อภินันทนาการ

การควบคุมหุ่นยนต์อัตโนมัติด้วยระบบควบคุมแบบฟัซซี่

Controlling Autonomous Robot with Fuzzy Logic Control System

นาย กฤษฎา ศิริวัฒนธานี รหัส 51364620

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2558

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน การควบคุมหุ่นยนต์อัตโนมัติด้วยระบบควบคุมแบบฟัซซี่ลอจิก

ผู้ดำเนินโครงงาน นาย กฤษฎา ศิริวัฒนธานี รหัส 51364620

ที่ปรึกษาโครงงาน ผู้ช่วยศาสตราจารย์ ดร.พรพิศุทธิ์ วรจิรันตน์

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2558

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

.....ที่ปรึกษาโครงงาน
(ผู้ช่วยศาสตราจารย์ ดร.พรพิศุทธิ์ วรจิรันตน์)

.....กรรมการ
(อาจารย์ รัฐภูมิ วรานุสาสน์)

.....กรรมการ
(อาจารย์ ภาณุพงศ์ สอนคม)

ชื่อหัวข้อโครงงาน	การควบคุมหุ่นยนต์อัตโนมัติด้วยระบบควบคุมแบบฟัซซี่
ผู้ดำเนินโครงงาน	นาย กฤษฎา ศิริวัฒนธานี รหัส 51364620
ที่ปรึกษาโครงงาน	ผู้ช่วยศาสตราจารย์ ดร.พรพิศุทธิ์ วรจิรันตน์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2558

.....

บทคัดย่อ

โครงงานนี้เป็นการศึกษาการทำงานของระบบ Feed-Back Control และระบบควบคุม Fuzzy Logic Control System โดยมีการเขียนโปรแกรมระบบควบคุมแบบป้อนกลับขึ้นจริงและใช้ระบบควบคุมแบบฟัซซี่ ในการควบคุมความเร็วของหุ่นยนต์ ซึ่งจะทำให้หุ่นยนต์มีผลการตอบสนองต่อสภาพแวดล้อมที่เปลี่ยนไปได้ดีขึ้น และสามารถผ่านอุปสรรคที่มีผลต่อความเร็วของหุ่นยนต์ได้เป็นอย่างดี โดยในโครงงานนี้ ได้ใช้ระบบควบคุมแบบฟัซซี่ในการทำระบบควบคุมแบบป้อนกลับให้กับหุ่นยนต์อัตโนมัติ ที่มีล้อโอมนิทั้งหมด 4 ล้อ ซึ่งสามารถสั่งให้หุ่นยนต์เคลื่อนที่ได้ 8 ทิศทาง และระบบควบคุมแบบฟัซซี่จะทำการควบคุมความเร็วของหุ่นยนต์เมื่อหุ่นยนต์เจออุปสรรค์ที่ส่งผลให้ความเร็วของมอเตอร์มีการเปลี่ยนแปลง เช่น ลงทางลาดชัน หรือ ขึ้นเนินสูง เป็นต้น ส่วนในการควบคุมหุ่นยนต์ จะใช้อุปกรณ์ไมโครคอนโทรลเลอร์ dsPIC30F4011 ในการควบคุมหุ่นยนต์และใช้ภาษาซีในการเขียนโปรแกรมให้กับไมโครคอนโทรลเลอร์ dsPIC30F4011

Project title	Controlling Autonomous Robot with Fuzzy Logic Control System
Name	Mr. Kitsada Siriwattanatanee ID. 51364620
Project advisor	Asst. Prof. Dr. Ponpisu Worrajiran
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic year	2015

Abstract

The Purpose of to study the function of the Feed-Back Control System and Fuzzy Logic Control System. The robot has to respond to a changing environment better and pass the obstacles that affect the speed of the robot as well. In this project, Using Fuzzy Logic Control System for Controlling to a robot that can move eight directions. As a result, the robot can maintain the required speed even there is an obstacle. The robot uses the dsPIC30F4011 microcontroller to control robots and it is programed using C language.

กิตติกรรมประกาศ

โครงงานวิศวกรรม เรื่องการควบคุมหุ่นยนต์อัตโนมัติด้วยระบบควบคุมแบบฟัซซี่ สำเร็จเป็นรูปเล่มได้เนื่องจากได้รับความกรุณาจาก อาจารย์ พรพิศุทธิ์ วรจิรันตน์ ผู้เป็นอาจารย์ที่ปรึกษา ได้วางรากฐานและมอบวิชาความรู้ทางด้านระบบควบคุมแบบฟัซซี่นี้ อีกทั้งยังให้ความช่วยเหลือและคำชี้แนะต่างๆเป็นอย่างดีให้แก่ผู้ร่วมงาน และอาจารย์ เศรษฐา ตั้งค้าวานิช อาจารย์ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ มหาวิทยาลัยนเรศวร ผู้ให้ความรู้เรื่องการควบคุมการทำงานของไมโครคอนโทรเลอร์ตระกูล dsPIC30F4011 อีกทั้งอาจารย์ ไพรัศ นูขุนทศ ซึ่งเป็นอาจารย์ที่วิทยาลัยเทคนิคสกลนคร ที่ได้มอบความรู้ต่างๆมากมายไม่ว่าจะเป็นความรู้เกี่ยวกับการเขียนโปรแกรมในตัวหุ่นยนต์ ความรู้ด้านเทคนิคที่ใช้ในการประกอบโครงหุ่นยนต์ขึ้น ความรู้ด้าน Software และ Hardware ที่นำมาใช้กับ Encoder มอเตอร์ของหุ่นยนต์ และคำชี้แนะต่างๆที่ทำให้โครงงานนี้สำเร็จได้

ขอแสดงความขอบคุณผู้ที่ช่วยให้โครงงานนี้สำเร็จได้ด้วยดีทุกคน

นาย กฤษฎา ศิริวัฒนธานี 51364620

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

ในปัจจุบันนี้หุ่นยนต์ (Robot) ได้รับความนิยมเป็นอย่างมาก มีการจัดแข่งขันการประกวดหุ่นยนต์ในรูปแบบต่างๆทั้งในระดับประเทศไปจนถึงระดับโลก ระบบในการควบคุมหุ่นยนต์นั้นมีหลากหลายวิธีที่จะทำให้หุ่นยนต์สามารถเคลื่อนที่ได้ตามที่ต้องการ แต่ระบบควบคุมที่สามารถสร้างวิธีการการเคลื่อนของหุ่นยนต์ได้ด้วยตัวเองหรือสามารถควบคุมความเร็วของหุ่นยนต์ได้ด้วยตัวเองโดยมีกฎพื้นฐานการควบคุมของมนุษย์นั้นมีน้อยมาก

เนื่องจากโครงงานหุ่นยนต์ควบคุมความเร็วส่วนใหญ่เป็นหุ่นยนต์ที่มีระบบควบคุมแบบเปิด (Open-Loop control system) ซึ่งทำให้หุ่นยนต์ไม่สามารถควบคุมความเร็วหรือทิศทางได้ด้วยตัวเอง จึงทำให้ผู้จัดทำสนใจที่จะศึกษาโครงงานเกี่ยวกับการควบคุมแบบป้อนกลับ (Feedback control system) และระบบควบคุมแบบฟัซซี่ (Fuzzy Logic Control System) ซึ่งการควบคุมแบบป้อนกลับนี้จะทำให้เกิดประโยชน์ต่อหุ่นยนต์อย่างมาก เช่น ทำให้หุ่นยนต์สามารถควบคุมความเร็วในการเคลื่อนที่ด้วยตัวเอง สามารถควบคุมทิศทางได้ด้วยตัวเอง ทำให้ระบบสามารถทำงานในสภาวะแวดล้อมที่เปลี่ยนแปลงได้อย่างถูกต้อง

การพัฒนาโครงงานนี้ได้ใช้อุปกรณ์ไมโครคอนโทรลเลอร์ dsPIC30F4011 ในการควบคุมหุ่นยนต์ และสำหรับการพัฒนาโปรแกรมควบคุมนี้ได้ทำการทดสอบบนระบบปฏิบัติการไมโครซอฟท์วินโดว์ 7 โดยใช้โปรแกรม MPLAB IDE V.8.76 ในการทดสอบระบบและใช้ CCSC Compiler ในการคอมไพล์โปรแกรม โดยมีการพัฒนาโปรแกรมควบคุมด้วยภาษาซี เพื่อให้หุ่นยนต์สามารถเคลื่อนที่ได้ด้วยระบบควบคุมแบบฟัซซี่

สำหรับผลลัพธ์ที่จะได้รับจากโครงงานนี้ คือ ทำให้หุ่นยนต์ที่สร้างขึ้นสามารถเคลื่อนที่ได้ทุกทิศทางและสามารถปรับค่าความเร็วของหุ่นยนต์ได้โดยใช้ระบบควบคุมแบบฟัซซี่

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 สร้างหุ่นยนต์ 1 ตัวให้เคลื่อนที่แบบ 8 ทิศทาง

1.2.2 สามารถปรับความเร็วของมอเตอร์โดยใช้เทคนิค PWM (Pulse Width Modulation) ได้

1.2.3 สร้างระบบควบคุมความเร็วให้หุ่นยนต์เคลื่อนที่ได้อย่างถูกต้องแม้มีการเปลี่ยนไปของสภาวะแวดล้อมโดยใช้ระบบควบคุมแบบฟัซซี่

1.3 ขอบเขตโครงงาน

1.3.1 สร้างโปรแกรมระบบการควบคุมแบบป้อนกลับของหุ่นยนต์เพื่อทำการควบคุมความเร็วของหุ่นยนต์ในขณะที่หุ่นยนต์วิ่งเจอสิ่งกีดขวาง

1.3.2 ค่าความผิดพลาด (Error) ที่เกิดจากค่าความเร็วรอบของมอเตอร์เทียบกับค่า Set Point ของระบบต้องอยู่ในช่วง บวกลบ 5 รอบ / Sec และความหน่วงของเวลา (Delay Time) ต้องไม่เกิน 1 วินาที และความเร็วในการกลับเข้าสู่ระบบ (Setting Time) ต้องอยู่ในช่วง 1 - 2 วินาที

1.3.3 หุ่นยนต์สามารถเคลื่อนที่ด้วยความเร็วที่เท่ากันเมื่อเปลี่ยนความชันของพื้นเอียงเป็น 10 20 และ 30 องศาได้

1.3.4 หุ่นยนต์สามารถปรับความเร็วได้ 3 ระดับ

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาการทำงานวงจรไฟฟ้าและการเขียนโปรแกรมบนไมโครคอนโทรลเลอร์ตระกูล dsPIC30F4011
2. ออกแบบโครงสร้างและวงจรไฟฟ้าสำหรับหุ่นยนต์
3. ซื้ออุปกรณ์สำหรับทำหุ่นยนต์และสร้างหุ่นยนต์ตามที่ออกแบบ
4. ศึกษาและจัดทำโปรแกรมด้วยภาษา C โดยใช้ MPLAB IDE V.8.76 ที่สามารถสั่งการให้ไมโครคอนโทรลเลอร์ทำงานตามชุดคำสั่งต่อไปนี้
    4.1 สามารถควบคุมหุ่นยนต์ได้ 8 ทิศทาง
    4.2 สามารถควบคุมความเร็วของมอเตอร์โดยใช้พีดับเบิ้ลยูเอ็ม
    4.3 หุ่นยนต์มีระบบการควบคุมแบบฟัซซี่
5. ทดสอบโปรแกรมที่เขียนลงในไมโครคอนโทรลเลอร์และการเคลื่อนที่ของหุ่นยนต์

1.5 แผนการดำเนินงาน

ตารางที่ 1.1 แสดงแผนการดำเนินงานของโครงงานนี้

กิจกรรม	ปี 2555							ปี 2556				
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.
ศึกษาการทำงานวงจรไฟฟ้าและการเขียนโปรแกรมบนไมโครคอนโทรลเลอร์ตระกูล dsPIC ของหุ่นยนต์	←			→								
ออกแบบโครงสร้างและวงจร ไฟฟ้าสำหรับหุ่นยนต์		←		→								
ซื้ออุปกรณ์สำหรับทำหุ่นยนต์			←		→							
สร้างหุ่นยนต์ตามที่ออกแบบ			←			→						
สร้างลายวงจรทั้งหมด					←					→		
ตรวจสอบและทดสอบวงจรไฟฟ้าที่ผ่านการเชื่อมวงจรไฟฟ้า					←					→		
ศึกษาและจัดทำโปรแกรมด้วยภาษา C โดยใช้ MPLAB IDE V. 8.76 ที่สามารถสั่งการให้ไมโครคอนโทรลเลอร์ทำงาน						←						→
ออกแบบโครงสร้างและวงจรไฟฟ้าสำหรับหุ่นยนต์								←				→

1.6 ประโยชน์ที่คาดว่าจะได้รับจากโครงงาน

1. สามารถสร้างหุ่นยนต์ให้เคลื่อนที่ได้ 8 ทิศทาง
2. หุ่นยนต์สามารถปรับความเร็วของมอเตอร์โดยวิธีพีดับเบิ้ลยูได้
3. หุ่นยนต์มีระบบการควบคุมแบบฟัซซี่สามารถควบคุมความเร็วของหุ่นยนต์ได้

1.7 งบประมาณ

1. ค่าทำเอกสาร		1,000	บาท
	รวม	1,000	บาท

(หนึ่งพันบาทถ้วน)

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

บทนี้จะอธิบายถึงหลักการและความรู้เบื้องต้นที่กลุ่มผู้จัดทำโครงงานได้ทำการศึกษาค้นคว้าหาข้อมูลเพื่อนำมาประยุกต์ใช้ในโครงงานนี้ โดยจะกล่าวถึงกฎโครงสร้าง วงจรไฟฟ้าและโปรแกรมที่ใช้ในการสร้างหุ่นยนต์และระบบการควบคุมแบบฟัซซี่ (Fuzzy Logic Control System)

2.1 ระบบการควบคุม (Control System)

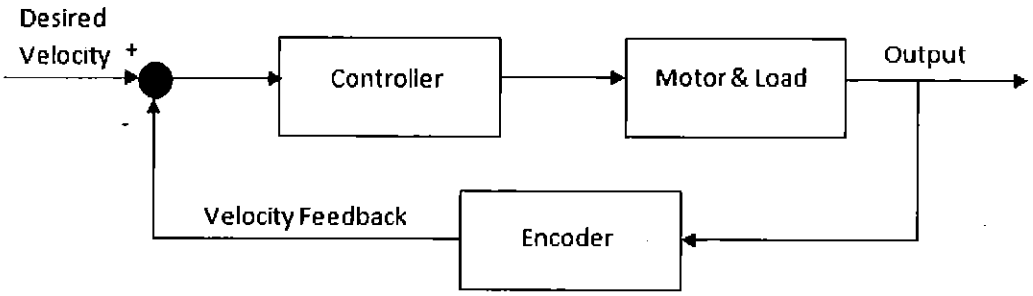
2.1.1 ระบบการควบคุมแบบวงเปิด (Open Loop Control System)

ระบบการควบคุมแบบวงเปิดเป็นระบบพื้นฐานที่สุดโดยการควบคุมระบบส่วนใหญ่ต้องอาศัยการคาดคะเนและการตัดสินใจของมนุษย์ ดังนั้นการควบคุมแบบนี้จึงไม่มีความน่าเชื่อถือและไม่แม่นยำ เนื่องจากผู้ควบคุมไม่สามารถรู้ค่าที่แน่นอนของระบบและปัจจัยภายนอกที่มีผลระบบได้

รูปที่ 2.1 แสดงระบบการควบคุมแบบวงเปิด

2.1.2 ระบบการควบคุมแบบป้อนกลับ (Feedback Control System)

เนื่องจากระบบการควบคุมแบบเปิดมีปัญหาเรื่องเสถียรภาพของเอาท์พุตและไม่สามารถควบคุมปัจจัยภายนอกได้อย่างสมบูรณ์ จึงมีการป้อนกลับของสัญญาณขาออกเพื่อสามารถตรวจจับความคลาดเคลื่อนระหว่างสัญญาณขาออก (Output) กับสัญญาณขาเข้า (Input) ได้

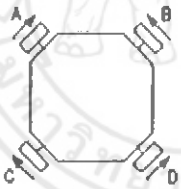
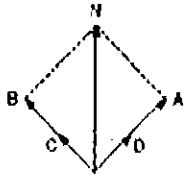
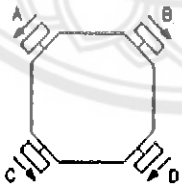
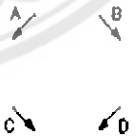
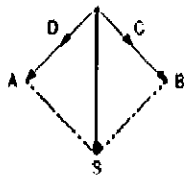


รูปที่ 2.2 แสดงระบบการควบคุมแบบป้อนกลับ

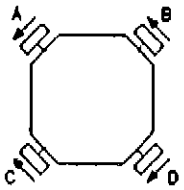
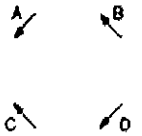
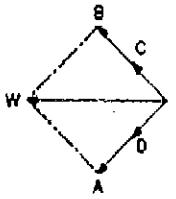
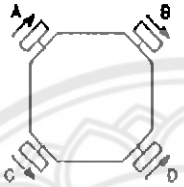
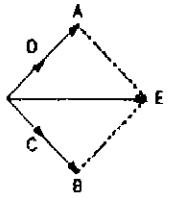
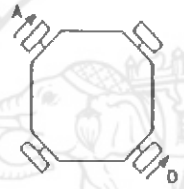
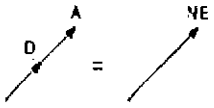
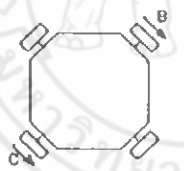
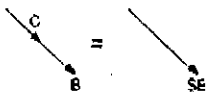
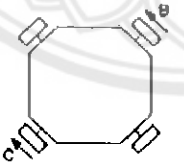
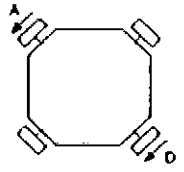
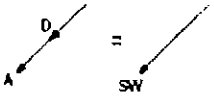
2.2 หลักการควบคุมทิศทางการเคลื่อนที่ของหุ่นยนต์ 4 ล้อ

ในโครงงานนี้สร้างหุ่นยนต์ 4 ล้อ โดยใช้ล้อเคลื่อนที่ได้ทุกทิศทาง (Omni Directional Wheel) ในการเคลื่อนที่ของหุ่นยนต์ ซึ่งสามารถเคลื่อนที่ได้ทุกทิศทางตามหลักการเวกเตอร์ ดังตารางที่ 2.1

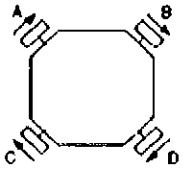
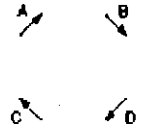
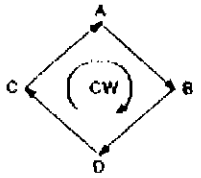
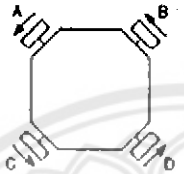
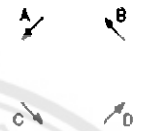
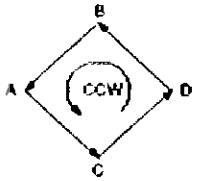
ตารางที่ 2.1 แสดงความสัมพันธ์ระหว่างทิศทางการหมุนของล้อกับทิศทางการเคลื่อนที่ของหุ่นยนต์

ทิศทางการเคลื่อนที่	ทิศทางการหมุนของล้อ	เวกเตอร์ที่เกิดขึ้น	การเคลื่อนที่ของหุ่นยนต์
เดินหน้า (N)			
ถอยหลัง (S)			

ตารางที่ 2.1 (ต่อ)

ทิศทางการเคลื่อนที่	ทิศทางการหมุนของล้อ	เวกเตอร์ที่เกิดขึ้น	การเคลื่อนที่ของหุ่นยนต์
ด้านซ้าย (W)			
ด้านขวา (S)			
ตะวันออกเฉียงเหนือ (NE)			
ตะวันออกเฉียงใต้ (SE)			
ตะวันตกเฉียงเหนือ (NW)			
ตะวันตกเฉียงใต้ (SW)			

ตารางที่ 2.1 (ต่อ)

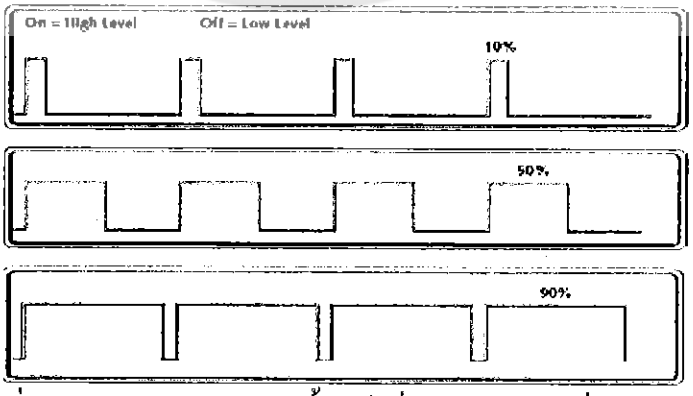
ทิศทางการเคลื่อนที่	ทิศทางการหมุนของล้อ	เวกเตอร์ที่เกิดขึ้น	การเคลื่อนที่ของหุ่นยนต์
หมุนตัวตามเข็มนาฬิกา (CW)			
หมุนตัวทวนเข็มนาฬิกา (CCW)			

ที่มา: http://www.nawattakam.com/talk/index.php?topic=368.0

2.3 การควบคุมแบบ PWM (Pulse Width Modulation)

เป็นเทคนิคการควบคุมกำลังไฟฟ้าเข้าสู่อุปกรณ์ไฟฟ้าโดยใช้สัญญาณเอาท์พุตแบบดิจิตอลจากไมโครคอนโทรลเลอร์เป็นตัวควบคุมสัญญาณที่สามารถปรับความกว้างของคลื่นความถี่ได้ ที่เรียกว่าดิวตี้ไซเคิล (Duty cycle) ซึ่งดิวตี้ไซเคิลจะเป็นตัวกำหนดปริมาณของพลังงานไฟฟ้าที่จ่ายให้กับอุปกรณ์ เช่น มอเตอร์ใช้ควบคุมความเร็วของมอเตอร์เป็นต้น [3]

หลักการทำงานของ พีดับเบิ้ลยูเอ็มคือ กำหนดความถี่คงที่แต่ความกว้างของพัลส์ไม่คงที่ ถ้าความกว้างของพัลส์มากความเร็วก็จะมาก แต่ถ้าความกว้างของพัลส์น้อยความเร็วก็จะน้อยตามเช่นกันดังรูปที่ 2.3



รูปที่ 2.3 แสดงสัญญาณพีดับเบิ้ลยูเอ็มที่มีค่ากำลังไฟฟ้าที่แตกต่างกัน

ที่มา: http://pic16f627a.blogspot.com/2012/02/PWM-by-pic.html [3]

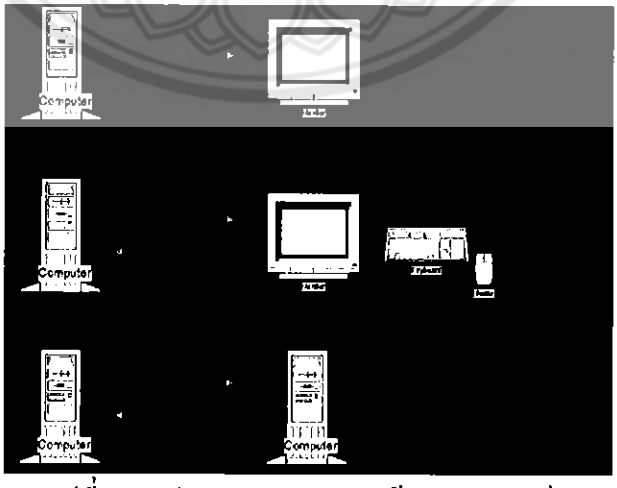
สาเหตุที่ใช้เทคนิคพีดับเบิ้ลยูเอ็ม ในการควบคุมความเร็วของมอเตอร์มีดังนี้

- พีดับเบิ้ลยูเอ็มง่ายต่อการอินเตอร์เฟสกับไมโครคอนโทรลเลอร์และใช้เพียงสัญญาณเอาท์พุตแค่สัญญาณเดียวในการควบคุมความเร็ว
- พีดับเบิ้ลยูเอ็มเป็นเทคนิคที่มีประสิทธิภาพเนื่องจากแหล่งจ่ายไฟสามารถจ่ายกำลังได้เต็มที่ทั้งสัญญาณเปิดและปิด
- พีดับเบิ้ลยูเอ็มสามารถจ่ายค่ากำลังสูงสุดของมอเตอร์ได้อย่างมีประสิทธิภาพ

2.4 UART

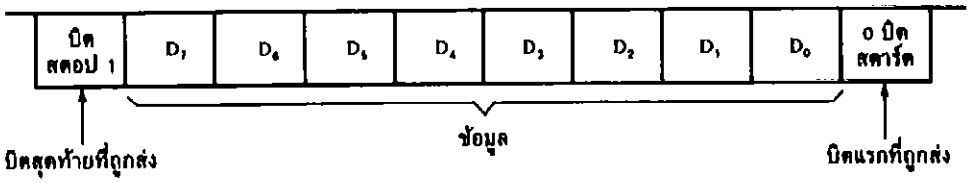
UART (Universal Asynchronous Receiver Transmitter) การสื่อสารอนุกรมบนคอมพิวเตอร์นั้น ยูอาร์ทถือเป็นหัวใจสำคัญ เนื่องจากยูอาร์ททำหน้าที่แปลงข้อมูลที่ส่งในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรม และทำหน้าที่แปลงสัญญาณอนุกรมแปลงเป็นสัญญาณแบบขนานก่อนเข้าคอมพิวเตอร์ด้วย แต่ในการส่งข้อมูลนั้นจะต้องแจ้งข้อมูลให้คอมพิวเตอร์ทราบด้วย เช่น อัตราในการรับส่งข้อมูล (Baud Rate) จำนวนบิตของข้อมูล และความผิดพลาดที่เกิดขึ้นระหว่างการส่งข้อมูล เช่น Parity Bit เป็นต้น [10]

ยูอาร์ต สามารถรับส่งข้อมูลได้ทั้งแบบ Half-Duplex คือการสื่อสารแบบทิศทางเดียว และFull-Duplex คือการสื่อสารที่สามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน ตามปกติยูอาร์ทส่งข้อมูลทีละ 8 บิตซึ่งได้ถูกกำหนดมาตรฐานการส่งข้อมูลในรูปแบบอนุกรม RS-232



รูปที่ 2.4a ประเภทการรับ / ส่งข้อมูลของยูอาร์ท

ที่มา: http://yyingyingy.blogspot.com/



รูปที่ 2.4b แสดงข้อมูลที่รับ / ส่งของยูอาร์ท

ที่มา: https://www.cpe.ku.ac.th/~yuen/204323/io/complete.htm

2.5 ASCII

ASCII (American Standard Code for Information Interchange) เป็นรหัสเลขฐานสอง 8 บิตใช้แทนสัญลักษณ์ ตัวอักษรต่างๆ 1 ตัว ดังรูปที่ 2.5 โดยที่ภาษาแอสกี้มีประโยชน์ในการสื่อสารข้อมูลระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

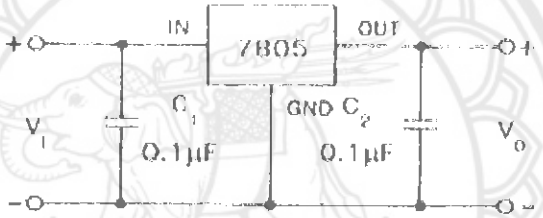
รูปที่ 2.5 แสดงผังอักขระของภาษาแอสกี้

ที่มา: http://www.asciitable.com/index/asciifull.gif

2.6 หลักการทำงานของวงจร

2.6.1 ไอซีเร็กกูเลเตอร์สามขาชนิดจ่ายแรงดันคงที่

ภายในประกอบด้วยวงจรเร็กกูเลเตอร์แบบอนุกรมมีขาต่อใช้งาน 3 ขาประกอบด้วยขาอินพุต เอาท์พุต และกราวด์ ซึ่งจะจ่ายแรงดันค่าใดค่าหนึ่งเฉพาะโดยจุดเด่นของไอซีเร็กกูเลเตอร์ค่าคงที่คือ สามารถต่อวงจรได้ง่ายไม่ต้องมีอุปกรณ์ภายนอกต่อเพิ่มเติม ในบางครั้งการเพิ่มอุปกรณ์เสริมจะมีผลดีขึ้น เช่น การใส่ตัวเก็บประจุอิเล็กทรอไลต์ ขนาด 0.1 ไมโครฟารัต 2 ตัว ดังรูปที่ 2.6 เพื่อป้องกันการเกิดออสซิลเลตที่ความถี่สูง ซึ่งทำให้วงจรขาดเสถียรภาพ เอาต์พุตที่ออกจากไอซีเร็กกูเลเตอร์จะได้แรงดันเอาท์พุตที่เรียบพออยู่แล้ว แรงดันไอซีเร็กกูเลเตอร์ชนิดจะให้แรงดันเอาท์พุตคงที่หลายแบบซึ่งหลายเบอร์เช่น 5 โวลต์ 12 โวลต์ 24 โวลต์ เป็นต้น

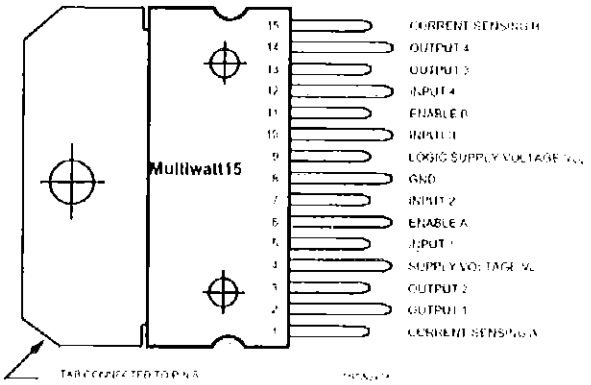


รูปที่ 2.6 แสดงการต่อไอซีเร็กกูเลเตอร์สามขาชนิดการจ่ายแรงดันคงที่กับตัวเก็บประจุ 0.1 ไมโครฟารัต

ที่มา: http://www.easycasyadmin.com/images/images/12Vto5V.png

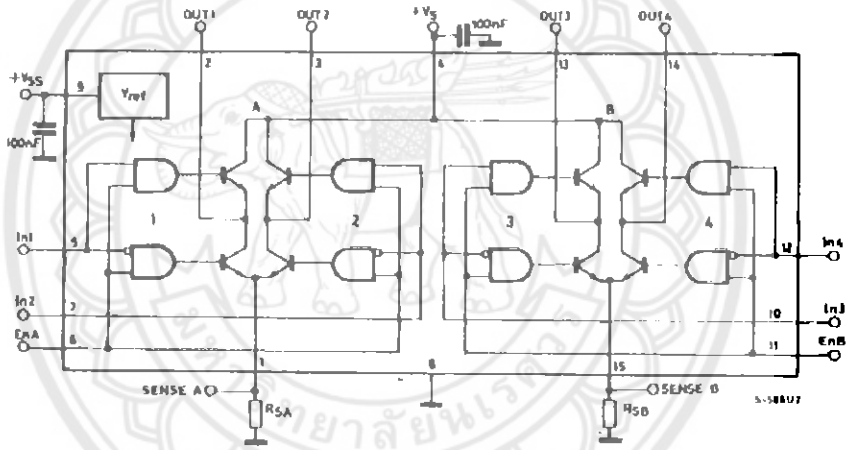
2.6.2 ไอซี Dual H-Bridge Switching L298

ไอซี L298 เป็นไอซีที่ทำหน้าที่รับค่ากำลังไฟฟ้าจากไมโครคอนโทรลเลอร์เพื่อนำมาสวิตช์แล้ว ได้เอาต์พุตตามที่เราต้องการ และมีขา Enable ไว้รองรับการทำพีดับเบิ้ลยูเอ็มได้ โครงสร้างของไอซี L298 เป็นไปตามรูป 2.7a [12]



รูปที่ 2.7a แสดงขาวงจรของไอซี L298

ที่มา: http://www.sjbaker.org/wiki/images/1/12/L298_pinout.png



รูปที่ 2.7b แสดงวงจรไฟฟ้าภายในไอซี L298

ที่มา : http://www.adrirobot.it/arduino/l298/l298_circuit.png

การทำงานของไอซี L298 เมื่อได้รับข้อมูลจากขาอินพุตมา 2 ขาสามารถควบคุมมอเตอร์ได้ 1 ตัว ถ้าอินพุตขา 5 เป็น 5 โวลต์และอินพุตขา 7 เป็น 0 โวลต์ เอาท์พุตขา 2 ออกเป็น 5 โวลต์และเอาท์พุตขา 3 จะออกเป็น 0 โวลต์ ซึ่งสามารถขับมอเตอร์ให้หมุนได้ แต่ถ้าสลับให้อินพุตขา 7 เป็น 5 โวลต์ และอินพุตขา 5 เป็น 0 โวลต์ จะทำให้มอเตอร์หมุนทิศตรงกันข้ามเดิม

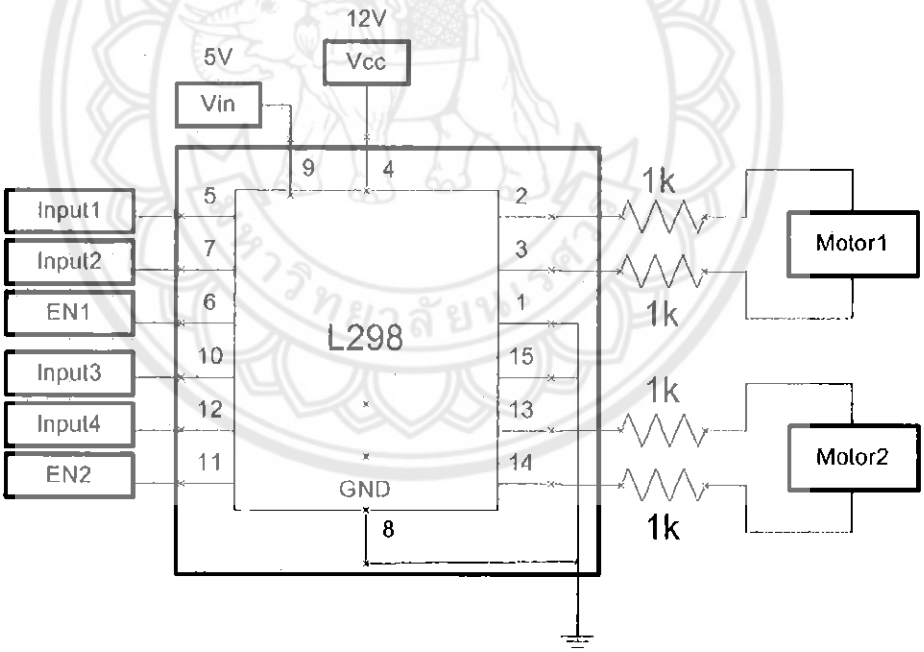
ไอซีนี้มีขา Enable โดยถ้าข้อมูลเข้าขา Enable เป็น 5 โวลต์จะสามารถขับมอเตอร์ได้ตามปกติ แต่ถ้าข้อมูลเข้าขา Enable เป็น 0 โวลต์จะทำให้มอเตอร์หยุดทันที สามารถสรุปการทำงานของไอซี L298 ได้ดังตารางที่ 2.2

ตารางที่ 2.2 แสดงการทำงานของไอซี L298

Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

ที่มา: http://www.st.com/content/st_com/en/products/motor-drivers/brushed-dc-motor-drivers/l298.html -> Datasheet

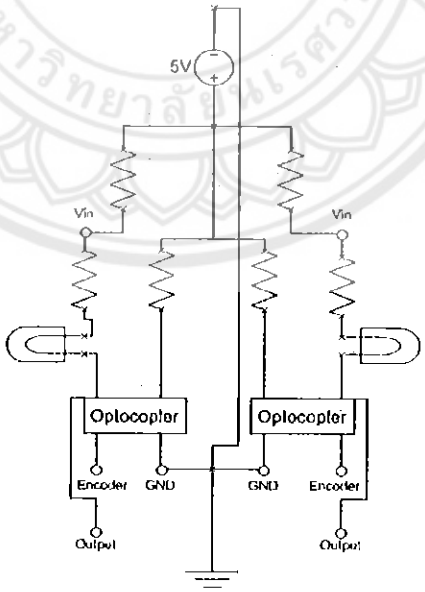


รูปที่ 2.8 วงจร L298 ที่ใช้งาน

2.6.3 Encoder

ตัวนับรอบความเร็ว (Encoder) ของมอเตอร์ถูกใช้งานอย่างแพร่หลายในเครื่องจักรและมอเตอร์ทั่วไปมีการใช้งานได้กว้างขวางหลายประเภท เช่น เป็น ฟีดแบค ของมอเตอร์ DC / AC มอเตอร์หรือเป็นอุปกรณ์ตรวจจับระยะทาง โดยตัวนับรอบจะแปลงสัญญาณความเร็วรอบหรือระยะทางออกมาเป็นคลื่นสัญญาณ (Digital Signal) ในการหมุน 1 รอบ โดยตัวมอเตอร์ที่ใช้มีความเร็วรอบ 100 RPM และมีเฟืองทดในตัวมอเตอร์อีก 80 ทด หมายความว่ามอเตอร์จะมีอัตราการหมุน 100 x 80 = 8000 รอบต่อ 1 นาที (ความเร็วรอบที่วัดได้ 133 รอบต่อวินาที)

รูปที่ 2.9a Encoder 12โวลต์ 150 RPM



รูปที่ 2.9b วงจรที่ใช้ต่อระหว่างตัวนับรอบ– ไมโครคอนโทรลเลอร์

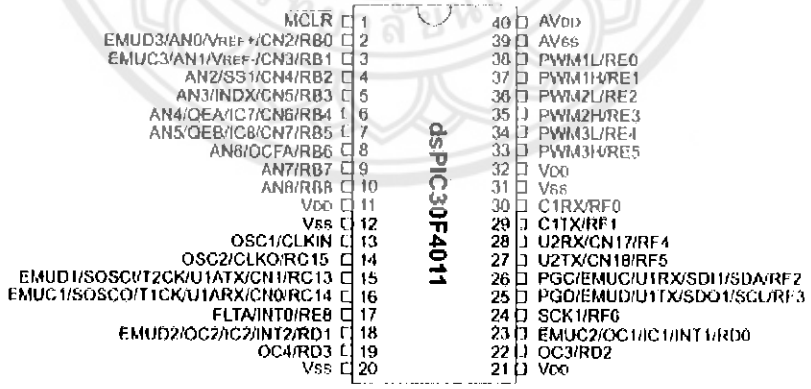
2.6.4 ET-Base dsPIC30F4011

เป็น Base Board ที่ใช้ไมโครคอนโทรลเลอร์ตระกูล PIC ใช้สถาปัตยกรรมแบบ RISC คือในหนึ่งคำสั่งทำงานใช้สัญญาณนาฬิกา 1 ลูกเป็นไมโครคอนโทรลเลอร์ เป็นที่ได้รับความนิยมจากโปรแกรมเมอร์เนื่องจากราคาถูก รองรับการใช้งานได้มากมายและสามารถเขียนโปรแกรมโดยลงโปรแกรมผ่าน Serial Port ได้ (อ้างอิง : [2] Microchip. (2008). คู่มือการใช้งาน ET-PGMPIC USB. กรุงเทพ ฯ : บริษัท อีทีที จำกัด)

คุณสมบัติสำคัญของ dsPIC30F4011 Base Board [2]

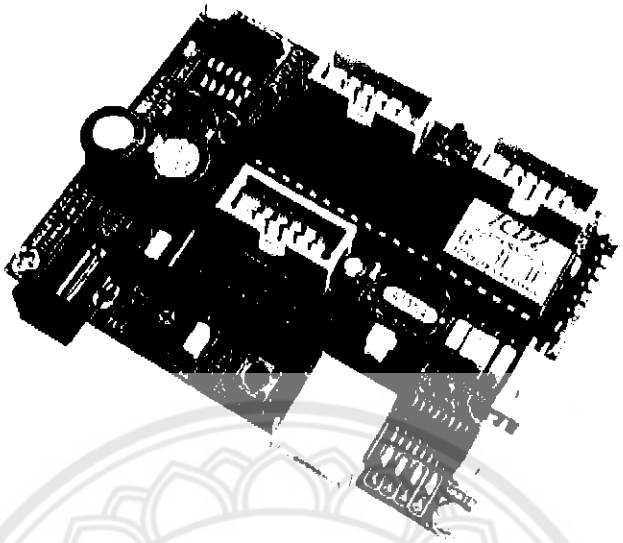
- มีหน่วยความจำแบบ FLASH ขนาด 48Kbytes RAM 2 Kbytes
- สถาปัตยกรรมแบบ RISC
- 16 Bit Timer/Counter 5 ชุด
- Input Capture จำนวน 4 ช่อง
- มีขั้วต่อ RS232 แบบ 4 พินสำหรับการต่อแบบ ETT 2 ช่อง
- POWER SUPPLY AC/DC INPUT 7-10 V ใช้เร็กกูเลเตอร์แบบ SWITCHING LM2575 ขนาด 5V / 1A ลดปัญหาความร้อนจากวงจรเร็กกูเลเตอร์
- ขั้วต่อใช้งาน I/O PORT จำนวน 10PIN 3 ชุด

40-Pin PDIP



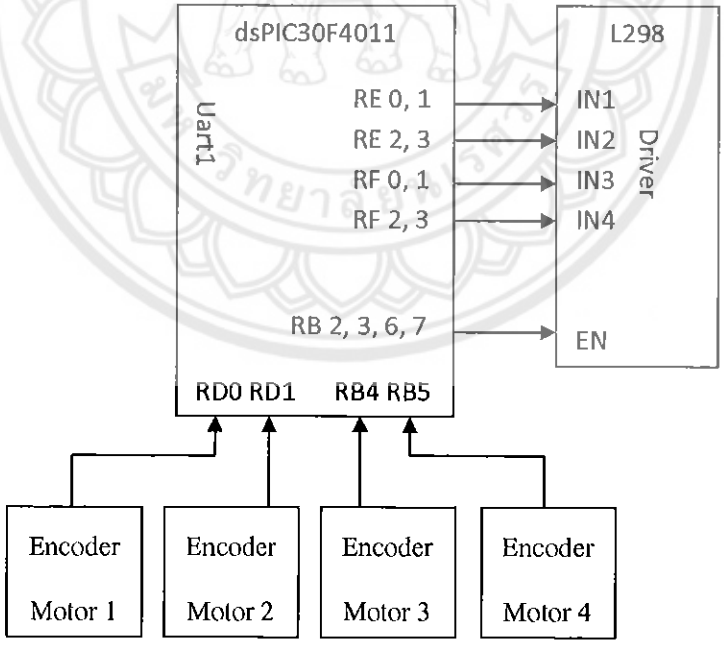
รูปที่ 2.10a ขาพอร์ต dsPIC30F4011 แบบ 40 ขา PDIP

ที่มา: http://q.lnwfile.com/_/q/_raw/mh/gd/1e.jpg



รูปที่ 2.10b ET-Base Board dsPIC30F4011

ที่มา: http://ett.co.th/product2009/ET-PIC/ET-BASE_dsPIC30F4011.html



รูปที่ 2.11 แบบการทำงานของบอร์ด dsPIC30F4011

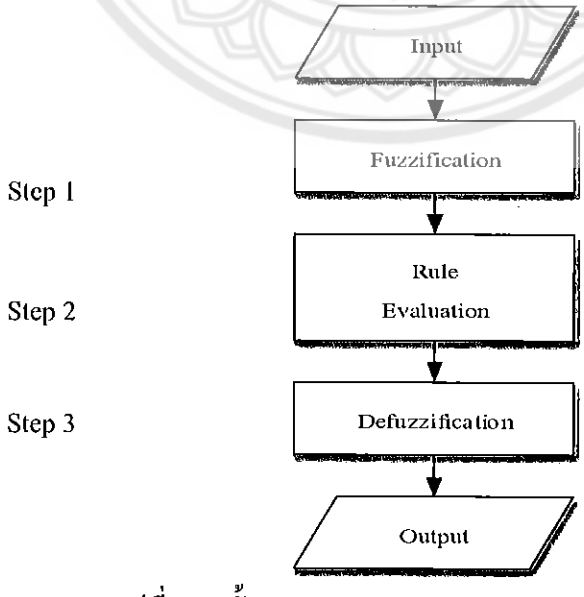
2.7 ทฤษฎีระบบการควบคุมแบบ Fuzzy

ระบบการควบคุมแบบฟัซซี่เป็นระบบที่เรียกว่า ระบบกฎเกณฑ์ (Rules-Base System) ซึ่งหมายความว่าจะมีกฎในระบบฟัซซี่เป็นตัวตัดสินใจในการทำงานของระบบควบคุมให้มีการปรับเปลี่ยนให้เป็นไปตามผลกระทบที่เกิดขึ้นในระบบ ซึ่งจุดมุ่งหมายของระบบการควบคุมแบบฟัซซี่คือการให้ระบบกฎเกณฑ์ของฟัซซี่ เข้าไปแทนการควบคุมแบบเก่า ซึ่งใช้มนุษย์เป็นผู้ควบคุม (Skilled Human Operator) [7]

2.7.1 ลักษณะเด่นของการควบคุมแบบฟัซซี่ [7]

- ไม่ใช้แบบจำลองทางคณิตศาสตร์ในการควบคุม (Model-Free Estimator)
- ไม่ใช้พื้นฐานทางคณิตศาสตร์ที่ยุ่งยากในการคำนวณ
- สามารถเข้าถึงปัญหาระบบได้เร็ว จึงสามารถทำการควบคุมได้อย่างทันเวลา
- ถ้าสร้างฟังก์ชันความเป็นสมาชิกได้อย่างถูกต้องจะใช้กฎในการควบคุมเพียงไม่กี่ข้อ
- สามารถทำการคำนวณค่าแต่ละค่าของแต่ละอินพุท-เอาท์พุทไว้ล่วงหน้าแล้วจัดเก็บไว้ในหน่วยความจำประเภทรอมได้ เพื่อให้การทำงานเร็วขึ้น เนื่องจากการคำนวณไว้ล่วงหน้าแล้วจึงสามารถควบคุมระบบได้โดยการใช้ข้อมูลจากรอม ซึ่งอยู่ในรูปของตารางค้นหา
- สามารถประยุกต์ใช้ได้อย่างดีในสาขาระบบผู้เชี่ยวชาญการทำการตัดสินใจและการประมวลผลสารสนเทศ
- เป็นการควบคุมโดยการวิเคราะห์เชิงคุณภาพ

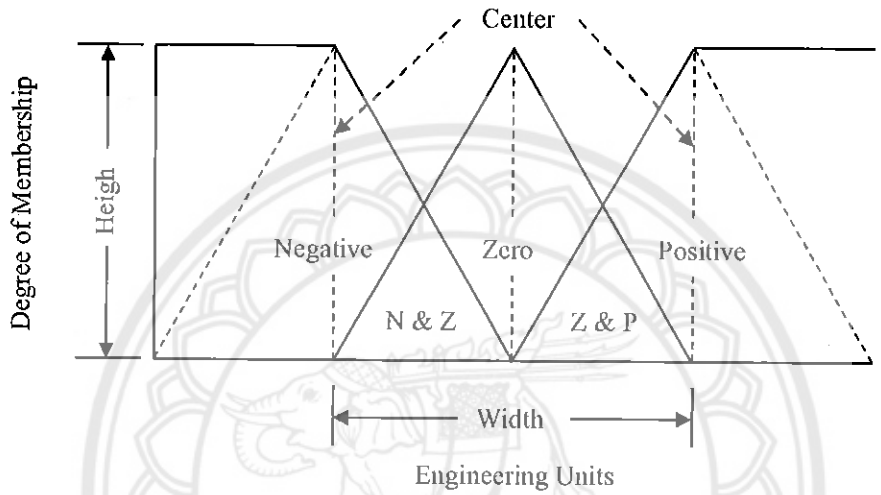
ขั้นตอนการออกแบบระบบการควบคุมแบบฟัซซี่



รูปที่ 2.12 ขั้นตอนการออกแบบระบบการควบคุมแบบฟัซซี่

2.7.2 Fuzzification

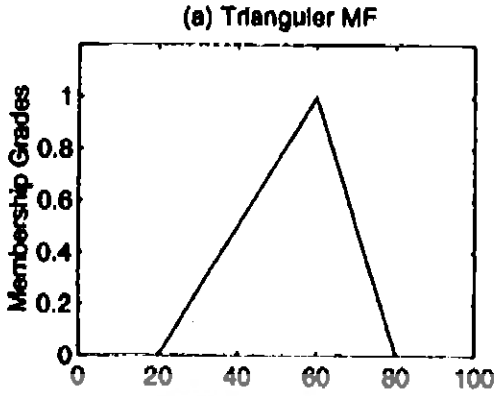
เป็นกระบวนการกำหนดค่าความเป็นสมาชิกของตัวแปรที่ใช้ โดยการแทนตัวแปรแบบฟัซซีด้วยฟังก์ชันความเป็นสมาชิกนั่นคือ กลุ่มของจำนวนสมาชิกที่อธิบายด้วยกราฟ อาจดูที่ความแตกต่างของระดับเอาต์พุต เช่น มุม ความเร็ว แล้วระบุจำนวนสมาชิกในกลุ่ม



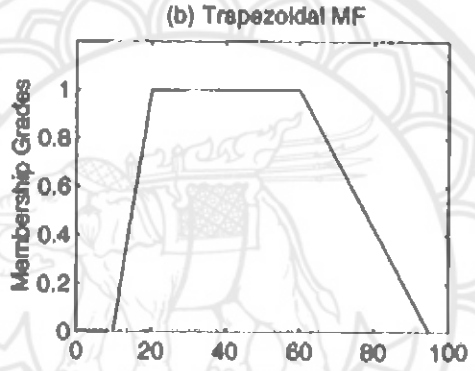
รูปที่ 2.13 ลักษณะทั่วไปของฟัซซี่เซต (Fuzzy Set)

ชนิดของเมมเบอร์ชิพฟังก์ชัน [14]

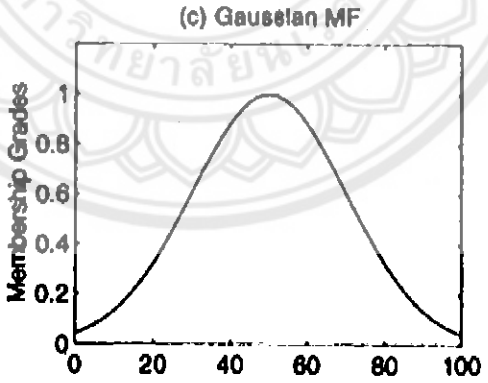
1. Triangular Membership Function
2. Trapezoidal Membership Function
3. Gaussian Membership Function
4. Generalized Membership Function
5. Sigmoid Membership Function
6. Left-Right Membership Function



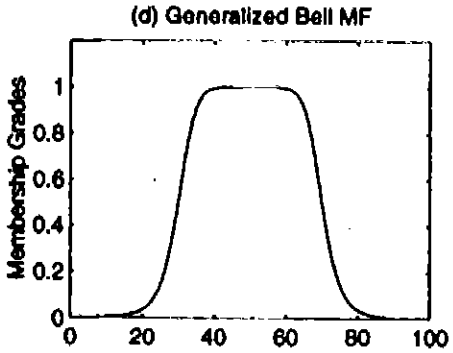
รูปที่ 2.14a Triangular Membership Function



รูปที่ 2.14b Trapezoidal Membership Function

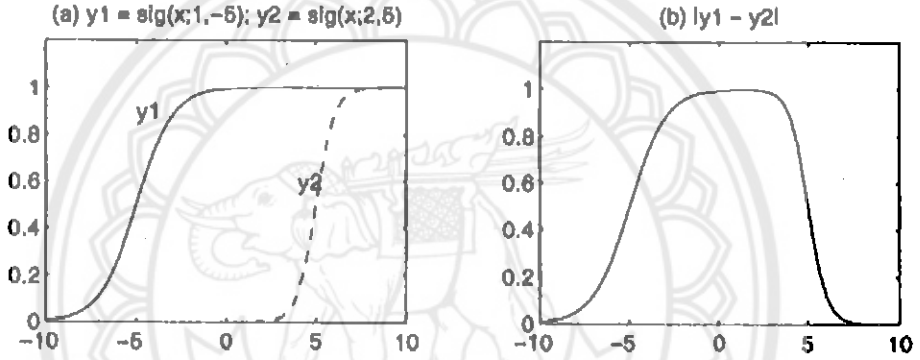


รูปที่ 2.14c Gaussian Membership Function



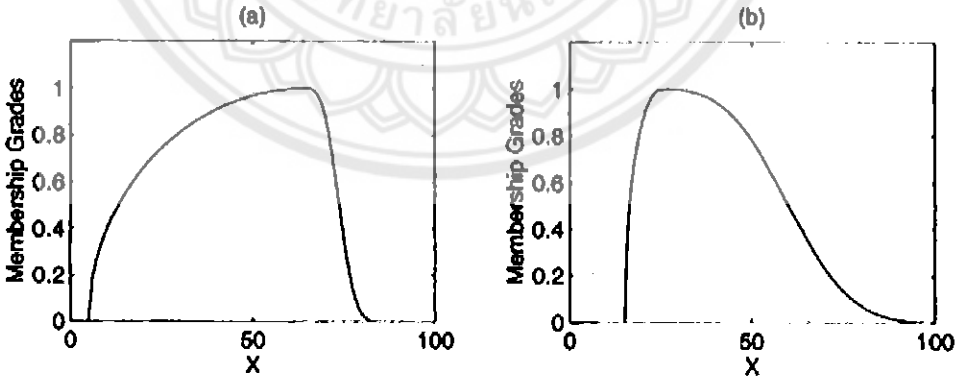
รูปที่ 2.14d Generalized Bell Membership Function

ที่มาของรูปที่ 2.14a – 2.14d: http://www.bindichen.co.uk/uploads/membership-function.jpg



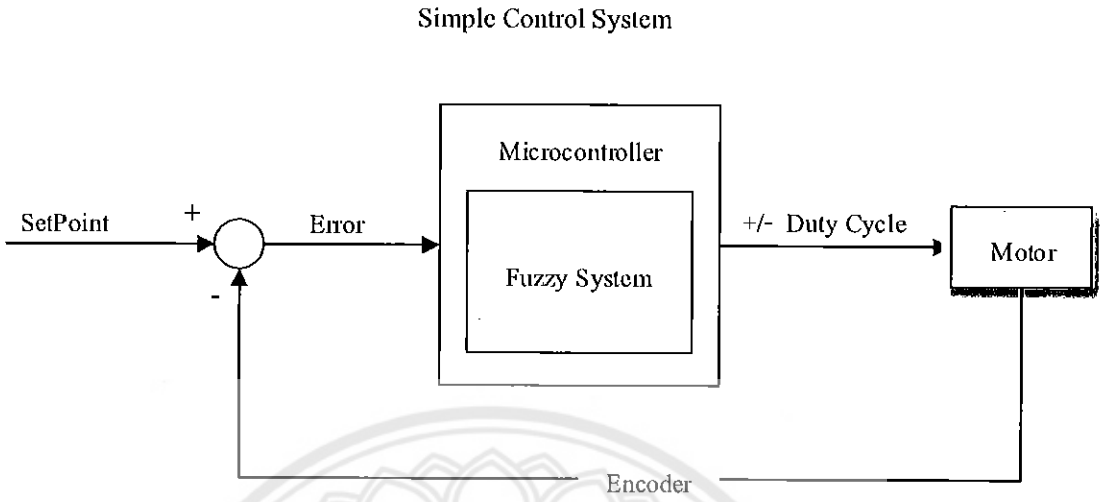
รูปที่ 2.14e Sigmoid Membership Function

ที่มา: http://www.bindichen.co.uk/uploads/sigmoid%20m%20function.jpg



รูปที่ 2.14f Left-Right Membership Function

ที่มา: http://www.bindichen.co.uk/uploads/left-right%20MF%202.jpg



รูปที่ 2.15 Block Diagram ของระบบตัวอย่าง

Set Point: ค่ารอบการหมุนของมอเตอร์ที่ตั้งไว้

Encoder: ค่าที่นับได้จากรอบการหมุนของมอเตอร์

Error: รอบการหมุนของมอเตอร์ที่ตั้งไว้ – ค่าที่นับได้จากรอบการหมุนของมอเตอร์

Change of Error: อัตราการเปลี่ยนแปลงของ Error

Duty Cycle: ค่ากำลังไฟฟ้าที่จ่ายให้กับมอเตอร์

ตัวอย่างที่ 1 สมมุติว่ากำหนดรถวิ่งที่บนพื้นด้วยค่ากำลังไฟฟ้า 50% ซึ่งหมายถึงว่าค่าเซตพอยน์ที่ตั้งไว้จะมีค่าความเร็วรอบที่ตัวนับรอบนับได้ 150 รอบต่อนาทีจากที่ความเร็วรอบสูงสุด 300 รอบต่อนาทีในรอบแรกของการอ่านค่า ค่าที่อ่านจากตัวนับรอบจะอ่านได้ 150 รอบต่อนาทีตามที่ตั้งไว้ แต่เมื่อรถเกิดการลงทางลาดชันทำให้รถวิ่งเร็วกว่าที่กำหนดทำให้ค่าที่ตัวนับรอบนับได้ 155 รอบต่อนาที เพราะฉะนั้น จะเกิดค่า Error = -5 และเกิดค่า Change of Error = 5

จากรูปที่ 2.15 เซตพอยน์คือรอบการหมุนของมอเตอร์ที่ระบบทำการเซตขึ้นมา จากนั้นระบบจะทำการเช็คค่าความเร็วรอบแล้วนำเปรียบเทียบกับเซตพอยน์เพื่อหาค่า Error ที่เกิดขึ้นแล้วนำไปผ่านระบบการควบคุมแบบฟัซซี่เพื่อหาค่าเอาต์พุต และนำไปสั่งไมโครคอนโทรลเลอร์ทำงานต่อไป ดังนั้นจากตัวอย่างที่ 1 สามารถแบ่งอินพุตเซตได้ตามตารางที่ 2.3a และ 2.3b

ตารางที่ 2.3a อินพุต Error ของระบบ

Input Error	
N	"Negative"
Z	"Zero"
P	"Positive"

ตารางที่ 2.3b อินพุต Change of Error ของระบบ

Input Change of Error	
N	"Negative"
Z	"Zero"
P	"Positive"

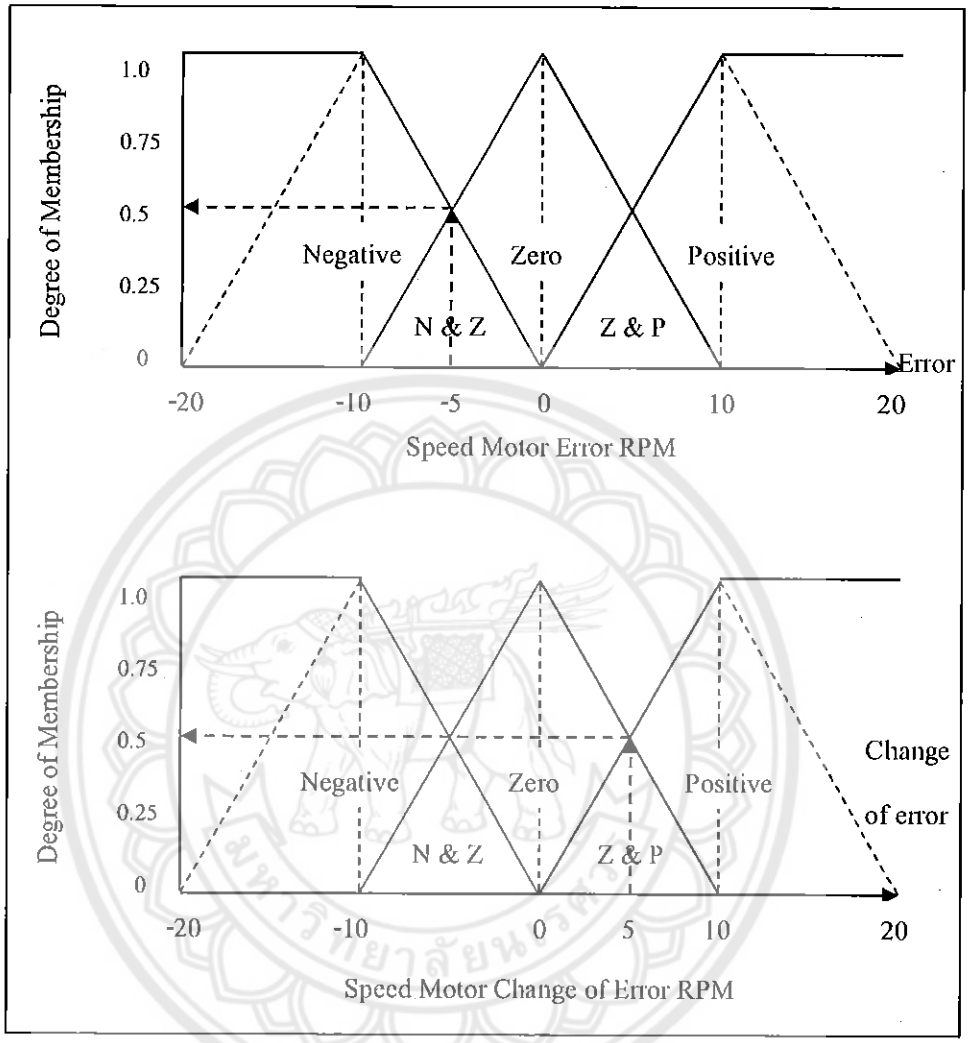
และจากตัวอย่างที่ 1 สามารถแบ่ง Output Set ได้ตามตารางที่ 2.4

ตารางที่ 2.4 เอาต์พุตของระบบ

เอาต์พุต	
N	"Negative" (Slow)
Z	"Zero"
P	"Positive" (Fast)

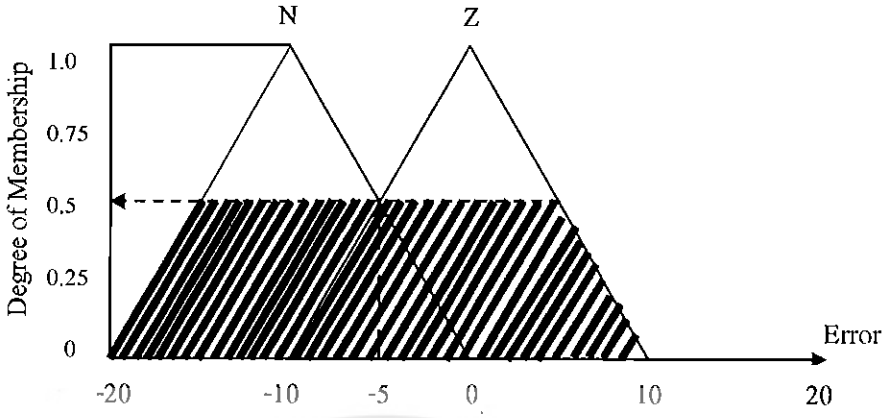
จากตัวอย่างที่ 1 ในระบบนี้กำหนดให้มีแค่ 3 กลุ่ม ถ้าจะให้งานมีความละเอียดสูงควรแบ่งให้มีกลุ่มของสมาชิกมากขึ้น

จากตัวอย่างที่ 1 มีค่า Error = -5 และ ค่า Change of Error = 5 สามารถเขียนอินพุตทั้งหมดในระบบโดยเลือกใช้ชนิดของค่าความเป็นสมาชิกคือ Triangular ได้ดังรูปที่ 2.16

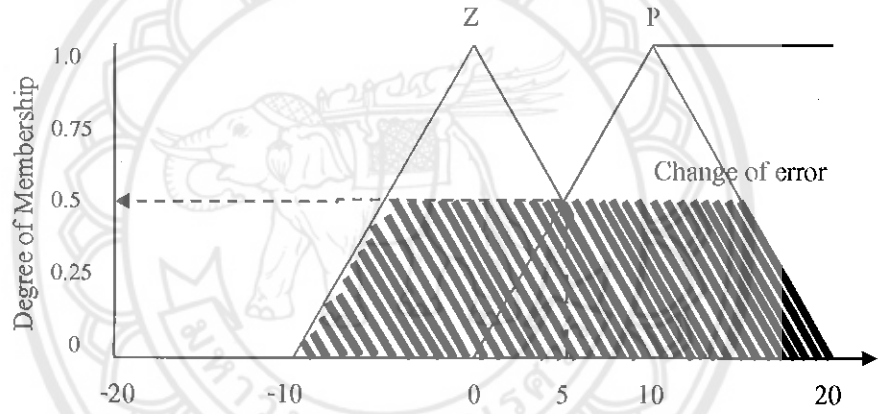


รูปที่ 2.16 ฟัซซี่เซตจากระบบตัวอย่างและอินพุตที่วัดได้

เพราะฉะนั้น เมื่อใส่ค่า Error = -5 และ Change of Error = 5 ค่าความเป็นสมาชิกที่สร้างขึ้นมาแล้วจะได้ค่าความเป็นสมาชิกของอินพุตทั้งหมดออกมาได้ดังรูปที่ 2.17a และ 2.17b



รูปที่ 2.17a ค่าความเป็นสมาชิกของ Error



รูปที่ 2.17b ค่าความเป็นสมาชิกของ Change of Error

2.7.3 Rule

คือลำดับอนุกรมที่ใช้ If-Then ในการอธิบาย เช่น ถ้าต้องการหาค่าทิป (เอาต์พุต) จากการรับประทานอาหารโดยค่าทิปที่ได้ดูจากคุณภาพการบริการ (อินพุต1) และรสชาติของอาหาร (อินพุต2) ถ้าคุณภาพการบริการดี และรสชาติของอาหารดี ค่าทิปที่ได้จะสูง แต่ถ้าคุณภาพการบริการต่ำ และรสชาติของอาหารต่ำ ค่าทิปที่ได้จะต่ำเช่นกัน หรือ ถ้าต้องการกำหนดราคาเสื้อผ้า (เอาต์พุต) โดยราคาเสื้อนั้นดูจากความต้องการของผู้ซื้อ (อินพุต1) และคุณภาพของเนื้อผ้า (อินพุต2) ถ้าความต้องการของผู้ซื้อมีมาก และคุณภาพของเนื้อผ้าดี ราคาเสื้อที่ได้จะอยู่ในระดับสูง แต่ถ้าความต้องการของผู้ซื้อมีน้อย และคุณภาพของเนื้อผ้าปานกลาง ราคาเสื้อที่ได้จะอยู่ในระดับปานกลางหรือต่ำ เป็นต้น [6]

จากตัวอย่างที่ 1 จะเห็นว่าสิ่งที่ต้องการคือแรงขับเคลื่อนของรถ (Output) ที่ต้องเพิ่มหรือลดในช่วงเวลาที่รถเจอสภาวะแวดล้อมที่เปลี่ยนไปทำให้ความเร็วของรถเกิดการเปลี่ยนแปลงซึ่งสิ่งที่ทำให้ความเร็วรถเปลี่ยนแปลงนั้นคือค่า Error (Input1) และ Change of Error (Input2) ซึ่งในตัวอย่างที่ 1 บอกว่าเมื่อรถวิ่งลงทางลาดชันทำให้รถวิ่งเร็วกว่าปกติจึงทำให้เกิด Error ขึ้นมีค่า = -5 ซึ่งมีค่าเป็น Negative และค่า Change of Error ที่เกิดขึ้นมีค่า = 5 ซึ่งมีค่าเป็น Positive สิ่งที่ระบบต้องทำในตอนนี้คือ ต้องลดความเร็วของรถเพื่อจะให้ความเร็วรถกลับมาเท่ากับความเร็วเดิม (Set Point) นั่นคือ ค่าเอาต์พุตต้องมีค่าเป็น Negative

เมื่อได้อินพุตและเอาต์พุต ครบแล้ว สามารถเขียนกฎที่จะใช้ควบคุมระบบทั้งหมดได้ดังตารางที่ 2.5

ตารางที่ 2.5 Rule Structure และ Rule Matrix ของตัวอย่างที่ 1

CE / E	N	Z	P
N	Z	P	P
Z	N	Z	P
P	N	N	Z

Rule Structure & Rule Matrix

1. If Error = N AND Change of Error = N Then Output = Z
2. If Error = Z AND Change of Error = N Then Output = N
3. If Error = P AND Change of Error = N Then Output = N
4. If Error = N AND Change of Error = Z Then Output = P
5. If Error = Z AND Change of Error = Z Then Output = Z
6. If Error = P AND Change of Error = Z Then Output = N
7. If Error = N AND Change of Error = P Then Output = P
8. If Error = Z AND Change of Error = P Then Output = P
9. If Error = P AND Change of Error = P Then Output = Z

2.7.4 Inferencing

ขั้นตอนนี้คือการหาค่าความเป็นสมาชิกของเอาต์พุตฟัซซี่ของแต่ละกฎและผลลัพธ์ทางลอจิก (Logical Products) จากแต่ละกฎจะต้องถูกนำมาคำนวณก่อนที่จะส่งต่อไปยังขั้นตอน ดีฟัซซิฟีเคชันเพื่อหาค่าความเป็นสมาชิกของเอาต์พุต [4]

วิธีการอิน-เฟอะเร็นซ์มีหลายวิธี เช่น

1. The MAX-MIN Method

สมการ $$r_p = \frac{\sum_{k=1}^{m} \min(x_{ij}, x_{jk})}{\sum_{k=1}^{m} \max(x_{ij}, x_{jk})}$$

2. The Averaging Method

สมการ $$r_p = \frac{\sum_{k=1}^{m} \min(x_{ij}, x_{jk})}{\frac{1}{2}\sum_{k=1}^{m} (x_{ij} + x_{jk})}$$

3. The Root-Sum-Square (Rss) Method

สมการ $$r_p = \sqrt{\sum_{k=1}^{m} [\min(x_{ij}, x_{jk})]^2}$$

r_p คือ ค่าความเป็นสมาชิกของเอาต์พุตฟัซซี่ของแต่ละกฎ

x คือ ค่าความเป็นสมาชิกของอินพุตฟัซซี่ของแต่ละกฎ

จากตัวอย่างที่ 1 แสดงการอิน-เฟอะเร็นซ์ด้วยวิธี Root-Sum-Square จากตารางที่ 2.5 กฎจะเขียนออกมาได้ดังตารางที่ 2.6

ตารางที่ 2.6 การอิน-เฟอะเร็นซ์ด้วยวิธี Root-Sum-Square

E / CE	N (0.5)	Z (0.5)
Z (0.5)	N1	Z1
P (0.5)	N2	N3

จากนั้นทำการคำนวณด้วยวิธี Root-Sum-Square จะได้ดังนี้

Output N $= \sqrt{[\min(N_E, Z_{CE})]^2 + [\min(N_E, P_{CE})]^2 + [\min(Z_E, P_{CE})]^2}$

$= \sqrt{[\min(0.5,0.5)]^2 + [\min(0.5,0.5)]^2 + [\min(0.5,0.5)]^2}$

$= \sqrt{(0.5)^2 + (0.5)^2 + (0.5)^2}$

$= \sqrt{0.25 + 0.25 + 0.25}$

$= \sqrt{0.75}$

$= 0.866$

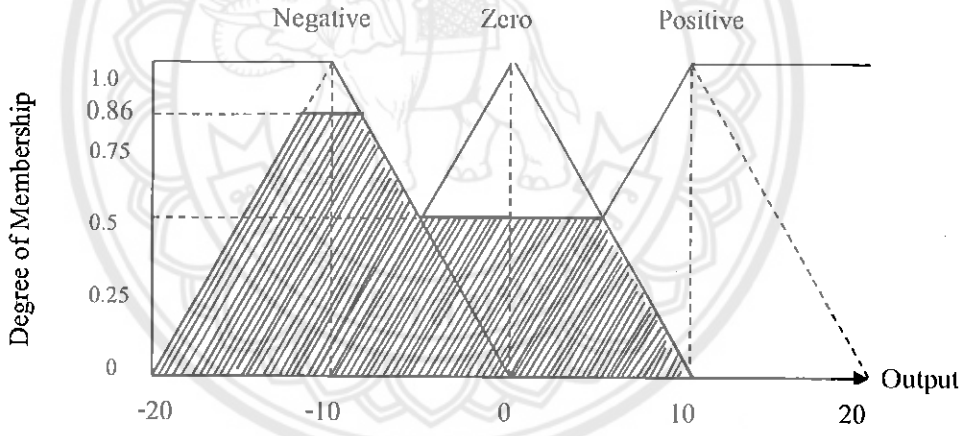
Output Z $= \sqrt{[\min(Z_E, Z_{CE})]^2}$

$= \sqrt{[\min(0.5,0.5)]^2}$

$= \sqrt{(0.5)^2}$

$= 0.5$

จะได้ผลลัพธ์ของแต่ละกฎคือพื้นที่ส่วนที่แรเงาในรูป 2.18



รูปที่ 2.18 จำนวนสมาชิกของเอาต์พุต

2.7.6 Defuzzification

การดีฟัซซิฟิเคชัน คือ การนำค่าที่ได้จากการอิน-เฟอะเร็นซ์มาคำนวณหาค่าที่เป็นผลลัพธ์เพื่อนำไปควบคุมระบบต่อไป [6]

การดีฟัซซิฟิเคชันมีหลายวิธีเช่น [5]

1. MAX-Membership Principle
2. Centroid Method
3. Weighted Average Method

4. Mean of Maximum (mom)

5. Centre of Sums

6. largest of maximum (lom)

7. Smallest (absolute) Value of Maximum (som)

จากตัวอย่างที่ 1 หลักจากที่ทำการอิน-เฟอะเร็นซ์โดยใช้วิธี Root-Sum-Square แล้ว ต่อไปจะทำการคำนวณดีฟัซซิฟิเคชัน ด้วยวิธี Weighted Average Method เพื่อหาผลลัพธ์ที่เกิดขึ้นได้ดังนี้ๆ

Output =

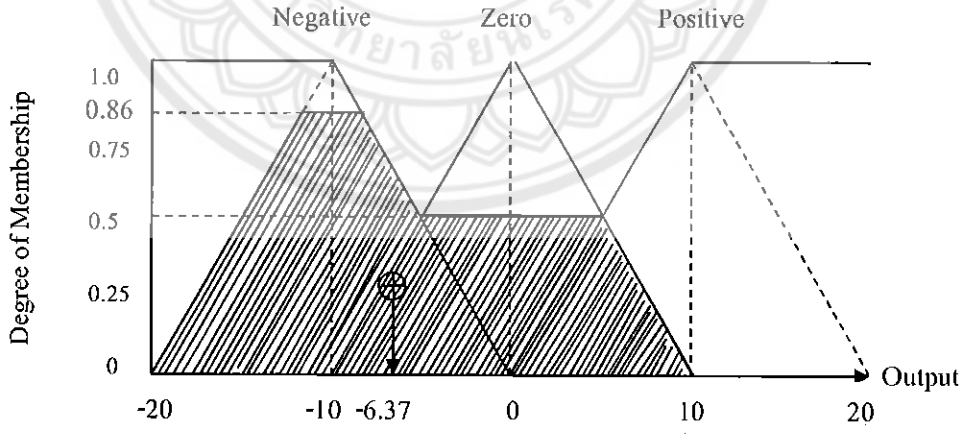
$$\frac{[(\text{สมาชิก}Negative \times CenterNegative) + (\text{สมาชิก}Zero \times CenterZero) + (\text{สมาชิก}Positive \times CenterPositive)]}{(\text{สมาชิก}Negative + \text{สมาชิก}Zero + \text{สมาชิก}Positive)}$$

$$\text{Output} = \frac{[(0.866 \times -10)+(0.5 \times 0)+(0 \times 10)]}{0.86+0.5+0}$$

$$\text{Output} = \frac{-8.66}{1.36}$$

$$\text{Output} = -6.37$$

เอาต์พุตที่ได้เท่ากับ -6.37% ซึ่งจะนำค่าที่ได้ไปใช้ในการปรับค่ากำลังไฟของระบบต่อไป



รูปที่ 2.19 จำนวนค่าเอาต์พุตที่ได้จากการคำนวณ Defuzzification

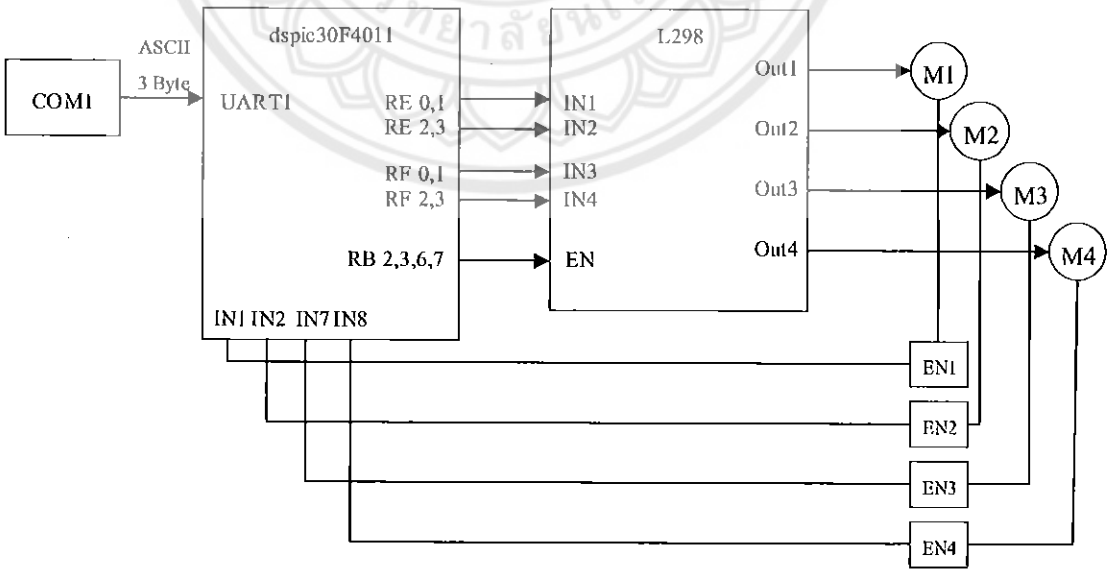
บทที่ 3
การออกแบบและทดลองระบบควบคุมแบบฟัซซี่

3.1 ภาพรวมของระบบการควบคุมมอเตอร์ DC โดยใช้ระบบควบคุมแบบฟัซซี่

เมื่อส่งค่าอินพุตที่มีค่าองศาทิศทางและค่ากำลังไฟฟ้าของหุ่นยนต์ผ่านโปรแกรม Dock light แล้ว ไมโครคอนโทรลเลอร์ dsPIC30F4011 จะเริ่มทำการคำนวณทิศทางและเซตพอยน์ของระบบ จากนั้น dsPIC30F4011 จะส่งค่าแรงดันไฟฟ้าที่คำนวณได้ให้บอร์ดขับมอเตอร์ L298 ซึ่ง L298 จะส่งแรงดันไฟฟ้าให้มอเตอร์แต่ละตัวเพื่อใช้ในการขับเคลื่อนหุ่นยนต์ ซึ่งมอเตอร์แต่ละตัวจะมีตัวนับความเร็วรอบใช้ทำหน้าที่ในการนับความเร็วรอบของมอเตอร์ จากนั้น dspic30F4011จะรับค่าตัวนับความเร็วรอบจากมอเตอร์ทั้ง 4 ตัว มาเก็บไว้โดยใช้ Input Capture ในการรับค่าตัวนับความเร็วรอบของมอเตอร์แต่ละตัว เมื่อ dsPIC30F4011 รับค่าความเร็วรอบจากมอเตอร์แล้วจะทำการกำหนดค่าเซตพอยน์หรือค่าความเร็วของแต่ละมอเตอร์ เพื่อนำมาเปรียบเทียบกับค่าความเร็วรอบในครั้งต่อไปโดยผ่านขบวนการทางฟัซซี่ลอจิก

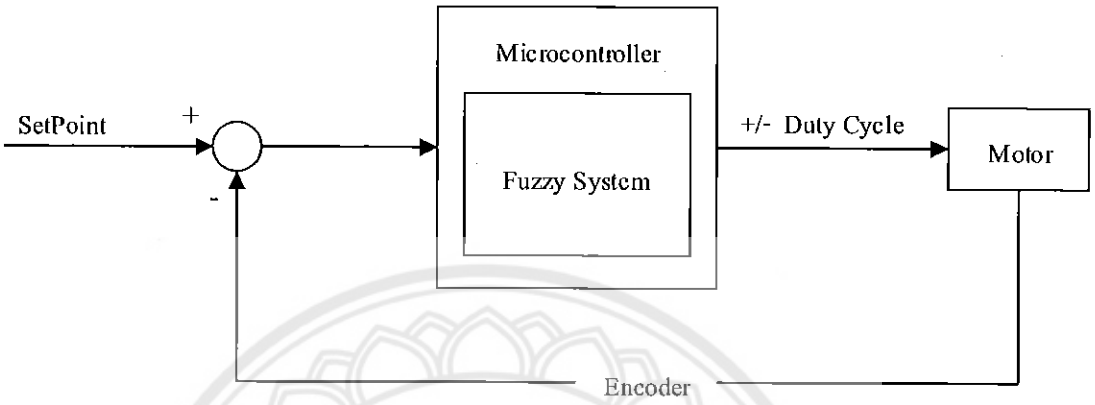
3.1.1. ระบบการทำงาน

รูปที่ 3.1 แสดงระบบการทำงานการส่งข้อมูลจากคอมพิวเตอร์ไปยังไมโครคอนโทรลเลอร์ dspic30F4011 พร้อมทั้งการทำงานของบอร์ดขับมอเตอร์ L298 และตัวนับรอบของแต่ละมอเตอร์



รูปที่ 3.1 ระบบการทำงานภายในหุ่นยนต์

รูปที่ 3.2 แสดงถึงระบบการควบคุมความเร็วของมอเตอร์ซึ่งประกอบด้วยตัวนับรอบมอเตอร์และระบบควบคุมแบบฟัซซี่



รูปที่ 3.2 ระบบการควบคุมความเร็วมอเตอร์

3.2 อุปกรณ์ที่ใช้

- มอเตอร์DC 12โวลต์ มีตัวนับรอบในตัว 4 ตัว
- วงจรรับค่าตัวนับรอบจากมอเตอร์
- วงจรแปลงไฟ 12โวลต์ เป็น 5โวลต์
- สายไฟสำหรับเชื่อมต่อวงจรต่างๆ
- หัวแร้งและอุปกรณ์ที่ต้องใช้ในการบัดกรี
- ET-PGM PIC USB V1
- โปรแกรม Docklight
- บอร์ด dsPIC30F เบอร์ 4011
- L298 พร้อมบอร์ดวงจร
- Serial Port สำหรับเชื่อมต่อยูอาร์ท
- ล้อโอมนิ จำนวน 4 ล้อ
- พาวเวอร์ซัพพลาย 12โวลต์ DC
- โปรแกรม MPLAB IDE
- โปรแกรม PICkit 2 v2.61

3.3 แนวคิดและการออกแบบ

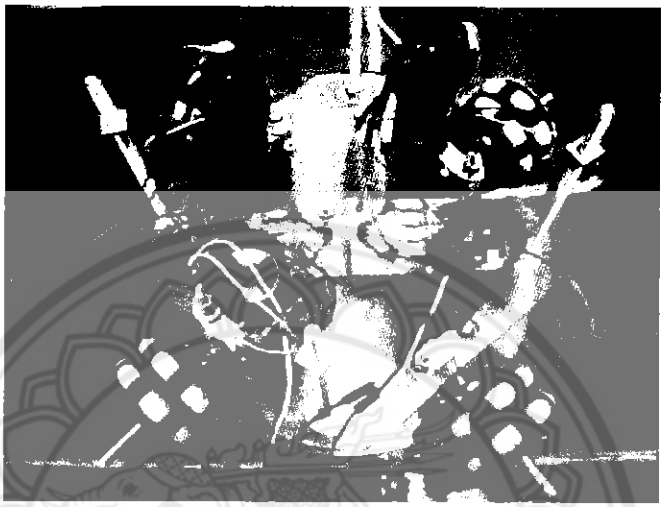
3.3.1 ทางด้าน Hardware

เริ่มจากติดตั้งมอเตอร์ DC 12โวลต์ที่มีตัวนับความเร็วรอบในตัวมอเตอร์ทั้งหมด 4 ตัวเข้ากับแผ่นอะคริลิกโดยวางมอเตอร์ทั้ง 4 ตัวในแนวตั้งเพื่อลดการใช้พื้นที่ของหุ่นยนต์และติดตั้งล้อโอมนิทั้ง 4 ล้อในแนวนอนโดยใช้เฟืองดอกจอกเป็นแกนยึดระหว่างมอเตอร์กับล้อโอมนิดังรูปที่ 3.3a และ 3.3b

รูปที่ 3.3a ล้อโอมนิกับมอเตอร์ยึดติดกันโดยใช้เฟืองดอกจอก

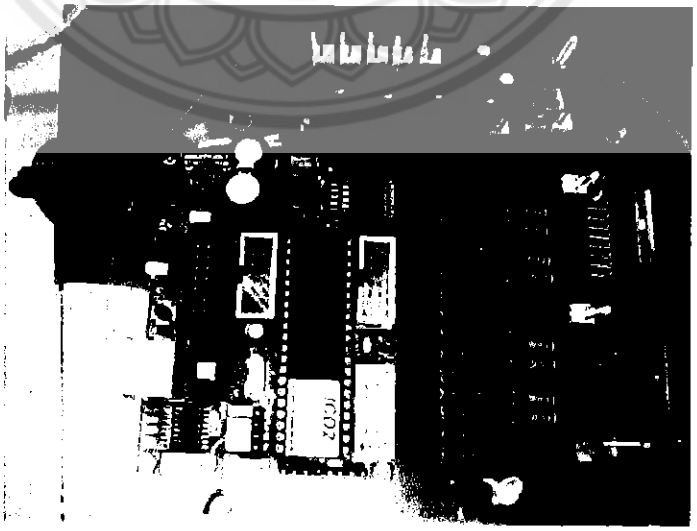
รูปที่ 3.3b ยึดเฟืองดอกจอกระหว่างมอเตอร์กับล้อโอมนิทั้ง 4 ล้อ

เมื่อทำการยึดมอเตอร์กับล้อโอมนิทั้ง 4 ล้อเสร็จแล้ว จากนั้นนำบอร์ดสำหรับรับค่าความเร็วรอบทั้งหมด 2 บอร์ด (บอร์ดรับค่าความเร็วรอบ 1 บอร์ดสามารถรับค่าความเร็วรอบจากมอเตอร์ได้ 2 ตัว) มาวางไว้ระหว่างมอเตอร์ทั้ง 4 ตัว เพื่อที่จะนำตัวนับรอบที่ติดอยู่กับมอเตอร์มาใช้งานได้ดังรูปที่ 3.4



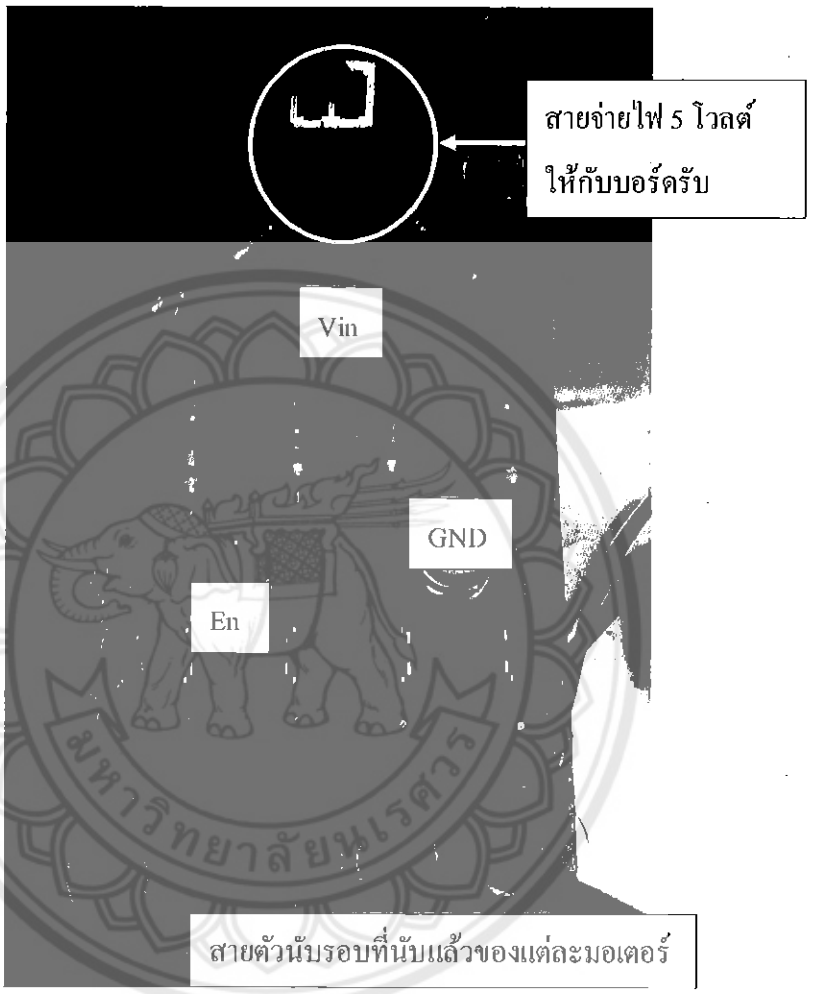
รูปที่ 3.4 นำบอร์ดรับค่าความเร็วรอบ 2 บอร์ด มายึดไว้ระหว่างมอเตอร์ทั้ง 4 ตัว
(หมายเหตุ: วงจรที่ใช้ทำบอร์ดรับค่าความเร็วรอบมอเตอร์ สามารถดูได้จากรูปที่ 2.9b)

จากนั้นนำบอร์ดวงจรแปลงไฟ 12 โวลต์เป็น 5 โวลต์บอร์ดวงจร L298 และบอร์ด dsPIC30F4011 มาติดตั้งบนแผ่นอะคริลิกแผ่นใหม่ให้เรียบร้อยดังรูปที่ 3.5



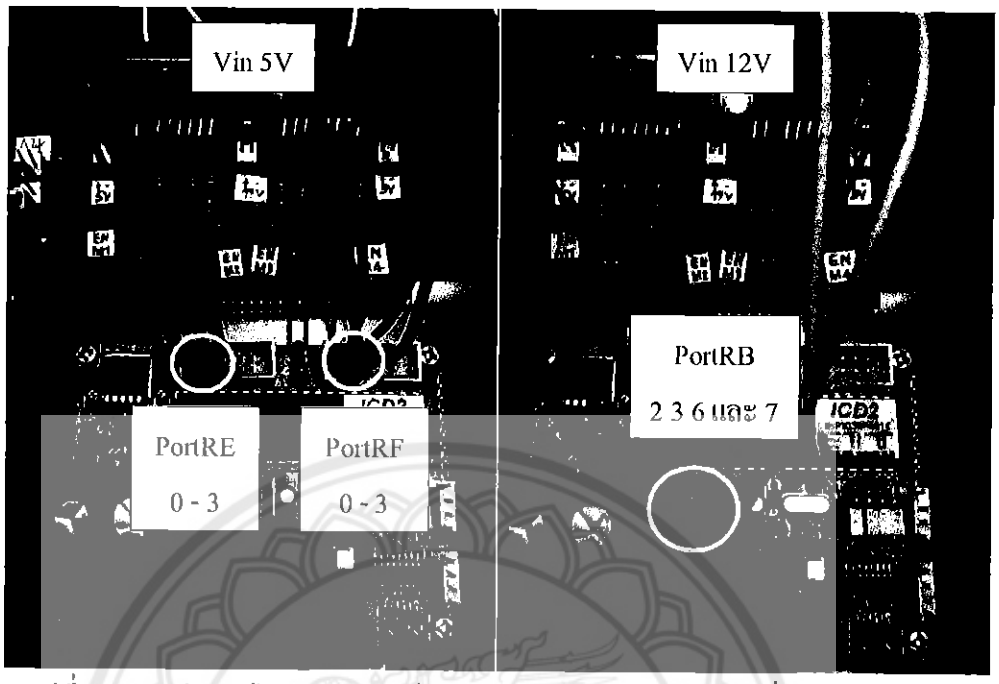
รูปที่ 3.5 บอร์ด dsPIC30F4011 วงจรแปลงไฟ 12 โวลต์ เป็น 5 โวลต์ L298 พร้อมบอร์ดวงจร

จากนั้นให้ทำการเชื่อมต่อสายไฟจากแต่ละจุดโดยเริ่มจากนำสาย Vin GND และสาย En ของตัวนับรอบต่อกับบอร์ดรับตัวนับรอบตามจุดดังรูปที่ 3.6 ต่อตามรูปทั้งหมด 4 ตัว



รูปที่ 3.6 การต่อสายไฟระหว่างตัวนับรอบกับบอร์ดตัวนับความเร็วรอบมอเตอร์

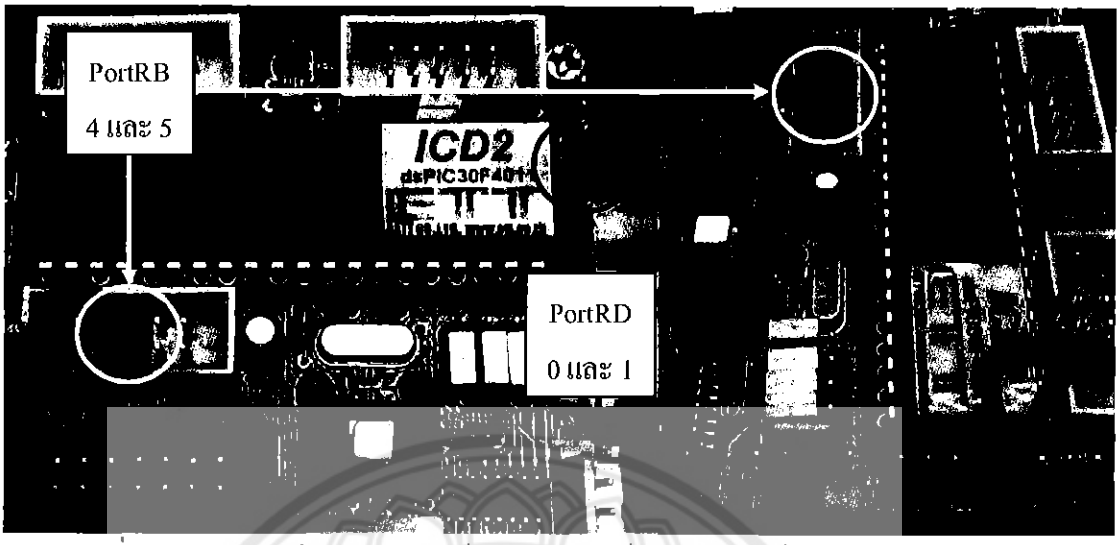
เมื่อเชื่อมต่อสายไฟของบอร์ดตัวนับความเร็วรอบของมอเตอร์เสร็จเรียบร้อยแล้ว ต่อไปให้ทำการต่อสายไฟของบอร์ด dspic30F4011 กับบอร์ดขับเคลื่อนมอเตอร์ L298 ดังรูปที่ 3.7



รูปที่ 3.7 การต่อสายไฟระหว่างบอร์ด dspic30F4011 กับบอร์ดขับเคลื่อนมอเตอร์ L298

จากรูปที่ 3.7 จะเห็นว่าบอร์ด dspic30F4011 ใช้ PortRE 0 - 3 และ PortRF 0 - 3 เป็นตัวจ่ายไฟให้กับ L298 เพื่อควบคุมความเร็วของมอเตอร์แต่ละตัวโดย PortRE 0 1 ทำหน้าที่ควบคุมความเร็วของมอเตอร์ตัวที่ 1 PortRE 2 3 ทำหน้าที่ควบคุมความเร็วของมอเตอร์ตัวที่ 2 PortRF 0 1 ทำหน้าที่ควบคุมความเร็วของมอเตอร์ตัวที่ 3 และ PortRF 2 3 ทำหน้าที่ควบคุมความเร็วของมอเตอร์ตัวที่ 4 ซึ่งการที่จะสั่งให้มอเตอร์แต่ละตัวให้ทำงานนั้นจะเป็นหน้าที่ของ PortRB 2 3 6 7 ที่ต่อเข้ากับ Enable ของ L298 โดย PortRB2 จะเป็นตัวสั่งให้มอเตอร์ตัวที่ 1 ทำงาน PortRB 3 จะเป็นตัวสั่งให้มอเตอร์ตัวที่ 2 ทำงาน PortRB6 จะเป็นตัวสั่งให้มอเตอร์ตัวที่ 3 ทำงานและPortRB 7 จะเป็นตัวสั่งให้มอเตอร์ตัวที่ 4 ทำงานตามลำดับ

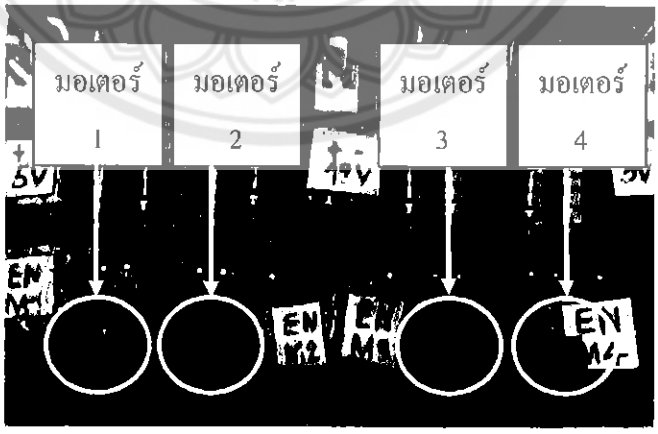
ต่อไปจะเป็นการต่อสายระหว่างบอร์ดตัวนับความเร็วรอบมอเตอร์กับบอร์ด dspic30F4011 ซึ่งตัวนับความเร็วรอบมอเตอร์นั้นจะใช้ Input Capture เป็นตัวนับความเร็วรอบมอเตอร์ ซึ่ง Input Capture ของ dspic30F4011 มีทั้ง 4 ตัวอยู่ที่ PortRD 0 1 และ PortRB 4 5 ดังรูปที่ 3.8



รูปที่ 3.8 การต่อสายไฟระหว่างบอร์ดตัวนับความเร็วรอบมอเตอร์กับบอร์ด dspic30F4011

จากรูปที่ 3.8 PortRD 0 จะทำหน้าที่เป็นตัวนับความเร็วรอบของมอเตอร์ตัวที่ 1 PortRD 1 จะทำหน้าที่เป็นตัวนับความเร็วรอบของมอเตอร์ตัวที่ 2 PortRB 4 จะทำหน้าที่นับรอบ Encoder ของมอเตอร์ตัวที่ 3 และ PortRB 5 จะทำหน้าที่นับรอบความเร็วรอบของมอเตอร์ตัวที่ 4 ตามลำดับ

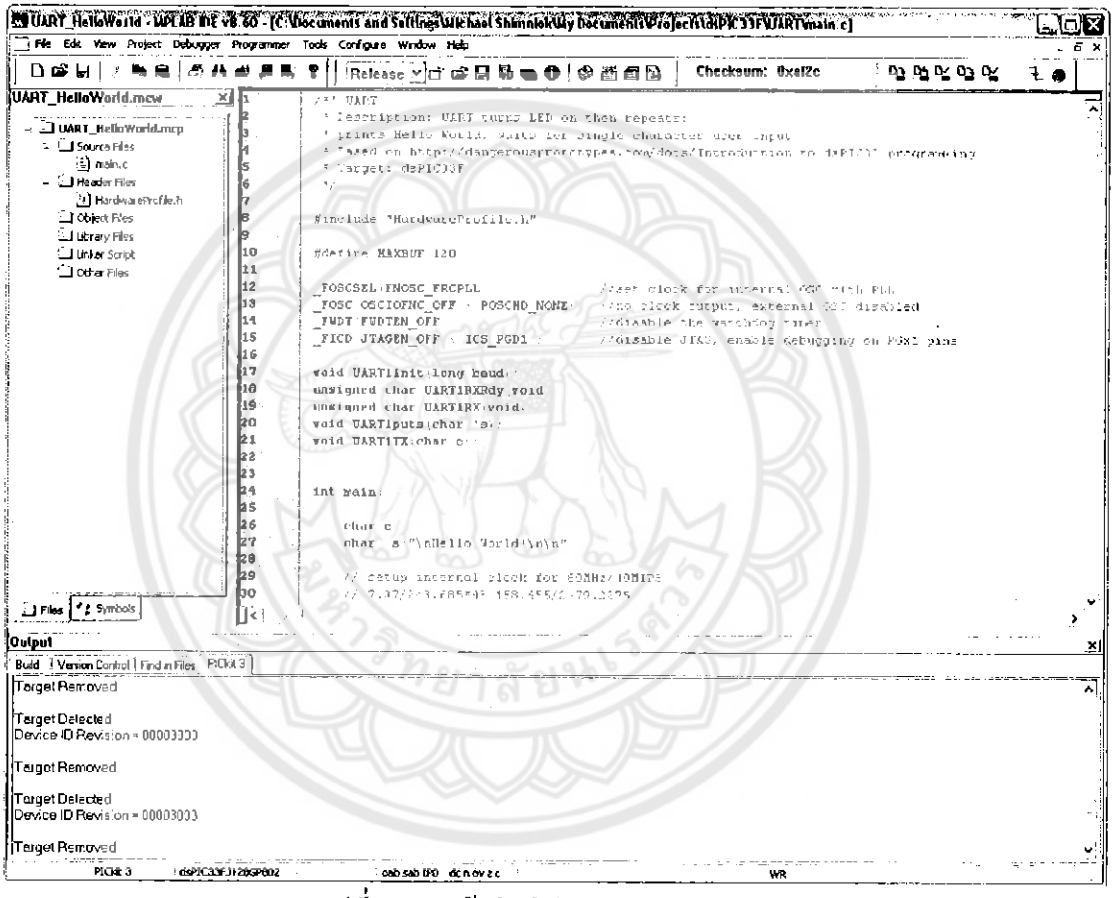
สุดท้ายจะเป็นการต่อสายไฟระหว่างมอเตอร์กับบอร์ด L298 ซึ่งบอร์ด L298 จะทำหน้าที่ขับเคลื่อนมอเตอร์ให้เป็นไปตามที่ไมโครคอนโทรลเลอร์ dspic30F4011 สั่งการมา สามารถดูการต่อสายไฟระหว่างมอเตอร์กับบอร์ด L298 ได้จากรูปที่ 3.9



รูปที่ 3.9 การต่อสายไฟระหว่างมอเตอร์แต่ละตัวกับบอร์ดขับเคลื่อนมอเตอร์ L298

3.3.2 ทางด้าน Software

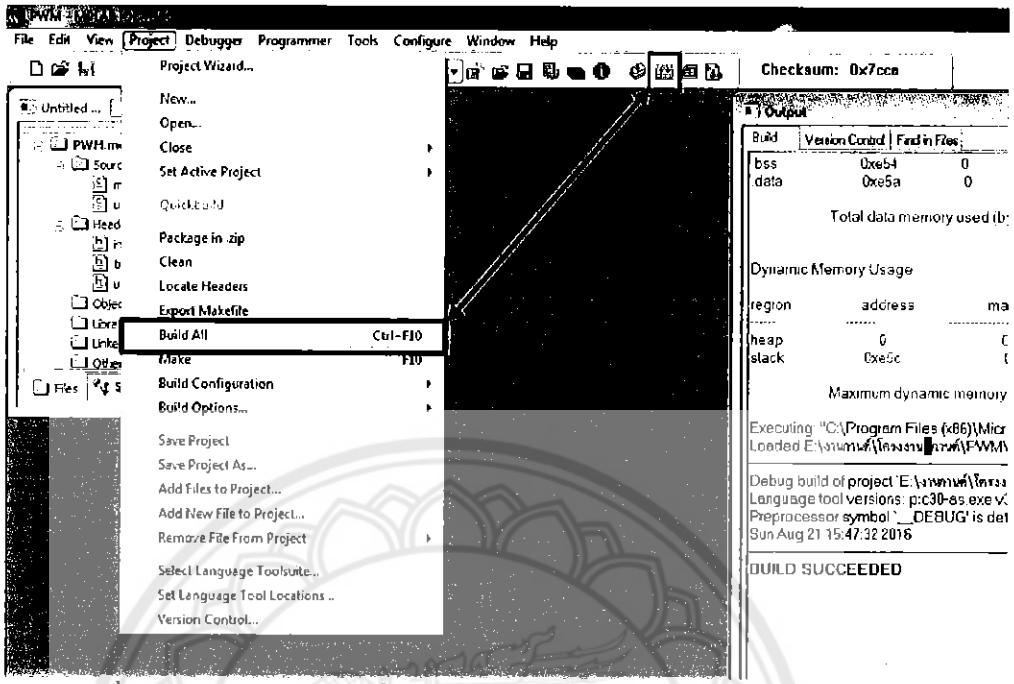
การเขียนโปรแกรมให้กับหุ่นยนต์นั้นจะใช้โปรแกรม MPLAB IDE V. 8.6x ขึ้นไปเนื่องจากโปรแกรม MPLAB IDE V. 8.6x มีการรองรับการทำงานของบอร์ด dspic30F4011 และ MPLAB IDE มีตัวคอมไพเลอร์ (Compiler) ที่สามารถเรียกใช้ได้มากมาย เช่น CCS-C HI-TECH SDCC C18 C30 ซึ่งในที่นี้จะใช้ตัว C30 ในการคอมไพล์ เพราะภาษาที่ใช้สั่งการไมโครคอนโทรลเลอร์นั้นคือภาษา C [15]



รูปที่ 3.10 หน้าต่างโปรแกรม MPLAB IDE

หลังจากทำการเขียนโปรแกรมสำหรับควบคุมหุ่นยนต์ผ่านโปรแกรม MPLAB IDE 8.6x เสร็จเรียบร้อยแล้วให้เราทำการคอมไพล์ตัวโค้ด เพื่อตรวจสอบความถูกต้องและทำการสร้าง File .Hex เพื่อจะนำไปใช้กับบอร์ดไมโครคอนโทรลเลอร์ dspic30F4011

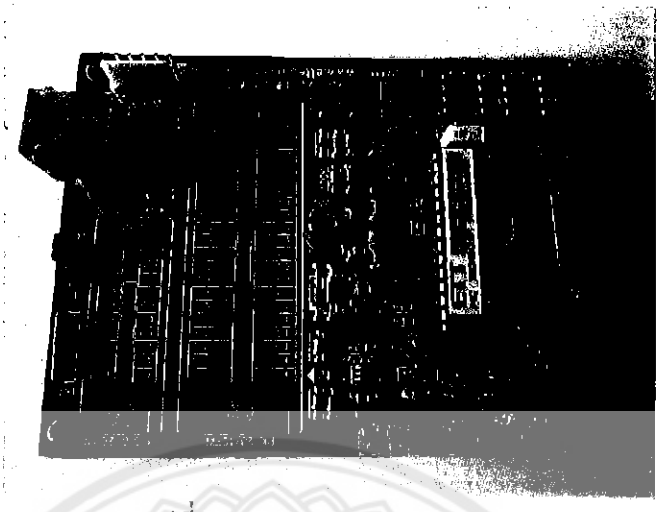
วิธีการคอมไพล์เพื่อสร้าง File .Hex ของโปรแกรม MPLAB IDE 8.6x นั้นสามารถกด CTRL + F10 ได้ทันที หรือจะกดปุ่มดังรูปที่ 3.11 ได้เช่นกัน



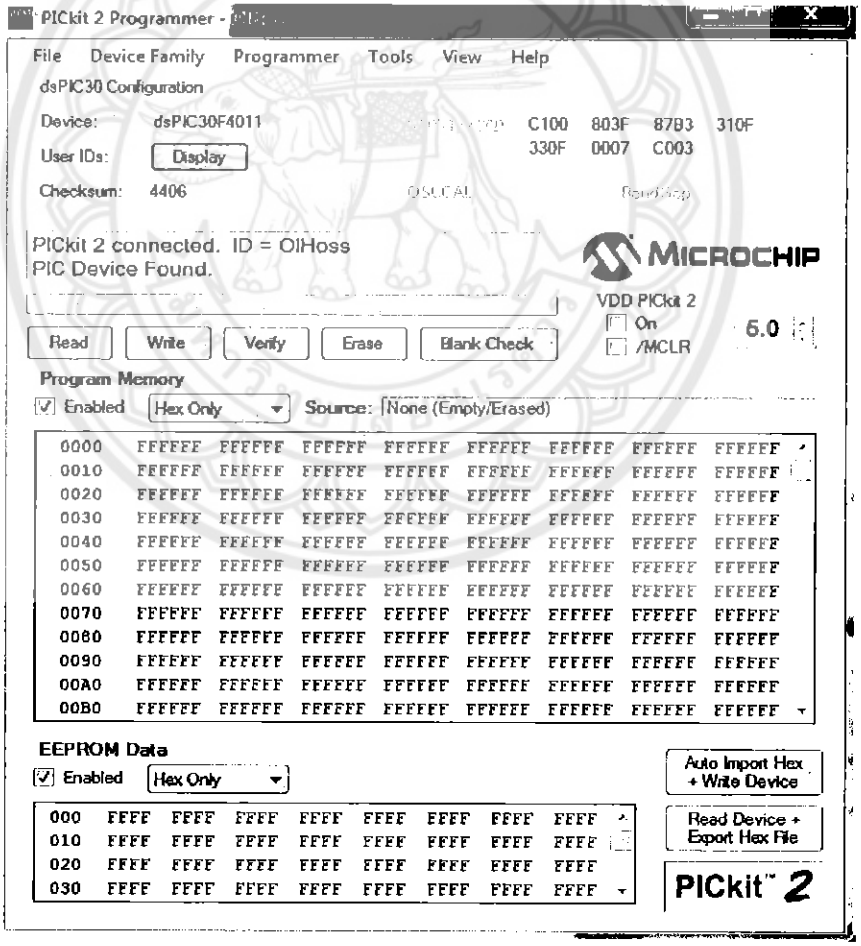
รูปที่ 3.11 วิธีการคอมไพล์เพื่อสร้าง File .Hex ของโปรแกรม MPLAB IDE 8.6x

หลังจากที่ทำการ Build All แล้วโปรแกรม MPLAB IDE 8.6x จะทำการคอมไพล์โค้ด และตรวจสอบไวยากรณ์ (Syntax) ของภาษาที่ใช้ว่าถูกต้องหรือไม่ ถ้าโค้ดทุกอย่างถูกต้อง โปรแกรมจะทำการสร้าง File .Hex ขึ้นมาและจะขึ้นแจ้งเตือนว่า "Build Succeeded" ซึ่งสามารถนำ File .Hex เข้าสู่กระบวรการส่ง File .Hex เข้าสู่ไมโครคอนโทรลเลอร์ต่อไป

การส่ง File .Hex เข้าสู่ไมโครคอนโทรลเลอร์นั้นจะใช้โปรแกรมที่มีชื่อว่า PICkit 2 v2.61 ซึ่งโปรแกรมนี้จะใช้งานได้ต้องมีอุปกรณ์ที่มีชื่อว่า ET-PGM PIC USB V1 ดังรูป 3.12 ซึ่งเป็นเครื่องโปรแกรมไมโครคอนโทรลเลอร์ตระกูล PIC วิธีการใช้งานคือให้เชื่อมต่อระหว่าง ET-PGM PIC USB V1 และคอมพิวเตอร์ผ่านทางสาย USB ที่อุปกรณ์มีมาให้และนำ ET-PGM PIC USB V1 เชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ dspic30F4011 และเปิดโปรแกรม PICkit 2 v2.61 ขึ้นมาดังรูปที่ 3.13

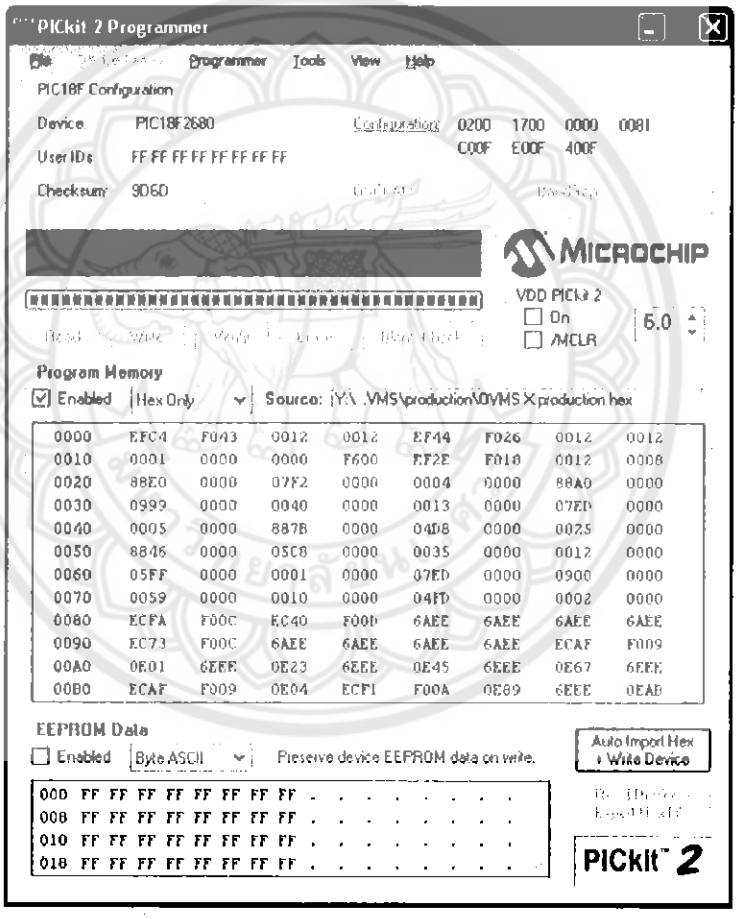


รูปที่ 3.12 ET-PGM PIC USB V1



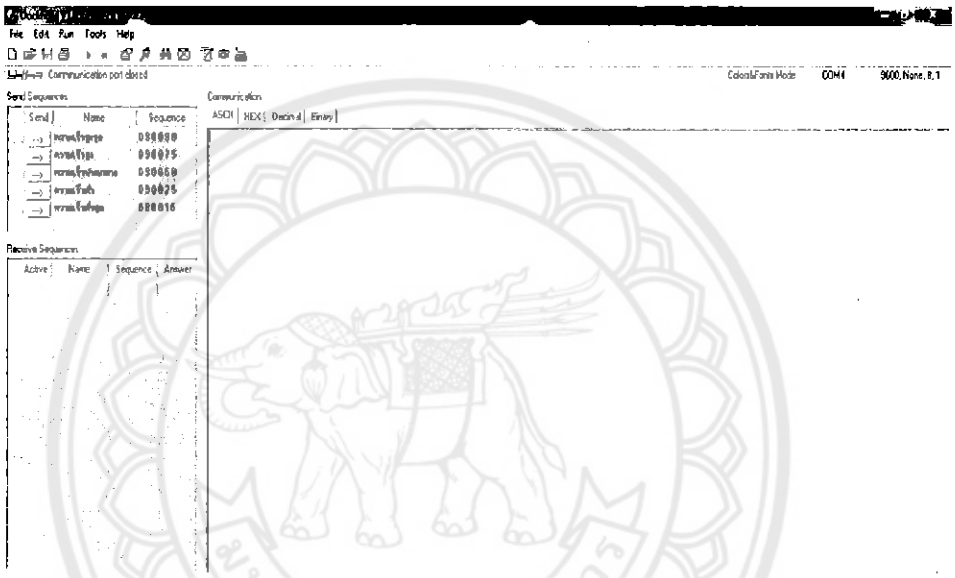
รูปที่ 3.13 หน้าต่างโปรแกรม PICkit 2 v2.61

ถ้าโปรแกรม PICkit 2 v2.61 ขึ้นตามรูปที่ 3.11 หมายความว่าการเชื่อมต่อระหว่าง บอร์ดไมโครคอนโทรลเลอร์ dspic30F4011 และคอมพิวเตอร์ ได้สมบูรณ์แบบ สามารถส่ง File.Hex ผ่านโปรแกรม PICkit 2 v2.61ได้ โดยกดปุ่ม Auto Import Hex + Write Device และเลือก File .Hex เมื่อทำการเลือก File. Hex แล้วโปรแกรม PICkit 2 v2.61 จะเริ่มทำการส่ง File .Hex เข้าสู่ไมโครคอนโทรลเลอร์ dspic30F4011 ซึ่งตรงจุดนี้จะใช้เวลาประมาณ 10 – 20 วินาที เมื่อโปรแกรม PICkit 2 v2.61 ทำงานเสร็จแล้วจะมีข้อความขึ้นบนหน้าจอสีเขียวว่า “Programming Successful. Waiting for file update... (Click button again to exit)” ดังรูปที่ 3.14

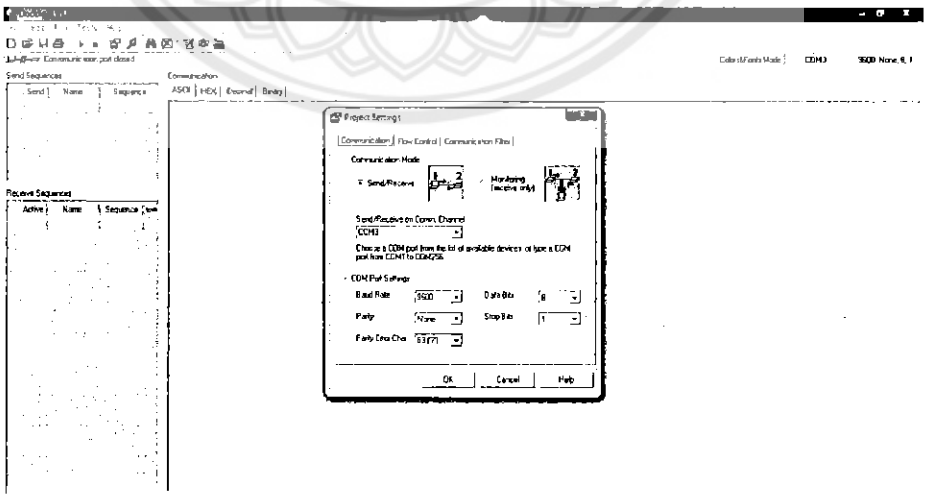


รูปที่ 3.14 หน้าต่างโปรแกรม PICkit 2 v2.61 ตอนส่ง File .Hex สำเร็จ

เมื่อทำการส่ง File .Hex เสร็จแล้ว ต่อไปจะเป็นขั้นตอนการทดสอบการทำงานของระบบ โดยใช้โปรแกรม Docklight ในการส่งข้อมูลที่เราต้องการทดสอบผ่านทาง USB Serial Port เพื่อเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ dspic30F4011 ผ่านทางยูอาร์ทโดยเริ่มจากทำการตั้งค่า ComPort ให้ตรงกับพอร์ตที่ USB Serial Port เชื่อมต่อกับคอมพิวเตอร์ไว้ (สามารถตรวจสอบได้จาก Device Manager -> Port) และตั้งค่า Baud Rate ให้ตรงกับค่าที่คำนวณไว้ในโปรแกรมดังรูปที่ 3.16



รูปที่ 3.15 หน้าต่างโปรแกรม Docklight



รูปที่ 3.16 การตั้งค่า Comport Channel และตั้งค่า Baud Rate ของโปรแกรม Docklight

หมายเหตุ: วิธีเข้าหน้าต่าง Settings ให้ไปที่ Tools -> Project Settings...

3.4 กระบวนการหาค่าอินพุตและเอาต์พุตของระบบ

ก่อนที่จะมีการวิเคราะห์ระบบด้วยวิธีการแบบฟัซซี่ (Fuzzy Analysis) จะต้องมีการคำนวณหาค่า อินพุต และ เอาต์พุตของระบบก่อน โดยค่า Error และ Change of Error จะเป็นอินพุตของระบบควบคุมแบบฟัซซี่ ซึ่งค่า Error และ Change of Error จะมีค่าสูงสุดอยู่ที่ 100 และมีค่าต่ำสุดอยู่ที่ -100 เพราะความเร็วรอบสูงสุดของมอเตอร์ตอนไม่มีสิ่งกีดขวาง Encoder สามารถนับได้อยู่ที่ประมาณ 93 - 95 รอบต่อวินาที จึงทำให้ค่า Error และ Change of Error ที่เกิดขึ้นไม่มีทางเกิน 100 และ – 100 แน่นอน ส่วนวิธีการคำนวณหาค่า Error และ Change of Error สามารถคำนวณได้จากสมการ Error = เซตพอยน์ - ค่าความเร็วรอบของแต่ละมอเตอร์ และ Change of Error = $Error_{ปัจจุบัน} - Error_{ก่อนหน้า}$

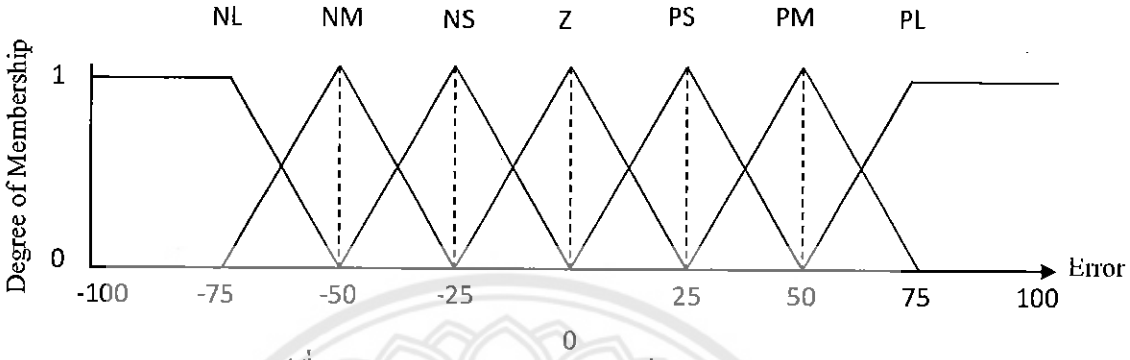
ซึ่งวิธีการคำนวณหาค่า Error และ Change of Error ในระบบจริงนั้น ระบบจะทำการเซตค่าเซตพอยน์ไว้ของแต่ละมอเตอร์ และเมื่อระบบรับค่าความเร็วรอบของมอเตอร์มาแล้ว ระบบจะทำการหาค่า Error ที่เกิดขึ้นโดยใช้สมการ Error = เซตพอยน์ – ความเร็วรอบของแต่ละมอเตอร์ ส่วนค่า Change of Error นั้นจะเกิดขึ้นหลังจากระบบผ่านการทำงานแล้ว 1 ครั้งเพราะว่าการเปลี่ยนแปลงค่าของ Error นั้นระบบต้องนำค่า Error ที่เกิดขึ้นในปัจจุบันมาเปรียบเทียบกับค่า Error ในรอบก่อนหน้านี้ เพราะฉะนั้นระบบจะทำการคำนวณหาค่า Change of Error ที่เกิดขึ้นโดยสมการที่ใช้หาค่า Change of Error คือ $Error_{ปัจจุบัน} - Error_{ก่อนหน้า}$ ค่า Change of Error นั้นเปรียบเสมือนรถที่กำลังวิ่งโดยวินาทีแรกความเร็วของรถอยู่ที่ 60 กิโลเมตรต่อชั่วโมง หลังจากผ่านไป 1 วินาที (เข้าวินาทีที่ 2) ความเร็วของรถอยู่ที่ 70 กิโลเมตรต่อชั่วโมงซึ่งหมายความว่าความเร็วของรถเปลี่ยนไปคือ 10 กิโลเมตรต่อชั่วโมงตรงจุดนี้คือค่า Change of Error ที่เกิดขึ้น

ส่วนค่าเอาต์พุตของระบบคือ ค่ากำลังไฟฟ้าที่ใช้สำหรับการเปลี่ยนแปลงความเร็วของแต่ละมอเตอร์อาจจะเป็นมีค่าเป็นบวกหรือลบก็ได้ ขึ้นอยู่กับสภาวะแวดล้อมที่เปลี่ยนไป

3.4.1 กำหนดสับเซตฟัซซี่ สำหรับ Inputs

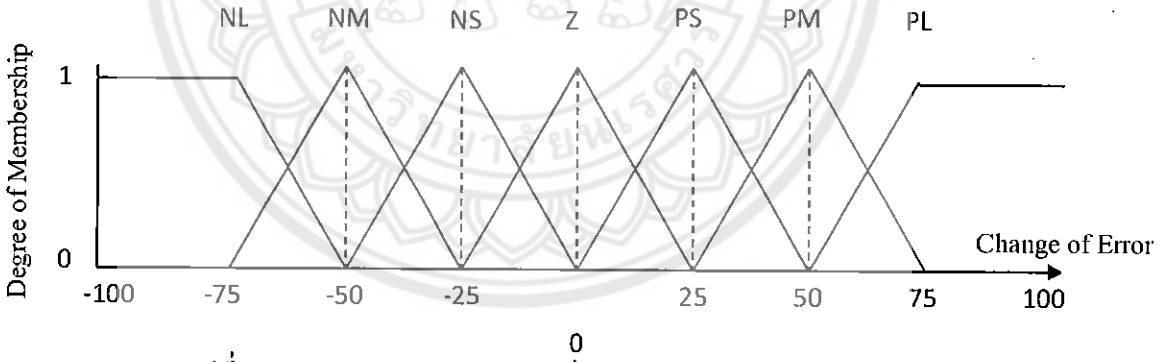
ขั้นตอนแรกการวิเคราะห์ระบบด้วยวิธีการแบบฟัซซี่ คือ การแบ่งเซตของ Error และ Change of Error เป็น 7 สับเซต ได้แก่ Positive Large (PL) Positive Medium (PM) Positive Small (PS) Zero (Z) Negative Small (NS) Negative Medium (NM) Negative Large (NL)

3.4.2 กำหนดค่าความสัมพันธ์สำหรับอินพุตของฟัซซี่



รูปที่ 3.17 รูปกราฟค่าความสัมพันธ์สำหรับอินพุต Error

จากรูปที่ 3.17 ค่าความสัมพันธ์สำหรับอินพุตของฟัซซี่ แสดงให้เห็นถึงค่าความเป็นสมาชิก (Degree of Membership) ของแต่ละอินพุตในแต่ละสับเซตและจากรูปที่ 3.17 นั้นเป็นกราฟของ Error ซึ่งเป็นแบบสามเหลี่ยม ประกอบด้วย 7 สับเซต ได้แก่ Positive Large (PL) Positive Medium (PM) Positive Small (PS) Zero (Z) Negative Small (NS) Negative Medium (NM) Negative Large (NL)



รูปที่ 3.18 รูปกราฟค่าความสัมพันธ์สำหรับอินพุตของ Change of Error

จากรูปที่ 3.18 จะเป็นกราฟค่าความสัมพันธ์ของ Change of Error ซึ่งเป็นแบบสามเหลี่ยม ประกอบด้วย 7 สับเซต ได้แก่ Positive Large (PL) Positive Medium (PM) Positive Small (PS) Zero (Z) Negative Small (NS) Negative Medium (NM) Negative Large (NL)

จากรูปที่ 3.17 และ 3.18 จะเห็นว่ากราฟที่ได้เป็นแบบสามเหลี่ยมทั้ง 2 กราฟ เป็นฟังก์ชันที่นิยมใช้เกี่ยวกับการควบคุมที่ต้องการความรวดเร็วให้ทันเวลาเนื่องจากมีการคำนวณน้อยและมีความเหมาะสมของการครอบคลุมของข้อมูลที่จะรับเข้ามา

ส่วนการแบ่งเซต ออกเป็น 7 สับเซต เพราะจะได้มีการควบคุมที่หลากหลายและละเอียดยิ่งขึ้นซึ่งถ้าเปรียบเทียบกับตัวอย่างที่ 1 (ในหน้าที่ 23) ซึ่งมีการควบคุมความเร็วแค่ เร็วขึ้น (P) ไม่เปลี่ยนแปลง (Z) และช้าลง (N) แต่ในที่นี้มีการควบคุมถึง 7 อย่างคือ เร็วขึ้นมาก (PL) เร็วขึ้นปานกลาง (PM) เร็วขึ้นนิดหน่อย (PS) ไม่เปลี่ยนแปลง (Z) ช้าลงเล็กน้อย(NS) ช้าลงปานกลาง (NM) ช้าลงมาก (NL) ทำให้มีความละเอียดในการควบคุมมากขึ้น

3.4.3 กำหนดสับเซตเพื่อหาค่าความเป็นสมาชิกของอินพุตฟัซซี่

สำหรับโครงงานนี้ Set ของเอาท์พุตแบ่งเป็น 7 สับเซต ได้แก่ Positive Large (PL) Positive Medium (PM) Positive Small (PS) Zero (Z) Negative Small (NS) Negative Medium (NM) Negative Large (NL)

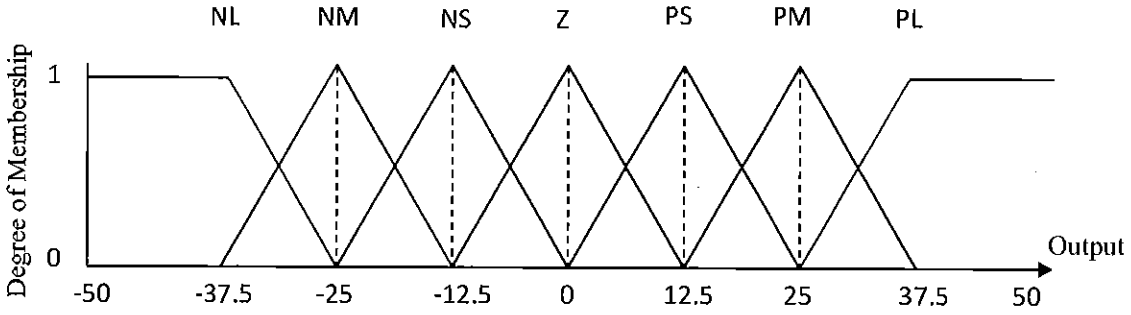
3.4.4 กำหนดค่าความเป็นสมาชิกสำหรับเอาต์พุตฟัซซี่

เอาต์พุต ในที่นี้คือค่ากำลังไฟฟ้าที่จะนำไปปรับในส่วนของพีดับเบิลยูเอมส่วนค่าความเป็นสมาชิกที่ออกแบบไว้จะมีส่วนหนึ่งที่เรียกว่า "Singleton" ซึ่งเป็นค่าตัวเลขค่าหนึ่งในแต่ละสับเซตของเอาต์พุต

ในโครงงานนี้ช่วงของค่าเอาต์พุตที่จะนำไปใช้งานจริง (Crisp Output) คือ -50 ถึง 50 และ Singleton ของแต่ละค่าคือ Positive Large (PL) = 37.5 Positive Medium (PM) = 25 Positive Small (PS) = 12.5 Zero (Z) = 0 Negative Small (NS) = -12.5 Negative Medium (NM) = -25 Negative Large (NL) = -37.5

ซึ่งเราได้กำหนดค่าเอาต์พุตสูงสุดเท่ากับ 50% และต่ำสุดเท่ากับ -50% โดยที่ค่าเอาต์พุตเป็นค่ากำลังไฟฟ้าที่ต้องให้ระบบเพิ่มหรือลดค่ากำลังไฟฟ้า คือ ถ้าเอาต์พุตมีค่าเท่ากับ 50% ระบบจะมีการปรับค่ากำลังไฟฟ้าที่ต้องการขึ้นอีก 50% แต่ถ้าเอาต์พุตมีค่าเท่ากับ -50% ระบบจะมีการปรับค่ากำลังไฟฟ้าลดลงไป 50%

เหตุผลที่แบ่งสับเซตของเอาต์พุตเป็นเช่นนี้เพราะจะได้มีการควบคุมที่หลากหลายและทำให้มีความละเอียดในการควบคุมมากขึ้น



รูปที่ 3.19 รูปกราฟของเอาต์พุตฟัซซี่ในแต่ละสับเซต

3.5 การสร้าง Rule matrix

ผลการตอบสนองของระบบถูกกำหนดโดยกฎ (Rule) ธรรมดาคือ If (A and B) Then C ในส่วนของ If คือ เงื่อนไขที่จะเกิดขึ้น ส่วน Then คือผลที่จะเกิดขึ้นของกฎนั้นๆ ส่วนที่ตามหลัง Then เรียกว่า "The Consequent" ไว้อธิบายสถานะของ เอาต์พุตฟัซซี่ ของระบบนั้นๆ ส่วน A B C คือ ประโยค Logic ซึ่งในระบบควบคุมแบบฟัซซี่นั้นมีค่าความจริงหรือค่าความเป็นสมาชิกอยู่ระหว่าง 0 ถึง 1

จำนวนของกฎส่วนใหญ่จะมีจำนวนเท่ากับ N x M เมื่อ N คือจำนวนสับเซต ของ Error และ M คือ จำนวนสับเซตของ Change of Error ขณะนี้ค่าของ N = M = 7 ทำให้มีกฎทั้งหมด 49 กฎ ตัวอย่างเช่น

If Error is Negative Large and Change of Error is Negative Large Then Output is Zero

If Error is Negative Medium and Change of Error is Negative Large Then Output is Positive Small

If Error is Negative Small and Change of Error is Negative Large Then Output is Positive Medium

If Error is Zero and Change of Error is Negative Large Then Output is Positive Large

If Error is Positive Small and Change of Error is Negative Small Then Output is Positive Medium

If Error is Positive Medium and Change of Error is Zero Then Output is Positive Medium

สามารถแสดงกฎทั้งหมดได้ดังตารางที่ 3.1

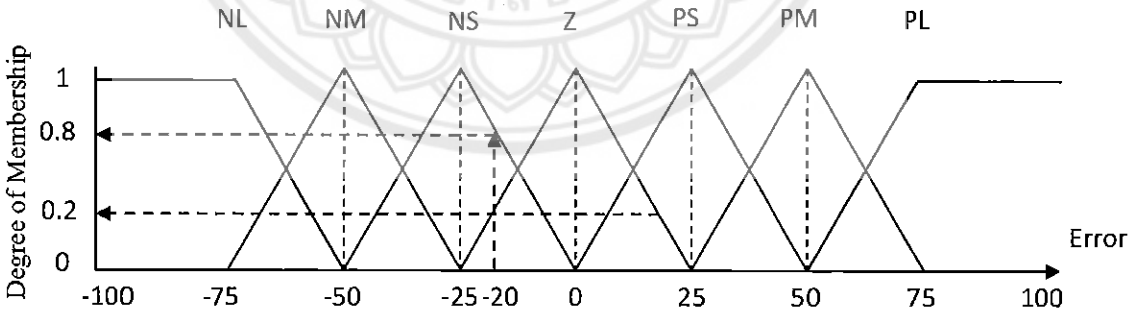
ตารางที่ 3.1 แสดงกฎ X = ฟัซซี่เอาต์พุต

CE / E	NL	NM	NS	Z	PS	PM	PL
NL	X = Z	X = PS	X = PM	X = PL	X = PL	X = PL	X = PL
NM	X = NS	X = Z	X = PS	X = PM	X = PL	X = PL	X = PL
NS	X = NM	X = NS	X = Z	X = PS	X = PM	X = PL	X = PL
Z	X = NL	X = NM	X = NS	X = Z	X = PS	X = PM	X = PL
PS	X = NL	X = NL	X = NM	X = NS	X = Z	X = PS	X = PM
PM	X = NL	X = NL	X = NL	X = NM	X = NS	X = Z	X = PS
PL	X = NL	X = NL	X = NL	X = NL	X = NM	X = NS	X = Z

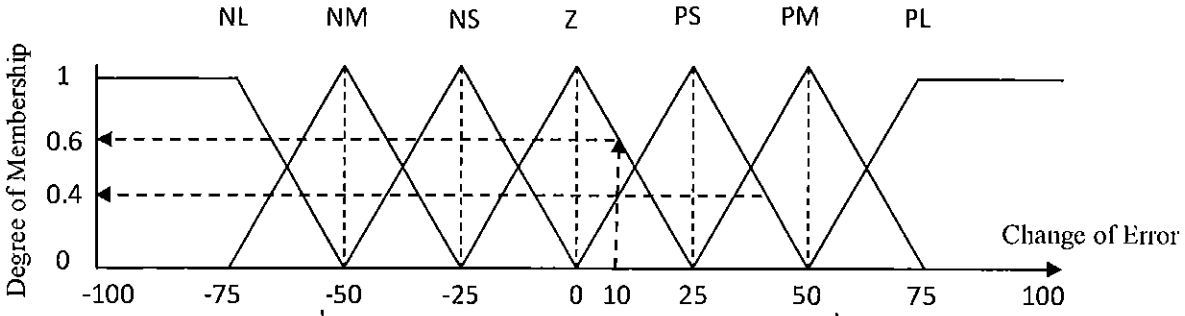
ค่าความเป็นสมาชิกของฟัซซี่เอาต์พุตจะขึ้นอยู่กับค่าความเป็นสมาชิกของอินพุตที่กำหนดไว้ดังรูปที่ 3.17 และ 3.18

ตัวอย่างที่ 2 สมมุติว่า Error = -20 และ Change of Error = 10

จากค่าความสัมพันธ์ของอินพุตกำหนดไว้ ดังรูปที่ 3.17 และ 3.18 จะได้ค่าความเป็นสมาชิกของอินพุตดังรูปที่ 3.20a และ 3.20b



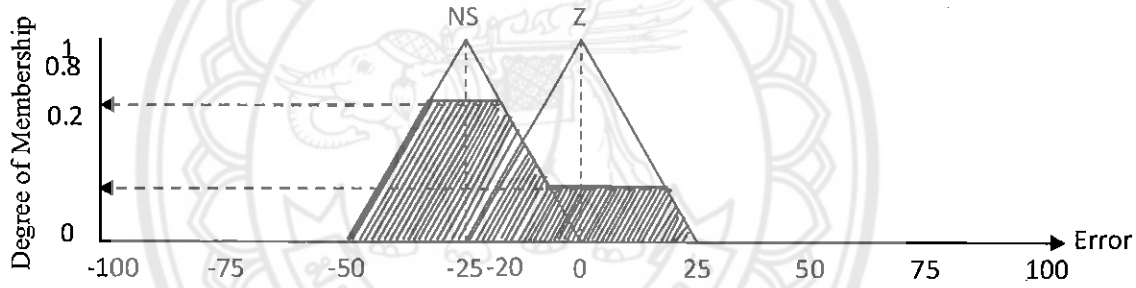
รูปที่ 3.20a รูปกราฟ Error จากตัวอย่างที่ 2



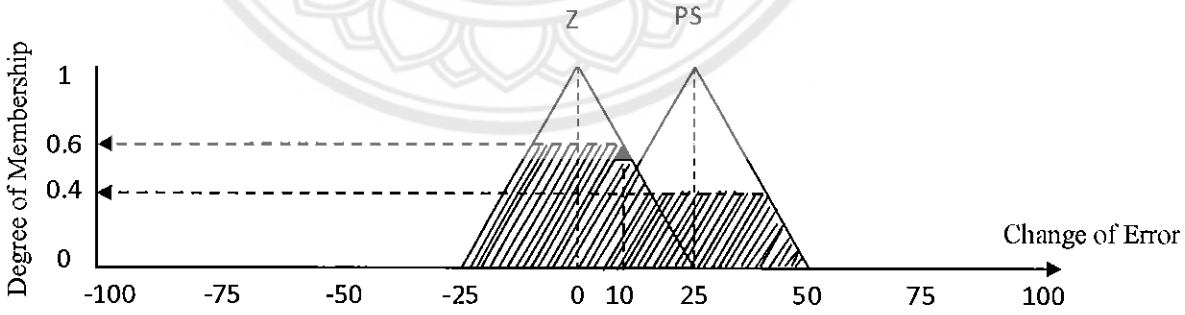
รูปที่ 3.20b รูปกราฟ Change of Error จากตัวอย่างที่ 2

3.6 ขบวนการ Inference

เป็นการนำค่าความเป็นสมาชิกของแต่ละสับเซตของทุกอินพุตมาเปรียบเทียบกับกฎที่กำหนดขึ้น เพื่อหาค่าความเป็นสมาชิกของเอาต์พุตฟัซซี่ในแต่ละกฎ ด้วยขบวนการอิน-เฟอะเร็นซ์โดยใช้วิธี ROOT-SUM-SQUARE (RSS)



รูปที่ 3.21a รูปกราฟ Error หลังทำการอิน-เฟอะเร็นซ์



รูปที่ 3.21b รูปกราฟ Change of Error หลังทำการอิน-เฟอะเร็นซ์

ค่าความเป็นสมาชิกของแต่ละเอาต์พุตฟัซซี่คำนวณได้จาก

$$x_i = \sqrt{\sum_{i=1}^{n} (\min(E_i, CE_i))^2} \qquad (3.1)$$

โดย E_i และ CE_i คือค่าความเป็นสมาชิก (Degree of Membership) ของ Error and Change of Error จากตารางที่ 3.1 จะได้ค่าความเป็นสมาชิกของแต่ละกฎดังนี้

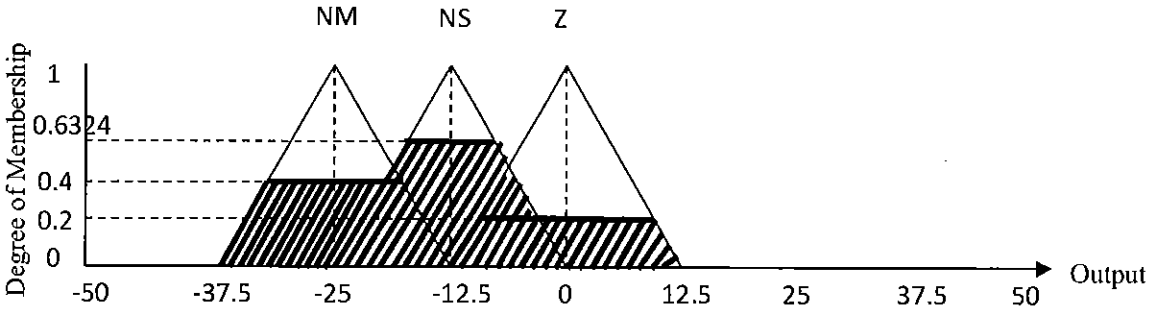
$$\text{Output NM} = \sqrt{[\min(NS_e, PS_{ce})]^2} = \sqrt{[min(0.8,0.4)]^2} = \sqrt{(0.4)^2}$$
$$= 0.4$$
$$\text{Output NS} = \sqrt{[\min(NS_e, Z_{ce})]^2 + [\min(Z_e, PS_{ce})]^2}$$
$$= \sqrt{[\min(0.8,0.6)]^2 + [\min(0.2,0.4)]^2}$$
$$= \sqrt{(0.6)^2 + (0.2)^2}$$
$$= \sqrt{0.36 + 0.04} = \sqrt{0.4} = 0.6324$$
$$\text{Output Z} = \sqrt{[\min(Z_e, Z_{ce})]^2} = \sqrt{[min(0.2,0.6)]^2} = \sqrt{(0.2)^2}$$
$$= 0.2$$

3.7 ขบวนการ Defuzzification

เมื่อได้ค่าความเป็นสมาชิกของแต่ละกฎแล้วต่อไปจะเป็นการคำนวณหาจำนวนสมาชิกของเอาต์พุตโดยใช้วิธีหา Weighted Average ดังสมการ

$$X = \frac{\sum_{i=1}^{n}(xi \times Si)}{\sum_{i=1}^{n}(xi)} \qquad (3.2)$$

โดย $x_i = \sqrt{\sum_{i=1}^{n}(\min(E_i, CE_i))^2}$ และ S_i คือค่า ความเป็นสมาชิกของเอาต์พุตฟัซซี่ในแต่ละสับเซต



รูปที่ 3.22 กราฟของเอาต์พุตที่ได้

จากสมการที่ (3.2) จะสามารถคำนวณค่าเอาต์พุตด้วยวิธีหา Weighted Average ได้ดังนี้

$$X = \frac{(OutputNM \times NMCenter)+(OutputNS \times NSCenter)+(OutputZ \times ZCenter)}{OutputNM+OutputNS+OutputZ}$$

$$X = \frac{(0.4 \times -25) + (0.6324 \times -12.5) + (0.2 \times 0)}{0.4 + 0.6324 + 0.2}$$

$$X = \frac{(-10) + (-7.905) + (0)}{1.2324} = \frac{(-17.905)}{1.2324} = -14.528$$

หลังจากที่ระบบควบคุมแบบฟัซซี่คำนวณเสร็จแล้วจะส่งค่าที่ได้ให้กับไมโครคอนโทรลเลอร์เพื่อปรับค่ากำลังไฟฟ้าให้เพิ่มขึ้นหรือลดลงซึ่งจากในตัวอย่างที่ 2 ค่ากำลังไฟฟ้าที่ได้คือ - 14.528 % นั่นคือไมโครคอนโทรลเลอร์จะทำการปรับค่ากำลังไฟฟ้าให้ลดลงจากเดิม 14.528% นั่นเอง

3.8 หลักการเขียนโปรแกรม

3.8.1 การสร้างความสัมพันธ์ระหว่างอินพุตฟัซซี่

ในตัวอย่างที่ยกมานี้จะเป็นการสร้างความสัมพันธ์ระหว่างอินพุต Error และอินพุต Change of Error ซึ่งจะยกตัวอย่างมาแค่ 1 สับเซต นั่นคือ สับเซต ของ Zero ซึ่งสามารถเขียนโปรแกรมการคำนวณหาค่าความเป็นสมาชิกของแต่ละ สับเซต ออกมาได้ดังนี้

```
void DegreeofMembership(double UB, double C, double LB, int Ch)
{       // สร้างฟังก์ชัน DegreeofMembership เพื่อใช้ในการคำนวณหาค่าความเป็นสมาชิก
        ของแต่ละสับเซต
        double UpperBoundary, Center, LowerBoundary
        int Select;
        UpperBoundary = UB;          // กำหนดตัวแปลรับค่าสูงสุดของ Error
                                     // หรือ Change of Error ในเซตนั้นๆ
        Center = C;                  // กำหนดตัวแปลรับค่ากึ่งกลางของ Error
                                     // หรือ Change of Error ในเซตนั้นๆ
        LowerBoundary = LB;          // กำหนดตัวแปลรับค่าต่ำสุดของ Error
                                     // หรือ Change of Error ในเซตนั้นๆ
```

```
Select = Ch;                    // เป็นการเลือกว่าจะทำงานในส่วนของ Error หรือ
                                Change of Error
                                ถ้า Ch = 1 จะทำงานในส่วนของ Error
                                ถ้า Ch = 2 จะทำงานในส่วนของ Change of Error
switch(Select)
{       case 1 :        // คำนวณหาค่าความเป็นสมาชิกของ Error ในเซตนั้นๆ
        // ***** Error > Center ***** //
        if (Error > Center) {
        OutputError =   (((1/(Center - UpperBoundary))* Error)
                        - (UpperBoundary/(Center - UpperBoundary)));
        }
        // ***** Error <= Center ***** //
        if(Error <= Center) {
        OutputError =   (((1/(Center - LowerBoundary)) * Error)
                        - (LowerBoundary/(Center - LowerBoundary)));
        }
        case 2 : // คำนวณหาค่าความเป็นสมาชิกของ Change of Error ในเซตนั้นๆ
        // ***** Change Of Error > Center ***** //
        if(ChangeError > Center) {
        OutputChangeError =     (((1/(Center - UpperBoundary)) * ChangeError)
                                - (UpperBoundary/(Center - UpperBoundary)));
        }
        // ***** Change Of Error <= Center ***** //
        if(ChangeError <= Center) {
        OutputChangeError =     (((1/(Center - LowerBoundary)) * ChangeError)
                                - (LowerBoundary/(Center - LowerBoundary)));
        }
}
}
```

เมื่อสร้างฟังก์ชันการหาค่าความเป็นสมาชิกให้กับระบบเสร็จแล้ว ต่อไปจะเป็นการสร้างสับเซต Zero ของ Error ซึ่งค่า Error สูงสุด (ขอบบน) อยู่ที่ 25 ค่า Error กึ่งกลาง (ขอบกลาง) อยู่ที่ 0 และค่า Error ต่ำสุด (ขอบล่าง) อยู่ที่ -25 ดังรูปที่ 3.17 และทำการคำนวณหาค่าความเป็นสมาชิกของแต่ละสับเซต โดยใช้ฟังก์ชัน DegreeofMembership ได้ทันที ซึ่งสามารถเขียนโปรแกรมออกมาได้ดังนี้

```
void Error_Zero()
  {
     UB = 25;                  // กำหนดค่า Error สูงสุดของ Set Zero คือ 25
     C = 0;                    // กำหนดค่า Error กึ่งกลางของ Set Zero คือ 0
     LB = -25;                 // กำหนดค่า Error ต่ำสุดของ Set Zero คือ -25
     Ch = 1;                   // ในฟังก์ชัน DegreeofMembership ได้กำหนดไว้ว่า
                               // ถ้า Ch = 1 คือ การคำนวณค่าความเป็นสมาชิกของ Error
DegreeofMembership(UB,C,LB,Ch); // ทำการเรียกใช้ฟังก์ชัน DegreeofMembership
     if (Error <= LB)          // ถ้าค่า Error มีค่าน้อยกว่า -25
     {
       OutputError = 0;        // จะทำให้ค่าความเป็นสมาชิกของ Set Zero มีค่า = 0 ทันที
     }                         // เนื่องจากค่า Error ที่รับมาไม่ได้อยู่ใน Set ของ Zero
     else if (Error >= UB)     // และถ้าค่า Error มีค่ามากกว่า 25
     {
       OutputError = 0;        // จะทำให้ค่าความเป็นสมาชิกของ Set Zero มีค่า = 0 ทันที
     }                         // เนื่องจากค่า Error ที่รับมาไม่ได้อยู่ใน Set ของ Zero
     rE_NL = OutputError;      // ทำการรับค่าความเป็นสมาชิกของ Error ใน Set Zero
                               // เพื่อนำไปคำนวณต่อในขั้นตอนอิน-เฟอะเร็นซ์
  }
```

ต่อไปจะเป็นการสร้าง สับเซต Zero ของ Change of Error ซึ่งตามรูปที่ 3.18 จะมีค่า Change of Error สูงสุดอยู่ที่ 25 ค่ากึ่งกลางอยู่ที่ 0 และค่า Change of Error ต่ำสุดอยู่ที่ -25 เช่นกัน ซึ่งสามารถเขียนโปรแกรมออกมาได้ดังนี้

```
void ChangeofError_Zero()
{
    UB = 25;          // กำหนดค่า ChangeofError สูงสุดของ Set Zero คือ 25
    C = 0;            // กำหนดค่า ChangeofError กึ่งกลางของ Set Zero คือ 0
    LB = -25;         // กำหนดค่า ChangeofError ต่ำสุดของ Set Zero คือ -25
    Ch = 2;           // ในฟังก์ชัน DegreeofMembership ได้กำหนดไว้ว่า
                      // ถ้า Ch = / คือ การคำนวณค่าความเป็นสมาชิกของ ChangeofError
    DegreeofMembership(UB,C,LB,Ch); // ทำการเรียกใช้ฟังก์ชัน DegreeofMembership
    if (ChangeofError <= LB)        // ถ้าค่า ChangeofError มีค่าน้อยกว่า -25
    {
     Output ChangeofError = 0;      // จะทำให้ค่าความเป็นสมาชิกของ Set Zero
    }                               // มีค่า = 0 ทันทีเนื่องจากค่า ChangeofError ที่รับมา
                                    // ไม่ได้อยู่ใน Set ของ Zero
    else if (ChangeofError >= UB)   // และถ้าค่า ChangeofError มีค่ามากกว่า 25
    {
     OutputError = 0;               // จะทำให้ค่าความเป็นสมาชิกของ Set Zero มีค่า = 0 ทันที
    }                               // เนื่องจากค่า ChangeofError ที่รับมา
                                    // ไม่ได้อยู่ใน Set ของ Zero
    rE_NL = OutputChangeofError;    // ทำการรับค่าความเป็นสมาชิกของ
                                    //ChangeofError ใน Set Zero
                                    // เพื่อนำไปคำนวณต่อในขั้นตอนอิน-เฟอะเร็นซ์
}
```

จากฟังก์ชัน DegreeofMembership จะแสดงให้เห็นการคำนวณค่าความเป็นสมาชิกของแต่ละสับเซตของ Error และ Change of Error ในโปรแกรมจะเห็นว่ามีการคำนวณอยู่ 2 แบบคือถ้า Error หรือ Change of Error มีค่ามากกว่า Center (ค่ากึ่งกลาง [ขอบกลาง] ของแต่ละ สับเซต) จะนำ UpperBoundary (ค่าที่มากที่สุด [ขอบบน] ของแต่ละสับเซต) มาใช้ในการคำนวณหาค่าความเป็นสมาชิกของแต่ละ สับเซต แต่ถ้า Error หรือ Change of Error น้อยกว่า Center จะนำ Lower Boundary (ค่าที่น้อยที่สุด [ขอบล่าง] ของแต่ละ สับเซต) มาใช้ในการคำนวณหาค่าความเป็นสมาชิกของแต่ละสับเซตแทน

เมื่อได้ค่าความเป็นสมาชิกของแต่ละสับเซตของอินพุตเรียบร้อยแล้ว ต่อไปจะเป็นการคำนวณหาค่าความเป็นสมาชิกของแต่ละสับเซตของเอาต์พุต

3.8.2 การหาค่าความเป็นสมาชิกของเอาต์พุต

ต่อไปจะเป็นการคำนวณการหาค่าความเป็นสมาชิกของเอาต์พุตด้วยวิธี Root-Sum-Square (RSS) นั่นคือการเลือกค่าที่น้อยที่สุดของแต่ละ สับเซต ของ Input ตามกฎของตารางที่ 3.1 มายกกำลัง 2 แล้วทำการบวกกันแล้วถอดรากโดยคำนวณจาก $X_i = \sqrt{\sum_{i=1}^{n}(\min(E_i, CE_i))^2}$ โดย X_i คือ ค่าความเป็นสมาชิกของเอาต์พุตในที่นี้จะยกตัวอย่างในส่วนของ Error ในส่วนนี้จะยกมาเฉพาะ Positive Small (PS)

```
void PositiveSmallError()       // Positive Small Error
{
    For (Z = 0; Z <= 7; Z++) // Z คือค่า n + 1 ที่ +1 เพราะ n ตัวสุดท้ายคือการถอดราก
    {
        if (Z == 1)         // n คือจำนวนสมาชิกของกฎที่เกิดขึ้น
        {
            Choice Minimum (MfE_NM, MfCE_NL); // PS1
            rulePSM1 = Output Minimum;
            rulePSM1 = rulePSM1 * rulePSM1;
            ZrulePSM1 = ZrulePSM1 + rulePSM1;              }
        else if(Z == 2)
        {
            Choice Minimum (MfE_NS, MfCE_NM); // PS2
            rulePSM1 = Output Minimum;
            rulePSM1 = rulePSM1 * rulePSM1;
            ZrulePSM1 = ZrulePSM1 + rulePSM1;              }
```

```
else if(Z == 3)
{
        Choice Minimum (MfE_Z, MfCE_NS); // PS3
        rulePSM1 = Output Minimum;
        rulePSM1 = rulePSM1 * rulePSM1;
        ZrulePSM1 = ZrulePSM1 + rulePSM1;                }
else if(Z == 4)
{
        Choice Minimum (MfE_PS, MfCE_Z); // PS4
        rulePSM1 = Output Minimum;
        rulePSM1 = rulePSM1 * rulePSM1;
        ZrulePSM1 = ZrulePSM1 + rulePSM1;                }
else if(Z == 5)
{
        Choice Minimum (MfE_PM, MfCE_PS); // PS5
        rulePSM1 = Output Minimum;
        rulePSM1 = rulePSM1 * rulePSM1;
        ZrulePSM1 = ZrulePSM1 + rulePSM1;                }
else if(Z == 6)
{
        Choice Minimum (MfE_PL, MfCE_PM); // PS6
        rulePSM1 = Output Minimum;
        rulePSM1 = rulePSM1 * rulePSM1;
        ZrulePSM1 = ZrulePSM1 + rulePSM1;                }
else if(Z == 7)
{
        OutputPSM1 = sqrt (ZrulePSM1); // [Sigma of PositiveSmall^2]
        rulePSM1 = 0;
        ZrulePSM1 = 0;                                   }
```

```
        }
        Z = 0;
    }
```

จากโปรแกรมข้างต้นจะเห็นว่ากฎที่จะทำให้เกิดสมาชิกของ Positive Small ได้มีทั้งหมด 6 กฎดังตารางที่ 3.1 ดังนั้นค่า n จากสมการ $X_i = \sqrt{\sum_{i=1}^{n}(\min(E_i, CE_i))^2}$ จะเท่ากับ 6 ดังนั้นจะมีค่า PS ที่จะทำการบวกกันทั้งหมด 6 ตัว เมื่อได้ค่า PS ที่บวกกันครบทั้ง 6 ตัวแล้วขั้นตอนสุดท้ายคือการถอดรากที่ 2

3.8.4 ส่วน **Defuzzification**

เมื่อได้ค่าความเป็นสมาชิกของแต่ละสับเซตของเอาต์พุตแล้ว ต่อไปจะเป็นการคำนวณหาค่าเอาต์พุตของฟัซซี่ด้วยวิธี Weighted Average โดยนำค่าความเป็นสมาชิกมาคูณกับค่าความเป็นสมาชิกของเอาต์พุตฟัซซี่ของแต่ละสับเซต แล้วนำมารวมกันจากนั้นก็หารด้วยผลรวมของค่าความเป็นสมาชิกทั้งหมดซึ่งจะได้ค่าเอาต์พุตที่เราต้องการออกมาเพื่อนำไปคำนวณใน ไมโครคอนโทรลเลอร์ เพื่อเพิ่มหรือลดค่ากำลังไฟฟ้าของแต่ละมอเตอร์ต่อไป

```
void CalVoltChange()
{
double NLCenter, NMCenter, NSCenter, ZCenter, PSCenter, PMCenter, PLCenter;
        NLCenter = -37.5;
        NMCenter = -25;
        NSCenter = -12.5;
        ZCenter = 0;
        PSCenter = 12.5;
        PMCenter = 25;
        PLCenter = 37.5;
```

```
VoltChangeM1 = ceil (((OutputNLM1 * NLCenter) + (OutputNMM1 * NMCenter) +
                    (OutputNSM1 * NSCenter) + (OutputZM1 * ZCenter) +
                    (OutputPSM1 * PSCenter) + (OutputPMM1 * PMCenter) +
                    (OutputPLM1 * PLCenter)) / (OutputNLM1 + OutputNMM1 +
                    OutputNSM1 + OutputZM1 + OutputPSM1 + OutputPMM1 +
                    OutputPLM1));
}
```

โปรแกรมข้างต้นเป็นการคำนวณหาเอาต์พุตของฟัซซี่ ด้วยวิธี Weighted avcrage ในที่นี้จะยกตัวอย่างเฉพาะมอเตอร์1 เมื่อเราได้ค่าเอาต์พุตของฟัซซี่ แล้วก็จะนำค่าที่ได้มาคำนวณกลับไปที่ค่ากำลังไฟฟ้าของพีดับเบิลยูเอ็ม เพื่อจะทำการเพิ่มหรือลดตามที่ค่าเอาต์พุตได้คำนวณออกมา

```
void CalEncoderM1()
{
    DutyEn1 = TrueDuty1;
    DutyEn1 = DutyEn1 + VoltChangeM1;
    if (DutyEn1 > 99)
    {
        DutyEn1 = 100;
    }else if (DutyEn1 < 0)
    {
        DutyEn1 = 0;
    }
}
```

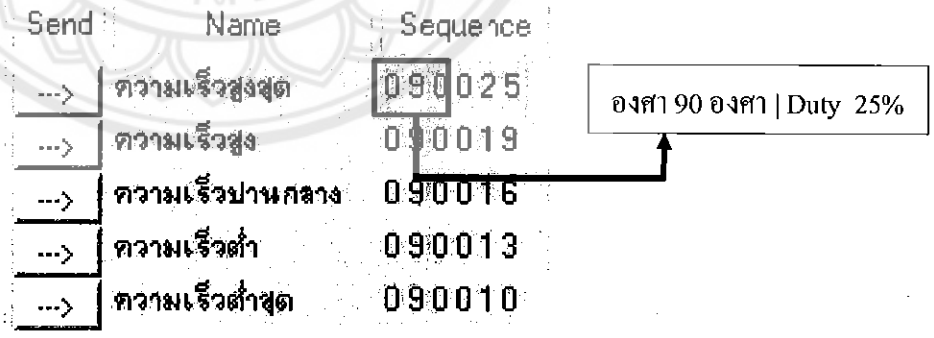
จากโปรแกรมข้างต้นจะยกตัวอย่างเฉพาะมอเตอร์1 จะเห็นในโปรแกรมมีการรับค่ากำลังไฟฟ้าเดิมของมอเตอร์1กลับมาเพื่อคำนวณว่าควรจะเพิ่มหรือจะลดตามค่าเอาต์พุตที่คำนวณได้

3.9 วิธีการใช้งานหุ่นยนต์

เมื่อทำการต่ออุปกรณ์ทั้งหมดเสร็จเรียบร้อยแล้วให้เชื่อมต่อไฟ 12 โวลต์ DC ให้บอร์ดแปลงไฟ 12 โวลต์ เป็น 5 โวลต์ของหุ่นยนต์และเชื่อมต่อไฟของบอร์ด dspic30F4011 เพื่อเปิดการทำงานของบอร์ด dsPIC30F4011 ดังรูปที่ 3.23

เปิดโปรแกรม Docklight แล้ว Setting ค่า ComPort Channel Baud Rate ดังรูปที่ 3.16 เมื่อทำการเซตค่าเสร็จแล้วให้ทำการป้อนอินพุต ซึ่งอินพุตในที่นี้คือ ค่าองศาการเคลื่อนที่ของหุ่นยนต์และค่ากำลังไฟฟ้าของหุ่นยนต์ โดยวิธีการเซตค่าอินพุตคือใส่เลขค่าตามตำแหน่ง 6 หลัก A B C D E F โดยตำแหน่ง A B C คือ ค่าองศาการเคลื่อนที่ของหุ่นยนต์ (ตั้งแต่ 0 – 360°) ส่วนตำแหน่ง D E F คือค่ากำลังไฟฟ้าซึ่งจะช้าหรือเร็วขึ้นอยู่กับเรากำหนด (ตั้งแต่ 15 – 100%) ดังรูปที่ 3.24

รูปที่ 3.23 วิธีเชื่อมต่อสายไฟ 12โวลต์ DC ต่อกับบอร์ดแปลงไฟ 12 โวลต์ เป็น 5 โวลต์



รูปที่ 3.24 การใส่ค่าองศาการเคลื่อนที่และค่ากำลังไฟฟ้าของหุ่นยนต์ในโปรแกรม Docklight

ระบบจะทำการรับค่าความเร็วรอบของมอเตอร์เข้ามาไว้ในตัวไมโครคอนโทรลเลอร์ dsPIC30F4011 และทำการคำนวณค่า Error ที่เกิดขึ้น จากนั้นจะเป็นการทำงานของระบบควบคุมแบบฟัซซี่

```
<CR>
 ****************** Test = 3 ****************** <CR>
Count1 = 0 Count2 = 50 Count3 = 69 Count4 = 71 <CR>
Count1 = 0 Count2 = 85 Count3 = 98 Count4 = 97 <CR>
Count1 = 0 Count2 = 112 Count3 = 127 Count4 = 124 <CR>
Count1 = 0 Count2 = 142 Count3 = 159 Count4 = 152 <CR>
Count1 = 0 Count2 = 172 Count3 = 190 Count4 = 181 <CR>
Count1 = 0 Count2 = 201 Count3 = 221 Count4 = 210 <CR>
<CR>
 SetPoint1 = 0 SetPoint2 = 225 SetPoint3 = 238 SetPoint4 = 218 <CR>
<CR>
 Duty1 = 50 Duty2 = 50 Duty3 = 50 Duty4 = 50 <CR>
<CR>
 ****************** Test = 4 ****************** <CR>
Count1 = 0 Count2 = 59 Count3 = 70 Count4 = 73 <CR>
Count1 = 0 Count2 = 86 Count3 = 99 Count4 = 99 <CR>
Count1 = 0 Count2 = 113 Count3 = 128 Count4 = 127 <CR>
Count1 = 0 Count2 = 142 Count3 = 159 Count4 = 156 <CR>
Count1 = 0 Count2 = 172 Count3 = 190 Count4 = 184 <CR>
Count1 = 0 Count2 = 201 Count3 = 221 Count4 = 211 <CR>
<CR>
 EnCoder1 = 0 EnCoder2 = 225 EnCoder3 = 238 EnCoder4 = 219 <CR>
<CR>
 Duty1 = 50 Duty2 = 50 Duty3 = 50 Duty4 = 50 <CR>
<CR>
```

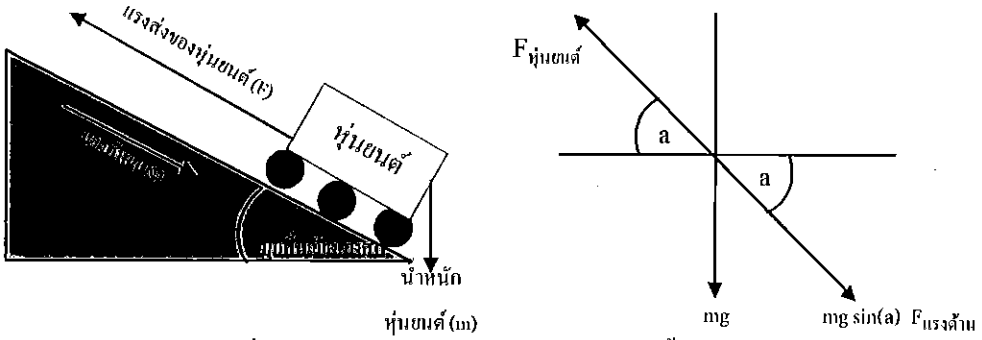
รูปที่ 3.25 ผลการทำงานของโปรแกรมในหน้าต่าง Docklight

- รอผลจากการทำงานของระบบ Fuzzy Logic และระบบจะนำค่าที่ได้มาเพิ่มหรือลดความเร็วของมอเตอร์

3.10 การทดลองระบบควบคุมแบบฟัซซี่

การทดลองในครั้งนี้ หุ่นยนต์ที่ใช้ทำการทดลองมีน้ำหนักอยู่ที่ 1.2 กิโลกรัม ซึ่งจะทำการทดลองระบบควบคุมแบบฟัซซี่โดยใช้พื้นเอียงทั้งหมด 3 ระดับ คือ พื้นเอียง 10 20 และ 30 องศา ตามลำดับ ดังรูป 3.26 ซึ่งค่าแรงต้านที่เกิดขึ้นจากพื้นเอียงในแต่ละระดับจะมีค่าแรงต้านที่แตกต่างกัน โดยสามารถดูค่าแรงต้านได้จากการคำนวณข้างล่าง และเพื่อทดสอบว่าระบบควบคุมแบบฟัซซี่นั้นสามารถควบคุมหุ่นยนต์ให้มีความเร็วตามที่ต้องการได้ แม่ว้าจะมีแรงต้านการเคลื่อนที่ที่ต่างกัน ซึ่งการทดลองนี้จะใช้ความเร็วของหุ่นยนต์ทั้งหมด 3 ระดับคือ ที่ความเร็วต่ำจะใช้ค่ากำลังไฟฟ้าที่ 25% ที่ความเร็วปานกลางจะใช้ค่ากำลังไฟฟ้าที่ 50% และ ที่ความเร็วสูงจะใช้ค่ากำลังไฟฟ้าที่ 75% ดังรูปที่ 3.24 และทิศทางการเคลื่อนที่ของหุ่นยนต์ที่ใช้คือให้หุ่นยนต์วิ่งขึ้นพื้นเอียงเป็นทางตรงทำมุม 90 องศา และวิ่งขึ้นพื้นเอียงเป็นทางเฉียงทำมุม 45 องศา เพื่อทดสอบการเปลี่ยนทิศทางของหุ่นยนต์ ส่วนพื้นที่ใช้ในการทดลองจะเป็นพื้นยางดังรูป 3.28

หมายเหตุ: เนื่องจากพื้นยางมีความฝืดของผิวดีกว่าพื้นไม้หรือพื้นพลาสติก ทำให้ล้อของหุ่นยนต์ไม่ลื่นและสามารถเก็บค่าการทดลองได้ดีกว่าพื้นอื่นๆ



รูปที่ 3.26 การแตกแรงหาค่าแรงต้านของพื้นเอียง

การหาแรงต้านทานของพื้นเอียงแต่ละมุมสามารถหาได้จากสมการ $F_{แรงต้าน} = m \times g \times \sin(a)$ โดย a คือ มุมของพื้นเอียงมีค่า = 10 20 และ 30 ตามลำดับ ; m คือ น้ำหนักของหุ่นยนต์มีค่าเท่ากับ 1.50 กิโลกรัม g = 9.83 m/s^2 ทำการแทนค่าตัวแปลทั้งหมดลงในสมการ

โดยให้ a = 10; $F_{แรงต้าน} = m \times g \times \sin(10)$; $\sin(10) = 0.17$

แทนค่า $F_{แรงต้าน} = 1.50 \times 9.83 \times 0.17$

$F_{แรงต้าน} = 2.56$ นิวตัน

แรงต้านที่พื้นเอียงทำมุม 20 องศาจะมีค่าเท่ากับ 2.56 นิวตัน

ให้ a = 20; $F_{แรงต้าน} = m \times g \times \sin(20)$; $\sin(20) = 0.34$

แทนค่า $F_{แรงต้าน} = 1.50 \times 9.83 \times 0.34$

$F_{แรงต้าน} = 5.04$ นิวตัน

แรงต้านที่พื้นเอียงทำมุม 20 องศาจะมีค่าเท่ากับ 5.04 นิวตัน

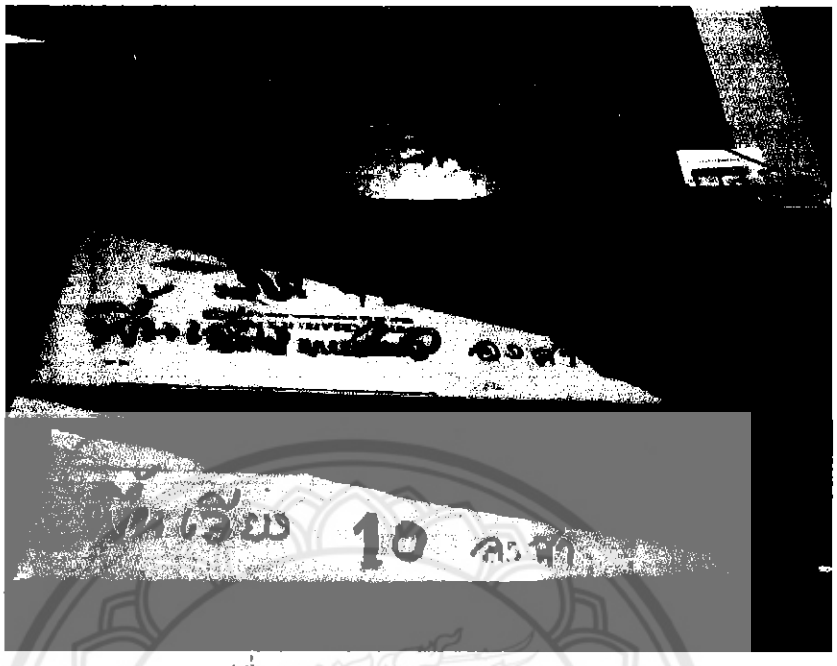
ให้ a = 30; $F_{แรงต้าน} = m \times g \times \sin(30)$; $\sin(30) = 0.50$

แทนค่า $F_{แรงต้าน} = 1.50 \times 9.83 \times 0.50$

$F_{แรงต้าน} = 7.37$ นิวตัน

แรงต้านที่พื้นเอียงทำมุม 30 องศาจะมีค่าเท่ากับ 7. 37 นิวตัน

จะเห็นได้ว่าทุกๆองศาที่เพิ่มขึ้นของพื้นเอียง จะมีค่าแรงต้านที่เพิ่มขึ้นด้วยเช่นกัน



รูปที่ 3.27 มุมองศา 10 20 และ 30 องศา

รูปที่ 3.28 พื้นเอียงที่เป็นพื้นยาง

บทที่ 4

ผลการออกแบบระบบการควบคุมโดยใช้ Fuzzy logic

4.1 ผลการดำเนินงาน

หลังจากที่ได้ทำโครงงานเกี่ยวกับการควบคุมความเร็วหุ่นยนต์อัตโนมัติและทำระบบควบคุมแบบฟัซซี่แล้วได้ผลการทดลองคือ หุ่นยนต์สามารถควบคุมความเร็วได้ตามที่ได้กำหนด Setpoint ไว้ หุ่นยนต์สามารถวิ่งได้ 8 ทิศทางและสามารถปรับความเร็วของหุ่นยนต์ได้ 3 ระดับ คือ

4.1.1 ช่วงการทดลองหุ่นยนต์วิ่งแบบมีสิ่งกีดขวาง(พื้นเอียง)

หุ่นยนต์ใช้ความเร็วดังนี้

1. ความเร็วต่ำที่ Duty Cycle 25% ได้ความเร็วรอบของมอเตอร์ 13 รอบ / การทำงานของโปรแกรม Fuzzy 1 ครั้ง (1 รอบการทำงานของโปรแกรมฟัซซี่ใช้เวลา = 0.067 Sec)
2. ความเร็วปานกลางที่ Duty Cycle 50% ได้ความเร็วรอบของมอเตอร์ 16 รอบ / การทำงานของโปรแกรม Fuzzy 1 ครั้ง (1 รอบการทำงานของโปรแกรมฟัซซี่ใช้เวลา = 0.067 Sec)
3. ความเร็วสูงที่ Duty Cycle 75% ได้ความเร็วรอบของมอเตอร์ 19 รอบ / การทำงานของโปรแกรม Fuzzy 1 ครั้ง (1 รอบการทำงานของโปรแกรมฟัซซี่ ใช้เวลา = 0.067 Sec)

4.1.2 ช่วงการทดลองหุ่นยนต์วิ่งแบบไม่มีสิ่งกีดขวาง

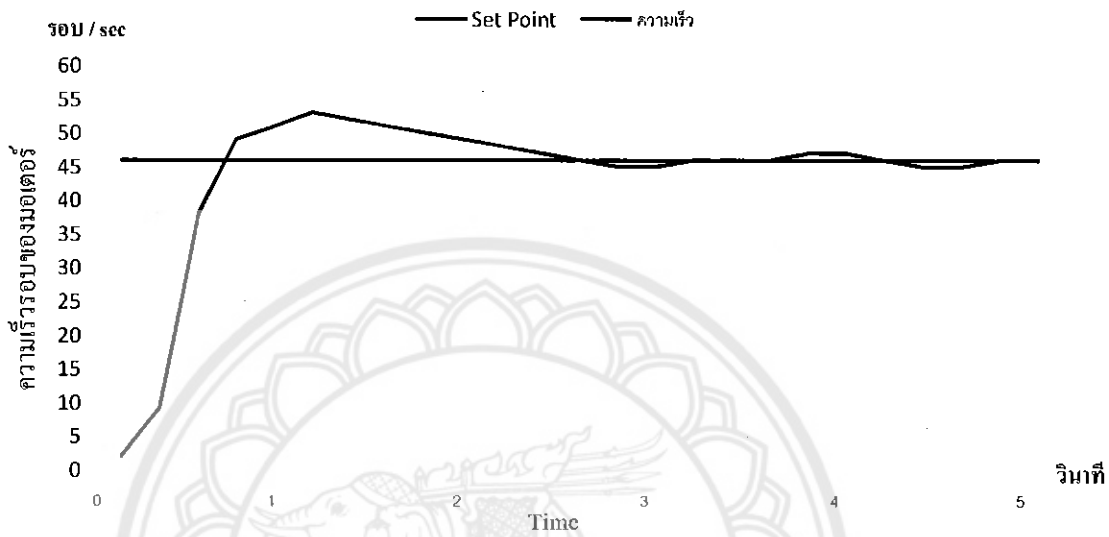
หุ่นยนต์ใช้ความเร็วดังนี้

1. ความเร็วต่ำที่ Duty Cycle 25% ได้ความเร็วรอบของมอเตอร์ 46 รอบ / การทำงานของโปรแกรม Fuzzy 1 ครั้ง (1 รอบการทำงานของโปรแกรมฟัซซี่ ใช้เวลา = 0.20 Sec)
2. ความเร็วปานกลางที่ Duty Cycle 50% ได้ความเร็วรอบของมอเตอร์ 50 รอบ / การทำงานของโปรแกรม Fuzzy 1 ครั้ง (1 รอบการทำงานของโปรแกรมฟัซซี่ ใช้เวลา = 0.20 Sec)
3. ความเร็วสูงที่ Duty Cycle 75% ได้ความเร็วรอบของมอเตอร์ 54 รอบ / การทำงานของโปรแกรม Fuzzy 1 ครั้ง (1 รอบการทำงานของโปรแกรมฟัซซี่ ใช้เวลา = 0.20 Sec)

หมายเหตุ: การทดลองหุ่นยนต์วิ่งแบบมีสิ่งกีดขวาง (พื้นเอียง) มอเตอร์ของหุ่นยนต์ยังทำงานปกติดีแต่การทดลองหุ่นยนต์วิ่งแบบไม่มีสิ่งกีดขวาง มอเตอร์เกิดความเสียหายภายในทำให้มอเตอร์ของหุ่นยนต์หมุนช้ากว่าปกติ ซึ่งทำให้ค่า Setpoint ของทั้ง 2 การทดลอง มีค่าไม่เท่ากัน

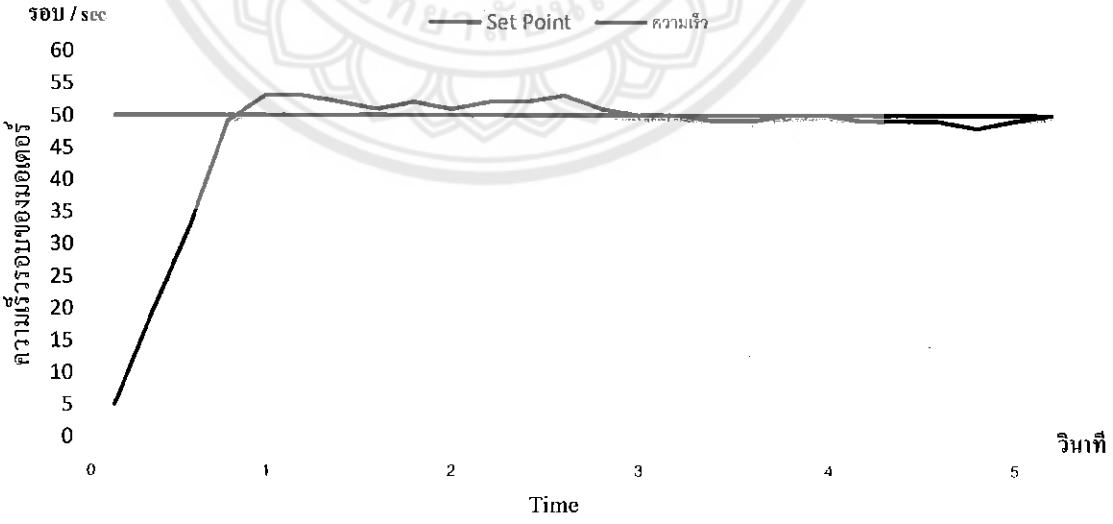
4.2 กราฟผลการทดลองหุ่นยนต์วิ่งแบบไม่มีสิ่งกีดขวาง (ใช้ระบบควบคุมแบบฟัซซี่)

4.2.1 หุ่นยนต์วิ่งด้วยความเร็วต่ำ



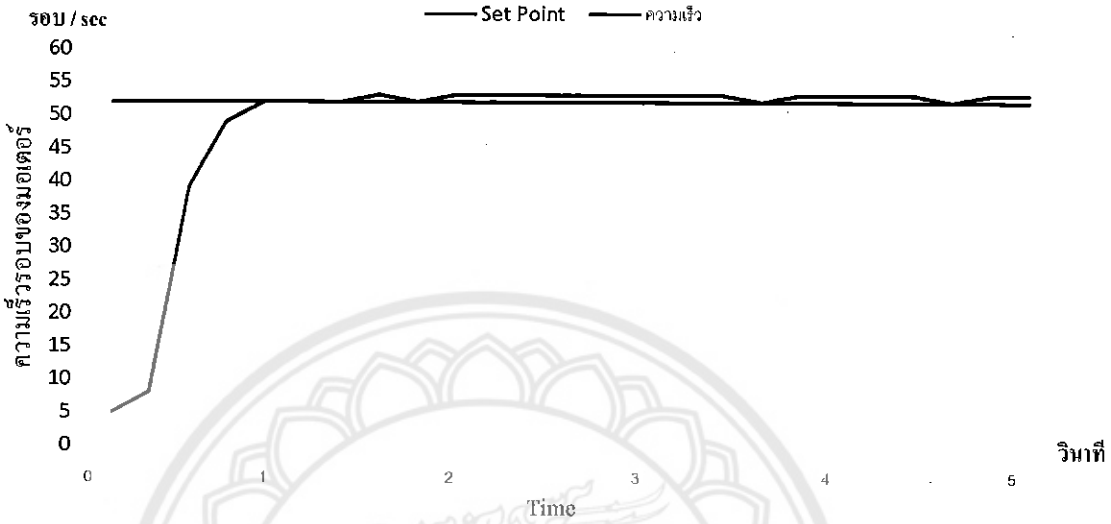
รูปที่ 4.1 กราฟผลการทดลองหุ่นยนต์วิ่งด้วยความเร็วต่ำ

4.2.2 หุ่นยนต์วิ่งด้วยความเร็วปานกลาง



รูปที่ 4.2 กราฟผลการทดลองหุ่นยนต์วิ่งด้วยความเร็วปานกลาง

4.2.3 หุ่นยนต์วิ่งด้วยความเร็วสูง

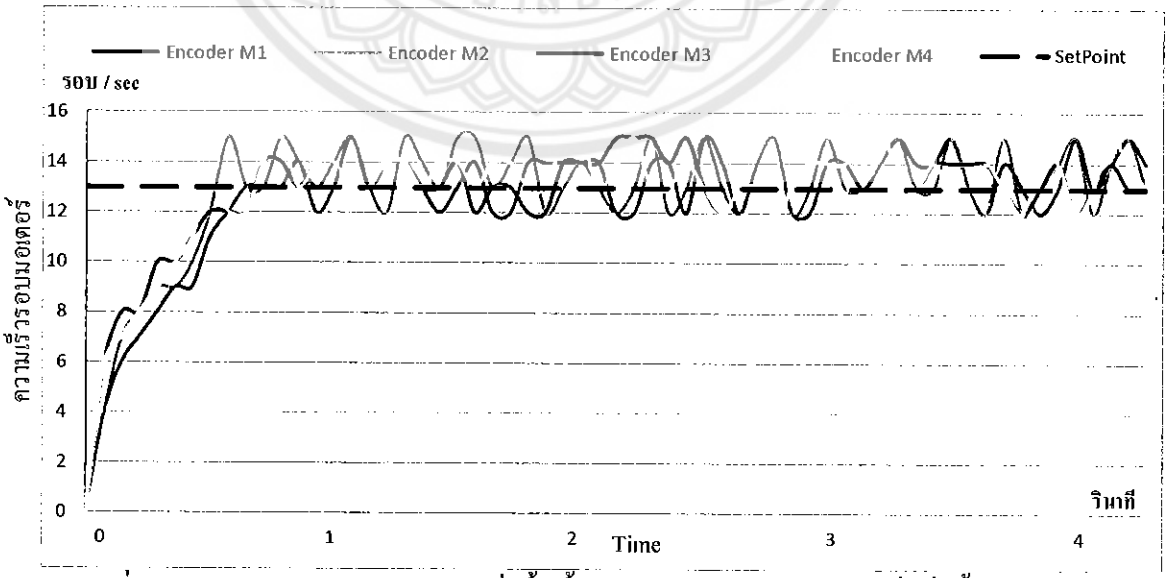


รูปที่ 4.3 กราฟผลการทดลองหุ่นยนต์วิ่งด้วยความเร็วสูง

หมายเหตุ: ค่า Setpoint ที่ได้จากการทดลอง 4.1 4.2 4.3 เป็นค่าการทดลองของปี 2559 (ค่าใหม่)

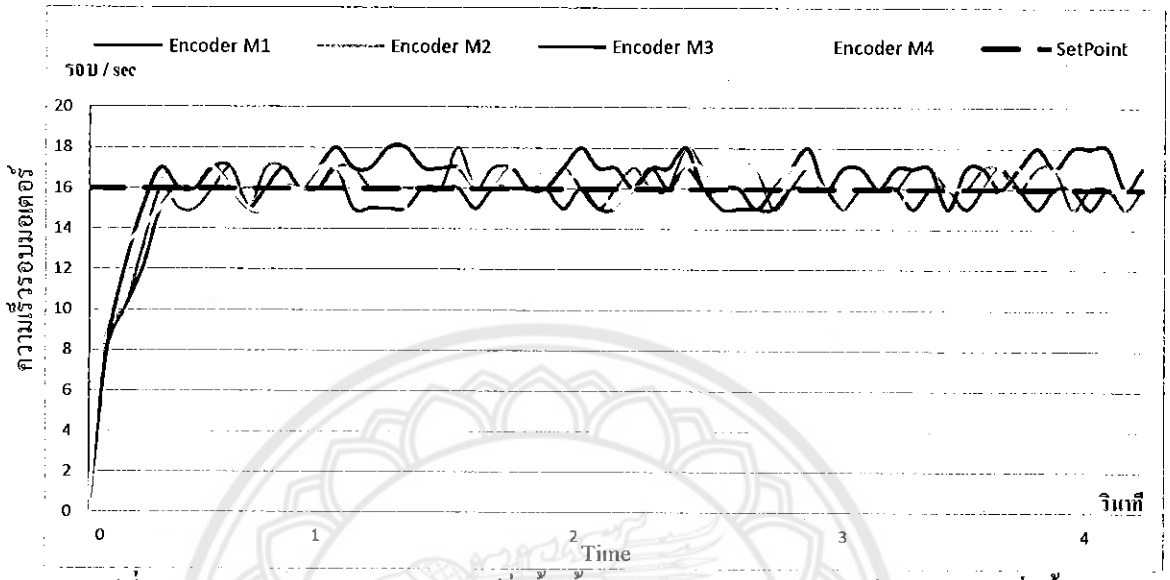
4.3 กราฟผลการทดลองของหุ่นยนต์วิ่งขึ้นพื้นเอียง

4.3.1 หุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ความเร็วต่ำ ที่ Duty Cycle 25%



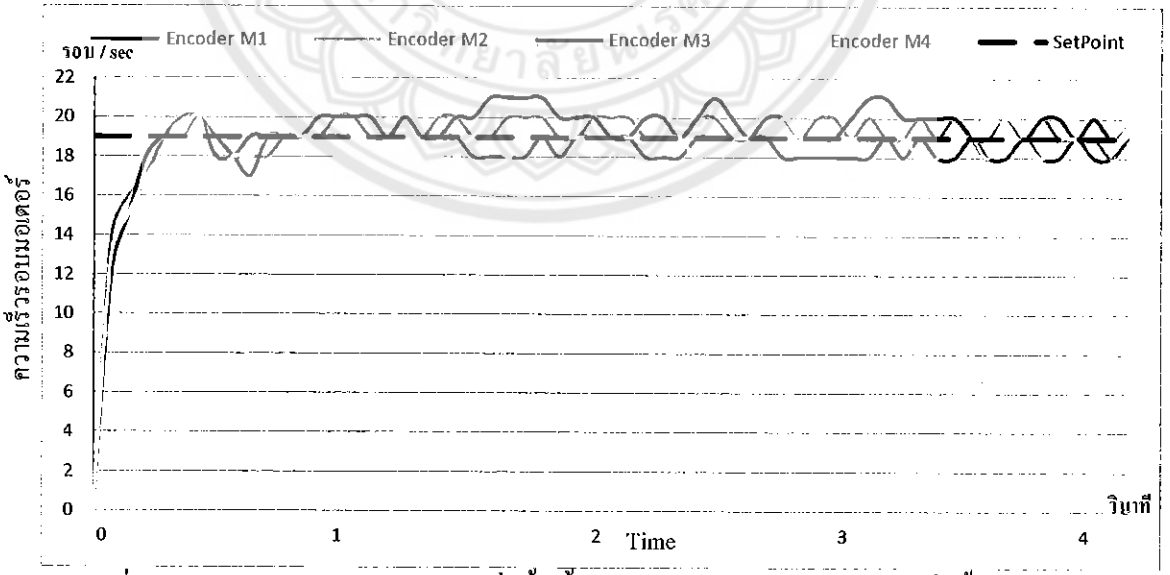
รูปที่ 4.4 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ด้วยความเร็วต่ำ วิ่งขึ้นเป็นเส้นตรง

4.3.2 หุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ความเร็วปานกลาง ที่ Duty Cycle 50%



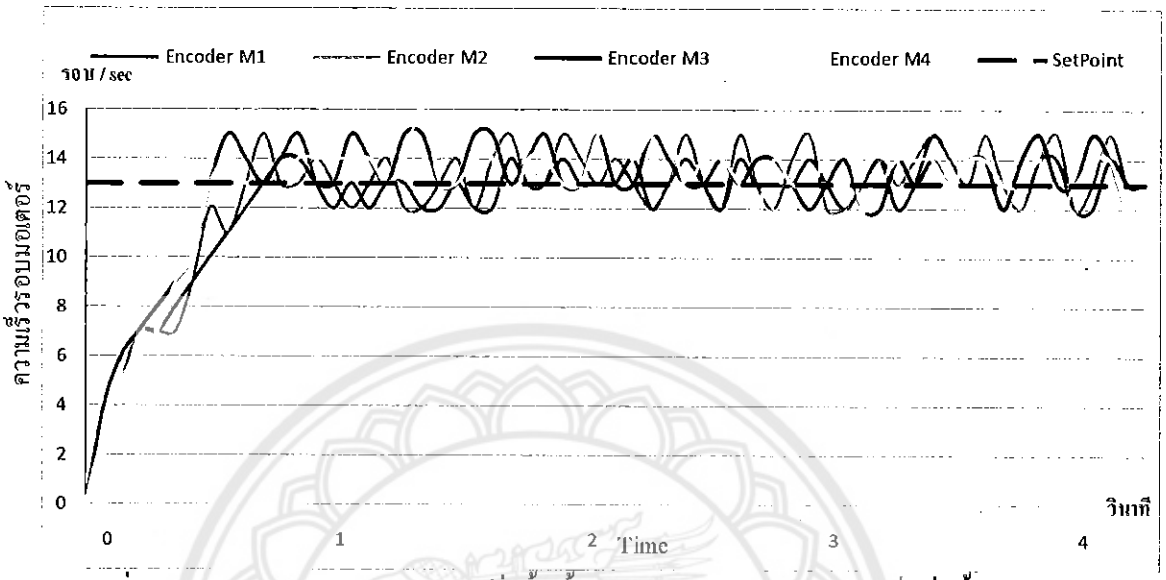
รูปที่ 4.5 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ด้วยความเร็วปานกลาง วิ่งขึ้นเป็นเส้นตรง

4.3.3 หุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ความเร็วสูง ที่ Duty Cycle 75%



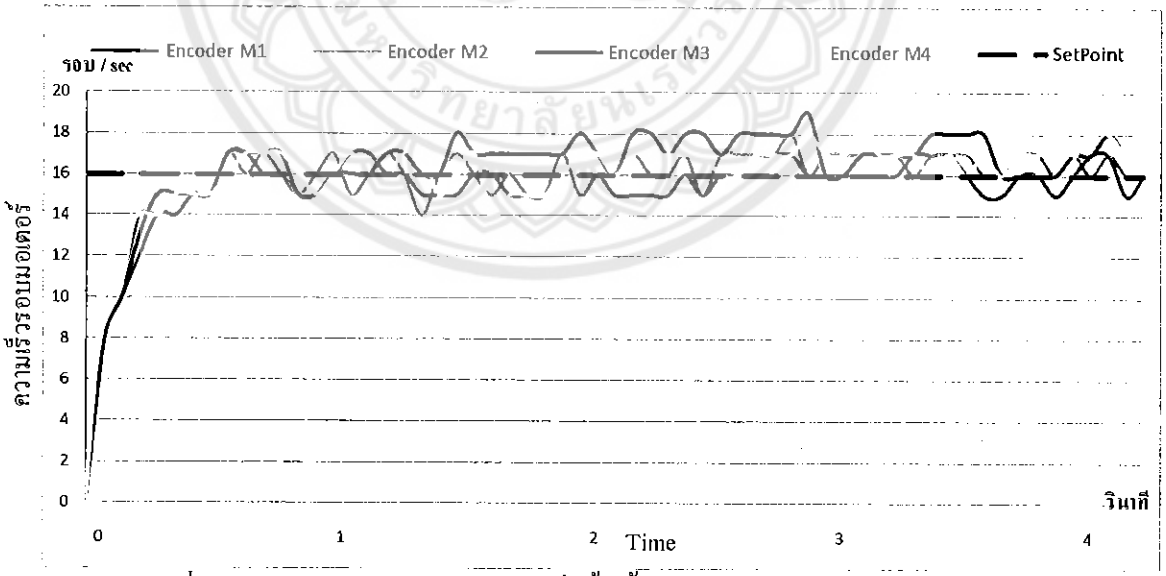
รูปที่ 4.6 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ด้วยความเร็วสูง วิ่งขึ้นเป็นเส้นตรง

4.3.4 หุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ความเร็วต่ำ ที่ Duty Cycle 25%



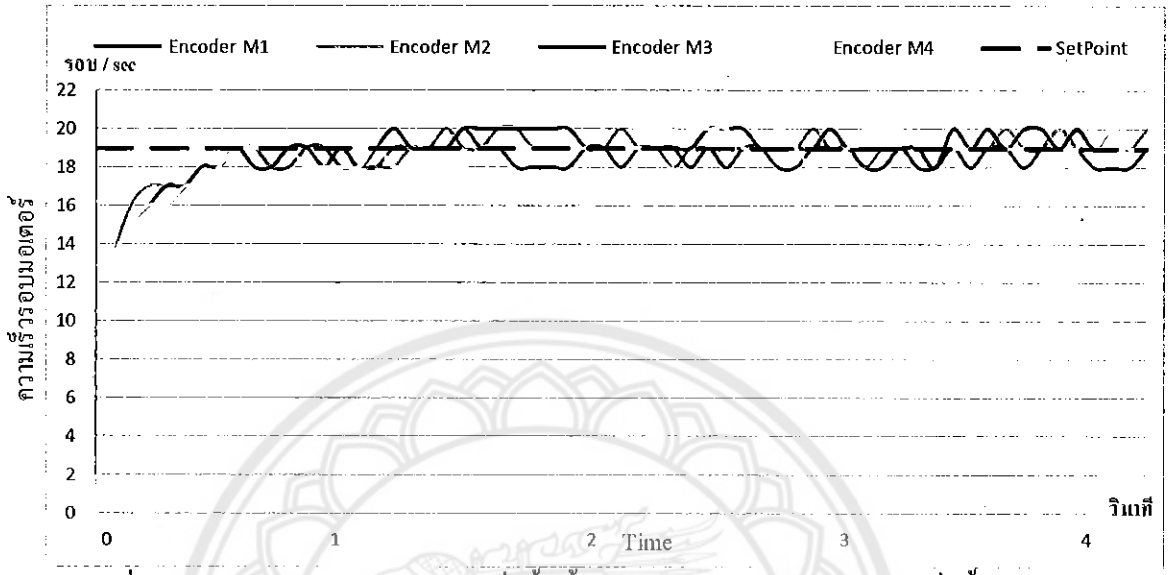
รูปที่ 4.7 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ด้วยความเร็วต่ำ วิ่งขึ้นเป็นเส้นตรง

4.3.5 หุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ความเร็วปานกลาง ที่ Duty Cycle 50%



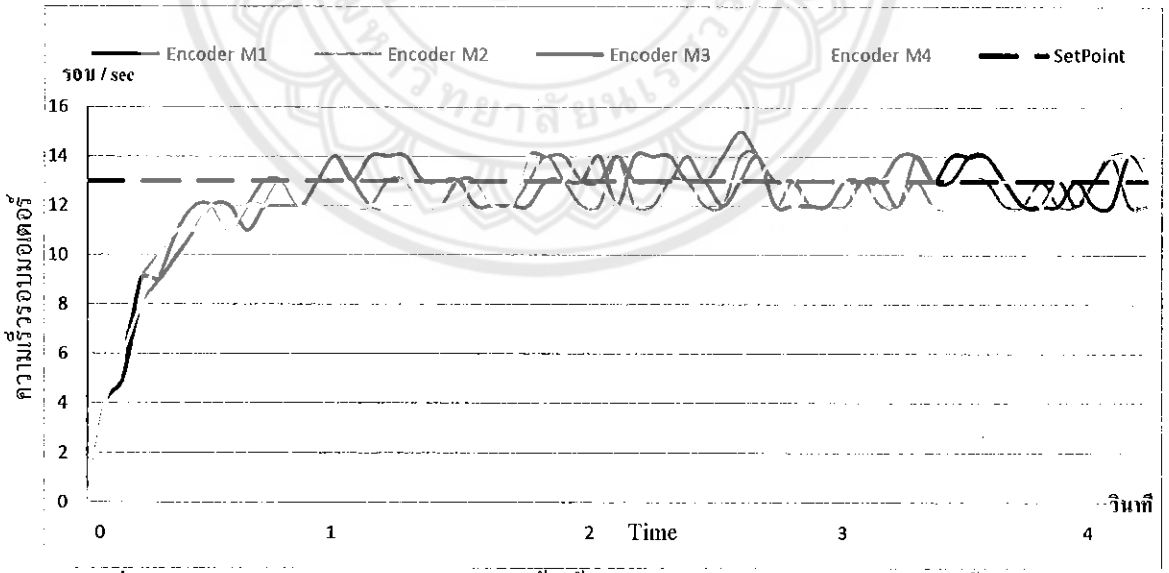
รูปที่ 4.8 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ด้วยความเร็วปานกลาง วิ่งขึ้นเป็นเส้นตรง

4.3.6 หุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ความเร็วสูง ที่ Duty Cycle 75%



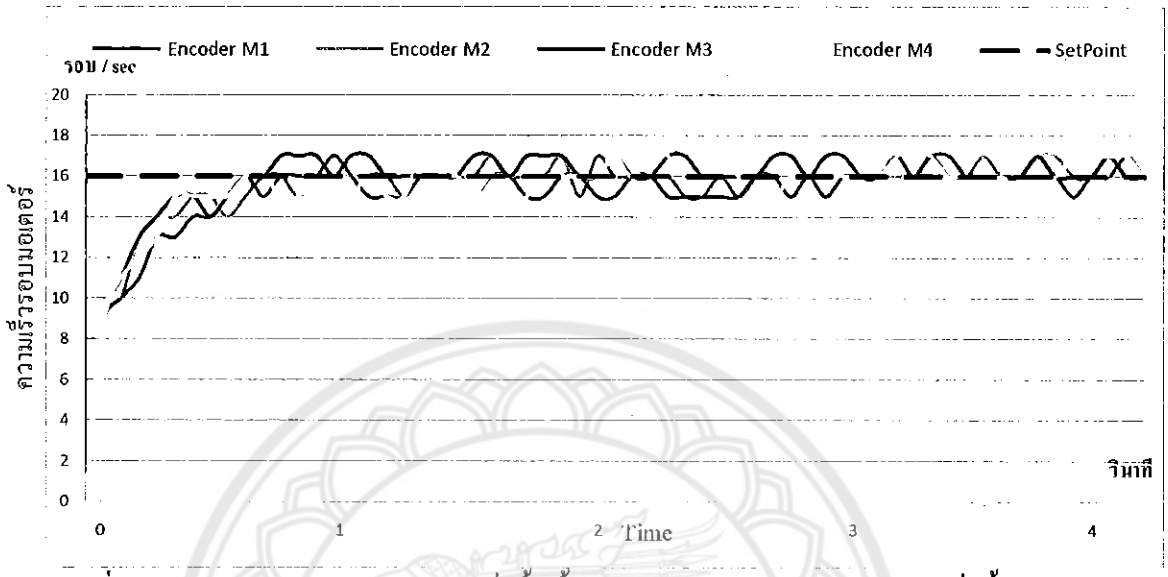
รูปที่ 4.9 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ด้วยความเร็วสูง วิ่งขึ้นเป็นเส้นตรง

4.3.7 หุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ความเร็วต่ำ ที่ Duty Cycle 25%



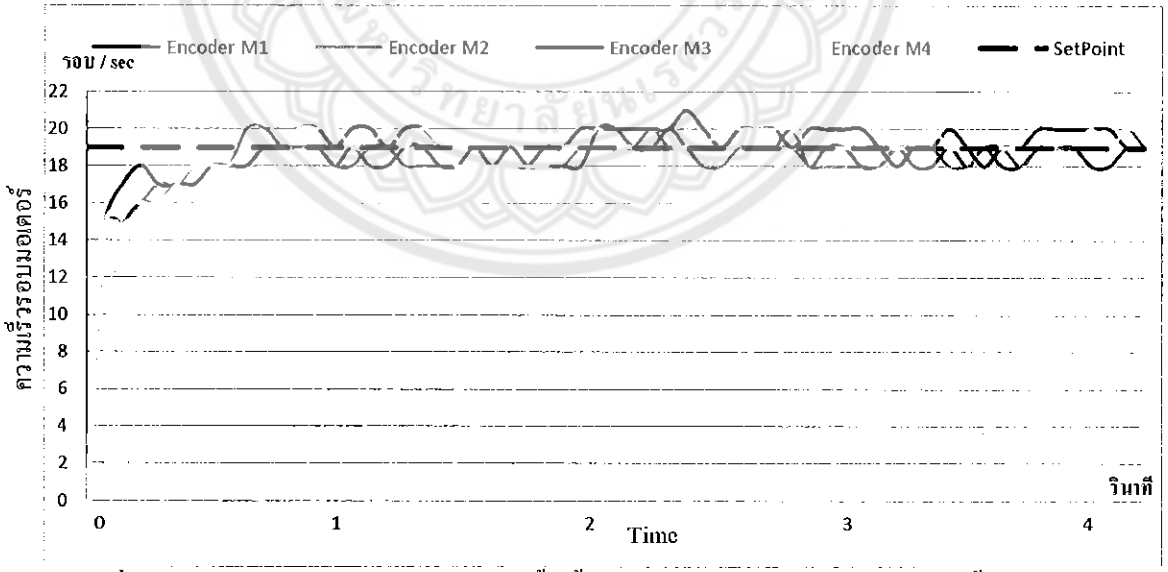
รูปที่ 4.10 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ด้วยความเร็วต่ำ วิ่งขึ้นเป็นเส้นตรง

4.3.8 หุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ความเร็วปานกลาง ที่ Duty Cycle 50%



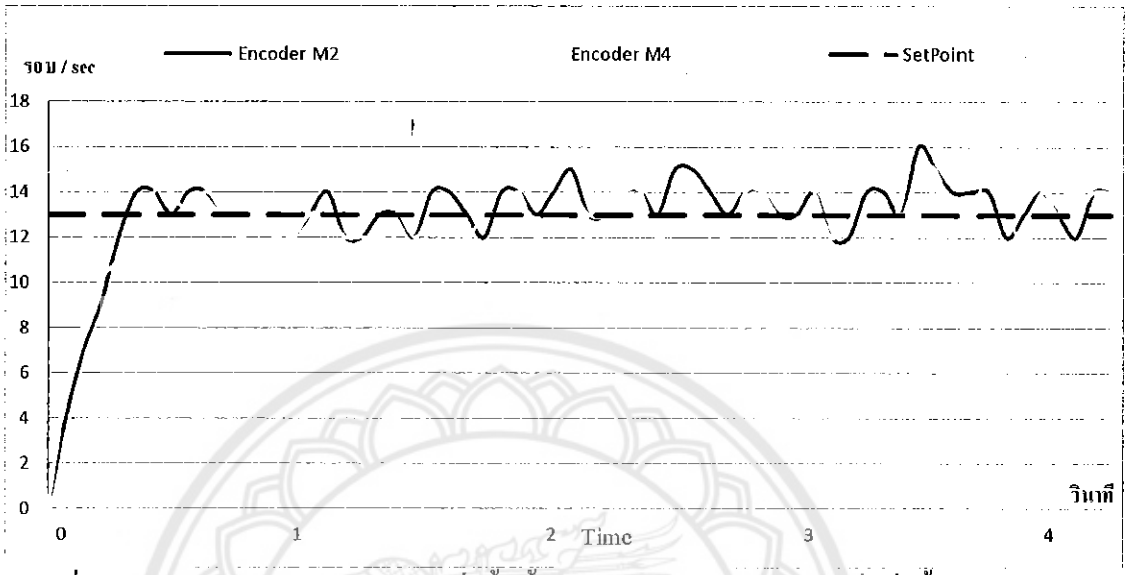
รูปที่ 4.11 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ความเร็วปานกลาง วิ่งขึ้นเป็นเส้นตรง

4.3.9 หุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ความเร็วสูง ที่ Duty Cycle 75%



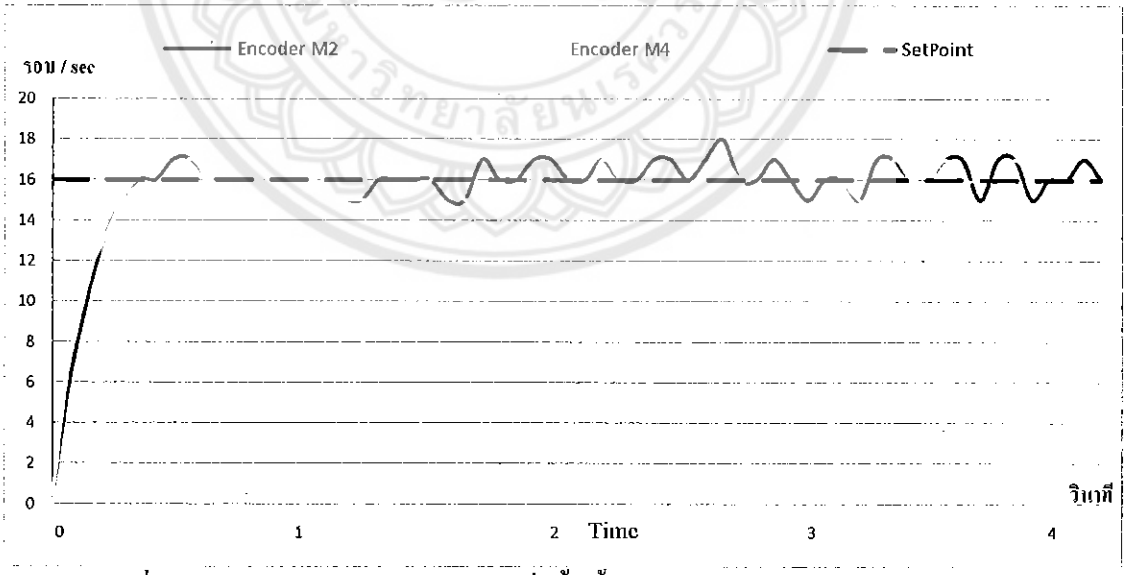
รูปที่ 4.12 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ด้วยความเร็วสูง วิ่งขึ้นเป็นเส้นตรง

4.3.10 หุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ความเร็วต่ำ ที่ Duty Cycle 25% วิ่งขึ้นเฉียง 45 องศา



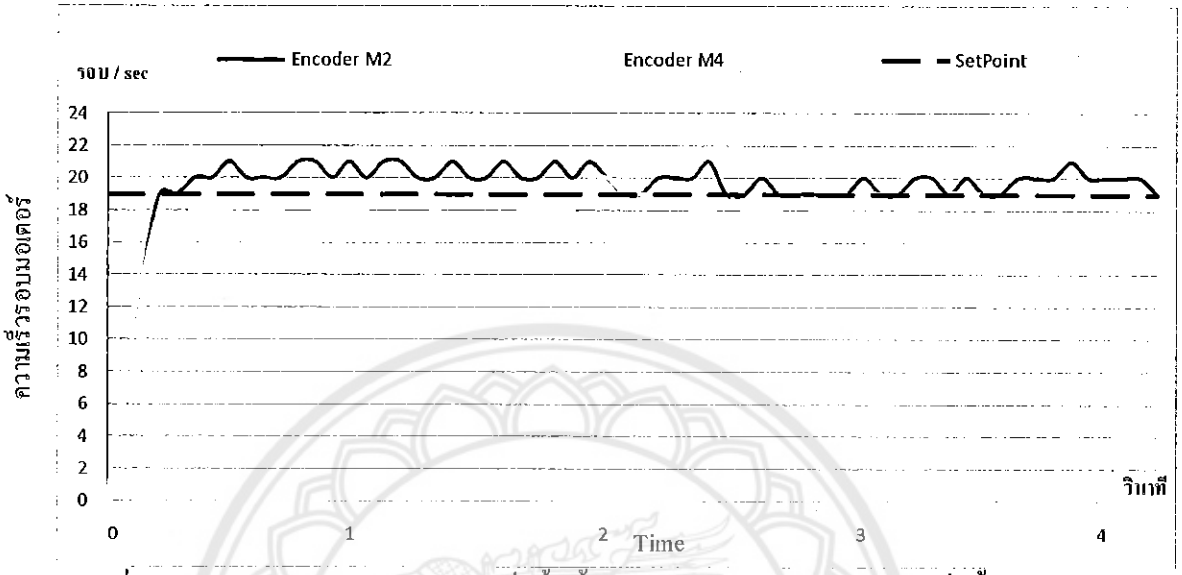
รูปที่ 4.13 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ด้วยความเร็วต่ำ วิ่งขึ้นเฉียง 45 องศา

4.3.11 หุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ความเร็วปานกลางที่ Duty Cycle 50% วิ่งขึ้นเฉียง 45 องศา



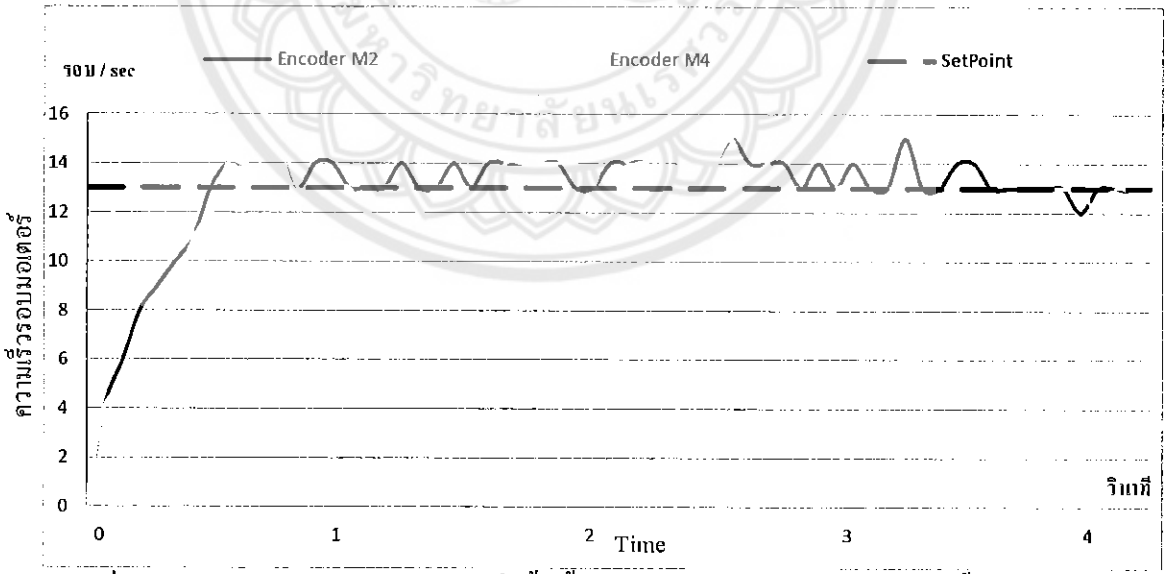
รูปที่ 4.14 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ด้วยความเร็วปานกลาง วิ่งขึ้นเฉียง 45 องศา

4.3.12 หุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ความเร็วสูง ที่ Duty Cycle 75% วิ่งขึ้นเฉียง 45 องศา



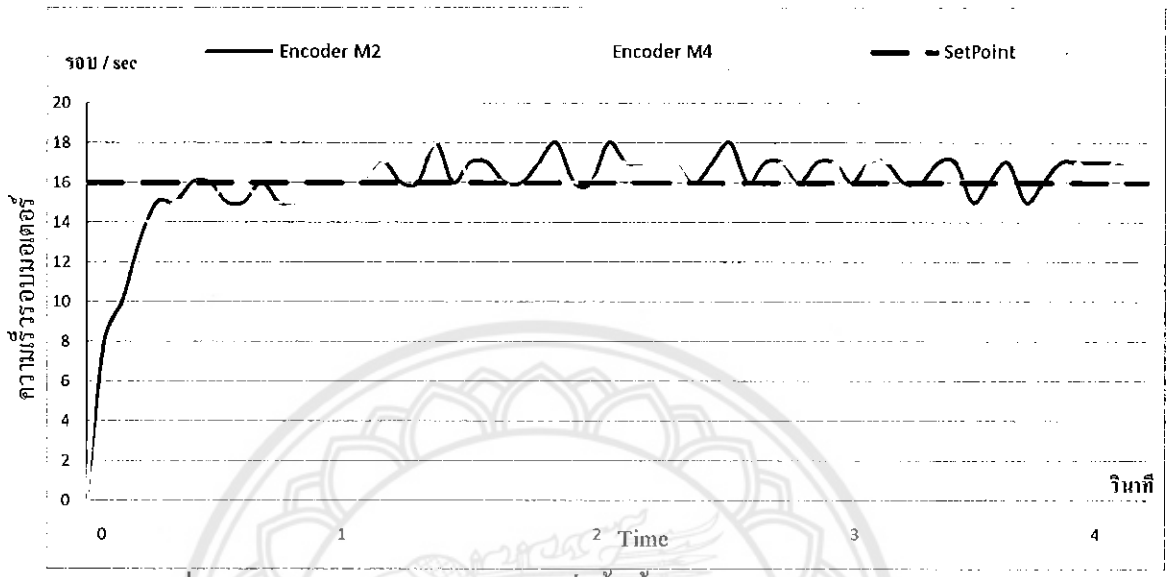
รูปที่ 4.15 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ด้วยความเร็วสูง วิ่งขึ้นเฉียง 45 องศา

4.3.13 หุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ความเร็วต่ำ ที่ Duty Cycle 25% วิ่งขึ้นเฉียง 45 องศา



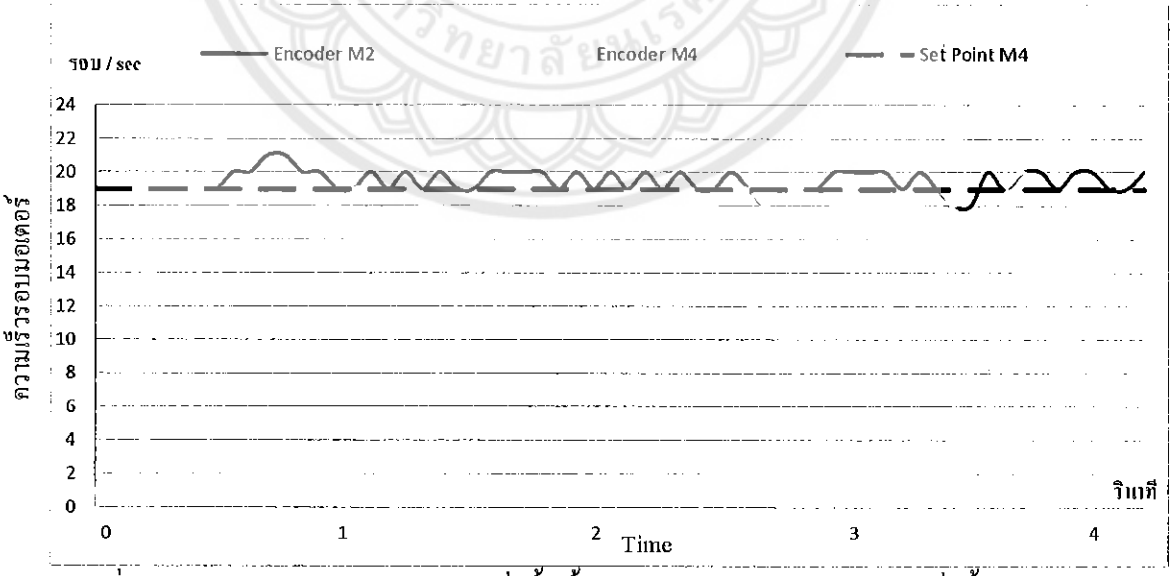
รูปที่ 4.16 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ด้วยความเร็วต่ำ วิ่งขึ้นเฉียง 45 องศา

4.3.14 หุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ความเร็วปานกลางที่ Duty Cycle 50% วิ่งขึ้นเฉียง 45 องศา



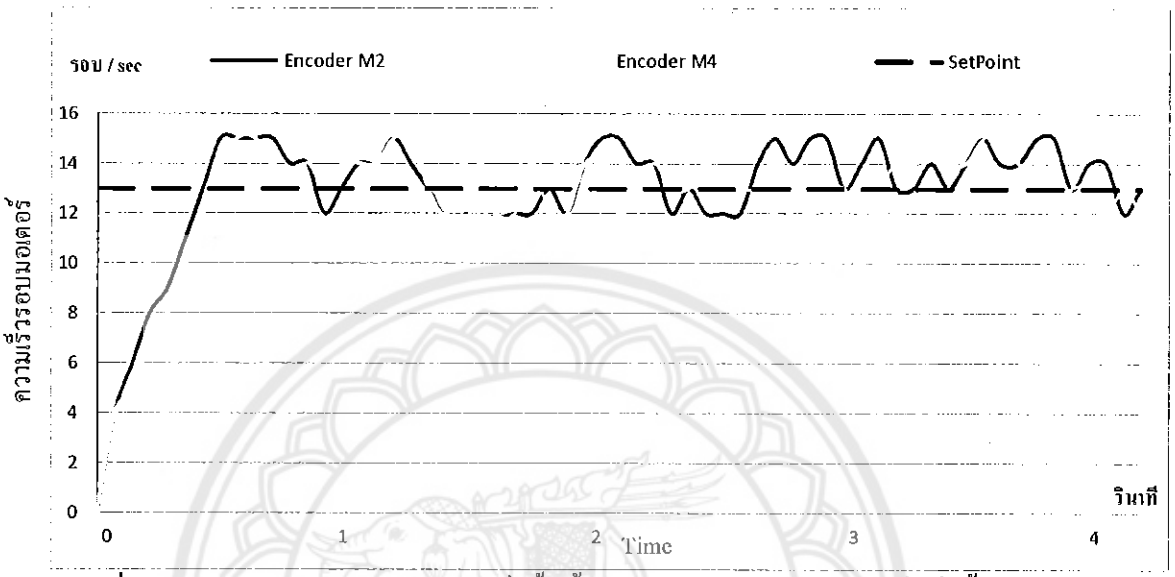
รูปที่ 4.17 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ด้วยความเร็วปานกลาง วิ่งขึ้นเฉียง 45 องศา

4.3.15 หุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ความเร็วสูง ที่ Duty Cycle 75% วิ่งขึ้นเฉียง 45 องศา



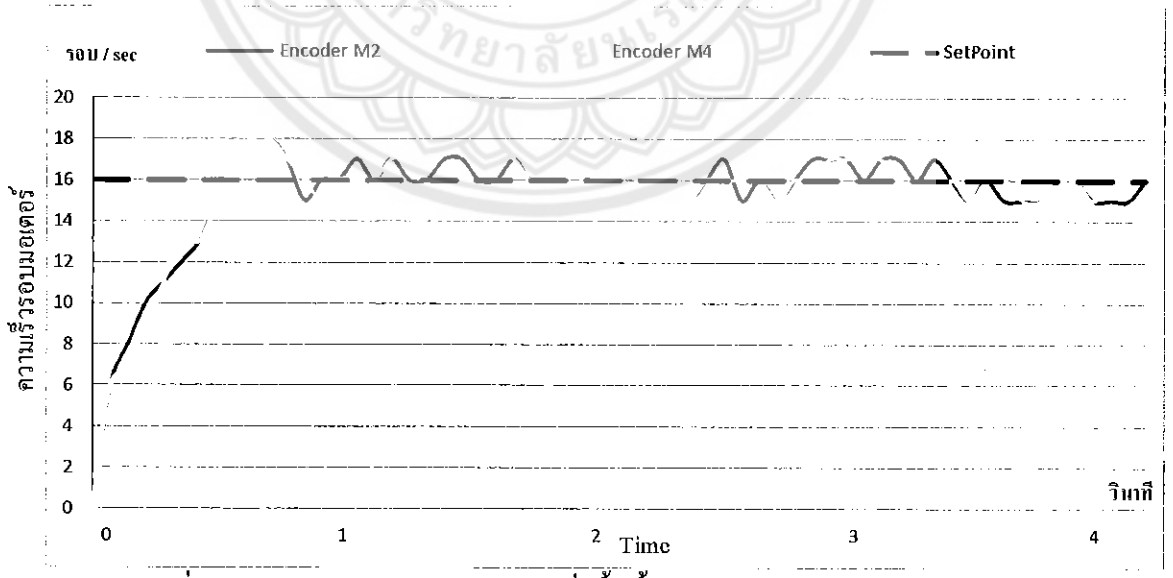
รูปที่ 4.18 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 20 องศา ด้วยความเร็วสูง วิ่งขึ้นเฉียง 45 องศา

4.3.16 หุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ความเร็วต่ำ ที่ Duty Cycle 25% วิ่งขึ้นเฉียง 45 องศา



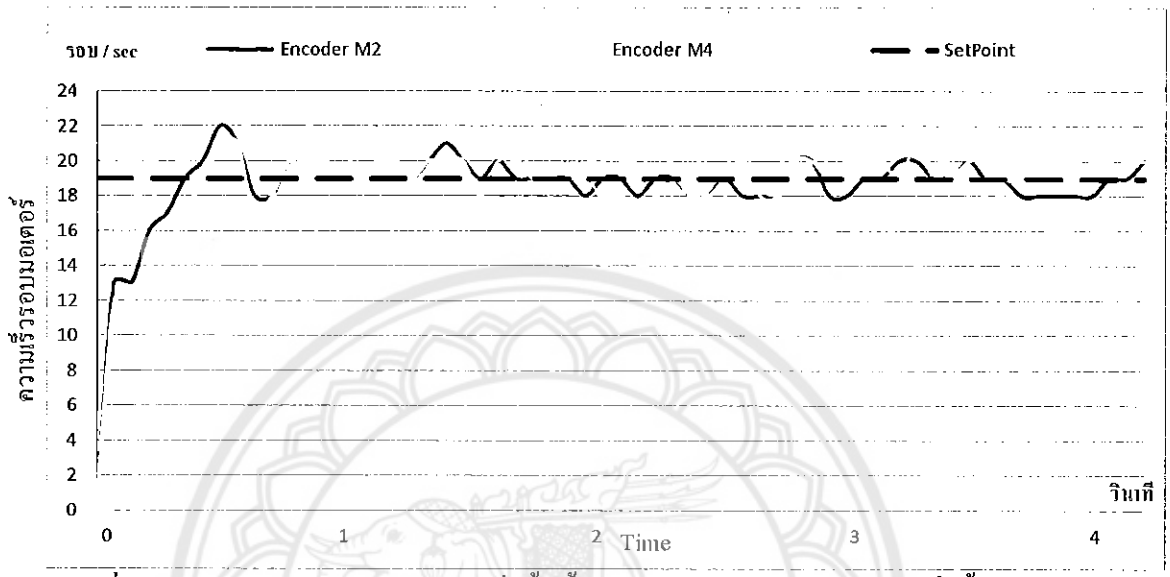
รูปที่ 4.19 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ด้วยความเร็วต่ำ วิ่งขึ้นเฉียง 45 องศา

4.3.17 หุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ความเร็วปานกลางที่ Duty Cycle 50% วิ่งขึ้นเฉียง 45 องศา



รูปที่ 4.20 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ด้วยความเร็วปานกลาง วิ่งขึ้นเฉียง 45 องศา

4.3.18 หุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ความเร็วสูง ที่ Duty Cycle 75% วิ่งขึ้นเฉียง 45 องศา



รูปที่ 4.21 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ด้วยความเร็วสูง วิ่งขึ้นเฉียง 45 องศา

หมายเหตุ: ค่า Setpoint ที่ได้จากการทดลองตั้งแต่รูปที่ 4.3 ถึง 4.21

เป็นค่าการทดลองของปี 2555 (ค่าเก่า)

4.4 สรุปประสิทธิภาพระบบควบคุมหุ่นยนต์จากกราฟ

จากรูปกราฟที่ 4.1 สรุปได้ดังตารางที่ 4.1

ตารางที่ 4.1 สรุปผลการทดลองรูปกราฟที่ 4.1

	มอเตอร์ 1 มอเตอร์ 2 มอเตอร์ 3 และมอเตอร์ 4
Delay Time (Sec)	0.60
Rise Time (Sec)	0.80
Setting Time (Sec)	2.40

จากรูปกราฟที่ 4.2 สรุปได้ดังตารางที่ 4.2

ตารางที่ 4.2 สรุปผลการทดลองรูปกราฟที่ 4.2

	มอเตอร์ 1 มอเตอร์ 2 มอเตอร์ 3 และมอเตอร์ 4
Delay Time (Sec)	0.40
Rise Time (Sec)	0.80
Setting Time (Sec)	1.60

จากรูปกราฟที่ 4.3 สรุปได้ดังตารางที่ 4.3

ตารางที่ 4.3 สรุปผลการทดลองรูปกราฟที่ 4.3

	มอเตอร์ 1 มอเตอร์ 2 มอเตอร์ 3 และมอเตอร์ 4
Delay Time (Sec)	0.40
Rise Time (Sec)	1.00
Setting Time (Sec)	1.20

จากรูปกราฟที่ 4.4 สรุปได้ดังตารางที่ 4.4

ตารางที่ 4.4 สรุปผลการทดลองรูปกราฟที่ 4.4

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	0.13	0.13	0.13	0.13
Rise Time (Sec)	0.60	0.53	0.46	0.46
Setting Time (Sec)	0.86	0.86	0.80	0.80

จากรูปกราฟที่ 4.5 สรุปได้ดังตารางที่ 4.5

ตารางที่ 4.5 สรุปผลการทดลองรูปกราฟที่ 4.5

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	0.13	0.13	0.13	0.13
Rise Time (Sec)	0.33	0.27	0.27	0.33
Setting Time (Sec)	0.60	0.60	0.60	0.60

จากรูปกราฟที่ 4.6 สรุปได้ดังตารางที่ 4.6

ตารางที่ 4.6 สรุปผลการทดลองรูปกราฟที่ 4.6

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	0.13	0.13	0.13	0.13
Rise Time (Sec)	0.27	0.27	0.27	0.27
Setting Time (Sec)	0.60	0.53	0.53	0.53

จากรูปกราฟที่ 4.7 สรุปได้ดังตารางที่ 4.7

ตารางที่ 4.7 สรุปผลการทดลองรูปกราฟที่ 4.7

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	0.20	0.20	0.20	0.20
Rise Time (Sec)	0.61	0.60	0.61	0.61
Setting Time (Sec)	0.87	0.73	0.61	0.61

จากรูปกราฟที่ 4.8 สรุปได้ดังตารางที่ 4.8

ตารางที่ 4.8 สรุปผลการทดลองรูปกราฟที่ 4.8

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	0.13	0.13	0.13	0.13
Rise Time (Sec)	0.53	0.53	0.47	0.47
Setting Time (Sec)	0.61	0.61	0.61	0.61

จากรูปกราฟที่ 4.9 สรุปได้ดังตารางที่ 4.9

ตารางที่ 4.9 สรุปผลการทดลองรูปกราฟที่ 4.9

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	0.13	0.13	0.13	0.13
Rise Time (Sec)	0.40	0.40	0.40	0.40
Setting Time (Sec)	0.61	0.61	0.61	0.61

จากรูปกราฟที่ 4.10 สรุปได้ดังตารางที่ 4.10

ตารางที่ 4.10 สรุปผลการทดลองรูปกราฟที่ 4.10

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	0.20	0.20	0.20	0.20
Rise Time (Sec)	0.61	0.61	0.73	0.61
Setting Time (Sec)	0.87	0.93	0.87	0.93

จากรูปกราฟที่ 4.11 สรุปได้ดังตารางที่ 4.11

ตารางที่ 4.11 สรุปผลการทดลองรูปกราฟที่ 4.11

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	0.13	0.13	0.13	0.13
Rise Time (Sec)	0.60	0.61	0.60	0.60
Setting Time (Sec)	0.73	0.73	0.61	0.61

จากรูปกราฟที่ 4.12 สรุปได้ดังตารางที่ 4.12

ตารางที่ 4.12 สรุปผลการทดลองรูปกราฟที่ 4.12

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	0.13	0.13	0.13	0.13
Rise Time (Sec)	0.61	0.60	0.61	0.60
Setting Time (Sec)	0.73	0.73	0.61	0.61

จากรูปกราฟที่ 4.13 สรุปได้ดังตารางที่ 4.13

ตารางที่ 4.13 สรุปผลการทดลองรูปกราฟที่ 4.13

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	-	0.20	-	0.20
Rise Time (Sec)	-	0.47	-	0.53
Setting Time (Sec)	-	0.61	-	0.61

จากรูปกราฟที่ 4.14 สรุปได้ดังตารางที่ 4.14

ตารางที่ 4.14 สรุปผลการทดลองรูปกราฟที่ 4.14

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	-	0.13	-	0.13
Rise Time (Sec)	-	0.33	-	0.33
Setting Time (Sec)	-	0.60	-	0.60

จากรูปกราฟที่ 4.15 สรุปได้ดังตารางที่ 4.15

ตารางที่ 4.15 สรุปผลการทดลองรูปกราฟที่ 4.15

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	-	0.13	-	0.13
Rise Time (Sec)	-	0.27	-	0.27
Setting Time (Sec)	-	0.40	-	0.40

จากรูปกราฟที่ 4.16 สรุปได้ดังตารางที่ 4.16

ตารางที่ 4.16 สรุปผลการทดลองรูปกราฟที่ 4.16

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	-	0.20	-	0.20
Rise Time (Sec)	-	0.47	-	0.53
Setting Time (Sec)	-	0.80	-	0.80

จากรูปกราฟที่ 4.17 สรุปได้ดังตารางที่ 4.17

ตารางที่ 4.17 สรุปผลการทดลองรูปกราฟที่ 4.17

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	-	0.13	-	0.13
Rise Time (Sec)	-	0.40	-	0.53
Setting Time (Sec)	-	0.53	-	0.61

จากรูปกราฟที่ 4.18 สรุปได้ดังตารางที่ 4.18

ตารางที่ 4.18 สรุปผลการทดลองรูปกราฟที่ 4.18

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	-	0.13	-	0.13
Rise Time (Sec)	-	0.40	-	0.40
Setting Time (Sec)	-	0.53	-	0.53

จากรูปกราฟที่ 4.19 สรุปได้ดังตารางที่ 4.19

ตารางที่ 4.19 สรุปผลการทดลองรูปกราฟที่ 4.19

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	-	0.20	-	0.20
Rise Time (Sec)	-	0.40	-	0.47
Setting Time (Sec)	-	0.87	-	0.87

จากรูปกราฟที่ 4.20 สรุปได้ดังตารางที่ 4.20

ตารางที่ 4.20 สรุปผลการทดลองรูปกราฟที่ 4.20

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	-	0.13	-	0.13
Rise Time (Sec)	-	0.47	-	0.47
Setting Time (Sec)	-	0.80	-	0.80

จากรูปกราฟที่ 4.21 สรุปได้ดังตารางที่ 4.21

ตารางที่ 4.21 สรุปผลการทดลองรูปกราฟที่ 4.21

	มอเตอร์ 1	มอเตอร์ 2	มอเตอร์ 3	มอเตอร์ 4
Delay Time (Sec)	-	0.13	-	0.13
Rise Time (Sec)	-	0.33	-	0.40
Setting Time (Sec)	-	0.73	-	0.73

4.5 สรุปผลการทดลอง

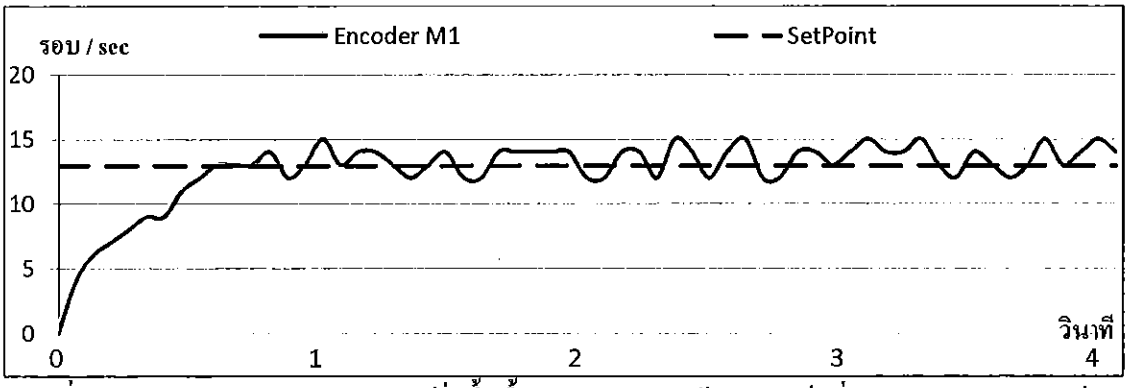
จากรูปกราฟทุกรูปจะเห็นว่าระบบจะทำการปรับความเร็วของมอเตอร์ให้อยู่ในค่า Setpoint ที่กำหนดไว้และเมื่อระบบเข้าสู่ช่วง Setting Time แล้วค่า Error จะอยู่ในช่วงระหว่างบวกลบ 1 - 4 ซึ่งถ้าเทียบเป็นเปอร์เซ็นของค่า Error สูงสุด จะเท่ากับบวกลบ 2-8 % และค่า Delay Time ของกราฟแต่ละกราฟจะมีค่าไม่เกิน 1 วินาที (ยกเว้นกราฟที่ 4.1 4.2 และ 4.3 เพราะมอเตอร์ที่ใช้ทดลองมีการเสื่อมสภาพตามระยะเวลา มอเตอร์วิ่งช้ากว่าปกติ ทำให้ต้องลดความเร็วการทำงานของโปรแกรมลงเพื่อที่จะสามารถนับความเร็วรอบของมอเตอร์ได้ปกติ ซึ่งมีผลทำให้ประสิทธิภาพการทำงานของหุ่นยนต์ลดลงตามไปด้วย) ส่วนค่า Setting Time มีค่าต่างกันตามอุปสรรคที่เจอ เช่น พื้นเอียง 10 องศา ค่า Setting Time จะมีค่าไม่มากเนื่องจากแรงต้านมีค่าน้อย (แรงต้านที่มุม 10 องศามีค่าเท่ากับ 2.60 N) พื้นเอียง 30 องศา ค่า Setting Time จะมีค่ามากขึ้นเนื่องจากเจอแรงต้านที่เยอะกว่า (แรงต้านที่มุม 30 องศามีค่าเท่ากับ 7. 37 N)

บทที่ 5
สรุปผลการปฏิบัติโครงงาน

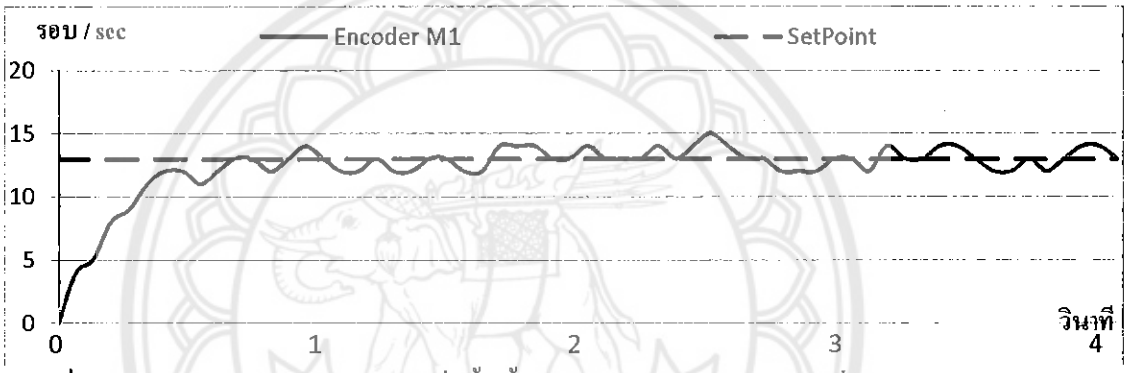
5.1 วิเคราะห์ผลการปฏิบัติงาน

จากรูปที่ 5.1 และรูปที่ 5.2 จะเห็นว่าค่า Setpoint มีค่าเท่ากันหมายความว่า ความเร็วที่ใช้มีค่าเท่ากันนั่นคือใช้ค่า ความเร็วต่ำ แต่อุปสรรคที่เจอนั้นต่างกัน คือ รูปที่ 5.1 เจอพื้นเอียง 10 องศา ส่วนรูปที่ 5.2 เจอพื้นเอียง 30 องศาและให้สังเกตในช่วงวินาทีที่ 1 จะเป็นช่วงเวลาที่หุ่นยนต์ปรับความเร็วเพื่อให้หุ่นยนต์วิ่งขึ้นพื้นเอียงได้ ซึ่งผลการตอบสนองของหุ่นยนต์จากรูปที่ 5.1 ค่า Delay Time มีค่าเท่ากับ 0.13 วินาที ค่า Rise Time มีค่าเท่ากับ 0.60 วินาที และค่า Setting Time มีค่าเท่ากับ 0.87 วินาที ส่วนรูปที่ 5.2 ค่า Delay Time มีค่าเท่ากับ 0.20 วินาทีและค่า Rise Time มีค่าเท่ากับ 0.67 วินาที ค่า Setting Time มีค่าเท่ากับ 0.87 วินาที

ผลลัพธ์ที่ได้คือ ผลการตอบสนองของหุ่นยนต์มีค่าไม่เท่ากัน ซึ่งค่า Delay Time ของทั้ง 2 รูปห่างกันประมาณ 0.07 วินาที ส่วนค่า Rise Time ของทั้ง 2 รูปห่างกันประมาณ 0.06 วินาที และค่า Setting Time ของทั้ง 2 รูปมีค่าเท่ากัน เหตุที่ทำให้ผลการตอบสนองของหุ่นยนต์มีค่าไม่เท่ากันคืออุปสรรคที่เจอนั้นแตกต่างกัน แต่ผลการตอบสนองของหุ่นยนต์นั้นมีค่าแตกต่างกันไม่มาก (ไม่ถึง 0.1 วินาที) นั่นหมายความว่า ระบบควบคุมแบบฟัซซี่ สามารถควบคุมความเร็วของหุ่นยนต์ได้เป็นอย่างดี จึงทำให้หุ่นยนต์ปรับความเร็วทันต่ออุปสรรคที่เจอ



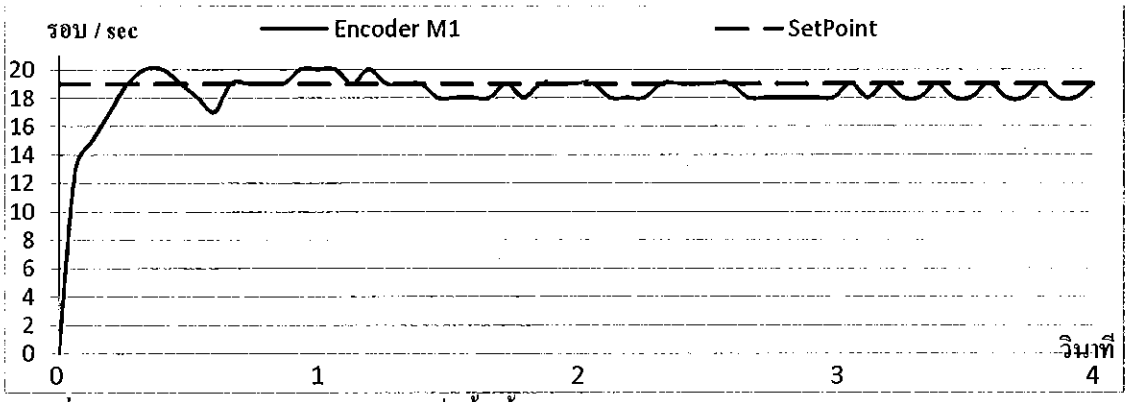
รูปที่ 5.1 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ด้วยความเร็วต่ำ ยกมาเฉพาะมอเตอร์ 1



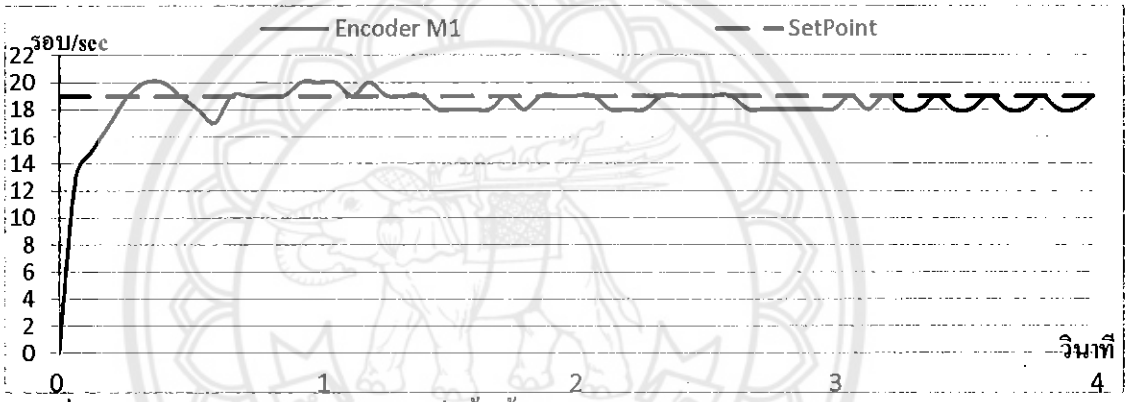
รูปที่ 5.2 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ด้วยความเร็วต่ำ ยกมาเฉพาะมอเตอร์ 1

และจากรูปที่ 5.3 และ 5.4 ความเร็วที่ใช้มีค่าเท่ากันนั่นคือใช้ค่า ความเร็วสูง แต่อุปสรรคที่เจอนั้นต่างกันเช่นกัน คือรูปที่ 5.3 เจอพื้นเอียง 10 องศา ส่วนรูปที่ 5.4 เจอพื้นเอียง 30 องศา ซึ่งผลการตอบสนองของหุ่นยนต์จากรูปที่ 5.3 ค่า Delay Time มีค่าท่ากับ 0.13 วินาที ค่า Rise Time มีค่าเท่ากับ 0.27 วินาที ค่า Setting Time มีค่าเท่ากับ 0.60 วินาที ส่วนรูปที่ 5.4 ค่า Delay Time มีค่าเท่ากับ 0.13 วินาที ค่า Rise Time มีค่าเท่ากับ 0.67 วินาที ค่า Setting Time มีค่าเท่ากับ 0.73 วินาที

ผลลัพธ์ที่ได้คือ ค่า Rise Time ที่จับได้ของทั้ง 2 รูป มีค่าห่างกันประมาณ 0.40 วินาที และค่า Setting Time ของทั้ง 2 รูปมีค่าห่างกันประมาณ 0.10 วินาที ส่วนค่า Delay Time ของทั้ง 2 รูป มีค่าเท่ากัน ทำให้บ่งบอกได้ว่า กำลังไฟที่จ่ายให้กับหุ่นยนต์มีผลต่อการตอบสนองของหุ่นยนต์ด้วย ถ้ากำลังไฟที่จ่ายให้หุ่นยนต์อยู่ในระดับสูง (Duty Cycle = 50% ขึ้นไป) จะทำให้ระบบควบคุมแบบฟัซซี่สามารถควบคุมความเร็วของหุ่นยนต์ได้ดียิ่งขึ้น



รูปที่ 5.3 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 10 องศา ด้วยความเร็วสูง ยกมาเฉพาะมอเตอร์ 1



รูปที่ 5.4 กราฟผลการทดลองหุ่นยนต์วิ่งขึ้นพื้นเอียง 30 องศา ด้วยความเร็วสูง ยกมาเฉพาะมอเตอร์ 1

5.2 สรุปผลการปฏิบัติงาน

หุ่นยนต์สามารถวิ่งได้ 8 ทิศทางและสามารถควบคุมความเร็วของมอเตอร์ได้โดยใช้ระบบควบคุมแบบฟัซซี่ เมื่อหุ่นยนต์เจออุปสรรคในการเคลื่อนที่ เช่น การเคลื่อนที่ขึ้นพื้นเอียงที่มีองศาไม่เท่ากัน หุ่นยนต์จะทำการปรับความเร็วของตัวเองจากการควบคุมระบบแบบฟัซซี่ทำให้หุ่นยนต์สามารถผ่านอุปสรรคได้ตามที่ต้องการ รวมถึงได้เรียนรู้และได้ใช้งานทฤษฎีการควบคุมแบบฟัซซี่ซึ่งสามารถนำมาประยุกต์ใช้งานในโรงงานอุตสาหกรรมและศูนย์วิจัยต่างๆ ได้และได้เรียนรู้ทฤษฎีการใช้งานของ Pulse Width Modulation และได้เรียนรู้การเขียนโปรแกรมให้กับไมโครคอนโทรลเลอร์ซึ่งเป็นภาษา C ได้เรียนรู้เกี่ยวกับ Library ของไมโครคอนโทรลเลอร์ตระกูล dsPIC30F และได้เขียนภาษา C ในการควบคุมไมโครคอนโทรลเลอร์จริง

5.2 ประโยชน์ที่ได้รับจากการทำงาน

1. สามารถควบคุมความเร็วของมอเตอร์ด้วยระบบควบคุมแบบป้อนกลับได้ตามต้องการ
2. ได้เรียนรู้และเข้าใจในการนำทฤษฎีฟัซซี่ มาประยุกต์กับงานที่ทำ
3. ได้เรียนรู้การทำงานของ dsPIC30F4011
4. ได้เรียนรู้การทำงานของ Pulse Width Modulation

5.3 แนวทางพัฒนาระบบต่อไปในอนาคต

1. สามารถนำระบบควบคุมแบบ ฟัซซี่ ไปใช้ในการควบคุมความเร็วของมอเตอร์ที่ต้องการค่าความละเอียดสูงกว่านี้ได้ เช่น ควบคุมความเร็วของเครื่องบิน, ควบคุมความเร็วของรถยนต์ เป็นต้น

2. สามารถนำหลักการทำงานของระบบควบคุมแบบ ฟัซซี่ ไปใช้ในการควบคุมการทำงานอื่นๆได้ เช่น หลักการทำงานของเครื่องซักผ้า, หลักการจ่ายค่าทิปของร้านค้า เป็นต้น

3. สามารถนำหุ่นยนต์ตัวนี้ไปติดตั้งกล่องสัญญาณ Wireless / Bluethooth เพื่อใช้ควบคุมหุ่นยนต์ในระยะไกลได้

4. สามารถติดกล้องบนตัวหุ่นยนต์เพื่อควบคุมทิศทางอัตโนมัติได้

5. สามารถนำกล้องถ่ายภาพติดไว้บนตัวหุ่นยนต์เพื่อถ่ายภาพในสถานที่ที่อันตรายหรือเอาไว้ตรวจสอบวัตถุอันตราย หรือ วัตถุระเบิด แทนมนุษย์เพื่อลดความเสี่ยงต่อชีวิตได้

6. สามารถนำหุ่นยนต์ตัวนี้ไปเปลี่ยนล้อเป็นล้อยาง เพื่อให้หุ่นยนต์สามารถเคลื่อนที่ไปในสถานที่เข้าถึงยากหรือมนุษย์ไม่สามารถเข้าถึงได้ โดยใช้หุ่นยนต์เป็นตัวช่วยในการสำรวจภาคพื้นดินได้

เอกสารอ้างอิง

[1] Microchip. (2004). dsPIC ® Language Tools Libraries. U.S.A: Microchip Technology Incorporated

[2] Microchip. (2008). คู่มือการใช้งาน ET-PGMPIC USB. กรุงเทพ ฯ : บริษัท อีทีที จำกัด

[3] จาก http://pic16f627a.blogspot.com/2012/02/pwm-by-pic.html

[4] Timothy J.Ross. In University of New Mexico,USA, pages 74, 2004

[5] จาก http://www.mathworks.com/help/fuzzy/defuzz.html

[6] ดร.พยุง มีสัจ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ http://suanpalm3.kmitnb.ac.th/teacher/phayung/powerpoint.asp?pno=1

[7] จาก http://www.st.kmutt.ac.th/~s8530007/student_files/Computation%20AI/Chapter18FuzzyLogic.pdf

[8] Prakash Nadkarni. (2013). Fuzzy control system.

[9] จาก http://www.nawattakam.com/talk/index.php?topic=368.0

[10] จาก http://www.thaieasyelec.com/article-wiki/basic-electronics/uart-ttl-rs232-max232-max3232.html

[11] จาก http://www.st.com/content/st_com/en/products/motor-drivers/brushed-dc-motor-drivers/l298.html

[12] จาก http://arduinoo.lnwshop.com/article/1/การใช้งาน-ic-voltage-regulator-780578097812

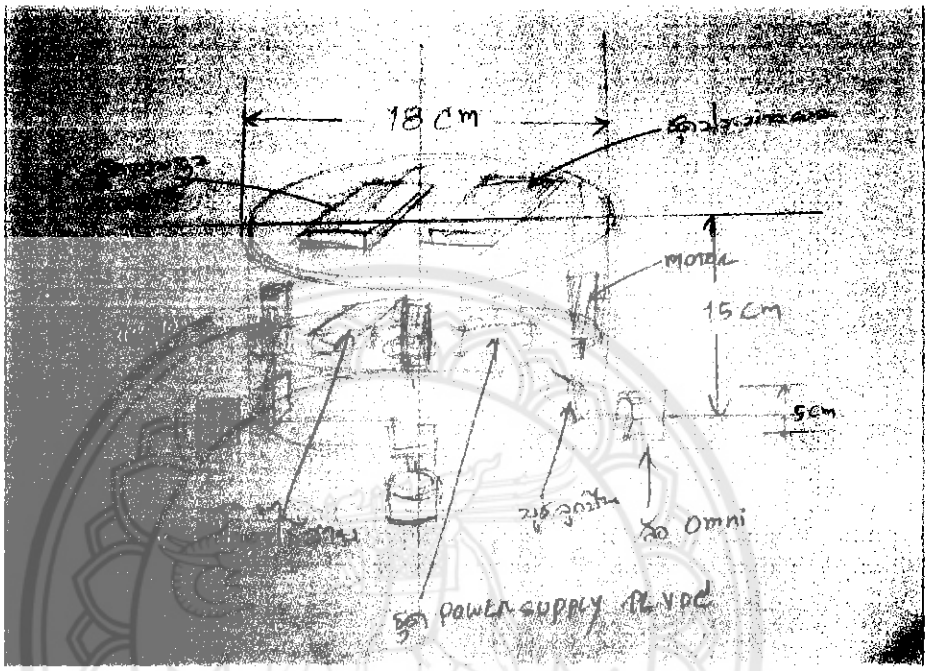
[13] จาก https://th.wikipedia.org/wiki/ตรรกศาสตร์คลุมเครือ

[14] จาก http://www.bindichen.co.uk/post/AI/fuzzy-inference-membership-function.html

[15] จาก http://www.shadowwares.com/forum/index.php/topic,803.0.html

ภาคผนวก

ภาพโครงสร้างการออกแบบหุ่นยนต์ล้อโอมนิ

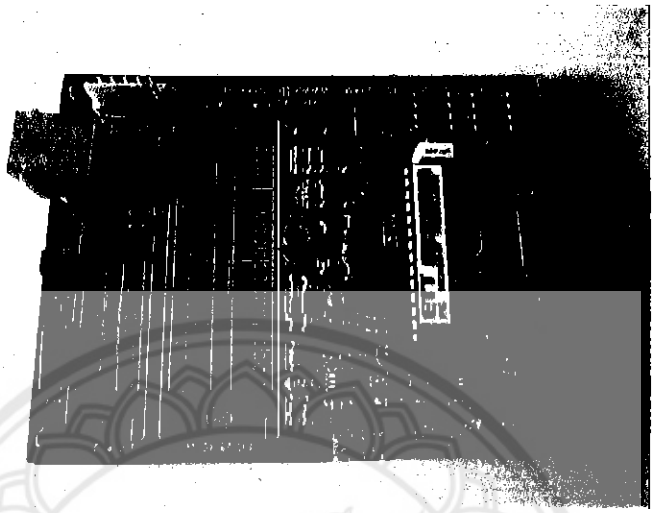


รูปที่ ก.1 ภาพโครงสร้างการออกแบบหุ่นยนต์ล้อโอมนิ

Power Supply 12 VDC ที่ใช้ในโครงงาน

รูปที่ ก.2 Power Supply 12 VDC

ET-PGM PIC USB v1 เป็นตัวที่ใช้ Burn โปรแกรมให้กับ Microcontroller dsPIC30F4011



รูปที่ ก.3 ET-PGM PIC USB V1

วงจร Encoder ที่ใช้ในการนับรอบมอเตอร์

รูปที่ ก.4 วงจรนับรอบ Encoder Motor

ประวัติผู้จัดทำโครงงาน

ชื่อ นาย กฤษฎา ศิริวัฒนธานี

ภูมิลำเนา 108/20 ถนนวิสุทธิกษัตริย์ ตำบลในเมือง อำเภอเมือง จังหวัดพิษณุโลก

เกิดวันที่ 2 ตุลาคม พ.ศ. 2532 ที่จังหวัด ลำปาง

ประวัติการศึกษา

- ระดับมัธยมศึกษาตอนต้น โรงเรียนพิษณุโลกพิทยาคม จังหวัดพิษณุโลก
- ระดับมัธยมศึกษาตอนปลาย โรงเรียนพิษณุโลกพิทยาคม จังหวัดพิษณุโลก
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรี
- สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail: kitsada34146@gmail.com