ระบบควบคุมไร้สาย

Wireless Control System

โดย

นาย สิรุตม์ บรูพาแสงสูรย์

นาย สุพจน์ อเนกธนโรจน์กุล

นายอภิวัฒน์ วันตา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

ปริญญานิพนธ์ปีการศึกษา 2550

ภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมไร้สาย

Wireless Control System

ผู้จัดทำ

นายสิรุตม์	**บูรพาแสงสูรย์**	47010843
นายสุพจน์	**อเนกธนโรจน์กุล**	47010872
นายอภิวัฒน์	**วันตา**	47010941

.....อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์สุมิตร พนาอุดมทรัพย์)

ระบบควบคุมไร้สาย

โดย

นายสิรุตม์ บูรพาแสงสูรย์ 47010843

นายสุพจน์ อเนกธนโรจน์กุล 47010872

นายอภิวัฒน์ วันตา 47010941

อาจารย์ที่ปรึกษา

อาจารย์ สุมิตร พนาอุดมทรัพย์

ปีการศึกษา 2550

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอ ทฤษฎีระบบควบคุมแบบไร้สาย สำหรับควบคุมหุ่นยนต์กู้ภัย โดยโครงสร้างประกอบด้วย Computer , Access Point , Ethernet I/O ,กล้องวีดีโอไร้สาย , การควบคุมความเร็วมอเตอร์กระแสตรง จุดมุ่งหมายของโครงงานนี้เพื่อสร้างเป็นหุ่นยนต์กู้ภัยเพื่อช่วยผู้ประสบภัยในภัยภิบัติต่าง ๆ

ขั้นตอนการดำเนินการเริ่มจากการออกแบบโครงสร้างตัวหุ่นยนต์ ศึกษาและออกแบบวงจรขับมอเตอร์ ศึกษาการเชื่อมต่อไร้สาย ศึกษาและออกแบบการเขียนโปรแกรมเพื่อรับและส่งสัญญาณการควบคุมไปยังหุ่นยนต์กู้ภัย ซึ่งโปรแกรมคอมพิวเตอร์ที่ใช้ในที่นี้คือ Visual Basic 6 และ C Language เพื่อส่ง – รับข้อมูลจากคอมพิวเตอร์ผ่าน UART ไปยังไมโครคอนโทรลเลอร์ไปยังตัวหุ่นให้มทำงานตามที่เราต้องการ สามารถนำการเคลื่อนที่ของหุ่นยนต์มาจำลองแผนที่ จากการทดลองพบว่าระบบควบคุมไร้สายสามารถส่งสัญญาณควบคุมไปยังหุ่นยนต์กู้ภัยให้ปฏิบัติตามได้อย่างแม่นยำ

Wireless LAN

By

Mr. Sirut burapasangsoon 47010843

Mr. Supoj Anekthanarojkul 47010872

Mr. Aphiwat wanta 47010941

Advisor

Mr. Sumit Phanaaudomsarp

Academic Year 2007

Abstract

This Thesis presents theory and implementation procedures of wireless control system for control the rescue robot. Structures are include computer , access point 2.4 GHz , Ethernet I/O , wireless camera and controlling DC motor. Purpose if for create rescue robot for help a patient in danger situation.

Starting with we design structure of robot and then we study and design drive motor circuit. We study a wireless connection. We study about writing program for send signal control a robot. In this term we use Visual Basic 6 and C Language for transfer data from computer t oUART and control robot what we want

The result wireless control can send control signal to a robot for absolutely work.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้ สามารถสำเร็จลุล่วงไปได้ด้วยดีเพราะได้รับความช่วยเหลือจากทางหลาย ๆ ฝ่ายโดยเฉพาะ อาจารย์สุมิตร พนาอุดมทรัพย์ ที่ได้กรุณาให้คำปรึกษามาอย่างดีโดยตลอดตั้งแต่ต้นรวมถึงอุปกรณ์หลาย ๆ อย่างที่จำเป็น ทั้งเป็นการลดต้นทุน และยังช่วยเหลือด้านอื่น ๆ ที่เป็นประโยชน์ต่อโครงงานผู้จัดทำรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณเพื่อน ๆ ที่คอยถามงานและตามงานตลอดเวลา ให้คำแนะนำเรื่องต่าง ๆ รวมถึงเอื้อเฟื้ออุปกรณ์ และคำปรึกษาทางด้านโปรแกรม

สุดท้ายนี้ผู้จัดทำขอกราบขอบพระคุณบิดา มารดา และครอบครัว ที่คอยเป็นกำลังใจที่ดีตลอดมา รวมถึงสนับสนุนเรื่องงบประมาณตลอดจนเป็นแรงบันดาลใจ ที่ทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี

ผู้จัดทำ

นาย สิรุตม์ บูพราแสงสูรย์

นาย สุพจน์ อเนกธนโรจน์กุล

นาย อภิวัฒน์ วันตา

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่1

บทนำ

1.1 กล่าวนำ

การศึกษาในสาขาวิชาวิศวกรรมศาสตร์ระบบควบคุมเป็นการศึกษาและประยุกต์ ทฤษฎีต่าง ๆ เพื่อการออกแบบและควบคุมระบบให้มีเสถียรภาพและมีสมรรถนะตามความต้องการหรือให้เป็นไปตามข้อกำหนด ดังนั้นเพื่อให้สอดคล้องกับวัตถุประสงค์นี้จึงจำเป็นต้องมีการศึกษา การควบคุมระยะไกลขึ้นมา เพื่อควบคุมระบบทางไกล อีกทั้งการศึกษาวงจรอิเล็กทรอนิคส์ รวมถึงการเขียนโปรแกรมคอมพิวเตอร์ การศึกษาและการเลือกอุปกรณ์วัดและแปลงสัญญาณ ตลอดจนบูรณาการเรื่องที่ศึกษาเหล่านี้ในการประยุกต์ใช้กับระบบควบคุมทางกายภาพจริง ซึ่งนับเป็นสิ่งจำเป็นในการศึกษาสาขาวิชานี้

ในโครงงานนี้จึงเลือกที่จะศึกษาระบบควบคุมไร้สาย ซึ่งเป็นตัวอย่างหนึ่งของระบบควบคุมและน่าสนใจในการศึกษา โดยจะทำการประยุกต์การควบคุมไร้สายเข้ากับ หุ่นยนต์กู้ภัย (Rescue Robot) เพราะเป็นการผสมผสานศาสตร์ในหลาย ๆ ด้าน นอกจากจะได้ความรู้เรื่อง Soft Ware , Hard ware และยังได้ความรู้ทางด้าน เมคานิคส์อีกด้วย

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. ทำการศึกษาและทดลองการใช้งานอุปกรณ์ต่าง ๆ ในระบบควบคุม
2. ทำการศึกษาระบบควบคุมไร้สาย (Wireless LAN) โดยประยุกต์เข้ากับหุ่นยนต์กู้ภัย ซึ่งประกอบด้วย วงจร PWM (Pulse-Width Modulation) มอเตอร์กระแสตรง (DC Mortor) อุปกรณ์จับภาพซึ่งในโครงงานนี้ใช้กล้อง CCTV รวมถึงส่วนควบคุมและประมวลผล โดยโครงงานนี้ใช้คอมพิวเตอร์และโปรแกรม Visual Basic 6.0 กับ ภาษาซี ในการเขียนโปรแกรมด้วย
3. ทำการศึกษาและออกแบบตัวหุ่น เพื่อนำไปใช้กับสถานการณ์จริง

1.3 ขั้นตอนการศึกษาและการจัดทำโครงงาน

เนื่องจาก Rescue Robot นั้นเป็นหุ่นยนต์ช่วยผู้ประสบภัย และจากศึกษากติกาการแข่งขันและสภาพสนามที่ได้รับมาจากผู้จัดการแข่งขันนั้น ทำให้ผู้จัดทำได้ออกแบบตัวหุ่นขึ้นมาโดยใช้หัวหุ่นเป็นล้อแบบ ตีนตะขาบ ซึ่งเล็งเห็นว่าล้อชนิดนี้สามารถปีนป่ายและทำการเข้าพื้นที่ ที่รกและชันได้ดีกว่าล้อชนิดยางทรงกลม อีกทั้งในการแข่งขัน หุ่นยนต์ต้องหาและระบุตำแหน่งเหยื่อ โดนจะต้องวัดว่าเหยื่อ (ผู้ประสบภัย) มีอุณหภูมิเท่าไหร่ อยู่ในสภาพอย่างไร ทำให้ผู้จัดทำ ต้องทำการติดกล้องและ Sensor อุนหภูมิเอาไว้กับตัวหุ่นและทางกติกายังกำหนดให้มีการวาดแผนที่ ทำให้ตัวหุ่นต้องมี Sensor Ultrasonic เอาไว้เพื่อวัดระยะทางของสิ่งกีดขวางถึงตัวหุ่น

โดยการขับมอเตอร์ เพื่อนำเอาไปขับหุ่นนั้นผู้จัดทำ จะใช้การควบคุมแบบ PWM เพราะการควบคุมชนิดนี้สามารถปรับความเร็วของมอเตอร์ได้ เพื่อความสะดวกในการทำงานของตัว Rescue Robot และไมโครคอนโทรลเลอร์ที่ใช้ในการควบคุมหุ่นคือ dsPic30f4011 อีกทั้งยังมีการใช้ Visual Basic 6.0 เนื่องจากเป็นเครื่องมือที่ใช้เขียนโปรแกรมบน Windows เพื่อนำมาสร้าง Applications ในการใช้งานด้วย

1.4 รายละเอียดของปริญญานิพนธ์

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย

บทที่ 1 บทนำ กล่าวนำถึงวัตถุประสงค์ หลักการ ขั้นตอนการศึกษาและการจัดทำโครงงาน พร้อมทั้งรายละเอียดของปริญญานิพนธ์แต่ละบท

บทที่ 2 ทฤษฎีและความรู้ที่เกี่ยวข้อง กล่าวถึงหลักการและทฤษฎีที่เกี่ยวข้องในการควบคุมไร้สาย (Wireless LAN) ระบบขับเคลื่อนมอเตอร์ วงจรอิเล็กทรอนิคส์ที่เกี่ยวข้อง ตัวโปรแกรมและการนำความรู้ไปประยุกต์ใช้ในการทำโครงงาน

บทที่ 3 หลักการออกแบบ นำเสนอการประกอบโครงสร้างของระบบ รวมถึงแนวคิดในการออกแบบ

บทที่ 4 การทดลอง เป็นส่วนของการทดสอบระบบต่าง ๆ ตลอดจนการทดลองใช้งานจริง

บทที่ 5 บทวิจารณ์และสรุป จะสรุปผลการดำเนินงาน ปัญหาทีเกิดขึ้น และแนวทางปรับปรุงโครงงานนี้ต่อไป

บทที่2
ทฤษฎีและหลักการพื้นฐาน

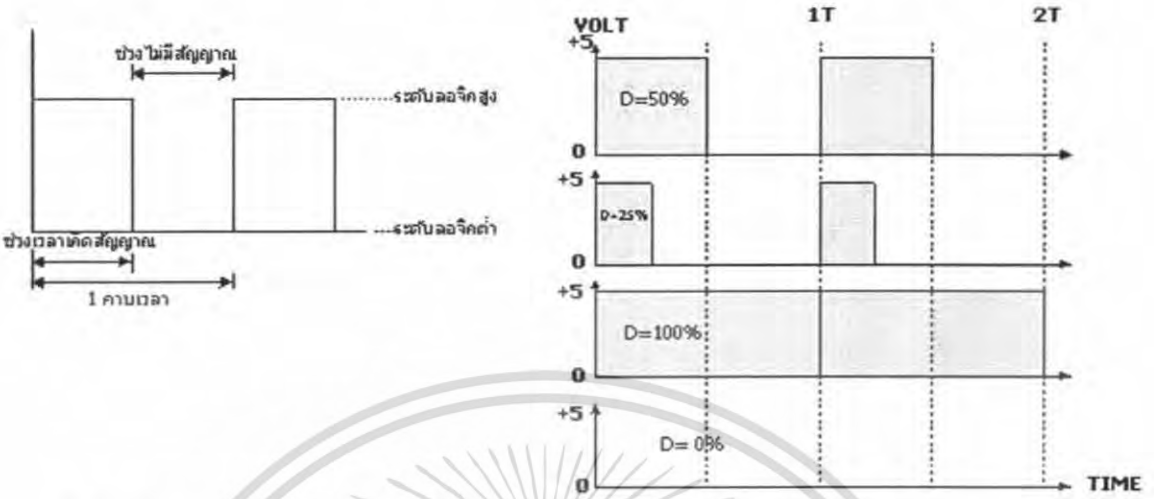
2.1 การควบคุมความเร็วของมอเตอร์กระแสตรง

การควบคุมความเร็วของมอเตอร์กระแสตรงมีหลายวิธีด้วยกัน ซึ่งอาจจะใช้วิธีการควบคุมแบบพื้นฐานทั่วไปเช่นการควบคุมด้วยวิธีการใช้ตัวต้านทานปรับค่าโดยต่ออนุกรมกับมอเตอร์ หรือใช้วิธีการการควบคุมโดยการเปลี่ยนค่าของระดับแรงดันที่ป้อนให้กับมอเตอร์ แต่การควบคุมในวิธีดังกล่าวถึงแม้ว่าจะควบคุมความเร็วมอเตอร์ให้คงที่ได้ แต่ที่ความเร็วต่ำจะส่งผลให้แรงบิดต่ำไปด้วย ดังนั้นเราจึงเลือกใช้วิธีการควบคุมโดยการจ่ายกระแสไฟให้กับมอเตอร์เป็นช่วงๆ โดยอาศัยกระแสไฟที่ป้อนให้กับมอเตอร์ให้เป็นค่าเฉลี่ยที่เกิดขึ้นในแต่ละช่วง ซึ่งเราเรียกว่าวิธีการของการมอดูเลชั่นทางความกว้างของพัลส์ PWM (Pulse Width Modulation)

2.1.1 วิธีการมอดูเลชั่นทางความกว้างของพัลส์ (PWM)

การมอดูเลชั่นทางความกว้างของพัลส์ PWM (Pulse Width Modulation) จะเป็นการปรับเปลี่ยนที่สัดส่วน และความกว้างของสัญญาณพัลส์ โดยความถี่ของสัญญาณพัลส์จะไม่มีการเปลี่ยนแปลง หรือเป็นการเปลี่ยนแปลงที่ค่าของดิวตี้ไซเกิล (Duty Cycle) นั้นเอง ซึ่งค่าของดิวตี้ไซเคิล คือช่วงความกว้างของพัลส์ที่มีสถานะลอจิกสูง โดยคิดสัดส่วนเป็นเปอร์เซตน์จากความกว้างของพัลส์ทั้งหมด ยกตัวอย่างเช่น ถ้าหากค่าดิวตี้ไซเคิลมีค่าเท่ากับเท่ากับ 50% ก็หมายถึงใน 1 รูปสัญญาณพัลส์จะมีช่วงของสัญญาณที่เป็นสถานะลอจิกสูงอยู่ครึ่งหนึ่ง และสถานะลอจิกต่ำอยู่อีกครึ่งหนึ่ง ดังรูป 2.1 และในทำนองเดียวกันถ้าหากค่าดิวตี้ไซเคิลมีค่ามาก หมายความว่าความกว้างของพัลส์ที่เป็นสถานะลอจิกสูงจะมีความกว้างมากขึ้น หากค่าดิวตี้ไซเคิลมีค่าเท่ากับ 100% ก็หมายความว่าจะไม่มีสถานะลอจิกต่ำเลย ซึ่งค่าดิวตี้ไซเคิลสามารถ จะหาได้จากค่าความสัมพันธ์ดังนี้

ค่าดิวตี้ไซเคิล = (ช่วงของสัญญาณพัลส์/คาบเวลาทั้งหมดของสัญญาณ) X100%



รูปที่2.1 แสดงความกว้างของพัลท์ขนาดต่างๆ และค่าดิวตี้ไซเคิล ของช่วงพัลท์ที่มีความถี่คงที่

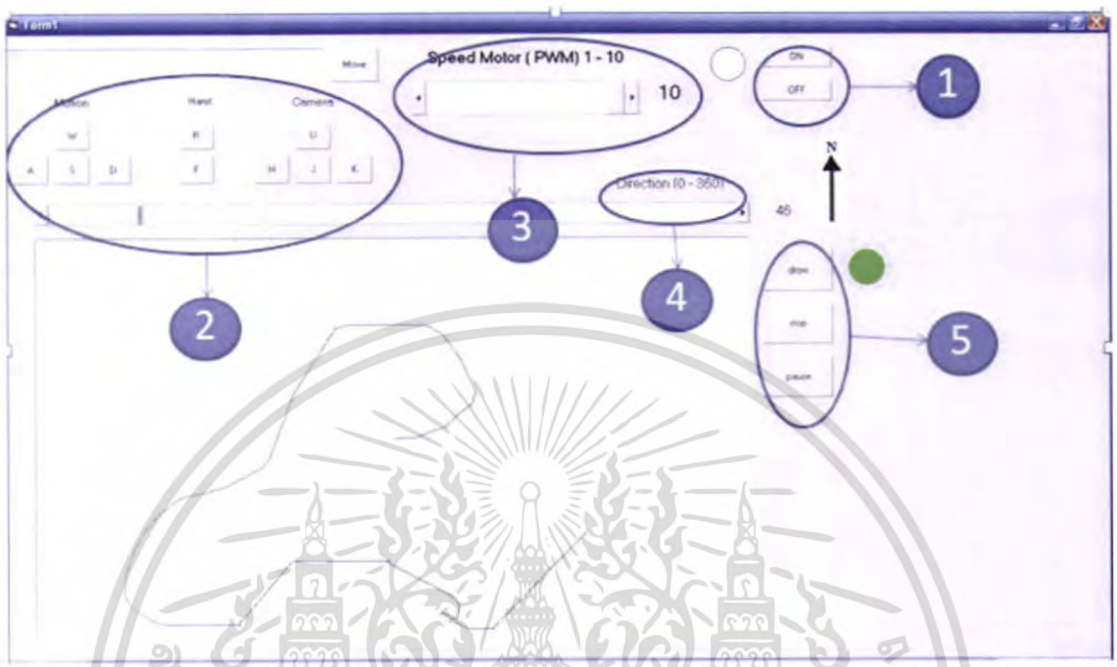
2.2 การเชื่อมต่อไร้สาย

ระบบเครือข่ายไร้สาย (WLAN = Wireless Local Area Network) คือ ระบบการสื่อสารข้อมูลที่นำมาใช้ทดแทนหรือเพิ่มต่อกับระบบเครือข่ายแลนใช้สายแบบดั้งเดิม โดยใช้การส่งคลื่นความถี่วิทยุในย่านวิทยุ RF และ คลื่นอินฟราเรด ในการรับและส่งข้อมูลระหว่างคอมพิวเตอร์แต่ละเครื่องผ่านอากาศ, ทะลุกำแพง, เพดานหรือสิ่งก่อสร้างอื่นๆ โดยปราศจากความต้องการของการเดินสาย นอกจากนั้นระบบเครือข่ายไร้สายก็ยังมีคุณสมบัติครอบคลุมทุกอย่างเหมือนกับระบบ LAN แบบใช้สาย ที่สำคัญก็คือ การที่มันไม่ต้องใช้สายทำให้การเคลื่อนย้ายการใช้งานทำได้โดยสะดวก ไม่เหมือนระบบ LAN แบบใช้สาย ที่ต้องใช้เวลาและการลงทุนในการปรับเปลี่ยนตำแหน่งการใช้งานเครื่องคอมพิวเตอร์

2.3 โปรแกรม Visual Basic 6

ในโครงงานนี้ เลือกใช้โปรแกรม Visual Basic 6 เนื่องจากเป็นเครื่องมือ ที่ใช้ในการเขียนโปรแกรม บน Windows ที่ได้รับความนิยมสูง และนอกจะง่ายต่อการเรียนรู้แล้ว Visual Basic 6 ยังมีเครื่องมือที่ช่วยในการเขียนโปรแกรมเป็นเรื่องที่ไม่ยุ่งยาก เพราะมีเครื่องมีช่วยให้ไม่ต้องจดจำไวยากรณ์ภาษาที่ยุ่งยาก สามารถตรวจสอบได้อัตโนมัติว่าโปรแกรมที่เขียนถูกต้องตามหลักภาษาหรือไม่ มีการแยกแยะส่วนของโปรแกรมอย่างเป็นระเบียบ ทำให้งานของโปรแกรมเมอร์ลดลงได้มาก

2.3.1 หลักการทำงานของ Visual Basic 6



รูปที่ 2.2 แสดงหน้าต่างการทำงานของVisual Basic 6

1. การรับส่งข้อมูล
2. การควบคุมการเคลื่อนที่
3. การควบคุมความเร็วมอเตอร์
4. ส่วนของการรับค่ามุม
5. การวาดแผ่นที่แสดงเส้นทางเดินของหุ่นยนต์

1. การรับส่งข้อมูล

การส่งข้อมูลจะเริ่มขึ้นก็ต่อเมื่อมีการกดปุ่ม ON และจะหยุดลงเมื่อมีการกดปุ่ม OFF ในการส่งข้อมูลนั้นเราจะใช้ ไทเมอร์ 1 ตัวเป็นการวนส่งค่า เมื่อมีการกดปุ่ม ON ก็จะเป็นการสั่งให้ ไทเมอร์ทำงาน จะมีไฟแสดงเป็นสีเขียวเมื่อทำงาน เมื่อมีการกดปุ่ม OFF ก็จะเป็นการสั่งให้ ไทเมอร์หยุดทำงาน ไฟที่แสดงก็จะดับ

2. การควบคุมการเคลื่อนที่

ในการควบคุมหุ่นยนต์ เริ่มจากการรับคำสั่งจาก คีย์บอร์ด จากนั้นทำการแปลงคำสั่งเป็นข้อมูล 8 บิต แล้วส่งข้อมูลนั้นออกทางพอร์ตอนุกรม

การรับค่าจากคีย์บอร์ด

ในที่นี้ได้ใช้คำสั่ง Private Sub Object_KeyDown(KeyCode As Integer, Shift As Integer) เพื่อรับค่าที่กดจากคีย์บอร์ด และให้คำสั่ง Private Sub Object_KeyUp(KeyCode As Integer, Shift As Integer) เพื่อตรวจสอบว่า มีการปล่อยปุ่มแล้ว

เมื่อกดปุ่ม W จะเป็นการสั่งให้หุ่นยนต์เคลื่อนที่ไปข้างหน้า พร้อมทั้งแสดงข้อความ "Forward" ใน กรอบสีขาว

เมื่อกดปุ่ม S จะเป็นการสั่งให้หุ่นยนต์เคลื่อนที่ไปข้างหลัง พร้อมทั้งแสดงข้อความ "Reward" ใน กรอบสีขาว

เมื่อกดปุ่ม A จะเป็นการสั่งให้หุ่นยนต์เคลื่อนที่ไปทางซ้าย พร้อมทั้งแสดงข้อความ " Turnleft" ใน กรอบสีขาว

เมื่อกดปุ่ม D จะเป็นการสั่งให้หุ่นยนต์เคลื่อนที่ไปทางขวา พร้อมทั้งแสดงข้อความ " Turnright" ใน กรอบสีขาว

เมื่อกดปุ่ม R จะเป็นการสั่งให้หุ่นยนต์ยกแขนขึ้น พร้อมทั้งแสดงข้อความ " Hand up" ใน กรอบสีขาว

เมื่อกดปุ่ม F จะเป็นการสั่งให้หุ่นยนต์ยกแขนลง พร้อมทั้งแสดงข้อความ " Hand down" ใน กรอบสีขาว

เมื่อกดปุ่ม U จะเป็นการสั่งให้กล้องหมุนขึ้น พร้อมทั้งแสดงข้อความ " Camera up" ใน กรอบสีขาว

เมื่อกดปุ่ม J จะเป็นการสั่งให้กล้องหมุนลงพร้อมทั้งแสดงข้อความ " Camera down" ใน กรอบสีขาว

เมื่อกดปุ่ม H จะเป็นการสั่งให้กล้องหันไปทางซ้าย พร้อมทั้งแสดงข้อความ " Camera right" ใน กรอบสีขาว

เมื่อกดปุ่ม K จะเป็นการสั่งให้กล้องหันไปทางขวาง พร้อมทั้งแสดงข้อความ " Camera left" ใน กรอบสีขาว

3. การควบคุมความเร็วในการเคลื่อนที่ของหุ่นยนต์โดยใช้ PWM

ในที่นี้ได้ใช้ SCrollBar เป็นการกำหนดค่าต่ำสุดและสูงสุดของย่าน PWM โดยจะสามารถปรับค่าได้ตั้งแต่ 1 – 10 และได้ใช้ ไทเมอร์ 1 ตัว เพื่อวนตรวจสอบค่าและส่งค่าตลอดเวลา ค่าที่ SCrollBar ชี้ จะถูกแสดงด้วย Label ทางขวามือเมื่อตั้งค่าเบื้องต้นของการรับ–ส่งข้อมูลผ่านพอร์ตอนุกรมเรียบร้อยแล้ว จากนั้นเราก็มากำหนดการรับส่งข้อมูลโดย การส่งข้อมูลเป็นการส่งเป็นชุด ชุด1ประกอบไปด้วยข้อมูล 7 ตัว ในการส่งจะส่งข้อมูลออกไปเป็น Char โดยจะส่งค่า Chr(88) ไปก่อน แล้วตามด้วยข้อมูล Y5 , Y4 , Y3 , Y2 , Y1 เป็นจำนวน 5ตัว และจะส่งค่า Chr(89) เป็นค่าสุดท้าย เพื่อให้ dsPic เช็คข้อมูลที่รับไปว่าถูกต้องหรือไม่

Y1 = ข้อมูลที่ใช้ควบคุมกล้องให้หมุนไปทางซ้ายหรือทางขวา

Y2 = ข้อมูลจาก SCrollBar เพื่อปรับค่า PWM

Y3 = ข้อมูลที่ใช้ควบคุมการเคลื่อนที่ของหุ่นยนต์

Y4 = ข้อมูลที่ใช้ควบคุมกล้องให้หมุนขึ้นหรือหมุนลง

Y5 = ข้อมูลที่ใช้ควบคุมแขนของหุ่นยนต์

4. ส่วนของการรับค่ามุม

หลักการรับค่ามุมคือ ให้ค่าของSCroolBar ตั้งแต่ 0 – 360 มาคำนวณหาค่าจุดที่เราต้องการวาดลงไปบนพื้นที่ โดยมีความสัมพันธ์ระหว่างจุดที่ต้องการจะวาดกับค่ามุมที่ได้จากSCrollBar ดังนี้

```
m = HScroll2.Value
r = -m / 360 * 2 * 3.14159
x = x1 + Cos(r)
y = y1+ Sin(r)
```

เมื่อ m = ค่าจากSCrollBar

r = ค่ามุมที่แปลงจาก องศา เป็น เรเดียน

x1 = ค่าจุดก่อนหน้าตามแกนแกน x

y1 = ค่าจุดก่อนหน้าตามแกน y

x = ค่าที่ต้องการตามแกน x

y = ค่าที่ต้องกการตามแกน y

5. การวาดแผ่นที่แสดงเส้นทางเดินของหุ่นยนต์

ในการวาดแผ่นที่นี้ เราได้ใช้การวาดกราฟบน Visual Basic 6 มาแสดงเส้นทางเดินของหุ่นยนต์ โดย รับค่ามุม และ ความเร็วจากผู้ควบคุม

จากนั้นให้กำหนดจุดเริ่มต้น โดยในที่นี้ เรากำหนดเป็นจุดกึ่งกลางของพื้นที่ทั้งหมด และกราฟจะถูกวาดทีละ 1 pixels โดยทิศทางจะขึ้นกับค่ามุมที่ผู้ควบคุมป้อนเข้าไป ส่วนความเร็วในการวาดจะขึ้นกับค่า PWM และเมื่อกราฟชน ขอบข้างใดข้างหนึ่ง จะกลับมาเริ่มที่จุดกึ่งกลางใหม่

เมื่อเรากดปุ่ม pause จะเป็นการสั่งให้หยุดวาดกราฟชั่วคราว โดยที่จะยังคงค่าเดิมไว้ และไฟบอกสถานะจะกลายเป็นสีแดง และ เมื่อกดปุ่ม drow ก็จะกลับมาวาดต่อได้ และ ไฟสถานะจะเปลี่ยนเป็นสีเขียว

เมื่อมีการกดปุ่ม stop การวาดกราฟจะหยุดลง และกราฟจะถูกลบทั้งหมด ค่าก็จะกลับไปที่จุดเริ่มต้นอีกครั้ง(กึ่งกลางพื้นที่)

2.4 หลักการทำงานในdsPic30f4011

การทำงานแบ่งออกเป็น 3 ส่วน

1. ส่วนการรับ – ส่งข้อมูลผ่านทาง UART
2. PWM
3. ส่วนควบคุม

2.4.1. การรับ – ส่ง ข้อมูลผ่านทาง UART

เป็นการส่งข้อมูลเป็นชุด โดย ข้อมูลแต่ละตัวส่งเป็นChar ใน 1ครั้งของการส่งประกอบไปด้วยข้อมูล 7 ตัว ตั้งแต่ X0 จนถึง X6 โดยที่

X0 , X6 = ข้อมูลสำหรับตรวจสอบว่าข้อมูลที่ส่งมาถูกต้องหรือไม่ โดย การข้อมูลที่ถูกต้อง ค่า X0 และX6 จะมีค่าเท่ากับ 0X58 และ 0X59 ตามลำดับ (เป็นค่าจากตาราง ASCII)

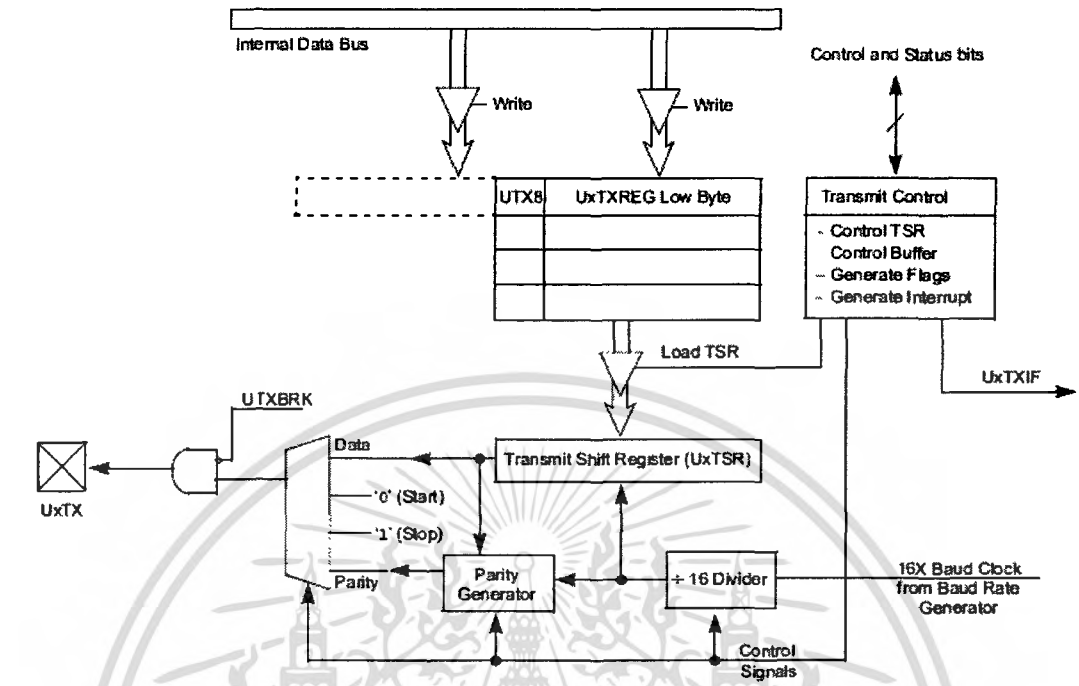
X1 = ข้อมูลสำหรับควบคุมหุ่นยนต์ยกแขนขึ้นหรือยกแขนลง โดย

X2 = ข้อมูลสำหรับควบคุมกล้องหมุนขึ้นหรือหมุนลง

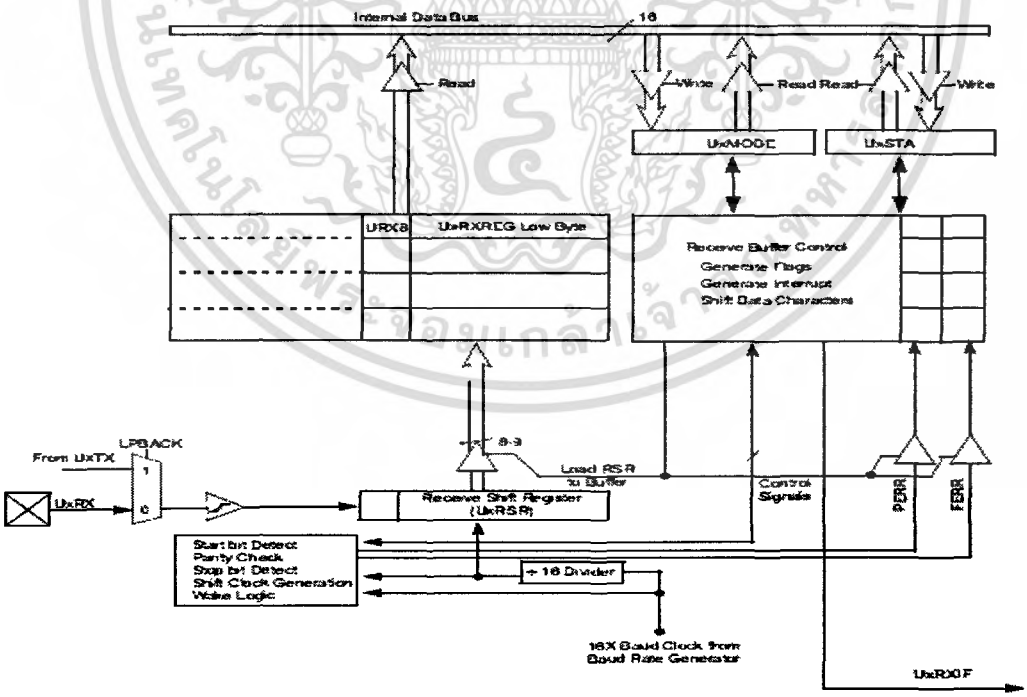
X3 = ข้อมูลสำหรับควบคุมการเคลื่อนที่ของหุ่นยนต์

X4 = ข้อมูลสำหรับปรับค่าความเร็วในการเคลื่อนที่ของหุ่นยนต์โดยปรับแบบPWM

X5 = ข้อมูลสำหรับควบคุมกล้องเคลื่อนที่ซ้ายหรือขวา



รูปที่ 2.3 ไดอะแกรมการทำงานของตัวส่งโมดูล UART ของdsPic



รูปที่ 2.4 ไดอะแกรมการทำงานของตัวรับโมดูล UART ของdsPic

2.4.1.1 การกำหนดการทำงานของUART ใน dsPic30f4011

UART ใน DSPIC ใช้รูปแบบมาตรฐาน NRZ คือ มีบิตเริ่มต้น 1 บิต ,ข้อมูล 8 หรือ 9 บิต และบิตปิดท้าย 1บิต ส่วนบิตพาริตี้สามารถเลือก แบบคู่,แบบคี่ หรือ ไม่มี ก็ได้ ในที่นี้ เรากำหนดแบบ 8N1 คือ มีบิตเริ่มต้น 1 บิต ,ข้อมูล 8 บิต และบิตปิดท้าย 1 บิต สามารถกำหนดได้ที่บิต PDSEL1 , PDSEL0 และ STSEL ให้เป็น 2 , 1 , 0 ตามลำดับ ส่วนการกำหนดอัตราบอดนั้น กำหนดผ่านรีจิสเตอร์ UXBRG ขนาด 16 บิต

การรับ – ส่งในUART จะรับและส่งข้อมูลแบบ LSB หรือบิตนัยสำคัญต่ำสุดก่อน โดยตัวรับและตัวส่งUART ใน dsPic จะทำงานเป็นอิสระแยกจากกัน โดยที่ใช้อัตราบอด และรูปแบบของข้อมูลเหมือนกัน จึงสามารถรับและส่งข้อมูลได้พร้อมกัน

2.4.1.2 การเอ็นเอเบิลโมดูล UART ใน dsPic30f4011

ทำได้โดยการเซตบิต UARTEN ซึ่งเป็นบิต 15 ในรีจิสเตอร์ U1MODE และเซตบิต UTXEN ซึ่งเป็นบิทที่ 10 ในรีจิสเตอร์ U1STA โดยทันทีที่เอ็นเอเบิล ขาU1TX และ ขาU1TX จะทำงานโดยไม่สนใจการกำหนดทิศทางที่เกิดขึ้นก่อนหน้านี้

2.4.1.3 การดิสเอเบิลโมดูล UART ใน dsPic30f4011

ทำได้โดยการเคลียร์บิต UARTEN โดยปกติโมดูล UART จะถูกดิสเอเบิลหลังจากที่เกิดการรีเซต ซึ่งเป็นการกำหนดสถานะเบื้องต้น จากนั้นหากต้องการให้ทำงานจะต้องเอ็นเอเบิลภายหลัง เมื่อ UART ดิสเอเบิล ขา U1TX และ U1RX จะถูกปลดออกจาก UART ทำให้สามารถเอาไปใช้งานอื่นๆ ได้

นอกจากนั้นในภาวะที่โมดูล UART ถูกดิสเอเบิล ค่าในบัฟเฟอร์ทั้งหมดจะถูกเคลียร์ เช่นเดียวกับบิตแฟลกแจ้งสถานะความผิดพลาดทั้งหมดจะถูกเคลียร์ด้วย ส่วนตัวนับอัตราบอดจะอยู่ในสภาพรีเซต หลังจากที่ดิสเอเบิล UART แล้วกลับมาเอ็นเอเบิลใหม่ การทำงานทั้งหมดจะเริ่มต้นใหม่อีกครั้ง

2.4.1.4 การคำนวณหาค่าอัตราบอด

การคำนวณหาค่าอัตราบอด คิดจากความสัมพันธ์ระหว่างสัญญาณนาฬิกาความถี่ของระบบกับัตราบอดในการสื่อสารสามารถแสดงในรูปสมการดังนี้

$$\text{อัตราบอด} = F_{cy} / (16 \times (BRG + 1))$$

และ

$F_{cy} = F_{osc} / 4$

โดยที่ BRG คือ ค่าข้อมูลของรีจิสเตอร์ UxBRG (x เท่ากับ 1 หรือ 2 ขึ้นกับUART ชุดที่ใช้)

F_{osc} คือ ความถี่สัญญาณนาฬิกาหลัก

ค่าตราบอดที่ใช้ คือ 9600 บิตต่อวินาทีโดยกำหนดตัวแปรขึ้นมาตัวหนึ่งเพื่อเก็บค่าในที่นี้คือ baudvalue โดยที่ค่ากำหนดจะถูกโหลดไปยังรีจิสเตอร์ U1BRG อีกทีหนึ่ง สามารถคำนวณค่า F_{cy} และค่าของ BRG ได้ดังนี้

$F_{cy} = F_{osc} / 4 =$ (7372800 x 4) / 4 = 7372800

อัตราบอด = F_{cy} / (16 x (BRG + 1))

9600 = 7372800 / (16 x (BRG + 1))

BRG = (7372800 / (16 x 9600)) – 1 = 47

ดังนั้นการกำหนดค่าเริ่มต้นของตัวแปร baudvalue จึงใช้ค่าเป็น 47

2.4.1.5 รีจิสเตอร์ที่ใช้ในโมดูล UART

มีทั้งสิ้น 5 ตัว คือ

UxMODE ใช้กำหนดโหมดการทำงานของโมดูล UART โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน DSPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด ในที่นี้กำหนดเป็น **U1MODE**

UxSTA ใช้แสดงสถานะการทำงานของโมดูล UART โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน DSPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด ในที่นี้กำหนดเป็น **U1STA**

UxRXREG ใช้เก็บข้อมูลที่รับเข้ามาทางขาเชื่อมต่อพอร์ตอนุกรม โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน DSPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด ในที่นี้กำหนดเป็น **U1RXREG**

UxTXREG ใช้เก็บข้อมูลสำหรับส่งออกทางขาเชื่อมต่อพอร์ตอนุกรม โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน DSPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด ในที่นี้กำหนดเป็น **U1TXREG**

UxBRG ใช้เก็บค่าสำหรับกำหนดอัตราบอด โดยตัวอักษร x เป็นเลข 1 หรือ 2 เพราะใน DSPIC สามารถบรรจุโมดูล UART ได้ 2 ชุด ในที่นี้กำหนดเป็น **U1BRG**

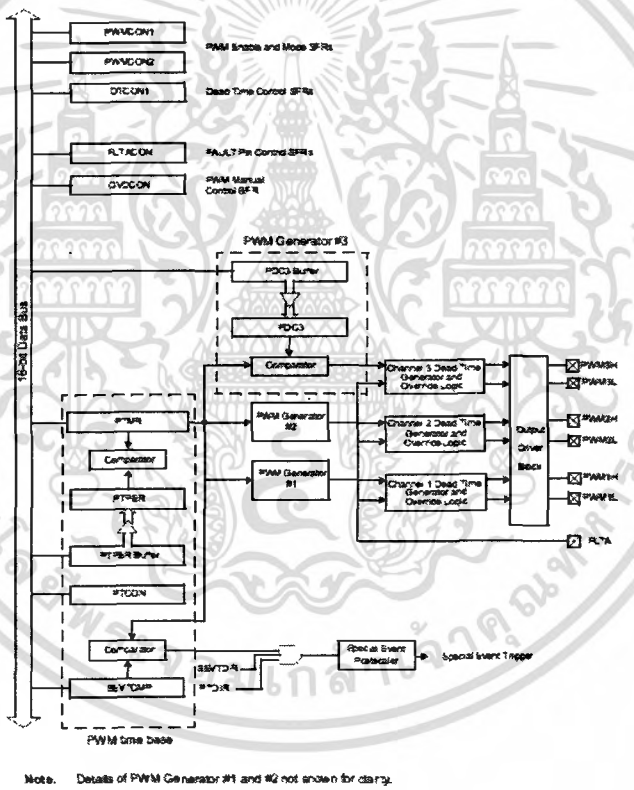
2.4.2 PWM

PWM คือการกำหนดให้โมดูล MCPWM ใน DSPIC สร้างสัญญาณ Pulse ออกมา เพื่อนำไปจ่ายให้กับมอเตอร์ เพื่อเพิ่มความเร็วหรือลดความเร็วได้ตามที่ต้องการ ในที่นี้สามารถ กำหนดความเร็วมอเตอร์ ได้ 10 ระดับ คือ ตั้ง แต่ 0 % ไปจนถึง 100 % โดยเพิ่มขึ้นที่ละ 10 %

2.4.2.1 ฐานเวลาสัญญาณ PWM

การกำหนดฐานเวลาของสัญญาณ PWM ได้มาจากการทำงานของไทเมอร์ 15 บิตทำงานร่วมกับ ปรีสเกลเลอร์และโพสต์สเกลเลอร์ในโมดูล MCPWM โดยข้อมูล 15 บิตนั้นจะบรรจุอยู่ใน 15 บิตล่างของรีจิสเตอร์ PTMR ส่วนบิต MSB คือบิต PTDIR เป็นบิตที่อ่านได้อย่างเดียวใช้ในการแสดงทิศทางการนับค่าในปัจจุบันของฐานเวลา PWM นี้ โดยถ้าบิตนี้เป็น 0 แสดงว่า PTMR กำลังนับขึ้น และเป็น 1 เมื่อกำลังนับลง

การเอ็นเอเบิลให้ส่วนฐานเวลา PWM นี้ทำงานต้องเซตบิต PTEN ซึ่งเป็นบิต 15 ของรีจิสเตอร์ PTCON อย่างไรก็ตาม ค่าในรีจิสเตอร์ PTMR จะไม่ถูกเคลียร์แม้ว่าส่วนฐานเวลาของ PWM นี้จะถูกดิสเอเบิลด้วยการเคลียร์บิต PTEN



Note. Details of PWM Generator #1 and #2 not shown for clarity.

รูปที่ 2.5 แสดง ไดอะแกรมของ โมดูล MCPWM

ฐานเวลา PWM ในโมดูล MCPWM สามารถกำหนดให้ทำงานได้ 4 โหมดคือ

1. โหมดเปลี่ยนแปลงอิสระ(Free Running mode)
2. โหมดทำงานครั้งเดียว(Single Event mode)
3. โหมดนับค่าขึ้นหรือลงอย่างต่อเนื่อง(Continuous Up/Down Count mode)

4.โหมดนับค่าขึ้นหรือลงอย่างต่อเนื่องพร้อมกับการอินเตอร์รัปต์เพื่อปรับปรุงค่า(Continuous Up/Down Count mode with interrupts for double - updates)

การเลือกโหมดทำได้โดยการกำหนดค่าบิตที่ PTMOD1 และ PTMOD0 ซึ่งเป็นบิต 1 และ 0 ในรีจิสเตอร์ PTCON

2.4.2.1.1 ฐานเวลา PWM ในโหมดเปลี่ยนแปลงค่าอิสระ

ในโหมดนี้ค่าฐานเวลาจะเพิ่มขึ้นจนกระทั่งตรงกับค่าในรีจิสเตอร์ PTPER จากนั้นรีจิสเตอร์ PTMR จะรีเซตและทำการนับค่าเพิ่มขึ้นต่อเนื่องไปอีกตราบเท่าที่บิต PTEN ยังเซตเป็น 1 อยู่

2.4.2.1.2 ฐานเวลา PWM ในโหมดทำงานครั้งเดียว

ในโหมดนี้ฐานเวลาจะมีการนับขึ้นเมื่อเซตบิต PTEN เมื่อค่าของรีจิสเตอร์ PTMR ตรงกับ PTPER รีจิสเตอร์ PTMR จะเซต บิต PTEN จะถูกเคลียร์โดยอัตโนมัติโดยกระบวนการทางฮาร์ดแวร์ทำให้ฐานเวลา PWM นี้หยุดทำงานตามไปด้วย

2.4.2.1.3 ฐานเวลา PWM ในโหมดนับค่าขึ้นหรือลงอย่างต่อเนื่อง

ในโหมดนี้ค่าฐานเวลาจะเพิ่มขึ้นจนกระทั่งเท่ากับค่าในรีจิสเตอร์ PTPER จากนั้นจะกลับทิศทางการนับเป็นนับค่าลงแทน จนกระทั่งเท่ากับ 0 แล้วกลับไปเริ่มต้นนับขึ้นใหม่ บิต PTDIR ซึ่งเป็นบิต 15 ของรีจิสเตอร์ PTMR จะแสดงให้ทราบถึงทิศทางการนับของปัจจุบัน โดยเป็น 0 เมื่อนับขึ้นและ เป็น 1 เมื่อนับลง

2.4.2.1.4 ปรีสเกลเลอร์ของฐานเวลา PWM

สัญญาณนาฬิกาที่ใช้ในการนับค่าของฐานเวลา PWM ในโมดูล MCPWM ก็คือสัญญาณนาฬิกาในการทำงานบองระบบนั่นเอง โดยมีค่าความถี่เท่ากับ FCY (ซึ่งเท่ากับ ¼ ของความถี่หลัก) สัญญาณนาฬิกาจะส่งผ่านไปรับค่าในรีจิสเตอร์ PTMR โดยมีตัวลดทอนหรือปรีสเกเลอร์เข้ามาจัดการเพื่อปรับอัตราการนับค่า ซึ่งส่งผลต่อความถี่ของสัญญาณ PWM ดูรูปที่ 14-2 ประกอบ อัตราการลดทอนของปรีสเกลเลอร์เลือกได้ 4 อัตราคือ 1:1 (ไม่ลดทอน) 1:4, 1:16 และ 1:64 โดยกำหนดจากบิต PTCKPS1 และ PTCKPS0 ซึ่งเป็นบิต 3 และ 2 ของรีจิสเตอร์ PTCON

ค่าของปรีสเกลเลอร์จะถูกเคลียร์เมื่อเกิดเหตุการณ์ดังนี้

1. เกิดการเขียนข้อมูลไปยังรีจิสเตอร์ PTMR
2. เกิดการเขียนข้อมูลไปยังรีจิสเตอร์ PTCON
3. เกิดการรีเซตขึ้นในไมโครคอนโทรลเลอร์

2.4.2.1.5 โพสต์สเกลเลอร์ของฐานเวลา PWM

เมื่อค่าของรีจิสเตอร์ PTMR ตรงกับค่าของรีจิสเตอร์ PTPER จะเกิดสัญญาณเอาต์พุตขึ้นเพื่อนำไปสร้างสัญญาณอินเตอร์รัปต์และในส่วนฐานเวลา PWM นี้ได้เพิ่มเติมความสามาถพิเศษตัวหนึ่งคือ โพสสเกลเลอร์เพื่อปรับอัตราสัญญาณเอาต์พุตนี้ โดยแทนที่จะส่งสัญญาณเอาต์พุตออกไปทันทีก็สามารถปรับอัตราเอาต์พุตนี้ได้ โดยเลือกได้ถึง 16 ระดับตั้งแต่ 1:1 (ไม่ปรับค่า) จนถึง 1:16 นั่นคือเกิดเหตุการณ์ค่าตรงกัน 16 ครั้ง จึงส่งสัญญาณเอาต์พุตออกมา ดังนั้นโพสต์สเกลเลอร์จะถูกใช้งานในกรณีที่ไม่ต้องการปรับค่าดิวตี้ไซเกิลในทุก ๆ ไซเกิลของสัญญาณ PWM

ค่าตัวนับในโพสต์สเกลเลอร์จะถูกเคลียร์เมื่อเกิดเหตุการณ์ดังนี้

1. เกิดการเขียนข้อมูลไปยังรีจิสเตอร์ PTMR
2. เกิดการเขียนข้อมูลไปยังรีจิสเตอร์ PTCON
3. เกิดการรีเซตขึ้นในไมโครคอนโทรลเลอร์

2.4.2.1.6 การอินเตอร์รัปต์ในส่วนฐานเวลา PWM

สัญญาณอินเตอร์รัปต์ที่เกิดขึ้นจากฐานเวลา PWM นี้สามารถกำหนดได้จากบิต PTMOD1 และ PTMOD0 (บิต 1 และ 0 ของรีจิสเตอร์ PTCON) ร่วมกับบิต PTOPS3 ถึง PTOPS0 (บิต 7 ถึง 4 ของรีจิสเตอร์ PTCON) โดยขึ้นกับโหมดการทำงานของฐานเวลา PWM ด้วยดังนี้

1. อินเตอร์รัปต์ของฐานเวลา PWM ในโหมดเปลี่ยนแปลงค่าอิสระ

ในโหมดนี้การอินเตอร์รัปต์จะเกิดขึ้นเมื่อรีจิสเตอร์ PTMR มีค่าตรงกับค่าในรีจิสเตอร์ PTPTER สามารถใช้โพสต์สเกลเลอร์เพื่อลดค่าความถี่ของเหตุการณ์อินเตอร์รัปต์นี้ได้

2. อินเตอร์รัปต์ของฐานเวลา PWM ในโหมดทำงานครั้งเดียว

ในโหมดนี้การอินเตอร์รัปต์จะเกิดขึ้นเมื่อรีจิสเตอร์ PTMR มีค่าตรงกับค่าในรีจิสเตอร์ PTPER ไม่สามารถใช้โพสต์สเกลเลอร์เพื่อลดค่าความถี่ของเหตุการณ์อินเตอร์รัปต์นี้ได้ นอกจากนั้นบิต PTEN จะถูกเคลียร์ ทำให้เกิดการดิสเอเบิลการทำงานของส่วนฐานเวลา PWM นี้ตามไปด้วย

3. อินเตอร์รับของฐานเวลา PWM ในโหมดนับค่าขึ้นลงอย่างต่อเนื่อง

ในโหมดนี้การอินเตอร์รัปต์จะเกิดขึ้นทุกครั้งที่รีจิสเตอร์ PTMR เป็น “0” จากนั้นค่าฐานเวลา PWM จะเริ่มนับค่าขึ้น สามารถใช้โพสต์สเกลเลอร์เพื่อลดค่าความถี่ของเหตุการณ์อินเตอร์รัปต์นี้ได้

4. อินเตอร์รัปต์ของฐานเวลา PWM ในโหมดนับค่าขึ้นลงอย่างต่อเนื่องพร้อมการอินเตอร์รัปต์เพื่อปรับปรุงค่า

ในโหมดนี้การอินเตอร์รัปต์จะเกิดขึ้นทุกครั้งที่รีจิสเตอร์ PTMR เป็น “0” และทุกครั้งที่ค่าของคาบเวลาตรงกันไม่สามรถใช้โพสต์สเกลเลอร์เพื่อลดค่าความถี่ของเหตุการณ์อินเตอร์รัปต์นี้ได้ นอกจากนั้นในโหมดนี้ยังกำหนดให้เกิดการปรับปรุงค่าดิวตี้ไซเกิล 2 ครั้งต่อคาบเวลา โดยสามารถเลือกได้วาต้องการให้ทำงานที่ขอบขาขึ้นหรือลงของสัญญาณ PWM

2.4.2.1.7 คาบเวลาของสัญญาณ PWM

รีจิสเตอร์ PTPER ถูกใช้สำหรับกำหนดค่าการนับคาบเวลาของรีจิสเตอร์ PTMR ผู้พัฒนาต้องระบุข้อมูลขนาด 15 บิตลงในบิต 0 ถึง 14 ของรีจิสเตอร์ PTPER เมื่อโมดูลนี้ทำงานจนกระทั่งค่าของรีจิสเตอร์ PTMR เท่ากับ PTPER ค่าฐานเวลาจะรีเซตเป็น “0” หรือเปลี่ยนทิศทางการนับค่าในสัญญาณนาฬิกาลูกถัดไป ขึ้นอยู่กับการกำหนดโหมดทำงาน

คาบเวลาของฐานเวลาทีขนาดของบัฟเฟอร์แป็น 2 เท่าเพื่อรองรับการเปลี่ยนแปลงค่าในระหว่างการทำงานได้ โดยปราศจากการรบกวน นั่นคือรีจิสเตอร์ PTPER จะมีรีจิสเตอร์บัฟเฟอร์สำหรับสำรองรับค่าที่ต้องการเปลี่ยนแปลงใหม่ ในระหว่างที่กำลังทำงานกับค่าเดิม โดยรีจิสเตอร์บัฟเฟอร์นี้ผู้ใช้งานไม่สามารถเข้าถึงได้ ข้อมูลสำหรับกำหนดค่าคาบเวลาจะถูกเขียนลงในรีจิสเตอร์ PTPTER แตกต่างกันไปตามโหมดการทำงานของฐานเวลา PWM ดังนี้

(ก) ในโหมดเปลี่ยนแปลงค่าอิสระและโหมดทำงานครั้งเดียว ข้อมูลจากรีจิสเตอร์ PTPTER จะถูกโหลดลงในรีจิสเตอร์คาบเวลาที่รีจิสเตอร์ PTMR ถูกรีเซตเป็น “0” หลังจากที่ค่าของรีจิสเตอร์ PTMR ตรงกับค่าของ PTER ดังแสดงด้วยไดอะแกรมเวลาในรูปที่ 14-3 (ก)

(ข) ในโหมดนับค่าขึ้นลง ข้อมูลจากรีจิสเตอร์ PTPER จะถูกโหลดลงในรีจิสเตอร์คาบเวลาเมื่อรีจิสเตอร์ PTMR มีค่าเป็น “0” ดังแสดงด้วยไดอะแกรมเวลาในรูปที่ 14-3 (ข)

นอกจากนั้นข้อมูลในรีจิสเตอร์ PTPER จะถูกโหลดไปยังรีจิสเตอร์คาบเวลาอย่างอัตโนมัติเมื่อฐานเวลา PWM ถูกดิสเอเบิลโดยกำหนดบิต PTEN เป็น “0”

2.4.2.2 โหมดการทำงานของส่วนกำเนิดสัญญาณ PWM ในโมดูล MCPWM

มีด้วยกัน 4 แบบหลักคือ

1. โหมดปรับขอบสัญญาณ (Edge aligned mode)

2. โหมดสัญญาณเดี่ยว (Single event mode)

3. โหมดปรับสัญญาณกึ่งกลาง (Center aligned mode)

4. โหมดปรับสัญญาณกึ่งกลางพร้อมปรับปรุงค่า (Center aligned mode with double updates)

ซึ่งจะสัมพันธ์กับโหมดการทำงานของฐานเวลา PWM โดย

- เมื่อฐานเวลา PWM ทำงานในโหมดเปลี่ยนแปลงค่าอิสระ ส่วนกำเนิดสัญญาณ PWM จะทำงานในโหมดปรับขอบสัญญาณ

- เมื่อฐานเวลา PWM ทำงานในโหมดทำงานครั้งเดียว ส่วนกำเนิดสัญญาณ PWM จะทำงานในโหมดสัญญาณเดี่ยว

- เมื่อฐานเวลา PWM ทำงานในโหมดนับค่าขึ้นลงอย่างต่อเนื่อง ส่วนกำเนิดสัญญาณ PWM จะทำงานในโหมดปรับสัญญาณกึ่งกลาง

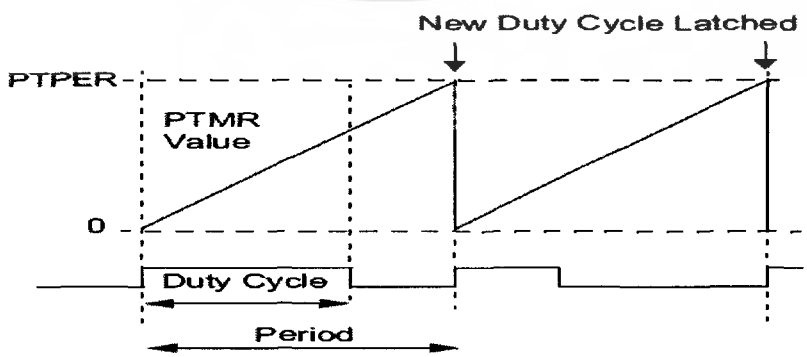
- เมื่อฐานเวลา PWM ทำงานในโหมดนับค่าขึ้นลงอย่างต่อเนื่องพร้อมปรับปรุงค่า ส่วนกำเนิดสัญญาณ PWM จะทำงานในโหมดปรับสัญญาณกึ่งกลางพร้อมปรับปรุงค่า

2.4.2.2.1 การทำงานของส่วนกำเนิดสัญญาณ PWM ในโหมดปรับขอบสัญญาณ

ไดอะแกรมเวลาแสดงการเกิดสัญญาณ PWM เมื่อทำงานในโหมดปรับขอบสัญญาณ เริ่มต้นด้วยการกำหนดให้ฐานเวลา PWM ทำงานในโหมดเปลี่ยนแปลงค่าอิสระ ดังนั้นคาบเวลาของสัญญาณ PWM จะถูกกำหนดโดยค่าที่โหลดให้แก่รีจิสเตอร์ PTPER ส่วนดิวตี้ไซเกิลถูกกำหนดโดยค่าในริสเตอร์ PDCx

ในกรณีที่ค่าดิวตี้ไซเกิลไม่เป็นศูนย์ วงจรเอาต์พุตของส่วนกำเนิดสัญญาณ PWM ทุกชุดที่ได้รับการเอ็นเอเบิลจะทำงานที่จุดเริ่มต้นของคาบเวลาสัญญาณ PWM หรือเมื่อค่าในรีจิสเตอร์ PTMR เท่ากับ 0 และหยุดทำงานเมื่อค่าของรีจิสเตอร์ PTMR ตรงกับค่าดิวตี้ไซเกิลของส่วนกำเนิดสัญญาณ PWM

ถ้าหากค่าในรีจิสเตอร์ PDCx เป็นศูนย์ วงจรเอาต์พุตของส่วนกำเนิดสัญญาณ PWM จะไม่ทำงานใด ๆ นั่นหมายความว่า ในโหมดนี้จะสามารถกำเนิดสัญญาณ PWM ได้ก็ต่อเมื่อค่าในรีจิสเตอร์ PDCx ต้องมากกว่าค่าที่กำหนดในรีจิสเตอร์ PTER



รูปที่ 2.6 ไดอะแกรมแสดงเวลาการเกิดสัญญาณ PWM เมื่อทำงานในโหมดปรับขอบสัญญาณ

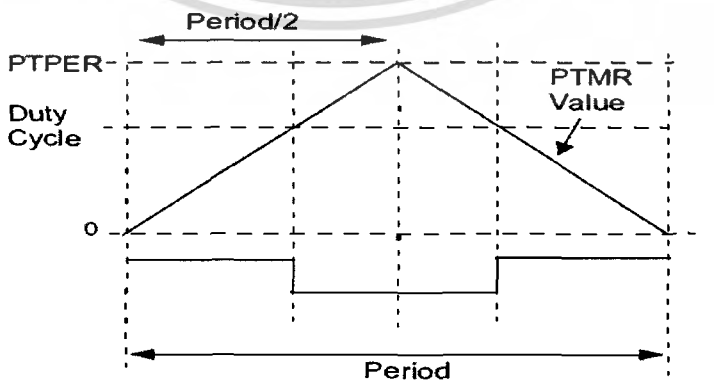
โมดูล MCPWM จะกำเนิดสัญญาณพัลส์ออกมาเพียงลูกเดียวหรือพัลส์เดี่ยวเมื่อฐานเวลา PWM ถูกกำหนดให้ทำงานในโหมดทำงานครั้งเดียว (ค่าของ PTMOD1 และ PTMOD0 เท่ากับ "01") ในรูปที่ 14-6 แสดงไดอะแกรมเวลาของการกำเนิดสัญญาณ PWM ในโหมดนี้ โดยการกำเนิดสัญญาณในลักษณะนี้จะถูกนำไปใช้ขับสวิตซ์รัลักแตนซ์มอเตอร์ความเร็วสูง โหมดนี้จะเริ่มต้นการทำงานนเมื่อบิต PTEN ถูกเซตเพื่อเอ็นเอเบิลการทำงานของฐานเวลา PWM วงจรเอาต์พุตของโมดูล MCPWM จะทำงานทันที

เมื่อค่าในรีจิสเตอร์ PTMR ตรงกับค่าดิวตี้ไซเกิลในรีจิสเตอร์ PDCx วงจรเอาต์พุตจะหยุดทำงาน

เมื่อค่าในรีจิสเตอร์ PTMR เท่ากับค่าในรีจิสเตอร์ PTPER ค่าในรีจิสเตอร์ PTMR ถูกเคลียร์ วงจรเอาต์พุตของโมดูล MCPWM จะหยุดทำงานทั้งหมด และเกิดการอินเตอร์รัปต์ขึ้น ถ้ามีการเอ็นเอเบิลไว้หลังจากนั้นส่วนกำเนิดสัญญาณ PWM จะหยุดทำงาน จนกว่าบิต PTEN จะถูกเซตด้วยซอฟต์แวร์อีกครั้ง

2.4.2.2.2 การทำงานของส่วนกำเนิดสัญญาณ PWM ในโหมดปรับกึ่งสัญญาณ

โมดูล MCPWM จะกำเนิดสัญญาณพัลส์ PWM ในโหมดนี้เมื่อฐานเวลา PWM ถูกำหนดให้ทำงานในโหมดนับค่าขึ้นลงอย่างต่อเนื่อง (ค่าของ PTMOD1 และ PTMOD0 เท่ากับ "10" หรือ "11") ในรูปที่ 14-7 แสดงไดอะแกรมเวลาของการกำเนิดสัญญาณ PWM ในโหมดนี้ วงจรเอาต์พุตของโมดูล MCPWM จะอยู่ในสภาวะทำงานเมื่อค่าของรีจิสเตอร์ PDCx เท่ากับค่าของ PTMR และฐานเวลา PWM อยู่ในภาวะนับลง (บิต PTDIR เป็น "1") วงจรเอาต์พุตของโมดูล MCPWM จะอยู่ในภาวะหยุดทำงานเมื่อฐานเวลา PWM อยู่ในภาวะนับขึ้น (บิต PTDIR เป็น "0") และค่าของรีจิสเตอร์ PTMR เท่ากับค่าของรีจิสเตอร์กำหนดค่าดิวตี้ไซเกิล



รูปที่ 2.7 ไดอะแกรมเวลาของการเกิดสัญญาณ PWM ในโหมดปรับสัญญาณกึ่งกลาง

ถ้าหากค่าในรีจิสเตอร์ PDCx เป็นศูนย์ วงจรเอาต์พุตของส่วนกำเนิดสัญญาณ PWM จะไม่ทำงานใด ๆ นั่นหมายความว่า ในโหมดนี้จะสามารถกำเนิดสัญญาณ PWM ได้ก็ต่อเมื่อค่าในรีจิสเตอร์ PDCx ต้องมากกว่าค่าที่กำหนดในรีจิสเตอร์ PTPER

2.4.3 ส่วนควบคุม

หลักการควบคุม คือการรับข้อมูลจะ Visual Basic 6 ที่ส่งผ่านพอร์ตอนุกรม มาทำการตรวจสอบข้อมูลว่าตรงกับกรณีไหน ก็จะให้ dsPic ทำคำสั่งที่อยู่ในกรณีนั้นๆ

ในการควบคุมนั้น เราแบ่งเป็น 2 อย่าง คือ

1. ควบคุมมอเตอร์โดยนำสัญญาณจาก dsPic มา AND กับ สัญญาณ PWM โดยใช้ AND GATE เบอร์ CD4081
2. ควบคุมมอเตอร์โดยนำสัญญาณจาก dsPic จ่ายผ่าน L293D

2.4.3.1 ควบคุมมอเตอร์โดยนำสัญญาณจาก dsPic มา AND กับ สัญญาณ PWM โดยใช้ AND GATE เบอร์ CD4081

นำข้อมูลจาก X3 และ X4 มา AND กัน โดยข้อมูล X3 นั้น เป็นการควบคุมการเคลื่อนที่รถ

- X3 = 1 เป็นการสั่งให้หุ่นยนต์เคลื่อนที่ไปข้างหน้า ขา RB0 , RB1 จะเป็น high และ ขา RB2 RB3 จะเป็น low
- X3 = 2 เป็นการสั่งให้หุ่นยนต์เคลื่อนที่ไปข้างหลัง ขา RB2 , RB3 จะเป็น high และ ขา RB0 , RB1 จะเป็น low
- X3 = 3 เป็นการสั่งให้หุ่นยนต์เคลื่อนที่ไปทางซ้าย ขา RB0 , RB3 จะเป็น high และ ขา RB1 , RB2 จะเป็น low
- X3 = 4 เป็นการสั่งให้หุ่นยนต์เคลื่อนที่ไปทางขวา ขา RB1 , RB2 จะเป็น high และ ขา RB0 , RB3 จะเป็น low

จากนั้น นำขา RB0 , RB1 , RB2 , RB3 ไปเข้า AND GATE เพื่อที่จะ นำแต่ละขามา AND กับสัญญาณ PWM เมื่อทำการ AND กันเสร็จแล้วจะส่งสัญญาณ ออกมา 4 เส้น แสดงตามตารางข้างล่างนี้

การเคลื่อนที่	RB0	RB1	RB2	RB3	PWM	M1	M2	M3	M4
ไปข้างหน้า	1	1	0	0	1	1	1	0	0
ถอยหลัง	0	0	1	1	1	0	0	1	1
เลี้ยวซ้าย	1	0	0	1	1	1	0	0	1
เลี้ยวขวา	0	1	1	0	1	0	1	1	0

ตาราง 2.1 แสดงผลการ AND กันของ RB0 , RB1 ,RB2 , RB3 กับ สัญญาณ PWM

M1 = ค่าที่ได้จากการ AND กันของ RB0 กับ สัญญาณ PWM

M2 = ค่าที่ได้จากการ AND กันของ RB1 กับ สัญญาณ PWM

M3 = ค่าที่ได้จากการ AND กันของ RB2 กับ สัญญาณ PWM

M4 = ค่าที่ได้จากการ AND กันของ RB3 กับ สัญญาณ PWM

โดยปกติเวลาขาของ dsPic เป็น สถานะ High จะส่งแรงดันไฟฟ้ามีค่า 5 Volt ออกมา และเมื่อเป็น สถานะ Low จะส่งแรงดันไฟฟ้า 0 Volt ออกมา ส่วน PWM จะส่งแรงดันได้ตั้งแต่ 0 – 5 Volt แล้วแต่ ค่า Duty cycle ว่าเป็นกี่เปอเซ็น

ตัวอย่าง ให้ค่า PWM เป็น 50% หรือ เท่ากับ 2.5 volt เมื่อทำการ AND กันจะได้ค่าตามตาราง

การเคลื่อนที่	RB0	RB1	RB2	RB3	PWM	M1	M2	M3	M4
ไปข้างหน้า	5	5	0	0	2.5	2.5	2.5	0	0
ถอยหลัง	0	0	5	5	2.5	0	0	2.5	2.5
เลี้ยวซ้าย	1	0	0	5	2.5	2.5	0	0	2.5
เลี้ยวขวา	0	5	5	0	2.5	0	2.5	2.5	0

ตาราง 2.2 แสดงการ AND กัน เมื่อค่า PWM = 25 %

จากนั้นเราจะนำค่า M1 ,M2 , M3 , M4 ไปจ่ายให้กับวงจร ขับมอเตอร์

บทที่3
การออกแบบและการทำงาน

3.1 ภาพรวมของโครงงาน

ในโครงงานนี้เป็นการรวมการ สั่งงานและการแสดงผลค่าต่างๆ ของการควบคุมความเร็วของมอเตอร์ มารวมกันบนหน้าจอคอมพิวเตอร์ เพื่อความสะดวก และรวดเร็วในการประมวลผล การทำงาน การสั่งงานการเก็บบันทึกค่าสถานะการทำงาน ในการออกแบบนั้นจึงแบ่งได้ 2 ส่วน คือ ส่วนการควบคุมการสั่งงาน และส่วนปฏิบัติการ

3.1.1 ส่วนการออกแบบ

ส่วนของโครงสร้าง

ล้อ

ล้อที่ใช้จะเป็นแบบสายพาน ซึ่งทำจากโซ่ และยาง ประกอบรวมกัน การที่ทำล้อเป็นแบบสายพานเพราะว่าสามารถ เคลื่อนที่ได้ทุกสภาพสนาม โดยล้อจะทำเป็น 2 ส่วน คือ ส่วนล้อหน้าและส่วนล้อหลัง

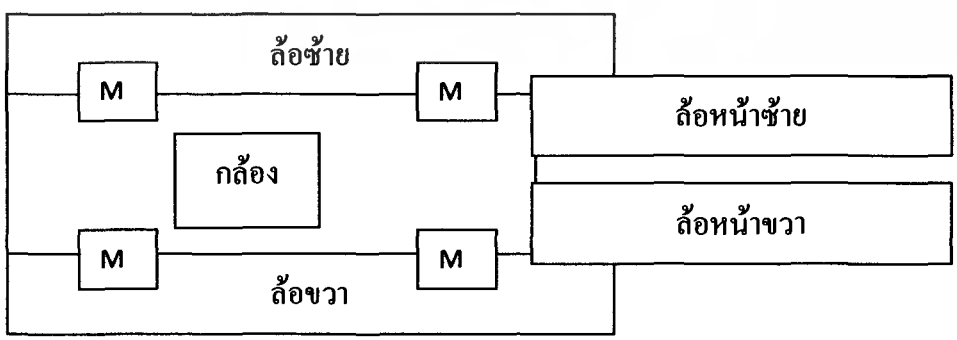
หน้าที่ ส่วนล้อหน้าจะเป็นตัวปีนสภาพสนามที่สูงกว่าตัวหุ่นโดยจะใช้มอเตอร์ กระแสตรง 90W, 15Kg-cm 2 ตัว ข้างละ1ตัว

หน้าที่ ส่วนล้อหลังจะเป็นตัวขับเคลื่อน ซึ่ง ส่วนนี้จะใช้มอเตอร์กระแสตรงทั้งหมด 4 ตัว ตัวละ 40W ,10 Kg-cm แบ่งเป็นข้างละ 2 ตัว ต่อขนานกัน

กล้อง

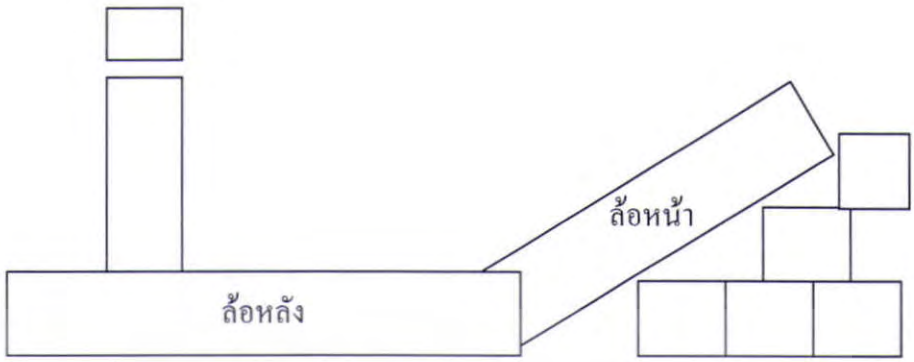
ส่วนกล้องจะหมุนได้ 360 องศา ขึ้น –ลง ได้ 180 องศา ทำให้สามารถ มองเห็นเส้นทางการเคลื่อนที่ และเหยื่อผู้ประสบภัยในสนามได้

Top view



รูปที่ 3.1 แสดงแบบจำลองหุ่นมองจากด้านบน

Side view



รูปที่ 3.2 แสดงแบบจำลองหุ่นจากด้านข้าง

3.1.2 ส่วนควบคุมและสั่งงาน

ในส่วนนี้จะอยู่บนหน้าจอคอมพิวเตอร์ เพื่อความสะดวกในการสั่งงาน และการดูผลการทำงาน จึงเลือกใช้โปรแกรม Visual Basic 6

3.1.3 ส่วนปฏิบัติงาน

ส่วนนี้จะเป็นตัวโครงสร้างของหุ่นยนต์ ซึ่งประกอบด้วยส่วนต่าง เช่น มอเตอร์ ซึ่งมีตัวคอนโทรลเลอร์ที่รับสัญญาณควบคุมจากคอมพิวเตอร์

รูปที่ 3.3 รูปตัวหุ่นยนต์

3.2 การเชื่อมต่อไร้สาย

3.2.1 โมดูลรับส่งสัญญาณ

โมดูลที่ใช้ในการรับส่งสัญญาณควบคุมในการทดลองนี้ ใช้ Access point 2.4GHz ยี่ห้อ Linksys

3.2.2 โมดูลแปลงสัญญาณเข้า คอนโทรลเลอร์

Ethernet I/O เป็นตัวแปลงสัญญาณจากคอนโทรลเลอร์เป็นสัญญาณ wireless เพื่อที่จะใช้ในการสื่อสารกับคอมพิวเตอร์

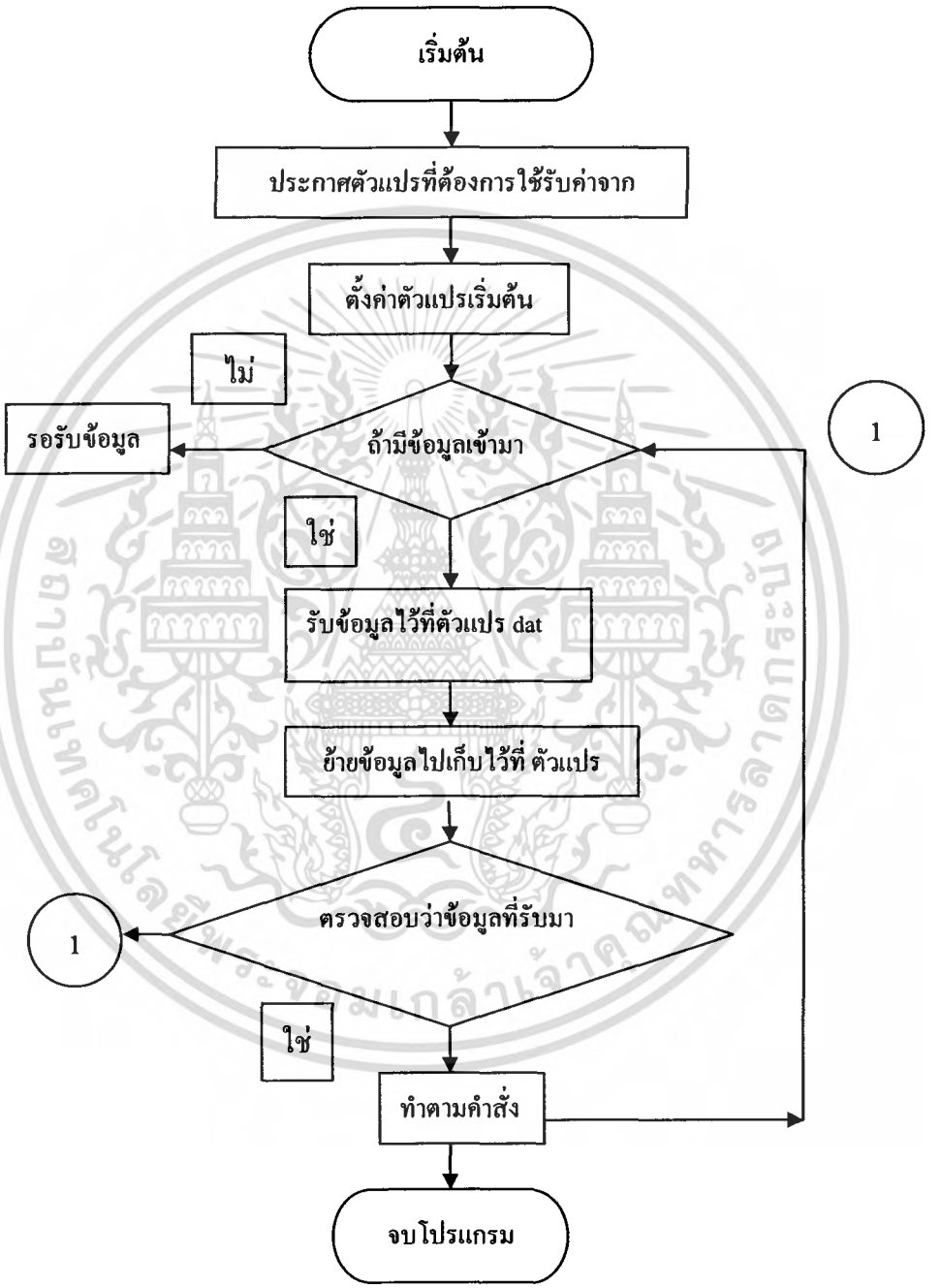
3.3 ส่วนของการมองของหุ่นยนต์

ส่วนของกล้องนั้นได้เลือก กล้องไร้สาย (Wireless Video Cacera) กล้องวีดีโอไร้สายรุ่น LYD208CV พร้อมตัวรับกำลังส่ง 50 mW ซึ่งสามารถต่อใช้งานร่วมกับโทรทัศน์ AV IN ได้โดยตรง

3.4 ส่วนโปรแกรมควบคุม

3.4.1 การออกแบบโปรแกรมในส่วน dsPic30f4011

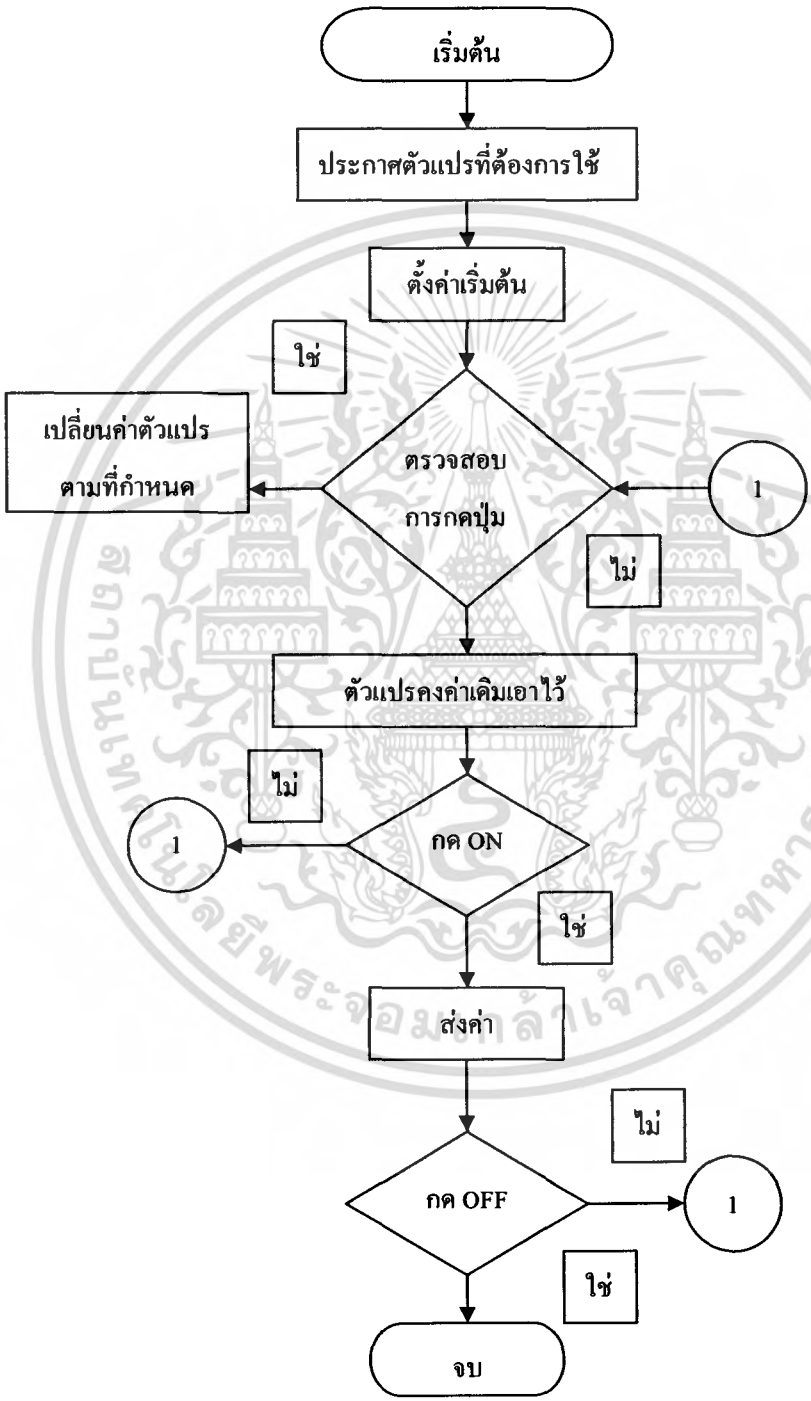
ในการออกแบบนี้สามารถอธิบายได้ด้วยโฟลชาร์ตข้างล่างนี้



3.4.2 การออกแบบโปรแกรมในส่วน Visual Basic 6

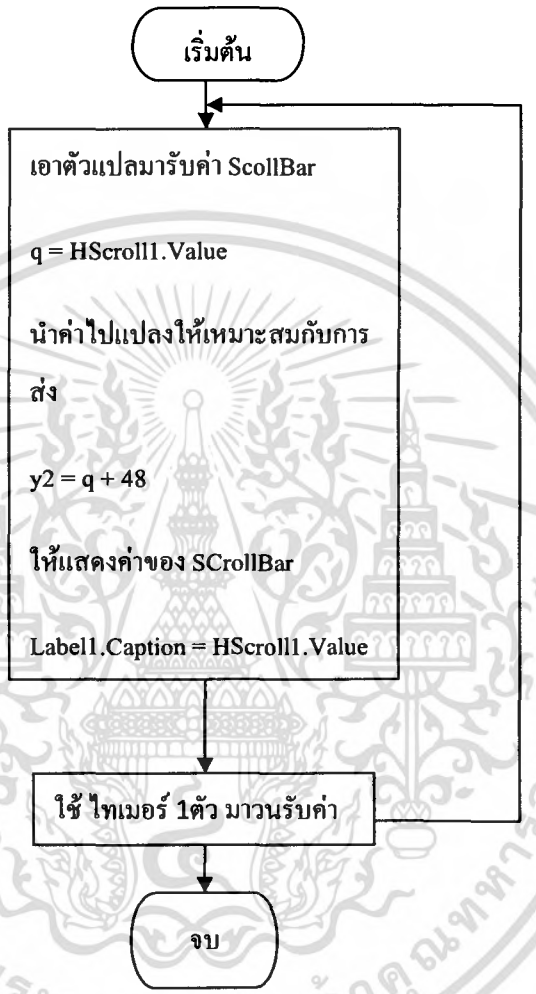
ในการออกแบบนี้สามารถอธิบายได้ด้วยโฟลชาร์ตข้างล่างนี้

โฟลชาร์ตแสดงการส่งข้อมูลเป็นชุดผ่านพอร์ตอนุกรม

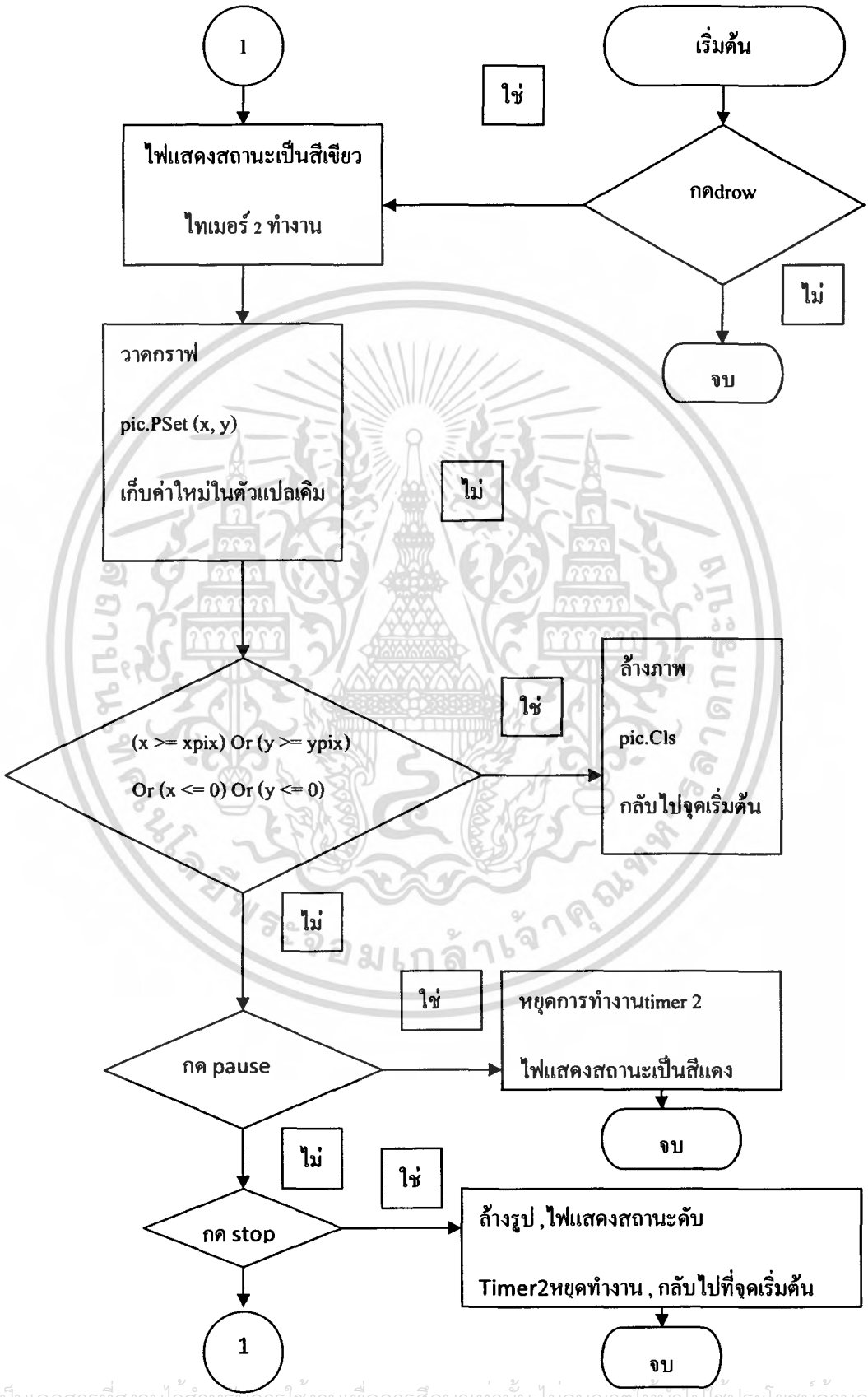


โฟลชาร์ตแสดงการปรับค่า PWM

ในโครงงานนี้ได้ใช้ SCrollBar มาเป็นตัวปรับค่า PWM สามารถปรับได้ตั้งแต่ 0 – 100 %

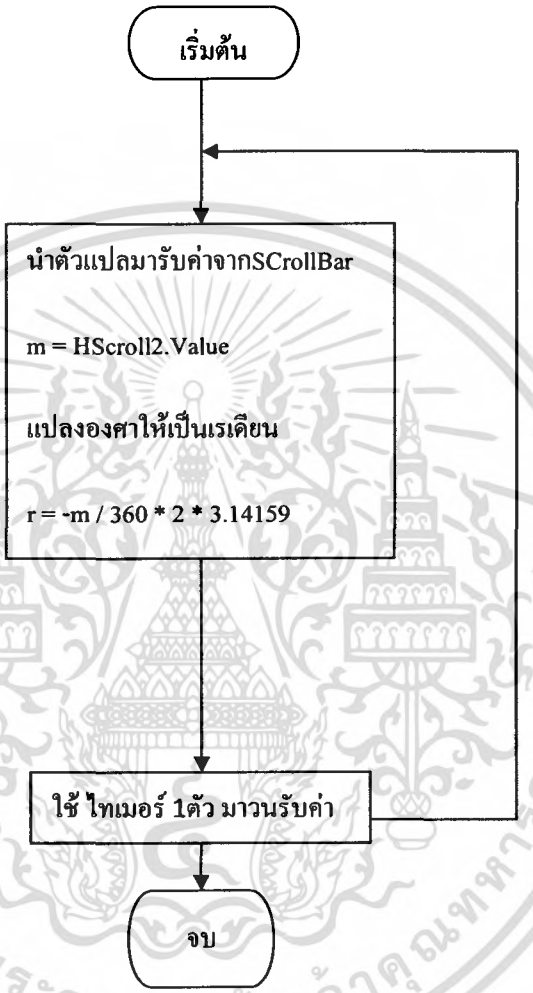


โฟลชาร์ตแสดงการวาดแผนที่

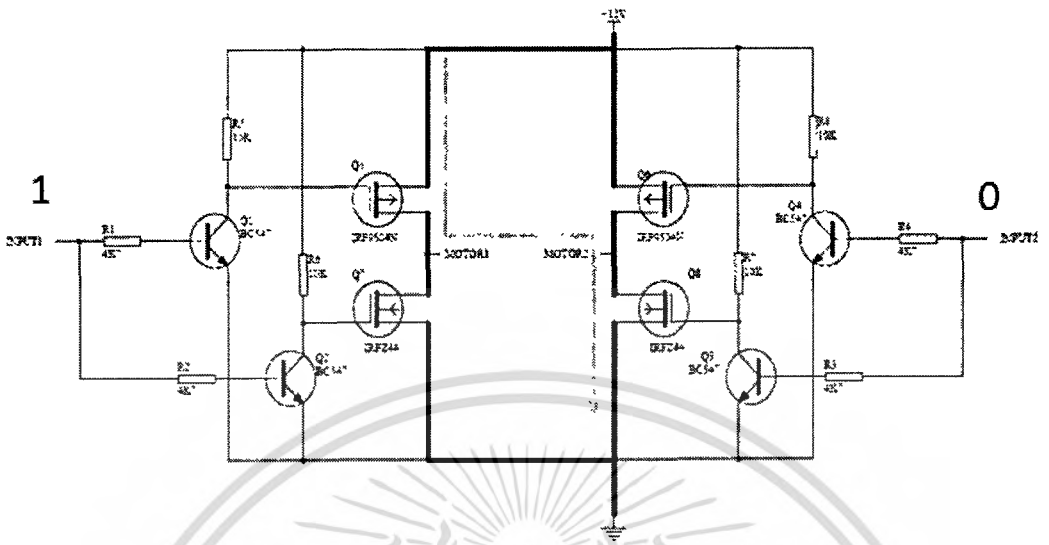


โฟลชาร์ตแสดงการกำหนดค่ามุม

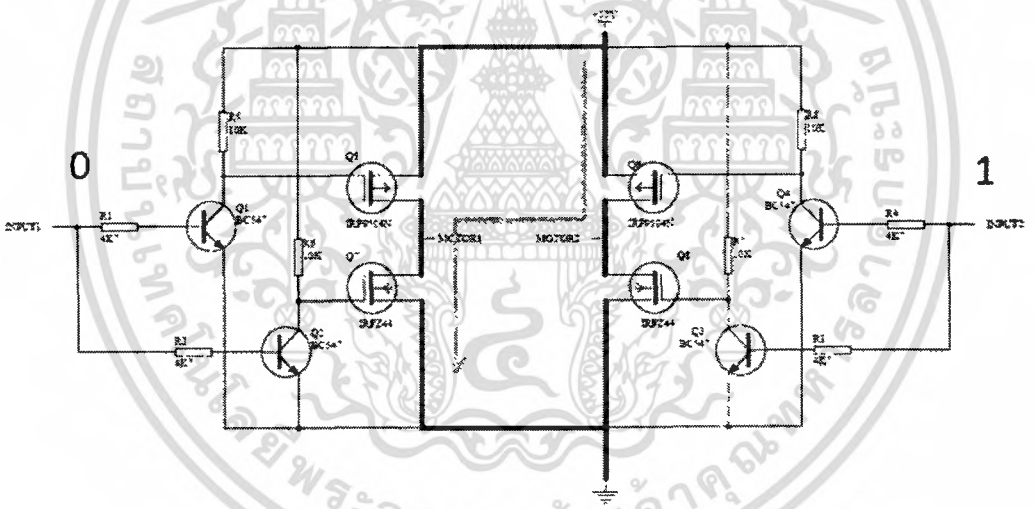
ในโครงงานนี้ได้ใช้ SCrollBar มาเป็นตัวปรับค่ามุม สามารถปรับได้ตั้งแต่ 0 – 360 องศา



3.5 วงจรขับมอเตอร์



รูปที่ 3.4 แสดงทิศทางตามเข็มนาฬิกาของมอเตอร์

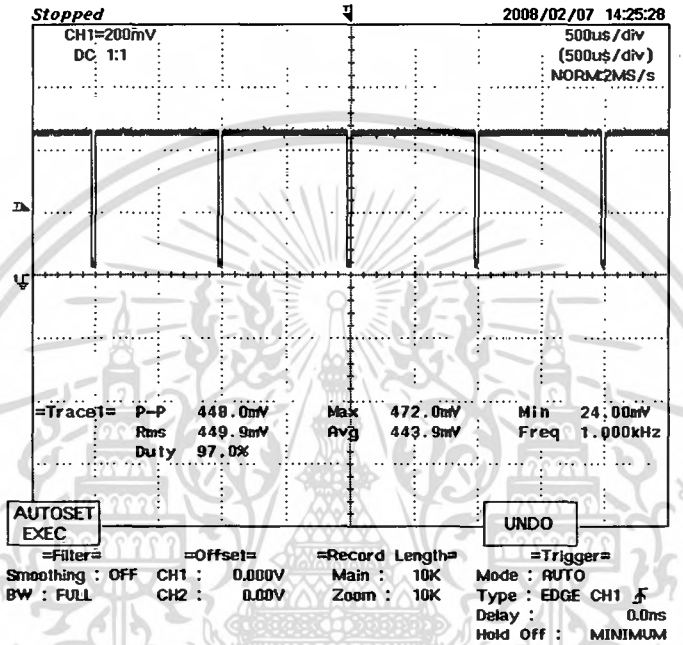


รูปที่ 3.5 แสดงทิศทางทวนเข็มนาฬิกาของมอเตอร์

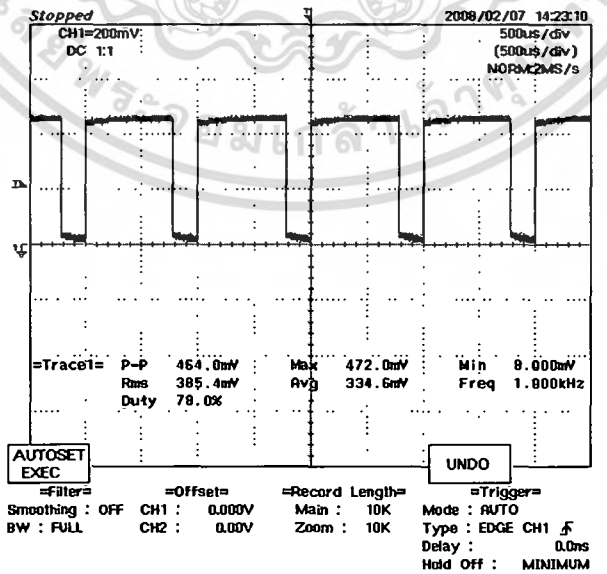
บทที่4

การทดลอง

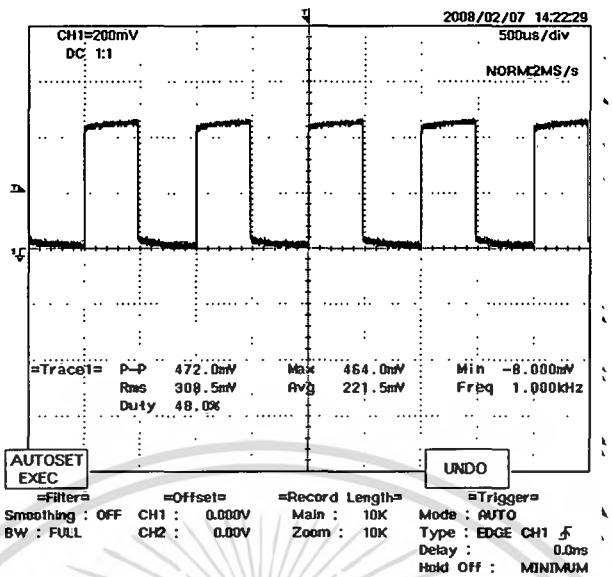
4.1สัญญาณควบคุมความเร็วมอเตอร์กระแสตรง



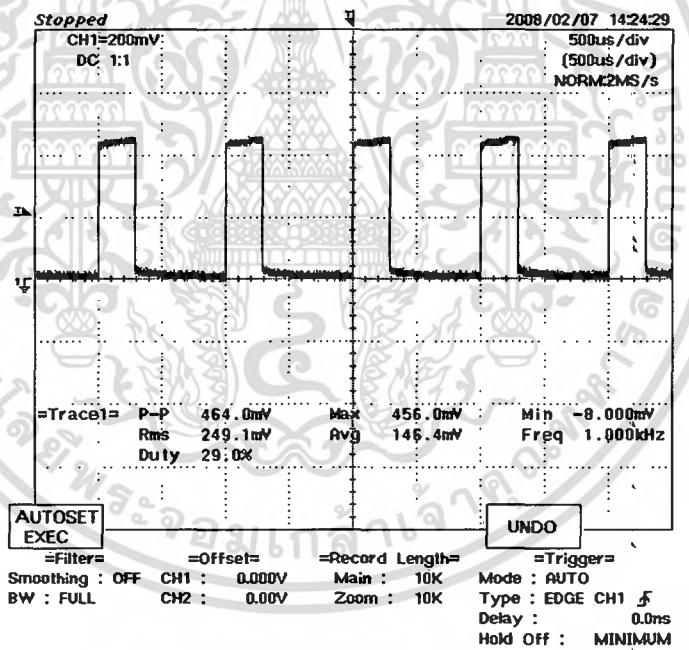
รูปที่ 4.1 แสดงกราฟสัญญาณพลัส 100%



รูปที่ 4.2 แสดงกราฟสัญญาณพลัส 80%



รูปที่4.3 แสดงกราฟสัญญาณพลัส์ 50%



รูปที่ 4.4 แสดงกราฟสัญญาณพลัส์ 30 %

บทที่ 5
บทวิจารณ์และสรุป

ในการทำปริญญานิพนธ์ฉบับนี้ ได้จัดทำขึ้นเพื่อการควบคุมกระบวนการแบบไร้สาย การควบคุมความเร็วมอเตอร์กระแสตรง ซึ่งสามารถทำการควบคุม สั่งงาน และบันทึกทิศทางการเคลื่อนที่ของหุ่นยนต์ ซึ่งในการส่งข้อมูล รับข้อมูลจะเป็นแบบไร้สาย โดยผ่านตัวส่ง Access point Wrieless – G (802.11g) 2.4GHz การควบคุมผ่านหน้าจอคอมพิวเตอร์ซึโดยใช้โปรแกรม Visual Basic ควบคุมการทำงาน

จากการทดลองระบบควบคุมไร้สายจะเห็นว่า สามารถควบคุมความเร็วของมอเตอร์ได้ตามเป้าหมาย ทำให้ความเร็วและแรงบิดของมอเตอร์คงที่ ส่วนของการควบคุมการเคลื่อนที่ของกล้องสามารถควบคุมได้ตามทิศทางที่ต้องการ และสามารถนำการเคลื่อนที่ของหุ่นยนต์กู้ภัย มาจำลองเป็นแผนที่ได้

5.1 ปัญหาที่พบและแนวทางแก้ไข

1. ตัวหุ่นใหญ่เกินไปทำให้เคลื่อนที่ในสภาพที่ขรุขระ ได้ยาก
2. แรงบิดของมอเตอร์ น้อยเกินไปทำให้การปีนที่สูงทำไม่ได้
3. เครือข่ายการเชื่อมต่อ เวลาอยู่บริเวณที่มีสัญญานรบกวน ทำให้การเชื่อต่อมีปัญหา
4. อุปกรณ์ในการทำงานแพงมาก
5. โปรแกรม VB ไม่สามารถทำงานทั้งหมด ได้พร้อมกัน เช่น เวลาสั่งเคลื่อนที่ กด Move จากนั้น ถ้าต้องการเอาเคอเซอร์ไปปรับค่า PWM คำสั่ง Move จะโดนยกเลิก ต้องสั่ง Move ใหม่อีกครั้ง

5.2 วิธีแก้ไขปัญหา

1. ออกแบบโครงสร้าง ให้เล็กลงและง่ายต่อการเคลื่อนที่ทุกสภาพสนาม
2. ติดตั้งมอเตอร์เพิ่ม และใช้มอเตอร์ชนิดที่มีแรงบิดมากๆ
3. ตั้งระบบความปลอดภัยที่ Access Point เพื่อป้องกันการรบกวนของสัญญาณ
4. ลดรายจ่าย ใช้อุปกรณ์ที่พอจะใช้ได้
5. เปลี่ยนโปรแกรมสั่งงานเป็น Visual C++

ภาคผนวก

โปรแกรมใน Visual Basic 6

Option Explicit

Dim y1 As Integer	ประกาศตัวแปร y1 เป็น Integer
Dim y2 As Integer	ประกาศตัวแปร y2 เป็น Integer
Dim y3 As Integer	ประกาศตัวแปร y3 เป็น Integer
Dim y4 As Integer	ประกาศตัวแปร y4 เป็น Integer
Dim y5 As Integer	ประกาศตัวแปร y5 เป็น Integer
Dim q As Integer	ประกาศตัวแปร q เป็น Integer
Dim code As String	ประกาศตัวแปร y1 เป็น String
Dim exp1 As Integer	ประกาศตัวแปร y1 เป็น Integer
Dim exp2 As Integer	ประกาศตัวแปร y1 เป็น Integer
Dim exp3 As Integer	ประกาศตัวแปร y1 เป็น Integer
Dim exp4 As Integer	ประกาศตัวแปร y1 เป็น Integer
Dim exp5 As Integer	ประกาศตัวแปร y1 เป็น Integer
Dim xpix As Double	ประกาศตัวแปร y1 เป็น Double
Dim ypix As Double	ประกาศตัวแปร y1 เป็น Double
Dim x As Double	ประกาศตัวแปร y1 เป็น Double
Dim y As Double	ประกาศตัวแปร y1 เป็น Double
Dim m As Double	ประกาศตัวแปร y1 เป็น Double
Dim r As Double	ประกาศตัวแปร y1 เป็น Double
Dim speed As Double	ประกาศตัวแปร y1 เป็น Double

```
Private Sub drow_Click()                    เมื่อกดปุ่ม drow
Shape2.BackColor = vbGreen                  ให้ Shape2.BackColor เป็นสีเขียว
Timer2.Enabled = True                       Timer2 ทำงาน
End Sub
```

```
Private Sub Form_Load()
pic.ScaleMode = vbPixels                    กำหนดสเกลเป็น pixels
xpix = pic.ScaleWidth – 1                   หาจำนวน pixelทั้งหมดตามแกนx -1
ypix = pic.ScaleHeight – 1                  หาจำนวน pixels ทั้งหมดตามแกน y - 1
x = xpix / 2
y = ypix / 2
y1 = 48                                     กำหนดค่าเริ่มต้นของ y1
y2 = 53                                     กำหนดค่าเริ่มต้นของ y2
y3 = 48                                     กำหนดค่าเริ่มต้นของ y3
y4 = 48                                     กำหนดค่าเริ่มต้นของ y4
y5 = 48                                     กำหนดค่าเริ่มต้นของ y5
MSComm1.CommPort = 7                        กำหนด ค่า port ของ ซีเรียส port
MSComm1.Settings = "9600,N,8,1"             กำหนด บอดเรด , ไม่มีไพริตี้,8bit, 1 stop bit
MSComm1.PortOpen = True                     เปิดport
End Sub
```

```
Private Sub move_KeyDown(KeyCode As Integer, Shift As Integer)    เมื่อกดปุ่ม
Select Case KeyCode                         ตรวจสอบการกดคีย์บอร์ด
Case vbKeyW                                 กรณีที่กด W
Text1.Text = "Forward"                      ให้ text1 แสดงคำว่า Forward
y3 = 49                                     ให้ y3 = 49
Case vbKeyS                                 กรณีที่กด S
```

```
Text1.Text = "Reward"                ให้ text1 แสดงคำว่า Reward
y3 = 50                              ให้ y3 = 50
Case vbKeyA                          กรณีที่กด A
Text1.Text = "Turnleft"              ให้ text1 แสดงคำว่า Turnleft
y3 = 51                              ให้ y3 = 51
Case vbKeyD                          กรณีที่กด D
Text1.Text = "Turnright"             ให้ text1 แสดงคำว่า Turnright
y3 = 52                              ให้ y3 = 52
Case vbKeyR                          กรณีที่กด R
Text1.Text = "Hand up"               ให้ text1 แสดงคำว่า Hand up
y5 = 49                              ให้ y5 = 49
Case vbKeyF                          กรณีที่กด F
Text1.Text = "Hand down"             ให้ text1 แสดงคำว่า Hand down
y5 = 50                              ให้ y5 = 50
Case vbKeyU                          กรณีที่กด U
Text1.Text = "Camera up"             ให้ text1 แสดงคำว่า Camera up
y4 = 49                              ให้ y4 = 49
Case vbKeyJ                          กรณีที่กด J
Text1.Text = "Camera down"           ให้ text1 แสดงคำว่า Camera down
y4 = 50                              ให้ y4 = 50
Case vbKeyK                          กรณีที่กด K
Text1.Text = "Camera right"          ให้ text1 แสดงคำว่า Camera right
y1 = 49                              ให้ y1 = 49
Case vbKeyH                          กรณีที่กด H
Text1.Text = "Camera left"           ให้ text1 แสดงคำว่า Camera left
y1 = 50                              ให้ y1 = 50
End Select
```

```
End Sub
```

```
Private Sub move_KeyUp(KeyCode As Integer, Shift As Integer)   เมื่อปล่อยปุ่ม
Text1.Text = ""                                                ให้ Text1 ไม่แสดงอะไรเลย
y3 = 48                                                        ให้ y3 = 48
y4 = 48                                                        ให้ y4 = 48
y5 = 48                                                        ให้ y5 = 48
y1 = 48                                                        ให้ y1 = 48
End Sub
```

```
Private Sub off_Click()                                        เมื่อคลิ๊กที่ off
t1.Enabled = False                                             t1 หยุดทำงาน
Shape1.BackColor = vbWhite                                     ให้ Shape1.BackColor เป็นสีขาว
End Sub
```

```
Private Sub on_Click()                                         เมื่อคลิ๊ก on
t1.Enabled = True                                              t1 ทำงาน
Shape1.BackColor = vbGreen                                     ให้ Shape1.BackColor เป็นสีเขียว
End Sub
```

```
Private Sub pause_Click()                                      เมื่อกด pause
Timer2.Enabled = False                                         Timer2 หยุดทำงาน
Shape2.BackColor = vbRed                                       ให้ Shape2.BackColor เป็นสีแดง
End Sub
```

```
Private Sub t1_Timer()
Select Case t1.Interval                 เลือกค่า Intcrval ของ t1
Case 1                                  กรณี Interval = 1
   t1.Interval = 2                      ให้ t1 มี Interval = 2
   MSComm1.Output = Chr(88)             ส่งค่า chr(88) ออกทาง ซีเรียส port
Case 2                                  กรณี Interval = 2
   t1.Interval = 3                      ให้ t1 มี Interval = 3
   MSComm1.Output = Chr(y5)             ส่งค่า chr(y5) ออกทาง ซีเรียส port
Case 3                                  กรณี Interval = 3
   t1.Interval = 4                      ให้ t1 มี Interval = 4
   MSComm1.Output = Chr(y4)             ส่งค่า chr(y4) ออกทาง ซีเรียส port
Case 4                                  กรณี Interval = 4
   t1.Interval = 5                      ให้ t1 มี Interval = 5
   MSComm1.Output = Chr(y3)             ส่งค่า chr(y3) ออกทาง ซีเรียส port
Case 5                                  กรณี Interval = 5
   t1.Interval = 6                      ให้ t1 มี Interval = 6
   MSComm1.Output = Chr(y2)             ส่งค่า chr(y2) ออกทาง ซีเรียส port
Case 6                                  กรณี Interval = 6
   t1.Interval = 7                      ให้ t1 มี Interval = 7
   MSComm1.Output = Chr(y1)             ส่งค่า chr(y1) ออกทาง ซีเรียส port
Case 7                                  กรณี Interval = 7
   t1.Interval = 1                      ให้ t1 มี Interval = 1
   MSComm1.Output = Chr(89)             ส่งค่า chr(89) ออกทาง ซีเรียส port
End Select
End Sub
```

```
Private Sub t2_Timer()
q = HScroll1.Value                    ให้ q = ค่าของ HScrll1
y2 = q + 48
Label1.Caption = HScroll1.Value       Label1 แสดงค่า HScrll1
Timer2.Interval = 1100 - (q * 100)    ให้ค่า Interval ของ Timer2 = 1100 – (q*100)
End Sub
```

```
Private Sub Timer2_Timer()
pic.PSet (x, y)                       วาดจุด (x,y)
x = x + Cos(r)
y = y + Sin(r)
If ((x >= xpix) Or (y >= ypix) Or (x <= 0) Or (y <= 0)) Then    ตรวจสอบไม่ให้ตกขอบ
Pic.Cls                               เคลียร์รูป
x = xpix / 2                          กำหนดจุดเริ่มต้นใหม่
y = ypix / 2                          กำหนดจุดเริ่มต้นใหม่
End If
End Sub
```

```
Private Sub Timer3_Timer()
m = HScroll2.Value                    ให้ m = ค่า HScroll2
r = -m / 360 * 2 * 3.14159            คำนวณมุม
Label7.Caption = HScroll2.Value       ให้Label7แสดงค่า HScroll2
End Sub
```

```
Private Sub stop_Click()              เมื่อกด stop
pic.Cls                               เคลียร์รูป
Shape2.BackColor = vbWhite            ให้ Shape2.BackColor เป็นสีขาว
```

```
Timer2.Enabled = False          หยุดงาน Timer2
x = xpix / 2                    กำหนดจุดเริ่มต้นใหม่
y = ypix / 2                    กำหนดจุดเริ่มต้นใหม่
End Sub
```

โปรแกรมในdsPic30f4011

```
#include<p30f4011.h>                    // Header file for dsPIC30F2010
#include<uart.h>                        // Module function for uart
#include <pwm.h>                        // Module function for pwm
#define PERIOD 0x0E65                   //Set period
#define POW 50                          //Fix pow 50%
_FOSC(CSW_FSCM_OFF & XT_PLL4); //Set ossilator
_FWDT(WDT_OFF);                         //Off watch dog timer
unsigned int speed = 50;
char  dat,x0,x1,x2,x3,x4,x5,x6,dut,comp;
unsigned int res,res1,res2,res3,res4;
char con0,con1,con2,con3,con4;
void _ISR _U1TXInterrupt(void)
{
        IFS0bits.U1TXIF = 0;            //Clear TX interrupt flag
}
void _ISR _U1RXInterrupt(void)
{
        IFS0bits.U1RXIF = 0;            //Clear RX interrupt flag
}
void _ISR _PWMInterrupt(void)
{
        IFS2bits.PWMIF = 0;             //Clear PWM interrupt flag
}
void Forward()
{
```

```
        LATBbits.LATB0 = 1;          // Configuration for RB0 to high
        LATBbits.LATB1 = 1;          // Configuration for RB1 to high
        LATBbits.LATB2 = 0;          // Configuration for RB2 to low
        LATBbits.LATB3 = 0;          // Configuration for RB3 to low
}
void Reward()
{
        LATBbits.LATB0 = 0;          // Configuration for RB0 to low
        LATBbits.LATB1 = 0;          // Configuration for RB1 to low
        LATBbits.LATB2 = 1;          // Configuration for RB2 to high
        LATBbits.LATB3 = 1;          // Configuration for RB3 to high
}
void Turnleft()
{
        LATBbits.LATB0 = 0;          // Configuration for RB0 to low
        LATBbits.LATB1 = 1;          // Configuration for RB1 to high
        LATBbits.LATB2 = 1;          // Configuration for RB2 to high
        LATBbits.LATB3 = 0;          // Configuration for RB3 to low
}
void Turnright()
{
        LATBbits.LATB0 = 1;          // Configuration for RB0 to high
        LATBbits.LATB1 = 0;          // Configuration for RB1 to low
        LATBbits.LATB2 = 0;          // Configuration for RB2 to low
        LATBbits.LATB3 = 1;          // Configuration for RB3 to high
}
void Stop()
```

```
{
        LATBbits.LATB0 = 0;          // Configuration for RB0 to low
        LATBbits.LATB1 = 0;          // Configuration for RB1 to low
        LATBbits.LATB2 = 0;          // Configuration for RB2 to low
        LATBbits.LATB3 = 0;          // Configuration for RB3 to low
}
void Stophand()
{
        LATBbits.LATB6 = 0;          // Configuration for RB6 to low
        LATBbits.LATB7 = 0;          // Configuration for RB7 to low
}
void hand_up()
{
        LATBbits.LATB6 = 1;          // Configuration for RB6 to high
        LATBbits.LATB7 = 0;          // Configuration for RB7 to low
}
void hand_down()
{
        LATBbits.LATB6 = 0;          // Configuration for RB6 to low
        LATBbits.LATB7 = 1;          // Configuration for RB7 to high
}
void camera_up()
{
        LATBbits.LATB4 = 1;          // Configuration for RB4 to high
        LATBbits.LATB5 = 0;          // Configuration for RB5 to low
}
void camera_down()
```

```
{
        LATBbits.LATB4 = 0;          // Configuration for RB4 to low
        LATBbits.LATB5 = 1;          // Configuration for RB5 to high
}
void camera_right()
{
        LATDbits.LATD0 = 1;          // Configuration for RD0 to high
        LATDbits.LATD1 = 0;          // Configuration for RD1 to low
}
void camera_left()
{
        LATDbits.LATD0 = 0;          // Configuration for RD0 to low
        LATDbits.LATD1 = 1;          // Configuration for RD1 to high
}
void Stopcamera1()
{
        LATDbits.LATD0 = 0;          // Configuration for RD0 to low
        LATDbits.LATD1 = 0;          // Configuration for RD1 to low
}
void Stopcamera()
{
        LATBbits.LATB4 = 0;          // Configuration for RB4 to low
        LATBbits.LATB5 = 0;          // Configuration for RB5 to low
}
```

```
//--------------------------------------- Function initial PWM ---------------------------------------//
void pwm_init()
{
        unsigned int config;        // Holds the PWM interrupt configuration value
        unsigned int period;        // Holds the value to be loaded into dutycycle register
        unsigned int sptime;        // Holds the value to be loaded into special event compare
register
        unsigned int config1;       // Holds PWM configuration value
        unsigned int config2;       // Holds the value be loaded into PWMCON1 register
        unsigned int config3;       // Holds the value to configure the special event trigger
                                    // postscale and dutycycle
        unsigned int dutycyclereg;// The value of 'dutycyclereg' determines the duty cycle
        unsigned int dutycycle;     // register(PDCx) to be written
        unsigned char updatedisable;        // Configure pwm interrupt enable/disable and set
                                    //  interrupt priorties
```

```
//------------------------------------ Configuration for FLTA control-------------------------//

ConfigIntMCPWM(PWM_INT_EN &                 // Enable interrupt for PWM
               PWM_FLTA_DIS_INT &           // Disable interrupt for FLTA
               PWM_INT_PR0);                // FLTA interrupt priority level 0

SetMCPWMFaultA(PWM_OVA1H_ACTIVE &           // FLTA override 1H active
               PWM_OVA2H_ACTIVE &           // FLTA override 2H active
               PWM_FLTA_MODE_LATCH&         // FLTA latch mode
               PWM_FLTA1_DIS&               // Disable FLTA CH1
               PWM_FLTA2_DIS);              // Disable FLTA CH2
sptime = 0x0;
config1 =      (PWM_EN &                    // Enable PWM module
                PWM_OP_SCALE1      &        // Output post scaler select 1:1
                PWM_IPCLK_SCALE1 &          // Input prescaler select 1:1
                PWM_MOD_DBL);               // PWM double update mode

config2 =      (PWM_MOD1_IND &              // 1th channel in independent mode
                PWM_MOD2_IND &              // 2th channel in indipendent mode
                PWM_PEN2H &                 // H of channel 2 works as PWM
                PWM_PEN1H  &                // H of channel 1 works as PWM
                PWM_PEN2L &                 // L of channel 2 works as PWM
                PWM_PEN1L);                 // L of channel 1 works as PWM

config3 =      (PWM_SEVOPS1 &               // Special event post scaler 1:1
                PWM_OSYNC_PWM&              // over ride syncronised with PWM
```

```
                PWM_UEN);                    // Update of PDCs and PTPER
                                             //enabled

OpenMCPWM(PERIOD,sptime,config1,config2,config3);
                                             // Setup parameter for PWM module
}
//------------------------------ Function for configuration UART1---------------------------------//
void uart1_init()
{
unsigned int baudvalue;                      // Keep baud rate value for Load into U1BRG
unsigned int U1MODEvalue;                    // Keep value for Load into U1MODE
unsigned int U1STAvalue;                     // Keep value for Load into U1STA

CloseUART1();                                // Disable UART1
ConfigIntUART1(UART_RX_INT_EN &              // Enable RX interrupt UART1
        UART_RX_INT_PR6 &                    // Set RX interrupt Priority ==>6
        UART_TX_INT_EN &                     // Enable TX interrupt UART1
        UART_TX_INT_PR2);                    // Set RX interrupt Priority ==>2
baudvalue = 47;                              // Baud rate 9600 bps

U1MODEvalue =   UART_EN &                    // Enable UART1
                UART_IDLE_CON &              // UART1 working in idle mode
                UART_RX_TX&                  // UART1 normal pin(TX==>RF3
                                             pin,RX==>RF2 pin)
                UART_DIS_WAKE &              // Disable wake-up on start UART
                UART_DIS_LOOPBACK &          // Disable loop back mode
                UART_DIS_ABAUD &             // Disable autobaud mode
```

```
                    UART_NO_PAR_8BIT &          // Set data 8 bit ,no parity bit
                    UART_1STOPBIT;              // Set 1 stop bit

U1STAvalue    = UART_INT_TX_BUF_EMPTY &     // Interrupt on buffer empty mode
                UART_TX_PIN_NORMAL &        // TX Break bit normal
                UART_TX_ENABLE &            // Enable TX for UART
                UART_INT_RX_3_4_FUL&        // UART interrupt receive mode
                UART_ADR_DETECT_DIS &       // Disable detect address mode
                UART_RX_OVERRUN_CLEAR;      // Reset buffer over run

OpenUART1(U1MODEvalue, U1STAvalue, baudvalue);// Execute configuration for UART1

}
//------------------ ------------------Function PWM ----------------------------------------//
void pwm(char chanel,char pow)
{
        unsigned int dutycycle,updatedisable;

        dutycycle =2*(pow*(PERIOD/100)) ;           // Keep dutycycle

        updatedisable = 0;                          // Disable update

        SetDCMCPWM(chanel,dutycycle,updatedisable);
}
```

```
//-------------------------------- Main Pregram ----------------------------------------------------//
int main(void)
{

        pwm_init();                     // Initial PWM
        uart1_init();                   // Initial UART1
        x5=0x30;                        //Set x5 = 0x30
        x4=0x30;                        //Set x4 = 0x30
        x3=0x30;                        //Set x3 = 0x30
        x2=0x30;                        //Set x2 = 0x30
        x1=0x30;                        //Set x1 = 0x30
        x0=0x30;                        //Set x0 = 0x30
        x6=0x30;                        //Set x6 = 0x30
        TRISB = 0;                      // Configuration for RB to output
        TRISD = 0;                      // Configuration for RD to output
        LATB=0;                         // Configuration for RB Low
        TRISDbits.TRISD1 = 0;           // Configuration for RD1 to output
        TRISDbits.TRISD0 = 0;           // Configuration for RD1 to output

while(1)                                // Infinite loop
        {

                pwm(1,speed);           //Call Function pwm
                if(DataRdyUART1())      //Check data receive
                {
                        dat = ReadUART1(); // Load data into buffer variable
                        x6=x1;          //Move data
```

```
            x1=x2;                  //Move Mata
            x2=x3;                  //Move data
            x3=x4;                  //Move data
            x4=x5;                  //Move data
            x5=x0;                  //Move data
            x0=dat;                 //Move data
    if(x0==0x59&&x6==0x58)          //Check data
    {
            if(x3==0x31)            //Check data
            {
            Forward();              //Call Function Forward
            }
            else if (x3==0x32)      //Check data
            {
            Reward();               //Call Function Reward
            }
            else if (x3==0x33)      //Check data
            {
            Turnleft();             //Call Function Turnleft
            }
            else if(x3==0x34)       //Check data
            {
            Turnright();            //Call Function Turnright
            }
            else if (x2==0x31)      //Check data
            {
            camera_up();            //Call Function camera up
```

```
}
else if (x2==0x32)          //Check data
{
camera_down();              //Call Function camera down
}
else if (x5==0x31)          //Check data
{
camera_right();             //Call Function camera right
}
else if (x5==0x32)          //Check data
{
camera_left();              //Call Function camera left
}
else if (x1==0x31)          //Check data
{
hand_up();                  //Call Function hand up
}
else if (x1==0x32)          //Check data
{
hand_down();                //Call Function hand down
}
else if(x1==0x30||x3==0x30||x2==0x30||x5==0x30) //Check data
{
Stop();                     //Call Function Stop
Stophand();                 //Call Function Stophand
Stopcamera();               //Call Function Stopcamera
Stopcamera1();              //Call Function Stopcamera1
```

```
                    }
                    dut=x4;                 // Move data
                    speed=dut-0x30;         //Change char to integer
                    speed=speed*10;         //ขยายค่า speed 10 เท่า
                    }
                }
            }
            return 0;
}
```

บรรณานุกรม

- DSpic Microcontroller ด้วยโปรแกรมภาษา C กับ MPLAB C30 , นคร ภักดีชาติ , ชัยวัฒน์ ลิ่มพรจิตรวิไล บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

- คู่มือ Visual Basic 6 ฉัททวุฒิ พืชผล , พิชิต สันติกุลานนท์ พร้อมเลิศ หล่อวิจิตร , พิมพ์ครั้งที่ 11 กรกฎาคม บริษัท โปรวิชั่น จำกัด