การศึกษาตัวควบคุมแบบพี/พีไอ สำหรับการควบคุมความเร็วของมอเตอร์
A Study on P/PI Controller for Controlling Speed of DC Motor

กฤติน สุขมี
นนทกร บุญยืน
นภัสสร เซ่งตระกูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2562

การศึกษาตัวควบคุมแบบพี/พีไอ สำหรับการควบคุมความเร็วของมอเตอร์
A Study on P/PI Controller for Controlling Speed of DC Motor

กฤติน สุขมี
นนทกร บุญยืน
นภัสสร เซ่งตระกูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2562

A Study on P/PI Controller for Controlling Speed of DC Motor

KRITTIN SUKMEE
NONTAKORN BOONYUEN
NAPASSORN SENGTRAKOOL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING
FUCULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2019

หัวข้อปริญญานิพนธ์	การศึกษาตัวควบคุมแบบพี/พีไอ สำหรับการควบคุมความเร็วของมอเตอร์			
	A Study on P/PI Controller for Controlling Speed of DC Motor			
นักศึกษาผู้จัดทำ	นายกฤติน	สุขมี	รหัสนักศึกษา	59010036
	นายนนทกร	บุญยืน	รหัสนักศึกษา	59010680
	นางสาวนภัสสร	เซ่งตระกูล	รหัสนักศึกษา	59010695
อาจารย์ที่ปรึกษา	ดร.นภศูล	วงษ์วานิช		
ปีการศึกษา	2562			

บทคัดย่อ

ในโครงงานปริญญานิพนธ์ฉบับนี้ เป็นการศึกษาและออกแบบตัวควบคุมแบบพี/พีไอ (P/PI) สำหรับมอเตอร์ไฟฟ้ากระแสตรง โดยจะควบคุมการทำงานของพารามิเตอร์ ซึ่งคือ ความเร็ว (RPM) ของมอเตอร์ โดยจะจำลองการควบคุม (Simulation Test) ผ่านโปรแกรม MATLAB โดยใช้ Simulink Toolbox หลังจากนั้น จะนำโปรแกรมที่สมบูรณ์แล้วไปสร้างเป็นคำสั่งในโปรแกรม Arduino IDE เพื่อควบคุมมอเตอร์ต่อไป เมื่อทำการศึกษาแล้วจะได้ค่าคงที่ของตัวควบคุม ผลลัพธ์ที่ได้จะทำให้สามารถออกแบบตัวควบคุมที่ทำให้มอเตอร์ไฟฟ้ากระแสตรง มีผลการตอบสนองตามที่ต้องการได้

Thesis Title	A Study on P/PI Controller for Controlling Speed of DC Motor	
Authors	Mr. krittin	Sukmee
	Mr. Nontakorn	Boonyuen
	Ms. Napassorn	Sengtrakool
Thesis Advisor	Dr. Napasool	Wongvanich
Year	2019	

Abstract

In this report, it is the study and design of P/PI controller for DC Motor by controlling the operation of speed of the motor which is the important parameter that can tell performance of DC Motor. Normally, DC Motor is dynamic system that have own settling time. Controller can make system faster or slower. First, create a simulation test in MATLAB by Simulink toolbox. After that, the completed program will be created as an instruction in the Arduino IDE program to control the real motor. After the study, the controller will receive the right constant value. The result is that the controller can be designed to make the DC Motor reach steady state as desired value.

กิตติกรรมประกาศ

โครงงานปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี เนื่องจากความกรุณาและคำปรึกษาจากอาจารย์ ดร.นภศูล วงษ์วานิช อาจารย์ที่ปรึกษาโครงงานนี้ และ ผศ.ดร.สังวาล บกสุวรรณ อาจารย์ประจำภาควิชาวิศวกรรมเมคคาทรอนิกส์ ที่ได้แนะนำแนวทางการทำโครงงาน และวิธีการแก้ไขข้อผิดพลาดต่างๆ มาโดยตลอด รวมไปถึงแนวคิดและวิธีการทำงานวิจัย จนโครงงานสำเร็จลุล่วงไปได้ด้วยดี อีกทั้งยังจัดหาอุปกรณ์ต่างๆ ในการทำโครงงาน ทำให้โครงงานสำเร็จลุล่วงได้รวดเร็วยิ่งขึ้น และขอขอบพระคุณคณาจารย์ภาควิชาวิศวกรรมการวัดคุม ทุกๆ ท่านที่ได้สอนวิชาความรู้ต่างๆ รวมไปถึงการอำนวยความสะดวกของห้องทดลองต่างๆ และหวังว่าความรู้ที่ได้จากการทำโครงงานในครั้งนี้จะสามารถนำไปใช้ในอนาคตต่อไปในภายภาคหน้าได้

คณะผู้จัดทำ

สารบัญ

หน้า

บทคัดย่อภาษาไทย I
บทคัดย่อภาษาอังกฤษ II
กิตติกรรมประกาศ III
สารบัญ IV
สารบัญตาราง IX
สารบัญรูป X

บทที่ 1 บทนำ 1
1.1 ความสำคัญของปริญญานิพนธ์ 1
1.2 วัตถุประสงค์ของปริญญานิพนธ์ 1
1.3 ขอบเขตของปริญญานิพนธ์ 1
1.4 ประโยชน์ที่คาดว่าจะได้รับ 2

บทที่ 2 ทฤษฎีและหลักการที่เกี่ยวข้อง 3
2.1 มอเตอร์ไฟฟ้ากระแสตรง (DC motor) 3
2.1.1 ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง 3
2.1.1.1 สเตเตอร์ (Stator) 3
2.1.1.2 โรเตอร์ (Rotor) 3
2.1.2 หลักการทำงานของมอเตอร์ไฟฟ้ากระแสตรง 4
2.1.3 การควบคุมมอเตอร์ไฟฟ้ากระแสตรง 4
2.2 เอ็นโค้ดเดอร์แบบแกนหมุน (Rotary Encoder) 4
2.2.1 ส่วนประกอบของเอ็นโค้ดเดอร์แบบแกนหมุน 5
2.2.2 การทำงานของเอ็นโค้ดเดอร์แบบแกนหมุน 5
2.2.3 ตัวอย่างของเอ็นโค้ดเดอร์แบบแกนหมุน 6
2.3 การควบคุมแบบ PID (Proportional Integrated Controller) 7
2.3.1 ตัวควบคุมแบบพี (P-Controller) 7
2.3.2 ตัวควบคุมแบบไอ (I-Controller) 8
2.3.3 ตัวควบคุมแบบดี (D-Controller) 8
2.4 การหาลักษณะเฉพาะของระบบ (System Identification) 8
2.4.1 ข้อมูลที่ใช้ในการหาลักษณะเฉพาะของระบบ 8
2.4.2 System Identification Toolbox 8
2.5 MATLAB 9
2.5.1 หลักการทำงานของ MATLAB 9

สารบัญ (ต่อ)

หน้า

2.5.2 ความสามารถของโปรแกรม MATLAB ในด้านวิศวกรรม 9

บทที่ 3 ขั้นตอนและวิธีการดำเนินงาน 10

3.1 การศึกษาตัวควบคุมในการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง 10

3.1.1 การควบคุมความเร็วของมอเตอร์แบบวงเปิด (Open Loop Control) 11

3.1.2 การควบคุมความเร็วของมอเตอร์แบบวงปิด (Closed Loop Control) 17

3.1.2.1 การควบคุมมอเตอร์ไฟฟ้ากระแสตรงโดยใช้ตัวควบคุมแบบพี 18

3.1.2.2 การควบคุมมอเตอร์ไฟฟ้ากระแสตรงโดยใช้ตัวควบคุมแบบพีไอ 21

3.2 การทดลองที่เกี่ยวข้องกับการทำโครงงานของปริญญานิพนธ์ 23

3.2.1 การทดลองการสร้างคาบการสุ่ม (Sampling Time) 23

3.2.1.1 จุดประสงค์การทดลองสร้างคาบการสุ่ม 23

3.2.1.2 อุปกรณ์ที่ใช้ในการทดลองสร้างคาบการสุ่ม 23

3.2.1.3 วิธีทำการทดลองสร้างคาบการสุ่ม 24

3.2.1.4 ผลการทดลองสร้างคาบการสุ่ม 25

3.2.1.4 สรุปผลการทดลองสร้างคาบการสุ่ม 25

3.2.2 การทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด 28
(Open loop control)

3.2.2.1 จุดประสงค์ของการทดลองการควบคุมความเร็ว 28
ของมอเตอร์แบบวงเปิด

3.2.2.2 อุปกรณ์ที่ใช้ในการทดลองการควบคุมความเร็ว 28
ของมอเตอร์แบบวงเปิด

3.2.2.3 วิธีทำการทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด 29

3.2.2.4 ผลการทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด 31

3.2.2.5 สรุปผลการทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด 39

3.2.3 การหาฟังก์ชันถ่ายโอนของระบบ (Transfer Function) 41

3.2.3.1 จุดประสงค์ของการทดลองการหาฟังก์ชันถ่ายโอนของระบบ 42

3.2.3.2 อุปกรณ์ที่ใช้ในการทดลองการหาฟังก์ชันถ่ายโอนของระบบ 42

3.2.3.3 วิธีการทำการทดลองการหาฟังก์ชันถ่ายโอนของระบบ 42

3.2.3.4 ผลการทดลองการหาฟังก์ชันถ่ายโอนของระบบ 48

3.2.3.5 สรุปผลการทดลองการหาฟังก์ชันถ่ายโอนของระบบ 48

3.2.4 การจำลองตัวควบคุมแบบพี (P controller) ในการควบคุม 49
ความเร็วของมอเตอร์ในโปรแกรม MATLAB

3.2.4.1 จุดประสงค์การทดลองการจำลองตัวควบคุมแบบพี 49
(P controller) ในการควบคุมความเร็วของมอเตอร์
ในโปรแกรม MATLAB

สารบัญ (ต่อ)

หน้า

3.2.4.2 อุปกรณ์การทดลองการจำลองตัวควบคุมแบบพี (P controller)..... 49
ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB
3.2.4.3 วิธีการทดลองการจำลองตัวควบคุมแบบพี (P controller)..... 49
ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB
3.2.4.4 ผลการทดลองการจำลองตัวควบคุมแบบพี (P controller) 56
ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB
3.2.4.5 สรุปผลการทดลองการจำลองตัวควบคุมแบบพี (P controller)..... 57
ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB
3.2.5 การจำลองตัวควบคุมแบบพี-ไอ (PI controller) ในการควบคุมความเร็ว 58
ของมอเตอร์ในโปรแกรม MATLAB
3.2.5.1 จุดประสงค์การทดลองการจำลองตัวควบคุมแบบพี-ไอ 58
(PI controller) ในการควบคุมความเร็วของมอเตอร์
ในโปรแกรม MATLAB
3.2.5.2 อุปกรณ์ที่ใช้ในการทดลองการจำลองตัวควบคุมแบบพี-ไอ 58
(PI controller) ในการควบคุมความเร็วของมอเตอร์
ในโปรแกรม MATLAB
3.2.5.3 วิธีการทดลองการจำลองตัวควบคุมแบบพี-ไอ (PI controller)..... 58
ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB
3.2.5.4 ผลการทดลองการจำลองตัวควบคุมแบบพี-ไอ (PI controller) 60
ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB
3.2.5.5 สรุปผลการทดลองการจำลองตัวควบคุมแบบพี-ไอ (PI controller)...60
ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB
3.2.6 การจำลองตัวควบคุมแบบพี-ไอโดยใช้เทคนิค Anti-Windup ในการ 62
ควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB
3.2.6.1 จุดประสงค์การทดลองการจำลองตัวควบคุมแบบพี-ไอ 63
โดยใช้เทคนิค Anti-Windup ในการควบคุมความเร็ว
ของมอเตอร์ในโปรแกรม MATLAB
3.2.6.2 อุปกรณ์ที่ใช้ในการทดลองการจำลองตัวควบคุมแบบพี-ไอ 63
โดยใช้เทคนิค Anti-Windup ในการควบคุมความเร็ว
ของมอเตอร์ในโปรแกรม MATLAB
3.2.6.3 วิธีการทดลองการจำลองตัวควบคุมแบบพี-ไอ..... 63
โดยใช้เทคนิค Anti-Windup ในการควบคุมความเร็ว
ของมอเตอร์ในโปรแกรม MATLAB

สารบัญ (ต่อ)

หน้า

3.2.6.4 ผลการทดลองการจำลองตัวควบคุมแบบพี-ไอ..... 66
โดยใช้เทคนิคAnti-Windup ในการควบคุมความเร็ว
ของมอเตอร์ในโปรแกรม MATLAB
3.2.6.5 สรุปผลการทดลองการจำลองตัวควบคุมแบบพี-ไอ..... 67
โดยใช้เทคนิค Anti-Windup ในการควบคุมความเร็ว
ของมอเตอร์ในโปรแกรม MATLAB
3.2.7 การจำลองตัวควบคุมแบบพี/พีไอ (P/PI controller) ในการ 68
ควบคุมตำแหน่งเชิงมุมของมอเตอร์ในโปรแกรม MATLAB
3.2.7.1 จุดประสงค์การทดลองการจำลองตัวควบคุมแบบพี/พีไอ 68
(P/PI controller) ในการควบคุมตำแหน่งเชิงมุม
ของมอเตอร์ในโปรแกรม MATLAB
3.2.7.2 อุปกรณ์ที่ใช้ในการทดลองการจำลองตัวควบคุมแบบพี/พีไอ 69
(P/PI controller) ในการควบคุมตำแหน่งเชิงมุม
ของมอเตอร์ในโปรแกรม MATLAB
3.2.7.3 วิธีการทดลองการจำลองตัวควบคุมแบบพี/พีไอ 69
(P/PI controller) ในการควบคุมตำแหน่งเชิงมุม
ของมอเตอร์ในโปรแกรม MATLAB
3.2.7.4 ผลการทดลองการจำลองตัวควบคุมแบบพี/พีไอ 70
(P/PI controller) ในการควบคุมตำแหน่งเชิงมุม
ของมอเตอร์ในโปรแกรม MATLAB
3.2.7.5 สรุปผลการทดลองการจำลองตัวควบคุมแบบพี/พีไอ..... 71
(P/PI controller) ในการควบคุมตำแหน่งเชิงมุม
ของมอเตอร์ในโปรแกรม MATLAB

บทที่ 4 ผลการทดลอง 72

4.1 ผลการทดลองของการควบคุมมอเตอร์โดยใช้ตัวควบคุมแบบพี (P Controller)..... 72
4.1.1 ความเร็วของมอเตอร์ของการใช้ตัวควบคุมแบบพี 72
ในแต่ละค่าของ Set Point
4.1.2 สัญญาณควบคุมของการใช้ตัวควบคุมแบบพี ในแต่ละค่าของ Set Point..... 78
4.2 ผลการทดลองของการควบคุมมอเตอร์โดยใช้ตัวควบคุมแบบพีไอ (PI Controller).... 83
4.2.1 ความเร็วของมอเตอร์ของการใช้ตัวควบคุมแบบพีไอ..... 84
ในแต่ละค่าของ Set Point
4.2.2 สัญญาณควบคุมของการใช้ตัวควบคุมแบบพีไอ ในแต่ละค่าของ Set Point.... 89

สารบัญ (ต่อ)

หน้า

บทที่ 5 สรุปผลการทดลองและข้อเสนอแนะ 94
5.1 สรุปผลการทดลอง 94
5.2 ข้อเสนอแนะ 94

บรรณานุกรม.....95

ภาคผนวก 96

สารบัญตาราง

ตารางที่ **หน้า**

3.1 ตารางแสดงคุณสมบัติ Specification ของ STM32f103..... 27
4.1 ตารางแสดงค่าของ Steady State Error ของแต่ละ Set Point..... 78

สารบัญรูป

รูปที่ **หน้า**

2.1 แสดงการทำงานของมอเตอร์ไฟฟ้ากระแสตรง....4
2.2 การทำงานของเอนโค้ดเดอร์แบบแกนหมุน....5
2.3 แสดงเอ็นโค้ดเดอร์แบบแกนหมุน ยี่ห้อ Kubler รุ่น 3700 6
2.4 แสดงหน้าที่ของเส้นแต่ละเส้นที่ต่อจากเอ็นโค้ดเดอร์ 6
3.1 แสดงมอเตอร์ไฟฟ้ากระแสตรงที่เลือกใช้ในโครงงาน 10
3.2 แสดงคุณสมบัติของมอเตอร์ไฟฟ้ากระแสตรงที่เลือกใช้ในโครงงาน 11
3.3 แสดงวงจรที่ใช้ในการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง 11
3.4 แสดง Block Diagram ของการควบคุมความเร็วของมอเตอร์แบบวงเปิด.... 12
3.5 แสดงการกำหนดตัวแปรของการควบคุมความเร็วของมอเตอร์แบบวงเปิด.... 13
3.6 แสดงการตั้งค่า Timer ของการควบคุมความเร็วของมอเตอร์แบบวงเปิด 14
ในโปรแกรม Arduino IDE
3.7 แสดงการเก็บค่าจากเอ็นโค้ดเดอร์ของการควบคุมความเร็วของมอเตอร์แบบวงเปิด.... 15
ในโปรแกรม Arduino IDE
3.8 แสดงการเก็บค่าความเร็วของมอเตอร์ จากการควบคุมความเร็วของมอเตอร์แบบวงเปิด..... 16
ในโปรแกรม Arduino IDE
3.9 แสดงคำสั่ง Interrupt Service Routine จากการควบคุมความเร็วของมอเตอร์แบบวงเปิด ... 17
ในโปรแกรม Arduino IDE
3.10 แสดง Block Diagram ของการควบคุมความเร็วของมอเตอร์แบบวงปิด 18
3.11 แสดงการกำหนดตัวแปรของตัวควบคุมแบบพีของการควบคุมความเร็วของมอเตอร์แบบ 19
วงปิดในโปรแกรม Arduino IDE
3.12 แสดงส่วนของการทำงานของตัวควบคุมแบบพีของการควบคุมความเร็วของมอเตอร์ 20
แบบวงปิดในโปรแกรม Arduino IDE
3.13 แสดงการกำหนดตัวแปรของตัวควบคุมแบบพีไอของการควบคุมความเร็วของมอเตอร์..... 21
แบบวงปิดในโปรแกรม Arduino IDE
3.14 แสดงการคำนวณค่าความผิดพลาดของการควบคุมความเร็วของมอเตอร์แบบวงปิด..... 22
ในโปรแกรม Arduino IDE
3.15 แสดง Oscilloscope ยี่ห้อ RIGOL รุ่น DS1102E.... 23
3.16 Datasheet ของไมโครคอนโทรลเลอร์ยี่ห้อ STM32f103C8T6.... 24
3.17 แสดงโปรแกรมสร้างคาบการสุ่มในโปรแกรม Arduino IDE.... 24
3.18 แสดงคาบการสุ่มที่ได้กำหนดไว้ 25
3.19 แสดงรีจิสเทอร์ (Register) ชนิดต่างๆ ในการสร้างคาบการสุ่ม 26
3.20 แสดงฟังก์ชัน ISR ภายในโปรแกรม.... 27
3.21 แสดง Motor Drive Board ยี่ห้อ Cytron รุ่น MDD10A.... 28
3.22 แสดง Multimeter ยี่ห้อ FLUKE รุ่น 179.... 29
3.23 แสดงวงจรการควบคุมความเร็วของมอเตอร์.... 29

สารบัญรูป

รูปที่ **หน้า**

3.24 แสดงโปรแกรมการทดสอบการควบคุมความเร็วของมอเตอร์แบบวงเปิด (Open Loop Control) 30
3.25 แสดงวิธีการวัดค่าความต่างศักย์ไฟฟ้าที่ Motor Drive Board 30
3.26 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 10% 31
3.27 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 20% 31
3.28 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 30% 32
3.29 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 40% 32
3.30 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 50% 33
3.31 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 60% 33
3.32 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 70% 34
3.33 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 80% 34
3.34 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 90% 35
3.35 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 100% 35
3.36 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 10% 36
3.37 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 20% 36
3.38 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 30% 36
3.39 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 40% 37
3.40 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 50% 37
3.41 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 60% 37
3.42 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 70% 38
3.43 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 80% 38
3.44 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 90% 38
3.45 กราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 100% 39
3.46 แสดงตัวอย่างความกว้างของสัญญาณ PWM ที่ 50%, 75% และ 100% 40
3.47 แสดงคำสั่ง Import ในโปรแกรม MATLAB 42
3.48 แสดงหน้าต่างการ Import ข้อมูลเข้าโปรแกรม MATLAB 43
3.49 แสดง Workspace ที่แสดงตัวแปรที่ import เข้ามา 43
3.50 แสดงการสร้างตัวแปร input ขนาด 1 คอลัมน์จำนวน 250 ตัว 44
3.51 แสดงหน้าต่างของ System Identification Toolbox 45
3.52 แสดงการ Import ข้อมูลอินพุตและเอาต์พุต เข้าไปยัง System Identification Toolbox 45
3.53 แสดงคำสั่งในการหาฟังก์ชันถ่ายโอน (Transfer Function) ของระบบ 46
3.54 แสดงการเลือกลักษณะของฟังก์ชันถ่ายโอน (Transfer Function) ที่ต้องการสร้าง 47
3.55 แสดงค่า Gain และ Tp1 ที่ได้จากการหา Process Models 47
3.56 แสดงค่าความเข้ากันได้ของ Process Model ที่ได้จากการประมาณค่ากับความเป็นจริง 48

สารบัญรูป

รูปที่ **หน้า**

3.57 แสดงฟังก์ชันถ่ายโอน (Transfer Function) ของระบบที่ได้จากประมาณค่า 48
3.58 แสดงการสร้างไฟล์ .m ในโปรแกรม MATLAB 50
3.59 แสดง Icon ของ Simulink Toolbox ในโปรแกรม MATLAB 50
3.60 แสดงหน้าต่างของ Simulink Toolbox 51
3.61 แสดงหน้าต่างของ Library ใน Simulink Toolbox 51
3.62 แสดงการจำลองระบบควบคุมแบบปิดใน Simulink Toolbox 52
3.63 แสดงการกำหนดค่าต่างๆ ใน Step Block 52
3.64 แสดงการกำหนดค่า Sample Time ใน Zero-Order Hold Block 53
3.65 แสดงการกำหนดค่า Saturation Block ใน Saturation Block 53
3.66 แสดงการกำหนดค่าใน Transfer Function Block 54
3.67 แสดงหน้าต่าง Discrete PID Controller Block ของตัวควบคุมแบบพี 54
3.68 แสดงการใช้งาน PID Tuner Toolbox 56
3.69 แสดงค่าพารามิเตอร์ของตัวควบคุมแบบพี 56
3.70 แสดงคำสั่งของการจำลองระบบควบคุมแบบวงปิด 56
3.71 แสดงผลตอบสนองและสัญญาณควบคุมของระบบที่ใช้ตัวควบคุมแบบพี 56
3.72 แสดงหน้าต่าง Discrete PID Controller Block ของตัวควบคุมแบบพีไอ 58
3.73 แสดงหน้าต่าง PID Tuner สำหรับตัวควบคุมแบบพีไอ 59
3.74 แสดงค่าพารามิเตอร์ของตัวควบคุมแบบพีไอ 59
3.75 แสดงผลตอบสนองและสัญญาณควบคุมของระบบที่ใช้ตัวควบคุมแบบพีไอ 60
3.76 แสดงการปรับจูนค่าพารามิเตอร์ของตัวควบคุมแบบดีใน Discrete PID Controller Block 62
3.77 แสดงผลตอบสนองของระบบที่มีค่า Kd เท่ากับ 0.1 63
3.78 แสดงไฟล์ที่เก็บค่าพารามิเตอร์ต่างๆ ของการจำลอง 64
3.79 แสดงการจำลองระบบควบคุมแบบวงปิดโดยใช้เทคนิค Anti-Windup ใน Simulink Toolbox 64
3.80 แสดงโปรแกรมใน Function Block สำหรับตัวควบคุมแบบพีไอ 65
3.81 แสดงโปรแกรมใน Function Block สำหรับตัวควบคุมแบบพีไอด้วยเทคนิค Anti-Windup 66
3.82 แสดงผลตอบสนองและสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ โดยใช้ Function Block 66
3.83 แสดงผลตอบสนองและสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ โดยใช้เทคนิค Anti-Windup (Tt เท่ากับ 0.045) 67
3.84 แสดง Block Diagram ของระบบ ที่ใช้ตัวควบคุมแบบพีไอและเทคนิค Anti-Windup 68
3.85 แสดงการจำลองการควบคุมตำแหน่งเชิงมุมของมอเตอร์แบบวงปิดของตัวควบคุมแบบพี 69
3.86 แสดงการจำลองการควบคุมตำแหน่งเชิงมุมของมอเตอร์แบบวงปิดของตัวควบคุมแบบพีไอ 69

สารบัญรูป

รูปที่ **หน้า**

3.87 แสดงผลตอบสนองของระบบ สำหรับการควบคุมตำแหน่งเชิงมุมของมอเตอร์ เมื่อใช้ตัวควบคุมแบบพี 70
3.88 แสดงผลตอบสนองของระบบ สำหรับการควบคุมตำแหน่งเชิงมุมของมอเตอร์ เมื่อใช้ตัวควบคุมแบบพีไอ 70
4.1 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 10% 72
4.2 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 20% 73
4.3 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 30% 73
4.4 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 40% 74
4.5 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 50% 74
4.6 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 60% 75
4.7 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 70% 75
4.8 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 80% 76
4.9 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 90% 76
4.10 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 100% 77
4.11 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 10%..... 78
4.12 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 20%..... 79
4.13 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 30%..... 79
4.14 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 40%..... 80
4.15 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 50%..... 80
4.16 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 60%..... 81
4.17 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 70%..... 81
4.18 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 80%..... 82
4.19 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 90%..... 82
4.20 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 100% 83
4.21 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 10% 84
4.22 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 20% 84
4.23 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 30% 85
4.24 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 40% 85
4.25 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 50% 86
4.26 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 60% 86
4.27 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 70% 87
4.28 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 80% 87
4.29 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 90% 88
4.30 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 100% 88
4.31 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 10%..... 89

สารบัญรูป

รูปที่ **หน้า**

4.32 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 20%..... 89
4.33 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 30%..... 90
4.34 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 40%..... 90
4.35 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 50%..... 91
4.36 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 60%..... 91
4.37 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 70%..... 92
4.38 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 80%..... 92
4.39 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 90%..... 93
4.40 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 100%..... 93

บทที่ 1
บทนำ

1.1 ความสำคัญของปริญญานิพนธ์

วิชาการควบคุม (Control) เป็นวิชาพื้นฐานในทางวิศวกรรมที่มีบทบาทอย่างมากในด้านอุตสาหกรรม เนื่องจากการทำงานของเครื่องจักรในโรงงาน รวมไปถึงระบบรักษาความปลอดภัยต่างๆ ล้วนใช้การควบคุมเข้ามาเกี่ยวข้องเป็นส่วนใหญ่ การควบคุมจึงเป็นสิ่งที่จำเป็นอย่างมากในกระบวนการต่างๆ เพื่อต้องการให้ผลของกระบวนการนั้นออกมาอย่างมีประสิทธิภาพ ดังนั้นการเข้าใจพื้นฐานของการควบคุม นับว่าเป็นสิ่งสำคัญอย่างมากและเป็นประโยชน์ที่จะได้เพิ่มความรู้ทางด้านการควบคุม

ในโครงงานนี้ได้มีการนำเอาตัวควบคุมแบบพีและพีไอ มาใช้ในการควบคุมมอเตอร์ไฟฟ้ากระแสตรง ซึ่งไมโครคอนโทรลเลอร์ทำหน้าที่เป็นตัวควบคุมความเร็วของมอเตอร์ ดังนั้นนอกจากความรู้ทางด้านการควบคุมแล้ว จึงจำเป็นที่จะต้องมีความรู้พื้นฐานในการเขียนโปรแกรมร่วมด้วย

มอเตอร์ไฟฟ้ากระแสตรงนั้น ได้ถูกนำมาใช้งานอย่างกว้างขวางในงานอุตสาหกรรมต่างๆ เนื่องจากเป็นตัวสร้างแรงบิด ที่สามารถทำให้วัตถุที่ต้องการ เคลื่อนที่ไปข้างหน้าได้ เช่น ล้อรถยนต์ เป็นต้น ซึ่งมอเตอร์แต่ละตัวนั้นมีประสิทธิภาพไม่เท่ากัน ถึงแม้ว่าจะเป็นรุ่นเดียวกันก็ตาม ดังนั้นจึงได้มีการนำตัวควบคุมแบบพีและพีไอมาใช้งานเพื่อควบคุมความเร็วของมอเตอร์ให้ได้ค่าตามที่ต้องการ ซึ่งการใช้งานตัวควบคุมนั้นจะเป็นการสร้างแบบจำลองผลลัพธ์ในโปรแกรม MATLAB หลังจากนั้นจะนำมาประยุกต์สร้างเป็นโค้ดในโปรแกรม Arduino IDE เพื่ออัปโหลดไปยังไมโครคอนโทรลเลอร์

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อเรียนรู้การวิเคราะห์ระบบ (System Identification) ในการหาแบบจำลองทางคณิตศาสตร์ของมอเตอร์ไฟฟ้ากระแสตรง (DC Motor)
2. เพื่อเรียนรู้การควบคุมแบบ Proportional-Integral-Derivative (PID) ในการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง (DC Motor)
3. เพื่อให้ได้ระบบของมอเตอร์ไฟฟ้ากระแสตรง (DC Motor) ที่สามารถควบคุมความเร็วของมอเตอร์ไฟฟ้าให้ได้ค่าที่ต้องการ
4. เพื่อศึกษากระบวนการทำวิจัย, กระบวนการค้นคว้าหาข้อมูลหรือบทความที่เกี่ยวข้อง และกระบวนการสร้างการทดลองแบบต่างๆ ที่สามารถนำไปใช้ประโยชน์กับโครงงานได้

1.3 ขอบเขตของปริญญานิพนธ์

1. ศึกษาการออกแบบตัวควบคุมแบบพี/พีไอ (P/PI Controller) , การสร้างแบบจำลองทางคณิตศาสตร์ (Mathematical Model) และการควบคุมระบบทางพลศาสตร์ (Dynamic)
2. การควบคุมความเร็ว (Speed) ของมอเตอร์ไฟฟ้ากระแสตรง (DC Motor)

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้เรียนรู้ทักษะการใช้งานของโปรแกรม MATLAB และโปรแกรม Arduino IDE มากยิ่งขึ้น
2. ได้เรียนรู้วิธีการวิเคราะห์ระบบ (System Identification) เพื่อหาแบบจำลองทางคณิตศาสตร์ของมอเตอร์ไฟฟ้ากระแสตรง (DC Motor)
3. ได้เรียนรู้วิธีการควบคุมแบบพีไอ (Proportional-Integral) เพื่อควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง (DC Motor)
4. ได้ระบบของมอเตอร์ไฟฟ้ากระแสตรง (DC Motor) ที่มีความเร็วตรงตามที่ต้องการและสามารถเพิ่มประสิทธิภาพ (Performance) ของมอเตอร์ได้
5. ได้เรียนรู้กระบวนการทำวิจัย, กระบวนการค้นคว้าหาข้อมูลหรือบทความที่เกี่ยวข้องและกระบวนการสร้างการทดลองแบบต่างๆ ที่สามารถนำไปใช้ประโยชน์กับโครงงานได้

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 มอเตอร์ไฟฟ้ากระแสตรง (DC motor)

2.1.1 ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรงมีส่วนประกอบสำคัญ 2 ส่วน คือ สเตเตอร์ (Stator) หรือส่วนที่อยู่กับที่ และโรเตอร์ (Rotor) หรือส่วนที่หมุน

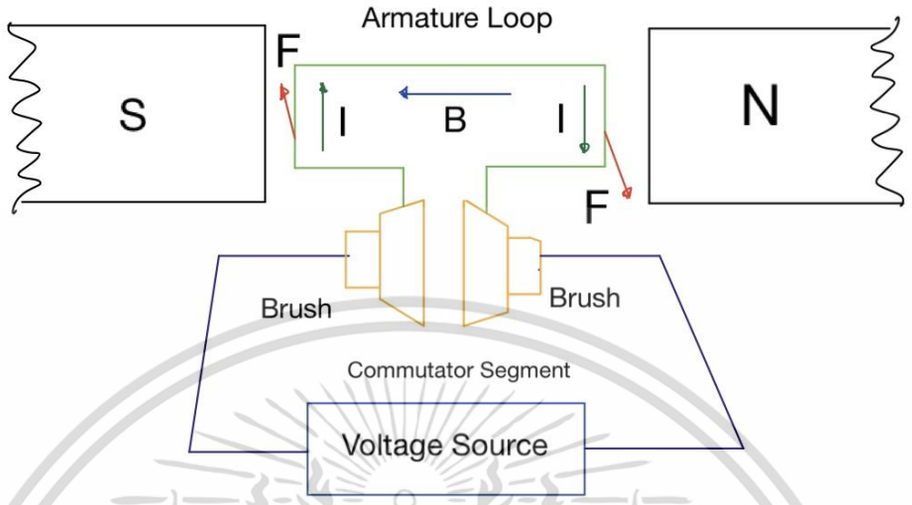
2.1.1.1 สเตเตอร์ (Stator) ประกอบด้วย 4 ส่วน คือ

1) เฟรม (Frame) เป็นโครงสร้างภายนอกทำหน้าที่ ยึดส่วนประกอบอื่นๆ ภายในเข้าด้วยกัน ส่วนใหญ่มีลักษณะเป็นทรงกระบอกทำจากเหล็ก
2) ขั้วแม่เหล็ก (Pole)
3) แกนขั้ว (Pole Core) ทำจากแผ่นเหล็กบางๆ มีฉนวนกั้น เป็นแท่งยึดติดกับเฟรม ทำหน้าที่สร้างเส้นแรงแม่เหล็กผ่านไปยังโรเตอร์ทำให้เกิดแรงบิด ส่งผลให้มอเตอร์เกิดการหมุน
4) ขดลวดสนามแม่เหล็ก (Field Coil) จะพันอยู่รอบๆ แกนขั้วแม่เหล็กทำหน้าที่รับกระแสจากภายนอกเพื่อสร้างเส้นแรงแม่เหล็ก เกิดการหักล้างและเสริมกับสนามแม่เหล็กของอาเมเจอร์ทำให้เกิดแรงบิด

2.1.1.2 โรเตอร์ (Rotor) ประกอบด้วย 4 ส่วนคือ

1) แกนเพลา (Shaft) เป็นตัวที่ยึดคอมมิวเตเตอร์ และยึดแกนเหล็กอาร์เมเจอร์ (Armature Core) โดยจะยึดอยู่บนแบริ่ง (Bearing) เพื่อยึดในหมุนอยู่ในแนว
2) แกนเหล็กอาร์เมเจอร์ (Armature Core) ทำจากแผ่นเหล็กบางอาบฉนวนทำหน้าที่ให้ขดลวดอาร์เมเจอร์มาพันไว้
3) คอมมิวเตเตอร์ (Commutator) ทำจากทองแดงมีลักษณะเป็นซี่แต่ละซี่มีฉนวนคั่น มีหน้าที่สัมผัสกับแปรงถ่าน (Carbon Brushes) เพื่อรับกระแสป้อนเข้าไปยังขดลวดอาร์เมเจอร์
4) ขดลวดอาร์เมเจอร์ (Armature Winding) เป็นขดลวดพันอยู่ในร่องของแกนอาร์เมเจอร์ โดยขนาดของลวดและจำนวนรอบจะขึ้นอยู่กับการออกแบบ

2.1.2 หลักการทำงานของมอเตอร์ไฟฟ้ากระแสตรง



รูปที่ 2.1 แสดงการทำงานของมอเตอร์ไฟฟ้ากระแสตรง

จากรูปที่ 2.1 เมื่อจ่ายไฟให้แก่มอเตอร์ผ่านทางแปรงสัมผัสซึ่งต่ออยู่กับคอมมิวเตเตอร์และขดลวด เมื่อกระแสไฟฟ้าไหลผ่านขดลวด จะทำให้เกิดสนามแม่เหล็กขึ้นทำให้เกิดแรงในทิศตั้งฉากกับ สนามแม่เหล็กทำให้ขดลวดสามารถหมุนได้ส่งผลให้แกนของมอเตอร์หมุนตามไปด้วย ซึ่งหากต้องการกลับทิศทางของการหมุน ให้สลับขั้วบวกและขั้วลบของแหล่งจ่ายไฟจะทำให้เกิดแรงหมุนไปในทิศทาง ตรงข้าม

2.1.3 การควบคุมมอเตอร์ไฟฟ้ากระแสตรง

สำหรับการควบคุมมอเตอร์ไฟฟ้ากระแสตรงนั้น จะใช้ความต่างศักย์ในการควบคุมเป็นหลักหากมอเตอร์ตัวนั้นมีความต่างศักย์ใช้งานเท่ากับ 12 โวลต์ เมื่อเราป้อนความต่างศักย์ขนาด 12 โวลต์ เข้าไป มอเตอร์จะหมุนด้วยความเร็วสูงสุด หากป้อนความต่างศักย์เท่ากับ 6 โวลต์ จะทำให้มอเตอร์หมุนที่ความเร็ว 50% การควบคุมมอเตอร์ส่วนใหญ่จะใช้หลักการนี้ หากต้องการกลับทิศการหมุนของมอเตอร์เพียงสลับขั้วไฟฟ้าที่ป้อนให้กับมอเตอร์มอเตอร์จะหมุนในทิศทางตรงกันข้าม

2.2 เอ็นโค้ดเดอร์แบบแกนหมุน (Rotary Encoder)

เอ็นโค้ดเดอร์ (Encoder) คือ อุปกรณ์ที่วัดการเคลื่อนที่ของวัตถุ มี 2 ชนิด คือ แบบเส้นตรง (Linear Encoder) และแบบแกนหมุน (Rotary Encoder) ซึ่งหลักการทำงานจะคล้ายคลึงกันเพียงแต่เป็นการวัดการเคลื่อนที่เชิงเส้นและเชิงมุมเท่านั้น

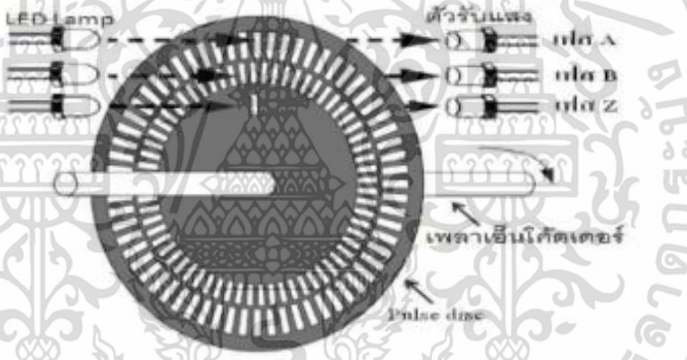
เอ็นโค้ดเดอร์แบบแกนหมุน (Rotary Encoder) จะมีลักษณะเป็นจานหมุนที่มีร่องเล็กในลักษณะทึบแสงและโปร่งแสงเพื่อเป็นตัวบอกตำแหน่งการเคลื่อนที่

2.2.1 ส่วนประกอบของเอ็นโค้ดเดอร์แบบแกนหมุน

เอ็นโค้ดเดอร์แบบแกนหมุนประกอบไปด้วยส่วนหลักๆ 4 ส่วน ดังนี้

1) เพลา (Shaft) คือ ส่วนของแกนที่ต้องรองรับจานหมุน โดยอาจจะต่อมาจากมอเตอร์หรือตัวที่ทำให้แกนหมุน โดยเพลาจะมีหลากหลายขนาดขึ้นอยู่กับสิ่งที่นำไปใช้งาน
2) แผ่นดิสก์ (Disc) มีลักษณะเป็นจานวงกลม โดยที่จะมีการทำร่องไว้ในลักษณะแบบทึบแสงและโปร่งแสง โดยร่องอาจจะมีมากกว่า 1 แถว เพื่อเพิ่มความละเอียดของตำแหน่งเชิงมุม
3) แหล่งกำเนิดแสง (Source) คือ แหล่งที่สร้างแสง เพื่อส่งผ่านแสงไปยังจานหมุน เพื่อให้ตัวตรวจจับสามารถวัดค่าของแสงได้ โดยทั่วไปอาจจะใช้ LED ขนาดเล็กในวงจรที่มีขนาดเล็กและเพิ่มขนาดของ LED ตามขนาดของแกนหมุน
4) ตัวรับแสง (Photodetector) คือ ตัวตรวจจับที่ทำหน้าที่แปลงความเข้มแสงที่ส่งมาจากแหล่งกำเนิดผ่านแผ่นดิสก์แล้วส่งไปยังหน่วยประมวลผล เพื่อแปลงเป็นตำแหน่งเชิงมุมต่อไป

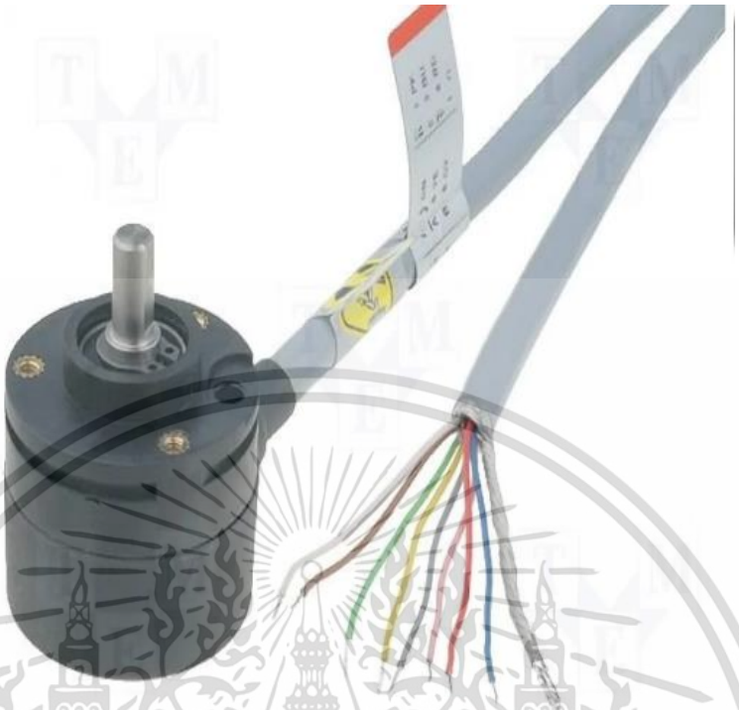
2.2.2 การทำงานของเอ็นโค้ดเดอร์แบบแกนหมุน



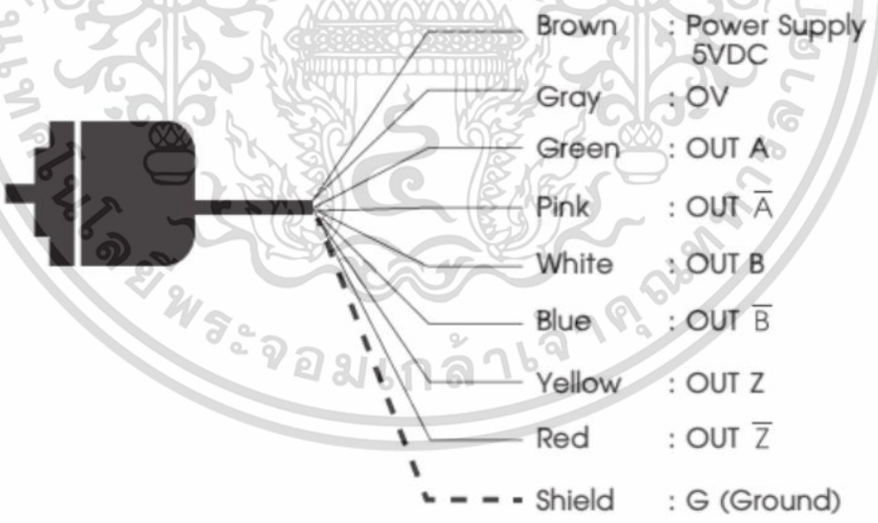
รูปที่ 2.2 แสดงการทำงานของเอนโค้ดเดอร์แบบแกนหมุน

จากรูปที่ 2.2 แหล่งกำเนิดแสง คือ หลอด LED และตัวรับแสงส่วนใหญ่จะมีด้วยกัน 2 เฟส แต่จะมีแบบพิเศษ ซึ่งจะมี 3 เฟส เพื่อนับรอบและเพิ่มความแม่นยำด้วย โดยเมื่อเพลามีการหมุนแผ่นดิสก์ก็จะหมุนตาม และแสงที่ปล่อยออกมาจากแหล่งกำเนิดแสงจะผ่านร่องของแผ่นดิสก์ เกิดเป็นสัญญาณที่เป็นดิจิตอล เช่น ทึบแสงคือ logic 0 และ โปร่งแสงคือ logic 1 เป็นต้น โดยที่ช่องสัญญาณของตัวตรวจจับช่องแรกจะวางเหลื่อมกับช่องที่สอง เพื่อตรวจจับทิศทางการหมุนด้วย และเพื่อบ่งบอกว่าแกนหมุนไปในทิศทางใด

2.2.3 ตัวอย่างของเอ็นโค้ดเดอร์แบบแกนหมุน



รูปที่ 2.3 แสดงเอ็นโค้ดเดอร์แบบแกนหมุน ยี่ห้อ Kubler รุ่น 3700



รูปที่ 2.4 แสดงหน้าที่ของเส้นแต่ละเส้นที่ต่อจากเอ็นโค้ดเดอร์

จากรูปที่ 2.4 เส้นของสายไฟที่ต่อออกจากเอ็นโค้ดเดอร์ ยี่ห้อ Kubler นั้นมีหลายเส้น เนื่องจากมีช่องสัญญาณของตัวตรวจจับ 3 ช่อง และแต่ละช่องมีการทำการกลับเฟส (Inverted) เอาไว้ด้วย อีกทั้งยังมีการต่อสายดินและต่อสายจากแหล่งพลังงานด้วย เอ็นโค้ดเดอร์จึงจะสามารถทำงานได้

2.3 การควบคุมแบบ PID (Proportional Integrated Controller)

ตัวควบคุมแบบ PID เป็นตัวควบคุมที่มีประสิทธิภาพสูง ซึ่งตัวควบคุมแบบ PID สามารถแก้ปัญหาได้มากกว่าร้อยละ 90 ในการควบคุมระบบต่างๆ ซึ่งความหมายของแต่ละตัวคือ P (Proportional) หมายถึง การปรับสัดส่วนสัญญาณ, I (Integral) หมายถึง การอินทิเกรตสัญญาณ และ D (Derivative) หมายถึง การอนุพันธ์สัญญาณ โดยกลไกทั้งสามนี้จะถูกรวมเข้าด้วยกัน เพื่อทำหน้าที่ปรับแต่งสัญญาณให้มีความเหมาะสม ซึ่งตัวควบคุมแบบ PID เปรียบเสมือนกับตัวชดเชยแบบเฟสล้ำหน้าต่อผสมแบบอนุกรมอยู่กับตัวชดเชยแบบเฟสล้าหลัง

PID Control เป็นการควบคุมในระบบวงปิด คือ ระบบควบคุมที่มีการป้อนกลับ ประกอบด้วยส่วนการควบคุมที่สำคัญดังนี้

1) Proportional Control Action (P-Action) เป็นการกำหนดการทำงานของเอาต์พุตให้เป็นสัดส่วนร้อยละกับค่าความผิดผลาด (Error)
2) Integral Control Action (I-Action) เป็นสัดส่วนของความผิดพลาดและระยะเวลาของความผิดพลาด (Error)
3) Derivative Control Action (D-Action) เป็นอัตราการเปลี่ยนแปลงของความผิดพลาดโดยการรบกวนระบบจากภายนอก ซึ่งจะทำงานตรงข้ามกับ Integral Control Action

โดยนิยมนำมาใช้ในการควบคุมอุณหภูมิ โดยสามารถแก้ปัญหา การเกิด Offset Error ที่สถานะคงตัวของระบบได้ โดยสามารถหาค่าตัวแปรของ PID ได้จากสมการ

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

สมการ 2.1 แสดงสมการสัญญาณเอาต์พุตของตัวควบคุมแบบ PID

เมื่อ $u(t)$ = สัญญาณเอาต์พุตของตัวควบคุม ที่เวลา t
K_p = ค่า Proportional Grain
K_i = ค่า Integral Gain
K_d = ค่า Derivative Gain
e = ค่าความผิดพลาดที่เกิดจาก set point value – process value
t = ค่าเวลาในขณะนั้น
τ = ผลรวมของตัวแปรค่าความผิดพลาดตั้งแต่เวลา 0 ถึง t

2.3.1 ตัวควบคุมแบบพี (P-Controller)

เป็นสัญญาณควบคุม โดยจะเป็นสัดส่วนโดยตรงกับค่าสัญญาณความผิดพลาดที่เกิดจากผลต่างระหว่างค่าสัญญาณอ้างอิงกับค่าสัญญาณเอาต์พุตของระบบที่ต้องการควบคุม โดยจุดประสงค์คือ การเพิ่มความเร็วในการตอบสนองของระบบ และลดค่าความผิดพลาดที่สภาวะอยู่ตัวของระบบ ซึ่งส่วนใหญ่จะเหมาะกับระบบที่ไม่ค่อยมีความสำคัญมาก เพียงแต่จะต้องการเร่งให้ระบบมีประสิทธิภาพมากขึ้นเท่านั้น

2.3.2 ตัวควบคุมแบบไอ (I-Controller)

สัญญาณควบคุมจะมีค่ามาก โดยที่สัญญาณความผิดพลาดมีค่าเป็นศูนย์ ในภายหลังเมื่อเวลาผ่านไปแล้วก็ตาม ทั้งนี้ก็เพราะว่าสัญญาณการควบคุมในกรณีของการควบคุมแบบอินทิกรัลขึ้นอยู่กับค่าก่อนหน้า (Past Value) ไม่เหมือนกับตัวควบคุมแบบสัดส่วน ซึ่งจะขึ้นอยู่กับค่าปัจจุบัน ประโยชน์ของตัวควบคุมแบบอินทิกรัล คือ เพื่อลดค่าความผิดพลาดที่สถานะอยู่ตัวของระบบให้น้อยลงจนหมดไป แต่จะทำให้ความเร็วในการตอบสนองและเสถียรภาพของระบบลดลงกล่าวคือลดค่า Overshoot ของระบบนั้นเอง

2.3.3 ตัวควบคุมแบบดี (D-Controller)

การควบคุมแบบอนุพันธ์ส่วนมากแล้วจะใช้ร่วมกับตัวควบคุมตัวอื่นด้วย หากใช้ตัวควบคุมแบบสัดส่วนรวมกับตัวควบคุมแบบอนุพันธ์ ตัวควบคุมแบบอนุพันธ์นี้จะช่วยเพิ่มค่าความหน่วง (Damping) ให้กับระบบที่ต้องการจะควบคุม นั่นคือทำให้ระบบมีเสถียรภาพมากขึ้นและเพิ่มค่าอัตราส่วนความหน่วงให้กับระบบ

2.4 การหาลักษณะเฉพาะของระบบ (System Identification)

การหาลักษณะเฉพาะของระบบ (System Identification) คือวิธีที่ใช้ในการสร้างแบบจำลองทางคณิตศาสตร์ของระบบพลศาสตร์ (Dynamics System) โดยใช้การวัดอินพุตและเอาต์พุตของระบบ โดยขั้นตอนในการหาลักษณะเฉพาะของระบบ ประกอบไปด้วยขั้นตอนหลักๆ ดังนี้

1) วัดค่าอินพุตและเอาต์พุตของระบบในลักษณะของ Time Domain หรือ Frequency Domain
2) เลือกระบบที่เราต้องการจะหาลักษณะเฉพาะของระบบ
3) ใช้วิธีการประมาณค่าเพื่อประมาณค่าพารามิเตอร์ต่างๆ ของระบบ
4) หาแบบจำลองของระบบที่ได้จากการประมาณค่า เพื่อนำไปใช้ต่อไป

2.4.1 ข้อมูลที่ใช้ในการหาลักษณะเฉพาะของระบบ

การสร้างลักษณะเฉพาะของระบบจำเป็นต้องใช้ค่าของอินพุตและเอาต์พุตที่ได้จากระบบเพื่อประมาณค่าของพารามิเตอร์ที่สามารถปรับได้ หากข้อมูลที่ทำการวัดดีมากเท่าไหร่จะทำให้แบบจำลองที่ได้มาสะท้อนพฤติกรรมของระบบได้ดีมากเท่านั้น ซึ่งข้อมูลจะมี 2 ส่วนคือ Time Domain Data และ Frequency Domain Data

โดยข้อมูลที่มีลักษณะเป็น Time-Domain นั้นประกอบไปด้วย ค่าอินพุตและเอาต์พุตของระบบที่ได้จากการสุ่มในช่วงเวลาหนึ่งๆ (Sampling Time) สำหรับข้อมูลที่มีลักษณะเป็น Frequency-Domain นั้นจะเป็นค่าของอินพุตและเอาต์พุต ส่วนใหญ่จะอยู่ในรูปของ Time-Domain ที่ทำการ Fourier Transform แล้ว

2.4.2 System Identification Toolbox

ในการหาลักษณะเฉพาะของระบบนั้น ในอดีตอาจจะใช้สมการแล้วทำการเก็บข้อมูลจากการทดลอง จากนั้นนำมาคำนวณแล้วใส่ลงในสมการ เพื่อหาแบบจำลองทางคณิตศาสตร์ของระบบ

แต่ในปัจจุบันนั้น ได้มีการพัฒนาเครื่องมือที่มีความสามารถในการหาแบบจำลองทางคณิตศาสตร์ของระบบได้อย่างง่ายดาย เช่น System Identification Toolbox ในโปรแกรม MATLAB ซึ่งการใช้งานสามารถทำได้โดยง่ายเพียงแค่มีข้อมูลอินพุต, เอาต์พุต และคาบการสุ่ม (Sampling Time) ของระบบ จึงมีการใช้งาน Toolbox ตัวนี้อย่างกว้างขวาง เนื่องจากสะดวกและมีความแม่นยำสามารถประมวลผลข้อมูลจำนวนมากได้ในเวลาไม่นานมากนัก

2.5 MATLAB

MATLAB เป็นภาษาคอมพิวเตอร์ระดับสูงที่ใช้สำหรับคำนวณเชิงตัวเลข (Numerical Computing) แสดงผลกราฟฟิกและเขียนแอปพลิเคชัน ทำให้เราสามารถคำนวณผลลัพธ์ และสามารถพัฒนาอัลกอริทึม สร้างแบบจำลอง และแอปพลิเคชันได้ง่ายและรวดเร็ว ภายในตัวโปรแกรม MATLAB ประกอบด้วยภาษาคอมพิวเตอร์, Toolbox, กลุ่มฟังก์ชันสำเร็จรูปในแต่ละสาขาวิชา และฟังก์ชันพื้นฐานจำนวนมาก ทำให้การวิเคราะห์สามารถทำได้หลากหลายวิธี พร้อมกับได้คำตอบที่รวดเร็ว

MATLAB สามารถนำไปประยุกต์ใช้งานได้หลากหลายสาขา ทั้งการประมวลผลสัญญาณ (Signal Processing), การสื่อสาร (Communication), การประมวลผลภาพและวิดีโอ (Image and Video Processing), ระบบควบคุม (Control System), การวัดและควบคุม (Instruments and Control) เป็นต้น

2.5.1 หลักการทำงานของ MATLAB

MATLAB สามารถทำงานได้ทั้งในลักษณะของการติดต่อโดยตรง คือการเขียนคำสั่งเข้าไปทีละคำสั่ง เพื่อให้โปรแกรมประมวลผลไปเรื่อยๆ ข้อสำคัญอย่างหนึ่งของ MATLAB คือ ข้อมูลทุกตัวจะถูกเก็บในลักษณะของแถวลำดับ กล่าวคือ ในแต่ละตัวแปรจะได้รับการแบ่งเป็นส่วนย่อยเล็กๆ ขึ้น ซึ่งการใช้ตัวแปรเป็นแถวลำดับ ใน MATLAB ไม่จำเป็นที่จะต้องจองมิติเหมือนกับการเขียนโปรแกรมในภาษาขั้นต่ำทั่วไป ซึ่งทำให้สามารถที่จะแก้ปัญหาของตัวแปรที่อยู่ในลักษณะของเมทริกซ์ (Matrix) และเวกเตอร์ (Vector) ได้โดยง่าย ซึ่งทำให้สามารถประหยัดเวลาในการทำงานได้อย่างมาก

2.5.2 ความสามารถของโปรแกรม MATLAB ในด้านวิศวกรรม

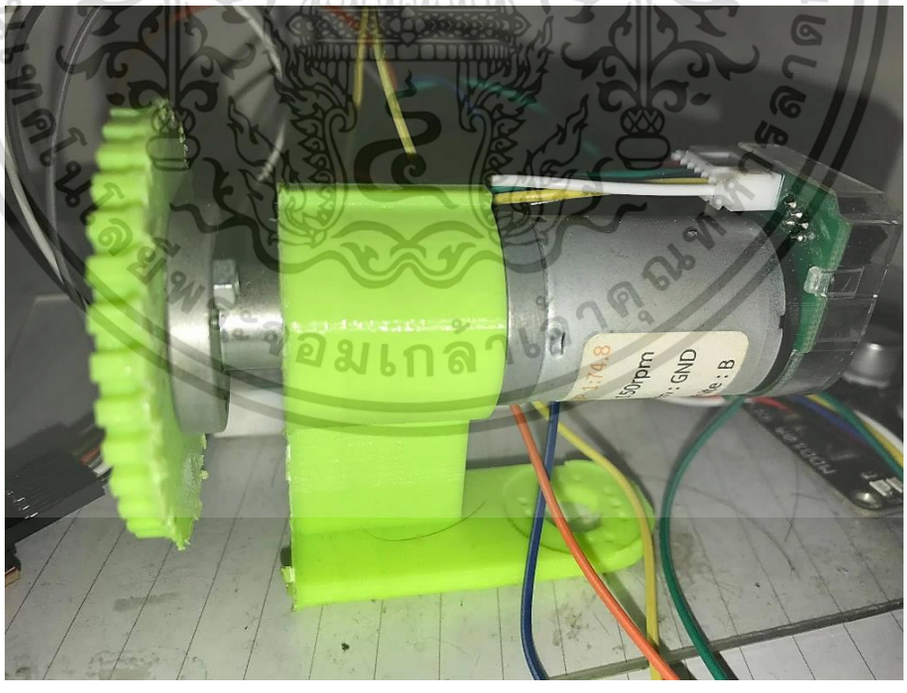
1) MATLAB เป็นโปรแกรมเพื่อการคำนวณและสามารถแสดงผลได้ทั้งตัวเลขและรูปภาพ
2) MATLAB จะสามารถควบคุมการทำงานด้วยชุดคำสั่งและยังสามารถรวบรวมชุดคำสั่งเป็นโปรแกรมได้
3) ลักษณะการเขียนโปรแกรมใน MATLAB จะใกล้เคียงการเขียนสมการทางคณิตศาสตร์ จึงง่ายกว่าการเขียนโปรแกรมโดยใช้ภาษาชั้นสูง
4) MATLAB มี Toolbox หรือชุดฟังก์ชันพิเศษสำหรับผู้ใช้ที่ต้องการใช้งานเฉพาะทางหรืองานด้านวิศวกรรมขั้นสูงอื่นๆ

บทที่ 3
ขั้นตอนและวิธีการดำเนินงาน

ในขั้นตอนการทำโครงงานนี้ ได้แบ่งการทำงานออกเป็น 2 ส่วนหลักๆ คือ ส่วนของการจำลอง (Simulation) และส่วนของการทำการทดลองจริงบนมอเตอร์ไฟฟ้ากระแสตรง (DC Motor) เนื่องจากวิชาควบคุม (Control) ต้องมีความรู้พื้นฐานจึงจะสามารถนำไปประยุกต์ใช้งานจริงได้ โครงงานนี้จึงได้ทำการสร้างการทดลองและการจำลองในโปรแกรม MATLAB เพื่อศึกษาการทำงานของตัวควบคุมชนิดต่างๆ, อิทธิพลของตัวควบคุมแต่ละตัว และการใช้งานโปรแกรม MATLAB ก่อนที่จะนำไปใช้งานจริงในการควบคุมมอเตอร์ไฟฟ้ากระแสตรง

3.1 การศึกษาตัวควบคุมในการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง

ในการควบคุมมอเตอร์ไฟฟ้ากระแสตรงนั้น ขั้นตอนแรกของการทดลอง คือ การเลือกมอเตอร์ไฟฟ้ากระแสตรงที่จะนำมาใช้ในการทดลอง ซึ่งได้มีการเลือกใช้มอเตอร์ไฟฟ้ากระแสตรงขนาด 12 โวลต์ เส้นผ่านศูนย์กลาง 25 มิลลิเมตร และมีเอ็นโค้ดเดอร์ (Encoder) ติดมาที่ด้านท้ายของตัวมอเตอร์ด้วย เนื่องจากในการควบคุมนั้น จะต้องอาศัยเอาต์พุตของระบบ ซึ่งในการควบคุมความเร็วของมอเตอร์นั้น เอาต์พุต คือ ความเร็วของมอเตอร์ ซึ่งการที่จะหาความเร็วของมอเตอร์ได้ จะต้องมีตัวที่วัดค่าด้วย ซึ่งเอ็นโค้ดเดอร์สามารถทำการวัดค่าได้ โดยลักษณะและคุณสมบัติของมอเตอร์ไฟฟ้ากระแสตรงที่เลือกใช้ แสดงดังรูปที่ 3.1 และ รูปที่ 3.2 ตามลำดับ



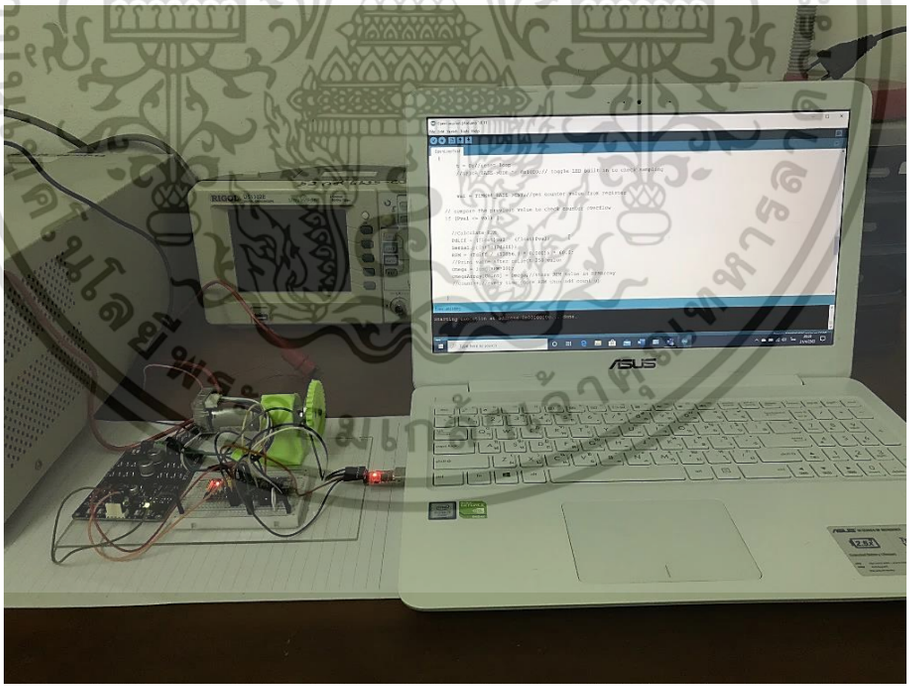
รูปที่ 3.1 แสดงมอเตอร์ไฟฟ้ากระแสตรงที่เลือกใช้ในโครงงาน

Diameter (mm)	25
Working Voltage (VDC)	12
Gear ratio	1:74.8
Max speed (RPM)	150
Encoder Resolution (CPR)	360 (2 channel)
Encoder Type	Optical
Red wire	M+
Black wire	M-
Yellow wire	Encoder A
White wire	Encoder B
Blue wire	VCC
Greed wire	GND

รูปที่ 3.2 แสดงคุณสมบัติของมอเตอร์ไฟฟ้ากระแสตรงที่เลือกใช้ในโครงงาน

3.1.1 การควบคุมความเร็วของมอเตอร์แบบวงเปิด (Open Loop Control)

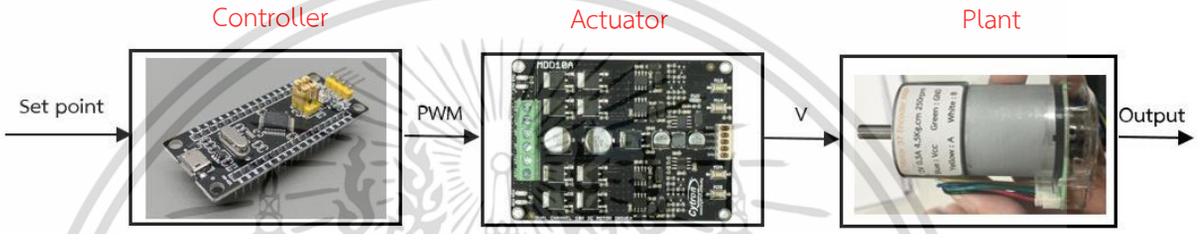
หลังจากการเลือกมอเตอร์ไฟฟ้ากระแสตรงที่ใช้ในการทำโครงงานแล้ว ทำการสร้างการวงจรการควบคุมมอเตอร์แบบวงเปิด เพื่อดูลักษณะและวิธีการควบคุมมอเตอร์ ซึ่งประกอบไปด้วยไมโครคอนโทรลเลอร์รุ่น STM32f103C8T6, บอร์ดขับมอเตอร์รุ่น Cytron MDD10A และตัวมอเตอร์ไฟฟ้ากระแสตรงที่ได้เลือกไว้ โดยมีการต่อวงจรจริง ดังรูปที่ 3.3



รูปที่ 3.3 แสดงวงจรที่ใช้ในการควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรง

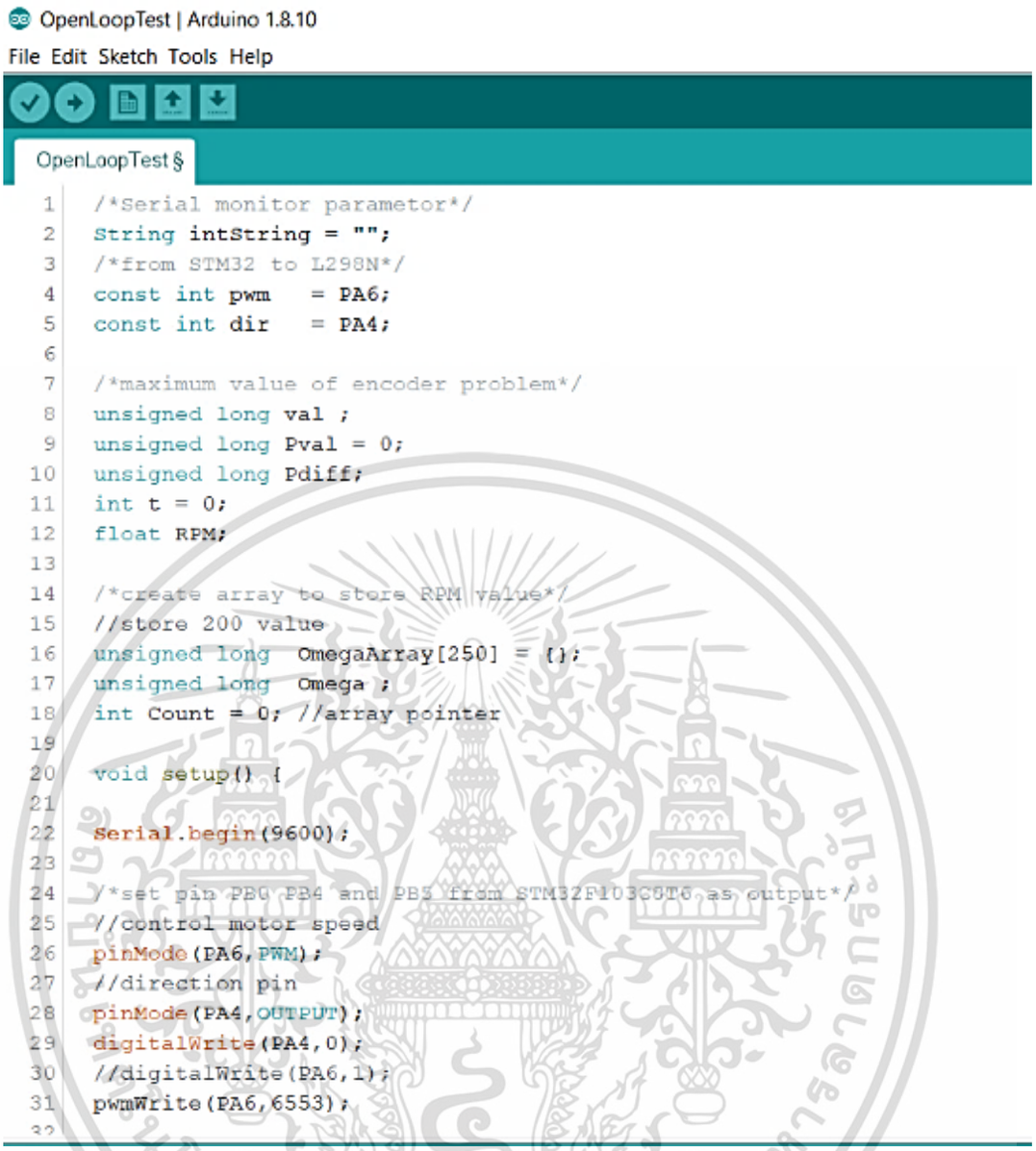
การทดลองประกอบด้วยชุดการควบคุมมอเตอร์, คอมพิวเตอร์ที่ติดตั้งโปรแกรม Arduino IDE และ Power Supply โดย Power Supply จะทำหน้าที่ป้อนกระแสไฟฟ้าไปยังบอร์ดขับมอเตอร์ เพื่อให้มีกระแสที่คงที่ตลอดเวลาในการทดลอง เนื่องจากการควบคุมมอเตอร์นั้นต้องการกระแสไฟฟ้าที่เสถียรและมีค่าคงที่ หากใช้เป็นแหล่งจ่ายไฟประเภทแบตเตอรี่อาจจะทำให้ค่าที่ออกมามีความผิดพลาดได้

สำหรับการควบคุมมอเตอร์แบบวงเปิดเป็นการควบคุม ที่ตัวควบคุมจะส่งสัญญาณไปยังมอเตอร์ เพื่อให้ได้ความเร็วที่ต้องการ และจะไม่มีผลป้อนกลับ (Feedback) จากนั้นระบบจะบอกว่าสัญญาณที่ส่งไปควบคุมนั้นมีความถูกต้องหรือไม่ และมีเพียงค่าเอาต์พุตที่แสดงออกมาเท่านั้น Block Diagram ของการควบคุมแบบวงเปิด แสดงดังรูปที่ 3.4



รูปที่ 3.4 แสดง Block Diagram ของการควบคุมความเร็วของมอเตอร์แบบวงเปิด

จากรูปที่ 3.4 ไมโครคอนโทรลเลอร์ทำหน้าที่เป็นตัวควบคุม เพื่อส่งสัญญาณไปยังบอร์ดขับมอเตอร์เพื่อแปลงสัญญาณ PWM (Pulse Width Modulation) ให้กลายเป็นความต่างศักย์เพื่อส่งไปยังมอเตอร์ให้ได้ความเร็วที่ต้องการ โดยจะทำการวัดความเร็วออกมาโดยสร้างโปรแกรมเพื่อคำนวณความเร็วในโปรแกรม Arduino IDE โดยการควบคุมความเร็วของมอเตอร์แบบวงเปิด จะเป็นการหาค่าความเร็วของมอเตอร์ในแต่ละค่าของ PWM กล่าวคือ เป็นการสั่งให้มอเตอร์มีความเร็วตามที่ต้องการ ซึ่งทำได้โดยการเปลี่ยนค่าของ PWM ในโปรแกรม Arduino IDE



รูปที่ 3.5 แสดงการกำหนดตัวแปรของการควบคุมความเร็วของมอเตอร์แบบวงเปิด ในโปรแกรม Arduino IDE

จากรูปที่ 3.5 เป็นการกำหนดตัวแปรต่างๆ ในโปรแกรม Arduino IDE ประกอบไปด้วย การกำหนดขาที่ใช้ในการสั่งงานบอร์ดขับมอเตอร์ ทั้งความเร็วและทิศทางการหมุน, การกำหนดค่าตัวแปรที่ใช้เก็บค่าจากเอ็นโค้ดเดอร์, การกำหนดตัวแปรที่ใช้ในการเก็บค่าของความเร็วของมอเตอร์ และกำหนด MODE ของขาต่างๆ ของไมโครคอนโทรลเลอร์

รูปที่ 3.6 แสดงการตั้งค่า Timer ของการควบคุมความเร็วของมอเตอร์แบบวงเปิด ในโปรแกรม Arduino IDE

จากรูปที่ 3.6 เป็นการสร้างคาบการสุ่ม (Sampling Time) เพื่อเก็บค่าความเร็วของมอเตอร์โดยจะใช้การ interrupt ของ Timer ในหน่วยประมวลผลของไมโครคอนโทรลเลอร์ในการสร้างคาบเวลา สำหรับการตั้งค่า, ความหมายของรีจิสเตอร์ และวิธีการคำนวณคาบเวลา สามารถดูได้จากการทดลองที่ 3.2.1


OpenLoopTest | Arduino 1.8.10

File Edit Sketch Tools Help

OpenLoopTest §

```
60        //GPIOB_BASE->ODR ^= 0x1000;// toggle LED built in to check sampling
61
62
63        val = TIMER4_BASE->CNT;//get counter value from register
64
65    // compare the previous value to check counter overflow
66    if (Pval <= val) {
67
68      //Calculate RPM
69      Pdiff = (float)val - (float)Pval;
70      Serial.println(Pdiff);
71      RPM = (Pdiff / (53856.0 * 0.001)) * 60.0;
72      speed_actual = RPM;
73      //Print value after collect 250 value
74      Omega = long(RPM*10);
75      OmegaArray[Count] = Omega;//store RPM value in RPMArray
76      Count++;//every time store RPM then add count up
77
78    }
79    else
80      {
81
82      Pdiff = (65536.0 - (float)Pval) + (float)val ;
83      Serial.println(Pdiff);
84      // gear ratio and encoder PPR
85      RPM = (Pdiff / (53856.0 * 0.001)) * 60.0;//Calculate RPM value every 1 * 10 = 1 mS
86      speed_actual = RPM;
87      Omega = long(RPM*10);
88      OmegaArray[Count] = Omega;//store RPM value in RPMArray
89      Count++;//every time store RPM then add count up
90        }
```

รูปที่ 3.7 แสดงการเก็บค่าจากเอ็นโค้ดเดอร์ของการควบคุมความเร็วของมอเตอร์แบบวงเปิด ในโปรแกรม Arduino IDE

จากรูปที่ 3.7 เป็นการกำหนดเงื่อนไขเพื่อเก็บค่าจากเอ็นโค้ดเดอร์ เพื่อนำมาคำนวณค่าความเร็วของมอเตอร์โดยใช้ Timer ที่ทำหน้าที่เหมือน Counter เพื่อใช้ในการนับสัญญาณพัลส์ที่ปรากฏในช่วงของคาบการสุ่ม แต่ Counter มีข้อจำกัด คือ สามารถรับค่าได้มากที่สุดที่ 65536 หรือ 2^{16} ตามขนาดที่ขาของไมโครคอนโทรลเลอร์ ซึ่งการคำนวณความเร็วจะใช้การนับความแตกต่างของสัญญาณพัลส์ในช่วงที่พัลส์ก่อนหน้าและพัลส์ปัจจุบัน ในช่วงที่ตัวนับมีการรีเซตค่ากลับมาเป็น 0 จะทำให้ความแตกต่างของพัลส์มีความผิดพลาดได้ จึงต้องมีการสร้างเงื่อนไขเพื่อตรวจจับเมื่อ Counter มีการรีเซตค่า

รูปที่ 3.8 แสดงการเก็บค่าความเร็วของมอเตอร์ จากการควบคุมความเร็วของมอเตอร์แบบวงเปิดในโปรแกรม Arduino IDE

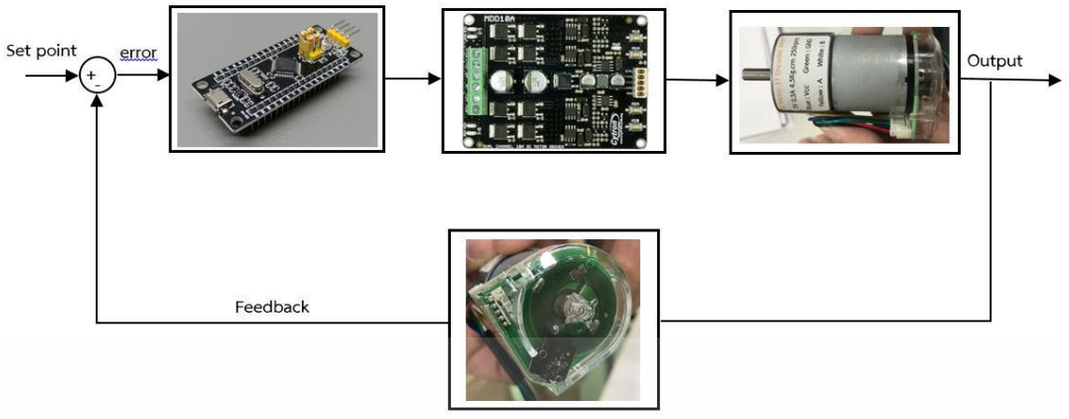
จากรูปที่ 3.8 เนื่องจากคำสั่ง Serial.print() ในโปรแกรม Arduino IDE นั้นเป็นการใช้ฮาร์ดแวร์ในการทำงาน ทำให้ทุกครั้งหากมีการใช้คำสั่งนี้จะทำให้คาบการสุ่มมีความผิดพลาด ในการเก็บค่านั้นจะไม่ใช้การแสดงค่าความเร็วของมอเตอร์ออกมาทาง Serial Monitor แต่จะใช้การเก็บค่าของความเร็วของมอเตอร์ที่คำนวณได้ไปเก็บไว้ในตัวแปร Array หลังจากที่โปรแกรมจะคำนวณจนได้ค่าครบตามที่ต้องการ แล้วจึงทำการแสดงค่าออกมาทั้งหมดเพียงครั้งเดียว การกระทำแบบนี้จะไม่ทำให้คาบการสุ่มมีการเปลี่ยนแปลง และการเก็บค่าของตัวแปรใน Array ค่าความเร็วของมอเตอร์ที่คำนวณออกมาได้จะเป็นตัวแปร float ซึ่งเป็นทศนิยม การเก็บตัวแปรทศนิยมในตัวแปร Array จะทำให้โปรแกรมทำงานหนัก โดยจะมีวิธีการแก้ปัญหา คือ การนำเลข 10 ไปคูณ เพื่อให้ได้เป็นตัวแปร integer หรือจำนวนเต็ม หากจะนำไปใช้งานก็ทำการหารด้วยเลข 10 ก่อนถึงจะสามารถนำไปใช้งานได้

รูปที่ 3.9 แสดงคำสั่ง Interrupt Service Routine จากการควบคุมความเร็วของมอเตอร์แบบวงเปิดในโปรแกรม Arduino IDE

จากรูปที่ 3.9 เป็นการสร้างโปรแกรมบริการอินเทอร์รัปต์ หรือ Interrupt Service Routine ซึ่งอธิบายไว้ในการทดลองที่ 3.2.1 เป็นโปรแกรมที่จะเรียกใช้งานทุกครั้งเมื่อเกิดการ Interrupt ไว้ใช้สำหรับการสร้างคาบเวลาที่แน่นอน เนื่องจากการกำหนดคาบการสุ่มจากส่วนของ Setup ใน Arduino IDE นั้น หากเวลาที่ต้องการน้อยเกินไปอาจจะกำหนดตัวแปรไปในส่วนนี้เพื่อเพิ่มระยะเวลาการทำงาน ในโครงงานนี้ได้ใช้ตัวแปร t ในการเพิ่มค่าของตัวมันเองทุกครั้งที่มีเกิดการ interrupt ขึ้นเพื่อเพิ่มระยะเวลาในการคำนวณความเร็วของโปรแกรม

3.1.2 การควบคุมความเร็วของมอเตอร์แบบวงปิด (Closed Loop Control)

หลังจากที่ได้ทำการควบคุมความเร็วของมอเตอร์แบบวงเปิดแล้ว ในขั้นตอนต่อไปจะทำการศึกษาการควบคุมความเร็วของมอเตอร์แบบวงปิดโดยมีการจำลอง (Simulation) ใน Simulink Toolbox ของโปรแกรม MATLAB ซึ่งขั้นตอนการทดลองต่างๆ แสดงอยู่ในการทดลองที่ 3.2.4 และการทดลองที่ 3.2.5 หลังจากได้ผลการตอบสนองที่ต้องการแล้ว จึงทำการสร้างโปรแกรมในโปรแกรม Arduino IDE ขึ้นมา โดยใช้โปรแกรมจากการควบคุมความเร็วของมอเตอร์แบบวงเปิดมาดัดแปลง โดยเพิ่มส่วนของการควบคุมขึ้นมา

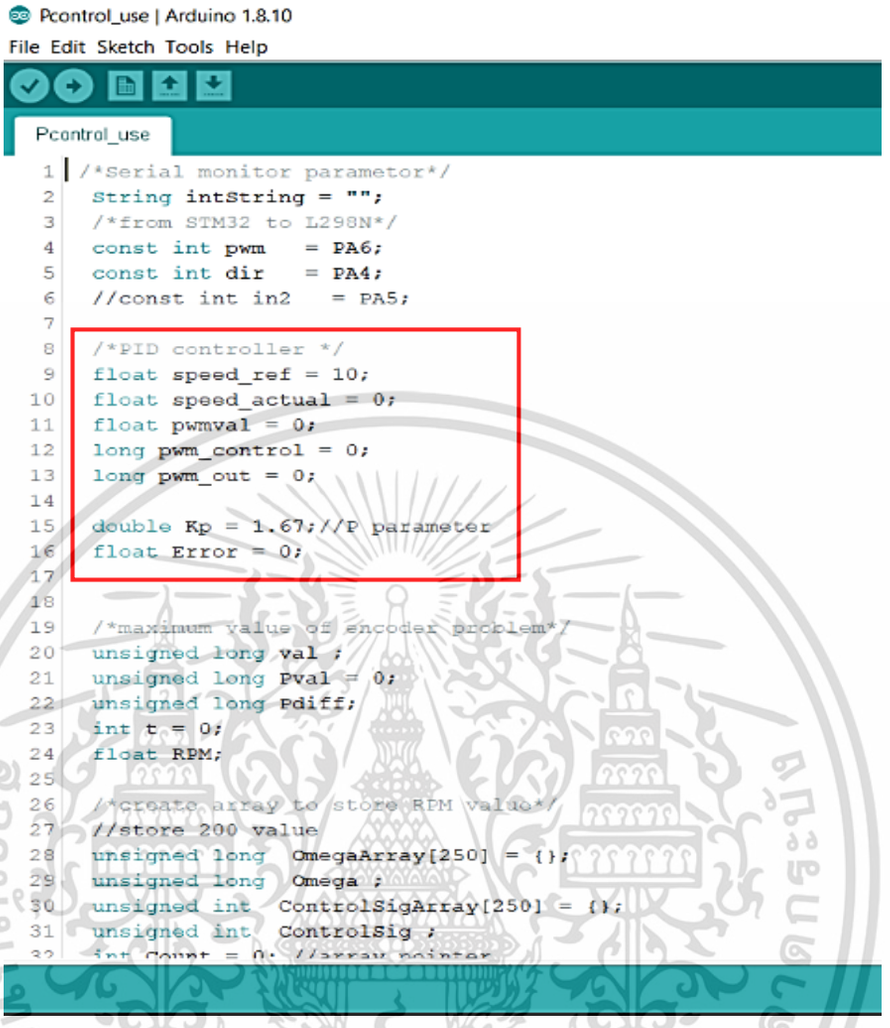


รูปที่ 3.10 แสดง Block Diagram ของการควบคุมความเร็วของมอเตอร์แบบวงปิด

จากรูปที่ 3.10 ในการควบคุมแบบวงปิดนั้นจะเป็นการส่งแบบป้อนกลับ (Feedback) ของค่าเอาต์พุตมายังตัวควบคุมเพื่อคำนวรหาค่าความผิดพลาดหลังจากนั้นจะทำการคำนวณเพื่อสร้างสัญญาณควบคุมส่งไปยังระบบให้มีผลตอบสนองตามค่าที่ได้ตั้งไว้ ในโครงงานนี้ได้มีการใช้ตัวควบคุมแบบพีและพีไอ (P/PI) ซึ่งมีการจำลองระบบควบคุมแบบปิดในโปรแกรม MATLAB (การทดลองที่ 3.2.4 และการทดลองที่ 3.2.5) หลังจากนั้นนำค่าที่ได้มาสร้างโปรแกรมใน Arduino IDE ผลดูผลตามสนองว่าตรงตามที่ต้องการหรือไม่

3.1.2.1 การควบคุมมอเตอร์ไฟฟ้ากระแสตรงโดยใช้ตัวควบคุมแบบพี

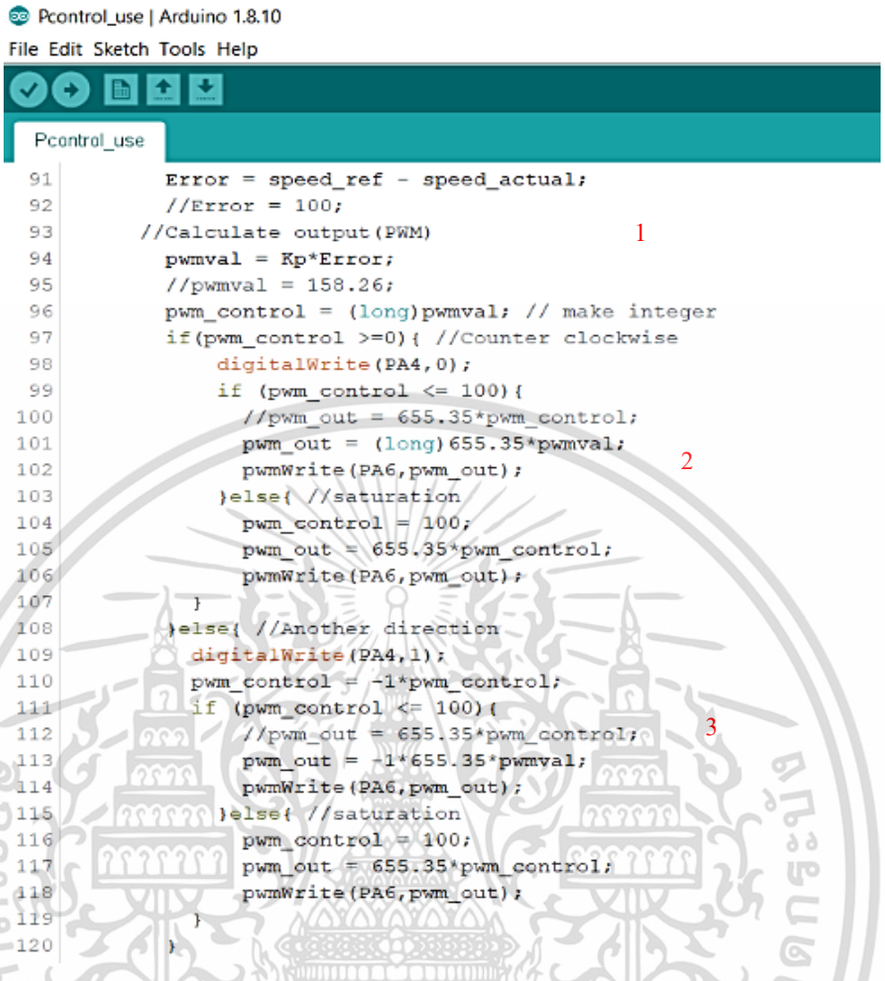
การใช้งานตัวควบคุมแบบพีบนมอเตอร์นั้น จะใช้โปรแกรมจากการควบคุมแบบวงเปิดมาประยุกต์ใช้งาน โดยจะเพิ่มในส่วนของการกำหนดตัวแปรของตัวควบคุมเพิ่มเข้ามา และการคำนวณค่าความผิดพลาดต่างๆ รวมไปถึงการกำหนดค่า Set Point ที่ต้องการ ในหัวข้อนี้จะอธิบายโปรแกรมในส่วนที่เพิ่มเข้ามา ส่วนอื่นๆยังคงใช้รูปแบบเดียวกันการการทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด



รูปที่ 3.11 แสดงการกำหนดตัวแปรของตัวควบคุมแบบพีของการควบคุมความเร็วของมอเตอร์แบบวงปิดในโปรแกรม Arduino IDE

จากรูปที่ 3.11 ในกรอบสีแดงแสดงค่าตัวแปรของตัวแปรต่างๆ ดังนี้

1) speed_ref คือ ตัวแปรที่กำหนด set point ของระบบเพื่อให้ระบบไปยังค่าเป้าหมายที่ต้องการ
2) speed_actual คือ ตัวแปรที่เก็บค่าความเร็วที่ counter register รับสัญญาณพัลส์มาจากเอ็นโค้ดเดอร์ เพื่อนำมาคำนวณค่าความเร็ว
3) pwmval คือ ค่าของเอาต์พุตที่ระบบคำนวณได้เป็นทศนิยม
4) pwm_control คือ ตัวแปรที่กำหนดขอบเขตของเอาต์พุต
5) pwmout คือ ตัวแปรที่นำค่า pwmval มาทำให้เป็นจำนวนเต็ม เพื่อส่งไปยังบอร์ดขับมอเตอร์
6) Kp คือ ค่าคงที่ของตัวควบคุมแบบพี
7) Error คือ ตัวแปรที่เก็บค่าความแตกต่างระหว่างค่าจริงและค่า Set Point



รูปที่ 3.12 แสดงส่วนของการทำงานของตัวควบคุมแบบพีของการควบคุมความเร็วของมอเตอร์แบบวงปิดในโปรแกรม Arduino IDE

จากรูปที่ 3.12 ในส่วนแรก เป็นการคำนวณค่าความผิดพลาดแล้วนำไปเก็บตัวแปร Error หลังจากนั้น จะนำค่าที่ได้ไปคูณกับค่าคงที่ของตัวควบคุมแบบพี จะได้เอาต์พุตหรือตัวแปร pwmval ที่ส่งสัญญาณควบคุมไปยังบอร์ดขับมอเตอร์ แต่สัญญาณควบคุมที่บอร์ดรับได้นั้นเป็นจำนวนเต็มจริง จึงต้องเปลี่ยนชนิดของแปรให้กลายเป็น integer แล้วนำไปเก็บในตัวแปร pwm_control

ในส่วนที่ 2 และ 3 จะเป็นการเปรียบเทียบตัวแปร pwm_control ว่ามีค่ามากกว่า 0 หรือน้อยกว่า 0 เนื่องจากมอเตอร์มีทิศทางการหมุน 2 ทิศทาง ในโปรแกรมกำหนดให้เมื่อมอเตอร์หมุนไปในทิศทางตรงข้ามจะทำให้ค่า pwm_control มีค่าเป็นลบ และในส่วนที่ 2 จะเป็นกรณีที่ pwm_control มีค่าเป็นบวก และในส่วนที่ 3 จะเป็นกรณีที่ pwm_control มีค่าเป็นลบ

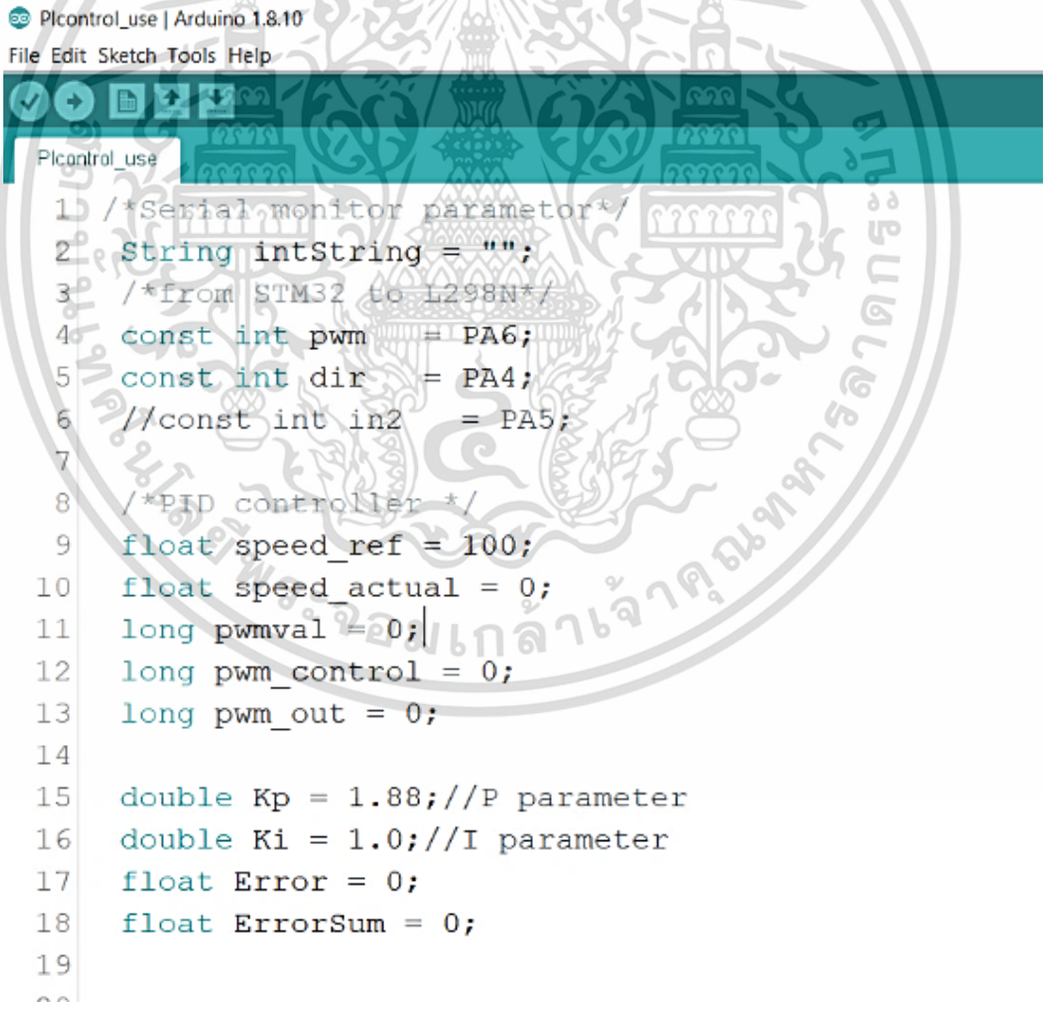
สำหรับเงื่อนไขย่อยในส่วนที่ 2 เงื่อนไขแรกจะทำงานเมื่อค่า pwm_control มีค่าน้อยกว่าหรือเท่ากับ 100 ค่าของ pwm_out จะเท่ากับการเอาค่า pwmval ไปคูณด้วย 655.35 แล้วทำเป็นจำนวนเต็ม หลังจากนั้นนำค่าที่ได้ไปสั่งงานขา pwmpin ของไมโครคอนโทรลเลอร์เพื่อส่งสัญญาณไปยังบอร์ดขับมอเตอร์ และเงื่อนไขที่ 2 จะทำงานเมื่อค่า pwm_control มีค่ามากกว่า 100

เป็นการกำหนดขีดจำกัดของค่าที่ส่งออกไปได้ โดยค่า pwm_out จะมีค่าเท่ากับ 655.35*100 หรือค่าที่ขีดจำกัดสูงสุดที่บอร์ดขับมอเตอร์รับได้ สำหรับเงื่อนไขย่อยในส่วนที่ 3 จะเหมือนกับส่วนที่ 2 ทุกประการ เพียงแต่จะมีการนำค่า pwm_control ไปคูณด้วย -1 เนื่องจากในตอนแรกค่า pwm_control มีค่าเป็นลบและบอร์ดขับมอเตอร์ไม่สามารถรับค่าที่เป็นจำนวนลบได้

หลังจากอัปโหลดโปรแกรมดังที่กล่าวมาข้างต้นเข้าไปยังไมโครคอนโทรลเลอร์ ทำการเก็บค่าของความเร็วเอาต์พุต และค่าของสัญญาณควบคุมเพื่อนำไปเปรียบเทียบกับการจำลองว่ามีความเหมือนหรือแตกต่างกันอย่างไร

3.1.2.2 การควบคุมมอเตอร์ไฟฟ้ากระแสตรงโดยใช้ตัวควบคุมแบบพีไอ

หลังจากการจำลองการควบคุมแบบวงปิดในโปรแกรม MATLAB แล้วทำการทดลองจริงเพื่อดูผลการตอบสนองและผลกระทบของตัวแปรต่างๆ แล้วจึงทำการศึกษาตัวควบคุมแบบพีไอเพื่อทดสอบระบบ เมื่อได้ผลตอบสนองตามที่ต้องการแล้ว จึงสร้างโปรแกรมในโปรแกรม Arduino IDE โดยตัวโปรแกรมจะมีความคล้ายคลึงกับตัวโปรแกรมของตัวควบคุมแบบพีเพียงแต่เพิ่มส่วนของตัวควบคุมแบบไอเพิ่มเข้ามา



รูปที่ 3.13 แสดงการกำหนดตัวแปรของตัวควบคุมแบบพีไอของการควบคุมความเร็วของมอเตอร์แบบวงปิดในโปรแกรม Arduino IDE

จากรูปที่ 3.13 ส่วนที่เพิ่มเข้ามาคือตัวแปร Ki และ ErrorSum สำหรับ Ki คือค่าคงที่ของตัวควบคุมแบบไอ และ ErrorSum คือ ผลรวมของค่าความผิดพลาดที่เกิดจากระบบ ตัวแปรอื่นๆ เหมือนกับโปรแกรมที่กล่าวมาก่อนหน้าทุกประการ

รูปที่ 3.14 แสดงการคำนวณค่าความผิดพลาดของการควบคุมความเร็วของมอเตอร์แบบวงปิดในโปรแกรม Arduino IDE

จากรูปที่ 3.14 การคำนวณค่าความผิดพลาดหรือตัวแปร Error คือ การนำค่า Set Point ลบกับค่าความเร็วที่คำนวณได้มาจากโปรแกรม หลังจากนั้นจะมีตัวแปร ErrorSum ที่เก็บค่า Error ของปัจจุบันและครั้งก่อนหน้ามารวมกัน เพื่อนำไปคูณกับ Ki หรือค่าคงที่ของตัวควบคุมแบบไอ แล้วนำค่าที่ได้ไปรวมกับพจน์ของตัวควบคุมแบบพี เพื่อคำนวณค่า PWM output หรือ pwmval ออกมา

หลังจากอัปโหลดโปรแกรมการควบคุมแบบพีไอเข้าไปยังไมโครคอนโทรลเลอร์แล้วทำการเก็บค่าของความเร็วเอาต์พุตและสัญญาณควบคุมเพื่อนำไปเปรียบเทียบกับการจำลองในโปรแกรม MATLAB

3.2 การทดลองที่เกี่ยวข้องกับการทำโครงงานของปริญญานิพนธ์

ในการทำโครงงานนี้ได้มีการสร้างการทดลองขึ้นหลายการทดลอง เนื่องจากโครงงานนี้เป็นลักษณะของงานวิจัย จำเป็นต้องมีการทดลองเพื่อหาผลลัพธ์ตามที่ได้ตั้งสมมติฐานไว้ โดยการทดลองจะแบ่งออกเป็น 2 ส่วนใหญ่คือการทดลองที่เกี่ยวกับโปรแกรม Arduino IDE และการจำลอง (Simulation) ในโปรแกรม MATLAB

3.2.1 การทดลองการสร้างคาบการสุ่ม (Sampling Time)

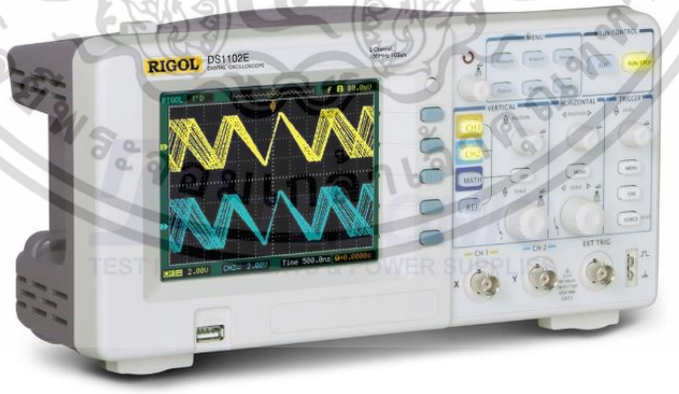
การเก็บข้อมูลจากการทดลองจะมีช่วงที่เรียกว่าคาบการสุ่ม (Sampling Time) ในระดับพื้นฐานนั้นสามารถกำหนดระยะเวลา

3.2.1.1 จุดประสงค์การทดลองสร้างคาบการสุ่ม

1) สร้างคาบการสุ่มที่แม่นยำเพื่อใช้ในการเก็บค่าความเร็ว (Speed) จากมอเตอร์ไฟฟ้ากระแสตรง (DC motor)
2) เพื่อเรียนรู้การใช้งานไมโครคอนโทรลเลอร์ (Microcontroller) ในระดับรีจิสเทอร์ (Register)
3) เพื่อเรียนรู้การใช้เครื่อง Oscilloscope

3.2.1.2 อุปกรณ์ที่ใช้ในการทดลองสร้างคาบการสุ่ม

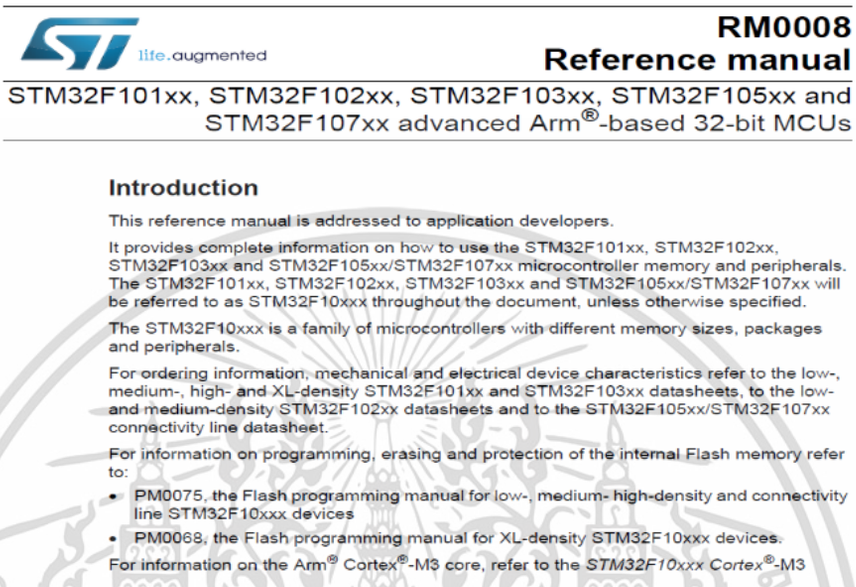
1) ไมโครคอนโทรลเลอร์ รุ่น STM32f1038TC6 จำนวน 1 ตัว
2) สายไฟขนาดเล็กจำนวน 2 เส้น
3) เครื่อง Oscilloscope ยี่ห้อ RIGOL รุ่น DS1102E จำนวน 1 เครื่อง แสดงดังรูป 3.15
4) คอมพิวเตอร์ที่ติดตั้งโปรแกรม Arduino IDE



รูปที่ 3.15 แสดง Oscilloscope ยี่ห้อ RIGOL รุ่น DS1102E

3.2.1.3 วิธีทำการทดลองสร้างคาบการสุ่ม

1) ทำการสร้างคาบการสุ่ม (Sampling Time) จาก Timer interrupt ในรีจิสเทอร์ของไมโครคอนโทรลเลอร์ โดยการกำหนดตัวแปรต่างๆ สามารถดูได้จาก Datasheet ของไมโครคอนโทรลเลอร์ตัวนั้นๆ



life.augmented

RM0008
Reference manual

STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs

Introduction

This reference manual is addressed to application developers.

It provides complete information on how to use the STM32F101xx, STM32F102xx, STM32F103xx and STM32F105xx/STM32F107xx microcontroller memory and peripherals. The STM32F101xx, STM32F102xx, STM32F103xx and STM32F105xx/STM32F107xx will be referred to as STM32F10xxx throughout the document, unless otherwise specified.

The STM32F10xxx is a family of microcontrollers with different memory sizes, packages and peripherals.

For ordering information, mechanical and electrical device characteristics refer to the low-, medium-, high- and XL-density STM32F101xx and STM32F103xx datasheets, to the low- and medium-density STM32F102xx datasheets and to the STM32F105xx/STM32F107xx connectivity line datasheet.

For information on programming, erasing and protection of the internal Flash memory refer to:

- PM0075, the Flash programming manual for low-, medium- high-density and connectivity line STM32F10xxx devices
- PM0068, the Flash programming manual for XL-density STM32F10xxx devices.

For information on the Arm® Cortex®-M3 core, refer to the *STM32F10xxx Cortex®-M3*

รูปที่ 3.16 Datasheet ของไมโครคอนโทรลเลอร์ยี่ห้อ STM32f103C8T6

2) เมื่อกำหนดค่าต่างๆ แล้วทำการเชื่อมต่อไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ แล้วทำการอัปโหลดโปรแกรมเข้าไปยังไมโครคอนโทรลเลอร์

SamplingLoopProgram | Arduino 1.8.10

File Edit Sketch Tools Help

SamplingLoopProgram §

```
1  void setup() {
2
3  /*Timer 2 as time base*/
4  TIMER2_BASE->PSC = 72; //72MHz/72=1,000,000 Hz
5  //f = 1,000,000 , T = 0.000001
6  TIMER2_BASE->ARR = 100; //100 * 0.000001 =  0.0001 S = 0.1 mS
7  TIMER2_BASE->CNT = 0;  //clear counter
8  timer_attach_interrupt(TIMER2, 0, handler_tim2); //interrupt on timer update
9  TIMER2_BASE->CR1 |= 0x0001; //enable timer.
10
11 GPIOB_BASE->CRH = 0x44424444; //enable PB12 as output
12 //0100 0100 0100 0010 0100 0100 0100 0100
13
14 }
15
16 void loop() {}
17
18 /*timer 2 interrupt*/
19 void handler_tim2(void){
20   GPIOB_BASE->ODR ^= 0x1000;// toggle LED built in to check sampling
21
22 }
```

รูปที่ 3.17 แสดงโปรแกรมสร้างคาบการสุ่มในโปรแกรม Arduino IDE

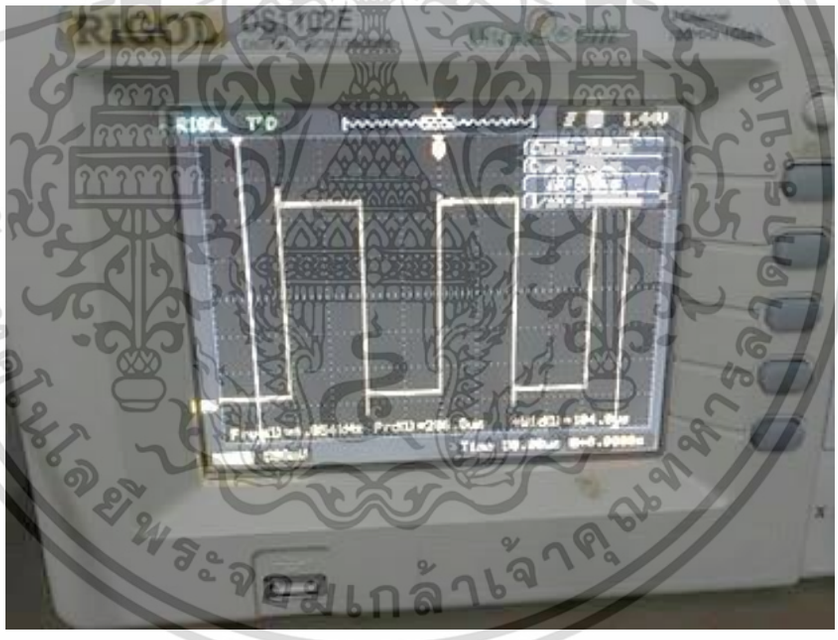
3) การวัดค่าจะใช้การกระพริบของหลอดไฟ LED ที่ built in อยู่ในตัวไมโครคอนโทรลเลอร์ โดยจะต่อสายไฟจาก LED ออกมา 1 เส้น และจาก GND ของไมโครคอนโทรลเลอร์อีก 1 เส้น ต่อเข้ากับ Oscilloscope เพื่อดูผลลัพธ์

4) สังเกตผลลัพธ์ว่าตรงตามที่กำหนดหรือไม่ หลังจากนั้นทำการเพิ่มหรือลดค่าของคาบการสุ่ม (Sampling Time) ตามที่ต้องการแล้วดูผลลัพธ์

หมายเหตุ : เนื่องจากโปรแกรม Arduino IDE รองรับไมโครคอนโทรลเลอร์ยี่ห้อ Arduino เท่านั้น การใช้งานยี่ห้ออื่นจำเป็นต้องลง Driver หรือ library ของยี่ห้อนั้นๆ ก่อนจึงจะสามารถใช้งานร่วมกันได้

3.2.1.4 ผลการทดลองสร้างคาบการสุ่ม

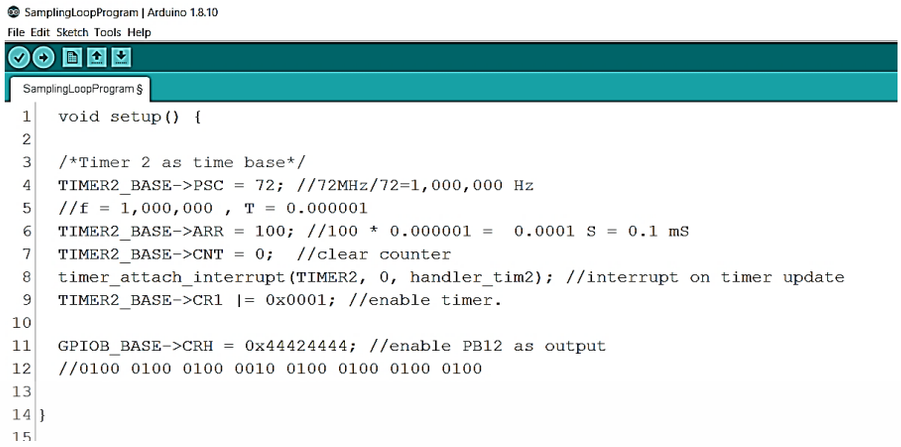
บนหน้าจอของเครื่อง Oscilloscope แสดงคาบ (Period) และความกว้าง (Width) ของช่วงสัญญาณ โดยมีความกว้างเท่ากับ 100 ไมโครวินาที หรือ 0.1 มิลลิวินาที ซึ่งเป็นช่วงเวลาที่ไฟ LED ติดและดับ ในส่วนของคาบนั้นมีค่าเท่ากับ 200 ไมโครวินาทีหรือ 0.2 มิลลิวินาที ตามที่ได้กำหนดไว้ในโปรแกรม Arduino IDE ซึ่งผลการทดลองแสดงดังรูปที่ 3.18



รูปที่ 3.18 แสดงคาบการสุ่มที่ได้กำหนดไว้

3.2.1.4 สรุปผลการทดลองสร้างคาบการสุ่ม

การสร้างคาบการสุ่ม (Sampling Time) โดยใช้รีจิสเทอร์ (Register) ของไมโครคอนโทรลเลอร์นั้นสามารถสร้างได้โดยการกำหนดค่าที่รีจิสเทอร์ดังต่อไปนี้



รูปที่ 3.19 แสดงรีจิสเทอร์ (Register) ชนิดต่างๆ ในการสร้างคาบการสุ่ม

1) TIMERx_BASE เป็นคำสั่งการเรียกใช้งาน Timer หรือตัวนับเวลาในรีจิสเทอร์โดยจำนวนของ Timer ในไมโครคอนโทรลเลอร์แต่ละตัวมีจำนวนของ Timer ไม่เท่ากัน (x = 0, 1, 2, ...) ในการทดลองใช้ Timer ตัวที่ 2 จึงกำหนดเป็น TIMER2_BASE
2) TIMERx_BASE->PSC เป็นรีจิสเทอร์ที่กำหนดค่าของตัวหาร (Prescaler) เพื่อหารความถี่ (Clock) จากคริสตัลหรือตัวกำเนิดความถี่ในไมโครคอนโทรลเลอร์ เพื่อนำความถี่มาสร้างคาบ (Period) ในการนับ สำหรับไมโครคอนโทรลเลอร์ชนิด STM32f103C8T6 นั้นสามารถกำหนดค่าได้ตั้งแต่ 1 ถึง 65536 หากเป็นยี่ห้ออื่นจะมีวิธีการกำหนดค่าที่แตกต่างกันออกไป
3) TIMERx_BASE->ARR เป็นรีจิสเทอร์ที่กำหนดจำนวนครั้งในการนับสามารถกำหนดค่าได้ไม่จำกัดแล้วแต่ความต้องการของผู้ใช้งาน
4) TIMERx_BASE->CNT เป็นรีจิสเทอร์ที่กำหนดค่าเริ่มต้นของตัวนับ (Counter) สำหรับรีจิสเทอร์ตัวนี้มีความสำคัญเป็นอย่างมาก เมื่อเกิดการนับแต่ละครั้ง หากไม่กำหนดให้ตัวนับเริ่มต้นที่ 0 จะทำให้คาบการสุ่มมีค่าผิดพลาดได้ เพราะจะวนกลับมานับที่ 0 ทุกครั้ง
5) TIMERx_BASE->CR1 เป็นรีจิสเทอร์กำหนดการเปิด/ปิด การใช้งานของ Timer หาก bit ที่ 0 มีค่าเป็น 1 จะเป็นการเปิดใช้งาน และหาก bit ที่ 0 มีค่าเป็น 0 จะเป็นการปิดการใช้งาน Timer
6) timer_attach_interrupt (TIMERx, mode, function) เป็นคำสั่งสร้างฟังก์ชันที่จะทำการเรียกทุกครั้งที่เกิดการ Interrupt โดย TIMERx คือ Timer ตัวที่สร้างคาบเวลาให้เกิดการ Interrupt Mode คือโหมดการทำงานของการ Interrupt โดย 0 คือ Falling Edge หรือการเกิด Interrupt ที่ขอบขาลง และ Function คือ คำสั่งที่จะให้ทำงานเมื่อเกิดการ Interrupt หรือเรียกว่า โปรแกรมบริการ Interrupt (Interrupt Service Routine)

ในการทดลองจะใช้หลอดไฟ LED เป็นตัววัดคาบการสุ่ม (Sampling Time) โดยจะใช้คำสั่ง GPIOB_BASE->ODR ในการ Toggle หลอดไฟ LED รูปแบบของคำสั่งจะใช้แบบรีจิสเทอร์ (Register) เช่นเดียวกันกับการสร้างคาบการสุ่ม โดย ODR คือ รีจิสเทอร์ที่กำหนดให้ขานั้นๆ เป็น OUTPUT ซึ่งการเขียนคำสั่งให้หลอดไฟ LED กระพริบในรูปแบบของการใช้รีจิสเทอร์ แสดงดังรูปที่ 3.20

```
/*timer 2 interrupt*/
void handler_tim2(void){
  GPIOB_BASE->ODR ^= 0x1000;// toggle LED built in to check sampling
  |
}
```

รูปที่ 3.20 แสดงฟังก์ชัน ISR ภายในโปรแกรม

การคำนวณคาบการสุ่ม (Sampling Time) ในการทดลองต้องการให้คาบการสุ่มมีค่าเท่ากับ 0.1 มิลลิวินาที หรือ 100 ไมโครวินาที การคำนวณสามารถทำได้ดังต่อไปนี้ (ความถี่คริสตัลของ STM32 ที่ใช้คือ 72 MHz)

ตารางที่ 3.1 Specification ของ STM32f103

Features	STM32F103
Clock Frequency	72 Mhz
I2C Buses	2
SPI Buses	2
CAN Bus	Yes
Analog Channel	10
PWM Channel	15
USART Buses	3
GPIO’s	32
On Board RTC	Yes
Architecture	ARM Cortex M3 32 bit
ADC Resolution	12 bit
Quantization Level	4096
Flash Memory	64KB
SRAM	20KB
Debugging	Serial, JTAG
PWM Resolution	16 bit

1) กำหนด TIMER2_BASE->PSC เป็น 72 ดังนั้น ความถี่ที่ใช้นับจะมีค่า 72 MHz / 72 มีค่าเท่ากับ 1 MHZ จะมีคาบการนับเท่ากับ 1 / 1MHz หรือ 1 ไมโครวินาที
2) กำหนด TIMERx_BASE->ARR เป็น 100 เพื่อให้โปรแกรมทำการนับ 100 ครั้ง จนเกิดการ Interrupt
3) คาบการสุ่มมีค่าเท่ากับ 100*1 ไมโครวินาที หรือ 100 ไมโครวินาที หรือ 0.1 มิลลิวินาที

การสร้างคาบการสุ่ม (Sampling Time) โดยการใช้รีจิสเทอร์ (Register) ของหน่วยประมวลผลของไมโครคอนโทรลเลอร์ สามารถสร้างคาบเวลาที่มีค่าความผิดพลาดไม่มากนัก ทำให้การประมวลผลรวมไปถึงการเก็บค่าภายในโปรแกรมมีความแม่นยำสูง

3.2.2 การทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด (Open loop control)

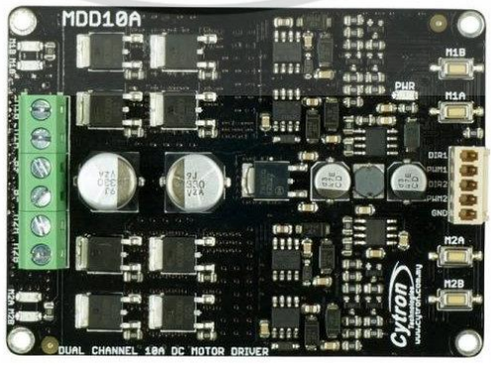
การควบคุม (Control) แบ่งออกเป็น 2 ชนิด คือ การควบคุมแบบวงเปิด (Open Loop Control) และการควบคุมแบบวงปิด (Closed Loop Control) ในการทดลองนี้จะเป็นการทดสอบการควบคุมแบบเปิด (Open Loop Control) ซึ่งการควบคุมแบบวงเปิดนั้น เป็นวิธีการพื้นฐานในการหาฟังก์ชันถ่ายโอน (Transfer Function) ของระบบ อีกทั้งยังสามารถวิเคราะห์ความเป็นเชิงเส้น (Linear) ของระบบได้อีกด้วย

3.2.2.1 จุดประสงค์ของการทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด

1) เพื่อเรียนรู้วิธีการควบคุมแบบวงเปิด (Open Loop Control)
2) เพื่อเรียนรู้วิธีการควบคุมความเร็วของมอเตอร์
3) เพื่อเรียนรู้วิธีการคำนวณความเร็วของมอเตอร์

3.2.2.2 อุปกรณ์ที่ใช้ในการทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด

1) ไมโครคอลโทรลเลอร์ STM32f103C8t6 จำนวน 1 ตัว
2) สายไฟ
3) มอเตอร์ไฟฟ้ากระแสตรง (DC motor) 12V จำนวน 1 ตัว
4) บอร์ดขับมอเตอร์ (Motor Drive Board) ยี่ห้อ Cytron รุ่น MDD10A



รูปที่ 3.21 แสดง Motor Drive Board ยี่ห้อ Cytron รุ่น MDD10A

5) Breadboard จำนวน 1 อัน

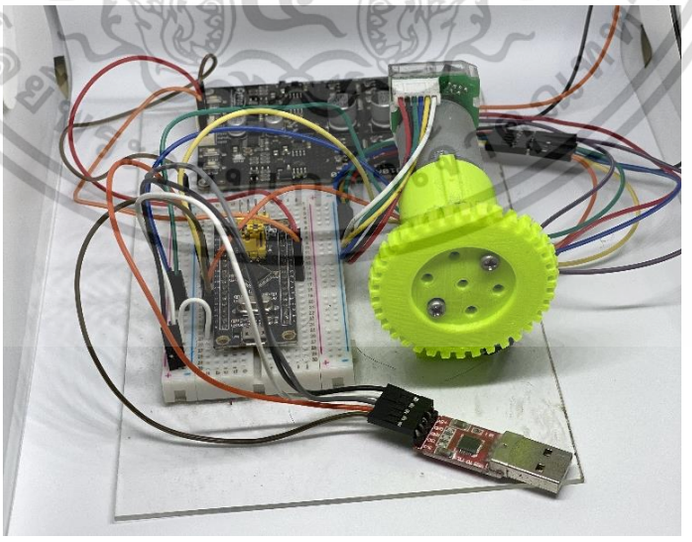
6) Multimeter ยี่ห้อ FLUKE รุ่น 179

รูปที่ 3.22 แสดง Multimeter ยี่ห้อ FLUKE รุ่น 179

7) คอมพิวเตอร์ที่ติดตั้งโปรแกรม Arduino IDE

3.2.2.3 วิธีทำการทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด

1) ทำการต่อวงจร ดังรูปที่ 3.23 จากนั้นทำการสร้างโปรแกรมคำนวณความเร็ว (Speed) ของมอเตอร์ แล้วทำการอัปโหลดโปรแกรมเข้าไปในไมโครคอนโทรลเลอร์



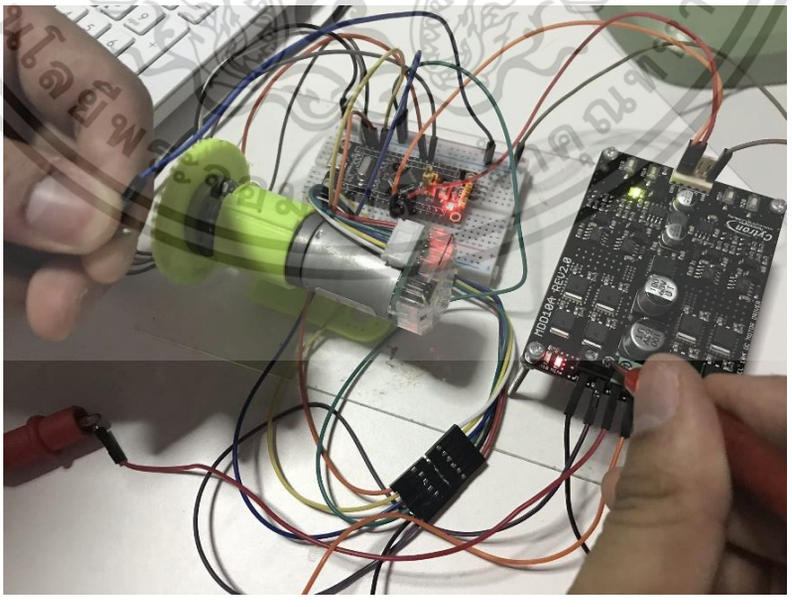
รูปที่ 3.23 แสดงวงจรการควบคุมความเร็วของมอเตอร์

2) ทำการเปลี่ยนค่า PWM ภายในโปรแกรม ตั้งแต่ 0 จนถึง 100% โดยเพิ่มค่าขึ้นทีละ 10 แล้วทำการอัปโหลดโปรแกรม สังเกตผลลัพธ์ที่ Serial Monitor ในโปรแกรม Arduino IDE แล้วนำค่าที่ได้ไปพล็อตกราฟดูเอาต์พุต

```
17
18  void setup() {
19
20  Serial.begin(9600);
21
22  /*set pin PB0 PB4 and PB5 from STM32F103C8T6 as output*/
23  //control motor speed
24  pinMode(PA6,PWM);
25  //direction pin
26  pinMode(PA4,OUTPUT);
27
28  digitalWrite(PA4,0);
29  pwmWrite(PA6,65535);//stop motor at the begining
30
```

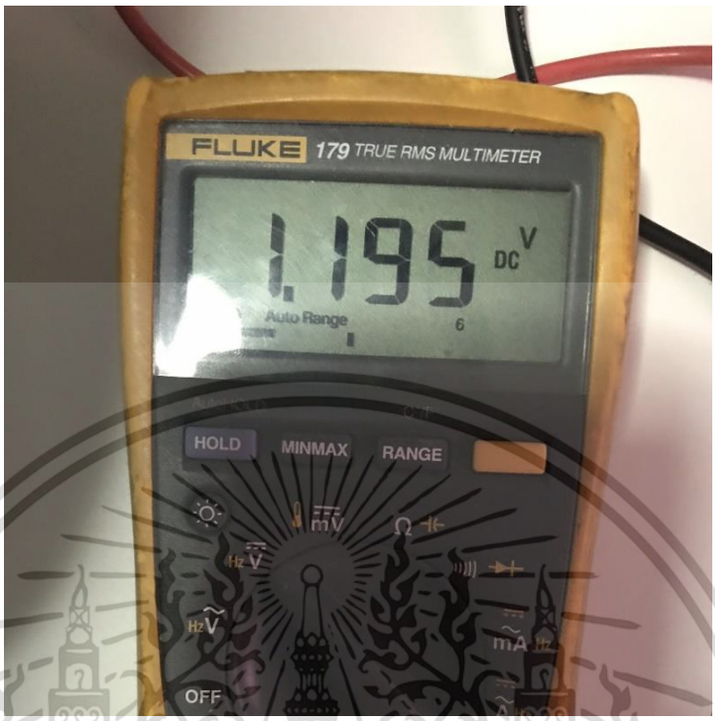
รูปที่ 3.24 แสดงโปรแกรมการทดสอบการควบคุมความเร็วของมอเตอร์แบบวงเปิด (Open Loop Control)

3) ในแต่ละครั้งที่ทำการเปลี่ยนค่า PWM ให้วัดค่าความต่างศักย์ไฟฟ้า (Voltage) ที่ Motor Drive Board แล้วบันทึกค่า โดยวิธีการวัดความต่างศักย์ไฟฟ้าที่บอร์ดขับมอเตอร์ คือนำขั้วบวกของ Multimeter ต่อเข้ากับขา PWM ของบอร์ดขับมอเตอร์ และขั้วลบของ multimeter ต่อเข้ากับ GND ของระบบ ดังแสดงในรูปที่ 3.25



รูปที่ 3.25 แสดงวิธีการวัดค่าความต่างศักย์ไฟฟ้าที่ Motor Drive Board

3.2.2.4 ผลการทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด

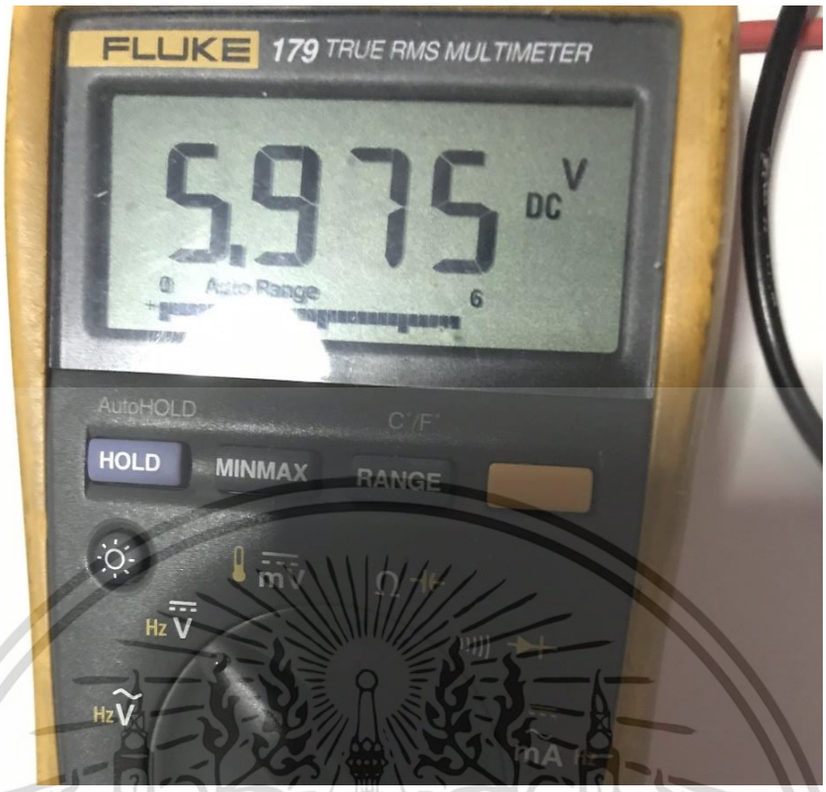


รูปที่ 3.26 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 10%

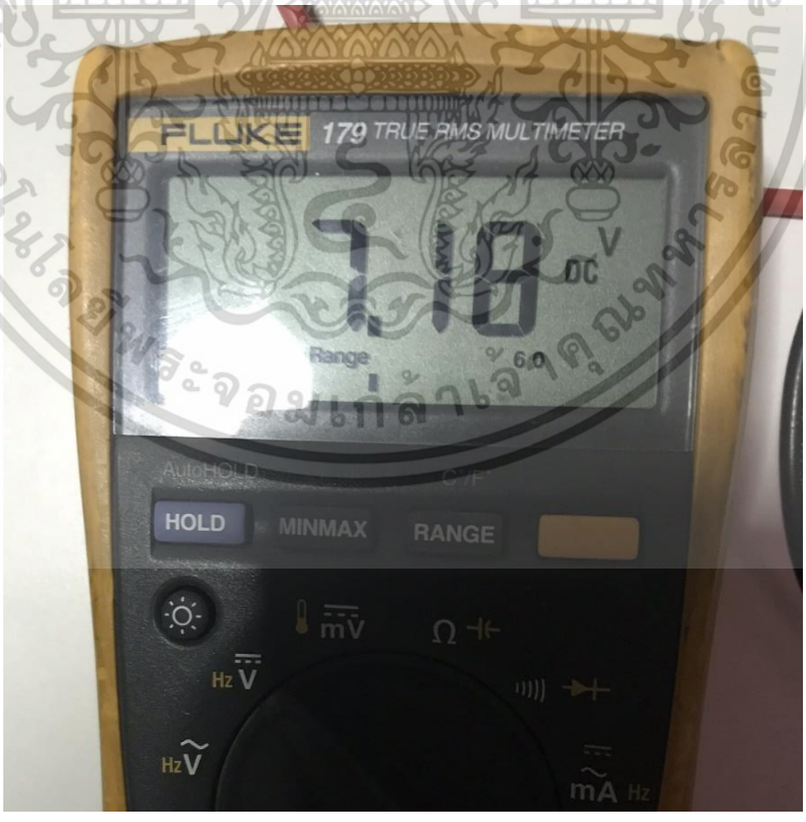
รูปที่ 3.27 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 20%

รูปที่ 3.28 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 30%

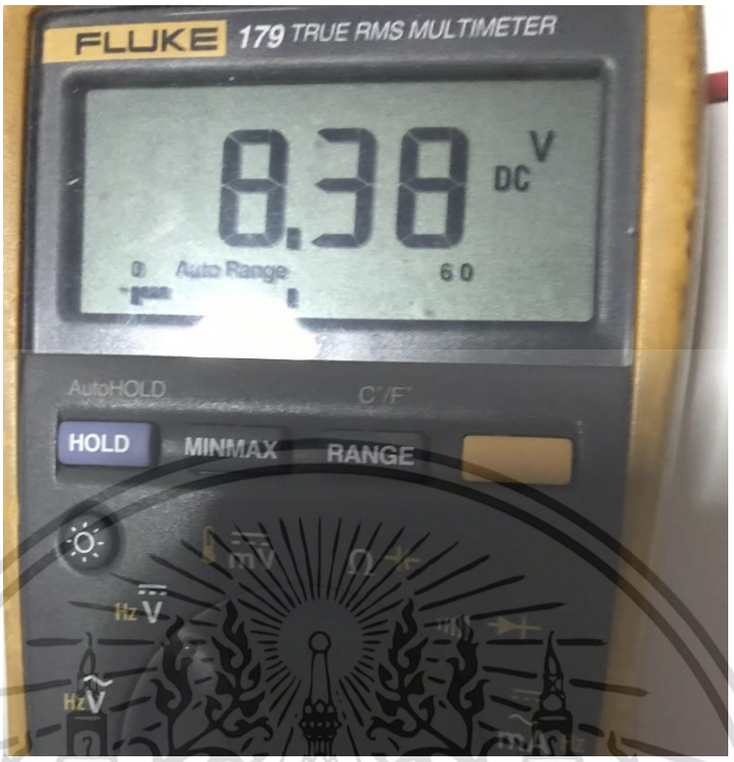
รูปที่ 3.29 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 40%



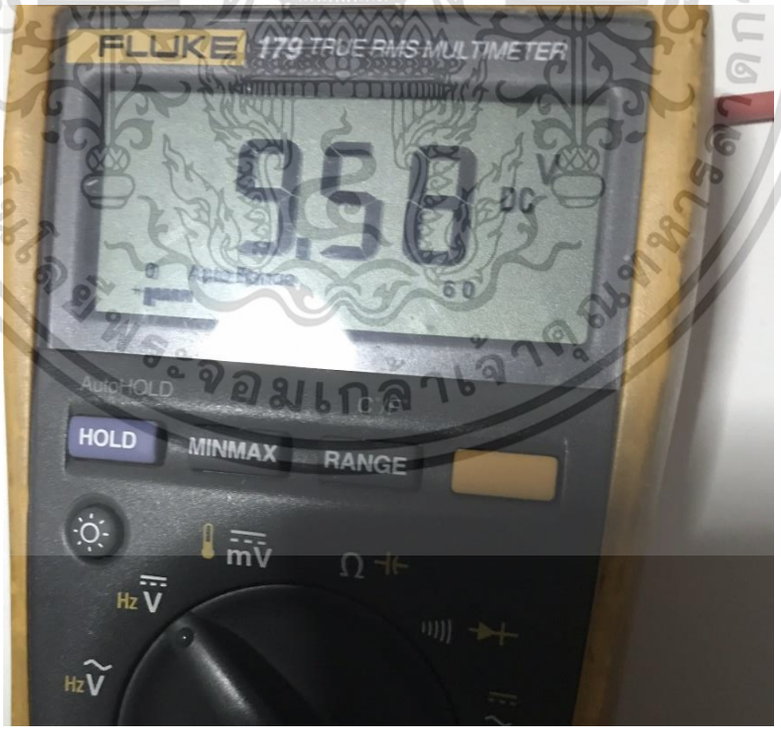
รูปที่ 3.30 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 50%



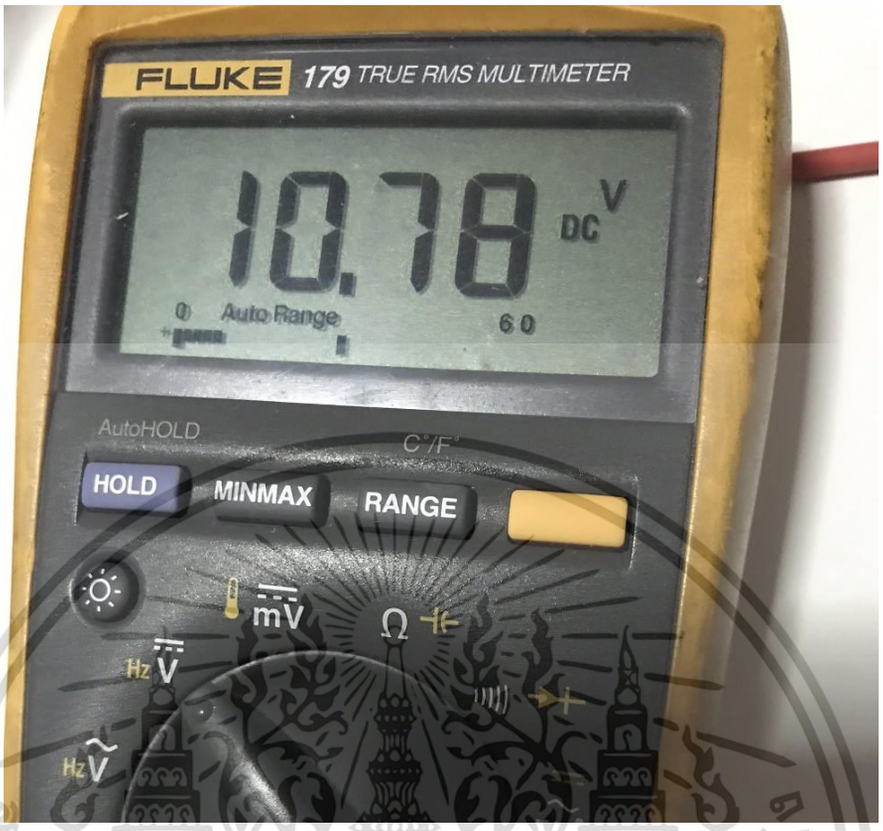
รูปที่ 3.31 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 60%



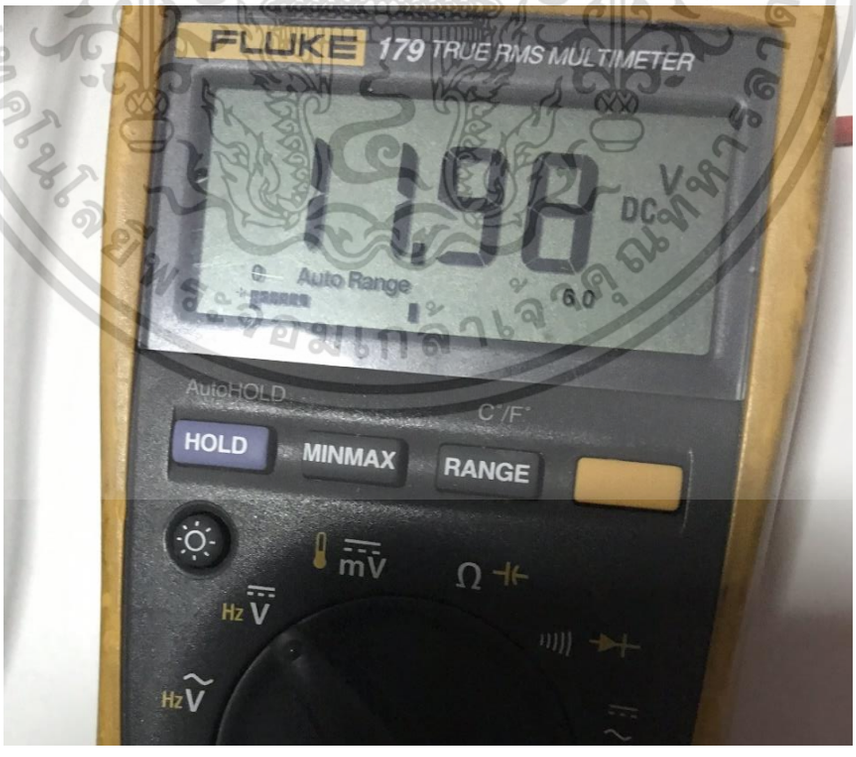
รูปที่ 3.32 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 70%



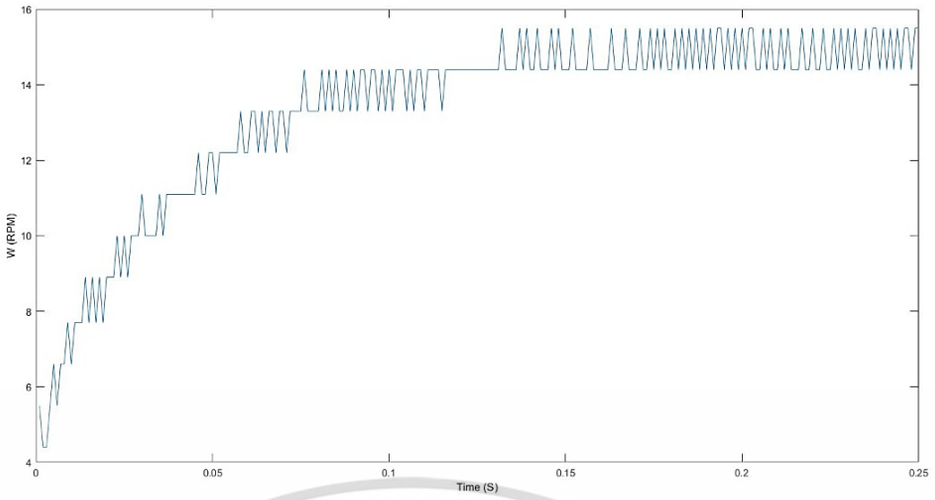
รูปที่ 3.33 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 80%



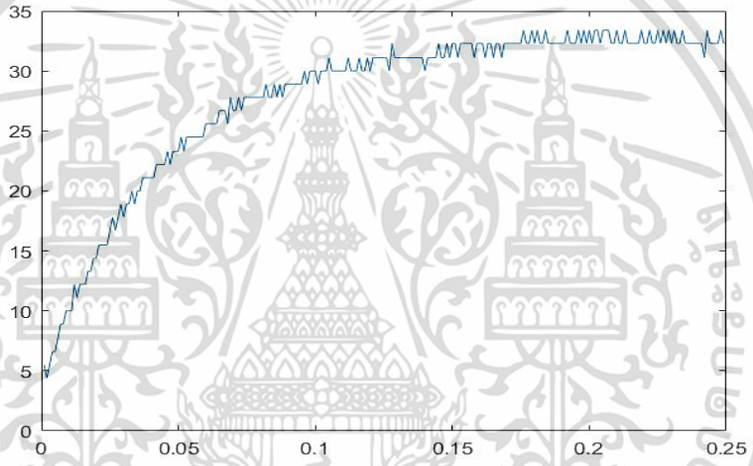
รูปที่ 3.34 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 90%



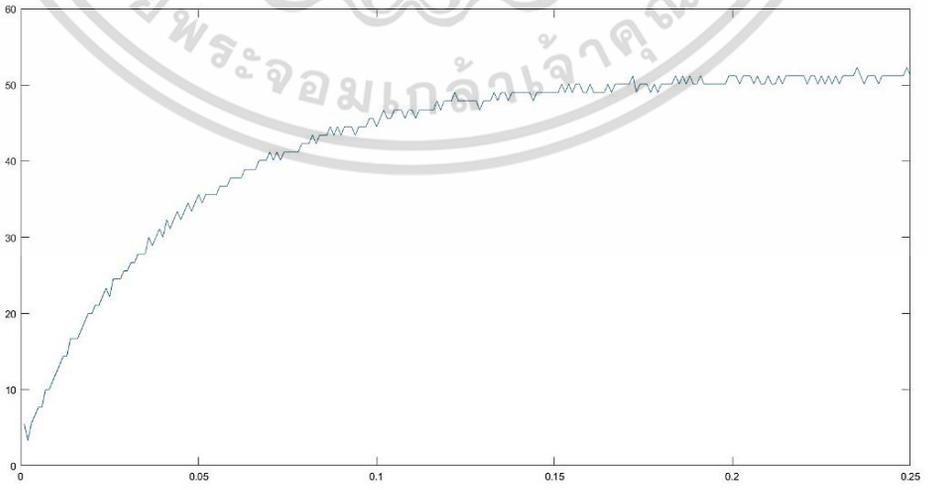
รูปที่ 3.35 แสดงค่าความต่างศักย์ของบอร์ดขับมอเตอร์ที่ PWM เท่ากับ 100%



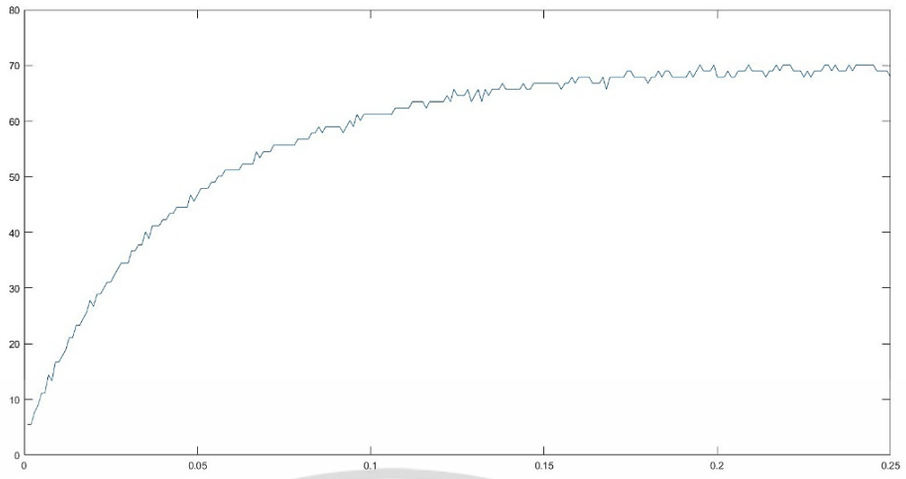
รูปที่ 3.36 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 10%



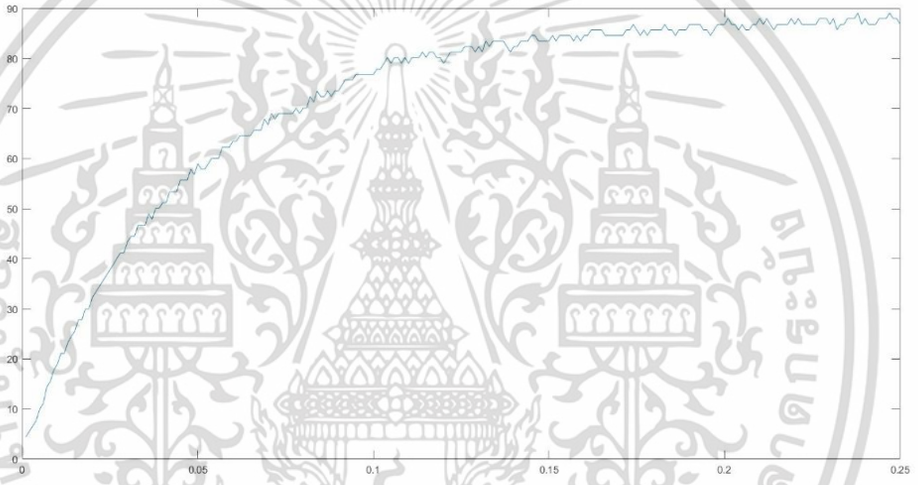
รูปที่ 3.37 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 20%



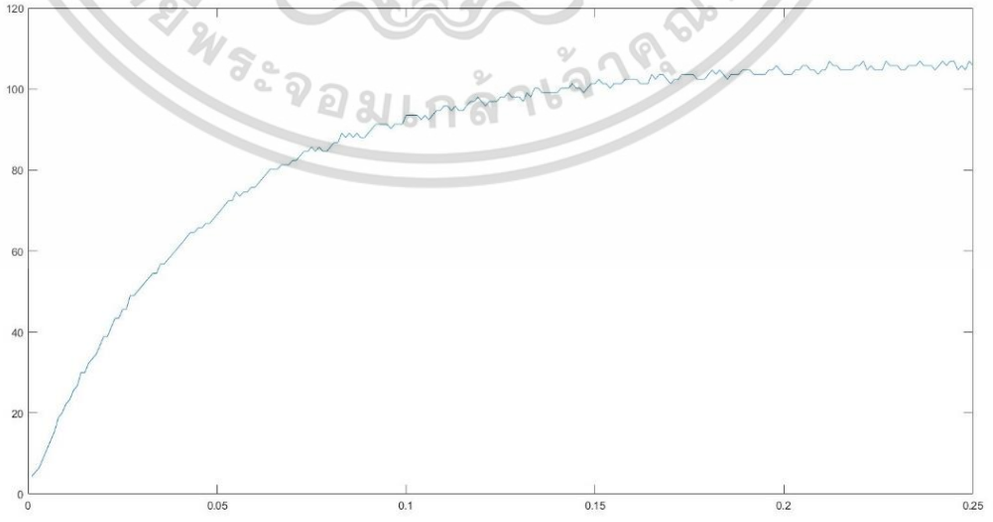
รูปที่ 3.38 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 30%



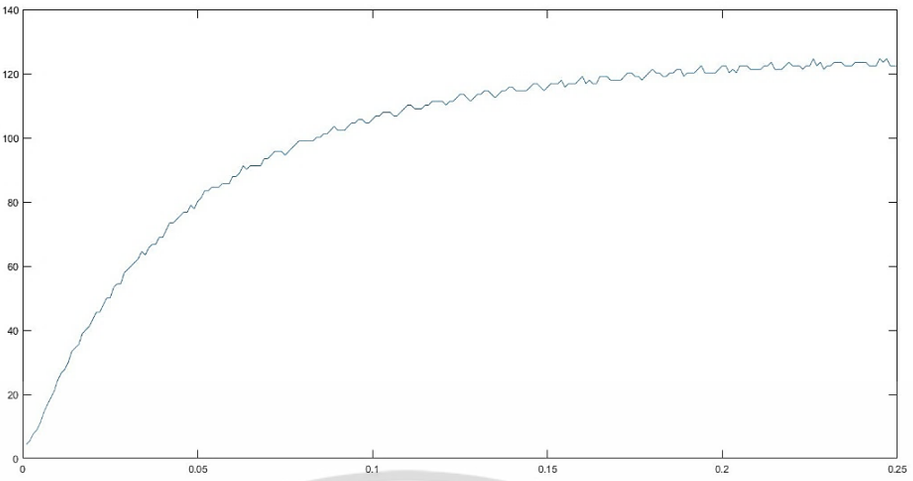
รูปที่ 3.39 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 40%



รูปที่ 3.40 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 50%

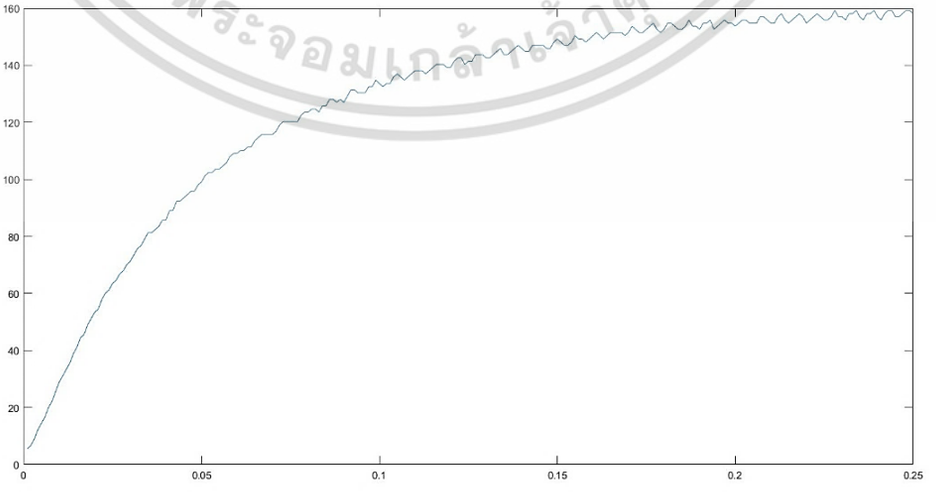


รูปที่ 3.41 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 60%

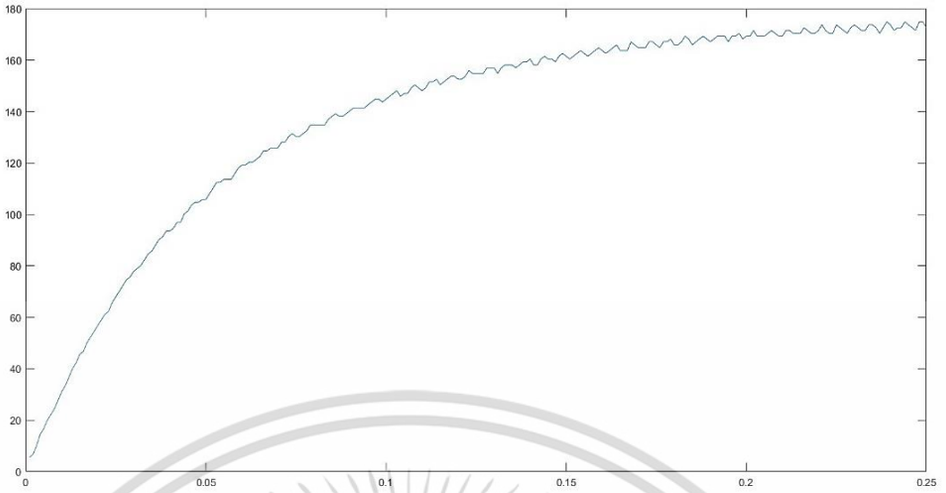


รูปที่ 3.42 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 70%

รูปที่ 3.43 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 80%



รูปที่ 3.44 แสดงกราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM เท่ากับ 90%

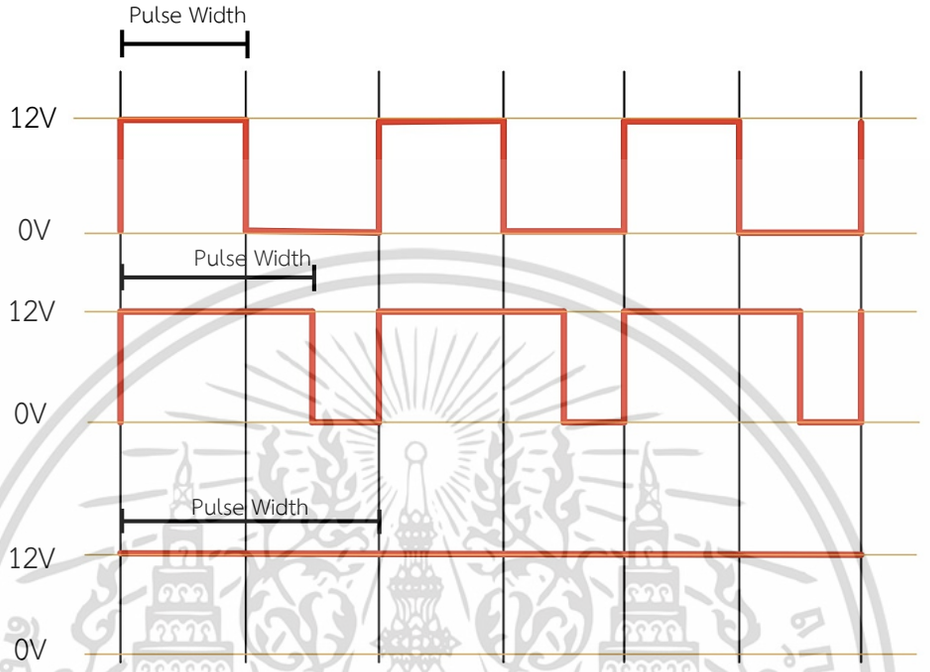


รูปที่ 3.45 กราฟความเร็วเอาต์พุตของมอเตอร์ที่ PWM = 100%

3.2.2.5 สรุปผลการทดลองการควบคุมความเร็วของมอเตอร์แบบวงเปิด

1) การควบคุมความเร็วของมอเตอร์แบบวงเปิด (Open Loop Control) คือ การควบคุมที่ไม่มีผลการป้อนกลับ (Feedback) มาจากระบบ เป็นเพียงการสั่งไปในทิศทางเดียวเท่านั้น ไม่สามารถรู้ได้ว่าเอาต์พุตที่ได้นั้นมีความผิดพลาดมากน้อยเพียงใด
2) การควบคุมมอเตอร์ไฟฟ้ากระแสตรงนั้นไม่สามารถทำได้โดยตรง กล่าวคือ จะไม่สามารถต่อไมโครคอนโทรลเลอร์ไปยังมอเตอร์โดยตรงได้ เนื่องจากเอาต์พุตที่ออกจากไมโครคอนโทรลเลอร์นั้นมีค่าที่น้อยมาก ทำให้มอเตอร์ไม่สามารถทำงานได้ อีกทั้งการจะทำให้ความเร็วของมอเตอร์นั้นคงที่ต้องการสัญญาณที่เสถียร ซึ่งสัญญาณที่ออกจากไมโครคอนโทรลเลอร์มีเสถียรภาพต่ำ ทำให้ไม่เหมาะสมที่จะต่อโดยตรง
3) การควบคุมมอเตอร์กระแสตรงนั้น สามารถทำได้โดยใช้อุปกรณ์ที่เรียกว่า บอร์ดขับมอเตอร์ (Motor Drive Board) ซึ่งจะเปลี่ยนค่าเอาต์พุตจากไมโครคอนโทรลเลอร์ไปเป็นความต่างศักย์ไฟฟ้า เพื่อควบคุมมอเตอร์ โดยอาศัยหลักการที่เรียกว่า PWM (Pulse Width Modulation) หรือการใช้ความถี่พัลส์ (Pulse) ที่สามารถเปลี่ยนแปลงความกว้างของสัญญาณได้ หลักการนี้จะสร้างสัญญาณรูปสี่เหลี่ยมที่มีค่าสูง (High) หรือต่ำ (Low) โดยจะจำลองเป็นความต่างศักย์ไฟฟ้าที่ 12 โวลต์ และ 0 โวลต์ ตามลำดับ กล่าวคือ นำความต่างศักย์ไฟฟ้ามาเฉลี่ยเป็น 0 ถึง 100% เพื่อควบคุมมอเตอร์ สามารถเห็นได้ดังรูปที่ 3.36 เมื่อกำหนดค่า PWM ที่ 10% ที่บอร์ดขับมอเตอร์ จะวัดได้ 1.195 โวลต์ ซึ่งเป็นไปตามทฤษฎี คือ 10% ของ 12 โวลต์ มีค่าเท่ากับ 1.2 โวลต์ ค่าอาจจะคลาดเคลื่อนเนื่องจากภายใน Multimeter มีค่าความต้านทานอยู่และความเสื่อมสภาพของเครื่องมือ เช่นเดียวกับ ค่าความต่างศักย์ไฟฟ้าที่ ค่า PWM 20%, 30%,

40%, 50%, 60%, 70%, 80%, 90% และ 100% ซึ่งแสดงดังรูปที่ 3.37, 3.38, 3.39, 3.40, 3.41, 3.42, 3.43, 3.44 และ 3.45 ตามลำดับ ซึ่งเป็นไปตามทฤษฎี



รูปที่ 3.46 แสดงตัวอย่างความกว้างของสัญญาณ PWM ที่ 50%, 75% และ 100%

4) ความเป็นเชิงเส้นของระบบ (Linearity) ในระบบทางพลศาสตร์ (Dynamic System) ส่วนใหญ่จะมีความเป็นเชิงเส้น (Linear) มีเพียงส่วนน้อยเท่านั้นที่ไม่เป็นเชิงเส้น (Non Linear) ในการควบคุมระบบเหล่านี้ก่อนการควบคุมจะมีการวิเคราะห์ความเป็นเชิงส้นของระบบก่อน เป็นการวิเคราะห์เบื้องต้น เพื่อหาวิธีการควบคุมที่เหมาะสม เนื่องจากระบบที่เป็นเชิงเส้นและไม่เป็นจะมีวิธีการที่ต่างกัน ในทางทฤษฎีที่กล่าวไว้ในบทที่ 2 เมื่อ X คือ อินพุตของระบบ และ Y คือ เอาต์พุตของระบบ หากระบบมีความเป็นเชิงเส้น เมื่อใส่ค่าอินพุตเข้าไปขนาด X เอาต์พุตที่ได้จะต้องมีค่าเท่ากับ Y และเมื่ออินพุตมีขนาด 2X เอาต์พุตที่ได้จะต้องมีขนาด 2Y ด้วยเช่นกัน และจะต้องมีความสัมพันธ์แบบนี้ทุกค่าของอินพุต กล่าวคือ ถ้ามอเตอร์มีความเร็วสูงสุดที่ 100 รอบต่อนาที เมื่อใส่อินพุตที่ 10% ความเร็วที่วัดได้ต้องมีค่าเท่ากับ 10 รอบต่อนาที ด้วยเช่นกัน จากผลการทดลองที่ได้ทำการเก็บค่าจำนวน 250 ค่า ที่คาบการสุ่ม 1 มิลลิวินาที ค่าของความเร็วที่ PWM 10% มีค่าประมาณ 15 รอบต่อนาที ซึ่งมอเตอร์ที่ใช้ทดลองมีค่าความเร็วสูงสุดคือ 170 รอบต่อนาที และที่ค่า PWM อื่นๆ ผลลัพธ์ที่ได้ก็มีความใกล้เคียงกัน จึงสรุปได้ว่าระบบของมอเตอร์ตัวนี้มีความเป็นเชิงเส้น (Linearity)

5) การคำนวณความเร็วของมอเตอร์สามารถคำนวณได้จากสมการด้านล่าง

$$Speed(RPM) = (\frac{EncoderValue}{GearRatio * PPR * SamplingTime}) * 60$$

เมื่อ Speed คือ ความเร็วของมอเตอร์ มีหน่วยเป็น RPM (Round per Minute)

Encoder Value คือ ค่าที่ได้จาก Register ของไมโครคอนโทรลเลอร์

Gear Ratio คือ อัตราทดเกียร์ของมอเตอร์

PPR คือ ความละเอียดของเอ็นโค้ดเดอร์ มีหน่วยเป็น PPR (Pulse per Rotation)

Sampling Time คือ คาบการสุ่มหรือช่วงเวลาที่เก็บค่าจากเอ็นโค้ดเดอร์เพื่อมาคำนวณ มีหน่วยเป็นวินาที

การคำนวณค่าความเร็วของมอเตอร์นั้น จำเป็นต้องรู้ค่า Encoder Value หรือจำนวนพัลส์ (Pulse) โดย Counter Register ใน Timer จะทำหน้าที่นับจำนวนพัลส์แล้วนำค่าที่ได้ไปหารด้วยอัตราทดเกียร์, ความละเอียดของเอ็นโค้ดเดอร์ และคาบการสุ่ม ที่กำหนดค่าได้จากการทดลองที่ 3.2.1 ค่าที่คำนวณได้จะมีหน่วยเป็น RPS หรือ รอบต่อวินาที จึงคูณด้วย 60 เพื่อแปลงหน่วยเป็น RPM หรือรอบต่อนาที

3.2.3 การหาฟังก์ชันถ่ายโอนของระบบ (Transfer Function)

การควบคุม (Control) เป็นศาสตร์ที่มีความสำคัญและหลากหลายเป็นอย่างมาก เนื่องจากเป็นวิชาพื้นฐานในทางวิศวกรรม ที่วิศวกรจำเป็นต้องมีทักษะและองความรู้ในวิชานี้ โดยการทำงานของอุปกรณ์ต่างๆ ไม่ว่าจะเป็นอุปกรณ์เครื่องมือวัด (Instrument), เครื่องจักร (Machine) และอุปกรณ์อีกมากมาย ล้วนใช้การควบคุมทั้งสิ้น ในโครงงานนี้ การควบคุมถือเป็นหัวใจหลักในการทำงาน เนื่องจากการออกแบบตัวควบคุม (Controller) นั้นจำเป็นต้องเข้าใจพื้นฐานการควบคุม, ส่วนประกอบ, ชนิด และหน้าที่ของตัวควบคุมแต่ละตัว

ในวิชาการควบคุม (Control) ส่วนสำคัญที่จะสามารถควบคุมระบบได้ จำเป็นต้องรู้เกี่ยวกับฟังก์ชันถ่ายโอน (Transfer Function) ก่อน เนื่องจากเป็นสิ่งที่แทนคุณลักษณะของระบบ โดยวิธีการหาจะมีหลายวิธีด้วยกัน แต่ในโครงงานฉบับนี้จะใช้วิธีการที่เรียกว่า การหาคุณลักษณะเฉพาะของระบบ (System Identification) โดยได้สร้างการทดลองขึ้นมา เพื่อศึกษาวิธีการหาฟังก์ชันถ่ายโอนของมอเตอร์ไฟฟ้ากระแสตรง

3.2.3.1 จุดประสงค์ของการทดลองการหาฟังก์ชันถ่ายโอนของระบบ

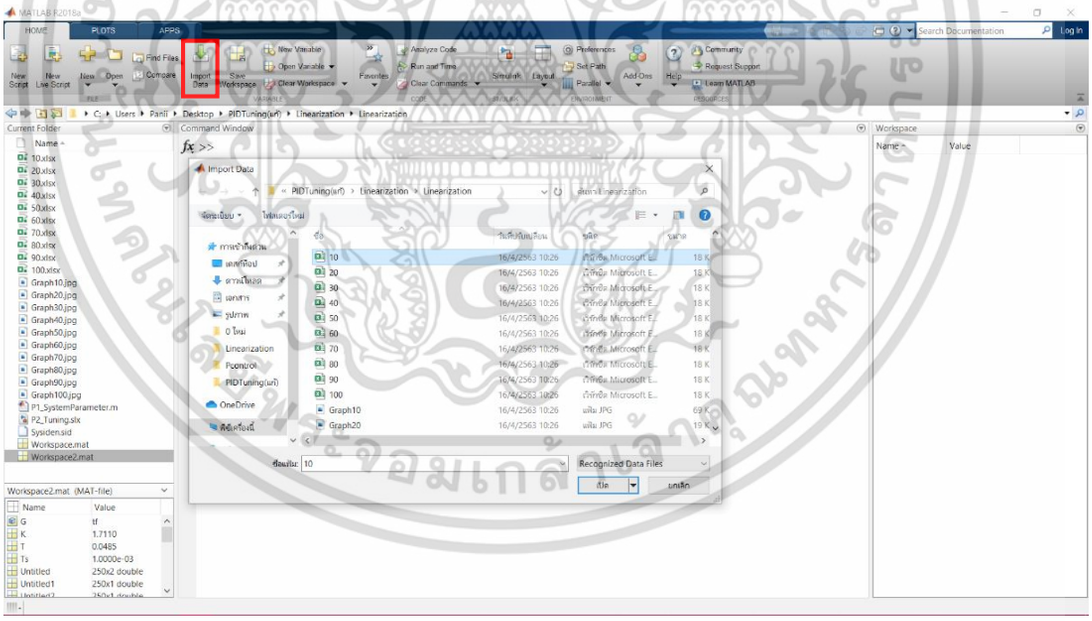
1) เพื่อหาฟังก์ชันถ่ายโอน (Transfer Function) ของระบบ
2) เพื่อศึกษาเกี่ยวกับ วิธีการหาคุณลักษณะเฉพาะของระบบ (System Identification)
3) เพื่อศึกษาการใช้งาน System Identification Toolbox ในโปรแกรม MATLAB

3.2.3.2 อุปกรณ์ที่ใช้ในการทดลองการหาฟังก์ชันถ่ายโอนของระบบ

1) คอมพิวเตอร์ที่ติดตั้งโปรแกรม MATLAB เอาไว้

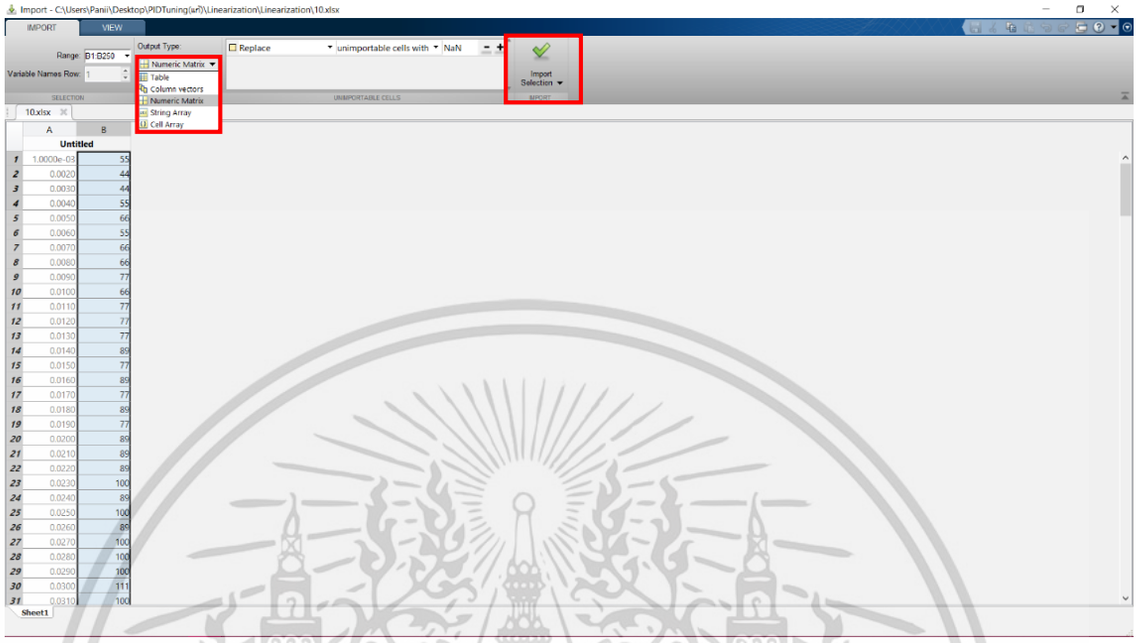
3.2.3.3 วิธีการทำการทดลองการหาฟังก์ชันถ่ายโอนของระบบ

1) ทำการ Import ข้อมูลตัวอย่างเข้าไปในโปรแกรม MATLAB โดยใช้คำสั่ง Import ดังแสดงในรูปที่ 3.47 โดยจะนำข้อมูลใส่ในโปรแกรม Excel เพื่อให้ง่ายต่อการจัดการ คำสั่ง Import นั้นจะอยู่ในส่วนของแถบ Manu Bar ซึ่งแสดงอยู่ในกรอบสี่เหลี่ยมสีแดง สำหรับข้อมูล ที่ทำการ import เข้ามาในโปรแกรมนั้น จำเป็นต้องรู้ช่วงเวลาของการเก็บข้อมูลแต่ละค่าหรือคาบการสุ่ม (Sampling Time) โดยข้อมูลตัวอย่างที่ใช้ในการทดลองนี้จะมีคาบการสุ่มเท่ากับ 1 มิลลิวินาที และมีจำนวน 250 ค่า



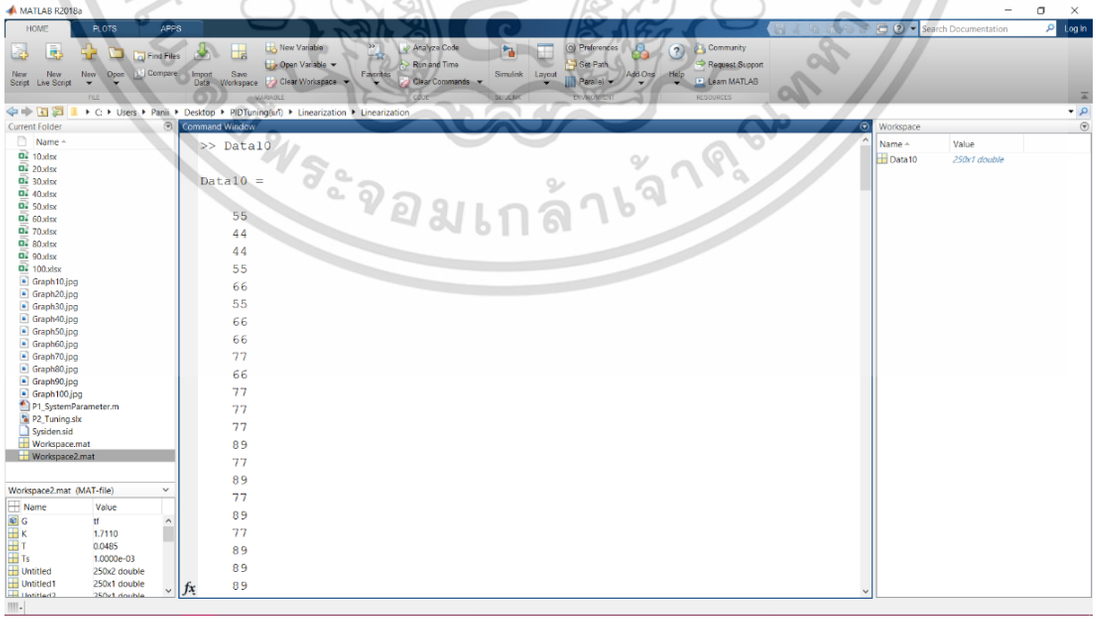
รูปที่ 3.47 แสดงคำสั่ง Import ในโปรแกรม MATLAB

2) เลือกชนิดของข้อมูลเป็น Numeric Matrix หลังจากนั้นคลิกที่เครื่องหมายถูกสีเขียว ดังรูปที่ 3.48



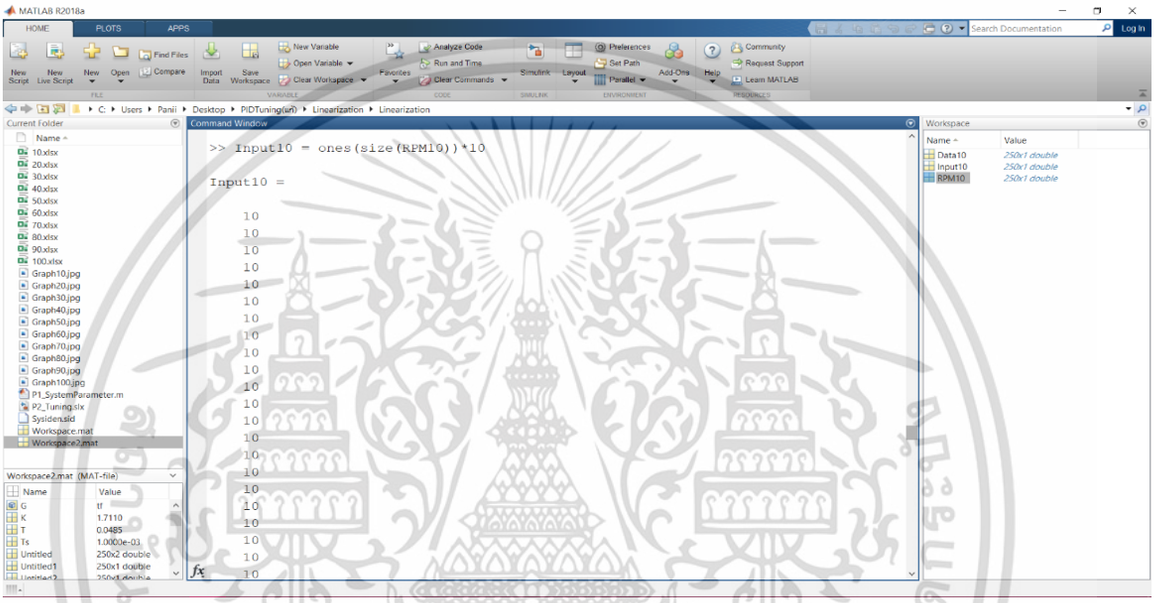
รูปที่ 3.48 แสดงหน้าต่างการ Import ข้อมูลเข้าโปรแกรม MATLAB

3) เมื่อ import ข้อมูลเข้ามาแล้ว จะไปปรากฏที่หน้าต่าง Workspace ในหน้าต่างดังรูปที่ 3.49 เพื่อจะนำไปใช้เป็นเอาต์พุตในการหาฟังก์ชันถ่ายโอนของระบบ โดยจะแสดงชื่อ, ชนิด และขนาดของตัวแปรแต่ละตัว



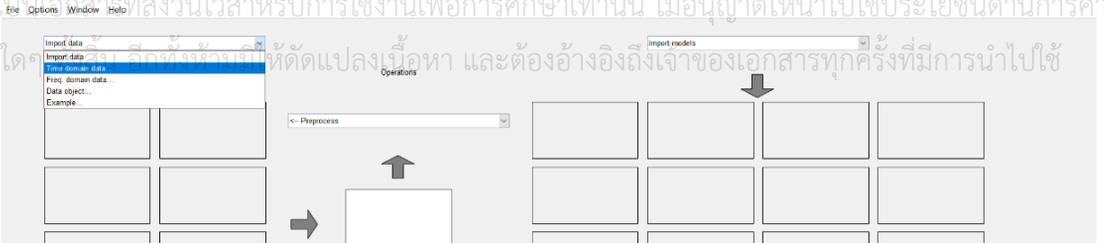
รูปที่ 3.49 แสดง Workspace ที่แสดงตัวแปรที่ import เข้ามา

4) ทำการสร้างตัวแปรอินพุต ซึ่งตัวแปรจะมีค่าเท่ากับ % ของ PWM ที่ทำการสั่ง เช่น หากข้อมูลที่นำเข้ามาเป็นการสั่งให้มอเตอร์หมุนที่ 10% จะต้องสร้างตัวแปรอินพุต ที่มีขนาด 10 จำนวน 250 ค่า ตามจำนวนข้อมูลของเอาต์พุต โดยใช้คำสั่ง VariableName = ones(size(DATA)) เมื่อ VariableName คือ ชื่อตัวแปรที่ต้องการสร้าง, ones() คือ คำสั่งสร้างตัวแปรที่มีจำนวนคอลัมน์เท่ากับ 1 และ size(DATA) คือ คำสั่งที่เก็บค่าขนาดของข้อมูล เมื่อรวมคำสั่งคือสร้างตัวแปรชื่อ VariableName ที่มีขนาดคอลัมน์เท่ากับ 1 และมีจำนวนข้อมูล 250 ตัว



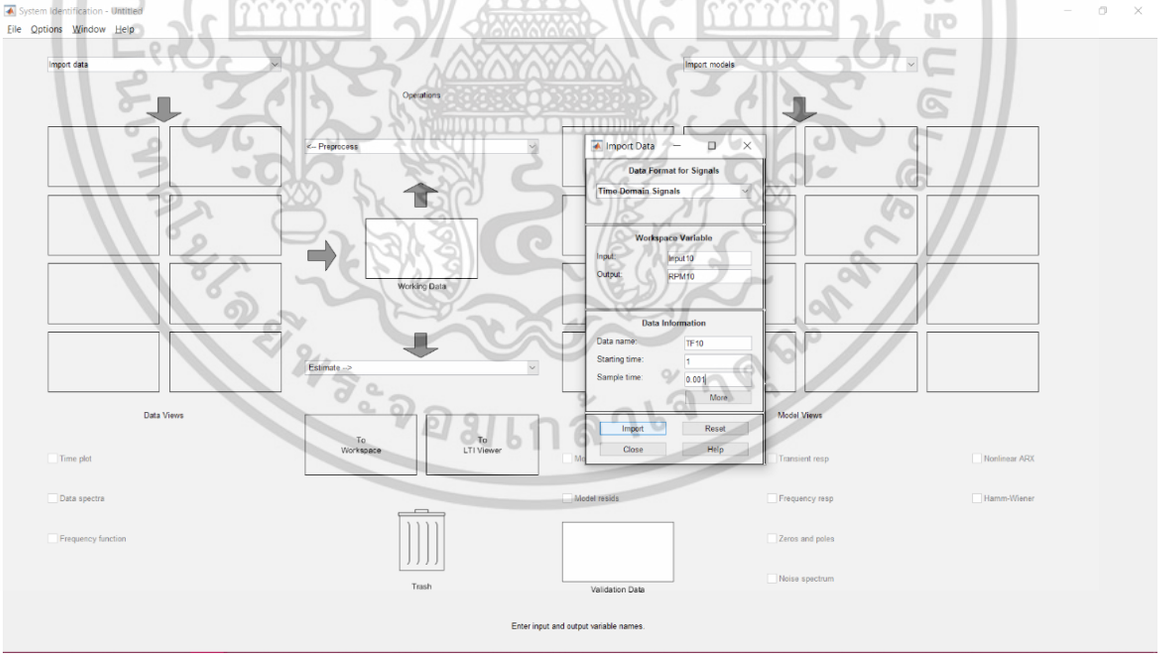
รูปที่ 3.50 แสดงการสร้างตัวแปร input ขนาด 1 คอลัมน์จำนวน 250 ตัว

5) ทำการเลือก System Identification Toolbox ที่อยู่ในแถบ APPS ของโปรแกรม MATLAB จะปรากฏหน้าต่างดังรูปที่ 3.51



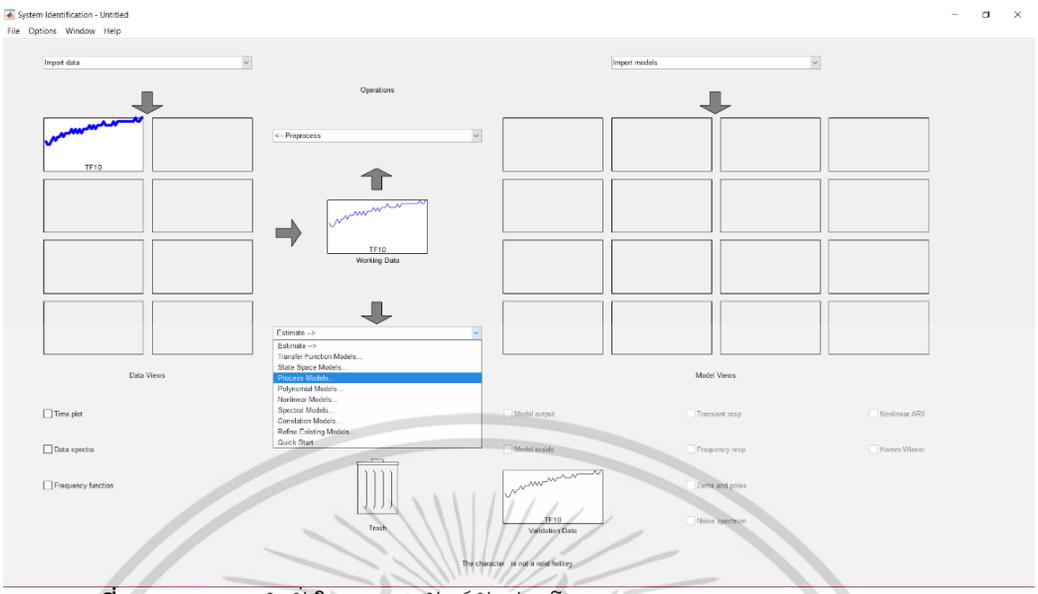
รูปที่ 3.51 แสดงหน้าต่างของ System Identification Toolbox

6) ทำการ Import ข้อมูลอินพุตและเอาต์พุต ที่สร้างขึ้นในขั้นตอนที่ 4 และ 3 ตามลำดับ หลังจากนั้นใส่ค่า คาบการสุ่มที่ทำการเก็บข้อมูลมาในช่อง Sample Time และกำหนดชื่อของ Data ที่จะสร้างขึ้น จากนั้นเลือก Import



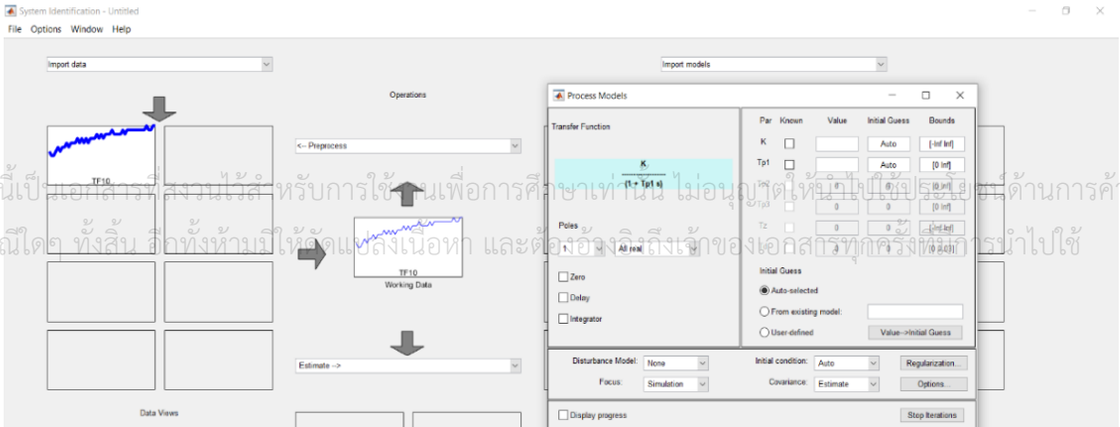
รูปที่ 3.52 แสดงการ Import ข้อมูลอินพุตและเอาต์พุต เข้าไปยัง System identification Toolbox

7) ข้อมูลที่ Import จะไปปรากฏในส่วนของ Data Views หลังจากนั้นในแถบ Estimate เลือกคำสั่ง Process Models เพื่อหาฟังก์ชันถ่ายโอน (Transfer Function) ของระบบ

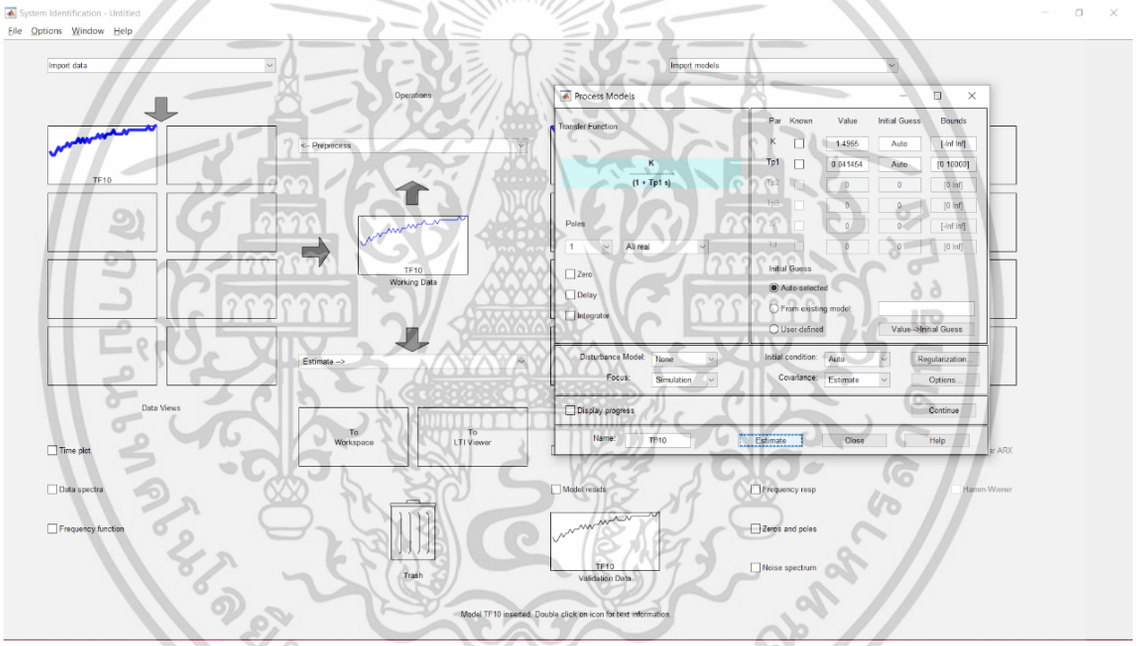


รูปที่ 3.53 แสดงคำสั่งในการหาฟังก์ชันถ่ายโอน (Transfer Function) ของระบบ

8) หลังจากเลือกคำสั่ง Process Models จะปรากฏหน้าต่างที่แสดงส่วนประกอบของฟังก์ชันถ่ายโอน (Transfer Function) ที่ต้องการ โดยสามารถเพิ่มจำนวน Pole และ Zero ได้ รวมไปถึงการหน่วงเวลา (Delay) ในการทดลองนี้จะกำหนดให้มี 1 Pole ไม่มี Zeros และการหน่วงเวลา เพื่อให้ง่ายในการทดลอง ทำการตั้งชื่อในช่อง Name และเลือก Estimate หน้าต่างจะแสดงค่า K (Gain) และ Tp1 ของระบบขึ้นมา ดังรูปที่ 3.54



รูปที่ 3.54 แสดงการเลือกลักษณะของฟังก์ชันถ่ายโอน (Transfer Function) ที่ต้องการสร้าง



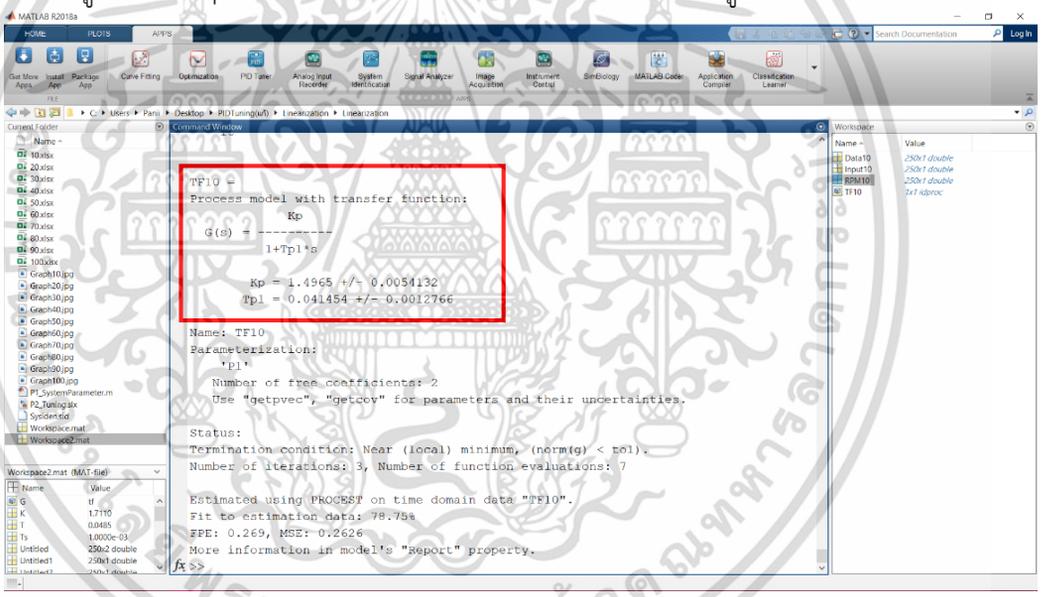
รูปที่ 3.55 แสดงค่า Gain และ Tp1 ที่ได้จากการหา Process Models

9) ฟังก์ชันถ่ายโอน (Transfer Function) ของระบบจะแสดงในแถบ Model Views หลังจากนั้นทำการเลือกปุ่ม Model Output จะปรากฏหน้าต่างที่แสดงค่าว่าฟังก์ชันถ่ายโอนที่ได้ ทำการประมาณค่ามานั้นมีความใกล้เคียง (Fits) กับความเป็นจริงมากน้อยเพียงใด หลังจากนั้นลากกรอบของฟังก์ชันที่ได้จากการประมาณค่าไปวางที่ช่อง To Workspace เพื่อนำเข้า Workspace และนำไปใช้งานต่อไป

รูปที่ 3.56 แสดงค่าความเข้ากันได้ของ Process Model ที่ได้จากการประมาณค่ากับความเป็นจริง

3.2.3.4 ผลการทดลองการหาฟังก์ชันถ่ายโอนของระบบ

หลังจากการใช้ System Identification Toolbox จะได้ฟังก์ชันถ่ายโอน (Transfer Function) ของระบบที่ทำการประมาณค่าออกมา ซึ่งมีจำนวน 1 Pole ซึ่งชื่อของฟังก์ชันถ่ายโอนแสดงอยู่ใน Workspace และ Command Window ดังแสดงในรูปที่ 3.57



รูปที่ 3.57 แสดงฟังก์ชันถ่ายโอน (Transfer Function) ของระบบที่ได้จากประมาณค่า

3.2.3.5 สรุปผลการทดลองการหาฟังก์ชันถ่ายโอนของระบบ

1) การหาฟังก์ชันถ่ายโอน (Transfer Function) ของระบบใดๆ มีตัวแปรที่สำคัญ 3 อย่างคือ อินพุต, เอาต์พุตและคาบการสุ่ม (Sampling Time) ดังนั้น คาบการสุ่มจำเป็นต้องมีความแม่นยำ กล่าวคือมีช่วงเวลาที่เก็บข้อมูลเอาต์พุตเท่ากัน ในทุกๆ ช่วง หากเกิดความผิดพลาดจะทำให้การประมาณค่ามีความผิดพลาดตามไปด้วย เนื่องจากในการทดลองที่ 3.2.1 ได้มีการสร้างคาบการสุ่มที่แม่นยำขึ้นมาทำให้ข้อมูลเอาต์พุต ที่ได้มีความน่าเชื่อถือ สามารถนำไปใช้งานได้
2) การหาฟังก์ชันถ่ายโอน (Transfer Function) ของระบบโดยใช้ System Identification Toolbox นั้นมีความแม่นยำที่ดีมาก สังเกตได้จากรูปที่

3.56 มีความเข้ากันได้ถึง 78.75% ทำให้เป็นเครื่องมือที่ดีในการประมาณค่าลักษณะเฉพาะของระบบ

3.2.4 การจำลองตัวควบคุมแบบพี (P controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

สำหรับขั้นตอนการวิจัยในโครงงานนี้จะแบ่งเป็น 2 ส่วน คือ ส่วนการทดลองจริง (Experiment) และการจำลอง (Simulation) โดยก่อนจะทดลองลงบนมอเตอร์ของจริงนั้น ทางผู้จัดทำได้ทำการทดสอบการควบคุมด้วยโปรแกรม MATLAB ก่อน เพื่อดูผลการตอบสนองของระบบว่าเป็นไปตามที่ต้องการและสอดคล้องในทางทฤษฎีหรือไม่

ในขั้นตอนของการจำลอง (Simulation) ได้ใช้ Simulink Toolbox ในโปรแกรม MATLAB เนื่องจากสามารถจำลองระบบควบคุมได้ง่าย มีลักษณะเป็นกล่อง (Block) ซึ่งแต่ละกล่องจะมีการทำงานที่แตกต่างกันออกไป การควบคุม (Control) ก็เป็นความสามารถอีก 1 อย่างของ Simulink ที่สามารถจำลองได้ทั้งแบบเปิด (Open Loop Control) และแบบปิด (Closed Loop Control) ในการทดลองนี้จะเป็นการจำลองกาควบคุมแบบวงปิด (Closed Loop Control)

3.2.4.1 จุดประสงค์การทดลองการจำลองตัวควบคุมแบบพี (P controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

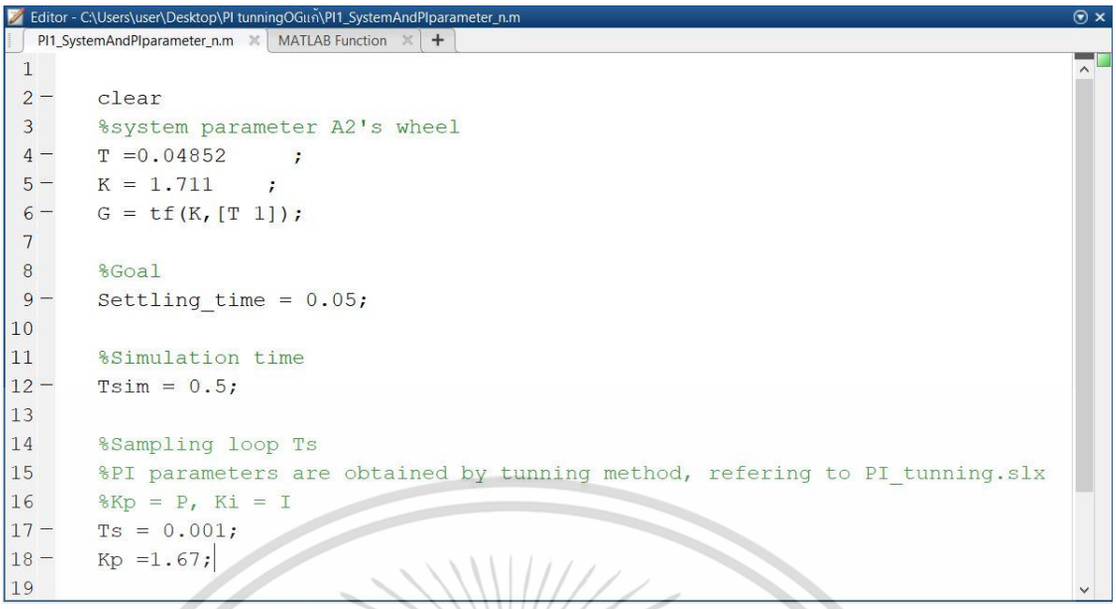
1) เพื่อศึกษาวิธีการใช้งาน Simulink Toolbox ในโปรแกรม MATLAB
2) เพื่อศึกษาวิธีการควบคุมความเร็วของมอเตอร์แบบวงปิด (Closed Loop Control)
3) เพื่อเรียนรู้การทำงานของตัวควบคุมแบบพี (P Controller)

3.2.4.2 อุปกรณ์การทดลองการจำลองตัวควบคุมแบบพี (P controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

1) คอมพิวเตอร์ที่ติดตั้งโปรแกรม MATLAB

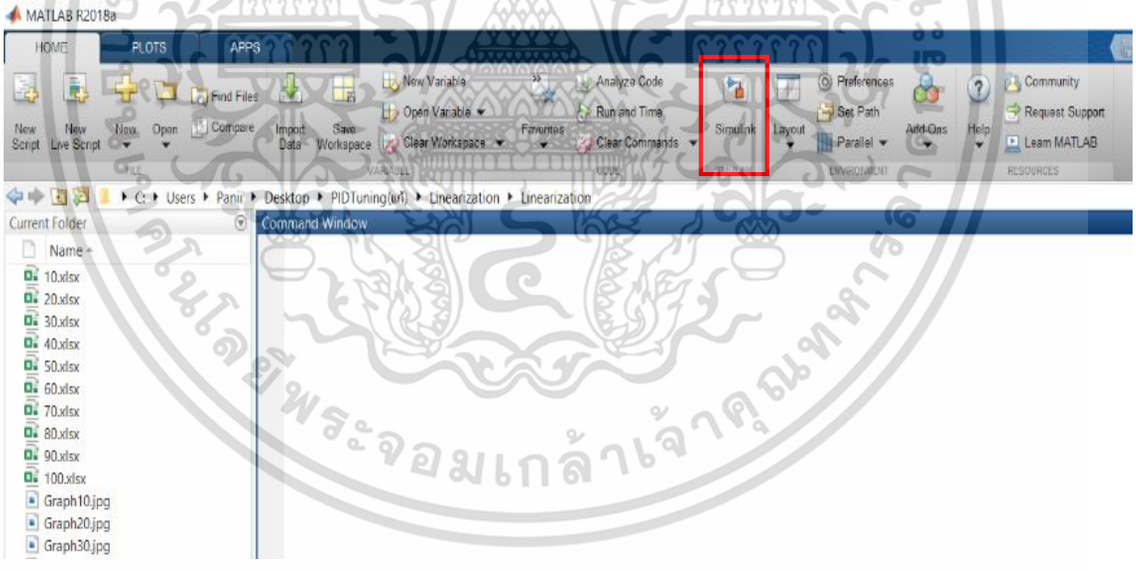
3.2.4.3 วิธีการทดลองการจำลองตัวควบคุมแบบพี (P controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

1) ทำการสร้างไฟล์ .m ไว้สำหรับแสดงฟังก์ชันถ่ายโอนของระบบที่ทำการจำลอง (G), คาบการสุ่มของข้อมูล (Ts), เวลาในการจำลอง (Tsim) และค่าที่ทำการประมาณของตัวควบคุมแบบพี (P controller)



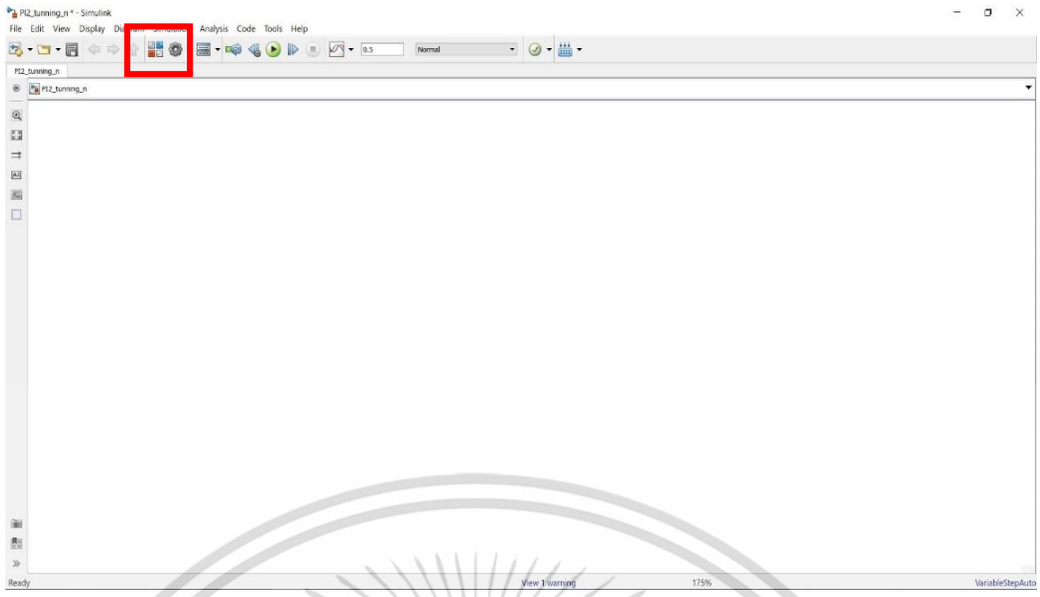
รูปที่ 3.58 แสดงการสร้างไฟล์ .m ในโปรแกรม MATLAB

2) หลังจากนั้นทำการเลือก Simulink Toolbox ในแถบ Home ของโปรแกรม MATLAB



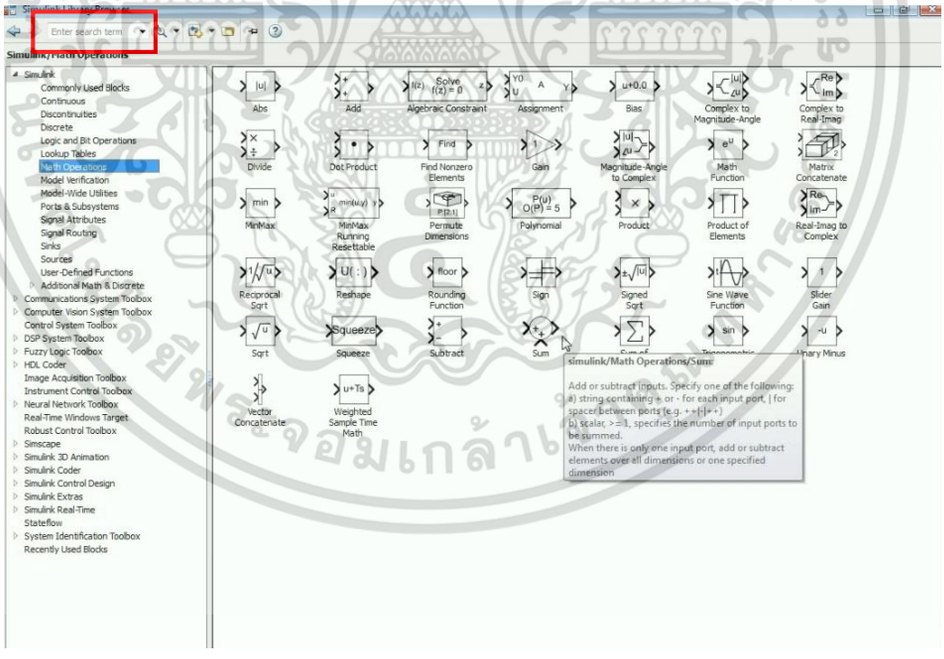
รูปที่ 3.59 แสดง Icon ของ Simulink Toolbox ในโปรแกรม MATLAB

3) จะปรากฏหน้าต่างของ Simulink Toolbox ขึ้นมา ให้เลือกแถบเมนู library ที่อยู่ในกรอบสีแดง ดังรูปที่ 3.60



รูปที่ 3.60 แสดงหน้าต่างของ Simulink Toolbox

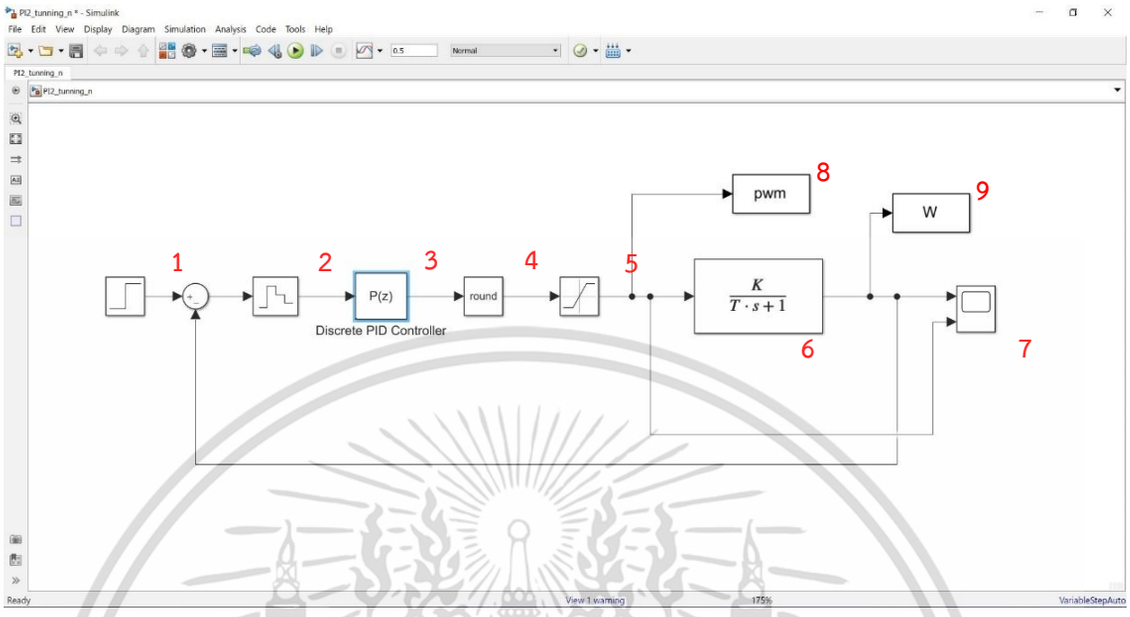
4) เมื่อเลือกคำสั่ง Library จะปรากฏหน้าต่างที่แสดง Block ชนิดต่างๆ ขึ้นมาโดยสามารถค้นหาชนิดของ Block ที่ต้องการได้จากแถบ Search เมื่อต้องการใช้งาน Block ชนิดนั้นๆ ให้ทำการลากมาไว้ที่หน้าต่างหลัก



รูปที่ 3.61 แสดงหน้าต่างของ Library ใน Simulink Toolbox

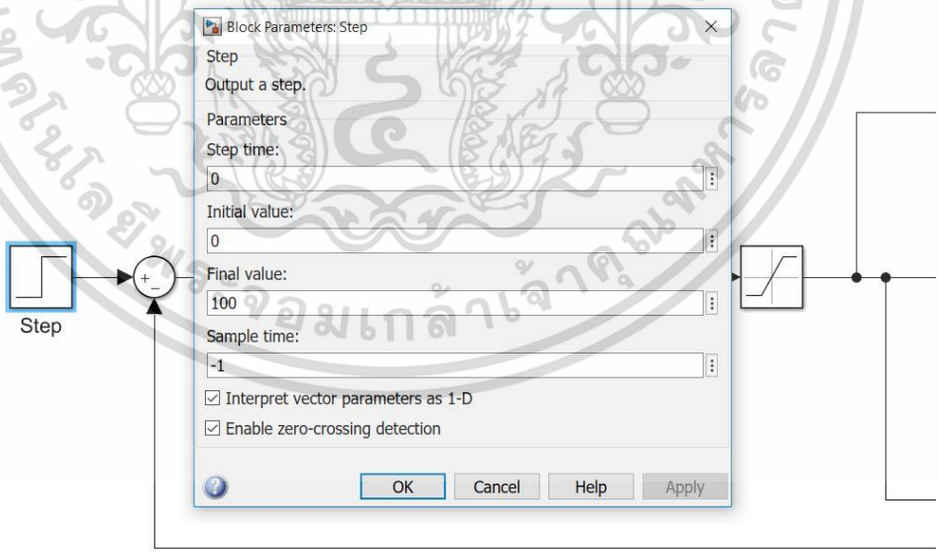
5) ทำการสร้างระบบควบคุมแบบปิด (Closed Loop Control) ขึ้นมา โดยจะประกอบไปด้วย Block ต่างๆ จำนวน 9 Block ดังนี้ Step Block, Zero-Order Hold Block, Discrete PID Controller Block, Round Block, Saturation Block, Transfer Function Block, Scope Block,

To Workspace Block (PWM) และ To Workspace Block (W) ตามลำดับ ดังแสดงในรูปที่ 3.62



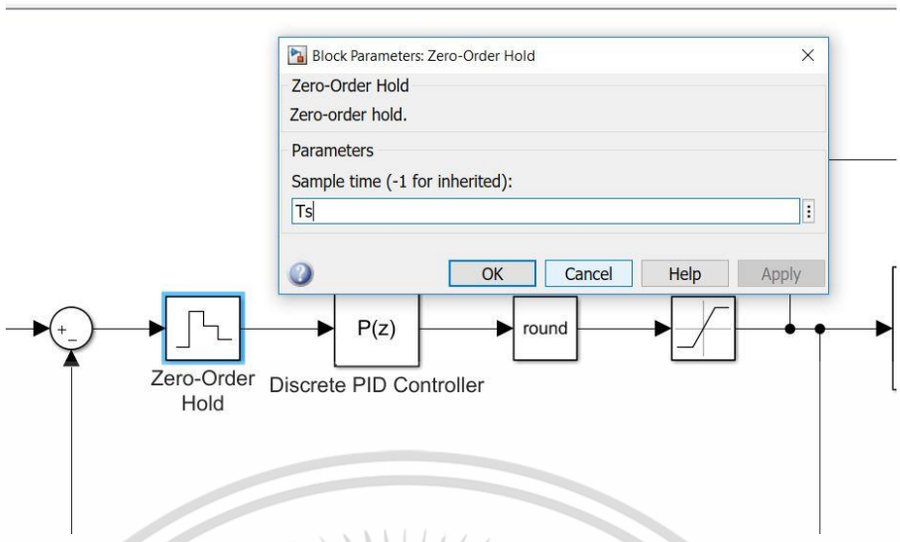
รูปที่ 3.62 แสดงการจำลองระบบควบคุมแบบปิดใน Simulink Toolbox

6) กำหนดค่าใน Step Block โดย Step time เท่ากับ 1 , Initial Value เท่ากับ 0 และ Final Value เท่ากับ 100



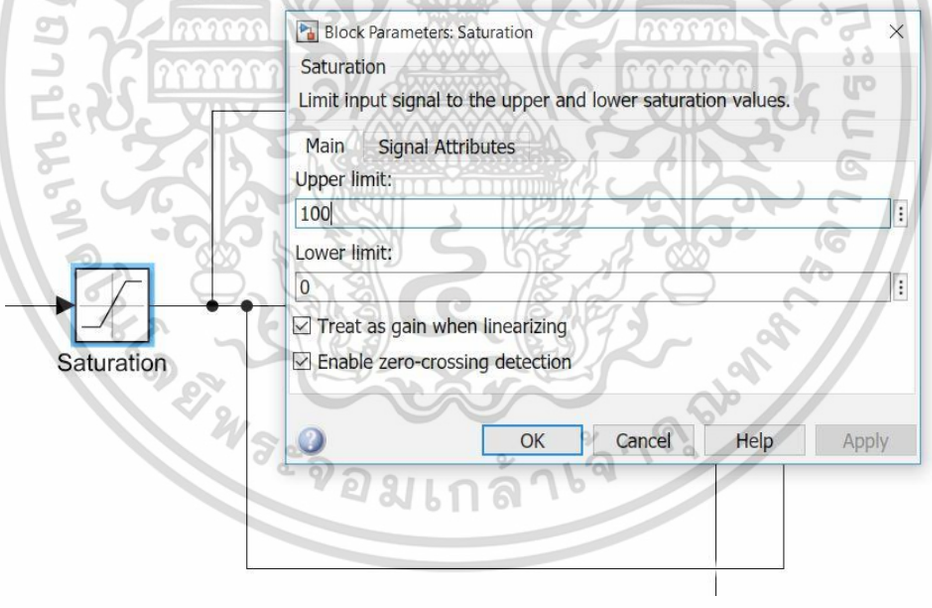
รูปที่ 3.63 แสดงการกำหนดค่าต่างๆ ใน Step Block

7) กำหนดค่า Sample Time ใน Zero-Order Hold Block ให้มีค่าเท่ากับคาบการสุ่มที่ใช้เก็บค่าเอาต์พุต



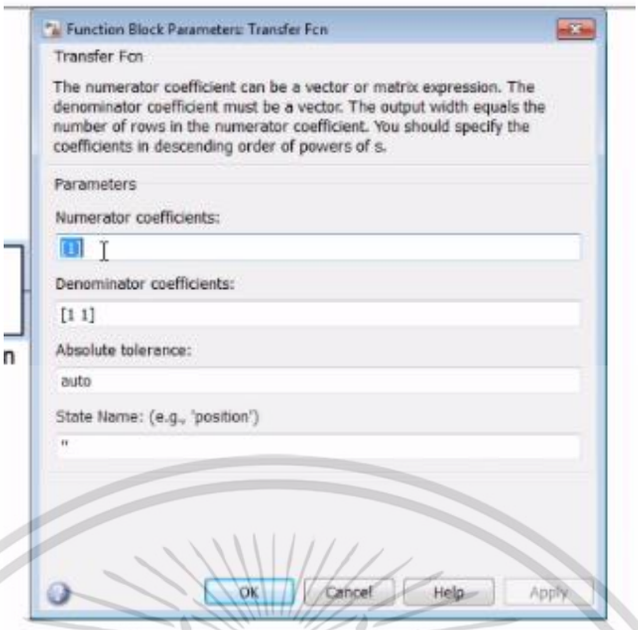
รูปที่ 3.64 แสดงการกำหนดค่า Sample Time ใน Zero-Order Hold Block

8) กำหนดค่าใน Saturation Block โดย Upper limit เท่ากับ 100 และ Lower limit เท่ากับ 0



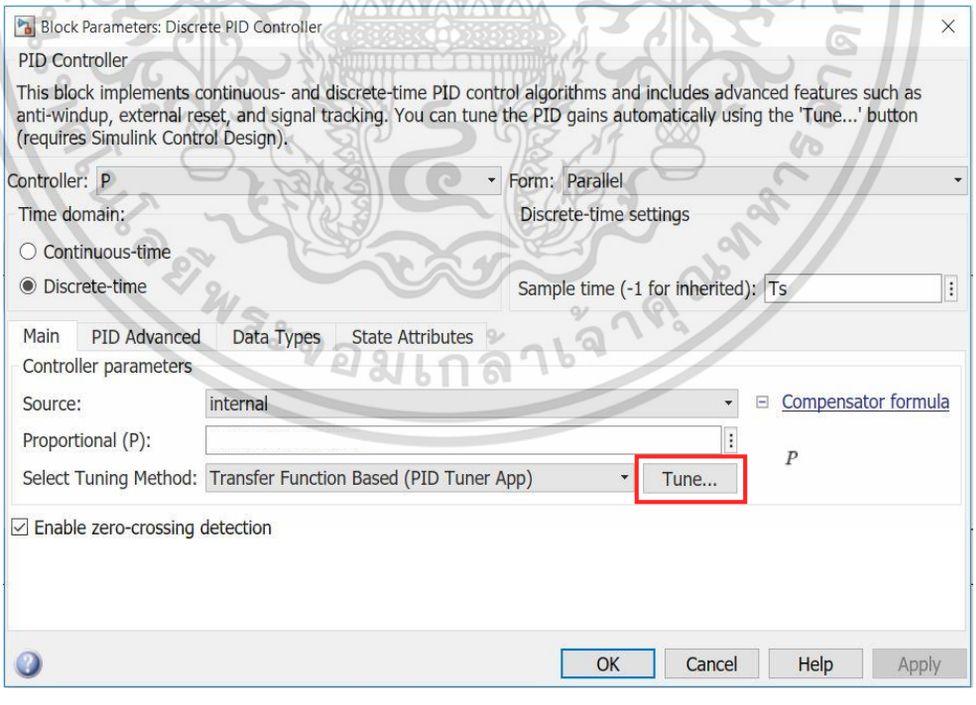
รูปที่ 3.65 แสดงการกำหนดค่า Saturation Block ใน Saturation Block

9) กำหนดฟังก์ชันถ่ายโอน (Transfer Function) ที่ต้องการจำลอง ในการทดลองนี้ ได้ใช้ฟังก์ชันถ่ายโอนที่ได้จากการทดลองที่ 3.2.3 โดย Numerator coefficients คือตัวแปร K และ Denominator coefficient คือตัวแปร T ดังรูปที่ 3.66



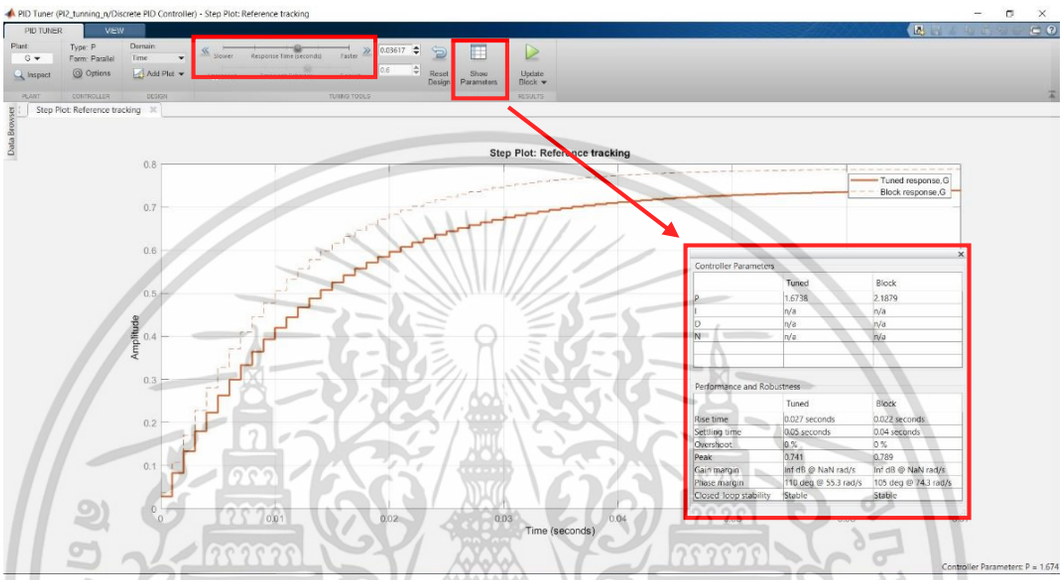
รูปที่ 3.66 แสดงการกำหนดค่าใน Transfer Function Block

10) กำหนดค่าใน Discrete PID controller Block โดยเลือกตัวควบคุมแบบพี (P Controller) และในช่อง Sample Time ให้ใส่ตัวแปรที่กำหนดเป็นคาบการสุ่ม (Ts) หลังจากนั้นให้เลือกคำสั่ง Tune



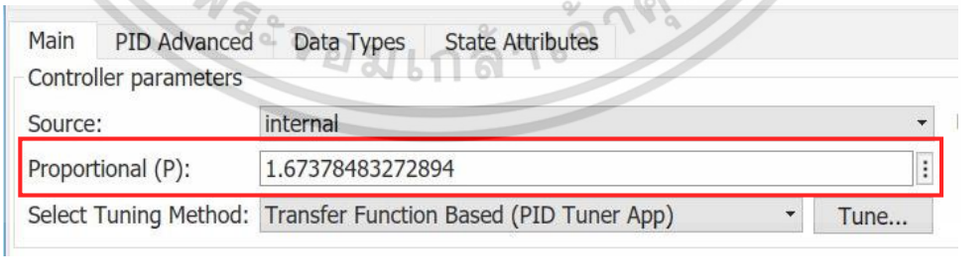
รูปที่ 3.67 แสดงหน้าต่าง Discrete PID Controller Block ของตัวควบคุมแบบพี

11) หลังจากเลือกคำสั่ง Tune จะปรากฏหน้าต่างของ PID Tuner Toolbox โดยจะมีแถบที่สามารถปรับให้ระบบมีผลตอบสนองที่เร็วขึ้นหรือช้าลงได้ตามความต้องการ หากต้องการดูค่าพารามิเตอร์ต่างๆ ของระบบให้เลือกเมนู Show Parameter หลังจากทำการปรับค่าเสร็จแล้วให้เลือก Update Block ในการทดลองต้องการให้ระบบเข้าสู่สมดุลที่เวลา 0.05 วินาที (Settling Time เท่ากับ 0.05 วินาที)



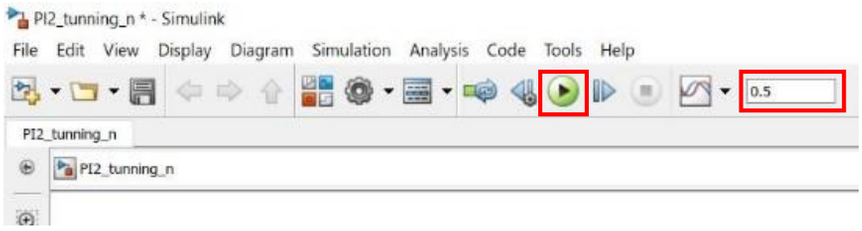
รูปที่ 3.68 แสดงการใช้งาน PID Tuner Toolbox

12) เมื่อเลือกคำสั่ง Update Block ใน PID Tuner Toolbox แล้วค่าตัวพารามิเตอร์ของตัวควบคุมแบบ P จะไปปรากฏที่ช่อง Proportional (P) ใน Discrete PID Controller Block (ค่า Kp ที่ได้จะนำไปใช้ในการทดลองจริงต่อไป)



รูปที่ 3.69 แสดงค่าพารามิเตอร์ของตัวควบคุมแบบพี

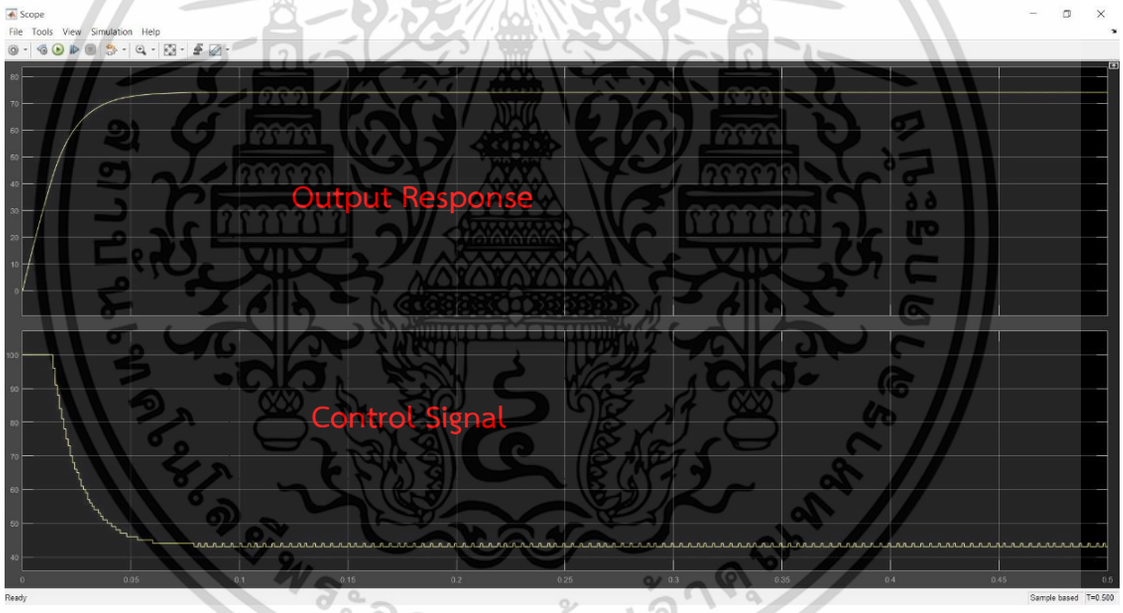
13) หลังจากกำหนดค่าตัวแปรทั้งหมดแล้ว ให้ทำการจำลอง (Simulation) โดยเลือกคำสั่ง Run และกำหนดช่วงเวลาในการจำลองได้ในการทดลองกำหนดเป็น 0.5 วินาที



รูปที่ 3.70 แสดงคำสั่งของการจำลองระบบควบคุมแบบวงปิด

14) เมื่อ Run โปรแกรมแล้ว ให้ทำการดับเบิลคลิกที่ Scope Block เพื่อดูผลลัพธ์ (ใน Scope Block แบ่งกราฟออกเป็น 2 ส่วน คือ ส่วนของผลตอบสนอง และส่วนของสัญญาณควบคุม)

3.2.4.4 ผลการทดลองการจำลองตัวควบคุมแบบพี (P controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB



รูปที่ 3.71 แสดงผลตอบสนองและสัญญาณควบคุมของระบบที่ใช้ตัวควบคุมแบบพี

จากรูปที่ 3.71 ในส่วนของผลตอบสนองนั้น จะสังเกตได้ว่าในตอนแรกจะมีค่าเพิ่มขึ้นอย่างรวดเร็ว หลังจากนั้นจะลดลงเรื่อยๆ จนเข้าสู่สภาวะสมดุลในที่สุด แต่ผลตอบสนองที่สภาวะสมดุลมีค่า 72 ไม่สามารถไปถึงค่า 100 ตามที่กำหนดใน Step Block ได้ อีกทั้งยังเข้าสู่สภาวะสมดุลที่เวลามากกว่า 0.05 วินาทีซึ่งมากกว่าที่ต้องการ

ในส่วนของสัญญาณควบคุม (Control Signal) ในช่วงแรกจะมีค่าคงที่เท่ากับ 100 ในช่วงระยะเวลาหนึ่ง หลังจากนั้นจะมีค่าลดลงเรื่อยๆ เนื่องจากผลตอบสนองของระบบใกล้ถึงค่าสูงสุดแล้วจนกระทั่งคงที่ที่ค่าเท่ากับ 45 ซึ่งสอดคล้องกับผลตอบสนองที่ในช่วงแรกต้องการค่าอินพุตหรือสัญญาณควบคุมที่มีค่ามาก ทำให้เพิ่มขึ้นอย่างรวดเร็วในตอนแรก หลังจากนั้นสัญญาณควบคุมมีค่าลดลงทำให้ผลตอบสนองมีค่าลดลงจนคงที่ในที่สุด

3.2.4.5 สรุปผลการทดลองการจำลองตัวควบคุมแบบพี (P controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

1) Simulink Toolbox ในโปรแกรม MATLAB สามารถจำลองการควบคุมได้อย่างมีประสิทธิภาพ เนื่องจากสามารถกำหนดค่าตัวแปรต่างๆ ได้ง่ายและมี Block ต่างๆ ให้เลือกใช้มากมาย แต่การใช้งาน Simulink Toolbox จำเป็นต้องมีพื้นฐานในการใช้งานระดับหนึ่ง โดยฟังก์ชันการทำงานของ Block แต่ละชนิดสามารถดูได้จากเว็บไซต์ของ MATLAB
2) จากผลตอบสนองในรูปที่ 3.71 ตัวควบคุมแบบพี (P Controller) ไม่สามารถทำให้ผลตอบสนองไปยังค่าที่ต้องการได้ เนื่องจากตัวควบคุมแบบพีเป็นการนำค่าความผิดพลาด (Error) ที่ได้ไปคูณกับค่าคงที่ของตัวควบคุมแบบพี (Kp) เพื่อให้ได้สัญญาณควบคุม ซึ่งในช่วงแรกค่า Set Point และค่าเอาต์พุตของระบบมีค่ามาก ทำให้ผลตอบสนองที่ได้มีค่ามาก (เพิ่มขึ้นอย่างรวดเร็ว) หลังจากนั้นเมื่อเอาต์พุตของระบบและค่า Set Point มีค่าใกล้เคียงกัน จะทำให้ผลตอบสนองมีค่าลดลงจนคงที่ ทำให้ไม่สามารถไปถึงค่าสูงสุดที่กำหนดไว้ได้ และใช้ระยะเวลานานในการทำให้ระบบเข้าสู่สภาวะสมดุล (Steady State) ดังแสดงในสมการที่ 3.1 และสมการที่ 3.2

$$u(t) = MV(t) = K_p * e(t)$$

สมการที่ 3.1 สมการสัญญาณเอาต์พุตของตัวควบคุมแบบพี

เมื่อ u(t) คือ เอาต์พุตของระบบ ในที่นี้คือสัญญาณควบคุมจากตัวควบคุมที่ส่งไปยังระบบ
K_p คือ ค่าคงที่ของตัวควบคุมแบบพี
e(t) คือ ค่าความผิดพลาดของระบบ

$$e(t) = SP - ActualValue$$

สมการที่ 3.2 สมการค่าความผิดพลาดของระบบ

เมื่อ SP คือ ค่า Set Point ของระบบ
Actual Value คือ เอาต์พุตที่ได้จากระบบ

3) การใช้ตัวควบคุมแบบพี (P Controller) เพียงชนิดเดียว ไม่เพียงพอที่จะควบคุมระบบให้มีประสิทธิภาพได้ เนื่องจากผลตอบสนองของระบบไม่สอดคล้องกับค่าที่ตั้งไว้ ควรจะใช้ตัวควบคุมชนิดอื่นร่วมด้วย

3.2.5 การจำลองตัวควบคุมแบบพี-ไอ (PI controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

จากการทดลองที่ 3.2.4 พบว่า การใช้ตัวควบคุมแบบพี (P Controller) เพียงชนิดเดียวจะไม่สามารถทำให้ระบบมีค่าตามที่ต้องการได้ จึงมีการสร้างการทดลองนี้ขึ้น เพื่อดูประสิทธิภาพของตัวควบคุมแบบพีไอ (PI Controller) โดยรูปแบบการทดลองจะคล้ายกับการทดลองที่ 3.2.4 เพียงแต่เปลี่ยนชนิดของตัวควบคุมเท่านั้น

3.2.5.1 จุดประสงค์การทดลองการจำลองตัวควบคุมแบบพี-ไอ (PI controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

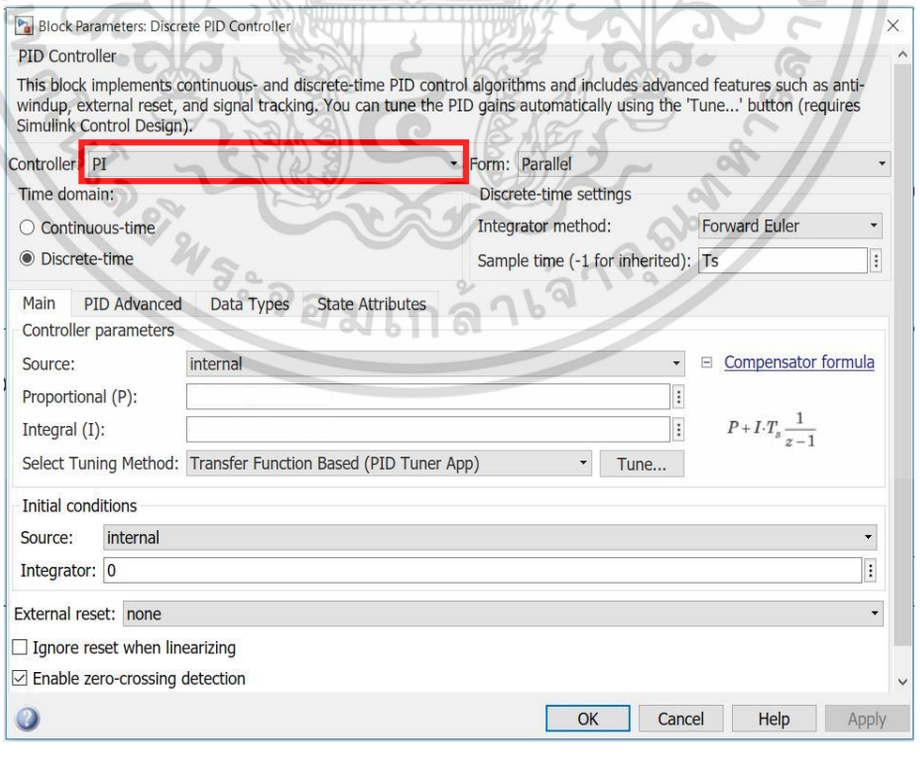
1) เพื่อเรียนรู้การทำงานของตัวควบคุมแบบพีไอ (PI Controller)
2) เพื่อเปรียบเทียบการทำงานของตัวควบคุมแบบพี (P) และพีไอ (PI)

3.2.5.2 อุปกรณ์ที่ใช้ในการทดลองการจำลองตัวควบคุมแบบพี-ไอ (PI controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

1) คอมพิวเตอร์ที่ติดตั้งโปรแกรม MATLAB

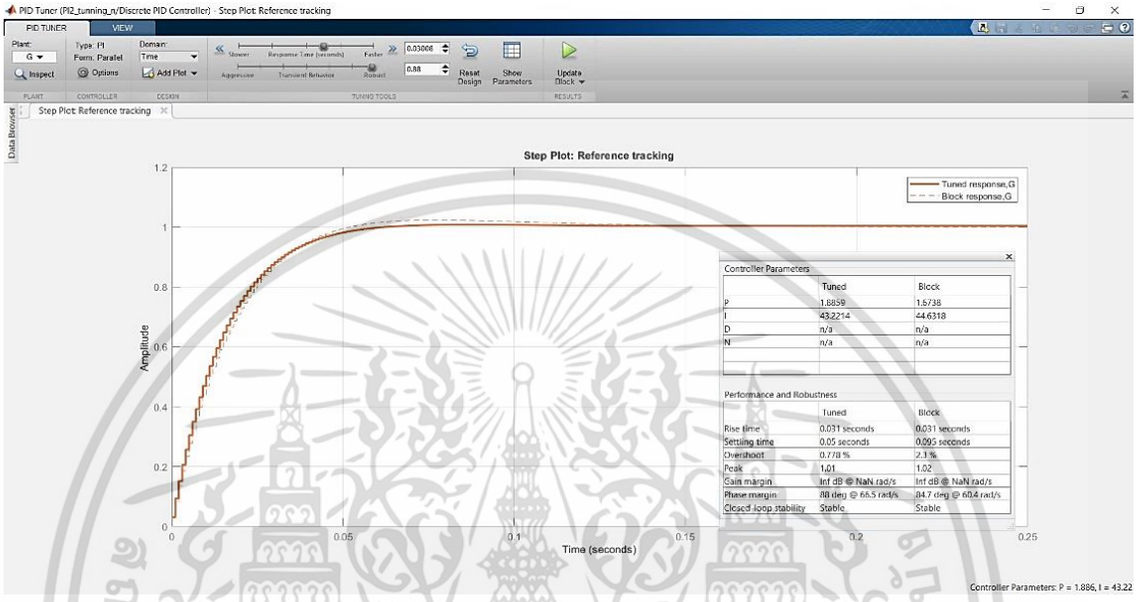
3.2.5.3 วิธีการทดลองการจำลองตัวควบคุมแบบพี-ไอ (PI controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

1) ขั้นตอนทั้งหมดจะเหมือนกับการทดลองที่ 3.2.4 เพียงแต่ ในขั้นตอนที่เลือกชนิดของตัวบควบคุมใน Discrete PID Controller Block ให้เลือกเป็นตัวควบคุมแบบพีไอ ดังรูป 3.72



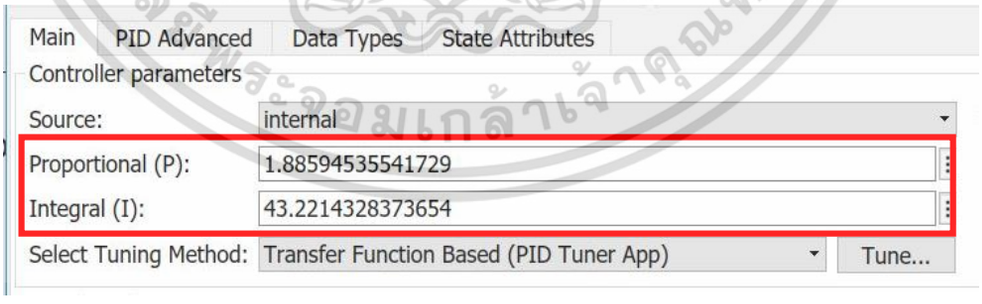
รูปที่ 3.72 แสดงหน้าต่าง Discrete PID Controller Block ของตัวควบคุมแบบพีไอ

2) เมื่อเลือกคำสั่ง Tune จะปรากฏ PID Tuner Toolbox ที่คล้ายกับการทดลองที่ 3.2.1 เพียงแต่จะมีแถบให้ปรับเพิ่มขึ้นมา 1 แถบ ไว้สำหรับปรับค่า Robust ของระบบ ว่าต้องการให้ระบบเข้าสู่สภาวะสมดุลเร็วขึ้นหรือช้าลง โดยต้องการให้ระบบเข้าสู่สภาวะสมดุลที่เวลา 0.05 วินาที (Settling Time เท่ากับ 0.05 วินาที)



รูปที่ 3.73 แสดงหน้าต่าง PID Tuner สำหรับตัวควบคุมแบบพีไอ

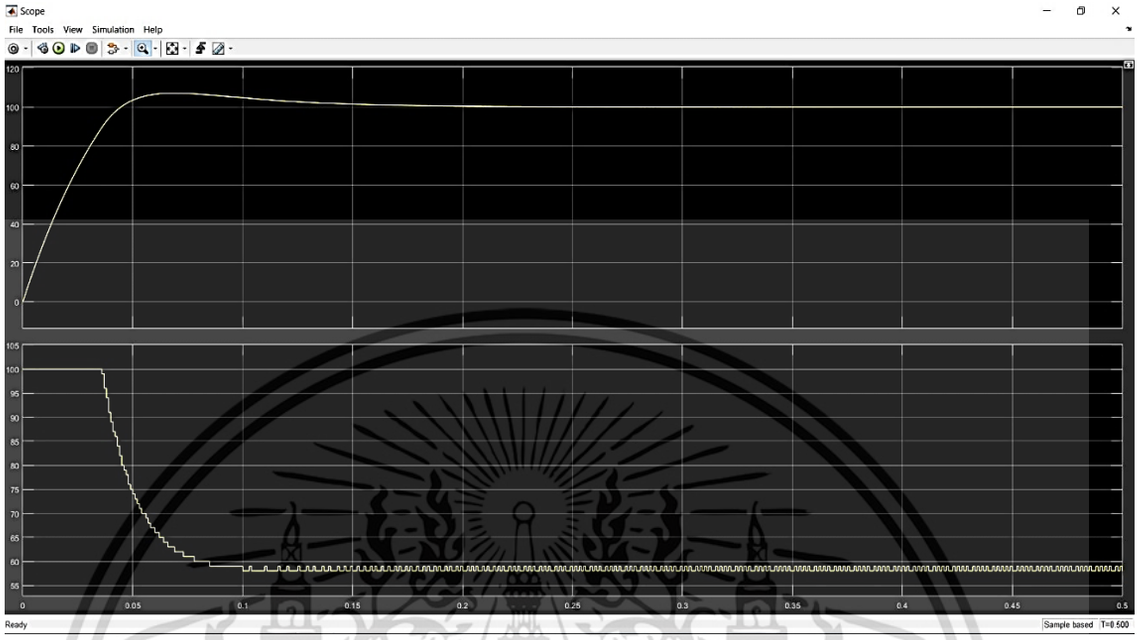
3) เมื่อทำการ Update Block จะมีค่าพารามิเตอร์ของตัวควบคุมแบบพีไอแสดงในส่วนของ Proportional (P) และ Integral (I) ใน Discrete PID Controller Block ดังรูปที่ 3.74



รูปที่ 3.74 แสดงค่าพารามิเตอร์ของตัวควบคุมแบบพีไอ

4) เมื่อทำการปรับค่าตามที่ต้องการแล้วเลือกคำสั่ง Run แล้วดูผลลัพธ์

3.2.5.4 ผลการทดลองการจำลองตัวควบคุมแบบพี-ไอ (PI controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB



รูปที่ 3.75 แสดงผลตอบสนองและสัญญาณควบคุมของระบบที่ใช้ตัวควบคุมแบบพีไอ

จากรูปที่ 3.75 ในส่วนของผลตอบสนองนั้นจะสังเกตได้ว่า ในตอนแรกจะมีค่าเพิ่มขึ้นอย่างรวดเร็วและจะมีค่ามากกว่าค่า Set Point ที่ได้ตั้งไว้ (มีค่าประมาณ 105) ในช่วงระยะเวลาหนึ่ง หลังจากนั้นจะค่อยๆ ลดลง จนกระทั่งมีค่าเท่ากับ Set Point ที่ได้ตั้งไว้ และเข้าสู่สภาวะสมดุลที่เวลามากกว่า 0.05 วินาที ซึ่งมีค่ามากกว่าที่ต้องการ

ในส่วนของสัญญาณควบคุม (Control Signal) ในช่วงแรกจะมีค่าคงที่เท่ากับ 100 ในช่วงระยะเวลาหนึ่ง ซึ่งนานกว่าสัญญาณควบคุมของตัวควบคุมแบบพี หลังจากนั้นจะมีค่าลดลงเรื่อยๆ เนื่องจากผลตอบสนองของระบบใกล้จะถึงค่าสูงสุดแล้วจนกระทั่งมีค่าคงที่ ที่ค่าเท่ากับ 57 ซึ่งสอดคล้องกับผลตอบสนอง ที่ในช่วงแรกต้องการค่าอินพุตหรือสัญญาณควบคุมที่มีค่ามาก ทำให้เพิ่มขึ้นอย่างรวดเร็วในตอนแรก หลังจากนั้นสัญญาณควบคุมจะมีค่าลดลงทำให้ผลตอบสนองมีค่าลดลงจนคงที่ในที่สุด

3.2.5.5 สรุปผลการทดลองการจำลองตัวควบคุมแบบพี-ไอ (PI controller) ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

1) ตัวควบคุมแบบพีไอ (PI Controller) สามารถทำให้ระบบมีค่าตามค่า Set Point ที่ตั้งไว้ได้ การใช้ตัวควบคุมแบบพีไอจึงมีประสิทธิภาพมากกว่าตัวควบคุมแบบพีเพียงอย่างเดียว
2) Saturation Block เปรียบเสมือนบอร์ดขับมอเตอร์หรือ Actuator ที่คอยแปลงสัญญาณจากตัวควบคุม ซึ่งก็คือไมโครคอนโทรลเลอร์ โดยที่กำหนด Upper limit เท่ากับ 100 และ Lower limit เท่ากับ 0 เป็นเพราะบอร์ดขับมอเตอร์สามารถรับสัญญาณ PWM ได้เพียง 0 ถึง 100% เท่านั้น หาก

ไม่ใส่ Saturation Block จะทำให้การจำลองเกิดความผิดพลาด เนื่องจากโปรแกรมจะไม่สามารถรับรู้ค่า limit ได้ ซึ่งอาจจะทำให้ผลตอบสนองมีความผิดพลาดได้

3) ถึงแม้ว่าตัวควบคุมแบบพีไอ จะสามารถทำให้ผลตอบสนองของระบบมีค่าเท่ากับ Set Point ได้ แต่ในช่วงระยะเวลาหนึ่ง ผลตอบสนองมีค่ามากเกินกว่า Set Point ที่ได้ตั้งไว้ เนื่องจากทางทฤษฎีนั้นตัวควบคุมแบบพีไอจะเพิ่มพจน์จากสมการของตัวควบคุมแบบพีมา 1 พจน์ ซึ่งจะนำเอาค่าคงที่ของตัวควบคุมแบบพีไอ (Ki) คูณด้วยผลรวมของค่าความผิดพลาด (Error Sum) ทำให้ในช่วงแรกที่ Set Point และผลตอบสนอง มีค่าต่างกันมาก ทำให้พจน์ของไอที่เพิ่มเข้ามามีอิทธิพลมาก ส่งผลให้ตัวควบคุมส่งสัญญาณควบคุมออกไปเป็นจำนวนมาก ทำให้เอาต์พุตที่ได้พุ่งขึ้นอย่างรวดเร็วจนเกิน Set Point อีกทั้งเมื่อระบบเข้าสู่สภาวะสมดุล ค่าความผิดพลาดจะมีค่าเป็นศูนย์ ทำให้พจน์ของตัวควบคุมแบบพีไม่มีผลแต่พจน์ของตัวควบคุมแบบไอยังคงมีค่า เนื่องจากเป็นผลรวมของค่าความผิดพลาดจะเกิดปรากฏการณ์ ที่เรียกว่า Integral Windup ขึ้น ดังสมการด้านล่าง

$$u(t) = MV(t) = K_p * e(t) + K_i * (e(t) + e(t-1))$$

สมการที่ 3.3 สมการสัญญาณเอาต์พุตของตัวควบคุมแบบพีไอ

เมื่อ u(t) คือ เอาต์พุตของระบบในที่นี้คือสัญญาณควบคุมจากตัวควบคุมที่ส่งไปยังระบบ

K_p คือ ค่าคงที่ของตัวควบคุมแบบพี

K_i คือ ค่าคงที่ของตัวควบคุมแบบไอ

e(t) คือ ค่าความผิดพลาดของระบบ

e(t-1) คือ ค่าความผิดพลาดของระบบครั้งก่อนหน้า

จากสมการที่ 3.3 เมื่อระบบเข้าสู่สภาวะสมดุลทำให้พจน์ของตัวควบคุมแบบพีหายไป จนเกิดเป็นสมการใหม่ดังนี้

$$u(t) = MV(t) = K_p * e(t) + K_i * (e(t) + e(t-1))$$

$$u(t) = MV(t) = K_i * (e(t) + e(t-1))$$

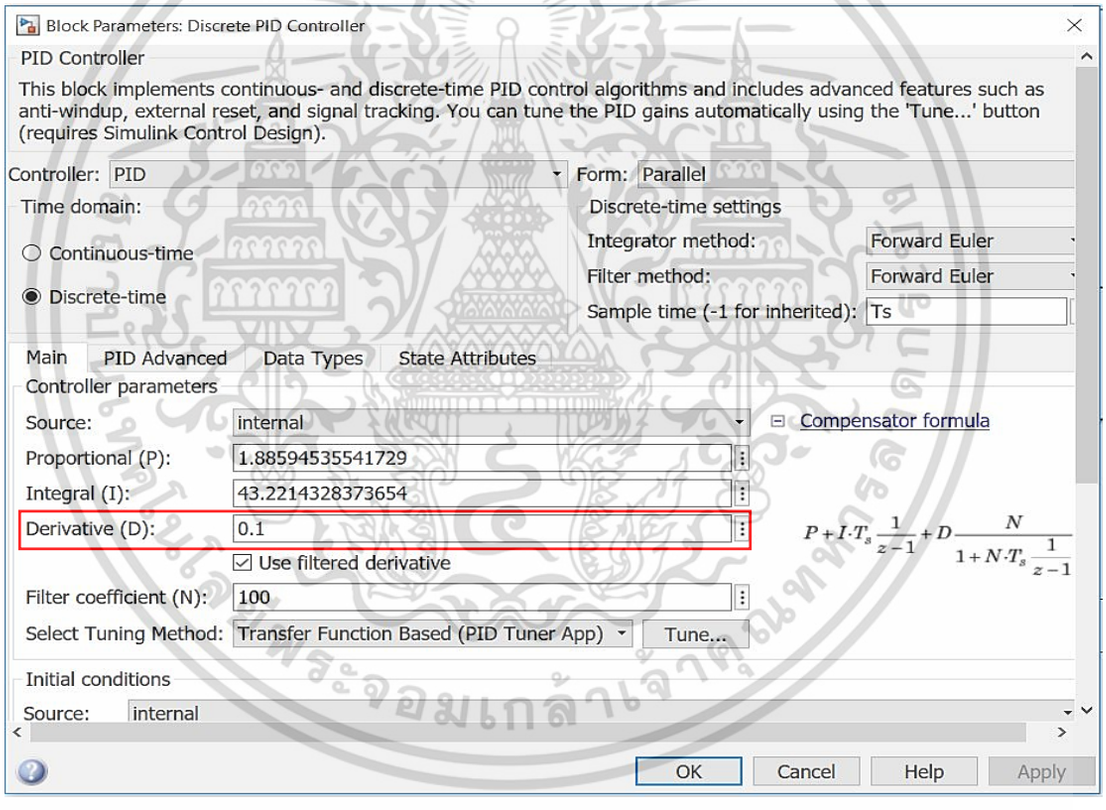
สมการที่ 3.4 สมการ Integral Windup

จากสมการที่ 3.4 จะเกิดค่าที่ยังคงอยู่ในระบบของพจน์ของตัวควบคุมแบบไอจะเกิดปรากฏการณ์ที่เรียกว่า Integral Windup ดังที่กล่าวไปข้างต้น

3.2.6 การจำลองตัวควบคุมแบบพี-ไอโดยใช้เทคนิค Anti-Windup ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

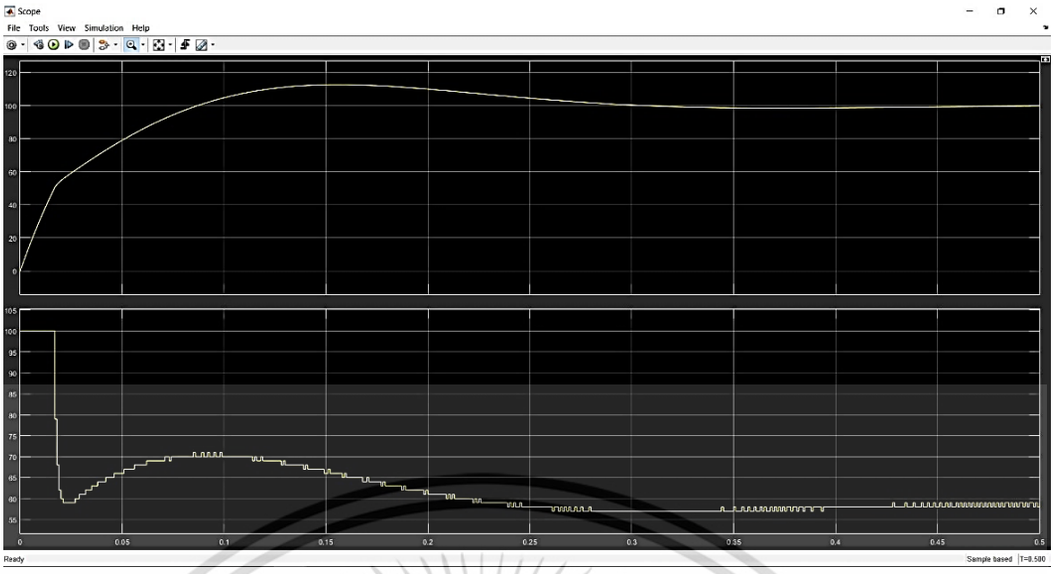
จากการทดลองที่ 3.2.5 ถึงแม้ว่าตัวควบคุมแบบพีไอ (PI Controller) จะถูกนำมาใช้ แต่ก็ไม่สามารถทำให้ระบบมีผลตอบสนองที่มีประสิทธิภาพได้ ผลตอบสนองยังคงมีค่าเกินกว่า Set Point ในช่วงแรกของการควบคุม เรียกปรากฏการณ์ที่สัญญาณควบคุมมีค่ามากเกินกว่าที่ระบบจะรับได้จนทำให้เกิดการพุ่งของผลตอบสนองว่า Overshoot

การกำจัด Overshoot นั้นมีหลากหลายวิธีด้วยกัน วิธีที่รู้กันทั่วไป คือ ใช้ตัวควบคุมแบบดี (D Controller) เข้ามาช่วยในการกำจัด overshoot เพราะในทางทฤษฎีตัวควบคุมแบบดี ทำงานตรงข้ามกับตัวควบคุมแบบไอ ทำให้เมื่อใช้ตัวควบคุมแบบพีไอดี (PID Controller) ร่วมกันแล้วจะทำให้ระบบมีเสถียรภาพไม่มี Overshoot ได้ แต่ตัวควบคุมแบบดี มีความไวที่สูงมากทำให้การปรับค่านั้นเป็นไปได้ยาก โครงงานนี้จึงได้มีการสร้างการทดลองที่ใช้ตัวควบคุมแบบดีเพิ่มเข้ามาพบว่าการปรับค่าใน Simulink Toolbox เป็นการปรับแบบลองผิดลองถูก (Trial and Error) ดังแสดงในรูปที่ 3.76



รูปที่ 3.76 แสดงการปรับจูนค่าพารามิเตอร์ของตัวควบคุมแบบดีใน Discrete PID Controller Block

จากรูปที่ 3.76 การปรับค่าพารามิเตอร์ของตัวควบคุมแบบดี (K_d) จะไม่เหมือน กับการปรับค่าของตัวควบคุมแบบพีและไอ เนื่องจากไม่ได้ใช้ PID Tuner Toolbox ในการปรับค่า ทำให้การใช้ตัวควบคุมแบบดีจึงเป็นเรื่องที่ต้องพิจารณาเป็นอย่างมาก



รูปที่ 3.77 แสดงผลตอบสนองของระบบที่มีค่า K_d = 0.1

จากรูปที่ 3.77 สังเกตได้ว่า ระบบขาดเสถียรภาพเป็นอย่างมาก และใช้ระยะ เวลาเข้าสู่สภาวะสมดุลที่นานพอสมควร ตัวควบคุมแบบดี จึงไม่เหมาะที่จะนำมาใช้งาน ในโครงงานนี้จึงเลือกใช้ตัวควรคุมแบบพีและพีไอเท่านั้น สำหรับวิธีที่จะกำจัด Overshoot อีกหนึ่งวิธี เรียกว่า Anti-Windup ในการทดลองนี้จะทดลองการใช้งาน Anti-Windup ใน Simulink Toolbox

3.2.6.1 จุดประสงค์การทดลองการจำลองตัวควบคุมแบบพี-ไอโดยใช้เทคนิค Anti-Windup ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

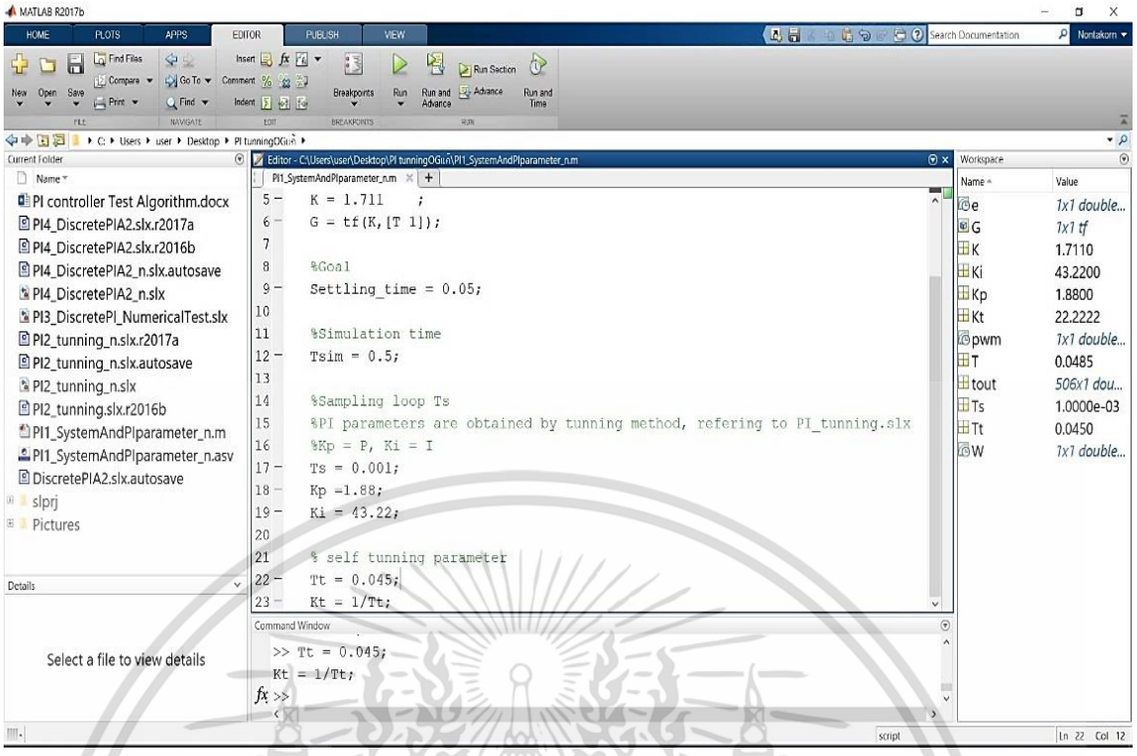
1) เพื่อศึกษาการทำงานของ Anti-Windup ในการลด Overshoot
2) เพื่อศึกษาการใช้งาน Function Block ใน Simulink Toolbox

3.2.6.2 อุปกรณ์ที่ใช้ในการทดลองการจำลองตัวควบคุมแบบพี-ไอโดยใช้เทคนิค Anti-Windup ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

1) คอมพิวเตอร์ที่ติดตั้งโปรแกรม MATLAB

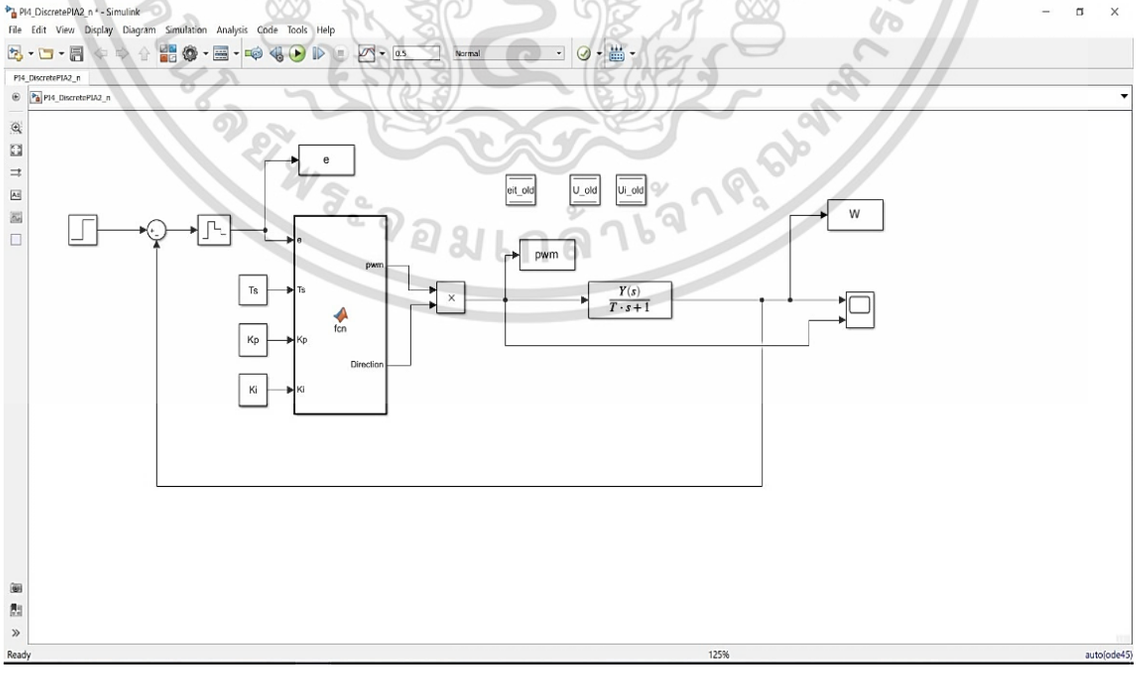
3.2.6.3 วิธีการทดลองการจำลองตัวควบคุมแบบพี-ไอโดยใช้เทคนิค Anti-Windup ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

1) ทำการสร้างไฟล์ .m เพื่อใช้เก็บและแสดงค่าพารามิเตอร์ต่างๆ ของระบบดังรูปที่ 3.88 โดยจะใช้ค่า K_p และ K_i จากการทดลองที่ 3.2.5 และเพิ่มตัวแปรของ Anti-Windup ขึ้นมา (T_t และ K_t) โดย K_t เท่ากับ 1 / T_t เมื่อ K_t คือพารามิเตอร์ของเทคนิค Anti-Windup และ T_t คือเวลาที่ต้องการกำจัด Overshoot



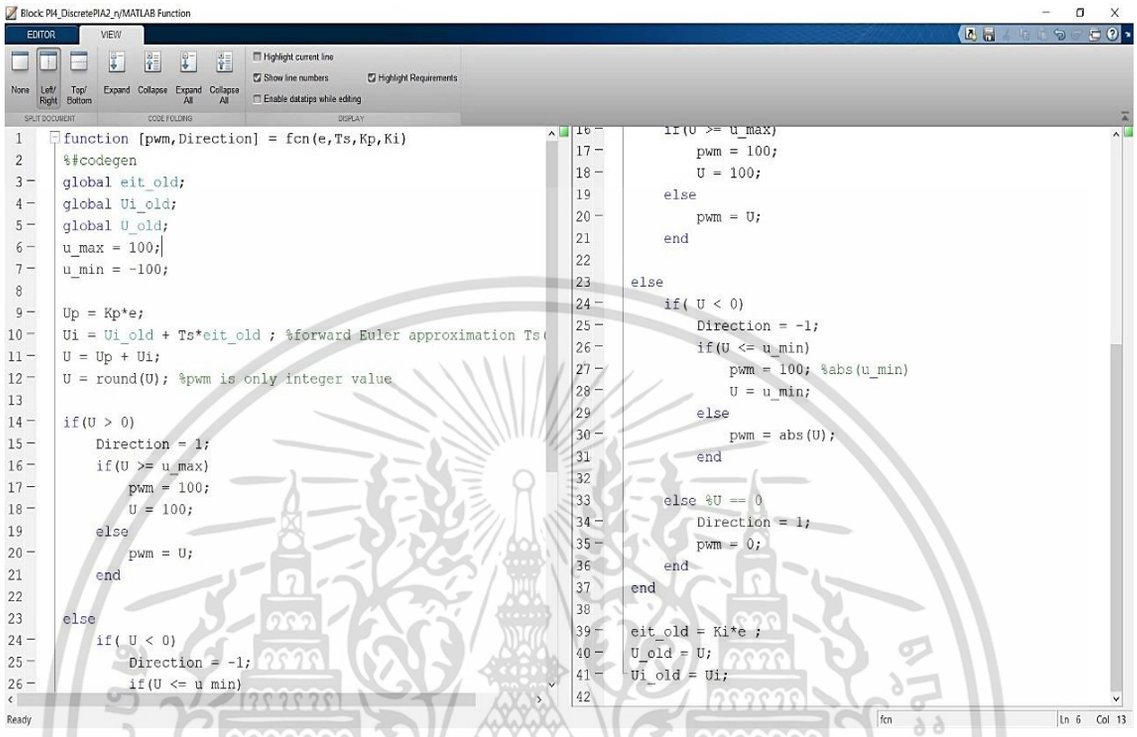
รูปที่ 3.78 แสดงไฟล์ที่เก็บค่าพารามิเตอร์ต่างๆ ของการจำลอง

2) ทำการสร้างระบบควบคุมแบบวงปิด ใน Simulink Toolbox เช่นเดียวกับขั้นตอนที่ 5 ในการทดลองที่ 3.2.3 เพียงแต่แทนที่ Discrete PID Controller Block ด้วย Function Block ดังรูปที่ 3.79



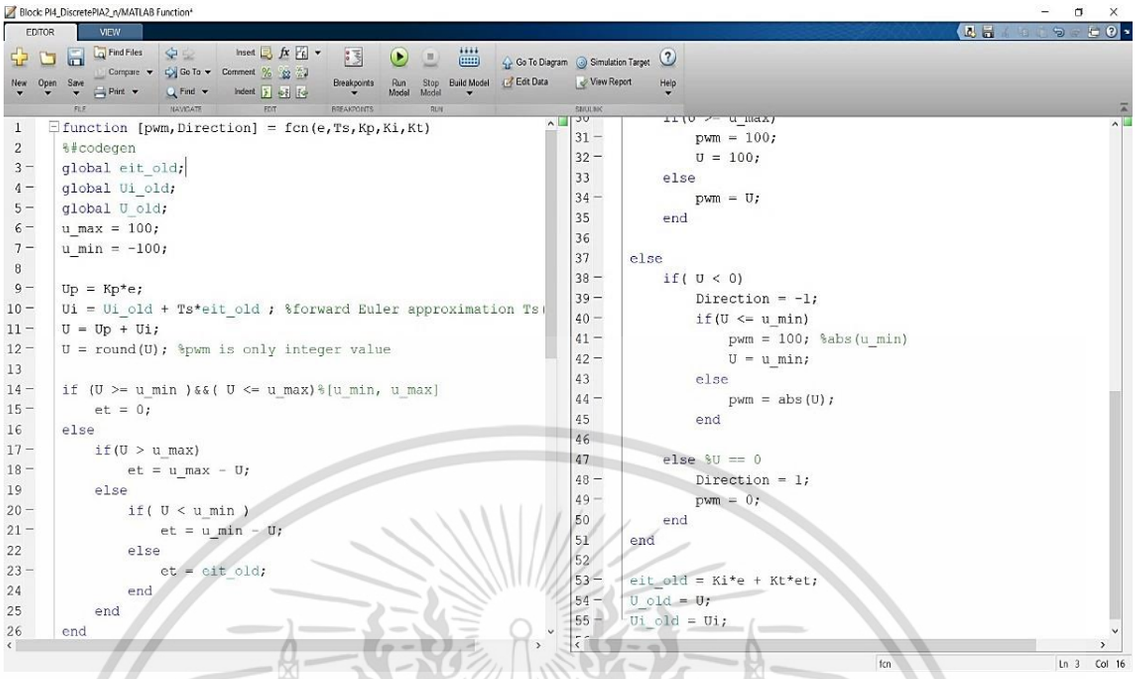
รูปที่ 3.79 แสดงการจำลองระบบควบคุมแบบวงปิดโดยใช้เทคนิค Anti-Windup ใน Simulink Toolbox

3) เมื่อทำการดับเบิลคลิกใน Function Block จะปรากฏหน้าต่าง ดังรูปที่ 3.80 เพื่อใช้ในการสร้างโปรแกรมจำลองเนื่องจากใน Discrete PID Toolbox ไม่มีฟังก์ชันของ Anti-Windup มาให้



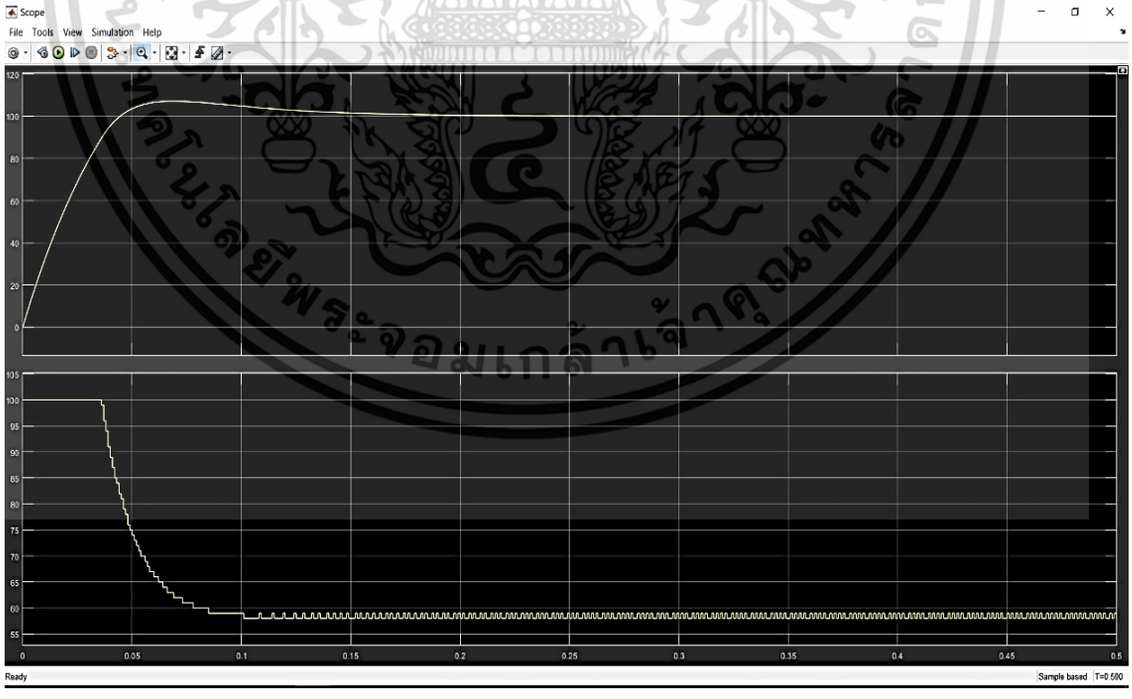
รูปที่ 3.80 แสดงโปรแกรมใน Function Block สำหรับตัวควบคุมแบบพีไอ

4) ทำการ Run โปรแกรมแล้วสังเกตผลลัพธ์ เพื่อทดสอบว่าได้ผลตอบสนองใกล้เคียงกับผลตอบสนองที่ใช้ Discrete PID Controller Block หรือไม่ก่อนเพิ่มเทคนิค Anti-Windup เข้าไป
5) ทำการสร้างโปรแกรมใน Function Block โดยเพิ่มเทคนิค Anti-Windup เข้าไป ดังรูปที่ 3.81 โดยกำหนดค่า T_t เท่ากับ 0.045
6) ทำการรันโปรแกรมเพื่อดูผลลัพธ์

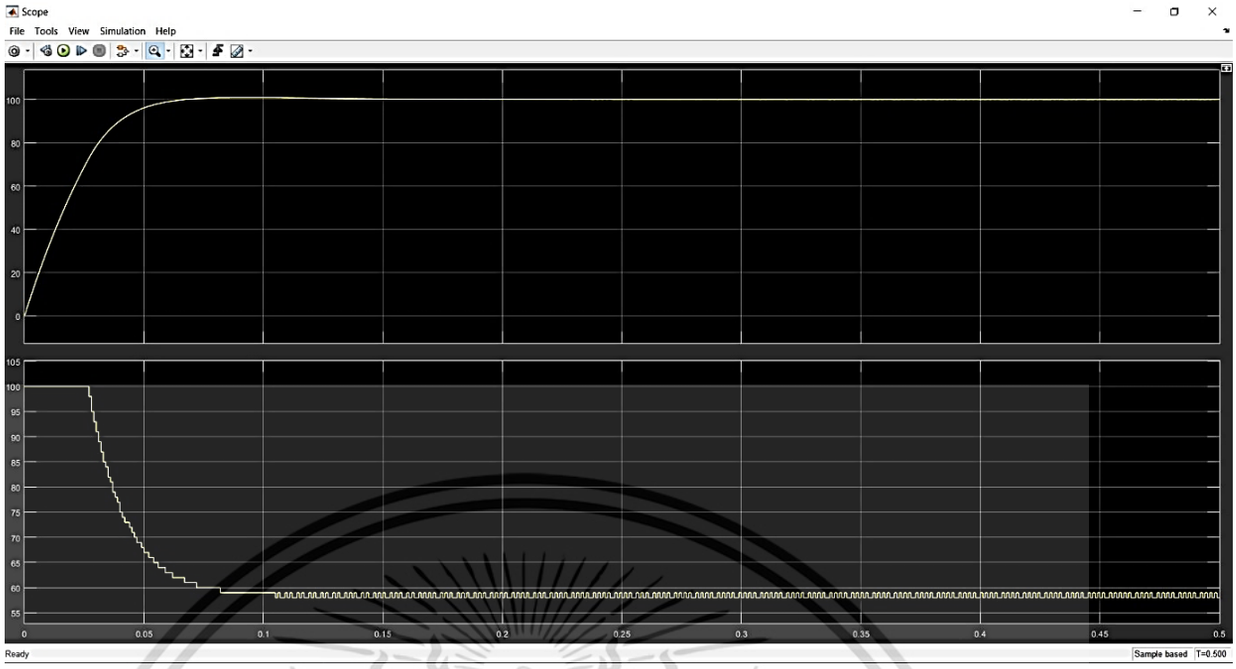


รูปที่ 3.81 แสดงโปรแกรมใน Function Block สำหรับตัวควบคุมแบบพีไอด้วยเทคนิค Anti-Windup

3.2.6.4 ผลการทดลองการจำลองตัวควบคุมแบบพี-ไอโดยใช้เทคนิค Anti-Windup ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB



รูปที่ 3.82 แสดงผลตอบสนองและสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ โดยใช้ Function Block

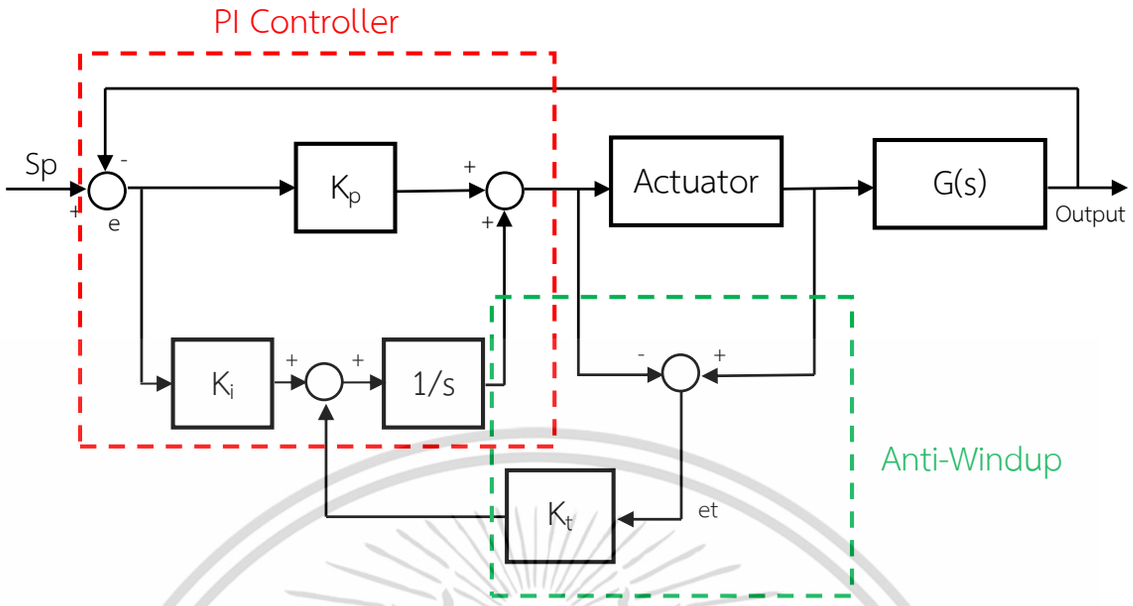


รูปที่ 3.83 แสดงผลตอบสนองและสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอโดยใช้เทคนิค Anti-Windup (T_t เท่ากับ 0.045)

จากรูปที่ 3.82 ผลการตอบสนองของตัวควบคุมแบบพีไอที่ใช้ Function Block แทนที่ Discrete PID Controller Block จะมีความใกล้เคียงกัน และจากรูปที่ 3.83 เมื่อใช้เทคนิค Anti-Windup ระบบมีผลตอบสนองที่ดีขึ้น Overshoot ลดลง และเข้าสู่สภาวะสมดุลที่เวลาใกล้เคียงกับ 0.05 วินาทีจามที่ได้กำหนดไว้

3.2.6.5 สรุปผลการทดลองการจำลองตัวควบคุมแบบพี-ไอโดยใช้เทคนิค Anti-Windup ในการควบคุมความเร็วของมอเตอร์ในโปรแกรม MATLAB

1) การลดผลของ Overshoot โดยใช้ตัวควบคุมแบบดี (D Controller) สามารถทำได้ยาก เนื่องจากตัวควบคุมแบบดีมีความไวต่อการเปลี่ยนแปลงเป็นอย่างมาก อีกทั้งภายใน Discrete PID Controller Block นั้น การปรับจูนค่า K_d เป็นการปรับแบบลองผิดลองถูก ซึ่งมีความเสี่ยงที่จะทำให้ผลตอบสนองของระบบขาดเสถียรภาพ ดังนั้นการใช้เทคนิค Anti-Windup จึงเป็นวิธีที่ดีกว่า
2) การใช้ตัวควบคุมแบบพีไอที่ใช้เทคนิค Anti-Windup สามารถลดผลของ Overshoot ได้ อีกทั้งยังไม่ทำให้ระบบขาดเสถียรภาพและเข้าสู่สภาวะสมดุลได้ตามระยะเวลาที่ต้องการ โดยหลักการทำงานของ Anti-Windup นั้นจะเป็นการนำค่าของสัญญาณควบคุมที่ออกจาก Actuator หรือ Saturation Block มาเปรียบเทียบด้วยสัญญาณที่เข้า Saturation Block เพื่อนำค่าไปคำนวณหาความแตกต่างแล้วนำค่าที่ได้ไปลดผลของ Overshoot วิธีนี้จะไม่นำผลของค่าความผิดพลาด (Error) มาคำนวณทำให้ระบบยังคงมีเสถียรภาพอยู่



รูปที่ 3.84 แสดง Block Diagram ของระบบ ที่ใช้ตัวควบคุมแบบพีไอและเทคนิค Anti-Windup

จากรูปที่ 3.84 การทำงานของ Anti-Windup จะใช้สัญญาณที่ออกจาก Actuator หรือบอร์ดขับมอเตอร์ หักล้างกับสัญญาณที่เข้ามา เพื่อคำนวณความแตกต่าง แล้วนำค่าที่ได้ไปลดผลของ Overshoot เทคนิคนี้จึงเป็นวิธีการที่กำจัดผลของ Overshoot ได้อย่างมีประสิทธิภาพ

3.2.7 การจำลองตัวควบคุมแบบพี/พีไอ (P/PI controller) ในการควบคุมตำแหน่งเชิงมุมของมอเตอร์ในโปรแกรม MATLAB

เนื่องจากการควบคุมมอเตอร์ จะมุ่งเน้นไปที่ตัวแปรหลักๆ 2 ตัวคือ ความเร็ว (Speed) และตำแหน่งเชิงมุม เนื่องจากความเร็วเป็นตัวบ่งบอกประสิทธิภาพของมอเตอร์ และตำแหน่งเชิงมุมเป็นความสามารถในการหมุนไปยังมุมที่ต้องการ สามารถนำไปใช้บังคับแขนกลต่างๆ ในการทดลองนี้ได้มีการจำลองตำแหน่งของมอเตอร์ไฟฟ้ากระแสตรง เพื่อเรียนรู้การจำลองตำแหน่งเชิงมุม แต่ไม่ได้มีการนำไปใช้ทดลองจริงบนมอเตอร์

สำหรับการจำลองตำแหน่งเชิงมุมนั้นขั้นตอนและวิธีการต่างๆ มีความคล้ายคลึงกับการจำลองความเร็วเพียงแต่เพิ่ม Integrator Block เข้ามาก่อนที่สัญญาณควบคุม จากตัวควบคุมจะเข้าสู่ระบบที่แทนด้วยฟังก์ชันถ่ายโอน

3.2.7.1 จุดประสงค์การทดลองการจำลองตัวควบคุมแบบพี/พีไอ (P/PI controller) ในการควบคุมตำแหน่งเชิงมุมของมอเตอร์ในโปรแกรม MATLAB

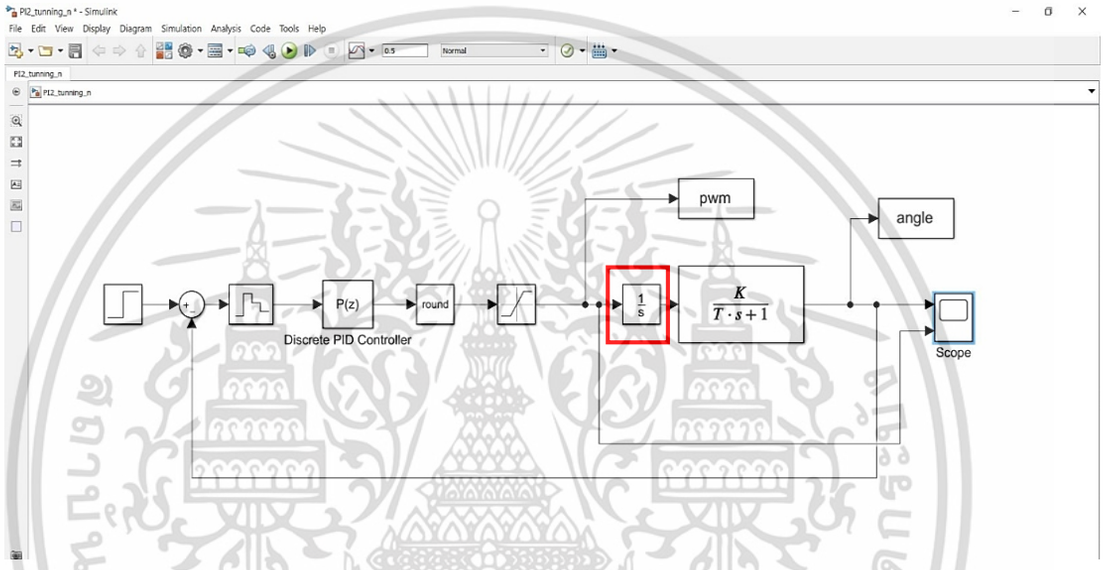
1) เพื่อศึกษาพฤติกรรมของตัวควบคุมแบบพีและพีไอที่มีผลต่อการควบคุมตำแหน่งเชิงมุม
2) เพื่อเรียนรู้การจำลองการควบคุมตำแหน่งเชิงมุมใน Simulink Toolbox ในโปรแกรม MATLAB

3.2.7.2 อุปกรณ์ที่ใช้ในการทดลองการจำลองตัวควบคุมแบบพี/พีไอ (P/PI controller) ในการควบคุมตำแหน่งเชิงมุมของมอเตอร์ในโปรแกรม MATLAB

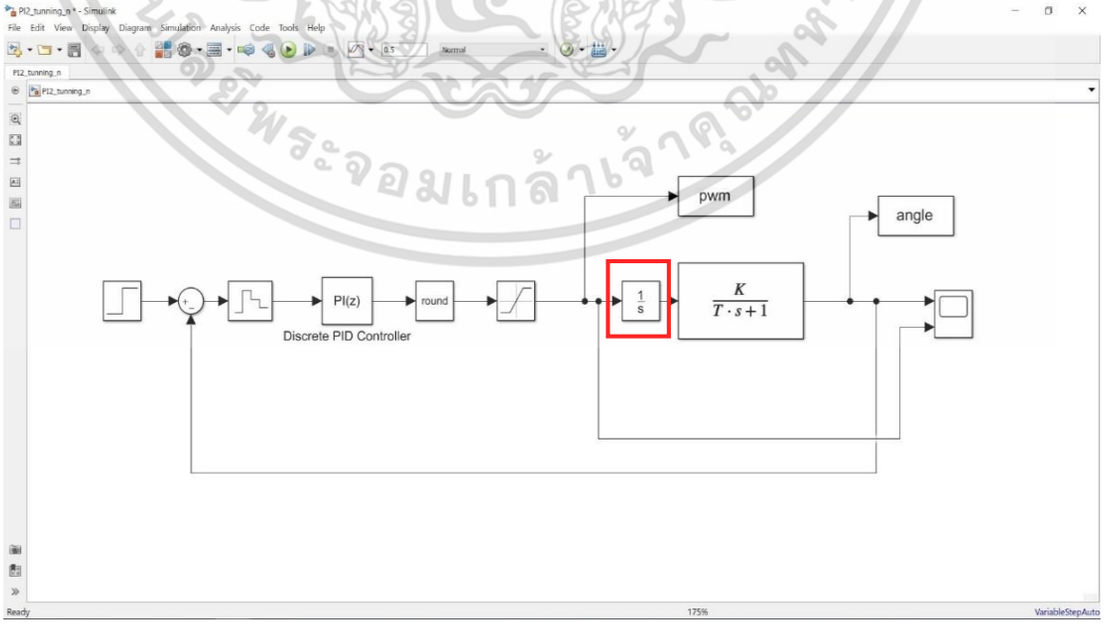
1) คอมพิวเตอร์ที่ติดตั้งโปรแกรม MATLAB

3.2.7.3 วิธีการทดลองการจำลองตัวควบคุมแบบพี/พีไอ (P/PI controller) ในการควบคุมตำแหน่งเชิงมุมของมอเตอร์ในโปรแกรม MATLAB

1) ทำการสร้างการควบคุมแบบวงปิดแบบเดียวกันกับการทดลองที่ 3.2.4 และ 3.2.5 เพียงแต่เพิ่ม Integrator Block เข้าไปดังรูปที่ 3.85 สำหรับการควบคุมแบบพีและดังรูปที่ 3.86 สำหรับการควบคุมแบบพีไอ



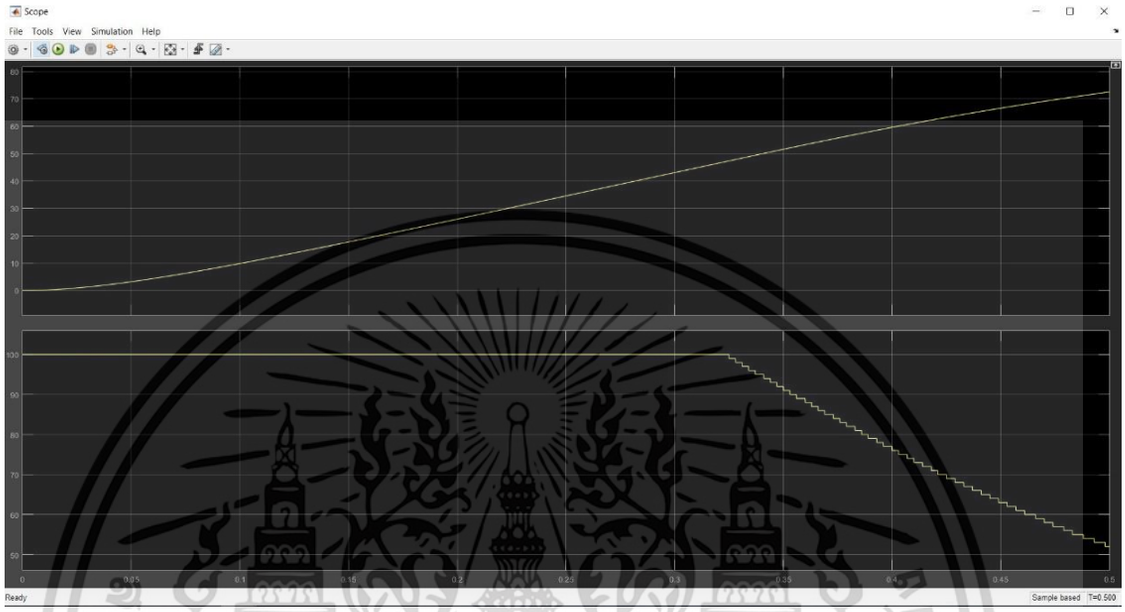
รูปที่ 3.85 แสดงการจำลองการควบคุมตำแหน่งเชิงมุมของมอเตอร์แบบวงปิดของตัวควบคุมแบบพี



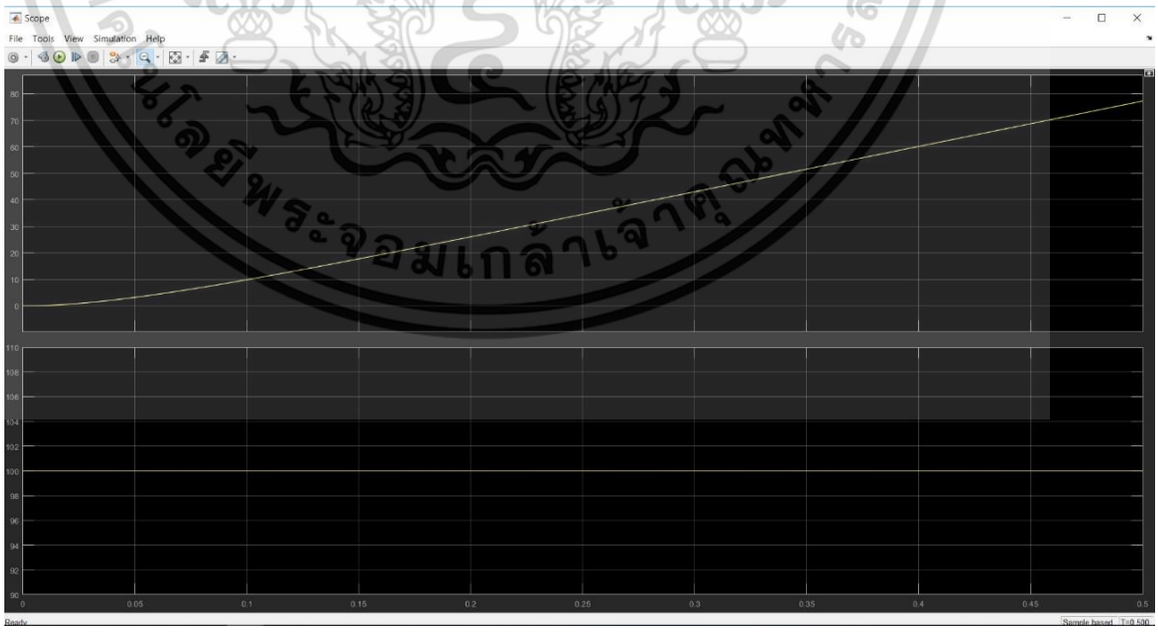
รูปที่ 3.86 แสดงการจำลองการควบคุมตำแหน่งเชิงมุมของมอเตอร์แบบวงปิดของตัวควบคุมแบบพีไอ

2) ทำการปรับจูนค่าพารามิเตอร์ของระบบตามที่ต้องการหลังจากนั้นทำการ Run โปรแกรมเพื่อดูผลลัพธ์

3.2.7.4 ผลการทดลองการจำลองตัวควบคุมแบบพี/พีไอ (P/PI controller) ในการควบคุมตำแหน่งเชิงมุมของมอเตอร์ในโปรแกรม MATLAB



รูปที่ 3.87 แสดงผลตอบสนองของระบบ สำหรับการควบคุมตำแหน่งเชิงมุมของมอเตอร์ เมื่อใช้ตัวควบคุมแบบพี



รูปที่ 3.88 แสดงผลตอบสนองของระบบ สำหรับการควบคุมตำแหน่งเชิงมุมของมอเตอร์ เมื่อใช้ตัวควบคุมแบบพีไอ

3.2.7.5 สรุปผลการทดลองการจำลองตัวควบคุมแบบพี/พีไอ (P/PI controller) ในการควบคุมตำแหน่งเชิงมุมของมอเตอร์ในโปรแกรม MATLAB

1) จากรูปที่ 3.87 เมื่อมีการใช้ตัวควบคุมแบบพี ควบคุมตำแหน่งเชิงมุมของมอเตอร์ จะทำให้มอเตอร์หมุนไปยังองศาที่ต้องการ สังเกตจากผลตอบสนองที่มีแนวโน้มขึ้นแล้วโค้งลงมา และสัญญาณควบคุมจะเพิ่มขึ้นอย่างช้าๆ เมื่อถึงจุดจุดหนึ่งจะลดลงอย่างช้าๆ เนื่องจากถึงตำแหน่งที่ต้องการแล้ว และจากรูปที่ 3.88 เมื่อมีการใช้ตัวควบคุมแบบพีไอควบคุมตำแหน่งเชิงมุมของมอเตอร์ สังเกตจากผลตอบสนองที่มีการพุ่งขึ้น ไม่มีแนวโน้มที่จะโค้งลงมา แสดงว่าอาจจะมีการสะบัดของแกนมอเตอร์ทำให้ต้องใช้ระยะเวลาหนึ่งถึงจะคงที่ และสัญญาณควบคุมอาจจะต้องใช้ขนาดที่เท่าเดิม เพิ่มประคองให้แกนของมอเตอร์คงที่ตลอดเวลา
2) การที่ใช้ Integrator Block เนื่องจากความเร็วเป็นอนุพันธ์ของระยะทาง กล่าวคือความเร็วเชิงมุมเป็นอนุพันธ์ของตำแหน่งเชิงมุม หากมีความเร็วแล้วต้องการตำแหน่ง ต้องทำการปริพันธ์หรือการ Integrate จึงต้องใส่ Integrator Block เข้าไปด้านหน้าของฟังก์ชันถ่ายโอนของระบบ
3) ฟังก์ชันถ่ายโอนของมอเตอร์ เมื่อต้องการควบคุมความเร็ว จะไม่เท่ากับฟังก์ชันถ่ายโอนของระบบที่ควบคุมตำแหน่งเชิงมุม หากจะนำมาจำลองต้องทำการ Open Loop Test เพื่อหาฟังก์ชันถ่ายโอนของระบบมาก่อน จึงจะสามารถจำลองได้อย่างไม่มีค่าผิดพลาด

บทที่ 4
ผลการทดลอง

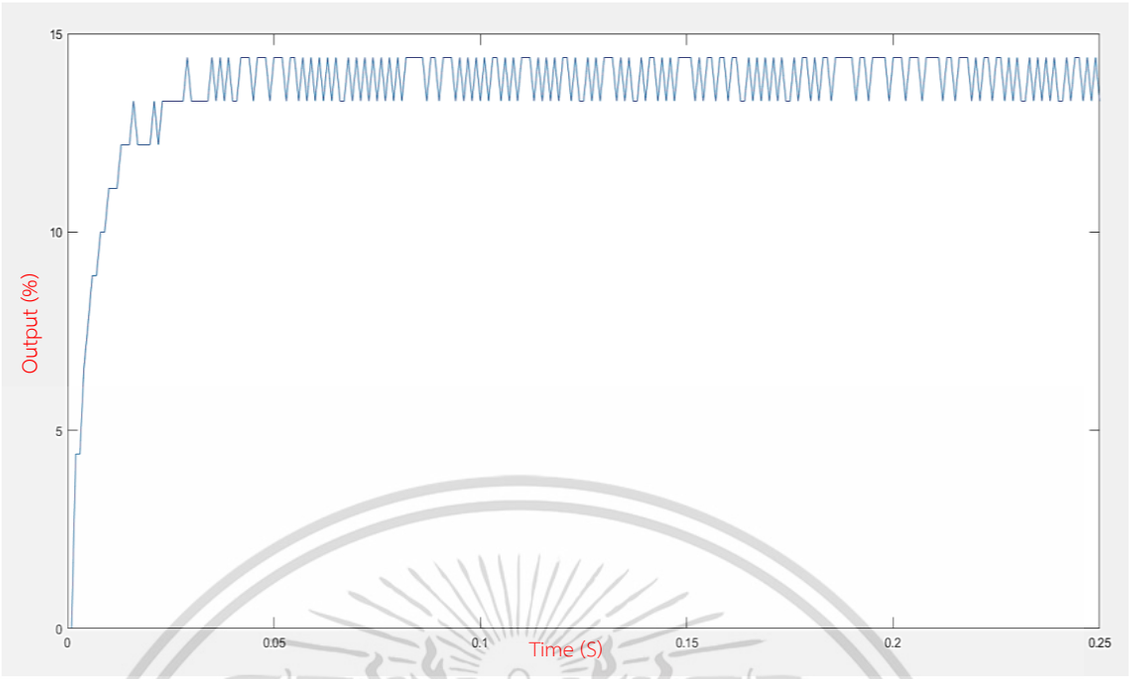
จากวิธีทำการจำลองและทดลองลงบนมอเตอร์ที่กล่าวไว้ในบทที่ 3 นั้น ทำการเก็บค่าของความเร็วของมอเตอร์ และสัญญาณควบคุมที่ Set Point ตั้งแต่ 0 ถึง 100% โดยเพิ่มที่ละ 10% โดยแยกเป็น 2 ส่วนคือส่วนของการตัวควบคุมแบบพี และส่วนของการใช้ตัวควบคุมแบบพีไอ

4.1 ผลการทดลองของการควบคุมมอเตอร์โดยใช้ตัวควบคุมแบบพี (P Controller)

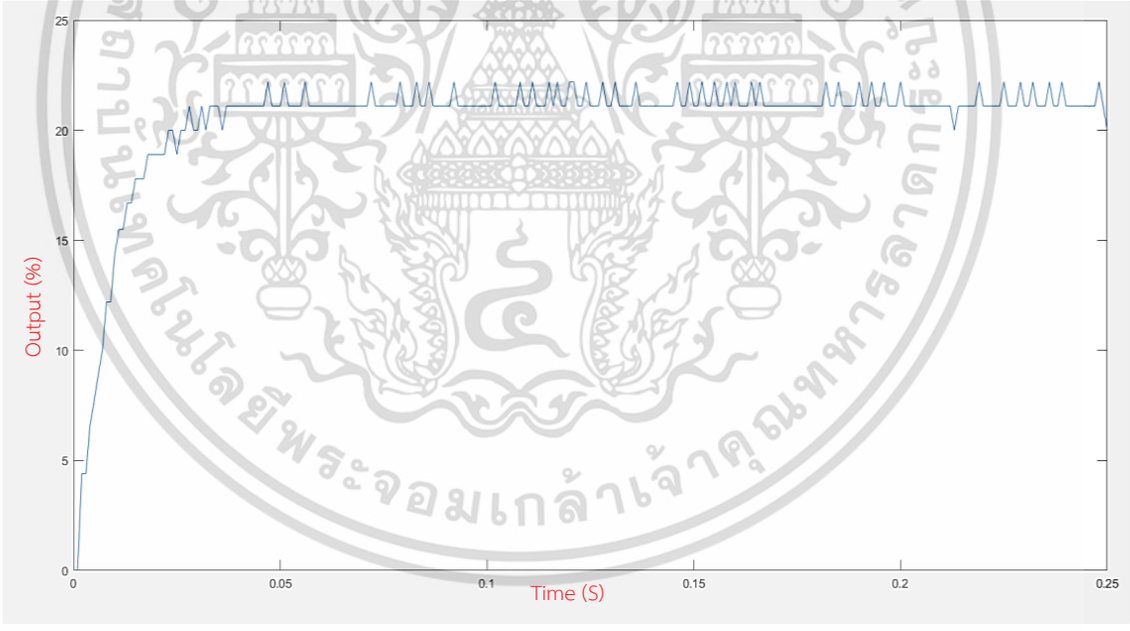
จากการทดลองที่ 3.2.4 ต้องการให้ระบบเข้าสู่สภาวะสมดุลที่เวลา 0.05 วินาที จะได้ค่าคงที่ของตัวควบคุมแบบพีเท่ากับ 1.67 (Kp เท่ากับ 1.67) จากนั้นนำไปทดลองจริงบนมอเตอร์

4.1.1 ความเร็วของมอเตอร์ของการใช้ตัวควบคุมแบบพี ในแต่ละค่าของ Set Point

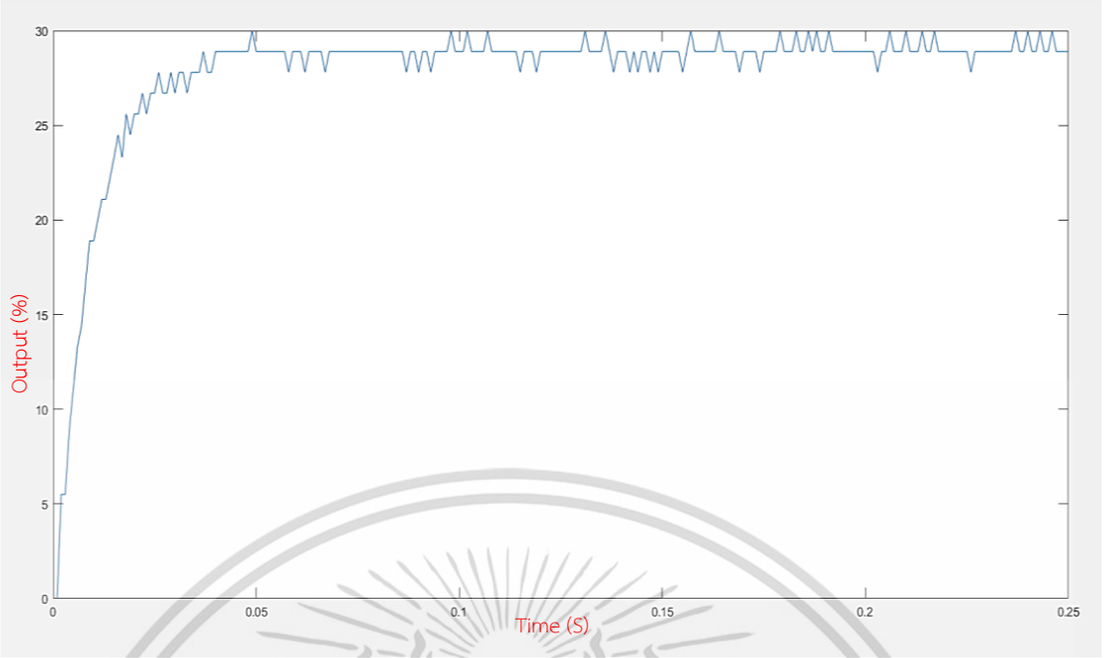
รูปที่ 4.1 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 10%



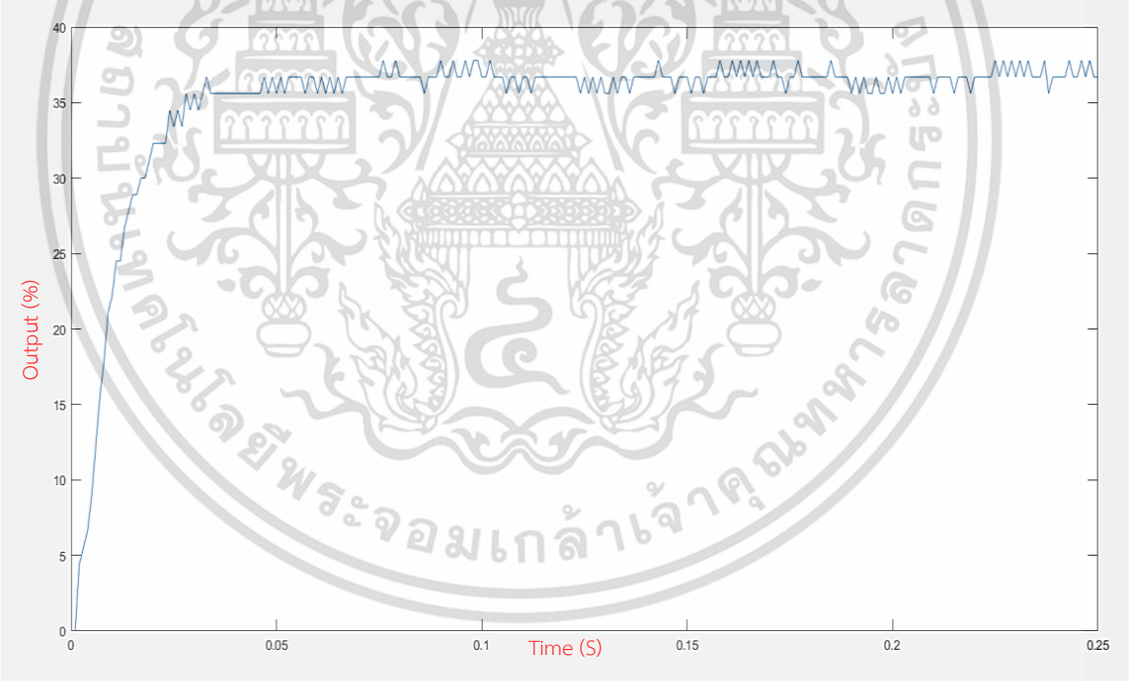
รูปที่ 4.2 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 20%



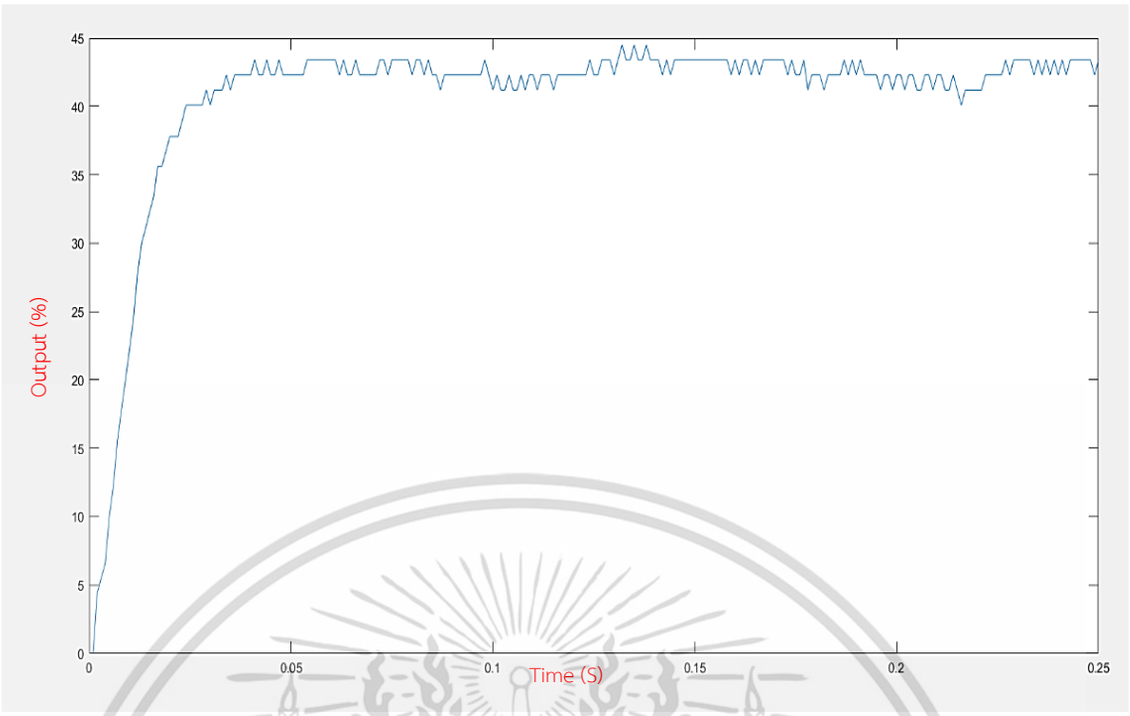
รูปที่ 4.3 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 30%



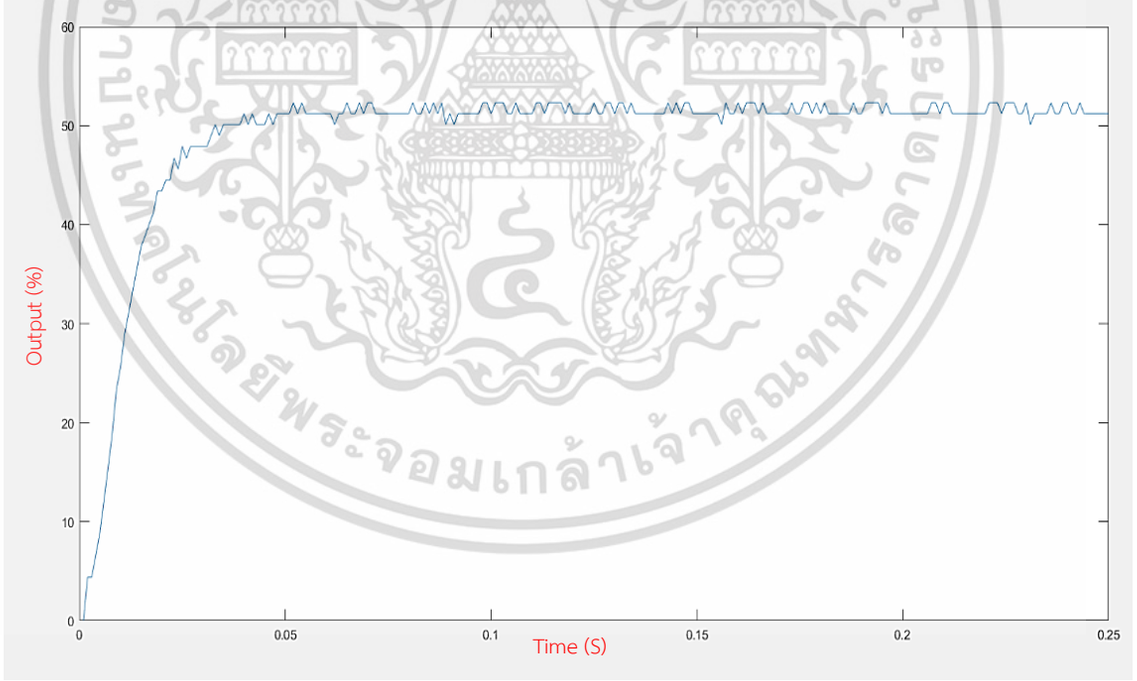
รูปที่ 4.4 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 40%



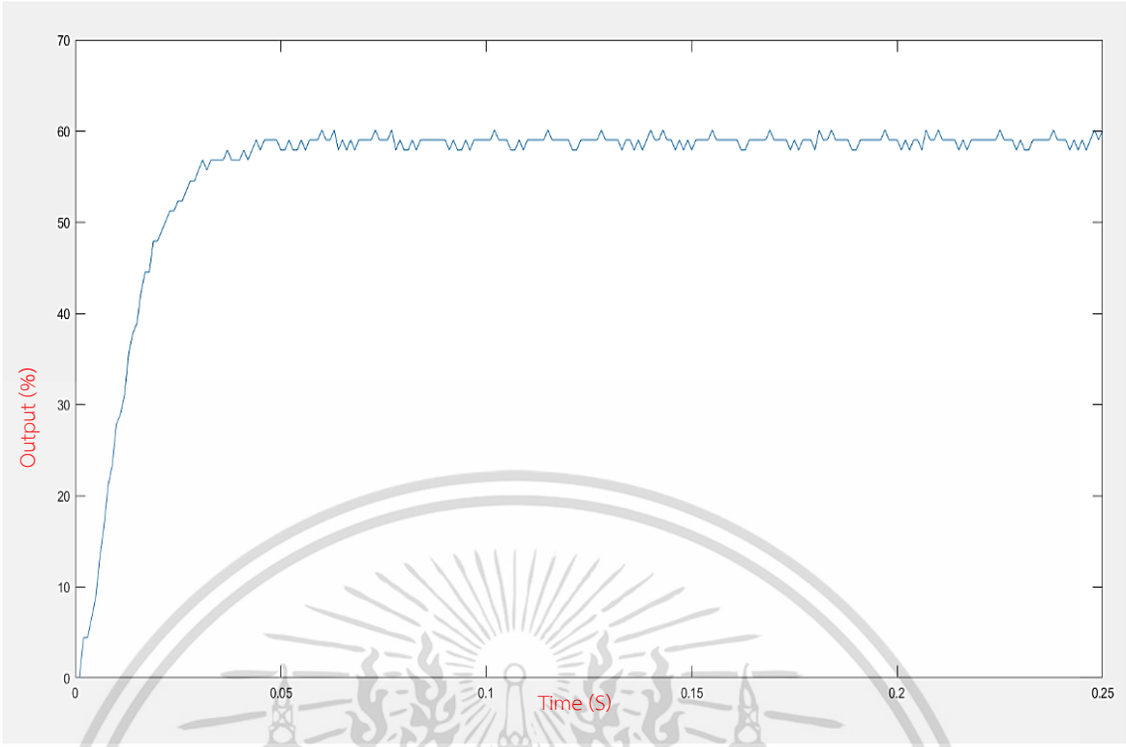
รูปที่ 4.5 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 50%



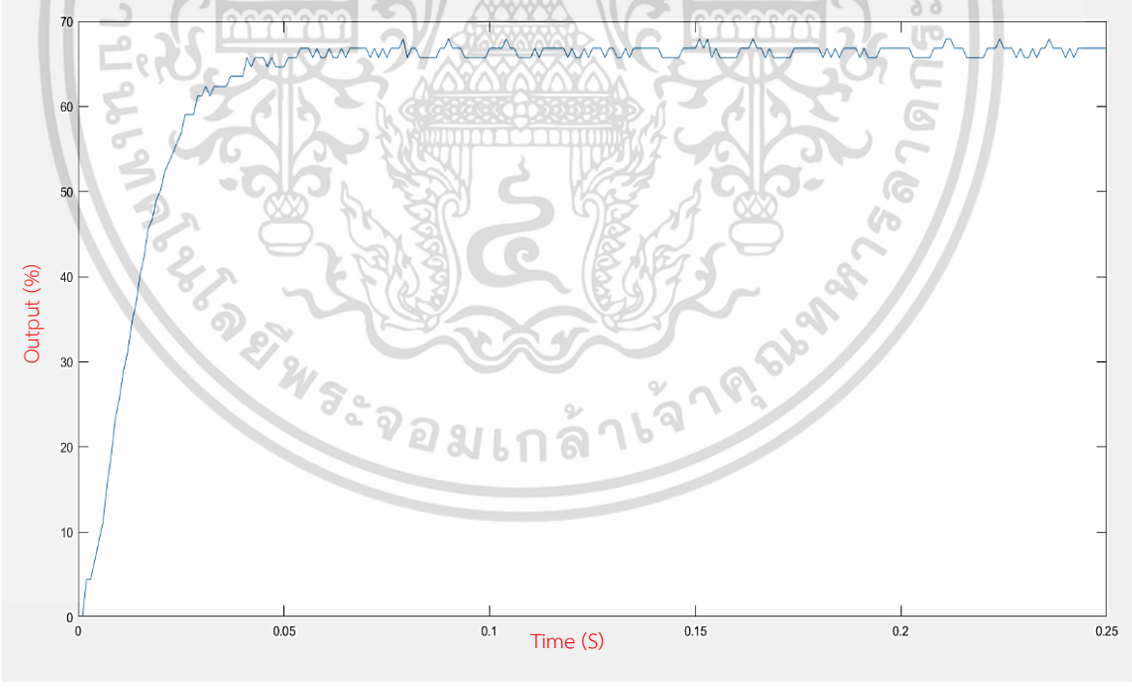
รูปที่ 4.6 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 60%



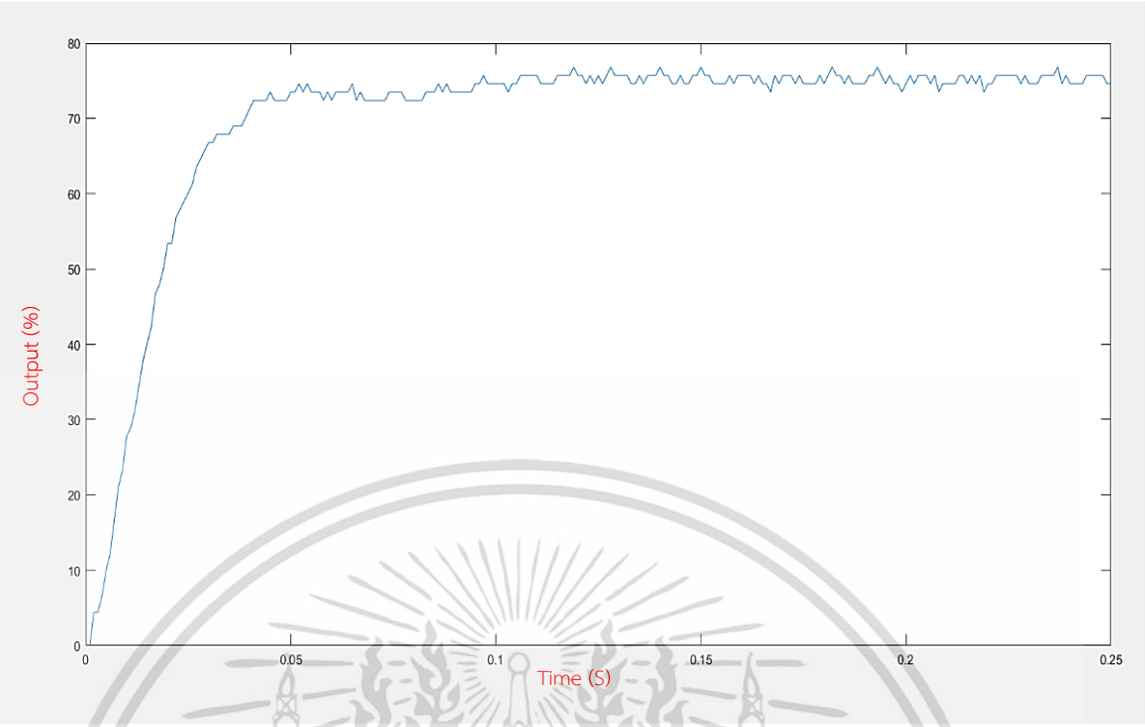
รูปที่ 4.7 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 70%



รูปที่ 4.8 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 80%



รูปที่ 4.9 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 90%



รูปที่ 4.10 แสดงผลตอบสนองของระบบที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 100%

การบอกค่าความผิดพลาดของผลตอบสนอง จะใช้สิ่งที่เรียกว่า Steady State Error หรือ Offset Error มีหน่วยเป็น % หากมีค่ามาก แสดงว่าตัวควบคุมไม่มีประสิทธิภาพเพียงพอ สมการการคำนวณ แสดงดังสมการที่ 4.1 และค่า Steady State Error ของแต่ละ Set Point แสดงดังตารางที่ 4.1

$$SSE = \frac{SP - Output}{SP}$$

สมการที่ 4.1 สมการค่าความผิดพลาดที่สภาวะสมดุล

เมื่อ	SSE	คือ	ค่าความผิดพลาดที่สภาวะสมดุล มีหน่วยเป็น %
	SP	คือ	ค่าของ Set Point ที่กำหนด
	Output	คือ	ค่าของเอาต์พุตที่สภาวะสมดุล

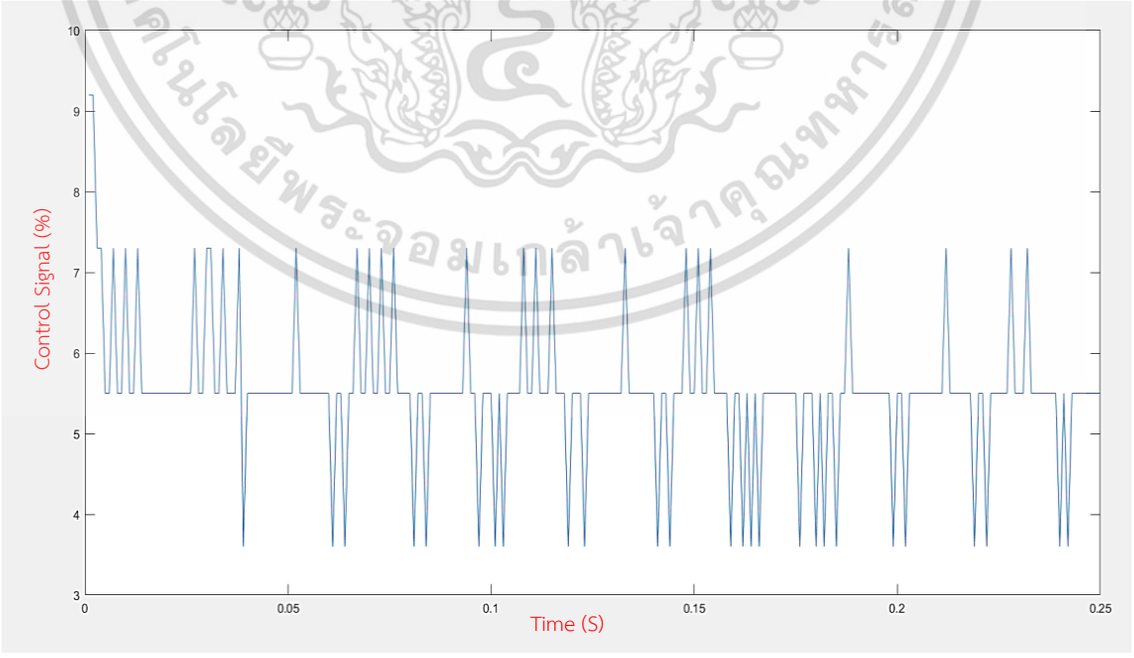
ตารางที่ 4.1 ค่าของ Steady State Error ของแต่ละ Set Point

Set Point (%)	Steady State Error (%)
10	34.0
20	28.0
30	29.7
40	27.8
50	24.4
60	27.7
70	25.3
80	26.2
90	27.0
100	25.4

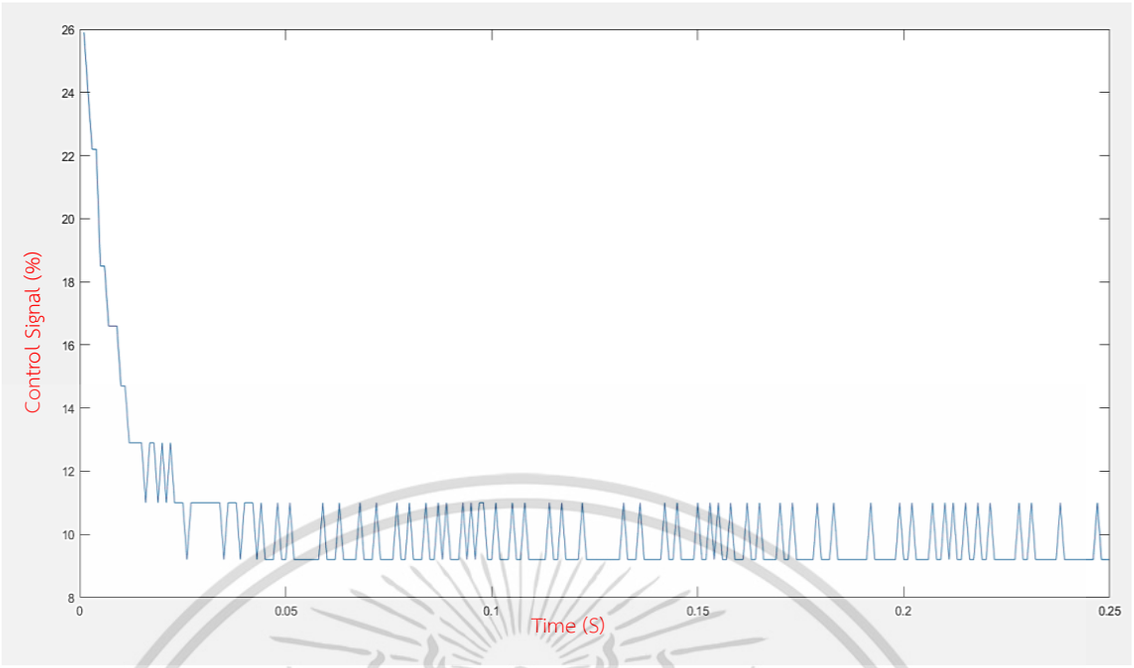
จากตารางที่ 4.1 พบว่า Steady State Error ของผลตอบสนอง ทุกค่าของ Set Point มีความใกล้เคียงกันที่ประมาณ 27.5%

4.1.2 สัญญาณควบคุมของการใช้ตัวควบคุมแบบพี ในแต่ละค่าของ Set Point

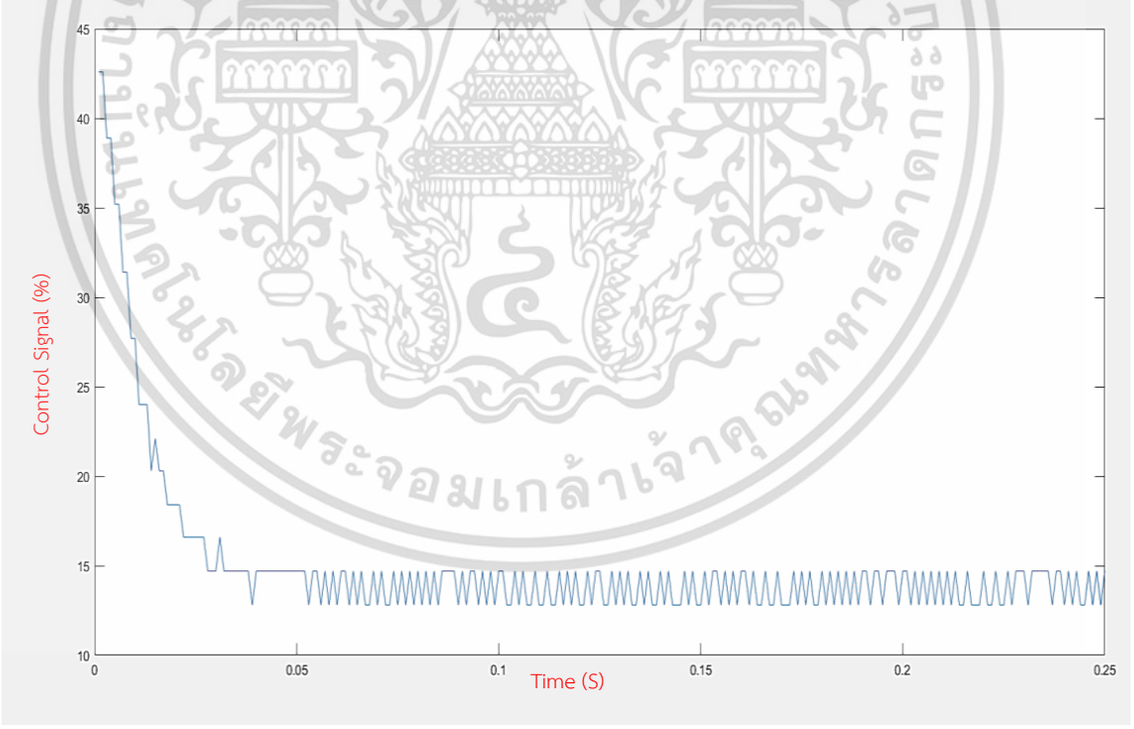
นอกจากการสังเกตผลตอบสนองของระบบแล้ว ในการจำลองการควบคุมนั้นจะดูแนวโน้มของสัญญาณควบคุมด้วยเนื่องจากเป็นตัวบ่งบอกว่าส่งสัญญาณไปให้ระบบถูกต้องตอบผลตอบสนองที่ออกมาหรือไม่มีผลต่างๆ ดังนี้



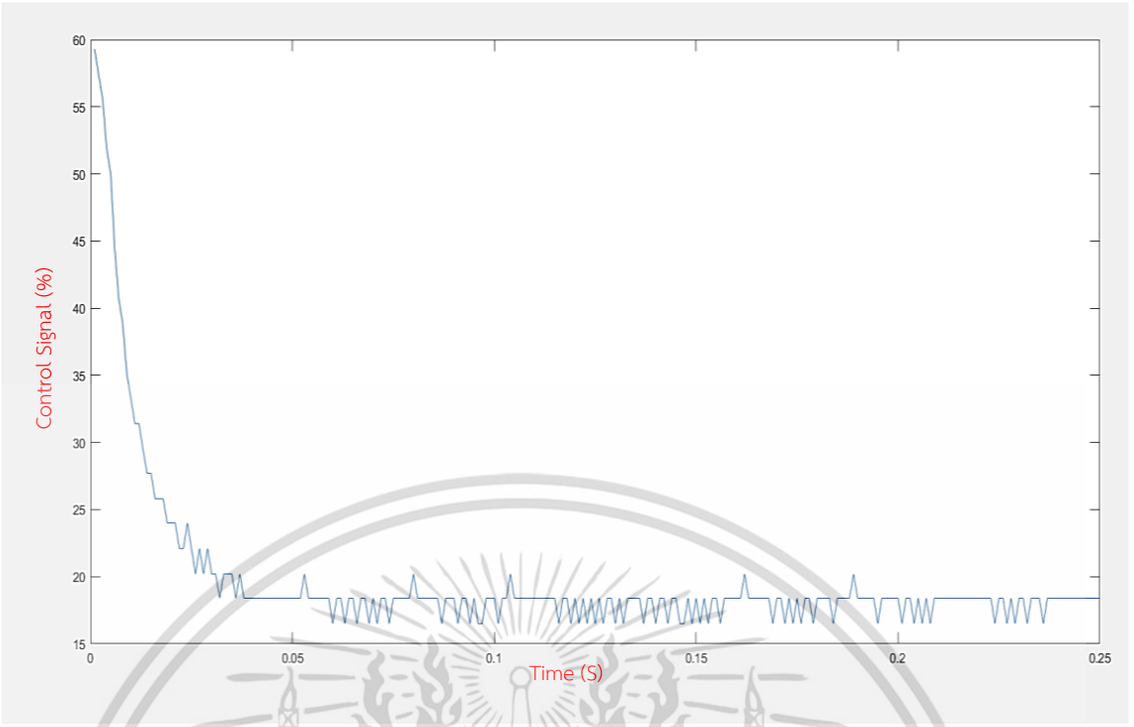
รูปที่ 4.11 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 10%



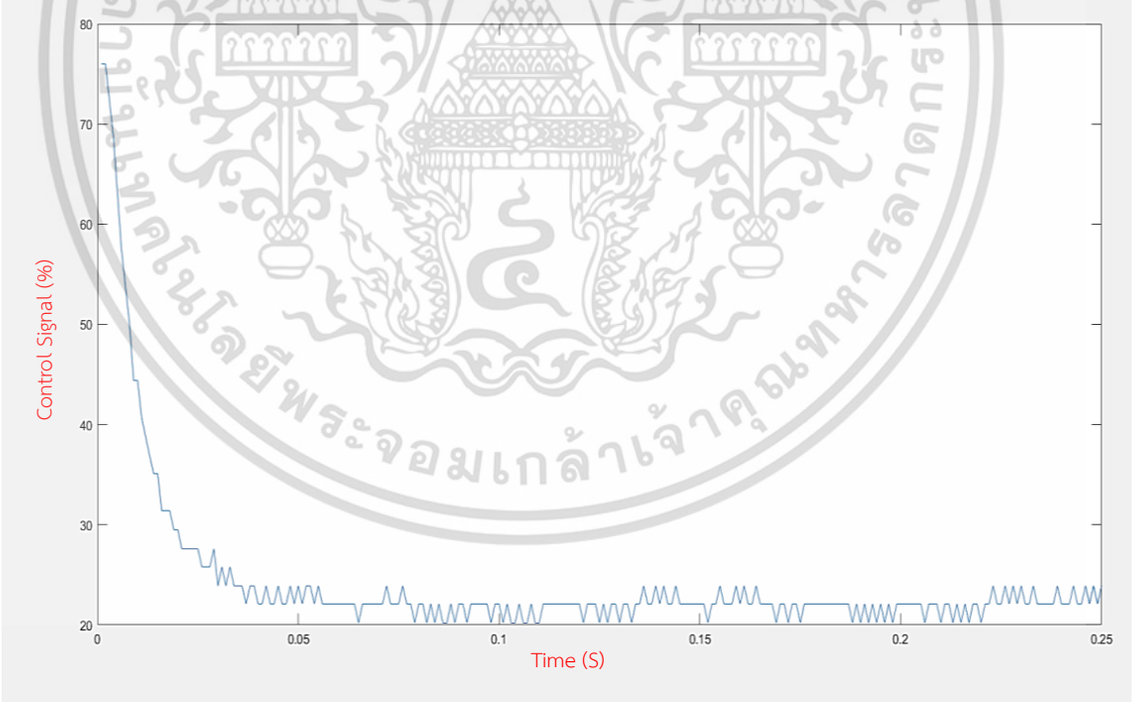
รูปที่ 4.12 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 20%



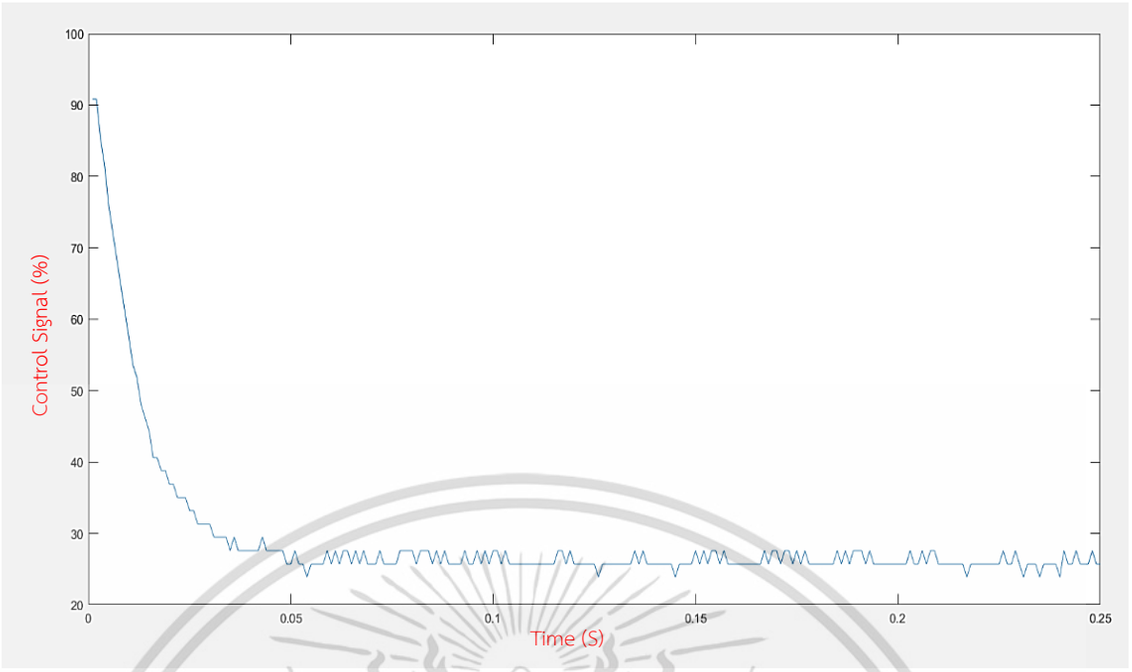
รูปที่ 4.13 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 30%



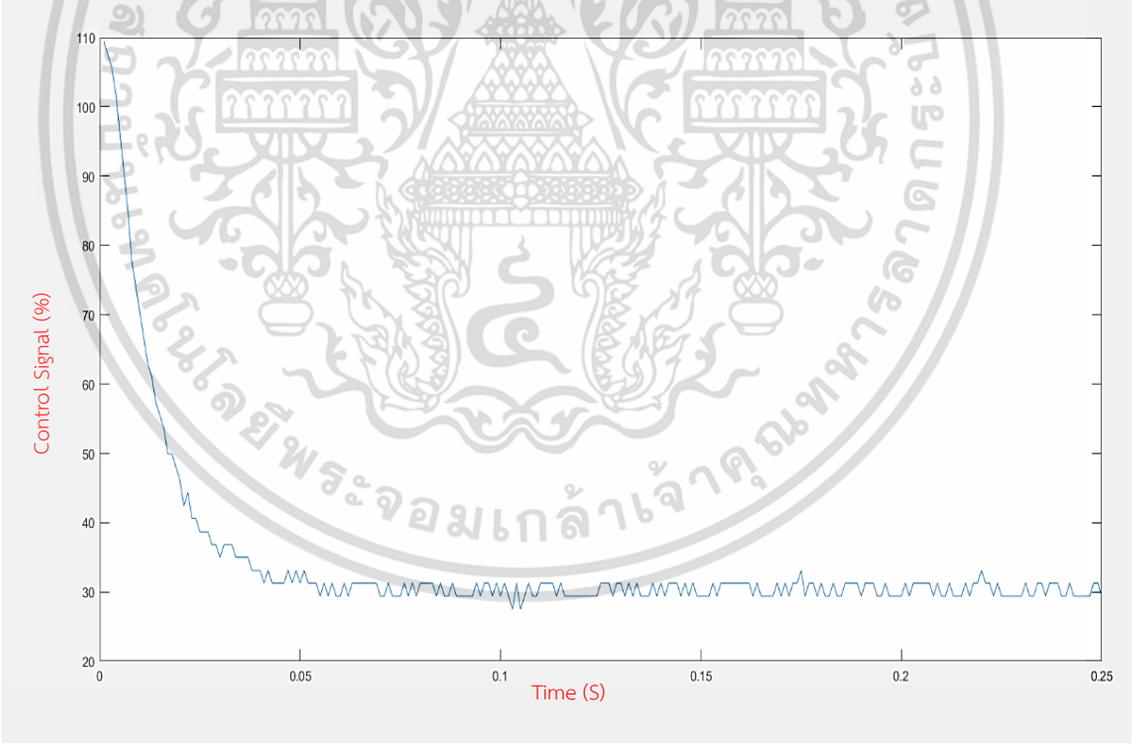
รูปที่ 4.14 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 40%



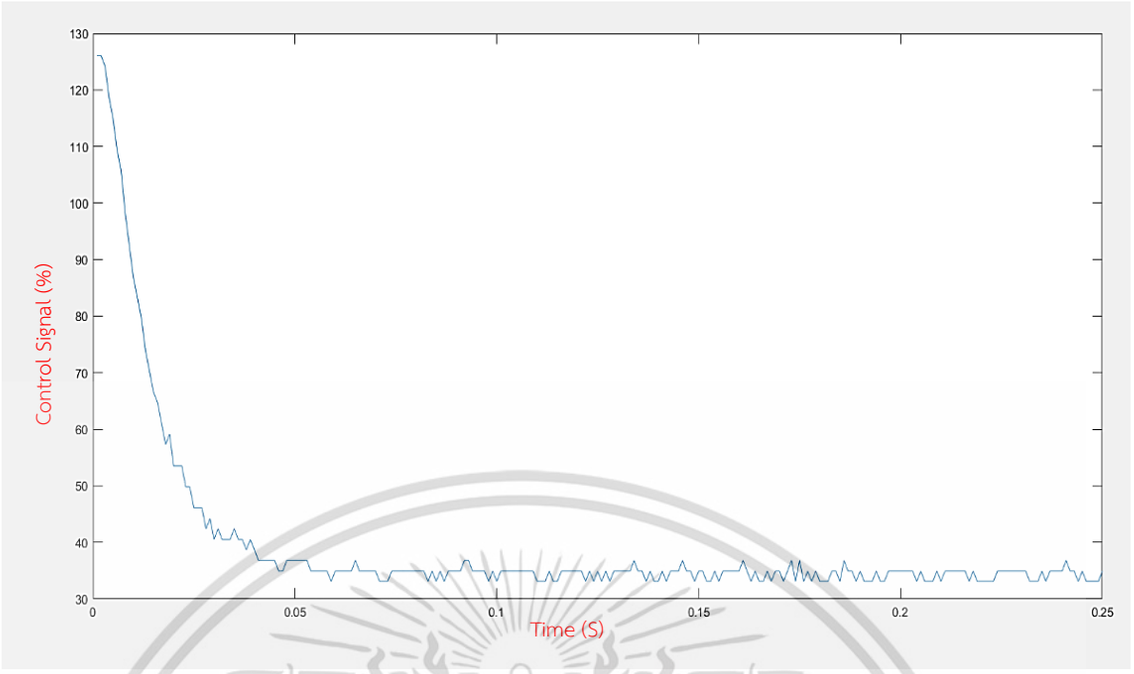
รูปที่ 4.15 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 50%



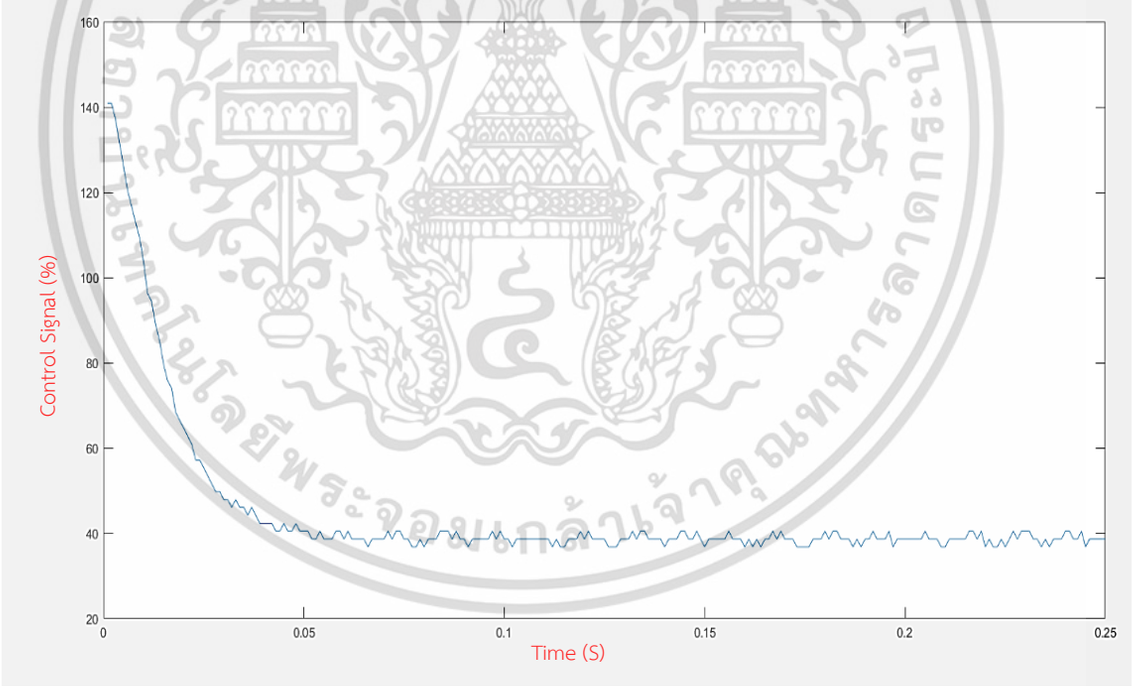
รูปที่ 4.16 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 60%



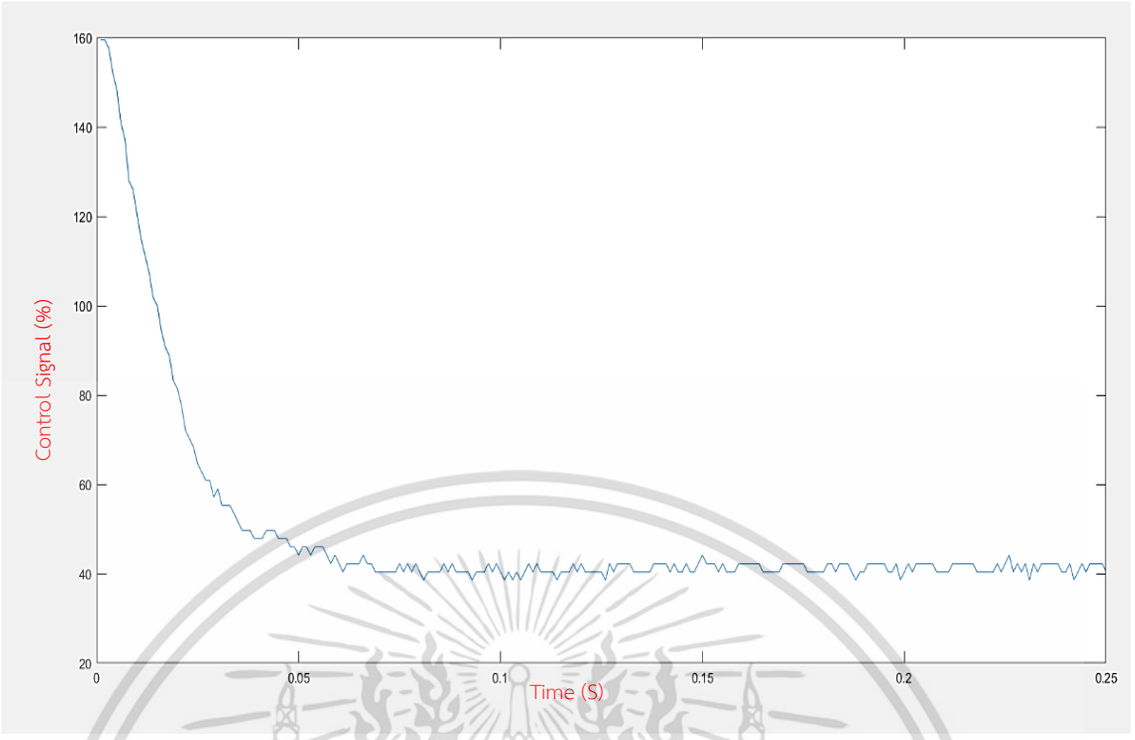
รูปที่ 4.17 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 70%



รูปที่ 4.18 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 80%



รูปที่ 4.19 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 90%



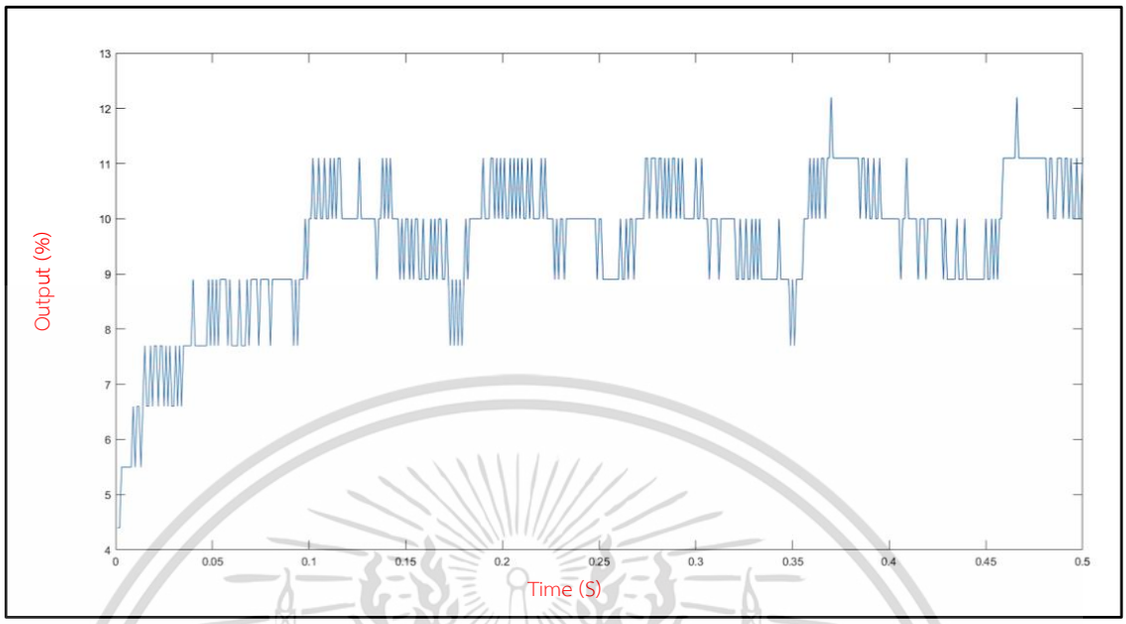
รูปที่ 4.20 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพี ที่ Set Point เท่ากับ 100%

จากกราฟของสัญญาณควบคุมที่ Set Point ค่าต่างๆ พบว่าในช่วงแรกสัญญาณควบคุมจะมีค่ามากเนื่องจากต้องทำให้ผลตอบสนองพุ่งขึ้นไปยังค่า Set Point แต่เมื่อเวลาผ่านไปสัญญาณควบคุมจะมีค่าลดลงเรื่อยๆ จนคงที่สอดคล้องกับผลตอบสนองที่เพิ่มขึ้นอย่างรวดเร็วในตอนแรกและคงที่ในเวลาต่อมา

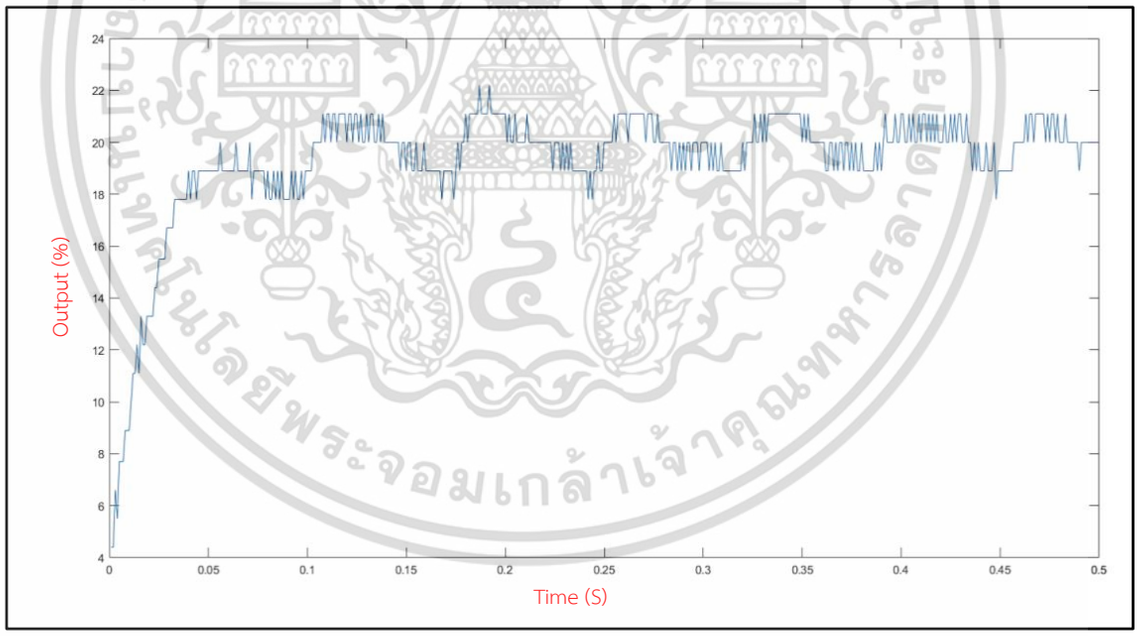
4.2 ผลการทดลองของการควบคุมมอเตอร์โดยใช้ตัวควบคุมแบบพีไอ (PI Controller)

จากการทดลองที่ 3.2.5 ต้องการให้ระบบเข้าสู่สภาวะสมดุลที่เวลา 0.05 วินาที จะได้ค่าคงที่ของตัวควบคุมแบบพี เท่ากับ 1.67 (Kp เท่ากับ 1.67) และค่าคงที่ของตัวควบคุมแบบไอเท่ากับ 0.05 (Ki เท่ากับ 0.05) จากนั้น จึงนำไปทดลองจริงบนมอเตอร์

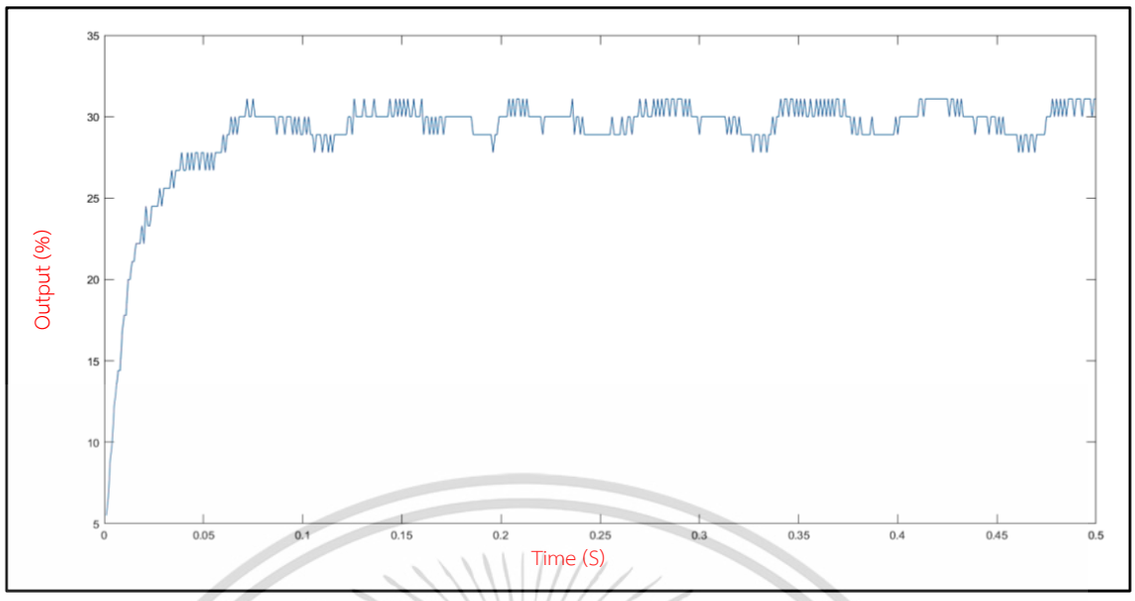
4.2.1 ความเร็วของมอเตอร์ของการใช้ตัวควบคุมแบบพีไอ ในแต่ละค่าของ Set Point



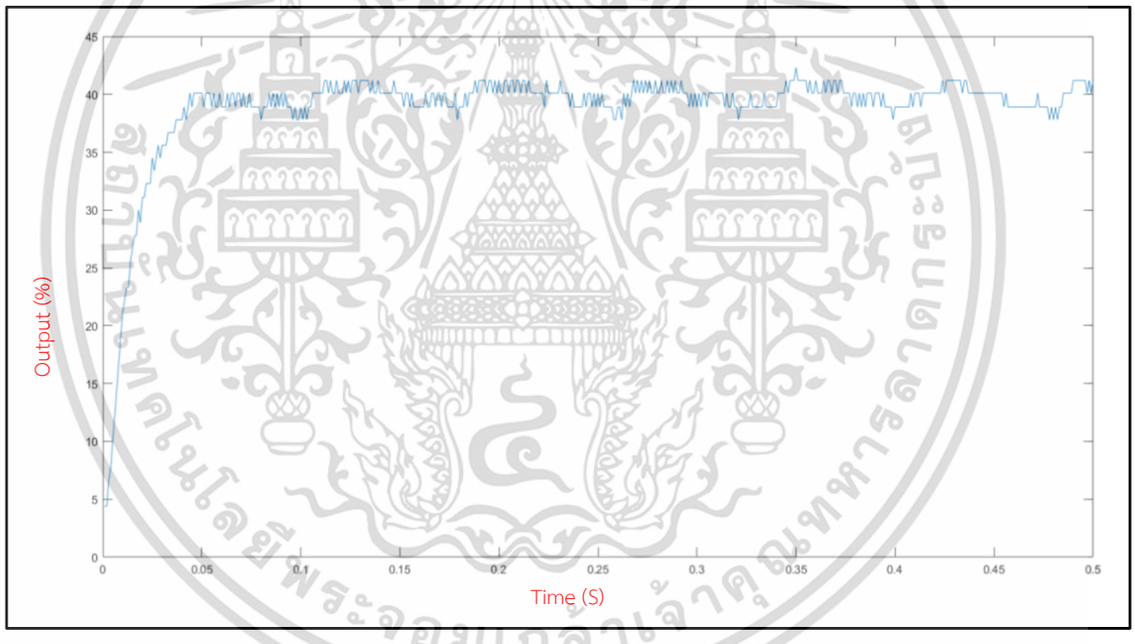
รูปที่ 4.21 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 10%



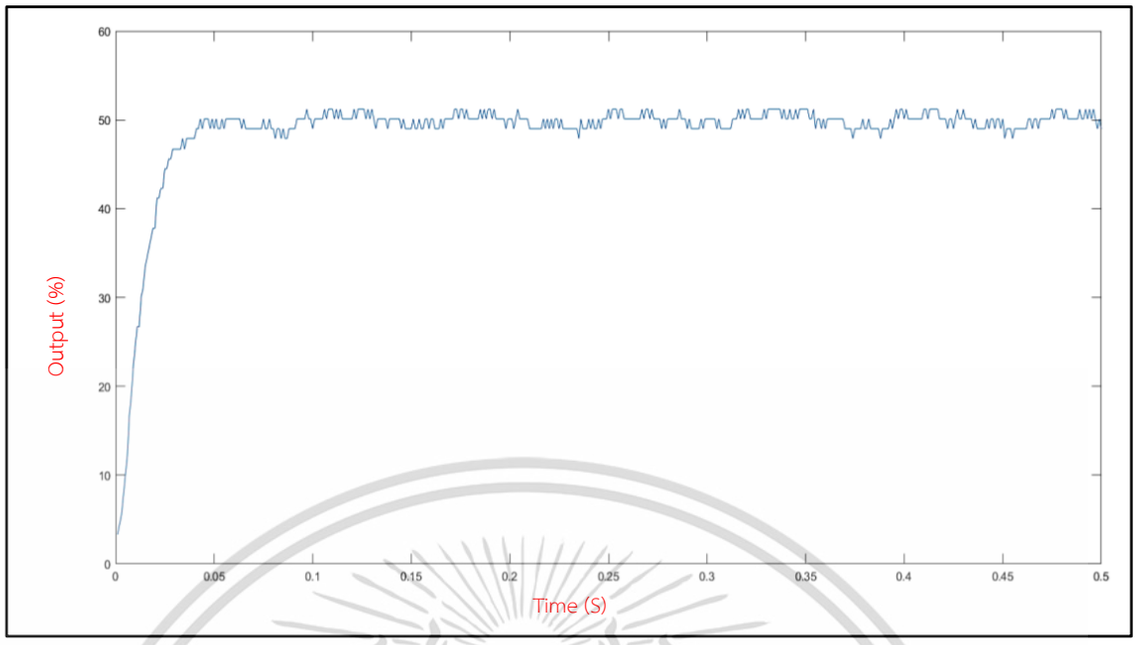
รูปที่ 4.22 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 20%



รูปที่ 4.23 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 30%

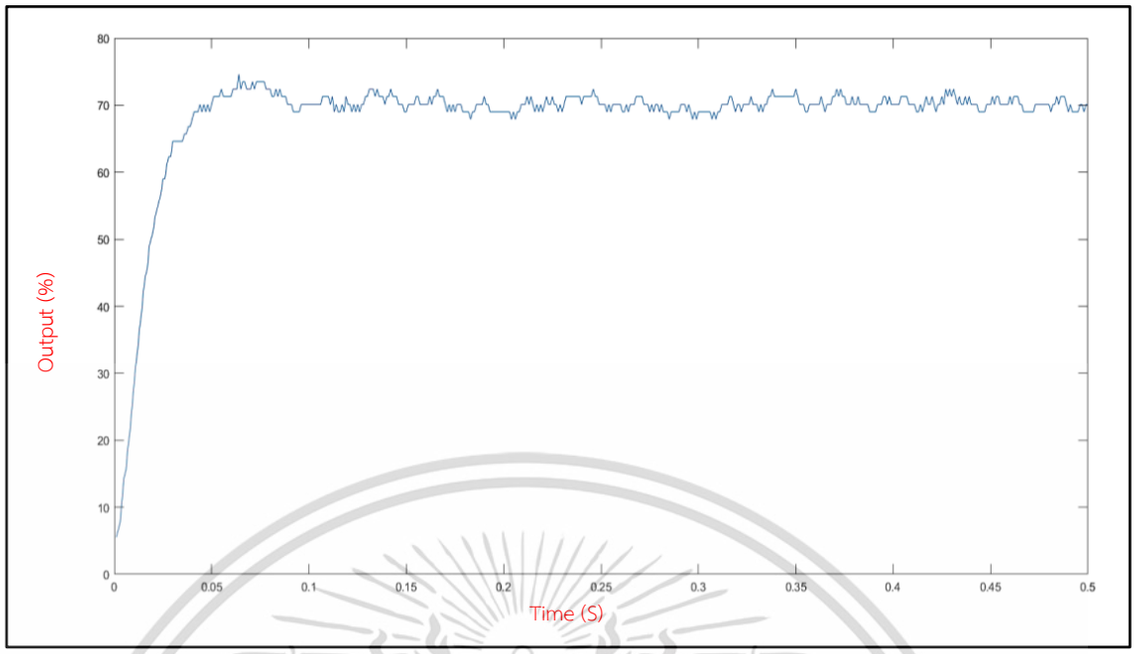


รูปที่ 4.24 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 40%



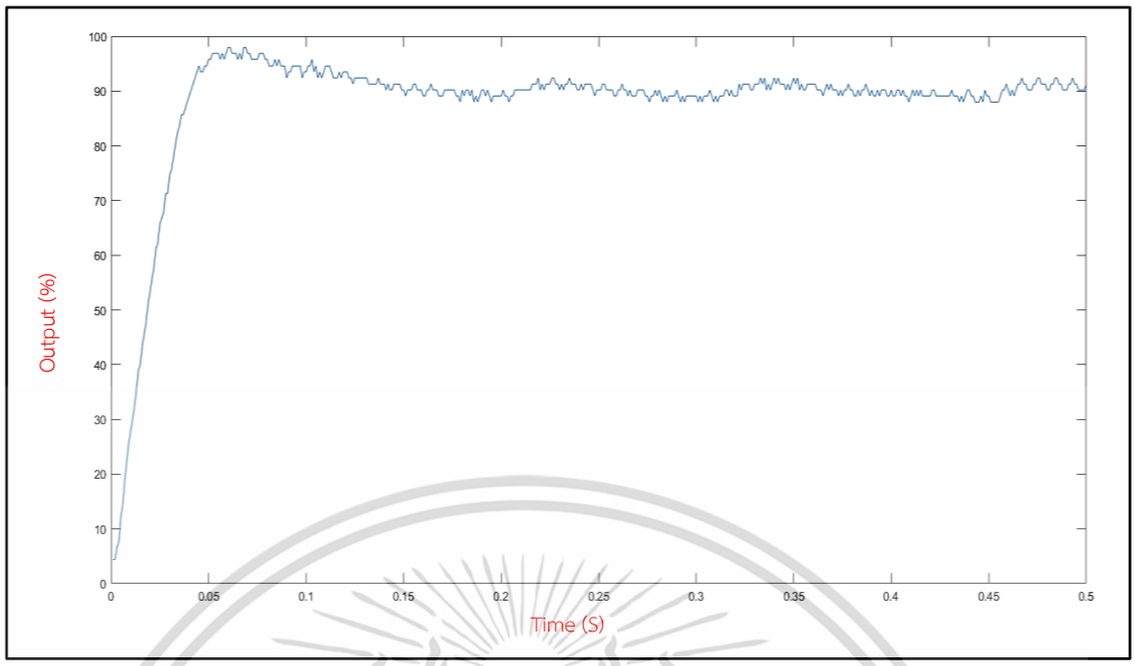
รูปที่ 4.25 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 50%

รูปที่ 4.26 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 60%

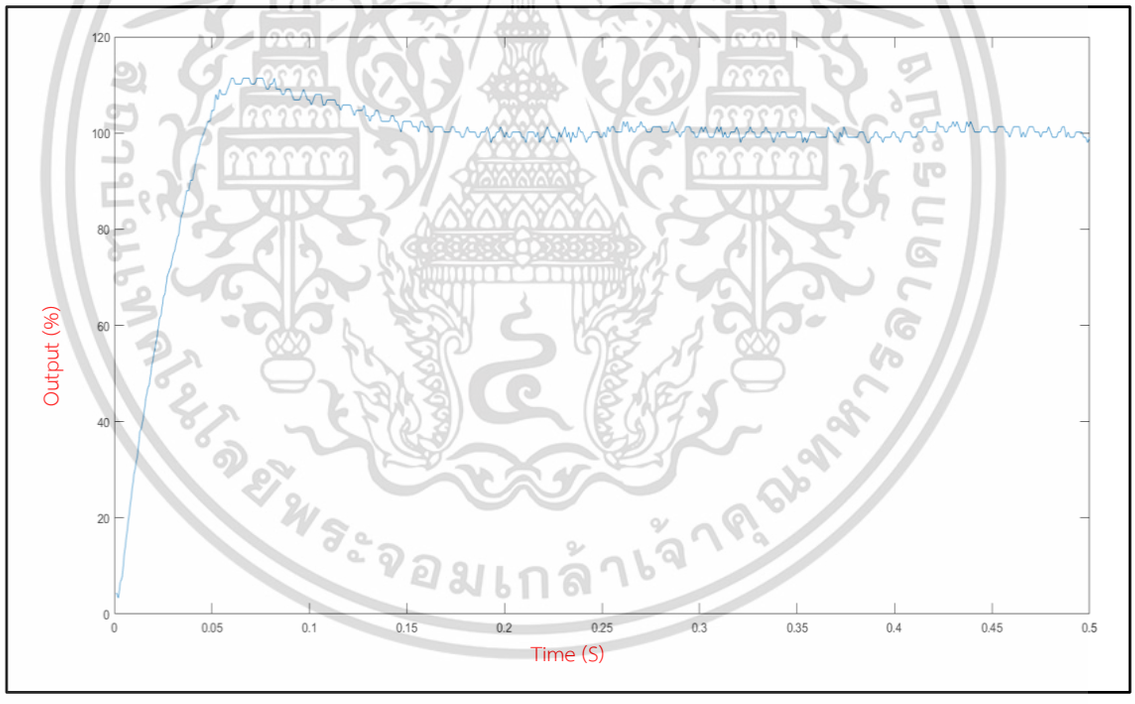


รูปที่ 4.27 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 70%

รูปที่ 4.28 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 80%



รูปที่ 4.29 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 90%

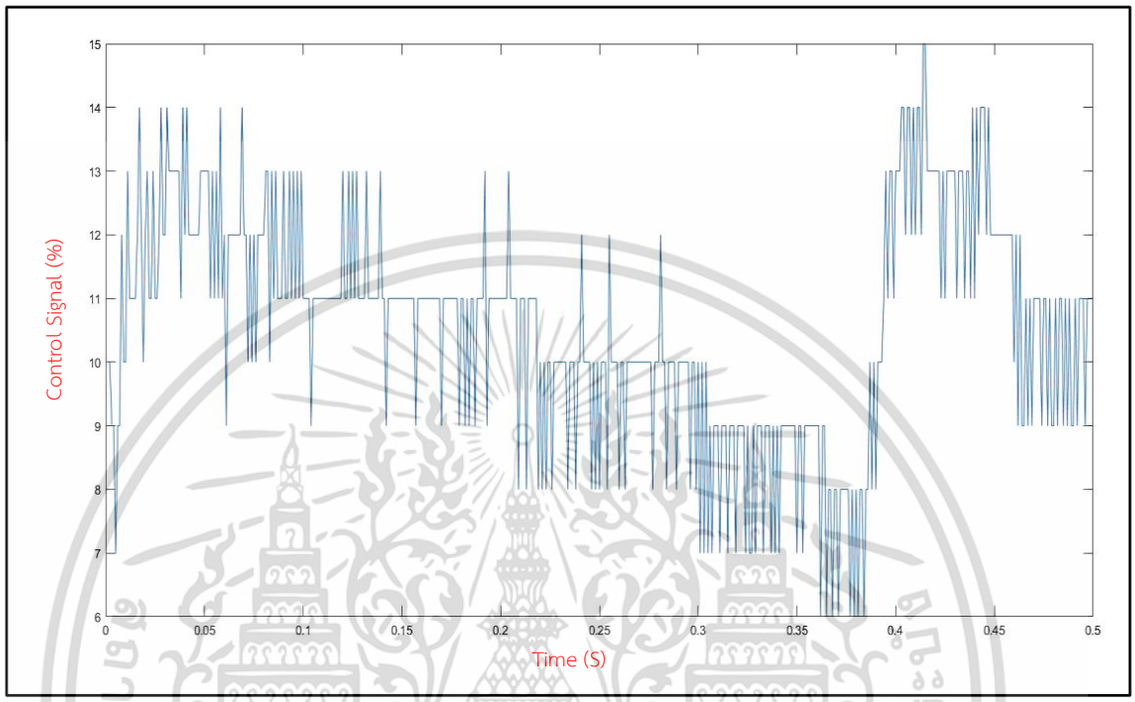


รูปที่ 4.30 แสดงผลตอบสนองของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 100%

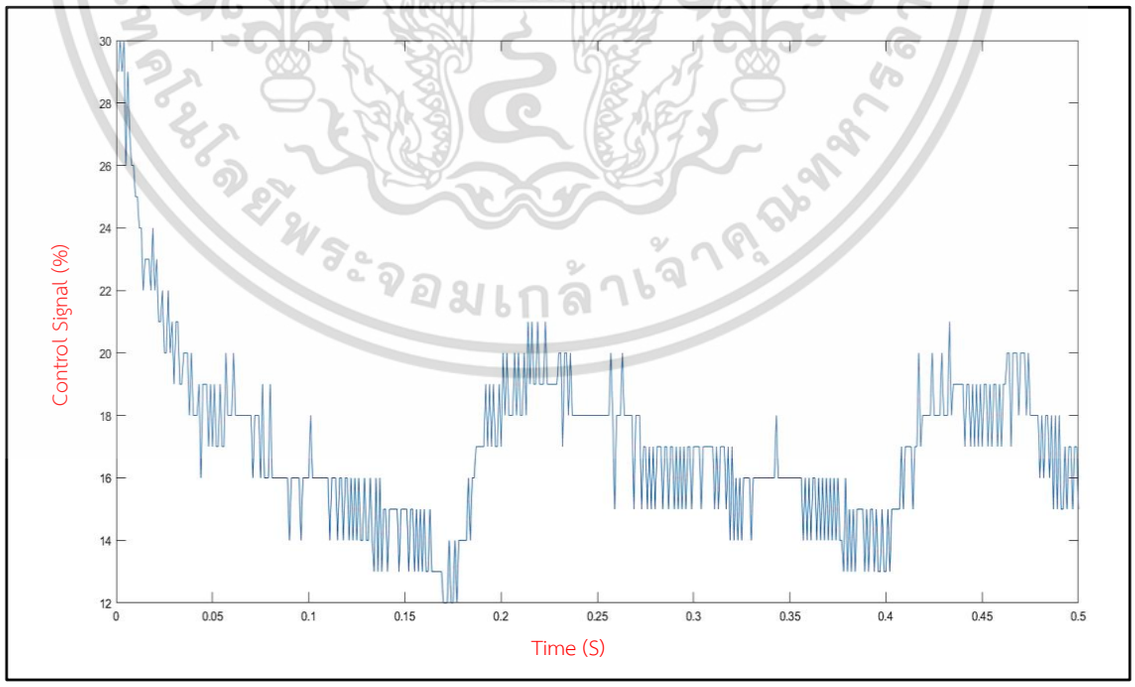
จากกราฟผลตอบสนองของระบบ เมื่อมีการใช้ตัวควบคุมแบบพีไอควบคุมความเร็วมอเตอร์ พบว่า ผลตอบสนองของระบบสามารถไปยังค่าที่ต้องการได้ เมื่อค่า Set Point มีค่าไม่มาก Overshoot ของระบบจะมีค่าน้อยหรือมีค่าเป็นศูนย์ แต่เมื่อค่าของ Set Point มีค่าใกล้ 100 ระบบจะเกิด Overshoot เนื่องจากสัญญาณควบคุมที่ส่งออกมาจากตัวควบคุม มีค่ามากกว่าที่บอร์ดขับมอเตอร์จะรับได้ จึงมีเกิดเหตุการณ์ดังกล่าวขึ้น วิธีแก้ไข คือ ใช้ตัวควบคุมแบบดีหรือเทคนิค Anti-Windup ที่ได้กล่าวไว้ในบทที่ 3 หัวข้อที่ 3.2

4.2.2 สัญญาณควบคุมของการใช้ตัวควบคุมแบบพีไอ ในแต่ละค่าของ Set Point

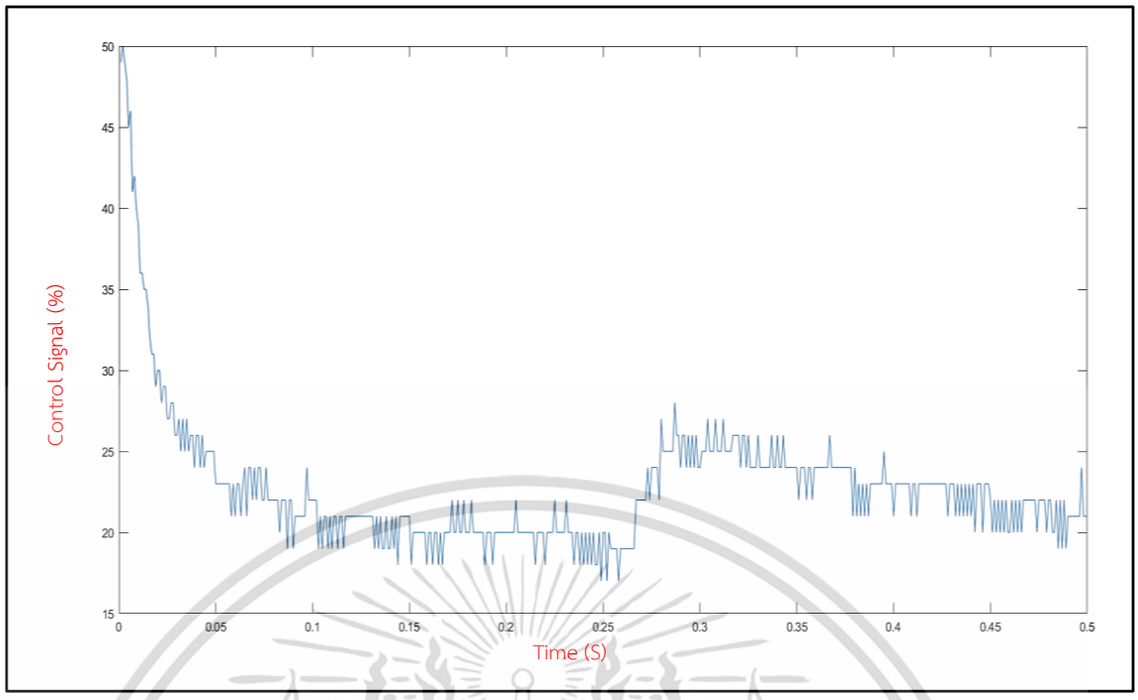
นอกจากการสังเกตผลตอบสนองของระบบแล้ว ในการจำลองการควบคุมนั้นจะดูแนวโน้มของสัญญาณควบคุมด้วย เนื่องจากเป็นตัวบ่งบอกว่าส่งสัญญาณไปให้ระบบถูกต้องตอบผลตอบสนองที่ออกมา หรือไม่มีผลต่างๆ ดังนี้



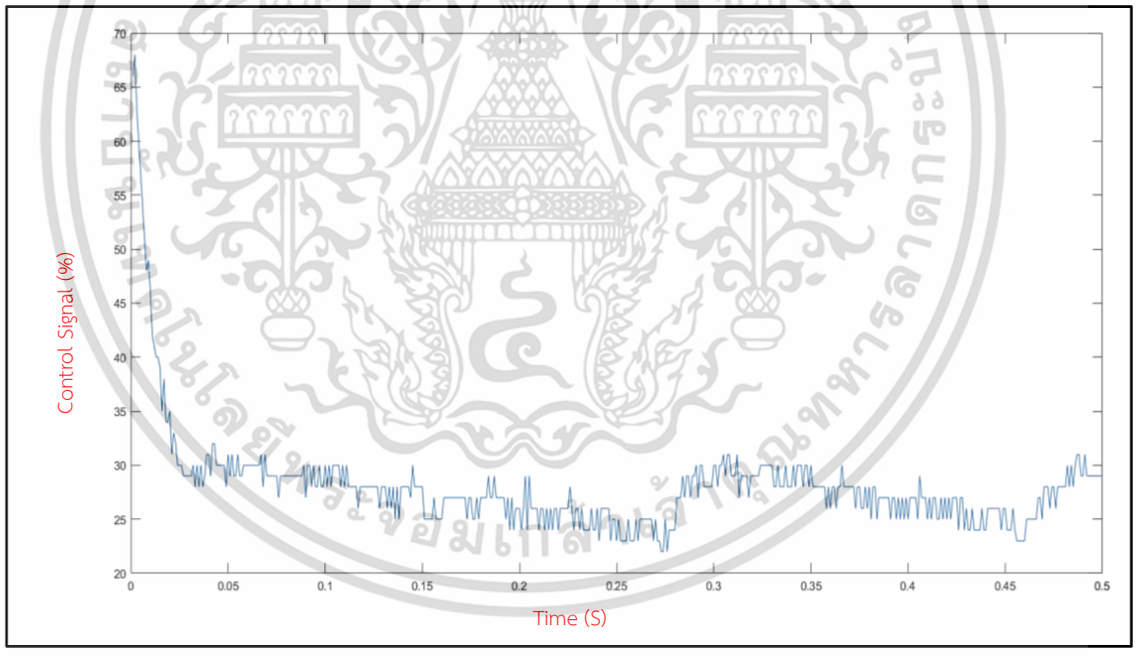
รูปที่ 4.31 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 10%



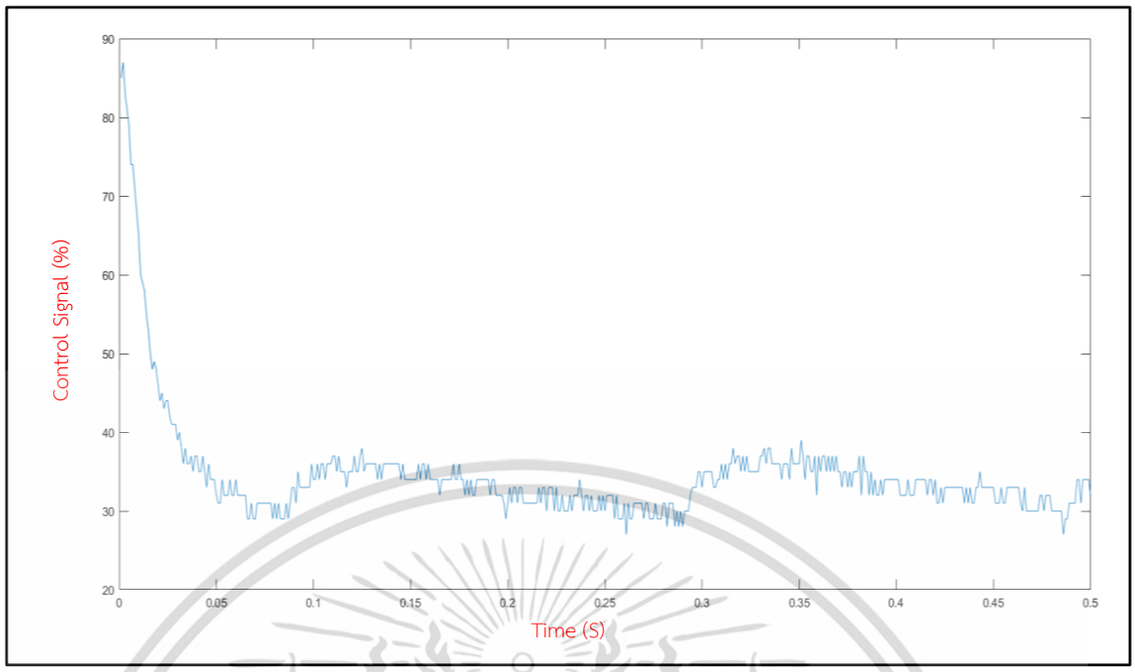
รูปที่ 4.32 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 20%



รูปที่ 4.33 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 30%

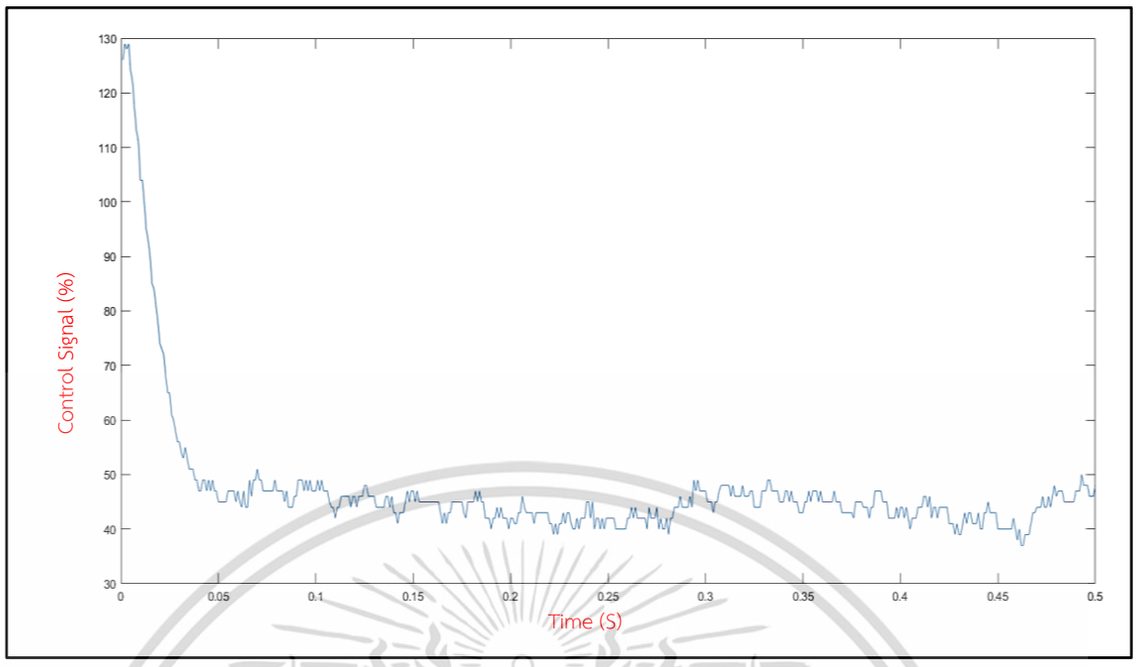


รูปที่ 4.34 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 40%

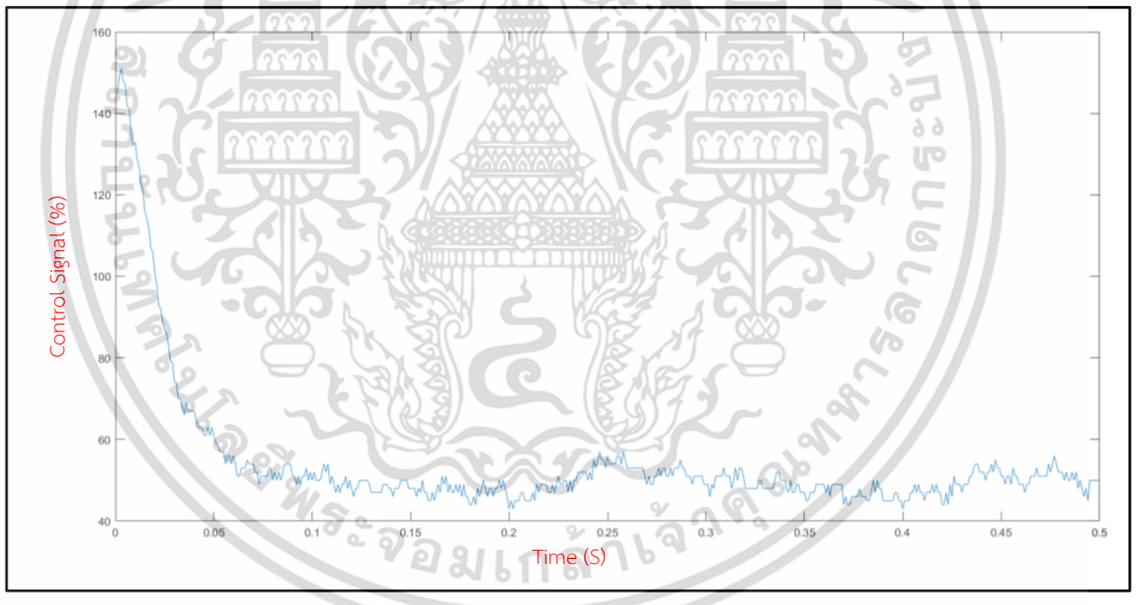


รูปที่ 4.35 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 50%

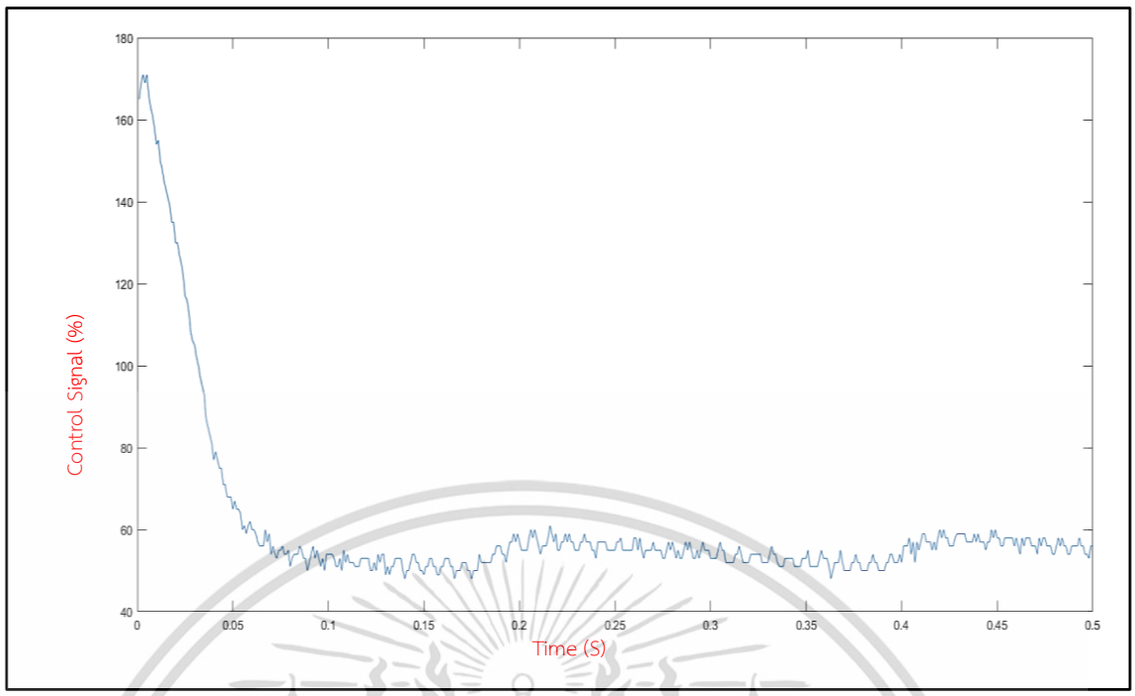
รูปที่ 4.36 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 60%



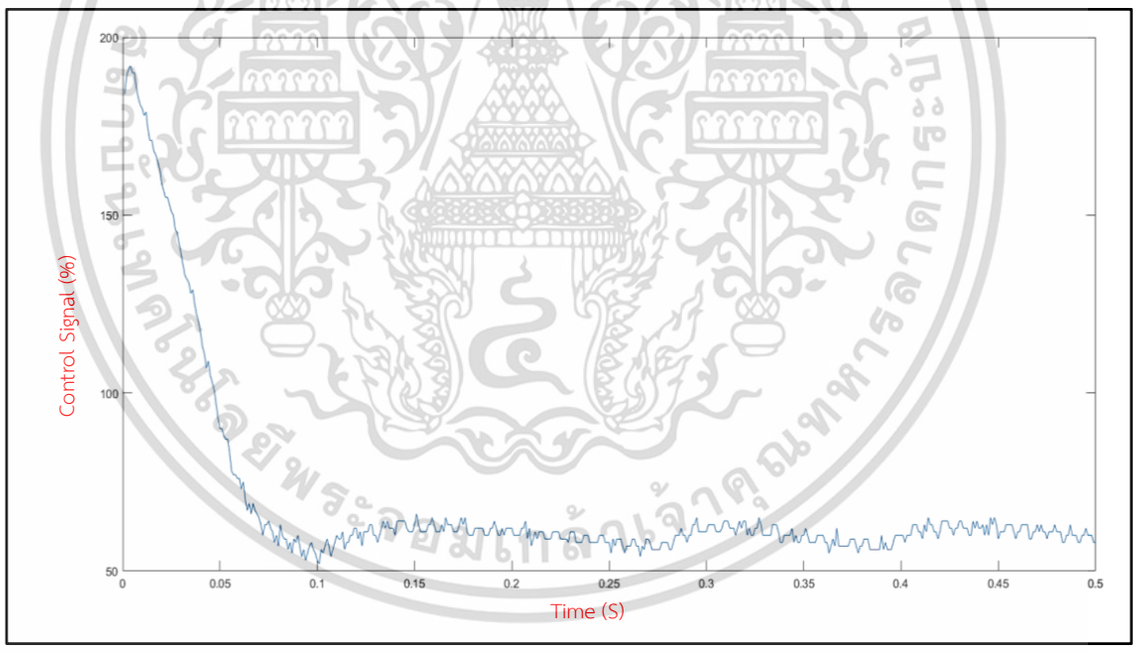
รูปที่ 4.37 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 70%



รูปที่ 4.38 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 80%



รูปที่ 4.39 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 90%



รูปที่ 4.40 แสดงสัญญาณควบคุมของระบบ ที่ใช้ตัวควบคุมแบบพีไอ ที่ Set Point เท่ากับ 100%

จากกราฟของสัญญาณควบคุมที่ Set Point ค่าต่างๆ พบว่า ในช่วงแรกของสัญญาณควบคุมจะมีค่ามาก เนื่องจากต้องทำให้ผลตอบสนองพุ่งขึ้นไปยังค่า Set Point ซึ่งมากกว่าสัญญาณควบคุม เมื่อใช้ตัวควบคุมแบบพีเพียงอย่างเดียว ส่งผลให้ผลตอบสนองของระบบมีค่าไปยังเป้าหมายได้ แต่ด้วยสัญญาณที่มากเกินไปทำให้เกิน Overshoot ได้ อาจจะต้องศึกษาวิธีการลด Overshoot ต่อไป

บทที่ 5
สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

จากการที่โครงงานนี้ ได้ทำการศึกษาตัวควบคุมแบบพีและพีไอ ที่ใช้ในการควบคุมมอเตอร์ไฟฟ้ากระแสตรง เนื่องจากต้องการควบคุมมอเตอร์ไฟฟ้ากระแสตรง ให้มีประสิทธิภาพตามที่ต้องการ ทำให้กระบวนการต่างๆ ที่เกี่ยวข้องกับมอเตอร์สามารถทำงานได้อย่างดี

จากการศึกษาพบว่า สามารถควบคุมความเร็วของมอเตอร์ไฟฟ้ากระแสตรงได้อย่างดี เมื่อนำตัวควบคุมแบบพีกับไปใช้งานมอเตอร์ จะทำให้ผลตอบสนองของระบบไปยังค่า Set Point ที่ต้องการได้ แต่จะมีประสิทธิภาพเมื่อ Set Point มีค่าน้อยๆ เมื่อค่า Set Point เพิ่มขึ้น จะทำให้ระบบไม่สามารถไปยังค่าที่ต้องการได้ แต่เมื่อมีการใช้ตัวควบคุมแบบพีไอ ผลตอบสนองของระบบสามารถไปยังค่า Set Point ที่ต้องการได้ สามารถปรับระยะเวลาที่ต้องการให้ระบบเข้าสู่สภาวะสมดุลได้ จากการจำลองโดยใช้โปรแกรม MATLAB แล้วนำมาใส่เข้าไปในโปรแกรมของไมโครคอนโทรลเลอร์ จึงสรุปได้ว่าการศึกษาในทางทฤษฎีต่างๆ ของการควบคุม สามารถนำมาออกแบบตัวควบคุมเพื่อใช้งานกับมอเตอร์ไฟฟ้ากระแสตรงได้

5.2 ข้อเสนอแนะ

ตัวควบคุมที่ได้ทำการออกแบบมานั้น สามารถนำไปควบคุมความเร็วของมอเตอร์ได้เพียงอย่างเดียว หากต้องการควบคุมพารามิเตอร์ตัวอื่นๆ ของมอเตอร์ สามารถทำได้โดยการหาฟังก์ชันถ่ายโอนของระบบ โดยที่เอาต์พุตที่ออกมานั้นต้องเป็นสิ่งที่ต้องการจะควบคุม หากต้องการควบคุมพารามิเตอร์หลายๆ ตัวพร้อมกัน ขั้นตอนและวิธีการออกแบบจะมีความคล้ายคลึง แต่ระบบจะมีความซับซ้อนเพิ่มมากขึ้น

บรรณานุกรม

[1] Greeshma Sarah John and Abhilash Vijayan T., “Anti-windup PI Controller for Speed Control of Brushless DC Motor”, 2017.

[2] Ihechiluru Okoro, “Feedback–Feedforward Compensation of a DC Motor”, 2019.

[3] Kruthika, K., Kiran Kumar, B.M. and Sanjay Lakshminarayanan, “Design and Development of a Robotic Arm”, 2016.

[4] Lelai Zhou, Shaoping Bai and Michael Rygaard Hansen, “Design Optimization on the Drive Train of a Light-Weight Robotic Arm”, pp. 560-569, 2011.

[5] Md Akram Ahmad, “Speed Control of a DC Motor Using Controllers”, 2014.

[6] Myo Maung, Maung Maung Latt and Chaw Myat New, “DC Motor Angular Position Control Using PID Controller with Friction Compensation”, 2018.

[7] Vinod Karvande and Akash Baviskar, “Speed Control of DC Motor by Using PWM Technique Project Report”, 2015.

[8] Wang Xitai, Zhang Xuexiu, Li Lifeng and Liu Bingshan, “Brushless DC Motor Speed Control System of the Walking Aids Machine”, 2009.

ภาคผนวก

15.4 TIMx registers

Refer to *Section 2.2* for a list of abbreviations used in register descriptions.

The 32-bit peripheral registers have to be written by words (32 bits). All other peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

15.4.1 TIMx control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CKD**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

00: $t_{DTS} = t_{CK\_INT}$

01: $t_{DTS} = 2 \times t_{CK\_INT}$

10: $t_{DTS} = 4 \times t_{CK\_INT}$

11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:5 **CMS**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)