนบัส I2C ต้องมีลักษณะเป็นวงจรเดรนเปิด (Open-Drain) หรือคอลเล็กเตอร์เปิด (Open-Collector)

อัตราการการถ่ายทอดข้อมูลบนบัส I2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติและสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง อุปกรณ์ที่ต่อร่วมอยู่บนบัส I2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400 pF การเข้าถึงอุปกรณ์บนบัส I2C ใช้ข้อมูลสำหรับการเข้าถึง 2 ค่า คือ 7 บิต(7-bit Addressing) หรือ 10 บิต(10-bit Addressing)

บัส I2C ประกอบด้วยสายสัญญาณ 2 เส้น คือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดต่อกันอยู่ และอุปกรณ์ใดเป็นตัวรับหรือตัวส่ง ซึ่งจะนิยามที่ควรรู้ ดังนี้

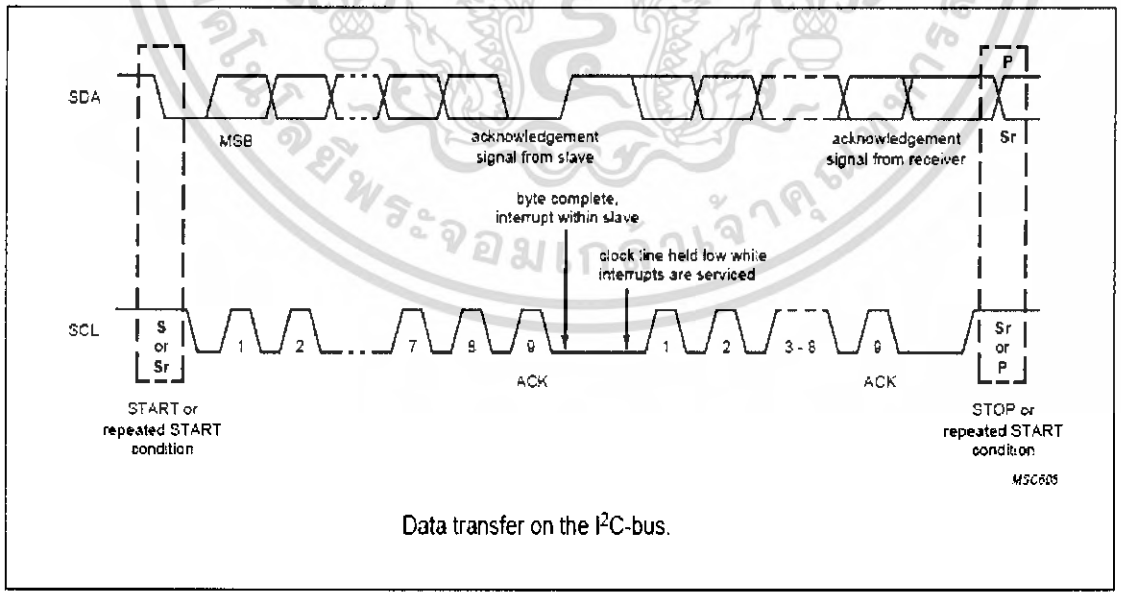
- อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I2C เรียกว่า มาสเตอร์ (Master)
- อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I2C เรียกว่า สเลฟ (Slave)

ข้อกำหนด 2 ประการที่สำคัญของการติดต่อบนบัส I2C มี ดังนี้

(1) การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

(2) ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

สภาวะที่เกิดขึ้นบนบัส I2C มีด้วยกัน 5 สภาวะ ดังรูปที่ 3.9 ซึ่งอธิบายได้ ดังนี้



Data transfer on the I²C-bus.

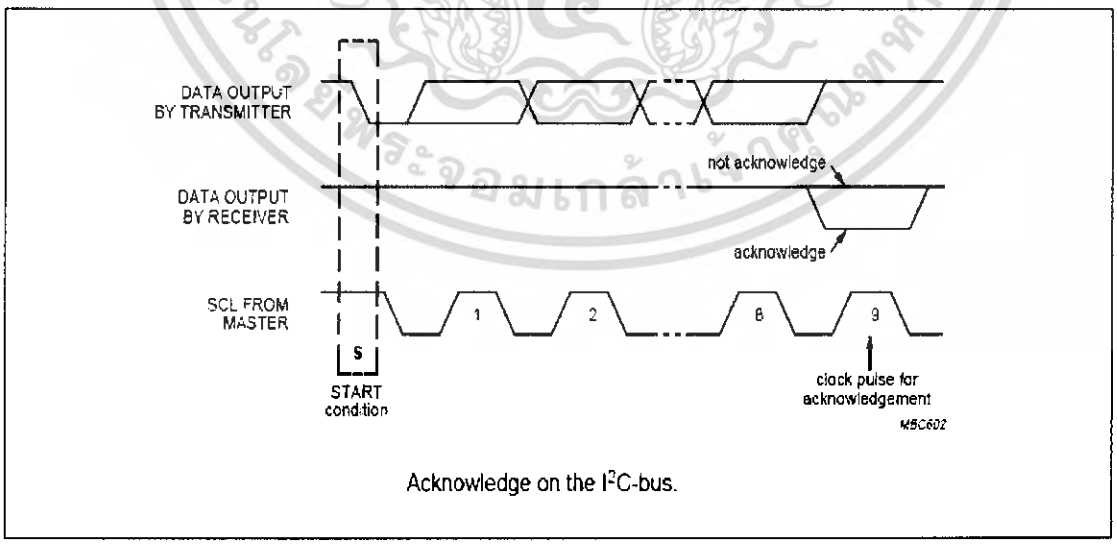
รูปที่ 3.9 สภาวะที่เกิดบนบัส I2C

(1) บัสว่าง (Bus Not Busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่นั่นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มขึ้นได้

(2) เริ่มต้นการถ่ายทอดข้อมูล (Start Data Transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นว่าสภาวะเริ่มต้น (START)

(3) ข้อมูลดำรงอยู่บนบัส (Data Valid) สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้นโดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือ ข้อมูลที่ทำการถ่ายทอดเมื่อสาย SCL มีสถานะเป็นลอจิกสูงและสถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่าเป็น "0" หรือ "1" ซึ่งข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL มีสถานะเป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่สาย SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะเป็นลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกของสาย SDA ในขณะที่สาย SCL มีสถานะลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูล จะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดนั้นเกิดความผิดพลาดขึ้น

(4) รับรู้ข้อมูล (Acknowledge) เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิตเรียกว่า บิตรับรู้ (Acknowledge bit) มีสถานะเป็นลอจิกสูง หลังจากที่ส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกาอุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดสัญญาณรับรู้ที่มีสถานะลอจิกต่ำเพื่อตอบสนองให้ทราบว่าได้รับข้อมูลเรียบร้อยแล้ว ดังรูปที่ 3.10



รูปที่ 3.10 Acknowledge บนสาย I2C

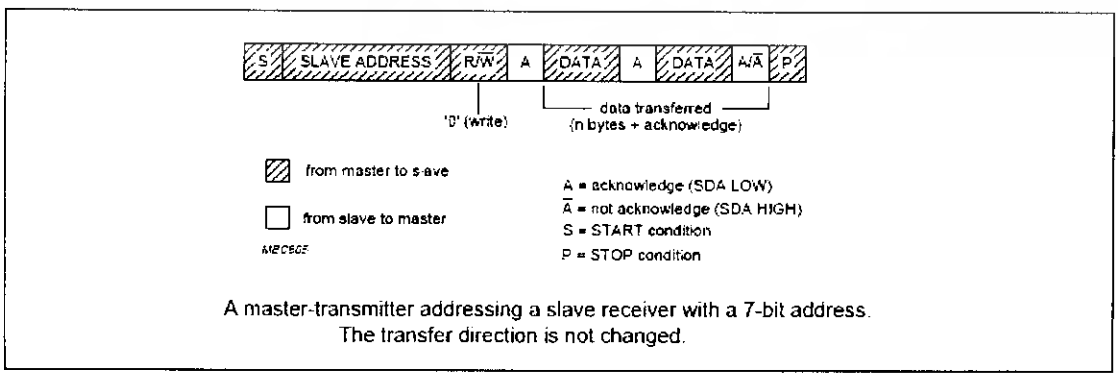
(5) หยุดการถ่ายทอดข้อมูล (Stop Data Transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูงในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นว่า สภาวะหยุด (STOP)

3.1.5.1 การทำงานบนบัส I2C

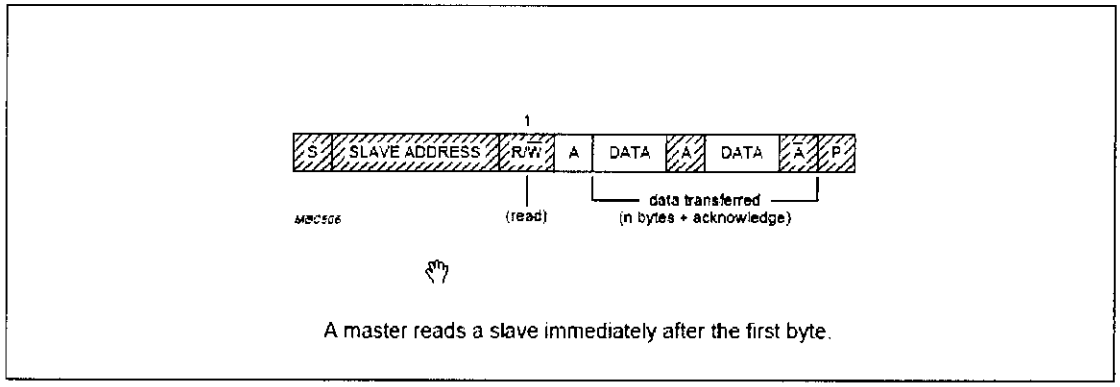
การทำงานบนบัส I2C จะเริ่มต้นด้วยการติดต่ออุปกรณ์เสียก่อน โดยการติดต่ออุปกรณ์บนบัส I2C นั้นจะใช้การเข้าถึงแบบ 7 บิต หรือ 10 บิต ก็ได้ ในกรณีที่มีอุปกรณ์ต่ออยู่บนบัสไม่มากนัก จะใช้การเข้าถึงแบบ 7 บิต แต่ในกรณีที่มีอุปกรณ์มากก็จำเป็นต้องใช้การเข้าถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้ว ก็จะเริ่มต้นการถ่ายทอดข้อมูล

3.1.5.2 การเข้าถึงแบบ 7 บิต (7-bit Addressing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างอิงถึงแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยจะแบ่งเป็นบิตกำหนดคงที่ (Fixed Address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิต จะเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (Programmable Address bit) โดยผู้ใช้งานจะต้องกำหนดสถานะลอจิกให้แก่ขา A0 , A1 , A2 ของอุปกรณ์ที่มีการเชื่อมต่อบนบัส I2C ส่วนในบิต LSB จะเป็นบิตที่ใช้กำหนดกาอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้นๆ หากบิต LSB มีค่าเป็น “0” จะหมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์สเลฟนั้น ถ้าเป็น “1” จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ ข้อมูลในไบต์ต่อมาคือ ข้อมูลควบคุม (Control byte) ในอุปกรณ์แต่ละตัวนั้นจะมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ข้อมูลในไบต์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง (Data) ซึ่งหลังจากที่มีการถ่ายทอดข้อมูลในแต่ละไบต์แล้ว อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง โดยที่อาจจะมีการรับ-ส่งข้อมูลหลายๆไบต์ติดต่อกันก็ได้ ตามรูปที่ 3.11 และ 3.12 ตามลำดับ จากนั้นจะจบการติดต่อกับอุปกรณ์สเลฟที่อ้างถึงด้วยสภาวะหยุด



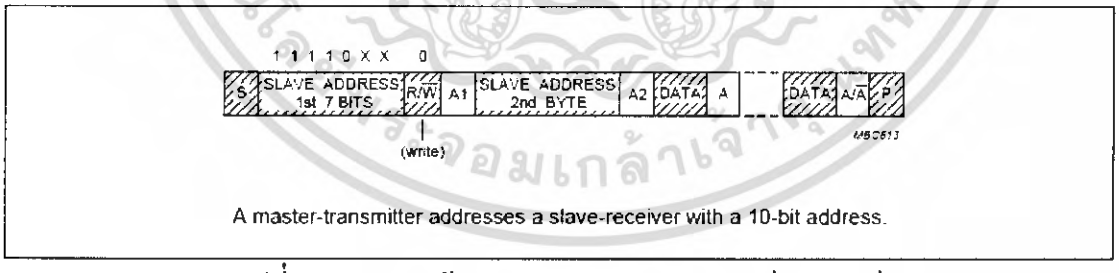
รูปที่ 3.11 การส่งข้อมูลแบบ 7 บิตของอุปกรณ์มาสเตอร์



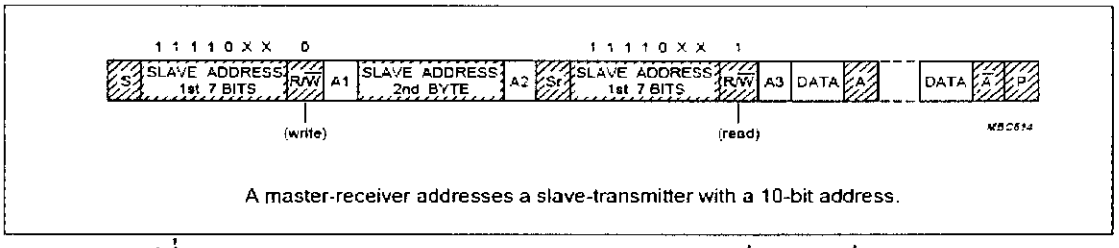
รูปที่ 3.12 การอ่านสัญญาณ Acknowledge ของอุปกรณ์มาสเตอร์แบบ 7 บิต

3.1.5.3 การเข้าถึงข้อมูลแบบ 10 บิต (10-bit Adressing)

จะเพิ่มเติมจากกาเข้าถึงแบบ 7 บิตอีกเล็กน้อย คือ โดยในไบต์แรกหลังจากสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบน มีข้อมูลเป็น 11110 ส่วนอีก 2 บิตถัดมาเป็นบิตแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่า ต้องการอ่านหรือเขียนกับข้อมูลกับอุปกรณ์สเลฟที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์สเลฟที่ต้องการติดต่อด้วย จากนั้นข้อมูลไบต์ถัดไปจึงเป็นข้อมูลควบคุม และก็จะตามด้วยไบต์ของข้อมูลที่ทำการถ่ายทอดจริงและเช่นเดียวกับการเข้าถึงแบบ 7 บิต หลังจากที่มีการถ่ายทอดข้อมูลในแต่ละไบต์แล้ว อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง โดยที่อาจจะมีการรับ-ส่งข้อมูลหลายๆไบต์ติดต่อกันก็ได้ ตามรูปที่ 3.13 และ3.14 ตามลำดับจากนั้นจะจบการติดต่อกับอุปกรณ์สเลฟที่อ้างถึงด้วยสภาวะหยุด



รูปที่ 3.13 การส่งข้อมูลแบบ 10 บิตของอุปกรณ์มาสเตอร์



รูปที่ 3.14 การอ่านสัญญาณ Acknowledge ของอุปกรณ์มาสเตอร์แบบ 10 บิต

3.1.5.4 การเขียนโปรแกรมเพื่อติดต่อกับบัส I2C

การเขียนโปรแกรมเพื่อติดต่อกับบัส I2C นั้น จะต้องเขียนโปรแกรมเพื่อสร้างสภาวะที่เกิดขึ้นบนบัสทั้ง 5 รูปแบบ คือ บัสว่าง (Bus Not Busy) , สภาวะเริ่มต้น (START) , ข้อมูลคำรงอยู่บนบัส (Data Valid) , รับรู้ข้อมูล (Acknowledge) และสภาวะหยุด (STOP) ซึ่งสามารถเขียนโปรแกรมแยกเป็นส่วนต่างๆได้ ดังนี้

```
sbit SDA=P2^0;                          กำหนด P2.0 เป็นสาย SDA
sbit SCL=P2^1;                          กำหนด P2.1 เป็นสาย SCL
void I2C_delay (void)                   ฟังก์ชั่นหน่วงเวลา
{
        unsigned char i;
        for(i=0;i<20;i++);
}

void I2C_clk (void)                     ฟังก์ชั่นสร้างสัญญาณนาฬิกา (clock) 1 ลูก
{
        SCL=0;
        I2C_delay();
        SCL=1;
        I2C_delay();
        SCL=0;
        I2C_delay();
}

void I2C_start (void)                   ฟังก์ชั่นสร้างสภาวะเริ่มต้น(START)
{
        SCL=0;
        SDA=1;
        SCL=1;
        SDA=0;
        I2C_delay();
}
```

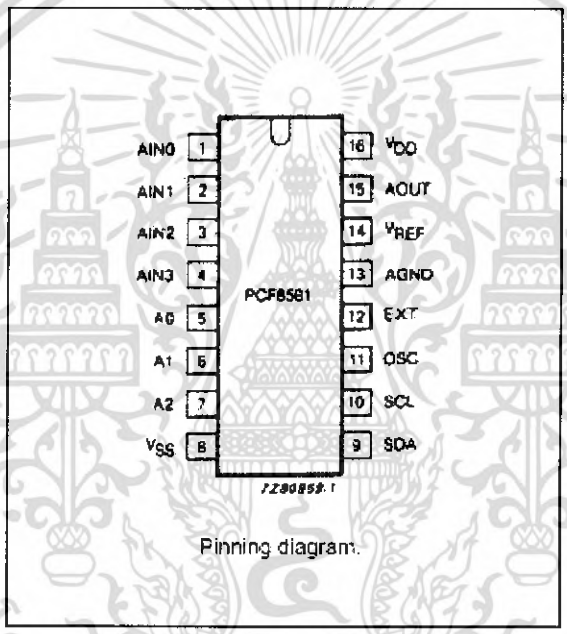
```
void I2C_stop (void)                    ฟังก์ชั่นสร้างสภาวะหยุด(STOP)
{
        SCL=0;
        SDA=0;
        SCL=0;
        SDA=1;
        I2C_delay();
}

void I2C_ack (void)                     ฟังก์ชั่นสร้างสัญญาณรับรู้ข้อมูล
                                        (Acknowledge bit) มีสถานะเป็นลอจิกต่ำ
{
        SDA=0;
        I2C_clk();
}

void I2C_Nack (void)                    ฟังก์ชั่นสร้างสัญญาณ Not Acknowledge
{                                       มีสถานะเป็นลอจิกสูง
        SDA=1;
        I2C_clk();
}

bit I2C_write (unsigned char dat)       ฟังก์ชั่นเขียนค่าข้อมูลลงบัส I2C
{
        bit send;
        unsigned char i;
        SCL=0;
        for(i=0;i<8;i++)
{                                       วนส่งค่าข้อมูลทีละบิต จนครบทั้ง 8 บิต
        send=dat&0x80;
        SDA=send;
```

```
        I2C_clk();
        dat=dat<<1;
}
        SDA=1;                          สร้างสัญญาณเพื่อรอรับบิตรับรู้ข้อมูล
        I2C_delay();                    (Acknowledge bit) จากตัวรับข้อมูล
        SCL=1;
        I2C_delay();
        send=SDA;                       เก็บค่าสถานะบิตรับรู้ข้อมูล (Acknowledge bit)
        return(send);                   ส่งค่าสถานะบิตรับรู้ข้อมูล (Acknowledge bit)
}                                       กลับไปยังส่วนที่เรียกใช้ โดยถ้าบิตรับรู้ข้อมูลมี
                                        สถานะเป็นลอจิกต่ำ แสดงว่าการส่ง-รับข้อมูล
                                        กับตัวรับข้อมูลทำได้สำเร็จแต่ถ้าบิตรับรู้ข้อมูล
                                        มีสถานะเป็นลอจิกสูงแสดงว่ามีความผิดพลาด
                                        ในการส่งข้อมูล

unsigned char I2C_read (void)           ฟังก์ชั่นอ่านค่าข้อมูลจากบัส I2C
{
        bit receive;
        unsigned char i;
        unsigned char dat=0x00;
        SDA=1;
        SCL=0;

        for(i=0;i<8;i++)                วนรับค่าข้อมูลทีละบิต จนครบทั้ง 8 บิต
{
        SCL=1;
        receive=SDA;
        dat=dat<<1;
        dat=dat|receive;
        SCL=0;
}
        SDA=0;                          ส่งสัญญาณบิตรับรู้ข้อมูล (Acknowledge bit)
```

```
I2C_clk();                    มีสถานะเป็นลอจิกต่ำ เพื่อให้ตัวส่งข้อมูลรู้ว่า
                              ว่าการรับข้อมูลเป็นไปอย่างถูกต้อง
return(dat);                  คืนค่าข้อมูลที่อ่านได้กลับไปยังส่วนที่เรียกใช้
}
```

ไอซี PCF8591 เป็นไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอล 4 ช่องทางและแปลงสัญญาณดิจิดอลเป็นอะนาลอก 1 ช่องทาง

3.2 คุณสมบัติทางเทคนิคที่สำคัญของ PCF8591



รูปที่ 3.15 ขาของ PCF8591

- ทำงานโดยใช้แหล่งจ่ายไฟชุดเดียว
- ทำงานที่แรงดัน 2.5 ถึง 6 Vdc
- กินกระแสขณะอยู่ในสภาวะสแตนบายต์ต่ำ
- ติดต่อกับไมโครคอมพิวเตอร์หรือไมโครคอนโทรลเลอร์ผ่านระบบบัส I2C
- สัญญาณอะนาลอกมีระดับแรงดันตั้งแต่ V_{SS} จนถึง V_{DD}
- วงจรแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิดอลแบบซักเซสซีฟ แอปพร็อกซิเมชั่น ขนาด 8 บิต
- วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณอะนาลอกขนาด 8 บิต 1 ช่องทาง

- เลือกตำแหน่งแอดเดรสทางฮาร์ดแวร์จากขา A0 - A2 ทำให้สามารถต่อพ่วงกันได้สูงสุดถึง 8 ตัว

- อัตราการสุ่มข้อมูล (Sampling) ขึ้นอยู่กับความเร็วสัญญาณนาฬิกาบนบัส I2C

- วงจรแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลสามารถรับสัญญาณอะนาลอกได้ 4 ช่องทาง ทั้งยังเลือกได้ว่าให้ทำงานแบบแยกช่องทางหรือทำงานเป็นวงจรดิฟเฟอเรนเชียล

การอ่านค่าสามารถกำหนดให้เลื่อนช่องโดยอัตโนมัติได้รายละเอียดของตำแหน่งขาต่างๆมีดังนี้

- AN0-AN3 อินพุตสำหรับป้อนสัญญาณอะนาลอกที่ต้องการแปลงค่า

- A0-A2 ขากำหนดแอดเดรสทางฮาร์ดแวร์ ทำให้สามารถต่อใช้งานร่วมกันได้สูงสุด 8 ตัว

- V_{SS} ขากราวด์

- SDA , SCL ขาเชื่อมต่อระบบบัส I2C

- OSC ขาต่อกับสัญญาณนาฬิกาภายนอกเมื่อขา EXT ต่อกับ +5 Vdc และเป็นเอาต์พุตสัญญาณนาฬิกาถ้าขา EXT ต่อลงกราวด์

- EXT ขาเลือกแหล่งกำเนิดสัญญาณนาฬิกา ถ้าต่อ +5 Vdc เป็นการเลือกใช้สัญญาณนาฬิกาจากภายนอกโดยต่อสัญญาณนาฬิกาเข้าที่ขา OSC ถ้าต่อขานี้ลงกราวด์ จะเป็นการเลือกใช้สัญญาณนาฬิกาจากภายใน

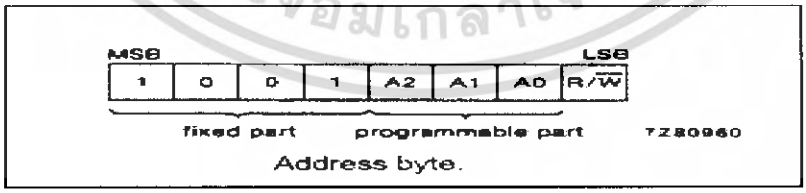
- AGND ขากราวด์ของแรงดันอ้างอิง ปกติต่อลงกราวด์

- V_{REF} ขาป้อนแรงดันอ้างอิง ปกติต่อกับ +5 Vdc

- AOUT ขาเอาต์พุตของแปลงสัญญาณดิจิตองเป็นสัญญาณอะนาลอก

- V_{DD} ขาต่อกับไฟเลี้ยง ใช้ได้ตั้งแต่ +2 ถึง +6 Vdc ปกติใช้ +5 Vdc

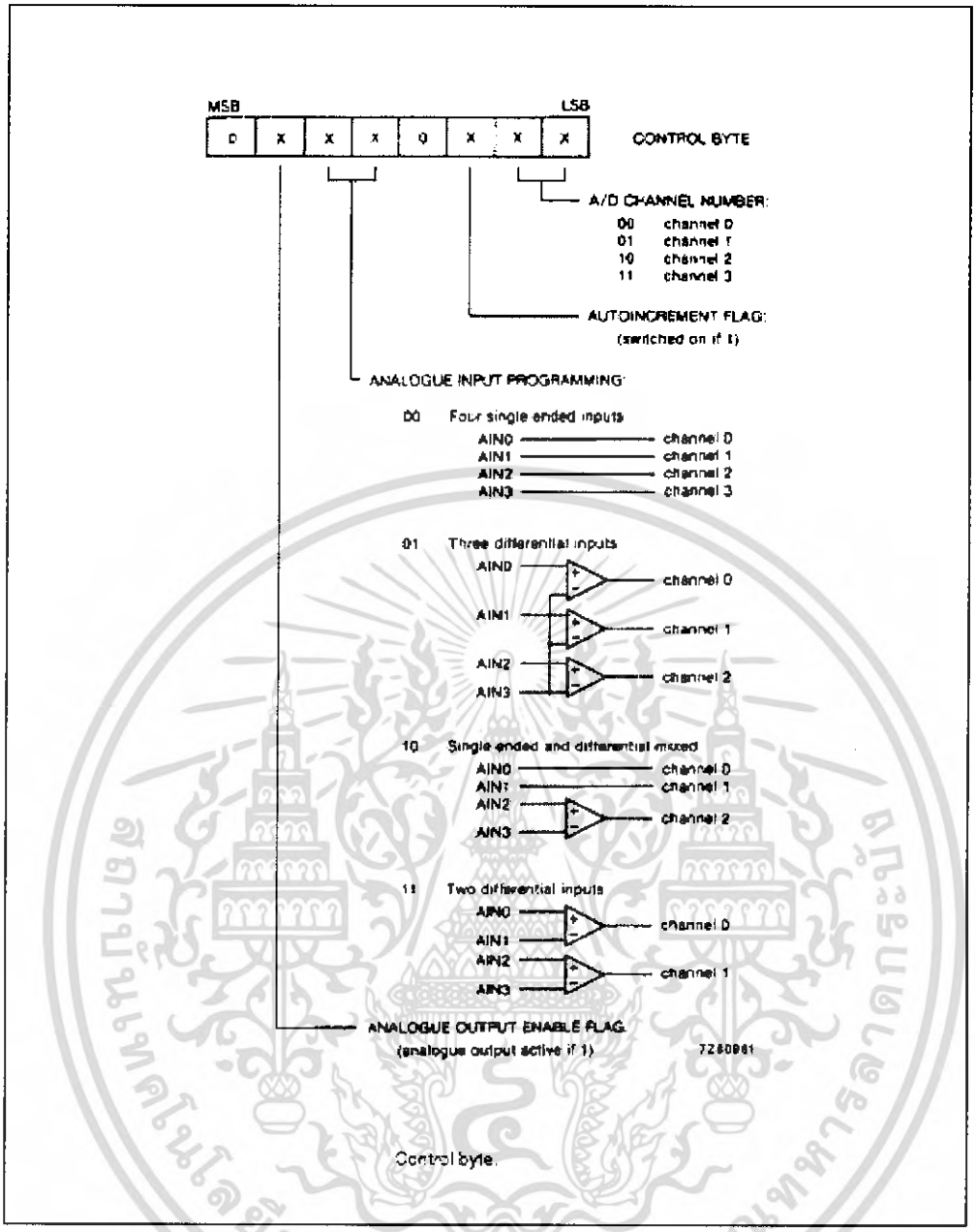
โดยบิตกำหนดคงที่ (Fixed Address bit) จำนวน 4 บิต ของไอซี PCF8591 นั้นจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ คือ 1001 ดังรูปที่ 3.16



รูปที่ 3.16 บิตกำหนดคงที่(Fixed Address bit) 4 บิต

3.2.1 ข้อมูลควบคุม

หลังจากที่ส่งข้อมูลกำหนดแอดเดรสให้กับ PCF8591 แล้ว จะต้องส่งข้อมูลควบคุมตามไปเพื่อกำหนดคุณสมบัติของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลและวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณอะนาลอกภายใน PCF8591 ดังรูปที่ 3.17 โดยสามารถอธิบายได้ดังนี้



รูปที่ 3.17 บิตควบคุมของ PCF8591

- บิตที่ 6 ของข้อมูลควบคุมใช้สำหรับการเอ็นเอเบิลขาอะนาลอกเอาต์พุต เมื่อต้องการเอ็นเอเบิลต้องกำหนดให้ เป็น “1”

- บิตที่ 4 และบิตที่ 5 ของข้อมูลควบคุมใช้สำหรับการกำหนดรูปแบบของสัญญาณอะนาลอกอินพุตป้อนให้แก่ PCF8591

- บิตที่ 2 ใช้เลือกรูปแบบการอ่านข้อมูลจากขาอินพุตอะนาลอกว่าจะเป็นการอ่านจากเพียงอินพุตเดียวหรืออ่านแบบเรียงลำดับจากทุกอินพุต ถ้าต้องการอ่านแบบเรียงลำดับต้องกำหนดให้บิตนี้เป็น “1”

- บิตที่ 0 และ บิตที่ 1 ใช้สำหรับการกำหนดช่องของอินพุตอะนาลอกที่ต้องการอ่าน ถ้ากำหนดให้บิตที่ 2 เป็น “1” หลังจากอ่านค่าจากบิตที่ 0 และบิตที่ 1 แล้ว ในการอ่านค่าครั้งต่อไปจะเป็นการอ่านค่าอินพุตจากช่องที่ 1 ซึ่งสามารถกำหนดค่าของข้อมูลควบคุมได้ดังนี้

- 0x40 เปิดการทำงานของวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอก และเลือกอ่านสัญญาณอะนาลอกจากช่อง 0

- 0x41 เปิดการทำงานของวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอก และเลือกอ่านสัญญาณอะนาลอกจากช่อง 1

- 0x42 เปิดการทำงานของวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอก และเลือกอ่านสัญญาณอะนาลอกจากช่อง 2

- 0x43 เปิดการทำงานของวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอก และเลือกอ่านสัญญาณอะนาลอกจากช่อง 3

โดยที่ข้อมูลควบคุมทั้งหมดจะถูกเก็บไว้ในรีจิสเตอร์ควบคุมภายใน PCF8591 และเมื่อจ่ายไฟให้แก่ PCF8591 เป็นครั้งแรก บิตต่างๆของข้อมูลควบคุมภายในรีจิสเตอร์ควบคุมจะมีค่าเป็น “0”

วงจรออสซิลเลเตอร์ภายใน PCF8591 จะสร้างสัญญาณนาฬิกาสำหรับการแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอล เมื่อต้องการใช้วงจรออสซิลเลเตอร์ภายใน ให้ต่อขา EXT ลงกราวด์ แต่ถ้าต้องการใช้วงจรออสซิลเลเตอร์จากภายนอก ขา EXT จะต้องต่อกับไฟบวก และป้อนสัญญาณนาฬิกาเข้าที่ขา OSC ของ PCF8591 โดยที่ความถี่ของสัญญาณนาฬิกาสูงสุดที่ป้อนให้กับออสซิลเลเตอร์ต้องไม่เกิน 1.25 MHz

3.2.2 โครงสร้างโปรแกรมสำหรับติดต่อกับไอซี PCF8591

ในโครงงานนี้ ทางคณะผู้จัดทำได้เลือกใช้การเข้าถึงข้อมูลแบบ 7 บิต และเนื่องจากหุ่นยนต์ Rocker-Bogie นั้นเป็นหุ่นยนต์ 6 ล้อ จึงต้องใช้ไอซี PCF8591 จำนวน 6 ตัว มาควบคุมการทำงานของวงจร Half-Bridge โดยที่จะมีแอดเดรสเริ่มจาก 1(001:A2,A1,A0) ไปจนถึง 6(110:A2,A1,A0) ซึ่งเมื่อต้องการติดต่อกับกับไอซี PCF8591 จะต้องส่งค่าแอดเดรส ดังนี้

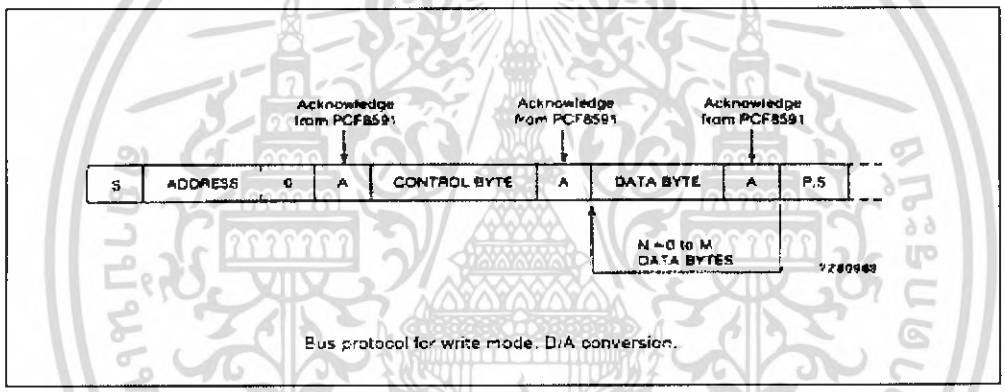
- 0x92 เพื่อเขียนข้อมูลกับไอซี PCF8591 ตัวที่ 1 เพื่อควบคุมความเร็วของล้อหน้าด้านซ้าย
- 0x94 เพื่อเขียนข้อมูลกับไอซี PCF8591 ตัวที่ 2 เพื่อควบคุมความเร็วของล้อหน้าด้านขวา
- 0x96 เพื่อเขียนข้อมูลกับไอซี PCF8591 ตัวที่ 3 เพื่อควบคุมความเร็วของล้อกลางด้านซ้าย
- 0x98 เพื่อเขียนข้อมูลกับไอซี PCF8591 ตัวที่ 4 เพื่อควบคุมความเร็วของล้อกลางด้านขวา
- 0x9A เพื่อเขียนข้อมูลกับไอซี PCF8591 ตัวที่ 5 เพื่อควบคุมความเร็วของล้อหลังด้านซ้าย
- 0x9C เพื่อเขียนข้อมูลกับไอซี PCF8591 ตัวที่ 6 เพื่อควบคุมความเร็วของล้อหลังด้านขวา
- 0x93 เพื่ออ่านข้อมูลกับไอซี PCF8591 ตัวที่ 1 เพื่ออ่านค่าความเร็วของล้อหน้าด้านซ้าย
- 0x95 เพื่ออ่านข้อมูลกับไอซี PCF8591 ตัวที่ 2 เพื่ออ่านค่าความเร็วของล้อหน้าด้านขวา

- 0x97 เพื่ออ่านข้อมูลกับไอซี PCF8591 ตัวที่ 3 เพื่ออ่านค่าความเร็วของล้อกลางด้านซ้าย
- 0x99 เพื่ออ่านข้อมูลกับไอซี PCF8591 ตัวที่ 4 เพื่ออ่านค่าความเร็วของล้อกลางด้านขวา
- 0x9B เพื่ออ่านข้อมูลกับไอซี PCF8591 ตัวที่ 5 เพื่ออ่านค่าความเร็วของล้อหลังด้านซ้าย
- 0x9D เพื่ออ่านข้อมูลกับไอซี PCF8591 ตัวที่ 6 เพื่ออ่านค่าความเร็วของล้อหลังด้านขวา

โดยสามารถเขียนฟังก์ชั่นเพื่อเขียนและอ่านข้อมูลกับไอซี PCF8591 ได้ดังนี้

1. ฟังก์ชั่นเขียนค่าให้วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณอะนาลอก (write_D2A)

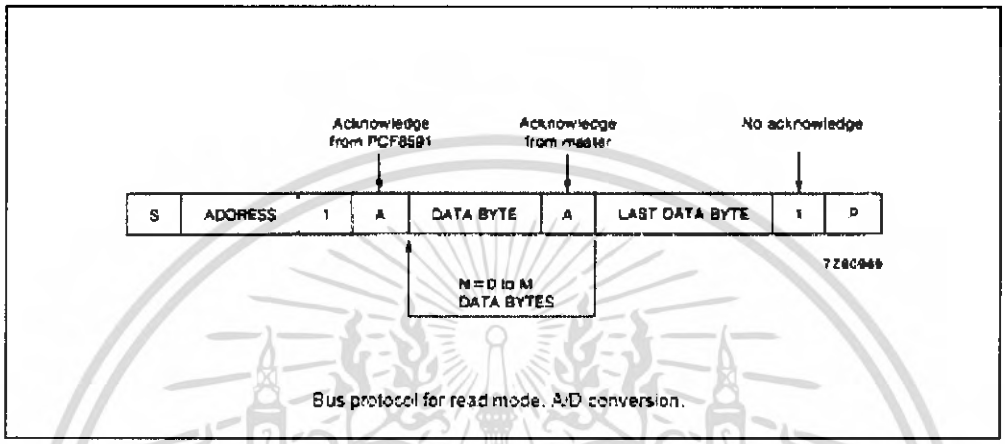
ฟังก์ชั่นนี้จะใช้ในการควบคุมความเร็วของมอเตอร์กระแสตรง โดยที่ฟังก์ชั่นนี้จะรับค่าอาร์กิวเมนต์ (Argument) 2 ค่า คือ ค่าแอดเดรสของไอซี PCF8591 ที่ต้องการติดต่อและค่าของความเร็วตามลำดับ จากนั้นจะเรียกใช้ฟังก์ชั่นการทำงานของบัส I2C ซึ่งจะเป็นไปตามรูปแบบดังรูปที่ 3.18



รูปที่ 3.18 การเรียกฟังก์ชั่นการทำงานของบัส I2C

```
void write_D2A (unsigned char address,unsigned char speed)
{
        I2C_start();                          สภาวะเริ่มต้น
        I2C _write(address);                  เขียนค่าแอดเดรสของไอซี PCF8591 ที่
                                              ต้องการติดต่อ
        I2C _write(0x40);                     เขียนค่าข้อมูลควบคุม

        I2C _write(speed);                    เขียนค่าข้อมูล ซึ่งก็คือค่าความเร็ว
        I2C _stop();                          สภาวะหยุด
}
```

ฟังก์ชั่นนี้จะใช้สำหรับอ่านค่าแรงดันป้อนกลับ (Back EMF) ที่ต่ออยู่กับอินพุตอะนาลอกช่องที่ 1 เพื่อนำค่ากลับมาแสดงผล โดยจะถูกส่งต่อไปที่โปรแกรม Rocker-Bogie Robot Control โดยที่ฟังก์ชั่นนี้จะรับค่าอาร์กิวเมนต์ ที่เป็นค่าแอดเดรสของไอซี PCF8591 ที่ต้องการติดต่อ จากนั้นจะเรียกใช้ฟังก์ชั่นการทำงานของบัส I2C เพื่ออ่านค่าจากไอซี PCF8591 และคืนค่ากลับไปยังส่วนที่เรียกใช้ด้วย ซึ่งจะเป็นไปตามรูปแบบดังรูปที่ 3.19

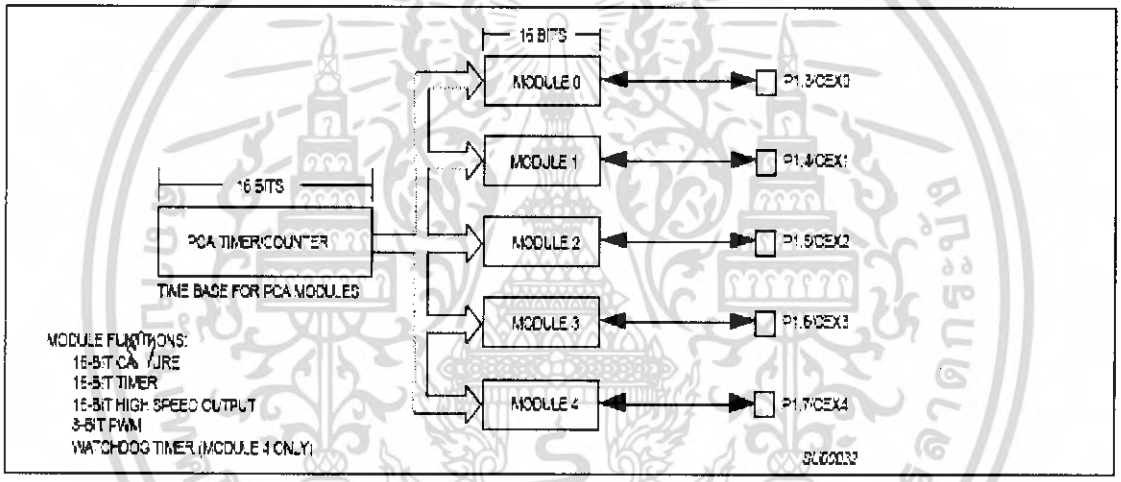


รูปที่ 3.19 การอ่านฟังก์ชั่นการทำงานของบัส I2C

```
unsigned char read_A2D (unsigned char address)
{
    unsigned char temp;                 ประกาศตัวแปรชั่วคราวเพื่อใช้เก็บค่าข้อมูล
    I2C _start();                       สภาวะเริ่มต้น
    I2C _write(address);                เขียนค่าแอดเดรสของไอซี PCF8591 ที่
                                        ต้องการติดต่อ
    I2C _write(0x40);                   เขียนค่าข้อมูลควบคุม
    I2C _stop();                        สภาวะหยุด
    I2C _start();                       สภาวะเริ่มต้น
    I2C c_write(address+1);             เขียนค่าแอดเดรสของไอซี PCF8591 เพื่ออ่าน
                                        ค่า
    temp = i2c_read();                  เก็บค่าไว้ที่ตัวแปร temp
    I2C _Nack();                        ส่งสัญญาณเพื่อหยุดการจองบัส I2C
    I2C _stop();                        สภาวะหยุด
    return(temp);                       คืนค่าข้อมูลกลับไปยังส่วนที่เรียกใช้
}
```

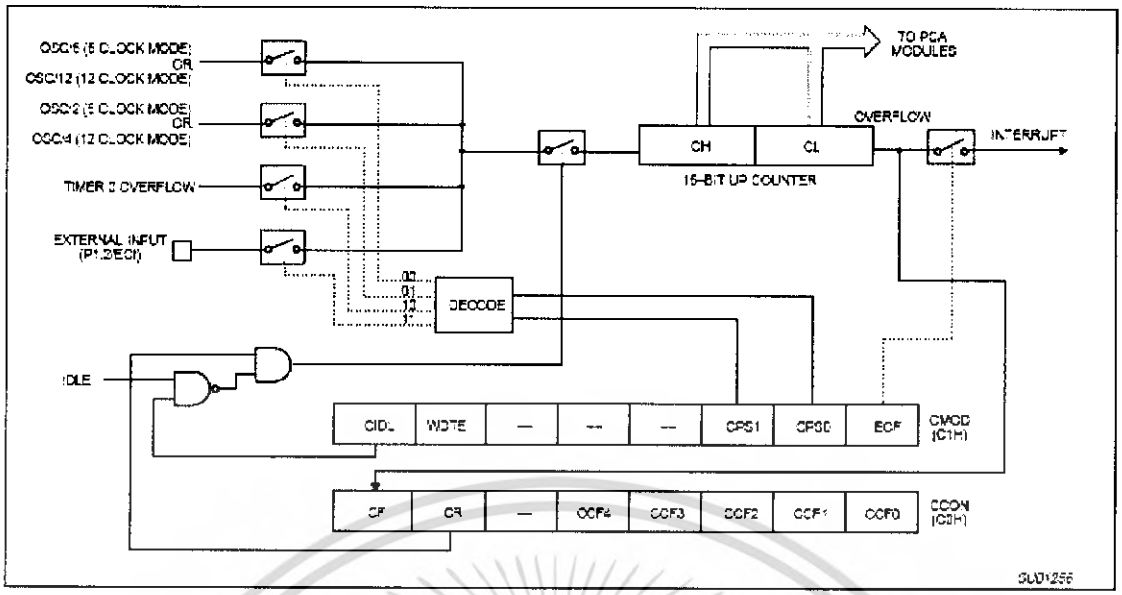
3.3 การเขียนโปรแกรมสร้างสัญญาณตามความกว้างพัลส์ (Pulse-Width Modulation)

ไมโครคอนโทรลเลอร์ P89C51RD2BN นั้นจะมีความสามารถพิเศษที่เพิ่มเติมเข้ามาคือ โมดูลวงจรนับโปรแกรมได้ (Programmable Counter Array : PCA) 5 ช่องทางอิสระต่อกัน ซึ่งภายในโมดูล PCA นั้นจะบรรจุความสามารถพิเศษอยู่ 5 อย่าง คือ วงจรตรวจจับสัญญาณ(PCA Capture Mode) , วงจรเปรียบเทียบสัญญาณ(PCA Compare Mode) , วงจรสัญญาณเอาต์พุตความเร็วสูง(PCA Hi-Speed Output Mode) , วงจรมอดูเลชั่นตามความกว้างพัลส์ (PCA Pulse-Width Modulation) และวงจรวอตช์ดอกไทเมอร์ (PCA Watchdog Timer Mode) โดยคณะผู้จัดทำจะใช้วงจรสร้างสัญญาณดามความกว้างพัลส์ เพื่อสร้างสัญญาณตามความกว้างพัลส์ 5 ช่องทาง อิสระต่อกัน ดังรูปที่ 3.20 เพื่อใช้ควบคุมการทำงานของเซอร์โวมอเตอร์ทั้ง 6 ตัว ที่ควบคุมทิศทางของหุ่นยนต์อยู่



รูปที่ 3.20 โมดูลวงจรนับโปรแกรมได้ (Programmable Counter Array : PCA)

โดยที่โมดูล PCA นั้นจะมีการทำงาน ดังรูปที่ 3.21 โดยโครงสร้างหลักของโมดูล PCA คือ ตัวนับ PCA (PCA TimerCounter) ขนาด 16 บิต ประกอบด้วย CH และ CL ซึ่งต่างเป็นตัวนับขนาด 8 บิตมารวมกัน โดยที่ฐานเวลาที่ใช้นี้จะใช้ฐานเวลาร่วมกันทั้ง 5 ช่องทาง ซึ่งสามารถเลือกแหล่งกำเนิดฐานเวลาได้โดยการกำหนดค่า CPS1 และ CPS0 ในรีจิสเตอร์ CMOD ดังรูปที่ 3.22 ซึ่งเป็นรีจิสเตอร์ที่ใช้กำหนดโหมดฐานเวลาของโมดูล PCA ซึ่งคณะผู้จัดทำได้เลือกใช้โหมด 2 คือ ใช้สัญญาณโอเวอร์โฟลวของไทเมอร์ 0 เพื่อจะสร้างฐานเวลา 20 มิลลิวินาที ซึ่งเป็นฐานเวลาที่เซอร์โวมอเตอร์ต้องการ



รูปที่ 3.21 โครงสร้างการทำงานของโมดูล PCA

CMOD Address = D9H — Reset Value = 00XX X000B

CIDL	WDTE	–	–	–	CPS1	CPS0	ECF
7	6	5	4	3	2	1	0

Bit:

Symbol	Function
CIDL	Counter idle control: CIDL = 0 programs the PCA Counter to continue functioning during idle Mode. CIDL = 1 programs it to be gated off during idle.
WDTE	Watchdog Timer Enable: WDTE = 0 disables Watchdog Timer function on PCA Module 4. WDTE = 1 enables it.
–	Not implemented, reserved for future use.*
CPS1	PCA Count Pulse Select bit 1.
CPS0	PCA Count Pulse Select bit 0.

CPS1	CPS0	Selected PCA Input**	
0	0	0	Internal clock, f_{OSC}/6 in 6-clock mode (f_{OSC}/12 in 12-clock mode)
0	1	1	Internal clock, f_{OSC}/2 in 6-clock mode (f_{OSC}/4 in 12-clock mode)
1	0	2	Timer 0 overflow
1	1	3	External clock at ECI/P1.2 pin (max. rate = f_{OSC}/4 in 6-clock mode, f_{OSC}/8 in 12-clock mode)

ECF	PCA Enable Counter Overflow interrupt: ECF = 1 enables CF bit in CCON to generate an interrupt. ECF = 0 disables that function of CF.

NOTE:
* User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.
** f_{OSC} = oscillator frequency

SU01318

รูปที่ 3.22 รีจิสเตอร์ CMOD

โดยสามารถเลือกการทำงานของโมดูล PCA ได้จากการกำหนดค่าในรีจิสเตอร์ CCAPMn ของทั้ง 5 ช่องทางซึ่งเป็นอิสระต่อกัน ซึ่งถ้าต้องการใช้วงจรสร้างสัญญาณความความกว้างพัลส์ (Pulse-Width Modultion) นั้น จะต้องกำหนดค่าเป็น 0x42 หรือ x1001x0x ดังรูปที่ 3.23

CCAPMn Address CCAPM0 0DAH
CCAPM1 0DBH
CCAPM2 0DCH
CCAPM3 0DDH
CCAPM4 0DEH

Reset Value = X000 0000B

Not Bit Addressable

–	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn
Bit: 7	6	5	4	3	2	1	0

Symbol	Function
–	Not implemented, reserved for future use*.
ECOMn	Enable Comparator. ECOMn = 1 enables the comparator function.
CAPPn	Capture Positive, CAPPn = 1 enables positive edge capture.
CAPNn	Capture Negative, CAPNn = 1 enables negative edge capture.
MATn	Match. When MATn = 1, a match of the PCA counter with this module's compare/capture register causes the CCFn bit in CCON to be set, flagging an interrupt.
TOGn	Toggle. When TOGn = 1, a match of the PCA counter with this module's compare/capture register causes the CEXn pin to toggle.
PWMn	Pulse Width Modulation Mode. PWMn = 1 enables the CEXn pin to be used as a pulse width modulated output.
ECCFn	Enable CCF interrupt. Enables compare/capture flag CCFn in the CCON register to generate an interrupt

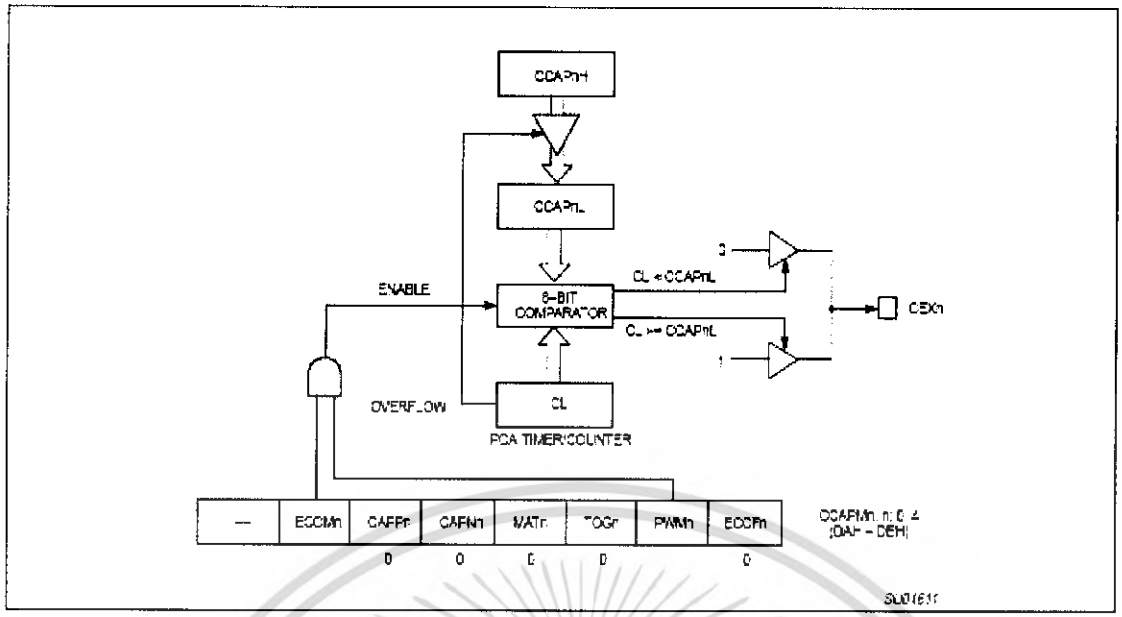
NOTE:
*User software should not write 1s to reserved bits. These bits may be used in future 8051 family products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1. The value read from a reserved bit is indeterminate.

SU01333

–	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	MODULE FUNCTION
X	0	0	0	0	0	0	0	No operation
X	X	1	0	0	0	0	X	16-bit capture by a positive-edge trigger on CEXn
X	X	0	1	0	0	0	X	16-bit capture by a negative trigger on CEXn
X	X	1	1	0	0	0	X	16-bit capture by a transition on CEXn
X	1	0	0	1	0	0	X	16-bit Software Timer
X	1	0	0	1	1	0	X	16-bit High Speed Output
X	1	0	0	0	0	1	0	8-bit PWM
X	1	0	0	1	X	0	X	Watchdog Timer

รูปที่ 3.23 รีจิสเตอร์ CCAPMn

โดยที่วงจรสร้างสัญญาณตามความกว้างพัลส์ (Pulse-Width Modulation) นั้นมีโครงสร้างดังรูปที่ 3.24 ซึ่งวงจร PWM นั้นจะใช้ตัวนับ PCA เพียง 8 บิต คือใช้เพียงแค่ CL เท่านั้น โดยการทำงานนั้นจะนำค่าของตัวนับ CL มาเปรียบเทียบกับค่าในรีจิสเตอร์ CCAPnL ซึ่งถ้าค่าของตัวนับ CL มีค่าน้อยกว่าค่าในรีจิสเตอร์ CCAPnL จะทำให้ได้สัญญาณเอาท์พุตมีสถานะลอจิกต่ำ แต่ถ้าค่าของตัวนับ CL มีค่ามากกว่าหรือเท่ากับค่าในรีจิสเตอร์ CCAPnL จะทำให้ได้สัญญาณเอาท์พุตมีสถานะลอจิกสูง โดยที่เมื่อตัวนับ CL นับค่าถึงค่าสูงสุดแล้วเกิดโอเวอร์โฟลวนั้น วงจรจะทำการโหลดค่าใน CCAPnH ซึ่งมีค่าเท่ากับค่าใน CCAPnL มาใส่ใน CCAPnL ซึ่งการโหลดค่าแบบนี้จะสามารถป้องกันการกระชากของสัญญาณได้ซึ่งเราสามารถเปลี่ยนค่าดิวตี้ไซเกิล (Duty-Cycle) ได้โดยการเปลี่ยนค่าในรีจิสเตอร์ CCAPnL และ CCAPnH ส่วนคาบเวลาของสัญญาณตามความกว้างพัลส์นั้น จะกำหนดจากการโอเวอร์โฟลวของไทเมอร์ 0



รูปที่ 3.24 วงจรสร้างสัญญาณตามความกว้างพัลส์ (Pulse-Width Modulation)

โดยถ้าต้องการคาบเวลา 20 มิลลิวินาที โดยใช้ XTAL = 11.0592 MHz และทำงานในโหมด 6 แมชชีนไซเกิล จะสามารถคำนวณได้ ดังนี้

1 machine cycle จะใช้เวลา = 6 / XTAL = 6 / 11.0592MHz = 0.5425 µs

1 timer overflow จะใช้เวลา = 256*0.5425 µs = 138.8 µs

จำนวนการโอเวอร์โฟลว = 20 ms / 138.8 µs = 144 µs

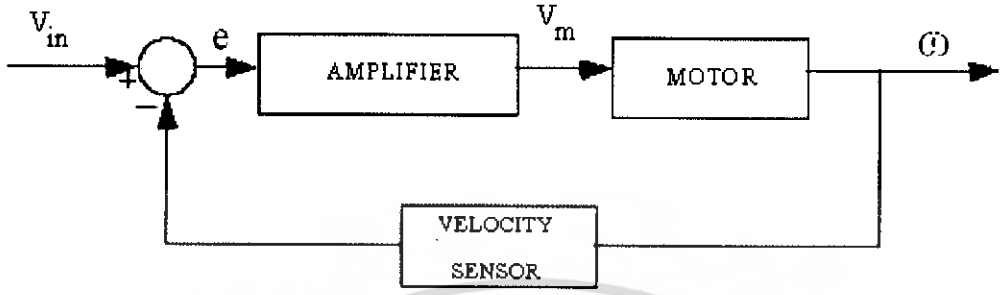
ดังนั้นต้องตั้งค่าไทเมอร์ 0 ให้ทำงานในโหมด 2 (Auto-Reload) และตั้งค่า TH0 และ TL0 ให้มีค่าเท่ากับ 256-144 ซึ่งมีค่าเท่ากับ 112 หรือ 0x70

เซอร์โวมอเตอร์นั้นต้องการสัญญาณตามความกว้างพัลส์เพื่อควบคุมมุมเลี้ยว โดยสัญญาณตามความกว้างพัลส์จะต้องอยู่ในช่วง 0.5-2.5 มิลลิวินาที ทุกๆ 20 มิลลิวินาที ซึ่งรีจิสเตอร์ CCAPnL ที่ใช้กำหนดค่าดิวตี้ (Duty Cycle) นั้นเป็นรีจิสเตอร์ขนาด 8 บิต ทำให้สามารถกำหนดค่าได้ตั้งแต่ 248-226 เพื่อได้ซึ่งสัญญาณที่มีความกว้าง 0.5-2.5 มิลลิวินาที ดังกล่าว

3.4 การควบคุมความเร็วของหุ่นยนต์

การควบคุมความเร็วของหุ่นยนต์นั้น จะใช้วงจร Half-Bridge เป็นวงจรขยายสัญญาณ (Driver Circuit) เพื่อควบคุมความเร็วรอบของดีซีมอเตอร์ทั้ง 6 ตัว โดยจะทำงานอยู่บนพื้นฐานของของการควบคุมแบบป้อนกลับ (Feedback Control) คือ การวัดค่าตัวแปรกระบวนการ (Process Variable) ซึ่งก็คือ ความเร็วรอบของดีซีมอเดอร์ เพื่อนำมาเปรียบเทียบกับตัวแปรเป้าหมาย (Set-Point Variable) แล้วนำค่าความคลาดเคลื่อนที่ได้ไปชดเชยกระบวนการอีกที ดังรูปที่ 3.25 ซึ่งการควบคุมแบบป้อนกลับนี้จะทำให้กระบวนการเข้าสู่สภาวะคงตัว (Steady-State) ได้เร็วขึ้น และยัง

สามารถปรับตัวเข้าสู่สภาวะคงตัวได้เมื่อเกิดสัญญาณรบกวน (Disturbance) ขึ้น ซึ่งการควบคุมความเร็วของดีซีมอเตอร์โดยใช้การป้อนกลับนั้น จำเป็นจะต้องวัดค่าความเร็วรอบของดีซีมอเตอร์ โดยจะสามารถวัดได้จากค่าแรงดันย้อนกลับ (Armature Feedback) ของดีซีมอเตอร์



รูปที่ 3.25 บล็อกไดอะแกรมของการควบคุมความเร็วของมอเตอร์แบบป้อนกลับ

3.4.1 หลักการทำงานของดีซีมอเตอร์

ดีซีมอเตอร์เป็นทรานสดิวเซอร์แรงบิดซึ่งมีการออกแบบให้มีคุณลักษณะพิเศษ คือ แรงบิดของเพลาของดีซีมอเตอร์จะได้มาจากผลคูณระหว่างสนามแม่เหล็กและขดลวดตัวนำ โดยถ้ามีกระแสไหลในขดลวดตัวนำจะทำให้เกิดสนามแม่เหล็กที่ประกอบด้วยเส้นแรงแม่เหล็ก Ø ซึ่งถ้าขดลวดตัวนำเหล่านั้นอยู่ห่างจากศูนย์กลางการหมุนเท่ากับ r จะได้ความสัมพันธ์ระหว่างแรงบิดของเพลากับกระแสอาร์เมเจอร์ คือ

$$T=K\varphi I \qquad (3\text{-}1)$$

เมื่อ T คือ แรงบิดของเพลามีหน่วยเป็นนิวตัน – เมตร
Ø คือ เส้นแรงแม่เหล็กมีหน่วยเป็นเวเบอร์
I คือ กระแสมีหน่วยเป็นแอมแปร์
K คือ ตัวคงที่

ซึ่งในขณะที่ขดลวดตัวนำเคลื่อนที่ตัดสนามแม่เหล็กก็จะทำให้เกิดแรงดันดกคร่อมตัวมันเอง ซึ่งแรงดันนี้จะเป็นสัดส่วนกับความเร็วของเพลาของมอเตอร์ดีซีและด้านการไหลของกระแสอาร์เมเจอร์ โดยจะเรียกว่า แรงดันย้อนกลับหรือแรงดันป้อนกลับ (Back emf,Armature Feedback) โดยจะได้ความสัมพันธ์ระหว่างแรงดันย้อนกลับกับความเร็วของเพลามอเตอร์ คือ

$$E=K\varphi\omega \qquad (3\text{-}2)$$

เมื่อ E คือ แรงดันย้อนกลับ emf มีหน่วยเป็นโวลท์
Ø คือ เส้นแรงแม่เหล็กมีหน่วยเป็นเวเบอร์
ω คือ ความเร็วของมอเตอร์มีหน่วยเป็นเรเดียน / วินาที

โดยรูปที่ 3.26 จะเป็นการแสดงถึงโมเดลของดีซีมอเตอร์แบบแยกขดกระตุ้น โดยที่

Ra : ความต้านทานของอาร์เมเจอร์

La : อินดัคแตนซ์ของอาร์เมเจอร์

Vg : แรงดันในอาร์เมเจอร์ (โวลท์เตจย้อนกลับ)

Rf : ความต้านทานของฟิลด์

Lf : อินดัคแตนซ์ของฟิลด์

ϕ : ช่องว่างอากาศของเส้นแรงสนามแม่เหล็ก

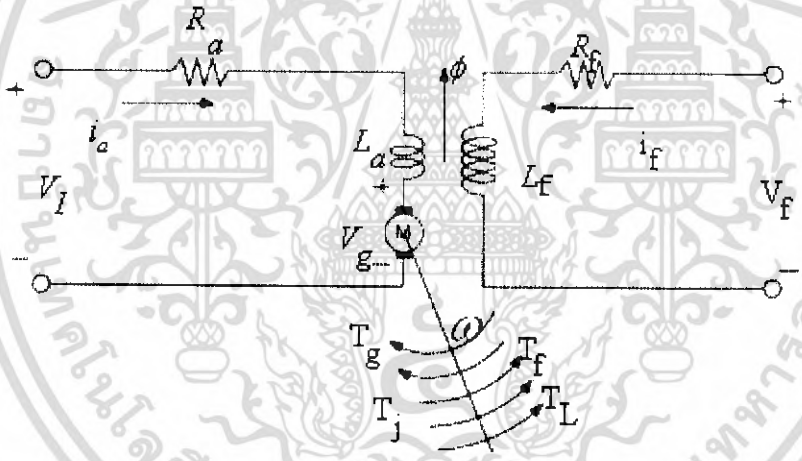
ω : ความเร็วของเพลาอาร์เมเจอร์

Tg : แรงบิดที่พัฒนาขึ้นในมอเตอร์

Tf : แรงบิดเสียดทานของมอเตอร์

Tj : แรงเฉื่อยของมอเตอร์

K_t : ค่าคงที่ของแรงบิดของมอเตอร์



รูปที่ 3.26 โมเดลของดีซีมอเตอร์แบบแยกขดกระตุ้น

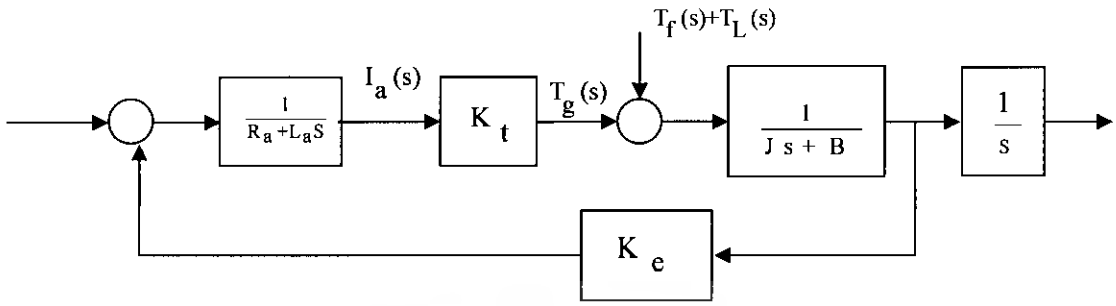
ซึ่งสามารถเขียนเป็นสมการได้ ดังนี้

$$Vi(t)=R_a i_a(t)+L_a \frac{di_a(t)}{dt}+V_g(t) \qquad (3\text{-}3)$$

$$T_g(t) = K_t i_a(t) \qquad (3\text{-}4)$$

$$V_g(t) = K_e \omega(t) \qquad (3\text{-}5)$$

และสามารถนำมาเขียนเป็นบล็อกไดอะแกรม ได้ดังนี้



รูปที่ 3.27 บล็อกไดอะแกรมของดีซีมอเตอร์แบบแยกขดกระตุ้น

3.4.2 การวัดค่าแรงดันย้อนกลับ

จากสมการของดีซีมอเตอร์

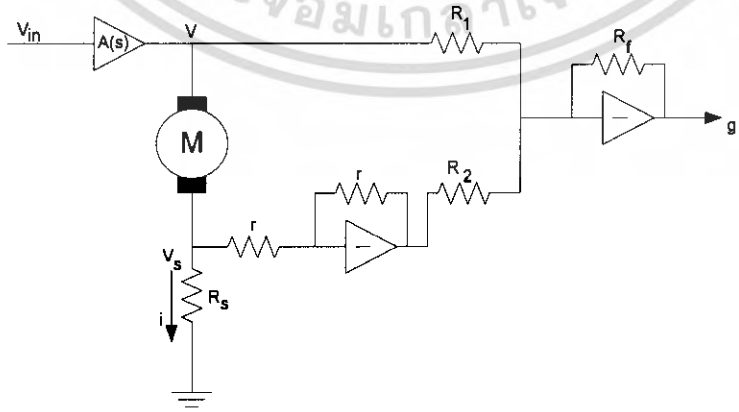
$$V_i(t) = R_a i_a(t) + K_e \omega(t) \tag{3-6}$$

เมื่อ K_e คือค่าคงที่ของแรงดันย้อนกลับ

ดังนั้นจะสามารถหาค่าความเร็วมอเตอร์ได้ คือ

$$\omega(t) = \frac{1}{K_e}\left[V_1(t) - R_a i_a(t)\right] \tag{3-7}$$

ซึ่งจากสมการที่ (3.7) สามารถนำมาสร้างเป็นวงจรได้ ดังแสดงในรูปที่ 3.28



รูปที่ 3.28 วงจรควบคุมความเร็วดีซีมอเตอร์

สมมติว่าเอาท์พุตของ Operational Amplify มีอัตราขยายเป็นอินฟินีตี้ แรงดัน V_g กำหนดได้โดย

$$V_g(t) = -\frac{R_f}{R_1}V(t) + \frac{R_f}{R_2}V_s(t) \quad (3\text{-}8)$$

เนื่องจาก

$$V_s(t) = R_s i(t) \quad (3\text{-}9)$$

ดังนั้น จะได้

$$V_g(t) = -\frac{R_f}{R_1}V(t) + \frac{R_f R_s}{R_2}i(t) \quad (3\text{-}10)$$

เพื่อให้ V_g เป็นสัดส่วนกับความเร็วของมอเตอร์ ω ดังนั้น จะต้องกำหนดให้

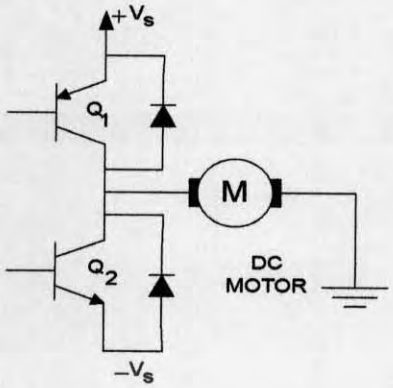
$$\frac{R_f R_s}{R_2} = R_a \quad (3\text{-}11)$$

ดังนั้น จะได้

$$V_g(t) = \frac{-K_e R_f}{R_1}\omega(t) \quad (3\text{-}10)$$

3.4.3 วงจร Half-Bridge

วงจร Half-Bridge นั้นจัดเป็นวงจรแบบลิเนียร์ไบโพลาร์ดังรูปที่3.29โดยในวงจรจะประกอบด้วย ทรานซิสเตอร์ 2 ตัว ตัวหนึ่งเป็นแบบ NPN อีกตัวหนึ่งเป็นแบบ PNP โดยถ้าให้สัญญาณเป้าหมาย (Set-Piont Variable) มีค่าเป็นบวกแล้ว จะทำให้ทรานซิสเตอร์ Q_1 ซึ่งเป็นทรานซิสเตอร์แบบ NPN นำกระแส ทำให้มอเตอร์หมุนไปด้านหนึ่ง แต่ถ้าให้สัญญาณเป้าหมาย (Set-Point Variable) มีค่าเป็นลบ จะทำให้ทรานซิสเตอร์ Q_2 ซึ่งเป็นทรานซิสเตอร์แบบ PNP นำกระแส จะทำให้มอเตอร์หมุนกลับด้าน โดยจะพบว่าวงจร Half-Bridge ต้องใช้แรงดันที่มีค่าทั้งบวกและลบเพื่อที่จะควบคุมมอเตอร์ให้หมุนได้ครบทั้ง 2 ทิศทาง ซึ่งจะทำงานได้ครบทั้ง 4 Quadrant

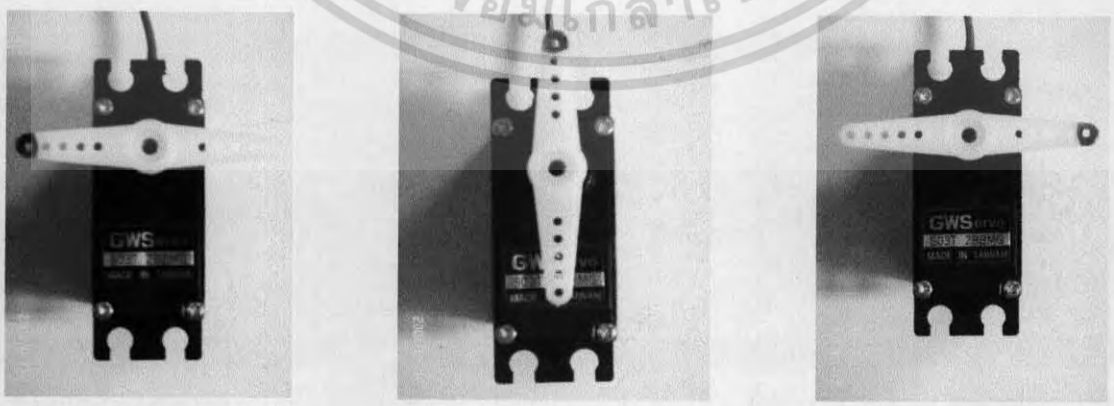


รูปที่ 3.29 วงจร Half-Bridge

3.5 การควบคุมทิศทางของหุ่นยนต์

การควบคุมทิศทางของหุ่นยนต์นั้นจะใช้เซอร์โวมอเตอร์ 6 ตัว มาเป็นตัวควบคุมมุมเลี้ยวของล้อทั้ง 6 ล้อ ซึ่งเซอร์โวมอเตอร์นั้นสามารถหมุนได้ 180 องศา คือจาก -90 องศา ถึง 90 องศา โดยที่เซอร์โวมอเตอร์นั้นจะรับสัญญาณควบคุมที่เป็นสัญญาณตามความกว้างพัลส์ (Pulse-Width Modulation) ที่มีความกว้างพัลส์ตั้งแต่ 0.5 ถึง 2.5 มิลลิวินาที ทุกๆ 20 มิลลิวินาทีหรือคาบนั่นเอง ดังรูปที่ 3.30 และสามารถอธิบายได้ดังนี้

- สัญญาณตามความกว้างพัลส์ กว้าง 0.5 มิลลิวินาที คาบ 20 มิลลิวินาทีจะทำให้เซอร์โวหมุนไปที่ -90 องศา
- สัญญาณตามความกว้างพัลส์ กว้าง 1.5 มิลลิวินาที คาบ 20 มิลลิวินาทีจะทำให้เซอร์โวหมุนไปที่ 0 องศา
- สัญญาณตามความกว้างพัลส์ กว้าง 2.5 มิลลิวินาที คาบ 20 มิลลิวินาทีจะทำให้เซอร์โวหมุนไปที่ +90 องศา



(ก) -90 องศา (ข) 0 องศา (ค) +90 องศา

รูปที่ 3.30 เซอร์โวมอเตอร์

ซึ่งสัญญาณตามความกว้างพัลส์ที่จะมาควบคุมการทำงานของเซอร์โวมอเตอร์ทั้ง 6 ตัวนั้น จะมาจากไมโครคอนโทรลเลอร์ซึ่งสามารถสร้างสัญญาณตามความกว้างพัลส์ได้ 5 ช่องทาง โดยที่ล้อกลางของหุ่นยนต์ด้านซ้ายและขวาจะใช้สัญญาณตามความกว้างพัลส์ช่องทางเดียวกัน ซึ่งจะทำให้สัญญาณตามความกว้างพัลส์ 5 ช่องทางนั้นเพียงพอต่อการควบคุมการทำงานของเซอร์โวมอเดอร์ทั้ง 6 ตัว โดยที่รีจิสเตอร์ CCAPnL ซึ่งเป็นรีจิสเตอร์ขนาด 8 บิต ที่จะนำมาเก็บค่าเพื่อสร้างสัญญาณตามความกว้างในช่วง 0.5-2.5 มิลลิวินาที และคาบเวลา 20 มิลลิวินาทีนั้น จะสามารถแปรค่าได้ 23 ค่า ทำให้ความละเอียดของมุมเลี้ยวของเซอร์โวมอเตอร์เป็น 8.18 องศาต่อการเปลี่ยนค่าในรีจิสเตอร์ CCAPnL และยังสามารถทำงานด้านการเลี้ยวได้ครบถ้วนทุกรูปแบบด้วย

3.6 การเคลื่อนที่ของหุ่นยนต์

การเคลื่อนที่ของหุ่นยนต์นั้นสามารถแบ่งออกเป็น 2 หัวข้อ คือ ความเร็วและการเลี้ยว

3.6.1 ความเร็วของหุ่นยนต์

จากที่ได้กล่าวมาแล้วนั้น ความเร็วของหุ่นยนต์จะถูกแบ่งออกเป็น 4 ช่วง หรือ 4 เกียร์นั่นเอง คือ เกียร์ว่าง เกียร์1 เกียร์2 เกียร์3 และเกียร์4 โดยที่ในแต่ละเกียร์ที่เพิ่มขึ้นจะเป็นการเพิ่มความเร็วที่ละ 25% คือ 0% 25% 50% 75% และ 100% ซึ่งหุ่นยนต์นั้นจะสามารถเคลื่อนที่ได้ก็ต่อเมื่อมีทั้งคำสั่งเดินหน้าและคำสั่งความเร็ว (ค่าเกียร์ไม่ใช่เกียร์ว่าง) พร้อมกัน ถ้ามีเฉพาะคำสั่งเดินหน้า(ความเร็วเป็นเกียร์ว่าง)หรือคำสั่งความเร็วอย่างใดอย่างหนึ่ง หุ่นยนต์จะไม่เคลื่อนที่

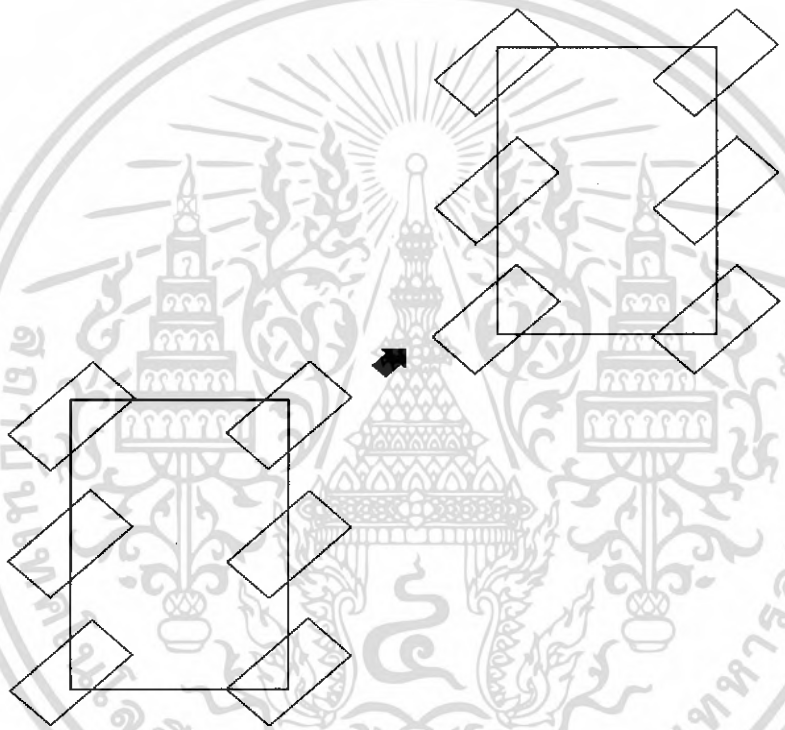
รูปที่ 3.31 ความเร็วของหุ่นยนต์แบ่งตามระดับเกียร์

3.6.2 การเลี้ยวของหุ่นยนต์

เนื่องจากหุ่นยนต์ Rocker-Bogie นั้นจะมีความสามารถในการเลี้ยวที่ดี คือสามารถเลี้ยวได้หลายรูปแบบ ซึ่งสามารถใช้ได้กับพื้นที่ๆต่างกัน และสามารถแบ่งได้เป็น 3 รูปแบบ ดังนี้

3.6.2.1 การเลี้ยวแบบกะทันหัน

การเลี้ยวแบบกะทันหัน คือ การเลี้ยวที่ล้อทั้ง 6 ล้อนั้นจะหมุนเลี้ยวไปที่มุมเดียวกันทั้งหมด ทำให้สามารถเลี้ยวได้ในพื้นที่แคบๆได้ ซึ่งสามารถใช้ได้ทั้งกับในบริเวณพื้นที่ทั่วไปหรือการออกจากบริเวณที่มีพื้นที่แคบๆ หรือเพื่อหลบหลีกสิ่งกีดขวาง

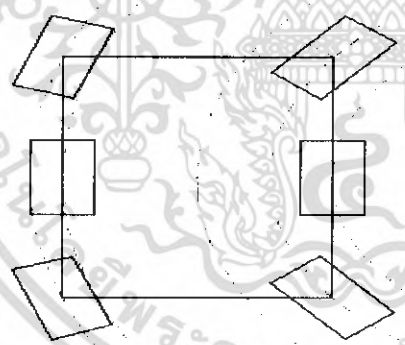


รูปที่ 3.32 การเลี้ยวแบบกะทันหัน

3.6.2.2 การเลี้ยวแบบรัศมี

การเลี้ยวแบบรัศมี คือ การเลี้ยวเป็นวงโค้งที่เส้นทางของการเคลื่อนที่จะเป็นรัศมีของวงกลม ซึ่งขนาดของรัศมีนั้นอาจจะมากหรือน้อยก็ได้ขึ้นอยู่กับมุมเลี้ยว ซึ่งจะใช้กับบริเวณพื้นที่ทั่วไปและไม่เหมาะกับบริเวณพื้นที่แคบๆ เนื่องจากต้องการพื้นที่ในการเลี้ยวมากพอสมควร การเลี้ยวแบบรัศมีของหุ่นยนต์ Rocker-Bogie ซึ่งเป็นหุ่นยนต์ 6 ล้อนั้นจะต้องควบคุมทั้งมุมเลี้ยวและความเร็วของล้อต่างๆให้สัมพันธ์กัน เพื่อที่หุ่นยนต์จะได้เลี้ยวได้อย่างถูกต้อง โดยจะบังคับให้มุมเลี้ยวของล้อกลางทั้ง 2 ข้าง คือ ล้อกลางด้ายซ้ายและล้อกลางด้านขวานั้นเป็น 0 องศา ตลอดเวลา คือ เป็นจุดเริ่มต้น (Origin) ของวงกลมรัศมี r ซึ่งมุมของล้อทั้ง 4 ล้อที่เหลือ คือ ล้อหน้าด้านซ้าย ล้อ

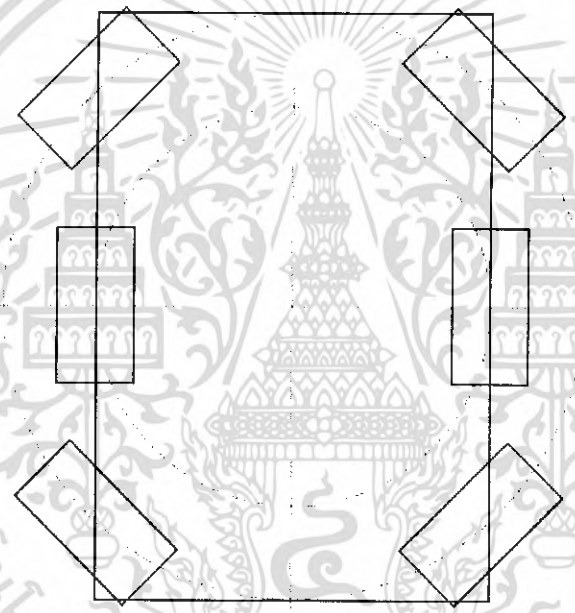
หน้าด้านขวา ล้อหลังด้านซ้าย และล้อหลังด้านขวานั้นจะต้องทำงานสัมพันธ์กัน ซึ่งมุมการเลี้ยวของล้อหน้าและล้อหลังของทั้งสองด้านนั้นจะต้องทำงานตรงข้ามกันเพื่อที่รูปแบบของการเลี้ยวจะได้เป็นรูปรัศมีวงกลมที่มีจุดเริ่มต้นเดียวกัน ตามรูปที่ 3.33 จากนั้นจะต้องแบ่งล้อออกเป็น 2 ด้านตามการเลี้ยว คือ ล้อด้านในและล้อด้านนอกโดยเมื่อเป็นการเลี้ยวแบบรัศมีด้านซ้ายนั้น ล้อด้านในหมายถึง ล้อหน้าและล้อหลังด้านซ้าย ส่วนล้อด้านนอกก็คือ ล้อหน้าและล้อหลังด้านขวา ส่วนในกรณีที่เป็นการเลี้ยวรัศมีด้านขวานั้น ล้อด้านในก็คือ ล้อหน้าและล้อหลังด้านขวา และล้อด้านนอกก็คือล้อหน้าและล้อหลังด้านซ้าย และต้องควบคุมความเร็วของล้อด้านในและล้อด้านนอกให้ทำงานสัมพันธ์กันเพื่อป้องกันการลื่นไถล (Slip) และทำงานได้อย่างถูกต้อง โดยที่จะกำหนดให้ความเร็วของหุ่นยนต์คือความเร็วของวงกลมที่มีรัศมีกึ่งกลางของล้อด้านในและล้อด้านนอก ซึ่งความเร็วของล้อด้านในจะต้องต่ำกว่าความเร็วของหุ่นยนต์ และความเร็วของล้อด้านนอกจะต้องสูงกว่าความเร็วของหุ่นยนต์



รูปที่ 3.33 การเลี้ยวแบบรัศมี

3.6.2.3 การเลี้ยวแบบวงกลม

การเลี้ยวแบบวงกลม คือ การเลี้ยวที่ล้อทั้ง 6 ล้อทำมุมเป็นรูปวงกลม ดังรูปที่ 3.34 ซึ่งใช้สำหรับหมุนเพื่อเปลี่ยนทิศทางการเคลื่อนที่ของหุ่นยนต์เพื่อออกจากพื้นที่แคบๆหรือหลบหลีกสิ่งกีดขวาง ซึ่งเมื่อจัดล้อทั้ง 6 ล้อให้เป็นรูปวงกลมโดยให้ล้อกลางของหุ่นยนต์ทั้ง 2 ข้างมีมุมเป็น 0 องศาแล้ว จะพบว่าประกอบด้วยวงกลม 2 วงที่มีรัศมีไม่เท่ากัน คือ วงกลมที่มีรัศมีมากกว่าที่ประกอบด้วยล้อหน้าด้านซ้าย ล้อหน้าด้านขวา ล้อหลังด้านซ้าย และล้อหลังด้านขวา และวงกลมที่มีรัศมีน้อยกว่าซึ่งประกอบด้วยล้อกลางด้านซ้ายและด้านขวา ซึ่งจะต้องควบคุมความเร็วของล้อในแต่ละวงกลมให้สัมพันธ์กันเพื่อป้องกันการลื่นไถล (Slip) และทำงานได้อย่างถูกต้อง



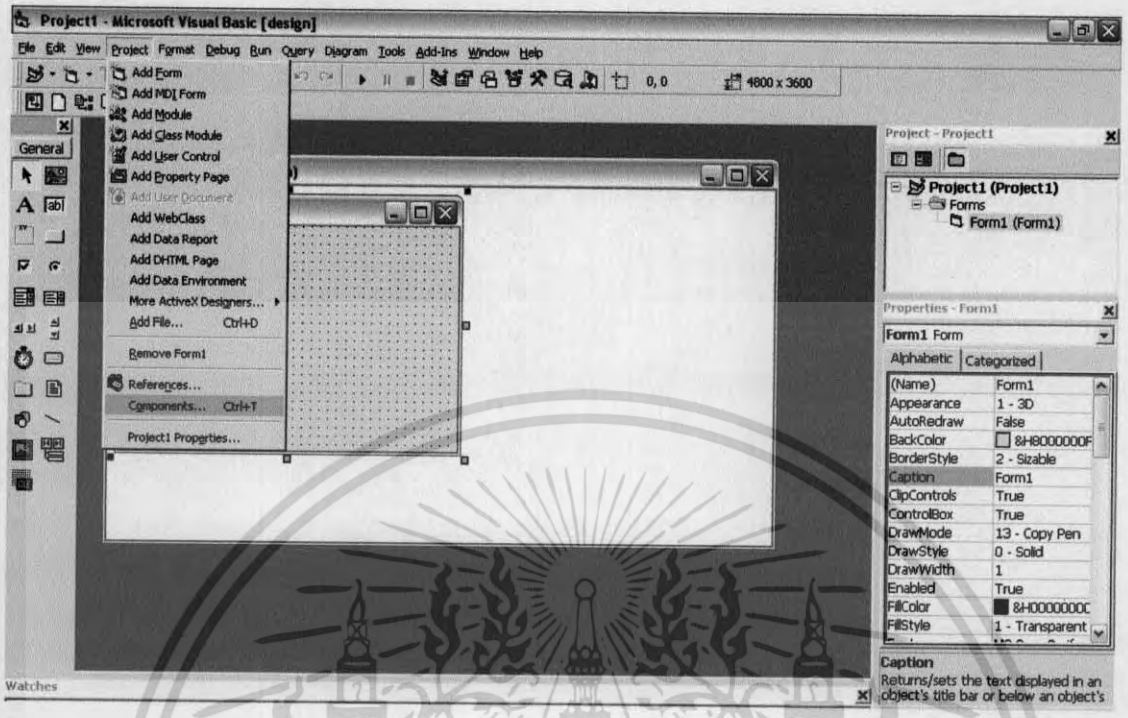
รูปที่ 3.34 การเลี้ยวแบบวงกลม

3.7 โปรแกรม Rocker-Bogie Robot Control

3.7.1 การติดต่อกับ Serial Port ด้วย Visual Basic

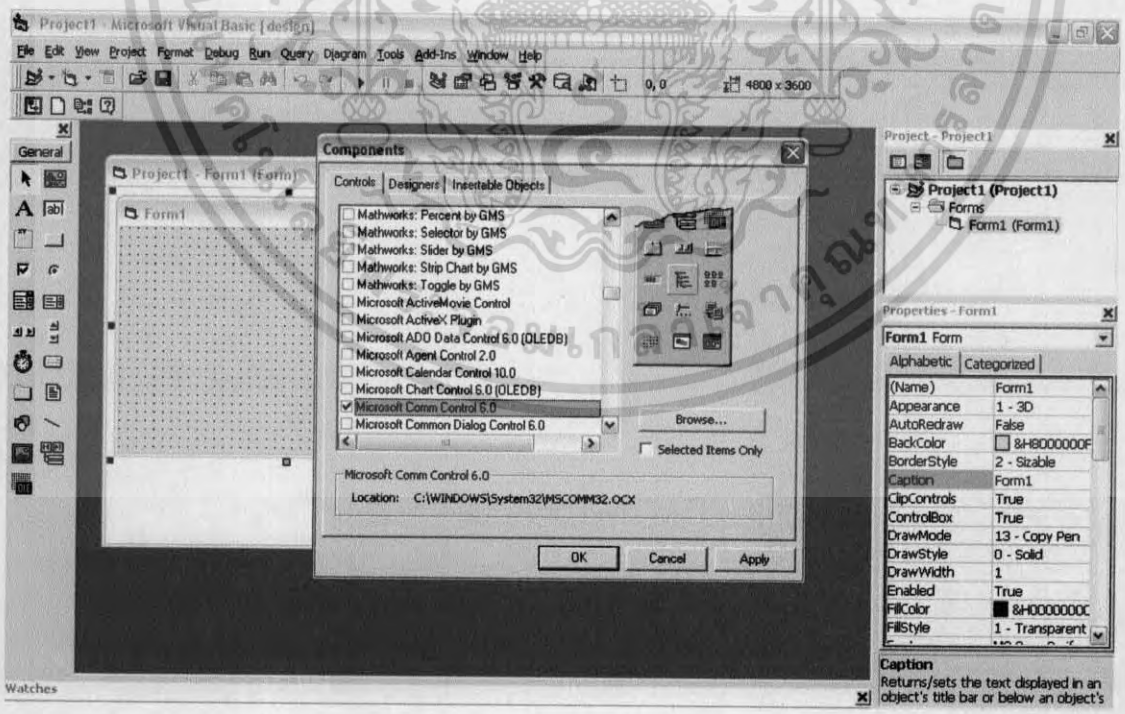
สามารถทำได้โดยใช้ VB Control ที่ชื่อว่า MSComm โดยต้องกำหนด Custom Control เข้าไปที่เมนู Project → Companents แล้วเลือกที่ช่อง MSComm จะปรากฏไอคอนรูปโทรศัพท์สีเหลืองคลิกที่ไอคอนลากนำมาไว้บน Form ใน Project โดยสามารถทำตามวิธีที่กล่าวมา ดังรูปต่อไปนี้

1. เลือกที่เมนูบาร์ด้านบนของปรแกรม Visual Basic ดังรูปด้านล่าง



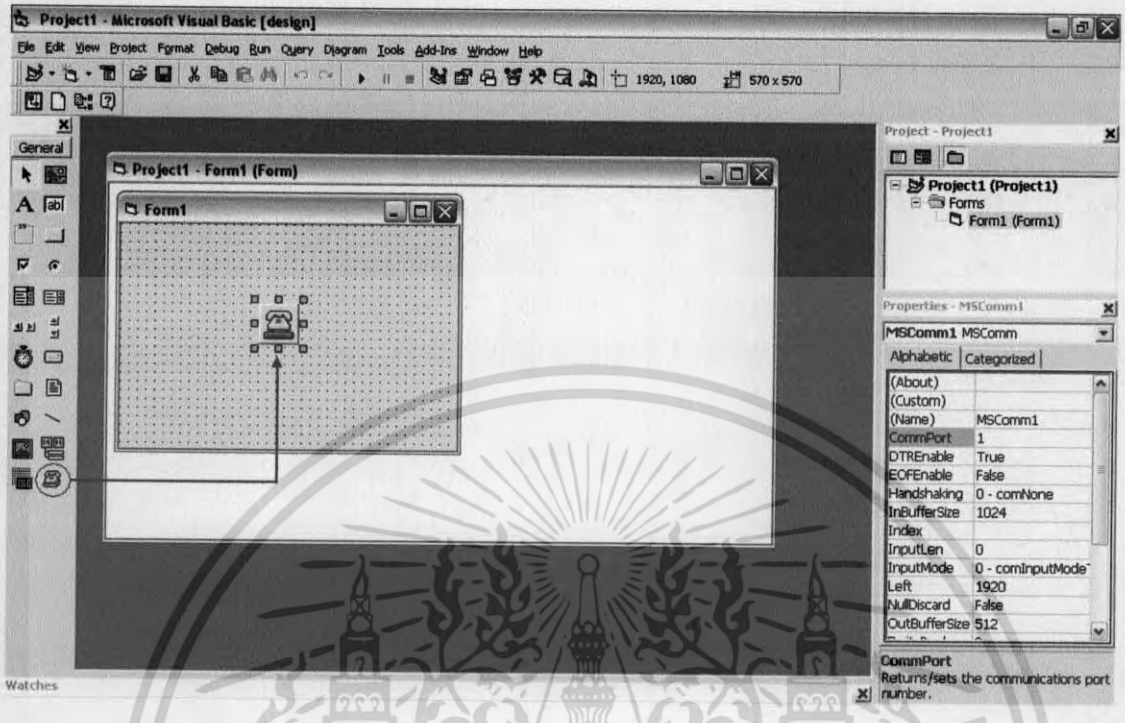
(ก)

2. เลือกชื่อ Control ชื่อ Microsoft Comm Control 6.0 แล้วคลิก OK



(ข)

3. ลาก Control ชื่อ Microsoft Comm จาก ToolBox มาวางไว้บน Form ดังรูปด้านล่าง



(ค)

รูปที่ 3.35 วิธีการติดต่อกับ Serial Port

3.7.1.1 การเขียนโปรแกรมติดต่อกับ Serial Port

สามารถทำได้ 2 วิธี คือ

1. การติดต่อแบบอินเตอร์รัพ

ขบวนการอินเตอร์รัพ อุปกรณ์รอบข้างเกือบทุกชิ้นจะต้องปฏิบัติงานอยู่เพื่อส่งสัญญาณไปให้แก่CPUเสมอ ถ้าอุปกรณ์นั้นพร้อมที่จะรับส่ง โดยในการเขียนโปรแกรมอินเตอร์รัพเมื่อมีข้อมูลเข้ามา ก็จะทำให้มี CommEvent เกิดขึ้นกับ OnCommevent

2. การติดต่อแบบโพลลิ่ง

เป็นการตรวจสอบข้อมูลตลอดเวลาการทำงานกับข้อมูลที่รับเข้ามา โดยเราจะใช้การตรวจสอบข้อมูลที่มาจาก Serial Port ตลอด โดยจะใช้ Control Timer เข้ามาช่วยในการเขียนโปรแกรมซึ่งสามารถตรวจสอบได้ถึงระดับ 1 มิลลิวินาที หรือจะใช้คำสั่ง Do.....Loop แทนก็ได้ซึ่งในโครงงานนี้ได้ใช้การติดต่อแบบอินเตอร์รัพ โดยที่ในตัวของคอนโทรล MSComm มี Event เพียง Eventเดียวเท่านั้นคือ OnComm Event ซึ่งจะใช้ในการติดต่อแบบอินเตอร์รัพ ซึ่งในการเขียนโปรแกรมติดต่อกับ Serial Port ในโครงงานนี้จะใช้ comEvent เพียง comEvReceive และ comEvSend เท่านั้น

3.7.1.2 องค์ประกอบในการใช้ MSComm

1. Property ชื่อ Commport คือ การเลือกคอมพอร์ตที่เราต้องการจะต่อใช้งาน โดยมีคำสั่งการใช้งานดังนี้

ตัวอย่าง MSComm1.Commport = 1

ในที่นี้เลือกใช้ Com1 ที่อยู่ด้านหลังเครื่องคอมพิวเตอร์

2. Property ชื่อ Settings คือ การตั้งค่าของการรับส่งข้อมูลซึ่งจะต้องรู้ด้วยว่าอัตราบอร์ดของอุปกรณ์ที่จะติดต่อด้วยเป็นเท่าไร โดยมีรายละเอียดการใส่ค่าต่างๆดังนี้

MSComm1.Setting = "อัตราการรับส่งข้อมูล,Parity(ถ้าไม่ใช้ใส่ N),จำนวนบิตข้อมูล,StopBit"

โดยมีคำสั่งการใช้งานดังนี้

ตัวอย่าง MSComm1.Settings = "9600,N,8,1"

3. Property ชื่อ InputLen คือ การกำหนดขนาดของข้อมูลที่ให้ไปอ่านจากข้อมูลที่อยู่ในบัฟเฟอร์ภาครับ โดยมีคำสั่งการใช้งานดังนี้

ตัวอย่าง MSComm1.InputLen = 1

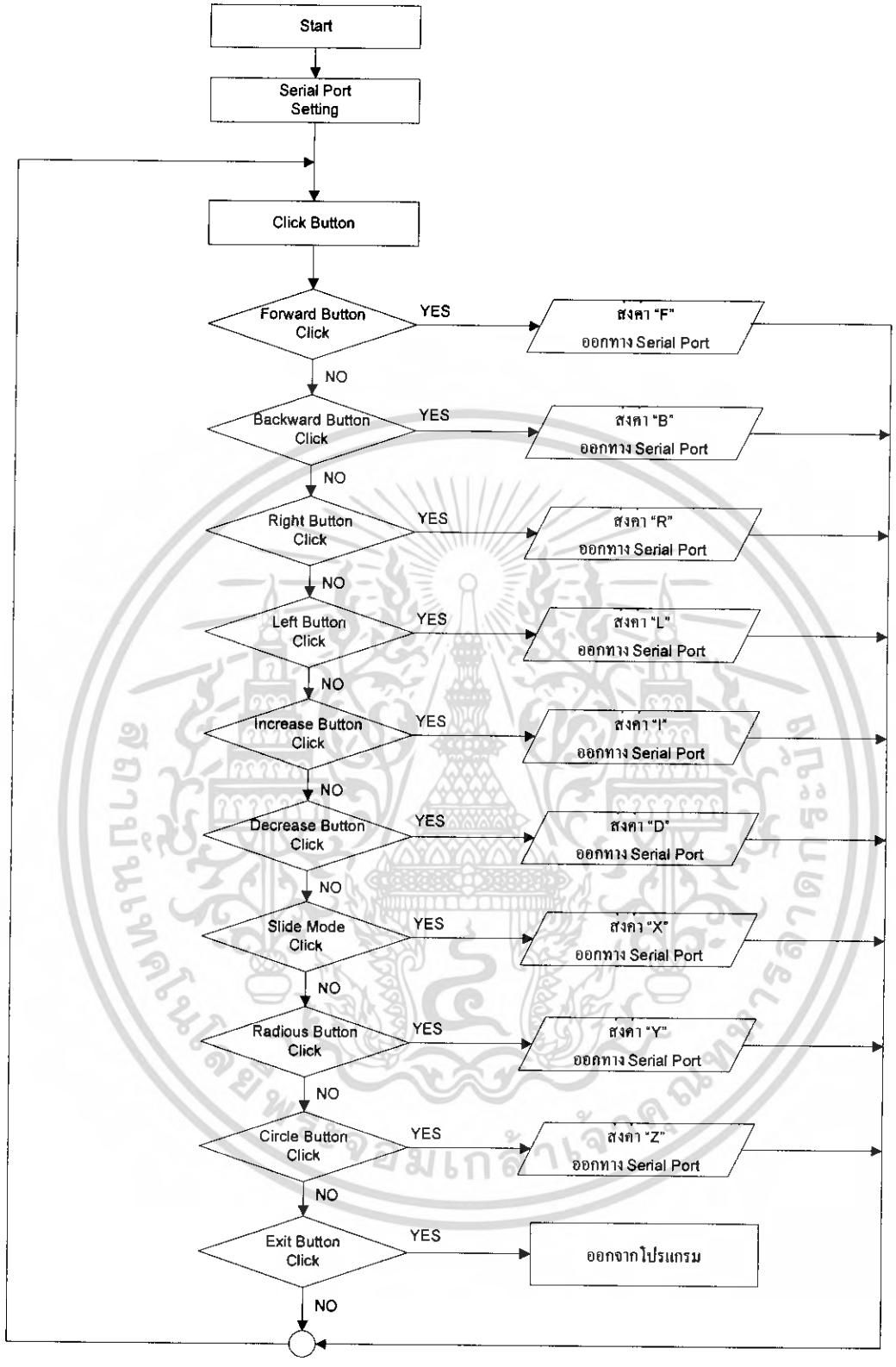
4. Property ชื่อ PortOpen คือ การกำหนดให้มีการเปิดใช้งานของพอร์ตอนุกรมหรือไม่ ถ้าเปิด = True แต่ถ้าปิด = False โดยมีคำสั่งการใช้งานดังนี้

ตัวอย่าง MSComm1.PortOpen =True

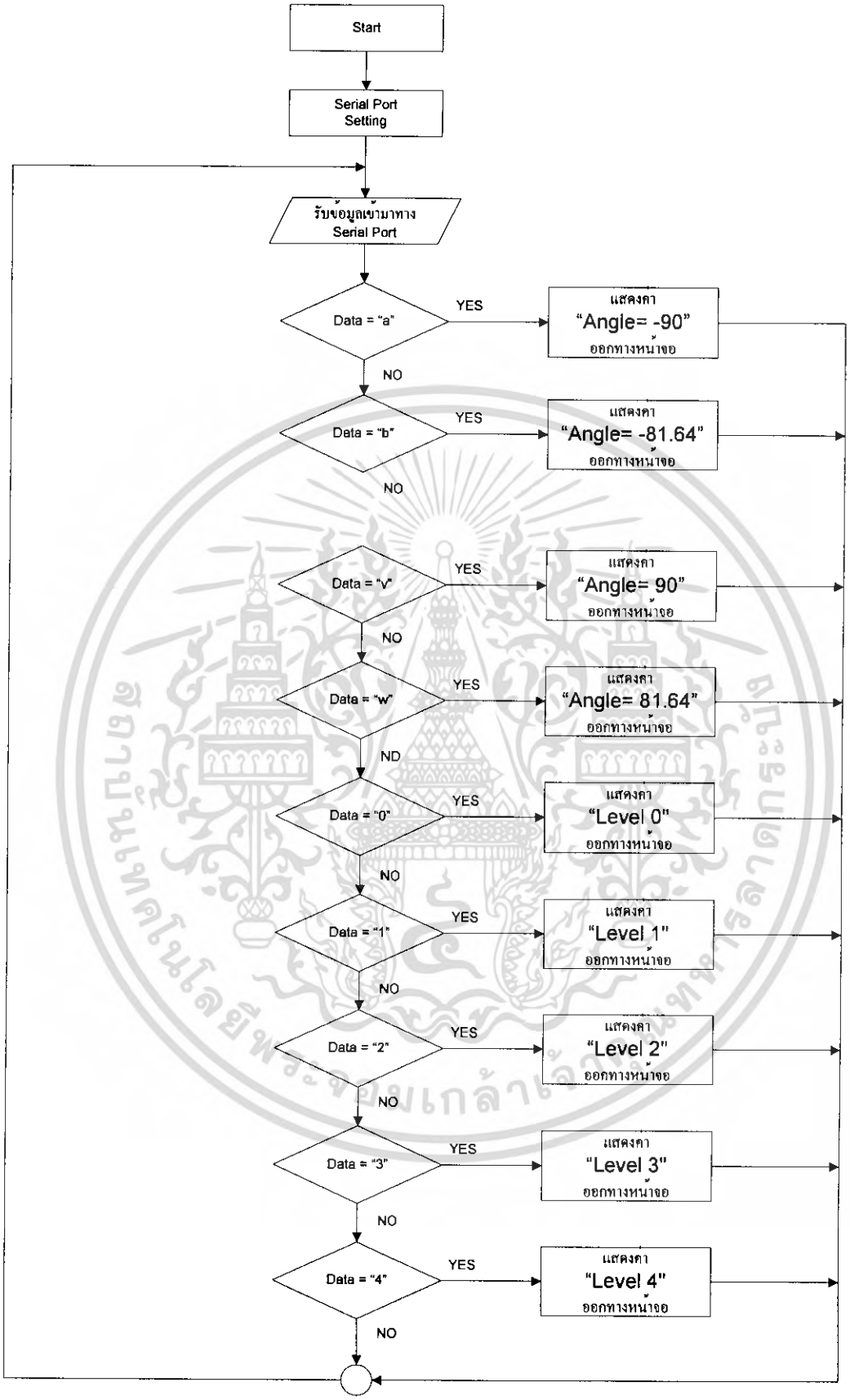
5. Property ชื่อ Rthreshold คือ ทำให้เกิดการกระตุ้นด้วย Even-driven เมื่อมีข้อมูลเข้ามาในบัฟเฟอร์ภาครับ ทำให้เกิด CommEvent ใน OnCommEvent โดยมีคำสั่งการใช้งานดังนี้

ตัวอย่าง MSComm1.Rthreshold = 1

ในโครงงานนี้ได้นำโปรแกรม Visual Basic มาใช้ในการติดต่อกับไมโครคอนโทรลเลอร์โดยผ่านพอร์ตอนุกรม (RS-232) โดยจะต้องเขียนโปรแกรมคำสั่งกำหนดในไมโครคอนโทรลเลอร์ว่าถ้ารับค่าจาก Serial Port มาแล้วจะให้ทำอะไรเพื่อที่เวลาเราส่งค่าไปที่ไมโครคอนโทรลเลอร์จะได้รู้จักคำสั่งว่าจะให้ตัวไมโครคอนโทรลเลอร์ทำอะไรหรือให้ส่งค่าอะไรกลับมา ซึ่งโปรแกรมที่ใช้ในการติดต่อกับไมโครคอนโทรลเลอร์เพื่อใช้ในการบังคับหุ่นยนต์ในโครงงานนี้มีหลักการเขียนง่ายๆซึ่งสามารถอธิบายได้ด้วย Flow Chat ดังนี้

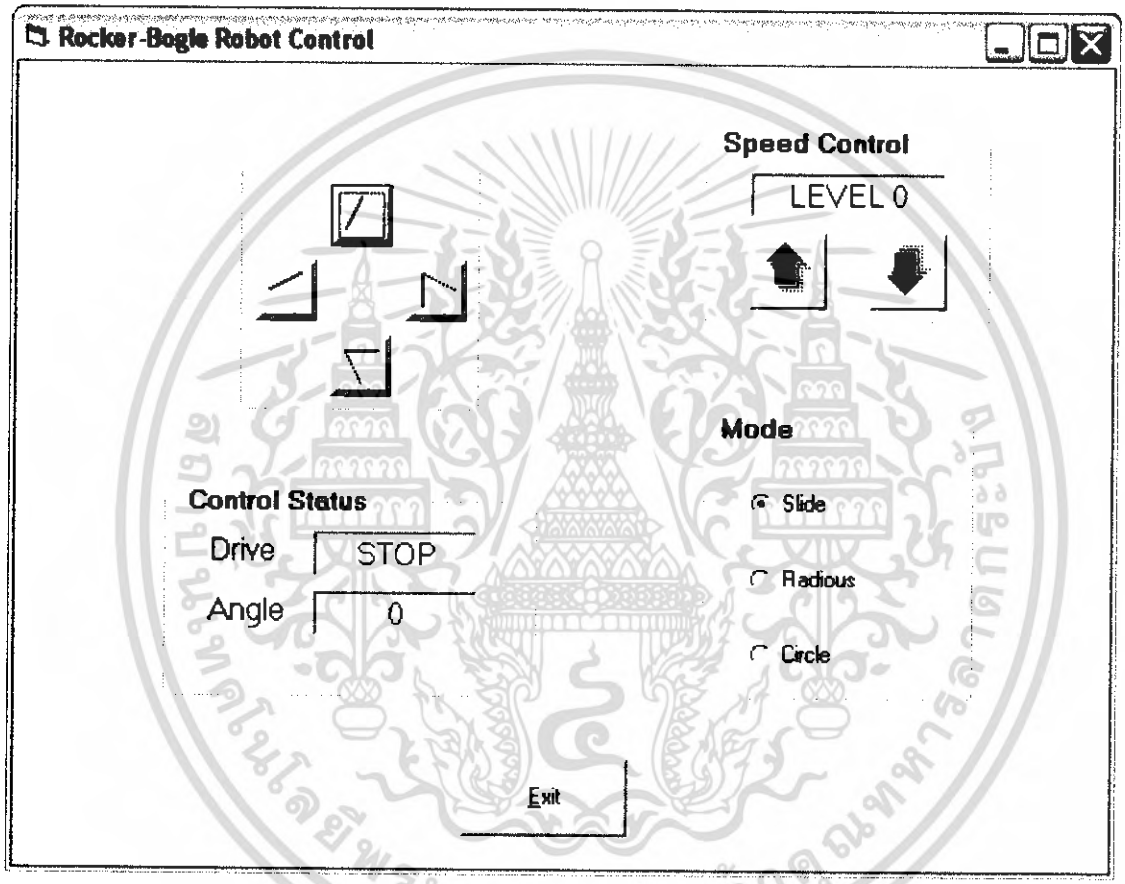


รูปที่ 3.36 Flowchart การส่งข้อมูลให้กับไมโครคอนโทรลเลอร์



รูปที่ 3.37 Flowchart การรับข้อมูลจากไมโครคอนโทรลเลอร์

ซึ่งโปรแกรมในส่วนของภาครับนั้นจะเป็นตัวแสดงการยืนยันจากไมโครคอนโทรลเลอร์ว่าได้กระทำตามคำสั่งที่ผู้ใช้งานสั่งให้กระทำแล้ว เช่น การเลี้ยวของหุ่นยนต์เมื่อมีการสั่งให้เกิดการเลี้ยวนั้นเมื่อไมโครคอนโทรลเลอร์ได้รับคำสั่งและสั่งให้หุ่นเกิดการเลี้ยวแล้วก็จะทำการส่งคืนค่ากลับมาหลังจากที่ไมโครคอนโทรลเลอร์สั่งให้หุ่นยนต์เกิดการเลี้ยวไปแล้ว ซึ่งค่าที่ส่งกลับมานี้จะถูกนำมาแบ่งแยกเป็นกรณีเพื่อนำมาแสดงเป็นแสดงค่ามุมให้รู้ต่อไป ซึ่งเป็นการบ่งบอกให้รู้ว่าหุ่นยนต์เกิดการทำงานขึ้นจริงตามที่ผู้ควบคุมบังคับ



รูปที่ 3.38 หน้าจอโปรแกรมที่ใช้ในการบังคับหุ่นยนต์ Rocker-Bogie Robot

3.7.1.3 วิธีการใช้โปรแกรม Rocker-Bogie Robot Control

ใช้ในการสั่งให้หุ่นยนต์เคลื่อนที่ไปข้างหน้า (จะเคลื่อนที่ได้เมื่อมีความเร็วในการเคลื่อนที่อย่างน้อย Level1)

ใช้ในการสั่งให้หุ่นยนต์เคลื่อนที่ไปข้างหลัง (จะเคลื่อนที่ได้เมื่อมีความเร็วในการเคลื่อนที่อย่างน้อย Level1)

ใช้ในการสั่งให้หุ่นยนต์เลี้ยวไปทางขวา

ใช้ในการสั่งให้หุ่นยนต์เลี้ยวไปทางซ้าย

ใช้ในการเพิ่มความเร็วในการเคลื่อนที่ของหุ่นยนต์

ใช้ในการลดความเร็วในการเคลื่อนที่ของหุ่นยนต์

◉ Slide เลือกโหมดการเลี้ยวแบบ Slide (ไปทางด้านข้าง)

◉ Radious เลือกโหมดการเลี้ยวแบบรัศมี

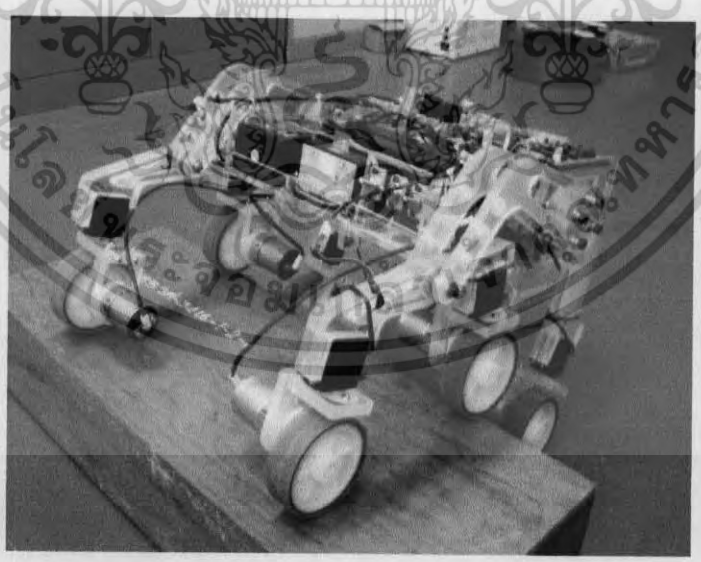
◉ Circle เลือกโหมดการเลี้ยวแบบเป็นวงกลมรอบจุดศูนย์กลางหุ่นยนต์

บทที่ 4

การทดลอง

4.1 ขั้นตอนการทดลอง

การทดลองนี้เป็นการทดสอบสมรรถนะของหุ่นยนต์ rocker-bogie หลังจากสร้างเสร็จว่ามีความสามารถเพียงใดในการที่จะแล่นผ่านอุปสรรคต่างๆ โดยทำการทดลองทั้งหมด 4 การทดลอง โดยการทดลองแรกเริ่มจากการวัดค่าความเร็วที่ตัวหุ่นยนต์สามารถแล่นได้ในแต่ละเกียร์ในสภาพพื้นผิวที่แตกต่างกัน คือ พื้นยาง พื้นคอนกรีต พื้นหญ้า และพื้นทราย การทดลองที่ 2 พิจารณาแรงกดที่เกิดขึ้นที่ล้อทั้ง 6 เมื่อแล่นผ่านพื้นที่มีความเอียงต่างๆ กันออกไปโดยการไต่พื้นเอียงนั้นเริ่มที่ระดับ 5 องศา แล้วค่อยๆเพิ่มระดับความเอียงไปเรื่อยๆแล้วจึงวิเคราะห์แรงที่เกิดขึ้นแต่ละล้อจนกว่าตัวรถจะเกิดการลอยและพลิกคว่ำหรือไม่สามารถที่จะขับเคลื่อนไปต่อได้ นั่นหมายถึงประสิทธิภาพในการแล่นผ่านพื้นเอียงของหุ่นยนต์ rocker-bogie จากนั้นการทดลองที่ 3 จะทำการแล่นข้ามผ่านสิ่งกีดขวางที่มีลักษณะตั้งฉากและมีความสูงมากกว่าครึ่งล้อว่าความสูงที่สูงที่สุดที่หุ่นสามารถข้ามผ่านไปได้เป็นเท่าใด และการทดลองสุดท้ายจะทำการนำตัวหุ่นยนต์ลองทำการแล่นผ่านพื้นที่จริงที่มีสภาวะที่ค่อนข้างขรุขระและมีอุปสรรคต่างๆนานาเป็นขั้นตอนสุดท้าย



รูปที่ 4.1 หุ่นยนต์ rocker-bogie เมื่อเสร็จสมบูรณ์

4.2 ผลการทดลอง

4.2.1 การทดลองที่ 1

การทดลองนี้จะทำการวัดความเร็วแต่ละเกียร์ของหุ่นยนต์ที่ที่สามารถวิ่งไปได้ในแต่ละสภาพพื้นผิว โดยทำการจับเวลา 10 วินาที แล้วทำการวัดระยะทางที่รถวิ่งไปได้ ทำการทดลองเกียร์ละ 3 ครั้ง นำมาหาค่าเฉลี่ย แล้วแปลงให้อยู่ในรูป เมตร/วินาที

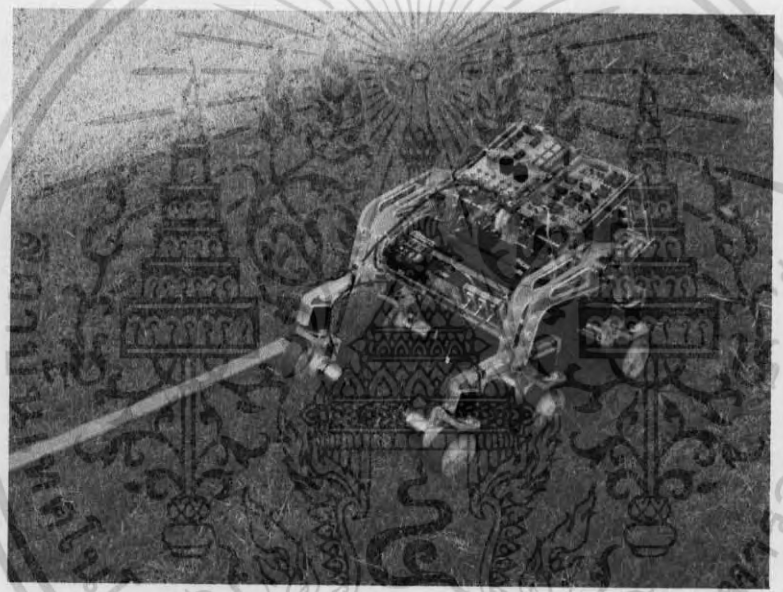
ตารางที่ 4.1 ความเร็วในแต่ละเกียร์ที่หุ่นยนต์วิ่งได้บนพื้นยาง

GEAR	ครั้งที่ (cm/10 sec)			ความเร็วเฉลี่ย (cm/10 sec)	ความเร็ว (m/s)
	1	2	3		
1	11.5	12	12	11.83	0.012
2	24.5	23.5	23.5	23.83	0.024
3	37.5	36	36.5	36.66	0.037
4	43	45.5	45.5	44.66	0.045

รูปที่ 4.2 พื้นยาง

ตารางที่ 4.2 ความเร็วในแต่ละเกียร์ที่หุ่นยนต์วิ่งได้บนพื้นหญ้า

GEAR	ครั้งที่ (cm/10 sec)			ความเร็วเฉลี่ย (cm/10 sec)	ความเร็ว (m/s)
	1	2	3		
1	10	9.5	10	9.83	0.010
2	23	22.5	22	22.5	0.023
3	34.5	34	35.5	34.66	0.035
4	41	40.5	40.5	40.66	0.041



รูปที่ 4.3 พื้นสนามหญ้า

ตารางที่ 4.3 ความเร็วในแต่ละเกียร์ที่หุ่นยนต์วิ่งได้บนพื้นคอนกรีต

GEAR	ครั้งที่ (cm/10 sec)			ความเร็วเฉลี่ย (cm/10 sec)	ความเร็ว (m/s)
	1	2	3		
1	12	11.5	12	11.83	0.012
2	24.5	24.5	24.5	24.5	0.025
3	37.5	37	37.5	37.33	0.037
4	45	45	45	45.00	0.045

รูปที่ 4.4 พื้นคอนกรีต

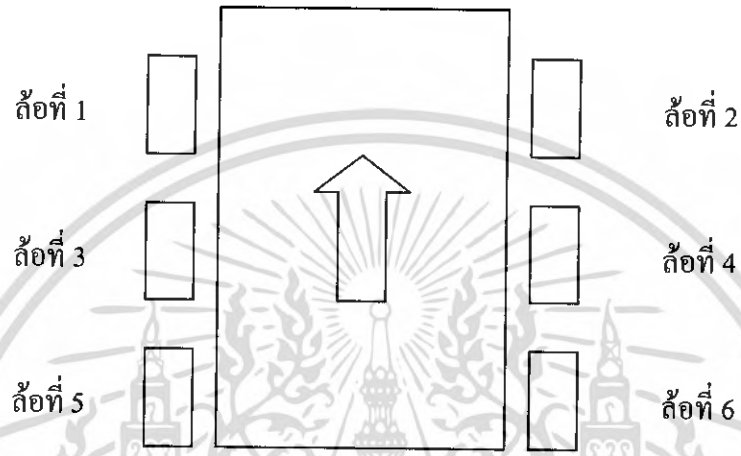
ตารางที่ 4.4 ความเร็วในแต่ละเกียร์ที่หุ่นยนต์วิ่งได้บนพื้นทราย

GEAR	ครั้งที่ (cm/10 sec)			ความเร็วเฉลี่ย (cm/10 sec)	ความเร็ว (m/s)
	1	2	3		
1	10.5	10.5	10	11.33	0.011
2	23	24	23	23.33	0.023
3	36.5	36	37	36.50	0.037
4	42	42	42.5	42.16	0.042

รูปที่ 4.5 พื้นทราย

4.2.2 การทดลองที่ 2

การทดลองนี้จะทำการทดลองแล่นผ่านพื้นเอียงที่มีระดับความเอียงของพื้นที่แตกต่างกัน โดยเริ่มที่ความเอียงที่มุม 5 องศา จากนั้นค่อยๆ เพิ่มมุมเอียงทีละ 5 องศาไปเรื่อยๆ แล้วทำการบันทึกค่าแรงที่เกิดขึ้นแต่ละล้อแล้วดูว่าที่มุมเอียงสูงสุดเท่าใดที่หุ่นยนต์สามารถแล่นผ่านได้



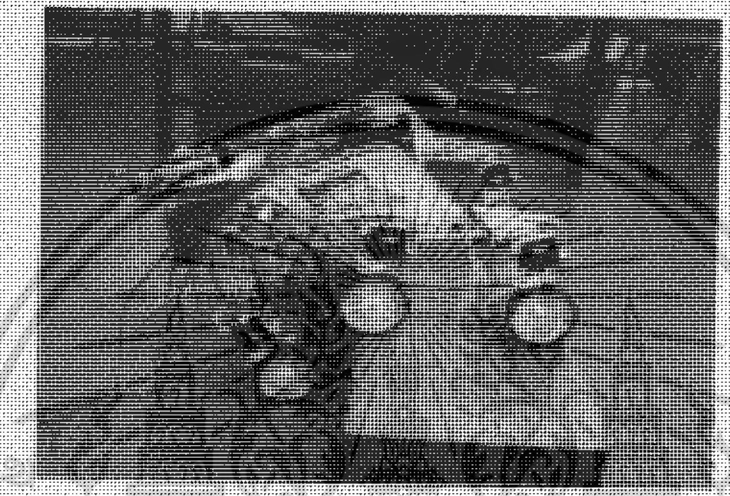
รูปที่ 4.6 ตำแหน่งอ้างอิงของแต่ละล้อ

ตารางที่ 4.5 แรงกด (N) ที่เกิดขึ้นแต่ละล้อเมื่อทำการแล่นผ่านพื้นที่มีมุมเอียงต่างๆกัน

มุม (°)	ล้อที่ 1	ล้อที่ 2	ล้อที่ 3	ล้อที่ 4	ล้อที่5	ล้อที่ 6
0	25.12	25.03	25.09	25.34	24.87	24.93
5	20.34	21.33	24.88	25.51	28.22	28.96
10	13.26	13.51	24.03	24.77	38.65	38.31
15	9.87	10.31	23.14	23.21	45.61	45.56
20	6.14	5.86	22.58	23.43	45.42	45.34
25	3.39	3.07	19.16	20.55	45.98	45.02
30	0.68	0.91	12.37	11.43	45.23	45.02
35	0.00	0.00	4.92	6.13	45.86	45.44
40	0.00	0.00	0.00	0.00	45.67	45.33

4.2.3 การทดลองที่ 3

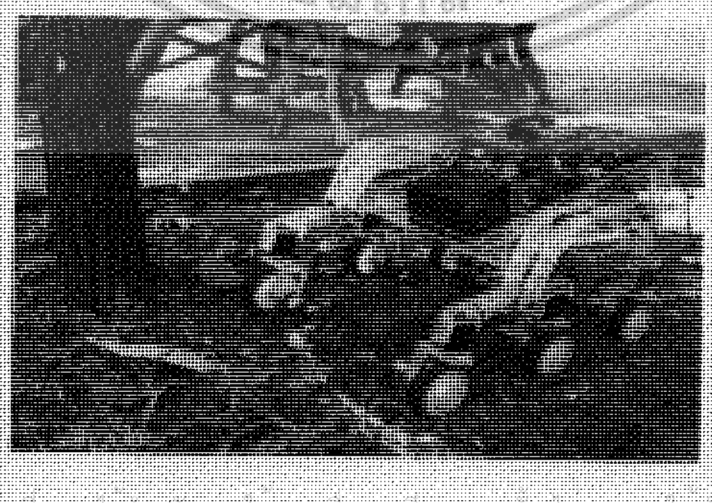
การทดลองนี้จะทำการให้หุ่นยนต์ข้ามสิ่งกีดขวางที่มีความสูงเกินครึ่งล้อว่ามีประสิทธิภาพในการปีนป่ายได้มากน้อยเพียงใด โดยเริ่มที่ความสูง 4 เซนติเมตร ซึ่งเป็นความสูงเท่ากับครึ่งล้อของหุ่นยนต์ rocker-bogie จากนั้นค่อย ๆ เพิ่มความสูงของสิ่งกีดขวางขึ้นทีละ 1 เซนติเมตร แล้วดูว่าหุ่นยนต์สามารถข้ามไปได้หรือไม่



รูปที่ 4.7 การให้ข้ามสิ่งกีดขวางของหุ่นยนต์ rocker-bogie

4.2.4 การทดลองที่ 4

การทดลองนี้เราได้ทำการนำหุ่นยนต์ rocker-bogie ไปแล่นในภูมิประเทศจริงที่มีหลุมกรวด และความขรุขระต่าง ๆ แล้วสังเกตและวิเคราะห์ การเคลื่อนที่ของตัวหุ่นยนต์ว่ามีการเคลื่อนที่เป็นอย่างไร ล้อมีการสัมผัสพื้นได้ดีหรือไม่ สามารถหลบหลีกสิ่งกีดขวางที่ไม่สามารถผ่านไปได้ได้ดีเพียงใด



รูปที่ 4.8 การวิ่งในสภาพภูมิประเทศจริงของหุ่นยนต์ rocker-bogie

บทที่ 5
สรุปผลการทดลอง ปัญหา และข้อเสนอแนะ

5.1 สรุปผลการทดลอง

5.1.1 สรุปผลการทดลองที่ 1

การที่วิ่งไปบนพื้นที่มีสภาวะที่แตกต่างกันนั้นพบว่าความเร็วที่ได้มีความแตกต่างกัน โดยที่เมื่อทำการวิ่งไปบนพื้นยาง และพื้นคอนกรีตความเร็วที่ได้จะมีค่ามากคือ ความเร็วสูงที่สุดอยู่ที่ประมาณ 0.045 m/s ส่วนพื้นหญ้า และพื้นทราย ความเร็วที่ได้จะลดไปเล็กน้อยเนื่องจากความหนืดของพื้นที่สัมผัสกับตัวล้อ ความเร็วสูงสุดจะอยู่ที่ 0.041 และ 0.042 m/s ตามลำดับ การที่จะวิ่งให้ได้ดีที่สภาพพื้นผิวต่างๆให้ได้ดีนั้นจะต้องทำการปรับปรุงที่แรงเสียดทานบนล้อ คือทำการเปลี่ยนดอกยาง หรือวัสดุที่ทำล้อให้มีความฝืดกับพื้นผิวที่สัมผัส เช่น ถ้าเป็นทรายก็ควรเพิ่มดอกยางให้มีความลึก เพื่อเนื้อทรายจะได้ผ่านล้อได้สะดวกโดยไม่มีการหน่วงล้อเอาไว้ เป็นต้น

5.1.2 สรุปผลการทดลองที่ 2

การวิ่งไปบนพื้นเอียงของหุ่นยนต์ Rocker-bogie ทำได้ค่อนข้างดีมากทั้งนี้ขึ้นอยู่กับกำลังขับของมอเตอร์ ของมอเตอร์ที่ใช้ และ Balance น้ำหนักของตัวหุ่นยนต์ การทดลองนี้ตัวหุ่นยนต์พลิกคว่ำที่มุม 40 องศา เนื่องจากน้ำหนักทิ้งไปเกินล้อด้านหลัง และกำลังมอเตอร์ไม่สามารถขับน้ำหนักที่ทิ้งมาทั้งหมดของตัวรถได้ รถจะหยุดและพลิกหงาย การแก้ไขทำได้โดย อาจเพิ่มกำลังของมอเตอร์ และ Balance น้ำหนักใหม่ให้มีโมเมนต์มากขึ้น หรือเพิ่มความยาวของฐานรถให้น้ำหนักตกไม่ให้เกินล้อด้านหลัง

5.1.3 สรุปผลการทดลองที่ 3

การปีนข้ามอุปสรรคที่ตั้งฉากกับพื้นผิว หุ่นยนต์ Rocker-bogie ทำได้ดีมาก คือทำได้ที่ความสูงเกือบสองเท่าของล้อ ที่ 15.2 ซม. โดยที่ความสูงมากกว่านี้ ตัวหุ่นยนต์นั้นจะไม่สามารถขับดันล้อหลังให้ข้ามไปได้ การแก้ไขทำได้โดยเพิ่มกำลังของมอเตอร์ และ Balance น้ำหนักให้มีความสมดุล

5.1.4 สรุปผลการทดลองที่ 4

การนำไปลงพื้นที่จริงพบว่าค่อนข้างเคลื่อนที่ไปได้ด้วยดี และข้ามอุปสรรคเป้าหมายที่ต้องการได้อย่างดีแต่มีบางอุปสรรคที่ต้องหลีกเลี่ยง คือ ที่ที่มีหินสูงต่อเนื่องกันเป็นทอดๆ ซึ่งจะทำให้ล้อไม่สามารถสัมผัสพื้นได้ ทำให้หุ่นยนต์เกิดการล้มและพลิกคว่ำได้

บรรณานุกรม

1. "Mars Pathfinder" : www.mpf.jpl.nasa.gov\MPF
2. "Sojourner" : http://mpf.jpl.nasa.gov\MPF\rover\sojourner.html
3. Harcot Herve , "Analysis and Simulation of a Exploration Rover",M.S. Thesis , Massachusetts Instutute of Technology , Cambridge , MA , 1998.
4. นคร ภักดีชาติ , ธีรบูลย์ หล่อวิเชียรรุ่ง , ชัยวัฒน์ ลิ้มพรจิตรวิไล , "ปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 ด้วยโปรแกรมภาษาซี" , Innovative Experiment Press.
5. ไสว พงศ์สวัสดิ์ , "อิเล็คทรอนิคส์กำลัง (power electronics)" , ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
6. ฉันทวุฒิ พืชผล , พิชิต สันติกุลานนท์ , พร้อมเลิศ หล่อวิจิตร , "คู่มือเรียน Visual Basic 6" , Provision
7. Engineering Innovator Club : http://www.eng.chula.ac.th/~merobot
8. "Rocker Bogie Robot" , Washington State University : http://www.wsu.edu
9. ThaiIO : http://www.thaiio.com

ภาคผนวก

ภาคผนวก ก

Source Code

Source Code ของการติดต่อโปรแกรมผ่านพอร์ตอนุกรม

```
Option Explicit
Public a As Integer
Public Data As String

Private Sub Form_Load()
On Error Resume Next
MSCom1.CommPort = 1
MSCom1.InputLen = 1
MSCom1.Settings = "9600 , n ,8,1"
MSCom1.PortOpen = True
MSCom1.RThreshold = 1
optSlide.Value = True
End Sub

Private Sub cmdForward_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
On Error Resume Next
If Button = vbLeftButton Then
   MSCom1.Output = "F"
   lblDrive.Caption = "Forward"
End If
End Sub

Private Sub cmdForward_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)
On Error Resume Next
If Button = vbLeftButton Then
   MSCom1.Output = "S"
   lblDrive.Caption = "Stop"
End If
End Sub

Private Sub cmdBack_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
On Error Resume Next
If Button = vbLeftButton Then
   MSCom1.Output = "B"
   lblDrive.Caption = "Backward"
End If
End Sub

Private Sub cmdBack_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
On Error Resume Next
If Button = vbLeftButton Then
   MSCom1.Output = "S"
   lblDrive.Caption = "Stop"
End If
End Sub

Private Sub cmdRight_ MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
On Error Resume Next
If Button = vbLeftButton Then
   MSCom1.Output = "R"
End If
End Sub

Private Sub cmdRight_ MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
On Error Resume Next
If Button = vbLeftButton Then
   MSCom1.Output = "L"
End If
End Sub

Private Sub cmdIncrease_click()
On Error Resume Next
MSCom1.Output = "I"
End Sub

Private Sub cmdDecrease_click()
On Error Resume Next
MSCom1.Output = "D"
End Sub

Private Sub optSlide_click()
On Error Resume Next
MSCom1.Output = "X"
End Sub

Private Sub optRadious_click()
On Error Resume Next
MSCom1.Output = "Y"
End Sub

Private Sub optCircle_click()
On Error Resume Next
MSCom1.Output = "Z"
End Sub

Private Sub MSCom1_OnComm()
On Error Resume Next
Select Case MSCom1.CommEvent
  Case comEvReceive
    Data = MSCom1.Input
    MSCom1.InBufferCount = 0
      Select Case Data
        Case "a"
          lblAngle.Caption = "-89.8"
        Case "b"
          lblAngle.Caption = "-81.64"
        Case "c"
          lblAngle.Caption = "-73.48"
        Case "d"
          lblAngle.Caption = "-65.32"
        Case "e"
          lblAngle.Caption = "-57.16"
        Case "f"
          lblAngle.Caption = "-49"
        Case "g"
          lblAngle.Caption = "-40.84"
        Case "h"
          lblAngle.Caption = "-32.68"
        Case "i"
          lblAngle.Caption = "-24.52"
        Case "j"
          lblAngle.Caption = "-16.36"
        Case "k"
          lblAngle.Caption = "-8.18"
        Case "l"
          lblAngle.Caption = "0"
```

```
        Case "m"
          lblAngle.Caption = "8.18"
        Case "n"
          lblAngle.Caption = "16.36"
        Case "o"
          lblAngle.Caption = "24.52"
        Case "p"
          lblAngle.Caption = "32.68"
        Case "q"
          lblAngle.Caption = "40.84"
        Case "r"
          lblAngle.Caption = "49"
        Case "s"
          lblAngle.Caption = "57.16"
        Case "t"
          lblAngle.Caption = "65.32"
        Case "u"
          lblAngle.Caption = "73.46"
        Case "v"
          lblAngle.Caption = "81.64"
        Case "w"
          lblAngle.Caption = "89.8"
        Case "0"
          lblLevel.Caption = "LEVEL 0"
        Case "1"
          lblLevel.Caption = "LEVEL 1"
        Case "2"
          lblLevel.Caption = "LEVEL 2"
        Case "3"
          lblLevel.Caption = "LEVEL 3"
        Case "4"
          lblLevel.Caption = "LEVEL 4"
        Case "X"
          lblMode.Caption = "Slide"
        Case "Y"
          lblMode.Caption = "Radius"
        Case "Z"
          lblMode.Caption = "Circle"
      End Select
End Select
End Sub

Private Sub from_keyDown(KeyCode As Integer, Shift As Integer)
On Error Resume Next
If a = 0 Then
  Select Case KeyCode
    Case vbKeyW
      MSCom1.Output = "F"
      lblDrive.Caption = "Forward"
    Case vbKeyS
      MSCom1.Output = "B"
      lblDrive.Caption = "Backward"
    Case vbKeyD
      MSCom1.Output = "R"
    Case vbKeyA
      MSCom1.Output = "L"
  End Select
End If
a = 1
End Sub

Private Sub from_keyUp(KeyCode As Integer, Shift As Integer)
On Error Resume Next
a = 1
Select Case KeyCode
    Case vbKeyW
      MSCom1.Output = "S"
      lblDrive.Caption = "Stop"
    Case vbKeyS
      MSCom1.Output = "S"
      lblDrive.Caption = "Stop"
  End Select
End Sub

Private Sub cmdExit_click()
End
End Sub
```

Source Code For MCS-51

```
#include<reg51rx.h>
#include<I2C.h>
#include<stdio.h>
////////////////////////////////////////////////////////////////////////////
//DECLARATIONS FOR GLOBAL
////////////////////////////////////////////////////////////////////////////

unsigned char input,mode;
unsigned char index_speed,index_servo;
bit enable;
sbit p1_0=P1^0;
sbit p1_1=P1^1;
sbit p1_2=P1^2;
sbit move_forward=P2^2;
sbit move_backward=P2^3;
sbit relay=P2^4;
////////////////////////////////////////////////////////////////////////////
//DECLARATIONS FOR SLIDE MODE
////////////////////////////////////////////////////////////////////////////

code unsigned char
pwm_slide[23]={248,247,246,245,244,243,242,241,240,239,238,237,236,235,234,233,232,231,2
30,229,228,227,226};
unsigned char speed_slide[5]={127,159,191,223,255};

////////////////////////////////////////////////////////////////////////////
//DECALRATIONS FOR RADIUS MODE
////////////////////////////////////////////////////////////////////////////

unsigned char index_servo_radius_left_2wheelrear,index_servo_radius_right_2wheelrear;
bit speed_radius_forward,speed_radius_backward;
code unsigned char zeta2[23]={0,0,0,0,0,0,0,0,245,241,238,237,236,233,229,0,0,0,0,0,0,0,0};
////////////////////////////////////////////////////////////////////////
//FORWARD GEAR1
code unsigned char speed_radius_outer_gear1_forward[4]={128,128+32,128+32,128+32};
code unsigned char speed_radius_inner_gear1_forward[4]={128,128+24,128+18,128+15};
code unsigned char
speed_radius_center_outer_gear1_forward[4]={128,128+32,128+31,128+29};
code unsigned char speed_radius_center_inner_gear1_forward[4]={128,128+24,128+16,128+7};
//FORWARD GEAR2
code unsigned char speed_radius_outer_gear2_forward[4]={128,128+64,128+64,128+64};
code unsigned char speed_radius_inner_gear2_forward[4]={128,128+49,128+36,128+30};
code unsigned char
speed_radius_center_outer_gear2_forward[4]={128,128+63,128+61,128+58};
code unsigned char
speed_radius_center_inner_gear2_forward[4]={128,128+48,128+31,128+14};
//FORWARD GEAR3
code unsigned char speed_radius_outer_gear3_forward[4]={128,128+96,128+96,128+96};
```

```
code unsigned char speed_radius_inner_gear3_forward[4]={128,128+73,128+54,128+45};
code unsigned char
speed_radius_center_outer_gear3_forward[4]={128,128+95,128+92,128+87};
code unsigned char
speed_radius_center_inner_gear3_forward[4]={128,128+72,128+47,128+20};
//FORWARD GEAR4
code unsigned char speed_radius_outer_gear4_forward[4]={128,128+127,128+127,128+127};
code unsigned char speed_radius_inner_gear4_forward[4]={128,128+97,128+72,128+60};
code unsigned char
speed_radius_center_outer_gear4_forward[4]={128,128+127,128+123,128+116};
code unsigned char
speed_radius_center_iuner_gear4_forward[4]={128,128+96,128+62,128+127};
////////////////////////////////////////////////////////////
//BACKWARD GEAR1
code unsigned char speed_radius_outer_gear1_backward[4]={128,128-32,128-32,128-32};
code unsigned char speed_radius_inner_gear1_backward[4]={128,128-24,128-18,128-15};
code unsigned char speed_radius_center_outer_gear1_backward[4]={128,128-32,128-31,128-
29};
code unsigned char speed_radius_center_inner_gear1_backward[4]={128,128-24,128-16,128-7};
//BACKWARD GEAR2
code unsigned char speed_radius_outer_gear2_backward[4]={128,128-64,128-64,128-64};
code unsigned char speed_radius_inner_gear2_backward[4]={128,128-49,128-36,128-30};
code unsigned char speed_radius_center_outer_gear2_backward[4]={128,128-63,128-61,128-
58};
code unsigned char speed_radius_center_inner_gear2_backward[4]={128,128-48,128-31,128-
14};
//BACKWARD GEAR3
code unsigned char speed_radius_outer_gear3_backward[4]={128,128-96,128-96,128-96};
code unsigned char speed_radius_inner_gear3_backward[4]={128,128-73,128-54,128-45};
code unsigned char speed_radius_center_outer_gear3_backward[4]={128,128-95,128-92,128-
87};
```

```
code unsigned char speed_radius_center_inner_gear3_backward[4]={128,128-72,128-47,128-
20};
//BACKWARD GEAR4
code unsigned char speed_radius_outer_gear4_backward[4]={128,128-128,128-128,128-128};
code unsigned char speed_radius_inner_gear4_backward[4]={128,128-97,128-72,128-60};
code unsigned char speed_radius_center_outer_gear4_backward[4]={128,128-127,128-123,128-
116};
code unsigned char speed_radius_center_iuner_gear4_backward[4]={128,128-96,128-62,128-
27};
////////////////////////////////////////////////////////////
//TEMPOLARY ARRAY
unsigned char temp_speed_radius_outer[4]={0,0,0,0};
unsigned char temp_speed_radius_inner[4]={0,0,0,0};
unsigned char temp_speed_radius_center_outer[4]={0,0,0,0};
unsigned char temp_speed_radius_center_inner[4]={0,0,0,0};

////////////////////////////////////////////////////////////////////////
//DECLARATIONS FOR CIRCLE MODE
////////////////////////////////////////////////////////////////////////

bit speed_circle_forward,speed_circle_backward;
unsigned char
speed_circle_outer_forward,speed_circle_outer_backward,speed_circle_inner_forward,speed_cir
cle_inner_backward;

////////////////////////////////////////////////////////////////////////
//
        DECLARATION FOR FUNCTION SEND_TORQUE
/
////////////////////////////////////////////////////////////////////////

unsigned char torque1,torque2,torque3,torque4,torque5,torque6;
```

```
float Kt=1;//constant Kt Nm.
float    mew=1;
//float gain=2.5;
float display1,display2,display3,display4,display5,display6;
float temp1,temp2,temp3,temp4,temp5,temp6;

////////////////////////////////////////////////////////////////////////
//        END OF ALL DECLARATIONS
/
////////////////////////////////////////////////////////////////////////

unsigned char read_torque(unsigned char address)
{
        unsigned char temp;
        i2c_start();
        i2c_write(address);
        i2c_write(0x40);
        i2c_stop();
        i2c_start();
        i2c_write(address+1);
        temp = i2c_read();
        i2c_Nack();
        i2c_stop();
        return(temp);
}

void write_D2A(unsigned char address,unsigned char speed)
{
        i2c_start();
        i2c_write(address);
        i2c_write(0x40);
        i2c_write(speed);
```

```
        i2c_stop();
}

void stop(void)
{
        write_D2A(0x92,0x7F);
        write_D2A(0x94,0x7F);
        write_D2A(0x96,0x7F);
        write_D2A(0x98,0x7F);
        write_D2A(0x9A,0x7F);
        write_D2A(0x9C,0x7F);
}

void delay (void)
{
        unsigned char i,j;
        for(i=0;i<255;i++)
//      for(j=0;j<255;j++);
}

void delay_infared (void)//delay time = 0.003 sec
{
        unsigned char i,j;
        for(i=0;i<255;i++)
        for(j=0;j<21;j++);
}

void send_inform_speed(unsigned char inform_speed)
{
        switch(inform_speed)
        {
```

```
            case 0:SBUF='0';
                        break;
            case 1:SBUF='1';
                        break;
            case 2:SBUF='2';
                        break;
            case 3:SBUF='3';
                        break;
            case 4:SBUF='4';
                        break;
        }
        delay();
}

void send_inform_servo(unsigned char inform_servo)
{
        switch(inform_servo)
        {
            case  0:SBUF='a';
                        break;
            case  1:SBUF='b';
                        break;
            case  2:SBUF='c';
                        break;
            case  3:SBUF='d';
                        break;
            case  4:SBUF='e';
                        break;
            case  5:SBUF='f';
                        break;
            case  6:SBUF='g';
                        break;
            case  7:SBUF='h';
                        break;
            case  8:SBUF='i';
                        break;
            case  9:SBUF='j';
                        break;
            case 10:SBUF='k';
                        break;
            case 11:SBUF='l';
                        break;
            case 12:SBUF='m';
                        break;
            case 13:SBUF='n';
                        break;
            case 14:SBUF='o';
                        break;
            case 15:SBUF='p';
                        break;
            case 16:SBUF='q';
                        break;
            case 17:SBUF='r';
                        break;
            case 18:SBUF='s';
                        break;
            case 19:SBUF='t';
                        break;
            case 20:SBUF='u';
                        break;
            case 21:SBUF='v';
                        break;
            case 22:SBUF='w';
                        break;
        }
        delay();
}

void send_torque (void)
{
            RI=0;
            TI=1;

            torque1=read_torque(0x92);
            temp1=torque1;
            display1=(2*(temp1/256));
            printf("%f",display1);
            delay();

            torque2=read_torque(0x94);
            temp2=torque2;
            display2=(2*(temp2/256));
            printf("%f",display2);
            delay();

            torque3=read_torque(0x96);
            temp3=torque3;
            display3=(2*(temp3/256));
            printf("%f",display3);
            delay();

            torque4=read_torque(0x98);
            temp4=torque4;
            display4=(2*(temp4/256));
            printf("%f",display4);
            delay();

            torque5=read_torque(0x9A);
            temp5=torque5;
            display5=(2*(temp5/256));
            printf("%f",display5);
            delay();

            torque6=read_torque(0x9C);
            temp6=torque6;
            display6=(2*(temp6/256));
            printf("%f",display6);
            delay();

            RI=0;
            TI=0;

            torque1=0;
            torque2=0;
            torque3=0;
            torque4=0;
            torque5=0;
            torque6=0;
            temp1=0;
            temp2=0;
            temp3=0;
            temp4=0;
```

```
        temp5=0;
        temp6=0;
        display1=0;
        display2=0;
        display3=0;
        display4=0;
        display5=0;
        display6=0;
}

void service_serial() interrupt 4
{
    if(RI)
    {
        RI=0;
        input=SBUF;
        switch(input)
        {
            case 'X':mode=0x01;
                     break;
            case 'Y':mode=0x02;
                     break;
            case 'Z':mode=0x03;
                     break;
            case 'L':{
                        if(index_servo>0)
                        index_servo--;
                        delay_infared();
                        send_inform_servo(index_servo);
                     }
                     break;
            case 'R':{
                        if(index_servo<22)
                        index_servo++;
                        delay_infared();
                        send_inform_servo(index_servo);
                     }
                     break;
            case 'I':{
                        if(index_speed<4)
                        index_speed++;
                        delay_infared();
                        send_inform_speed(index_speed);
                     }
                     break;
            case 'D':{
                        if(index_speed>0)
                        index_speed--;
                        delay_infared();
                        send_inform_speed(index_speed);
                     }
                     break;
            case 'T':{
                        send_torque();
                        delay_infared();
                     }
                     break;
            case 'F':{
                     speed_slide[0]=127;
                     speed_slide[1]=159;
                     speed_slide[2]=191;
                     speed_slide[3]=223;
                     speed_slide[4]=255;
                     enable=1;
                     if(index_speed!=0)
                     {
                        move_forward=1;
                        move_backward=0;
                     }
                     relay=0;
                     speed_radius_forward=1;
                     speed_radius_backward=0;
                     speed_circle_forward=1;
                     speed_circle_backward=0;
                     }
                     break;
            case 'B':{
                     speed_slide[0]=127;
                     speed_slide[1]=95;
                     speed_slide[2]=63;
                     speed_slide[3]=31;
                     speed_slide[4]=0;
                     enable=1;
                     if(index_speed!=0)
                     {
                        move_forward=0;
                        move_backward=1;
                     }
                     relay=1;
                     speed_radius_forward=0;
                     speed_radius_backward=1;
                     speed_circle_forward=0;
                     speed_circle_backward=1;
                     }
                     break;
            case 'S':{
                     enable=0;
                     move_forward=0;
                     move_backward=0;
                     relay=0;
                     speed_radius_forward=0;
                     speed_radius_backward=0;
                     speed_circle_forward=0;
                     speed_circle_backward=0;
                     }
                     break;
        }
    }

    if(TI)
    {
        TI=0;
    }
}

void slide(void)
{
    p1_0=1;
    p1_1=0;
    p1_2=0;

    CCAP0L=pwm_slide[index_servo];
    CCAP0H=pwm_slide[index_servo];

    CCAP1L=pwm_slide[index_servo];
    CCAP1H=pwm_slide[index_servo];

    CCAP2L=pwm_slide[index_servo];
```

```
    CCAP2H=pwm_slide[index_servo];

    CCAP3L=pwm_slide[index_servo];
    CCAP3H=pwm_slide[index_servo];

    CCAP4L=pwm_slide[index_servo];
    CCAP4H=pwm_slide[index_servo];

    if(index_speed==0)
    enable=0;

    if(enable==1)
    {
        write_D2A(0x92,speed_slide[index_speed]);
        write_D2A(0x94,speed_slide[index_speed]);
        write_D2A(0x96,speed_slide[index_speed]);
        write_D2A(0x98,speed_slide[index_speed]);
        write_D2A(0x9A,speed_slide[index_speed]);
        write_D2A(0x9C,speed_slide[index_speed]);
    }
    else
    stop();
}

void radius_center(void)
{
    p1_0=0;
    p1_1=1;
    p1_2=0;

    CCAP0L=pwm_slide[index_servo];
    CCAP0H=pwm_slide[index_servo];
```

```
    CCAP1L=pwm_slide[index_servo];
    CCAP1H=pwm_slide[index_servo];

    CCAP2L=pwm_slide[index_servo];
    CCAP2H=pwm_slide[index_servo];

    CCAP3L=pwm_slide[index_servo];
    CCAP3H=pwm_slide[index_servo];

    CCAP4L=pwm_slide[index_servo];
    CCAP4H=pwm_slide[index_servo];

    if(index_speed==0)
    enable=0;

    if(enable==1)
    {
        write_D2A(0x92,speed_slide[index_speed]);
        write_D2A(0x94,speed_slide[index_speed]);
        write_D2A(0x96,speed_slide[index_speed]);
        write_D2A(0x98,speed_slide[index_speed]);
        write_D2A(0x9A,speed_slide[index_speed]);
        write_D2A(0x9C,speed_slide[index_speed]);
    }
    else
    stop();
}

void radius_left(void)
{
    unsigned char i;
```

```
    if(index_servo<8)
    index_servo=8;

    index_servo_radius_left_2wheelrear=(11+(11-index_servo));

    CCAP1L=pwm_slide[index_servo];
    CCAP1H=pwm_slide[index_servo];

    CCAP4L=pwm_slide[(index_servo_radius_left_2wheelrear)];
    CCAP4H=pwm_slide[(index_servo_radius_left_2wheelrear)];

    CCAP2L=237;
    CCAP2H=237;

    CCAP0L=zeta2[index_servo];
    CCAP0H=zeta2[index_servo];

    CCAP3L=zeta2[index_servo_radius_left_2wheelrear];
    CCAP3H=zeta2[index_servo_radius_left_2wheelrear];

    if(index_speed==0)
    enable=0;

    if(enable==1)
    {
        if((speed_radius_forward==1)&(speed_radius_backward==0))
        {
            switch(index_speed)
            {
                case 1:{
                        for(i=0;i<4;i++)
                        {
```

```
    temp_speed_radius_outer[i]=speed_radius_outer_gear1_forward[i];

    temp_speed_radius_inner[i]=speed_radius_inner_gear1_forward[i];

    temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear1_forward[i];

    temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear1_forward[i];
                        }
                    }
                    break;
                case 2:{
                        for(i=0;i<4;i++)
                        {

    temp_speed_radius_outer[i]=speed_radius_outer_gear2_forward[i];

    temp_speed_radius_inner[i]=speed_radius_inner_gear2_forward[i];

    temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear2_forward[i];

    temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear2_forward[i];
                        }
                    }
                    break;
                case 3:{
                        for(i=0;i<4;i++)
                        {

    temp_speed_radius_outer[i]=speed_radius_outer_gear3_forward[i];

    temp_speed_radius_inner[i]=speed_radius_inner_gear3_forward[i];
```

```
        temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear3_forward[i];

        temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear3_forward[i];
                                                }
                                        }
                                        break;
                        case 4:{
                                                for(i=0;i<4;i++)
                                                {

        temp_speed_radius_outer[i]=speed_radius_outer_gear4_forward[i];

        temp_speed_radius_inner[i]=speed_radius_inner_gear4_forward[i];

        temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear4_forward[i];

        temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear4_forward[i];
                                                }
                                        }
                                        break;
    }
}

        if((speed_radius_forward==0)&(speed_radius_backward==1))
        {
                switch(index_speed)
                {
                        case 1:{
                                                for(i=0;i<4;i++)
                                                {
```

```
        temp_speed_radius_outer[i]=speed_radius_outer_gear1_backward[i];

        temp_speed_radius_inner[i]=speed_radius_inner_gear1_backward[i];

        temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear1_backward[i];

        temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear1_backward[i];
                                                }
                                        }
                                        break;
                        case 2:{
                                                for(i=0;i<4;i++)
                                                {

        temp_speed_radius_outer[i]=speed_radius_outer_gear2_backward[i];

        temp_speed_radius_inner[i]=speed_radius_inner_gear2_backward[i];

        temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear2_backward[i];

        temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear2_backward[i];
                                                }
                                        }
                                        break;
                        case 3:{
                                                for(i=0;i<4;i++)
                                                {

        temp_speed_radius_outer[i]=speed_radius_outer_gear3_backward[i];

        temp_speed_radius_inner[i]=speed_radius_inner_gear3_backward[i];
```

```
        temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear3_backward[i];

        temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear3_backward[i];
                                                }
                                        }
                                        break;
                        case 4:{
                                                for(i=0;i<4;i++)
                                                {

        temp_speed_radius_outer[i]=speed_radius_outer_gear4_backward[i];

        temp_speed_radius_inner[i]=speed_radius_inner_gear4_backward[i];

        temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear4_backward[i];

        temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear4_backward[i];
                                                }
                                        }
                                        break;
    }
}

        write_D2A(0x94,temp_speed_radius_outer[index_speed]);
        write_D2A(0x9C,temp_speed_radius_outer[index_speed]);

        write_D2A(0x96,temp_speed_radius_center_inner[index_speed]);

        write_D2A(0x98,temp_speed_radius_center_outer[index_speed]);

        write_D2A(0x92,temp_speed_radius_inner[index_speed]);
```

```
        write_D2A(0x9A,temp_speed_radius_inner[index_speed]);
    }

    else
    stop();
}

void radius_right(void)
{

    unsigned char i;
    if(index_servo>14)
    index_servo=14;

    index_servo_radius_right_2wheelrear=(11-(index_servo-11));

    CCAP0L=pwm_slide[index_servo];
    CCAP0H=pwm_slide[index_servo];

    CCAP3L=pwm_slide[(index_servo_radius_right_2wheelrear)];
    CCAP3H=pwm_slide[(index_servo_radius_right_2wheelrear)];

    CCAP2L=237;
    CCAP2H=237;

    CCAP1L=zeta2[index_servo];
    CCAP1H=zeta2[index_servo];

    CCAP4L=zeta2[index_servo_radius_right_2wheelrear];
    CCAP4H=zeta2[index_servo_radius_right_2wheelrear];
    if(index_speed==0)
    enable=0;
```

```
if(enable==1)
{
        if((speed_radius_forward==1)&(speed_radius_backward==0))
        {
                switch(index_speed)
                {
                        case 1:{
                                                for(i=0;i<4;i++)
                                                {

temp_speed_radius_outer[i]=speed_radius_outer_gear1_forward[i];

temp_speed_radius_inner[i]=speed_radius_inner_gear1_forward[i];

temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear1_forward[i];

temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear1_forward[i];
                                                }
                                        }
                                        break;
                                case 2:{
                                                for(i=0;i<4;i++)
                                                {

temp_speed_radius_outer[i]=speed_radius_outer_gear2_forward[i];

temp_speed_radius_inner[i]=speed_radius_inner_gear2_forward[i];

temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear2_forward[i];

temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear2_forward[i];
                                                }
                                        }
                                        break;
                                case 3:{
                                                for(i=0;i<4;i++)
                                                {

temp_speed_radius_outer[i]=speed_radius_outer_gear3_forward[i];

temp_speed_radius_inner[i]=speed_radius_inner_gear3_forward[i];

temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear3_forward[i];

temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear3_forward[i];
                                                }
                                        }
                                        break;
                                case 4:{
                                                for(i=0;i<4;i++)
                                                {

temp_speed_radius_outer[i]=speed_radius_outer_gear4_forward[i];

temp_speed_radius_inner[i]=speed_radius_inner_gear4_forward[i];

temp_speed_radius_center_outer[i]=speed_radius_center_outer_gear4_forward[i];

temp_speed_radius_center_inner[i]=speed_radius_center_inner_gear4_forward[i];
                                                }
                                        }
                                        break;
}
```

```
}
        if((speed_radius_forward==0)&(speed_radius_backward==1))
        {
                switch(index_speed)
                {
                        case 1:{

temp_speed_radius_outer[4]=speed_radius_outer_gear1_backward[4];

temp_speed_radius_inner[4]=speed_radius_inner_gear1_backward[4];

temp_speed_radius_center_outer[4]=speed_radius_center_outer_gear1_backward[4];

temp_speed_radius_center_inner[4]=speed_radius_center_inner_gear1_backward[4];
                                        }
                                        break;
                                case 2:{

temp_speed_radius_outer[4]=speed_radius_outer_gear2_backward[4];

temp_speed_radius_inner[4]=speed_radius_inner_gear2_backward[4];

temp_speed_radius_center_outer[4]=speed_radius_center_outer_gear2_backward[4];

temp_speed_radius_center_inner[4]=speed_radius_center_inner_gear2_backward[4];
                                        }
                                        break;
                                case 3:{

temp_speed_radius_outer[4]=speed_radius_outer_gear3_backward[4];

temp_speed_radius_inner[4]=speed_radius_inner_gear3_backward[4];

temp_speed_radius_center_outer[4]=speed_radius_center_outer_gear3_backward[4];

temp_speed_radius_center_inner[4]=speed_radius_center_inner_gear3_backward[4];
                                        }
                                        break;
                                case 4:{

temp_speed_radius_outer[4]=speed_radius_outer_gear4_backward[4];

temp_speed_radius_inner[4]=speed_radius_inner_gear4_backward[4];

temp_speed_radius_center_outer[4]=speed_radius_center_outer_gear4_backward[4];

temp_speed_radius_center_inner[4]=speed_radius_center_inner_gear4_backward[4];
                                        }
                                        break;
        }
}


        write_D2A(0x92,temp_speed_radius_outer[index_speed]);
        write_D2A(0x9A,temp_speed_radius_outer[index_speed]);

        write_D2A(0x98,temp_speed_radius_center_inner[index_speed]);
        write_D2A(0x96,temp_speed_radius_center_outer[index_speed]);
        write_D2A(0x94,temp_speed_radius_inner[index_speed]);
        write_D2A(0x9C,temp_speed_radius_inner[index_speed]);
```

```
        }
        else
        stop();
}

void radius(void)
{
        p1_0=0;
        p1_1=1;
        p1_2=0;

        if(index_servo<11)
        radius_left();
        if(index_servo==11)
        radius_center();
        if(index_servo>11)
        radius_right();
}

void circle(void)
{
        p1_0=0;
        p1_1=0;
        p1_2=1;

        CCAP0L=230; // abouth 57.27 degree but the thruth must be 60 degree
        CCAP0H=230;

        CCAP1L=244;
        CCAP1H=244;

        CCAP2L=237;
        CCAP2H=237;

        CCAP3L=244;
        CCAP3H=244;

        CCAP4L=230;
        CCAP4H=230;

        if(index_speed==0)
        enable=0;

        if(enable==1)
        {
                if((speed_circle_forward==1)&(speed_circle_backward==0))
                {
                        switch(index_speed)
                        {
                                case 1 :{
                                                speed_circle_outer_forward=128+32;
                                                speed_circle_inner_forward=128+21;
                                                }
                                                break;
                                case 2 :{
                                                speed_circle_outer_forward=128+64;
                                                speed_circle_inner_forward=128+41;
                                        }
                                                break;
                                case 3 :{
                                                speed_circle_outer_forward=128+96;
                                                speed_circle_inner_forward=128+62;
                                        }
                                                break;
                                case 4 :{
                                                speed_circle_outer_forward=128+127;
                                                speed_circle_inner_forward=128+82;
                                                }
                                                break;
                        }

                        write_D2A(0x92,speed_circle_outer_forward);//outer wheel
                        write_D2A(0x94,speed_circle_outer_backward);//outer wheel

                        write_D2A(0x9A,speed_circle_outer_forward);//outer wheel
                        write_D2A(0x9C,speed_circle_outer_backward);//outer wheel

                        write_D2A(0x96,speed_circle_inner_forward);//inner wheel
                        write_D2A(0x98,speed_circle_inner_backward);//inner wheel
                }

                if((speed_circle_forward==0)&(speed_circle_backward==1))
                {
                        switch(index_speed)
                        {
                                case 1 :{
                                                speed_circle_outer_backward=128-32;
                                                speed_circle_inner_backward=128-21;
                                                }
                                                break;
                                case 2 :{
                                                speed_circle_outer_backward=128-64;
                                                speed_circle_inner_backward=128-41;
                                        }
                                                break;
                                case 3 :{
                                                speed_circle_outer_backward=128-96;
                                                speed_circle_inner_backward=128-62;
                                        }
                                                break;
                                case 4 :{
                                                speed_circle_outer_backward=128-128;
                                                speed_circle_inner_backward=128-82;
                                                }
                                                break;
                        }

                        write_D2A(0x92,speed_circle_outer_backward);//outer wheel
                        write_D2A(0x94,speed_circle_outer_forward);//outer wheel
                        write_D2A(0x9A,speed_circle_outer_backward);//outer wheel
                        write_D2A(0x9C,speed_circle_outer_forward);//outer wheel

                        write_D2A(0x96,speed_circle_inner_backward);//inner wheel
                        write_D2A(0x98,speed_circle_inner_forward);//inner wheel
                }
        }

        else
        stop();
}

void main(void)
{
        enable=0;
        move_forward=0;
        move_backward=0;
        relay=0;
        speed_radius_forward=0;
```

```
    speed_radius_backward=0;
    speed_circle_forward=0;
    speed_circle_backward=0;
    mode=0x01;
    p1_0=1;
    p1_1=0;
    p1_2=0;

    index_servo=11;
    index_speed=0;

    CMOD=0x04;

    TMOD=0x22;
    SCON=0x50;
    TH0=0x70;
    TL0=0x70;
    TH1=0xFA;
    TL1=0xFA;
    TF1=0;
    RI=0;
    TI=0;
    EA=1;
    ES=1;

    CCAPM0=0x42;
    CCAPM1=0x42;
    CCAPM2=0x42;
    CCAPM3=0x42;
    CCAPM4=0x42;

   CCAP0L=237;
    CCAP0H=237;

    CCAP1L=237;
    CCAP1H=237;

    CCAP2L=237;
    CCAP2H=237;

    CCAP3L=237;
    CCAP3H=237;

    CCAP4L=237;
    CCAP4H=237;

    TR0=1;
    TR1=1;
    CCON=0x40;

    write_D2A(0x92,0x7F);
    write_D2A(0x94,0x7F);
    write_D2A(0x96,0x7F);
    write_D2A(0x98,0x7F);
    write_D2A(0x9A,0x7F);
    write_D2A(0x9C,0x7F);
slide();
    while(1)
    {
                switch(mode)
                {
                    case  0x01:slide();
                                break;
                    case  0x02:radius();
                                break;
                case  0x03:circle();
                                break;
                }
    }
}
//The End.
```