การประยุกต์การควบคุมด้วย PLC/PID ของวัตถุทรงกลมเคลื่อนที่แบบสมดุลของแรงในระนาบเดียว

The Applying PLC/PID Control of Counterbalance Sphere Object On Single Plane

ธนกร ด้วงสีทอง

นาทบดี อริยศิริชาติ

พรวุฒิ วังซ้าย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2562

The Applying PLC/PID Control of Counterbalance Sphere Object On Single Plane

Thanakorn Duangseethong

Nartbodee Ariyasirichart

Pornwut Wangsai

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUE OF TECHNOLOGY LARDKRABANG

ACADEMIC YEAR 2019

หัวข้อปริญญานิพนธ์ การประยุกต์การควบคุมด้วย PLC/PID ของวัตถุทรงกลมเคลื่อนที่แบบสมดุลของแรงในระนาบเดียว

The Applying PLC/PID Control of Counterbalance Sphere Object On Single Plane

นักศึกษาผู้จัดทำ นาย ธนกร ด้วงสีทอง รหัสนักศึกษา 59010546

นาย นาทบดี อริยศิริชาติ รหัสนักศึกษา 59010726

นาย พรวุฒิ วังซ้าย รหัสนักศึกษา 59010915

อาจารย์ที่ปรึกษา รศ.ดร. สุพรรณ กุลพาณิชย์

รศ. วิริยะ กองรัตน์

ปีการศึกษา 2562

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอโครงงาน Machine Vision โดยอาศัยแบบจำลองกระบวนการควบคุมการเคลื่อนที่วัตถุทรงกลมให้มีความสมดุลของแรงบนคานระนาบเดียวโดยมีจุดหมุนอยู่ที่กึ่งกลางคานส่วนของการควบคุมจะอาศัยเครื่องควบคุม PLC เป็นเครื่องควบคุมหลักและมี Servo Motor เป็นอุปกรณ์ขับ (Actuator) มีเพลาศูนย์กลางร่วมกับลูกเบี้ยวให้เกิดการเคลื่อนที่ ขึ้น-ลง ของคาน และมีกล้อง Vision Camera ทำงานร่วมกับ บอร์ด Arduino เพื่อตรวจจับการเคลื่อนที่และส่งค่าตำแหน่งของวัตถุบนคานแบบ On-Line ผ่านพอร์ทสื่อสารข้อมูล RS232Cด้วย Host-Link Protocol ใช้เป็นสัญญาณป้อนกลับแบบวงรอบปิด (Closed loop Control) ให้กับเครื่องควบคุม PLC ในการประมวลผลจะอาศัยฟังค์ชั่น PID ที่สามารถปรับค่าตัวแปร K_p, T_i และ T_d เพื่อควบคุมให้วัตถุไปหยุดอยู่ที่ตำแหน่งใด ๆ บนคานได้ตามต้องการ ผลการทดลองควบคุมลูกปิงปองที่มีผิวเรียบให้หยุดที่ระยะ 10, 20 และ 30 เซนติเมตร โดยอ้างอิงจากด้านที่มีตะกร้าให้เป็นจุดเริ่มต้น สังเกตผลตอบสนองต่อเวลาการควบคุมของตัวแปรกระบวนการ Set Value(SV) , Present Value(PV) และ Manipulate Value(MV) ในรูปแบบของกราฟแนวโน้ม (Trend Graph) การกำหนดข้อมูลเพื่อการแสดงผลสามารถกระทำได้ที่ HMI ที่ถูกเชื่อมต่อกับ PLC ผ่านพอร์ท RS232C

Thesis Title	The Applying PLC/PID Control of Counterbalance Sphere Object On Single Plane		
Producer	Thanakorn	Duangseethong	59010546
	Nartbodee	Ariyasirichart	59010726
	Pornwut	Wangsai	59010915
Thesis Advisor	Assoc. Prof. Dr. Suphan Kullapanich		
	Assoc. Prof Viriya Kongratana		
Year	2019		

Abstract

This thesis presents a Machine Vision project based on a spherical object movement control model that balances the force on a single plane beam with a rotating point in the center of the beam. The control section relies on the PLC controller as the main controller, and servo motor is an Actuator device with a hub shaft together with the beam's up-down motion cam, and vision camera works with arduino boards to detect motion and send the position of objects on the on-line beam via the RS232C data communication port with host link protocol, Used as a closed loop control signal to the PLC controller, the processing relies on the PID function that can adjust the variables K_p, T_i and T_d. The results of the control of ping pong balls with a smooth surface, stop at a distance of 10, 20 and 30 centimeters, referring to the side with the basket as the starting point by customizing the value of PID in line with that type of object. Observe the response to the control time of the Set Value (SV) process variable, Present Value (PV) and Manipulate Value (MV) in the form of trend graphs, setting data for impressions can be performed at HMI that is connected to PLC through the RS232C port.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จรุร่วงไปได้ด้วยดีด้วยการช่วยเหลือจากหลายท่านโดยเฉพาะอย่างยิ่ง ต้องขอขอบคุณ ผศ.ดร.สุพรรณ กุลพาณิชย์ และ รศ.วิริยะ กองรัตน์ อาจารย์ที่ปรึกษาที่ได้กรุณาให้คำแนะนำและข้อคิดเห็นต่างๆชี้แนะแนวทางในการแก้ปัญหาและประสบการณ์ที่ดีแก่ข้าพเจ้ารวมถึงสนับสนุนอุปกรณ์สำหรับทำโครงงานที่เป็นประโยชน์ต่อโครงงานมาโดยตลอดและได้กรุณาตรวจแก้ไขปริญญานิพนธ์จนสำเร็จเรียบร้อยเป็นอย่างดี

ขอขอบคุณผู้แต่งหนังสือเอกสารอ้างอิงและเว็บไซต์ต่างๆ คณะผู้จัดทำได้นำมาใช้อ้างอิง ประกอบการศึกษา และจัดทำปริญญานิพนธ์ฉบับนี้จนปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องด้วยคณะผู้วิจัยอยากจะศึกษาการทำงานของ PLC ที่ทำงานร่วมกับ Servo Drive เพื่อไปขับเคลื่อน Servo Motor อีกทั้งยังรวมถึงการทำงานร่วมกันของ HMI กับ PLC ผ่านโปรแกรม NB-Designer & CX-Programmer และ ยังมีการใช้งานกล้อง Pixy camera กับ Arduino UNO ที่ต่อเข้ากับ PLC

1.2 วัตถุประสงค์ของโครงการ

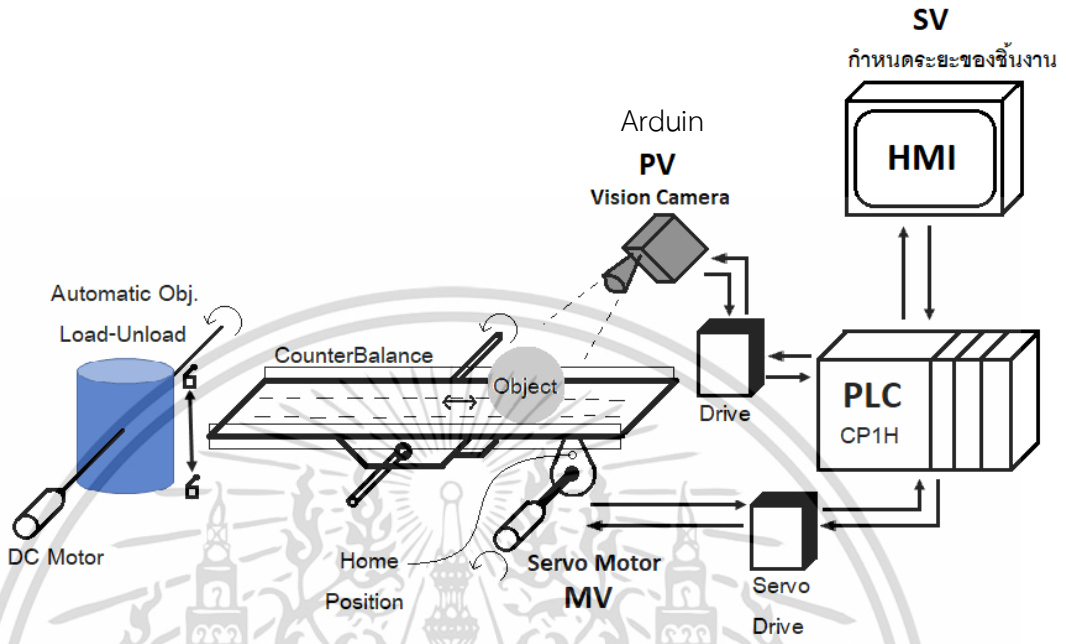
1.2.1 เพื่อศึกษาการควบคุมการเคลื่อนที่ทรงกลมในระนาบเดียว โดยอาศัยเครื่องควบคุม PLC และ Servo Motor ซึ่งอาศัยโปรแกรม CX-programmer

1.2.2 เพื่อศึกษาการตรวจจับตำแหน่งวัตถุทรงกลม โดยอาศัยกล้องตรวจจับ Pixy ที่ใช้ร่วมกับ Arduino

1.2.3 เพื่อศึกษาหน้าจอ HMI ที่ใช้ร่วมกับ PLC ซึ่งอาศัยโปรแกรม NB-Designer

1.3 ขอบเขตโครงงาน

ในการศึกษาการควบคุมสมดุลของวัตถุทรงกลมบนคานในระนาบเดียว จะใช้หลักการการสมดุลของแรงที่เกิดจากระยะหรือตำแหน่งของวัตถุทรงกลม ในการควบคุมจะอาศัยเครื่องควบคุม PLC & Servo Motor ซึ่งมีกล้อง Vision Camera เป็นตัวตรวจจับตำแหน่งของวัตถุทรงกลม แล้วส่งค่าตำแหน่งปัจจุบันไปยัง PLC เพื่อเปรียบเทียบกับตำแหน่ง Set Point เครื่องควบคุมจะทำการวิเคราะห์และประมวลผลให้กับ Servo Motor ที่เป็น MV (Manipulate Value) ของระบบควบคุมในรูปแบบการควบคุมแบบวงรอบปิด ดังนั้นวัตถุทรงกลมสามารถเคลื่อนที่ไปยังตำแหน่งที่ต้องการได้ การทำงานโดยรวมจะแสดงในรูปที่ 1.1



รูปที่ 1.1 โครงสร้างของ PID CONTROL OF COUNTERBALANCE SPHERE OBJECT ON SINGLE PLANE

1.4 วิธีที่ใช้ในโครงงาน

1.4.1 ศึกษาการทำงานของ PLC (Programmable Logic Controller) ซึ่งใช้เป็นอุปกรณ์ควบคุม

1.4.2 ศึกษาการทำงานของกล้อง PIXY MON ซึ่งใช้เป็นอุปกรณ์ใช้จับตำแหน่งของวัตถุ

1.4.3 ศึกษาการทำงานของ Arduino UNO ซึ่งใช้เป็นอุปกรณ์ที่ใช้ในการแปลงสัญญาณตำแหน่งจากกล้อง PIXY MON ให้แก PLC

1.4.4 ศึกษาการทำงานของการส่งสัญญาณแบบ Serial Port ระหว่าง Arduino กับ PLC และหน้าจอ HMI กับ PLC โดยผ่านสายส่ง RS-232

1.4.5 ศึกษาการทำงานของจอ HMI ซึ่งเป็นหน้าที่แสดงผล ควบคุมคำสั่งต่างๆ และกำหนดค่า Set point และ PID ของระบบ

1.4.6 ศึกษาการโปรแกรม CX-Programmer ซึ่งเป็นโปรแกรมที่ใช้สำหรับ PLC

1.4.7 ศึกษาการโปรแกรม NB-Designer ซึ่งเป็นโปรแกรมที่ใช้สำหรับหน้าจอ HMI

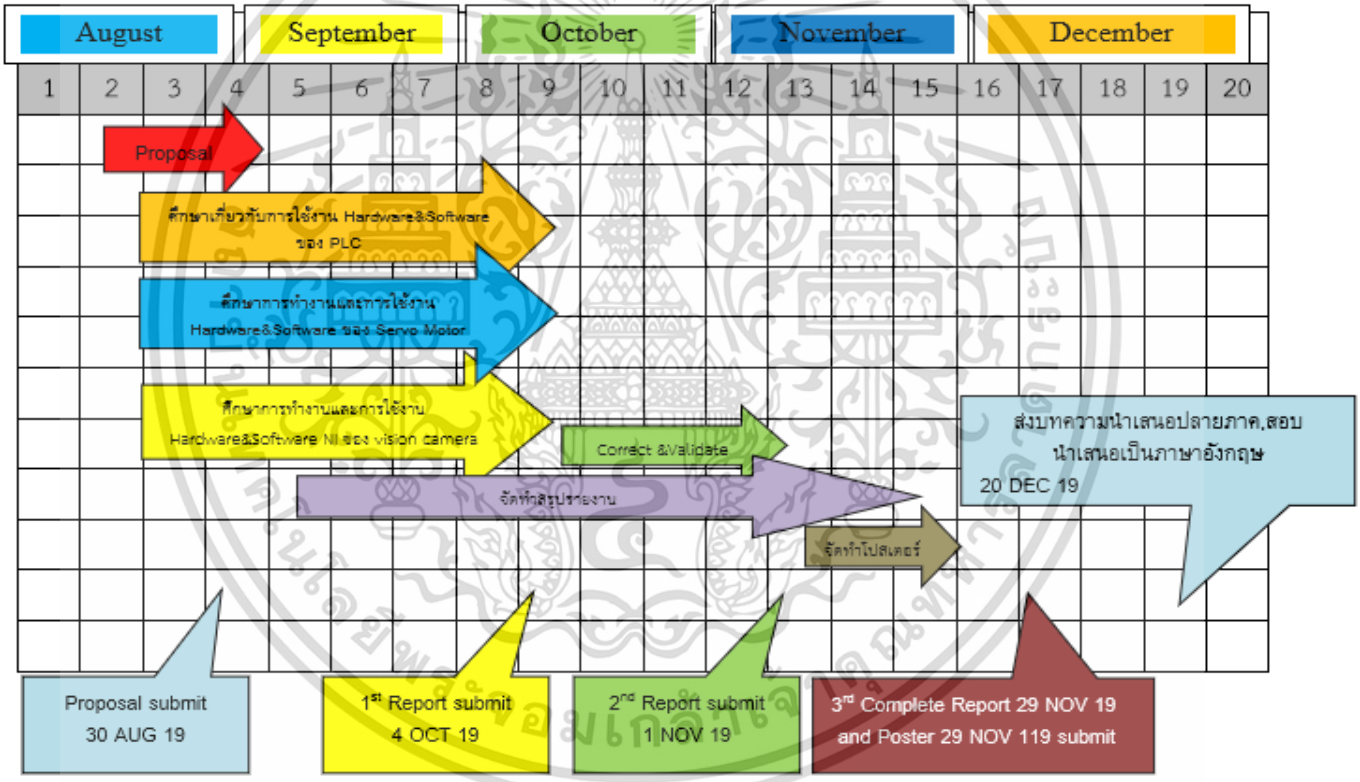
1.4.8 ศึกษาการโปรแกรม Arduino IDE ซึ่งเป็นโปรแกรมที่ใช้สำหรับ Arduino

1.4.9 ศึกษาการควบคุมวัตถุให้สมดุลบนคานด้วยมือ เพื่อให้เข้าใจการเคลื่อนที่ของวัตถุ

1.4.10 ศึกษาการควบคุมแบบ PID ศึกษาค่าพารามิเตอร์ต่างๆของ PID เพื่อการควบคุมที่ดีที่สุด

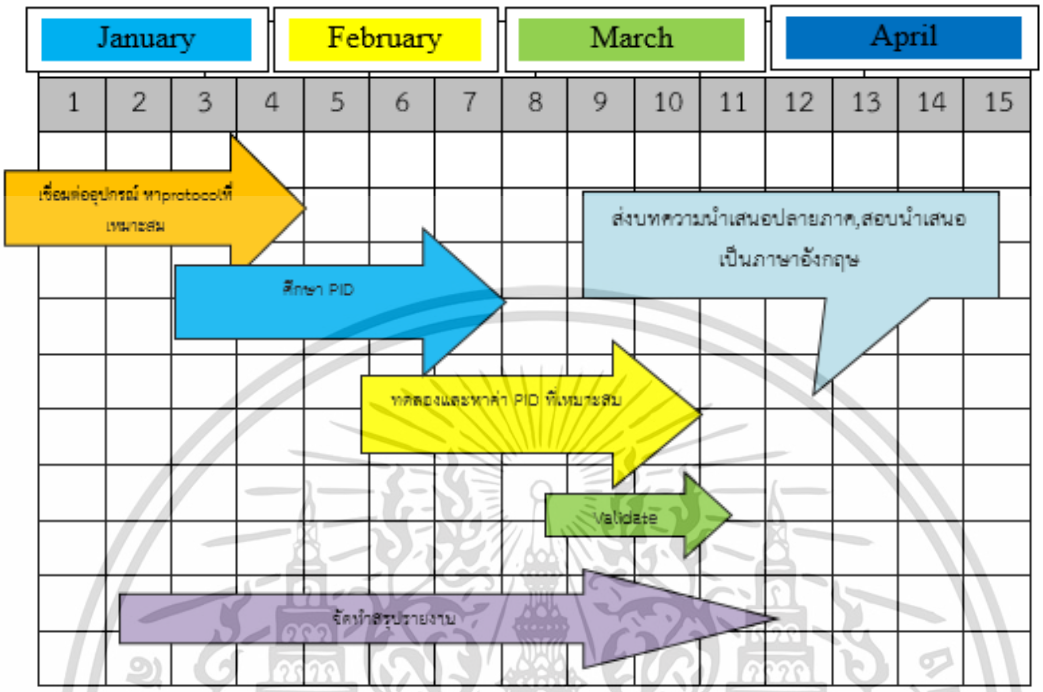
1.5 แผนการดำเนินโครงงาน

1.5.1 แผนการดำเนินโครงงานภาคเรียนที่ 1



ตารางที่ 1.1 แผนการดำเนินโครงงานภาคเรียนที่ 1

1.5.2 แผนการดำเนินโครงงานภาคเรียนที่ 2



ตารางที่ 1.2 แผนการดำเนินโครงงานภาคเรียนที่ 2

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1.6.1 เป็นต้นแบบในการศึกษาหาความรู้เกี่ยวกับการใช้กล้องตรวจจับวัตถุ แล้วใช้ Servo ที่ควบคุมโดย PLC ขับเคลื่อนวัตถุกลับมายัง Set point

1.6.2 ได้ความรู้ในการเขียนโปรแกรมควบคุมการทำงานของ PLC (Programmable Logic Controller) และการต่อวงจรควบคุมการทำงานของอุปกรณ์

1.6.3 ในโครงงานนี้ประกอบไปด้วยองค์ความรู้เพื่อการศึกษาระบบควบคุมแบบปิด หรือ Closed loop Control

1.6.4 องค์ความรู้เพื่อการศึกษาในเรื่องการใช้งาน Hardware อันประกอบไปด้วย PLC, Arduino, Servo Motor, Servo Drive, Pixy Camera และ HMI

1.6.5 องค์ความรู้เพื่อการศึกษาในเรื่องการใช้งาน Software อันประกอบไปด้วย CX-Programmer, Arduino IDE, NI-Vision Builder และ NB-Designer

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

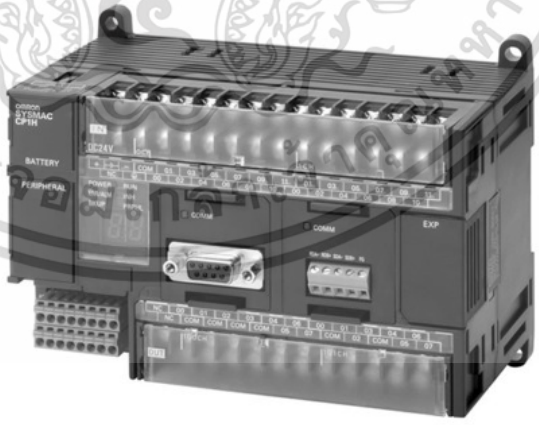
2.1 Programmable Logic Control (PLC)

2.1.1 Programmable Logic Control (PLC) คืออะไร

Programmable Logic Control (PLC) เป็นอุปกรณ์ควบคุมการทำงานของเครื่องจักรหรือกระบวนการทำงานต่างๆ โดยภายในมี Microprocessor เป็นมันสมองสั่งการที่สำคัญ PLC จะมีส่วนที่เป็นอินพุตและเอาต์พุตที่สามารถต่อออกไปใช้งานได้ทันที ตัวตรวจวัดหรือสวิตซ์ต่างๆ จะต่อเข้ากับอินพุต ส่วนเอาต์พุตจะใช้ต่อออกไปควบคุมการทำงานของอุปกรณ์หรือเครื่องจักรที่เป็นเป้าหมาย เราสามารถสร้างวงจรหรือแบบของการควบคุมได้โดยการป้อนเป็นโปรแกรมคำสั่งเข้าไปใน PLC นอกจากนี้ยังสามารถใช้งานร่วมกับอุปกรณ์อื่นเช่น เครื่องอ่านบาร์โค๊ด (Barcode Reader) เครื่องพิมพ์ (Printer) ซึ่งในปัจจุบันนอกจากเครื่อง PLC จะใช้งานแบบเดี่ยว (Stand Alone) แล้วยังสามารถต่อ PLC หลายๆ ตัวเข้าด้วยกัน (Network) เพื่อควบคุมการทำงานของระบบให้มีประสิทธิภาพมากยิ่งขึ้นด้วยจะเห็นได้ว่าการใช้งาน PLC มีความยืดหยุ่นมากดังนั้นในโรงงานอุตสาหกรรม

2.1.2 Programmable Logic Control (PLC): Omron CP1H-XA

ในโครงการนี้จะเลือกใช้เป็น PLC CP1H-XA ซึ่งเป็น PLC ของบริษัท Omron



รูปที่ 2.1 PLC CP1H-XA ของบริษัท Omron

โดยมีคุณสมบัติทางเทคนิคเฉพาะ ดังต่อไปนี้

Type		X CPU Units	XA CPU Units	Y CPU Units
Model		CP1H-X40DR-A CP1H-X40DT-D CP1H-X40DT1-D	CP1H-XA40DR-A CP1H-XA40DT-D CP1H-XA40DT1-D	CP1H-Y20DT-D
I/O Areas	Input bits	272 bits (17 words): CIO 0.00 to CIO 16.15		
	Output bits	272 bits (17 words): CIO 100.00 to CIO 116.15		
	Built-in Analog Input Area	---	CIO 200 to CIO 203	---
	Built-in Analog Output Area	---	CIO 210 to CIO 211	---
	Data Link Area	3,200 bits (200 words): CIO 1000.00 to CIO 1119.15 (words CIO 1000 to CIO 1119)		
	CJ-series CPU Bus Unit area	6,400 bits (400 words): CIO 1500.00 to CIO 1899.15 (words CIO 1500 to CIO 1899)		
	CJ-series Special I/O Unit Area	15,360 bits (960 words): CIO 2000.00 to CIO 2959.15 (words CIO 2000 to CIO 2959)		
	Serial PLC Link Area	1,440 bits (90 words): CIO 3100.00 to CIO 3189.15 (words CIO 3100 to CIO 3189)		
	DeviceNet Area	9,600 bits (600 words): CIO 3200.00 to CIO 3799.15 (words CIO 3200 to CIO 3799)		
	Work bits	4,800 bits (300 words): CIO 1200.00 to CIO 1499.15 (words CIO 1200 to CIO 1499) 37,504 bits (2,344 words): CIO 3800.00 to CIO 6143.15 (words CIO 3800 to CIO 6143)		
Work bits		8,192 bits (512 words): W000.00 to W511.15 (words W0 to W511)		
TR Area		16 bits: TR0 to TR15		
HR Area		8,192 bits (512 words): H0.00 to H511.15 (words H0 to H511)		
AR Area		Read-only (Write-prohibited) 7,168 bits (448 words): A0.00 to A447.15 (words A0 to A447) Read/Write 8,192 bits (512 words): A448.00 to A959.15 (words A448 to A959)		
Timers		4,096 bits: T0 to T4095		
Counters		4,096 bits: C0 to C4095		
DM Area		32 Kwords: D0 to D32767 **Note** Initial data can be transferred to the CPU Unit's built-in flash memory using the data memory initial data transfer function. A setting in the PLC Setup can be used so that the data in flash memory is transferred to RAM at startup. DM Area words for CJ-series Special I/O Units: D20000 to D29599 (100 words × 96 Units) DM Area words for CJ-series CPU Bus Units: D30000 to D31599 (100 words × 16 Units) DM fixed allocation words for Modbus-RTU Easy Master D32200 to D32249 for Serial Port 1, D32300 to D32349 for Serial Port 2		
Data Register Area		16 registers (16 bits): DR0 to DR15		
Index Register Area		16 registers (16 bits): IR0 to IR15		
Task Flag Area		32 flags (32 bits): TK0000 to TK0031		
Trace Memory		4,000 words (500 samples for the trace data maximum of 31 bits and 6 words.)		

ตารางที่ 2.1 Specification PLC CP1H

2.1.3 ส่วนประกอบของ PLC CP1H-XA

PLC CP1H-XA ประกอบไปด้วยส่วนต่างๆ ดังรูปที่ 2.2

2.1.3.1 Peripheral USB port เป็นจุเชื่อมต่อสำหรับส่งข้อมูลผ่านสาย USB

2.1.3.2 Input terminal block เป็นจุดต่อสัญญาดิจิตอลขาเข้าของ PLC

2.1.3.3 Output terminal block เป็นจุดต่อสัญญาดิจิตอลขาออกของ PLC

2.1.3.4 Two Option Board Slots เป็นช่องสำหรับ RS232, RS422 และ Ethernet

2.1.3.5 Built-in analog input เป็นจุดต่อสัญญาณอนาล็อกขาเข้า

2.1.3.6 Built-in analog output เป็นจุดต่อสัญญาณอนาล็อกขาออกมีเฉพาะแบบ XA

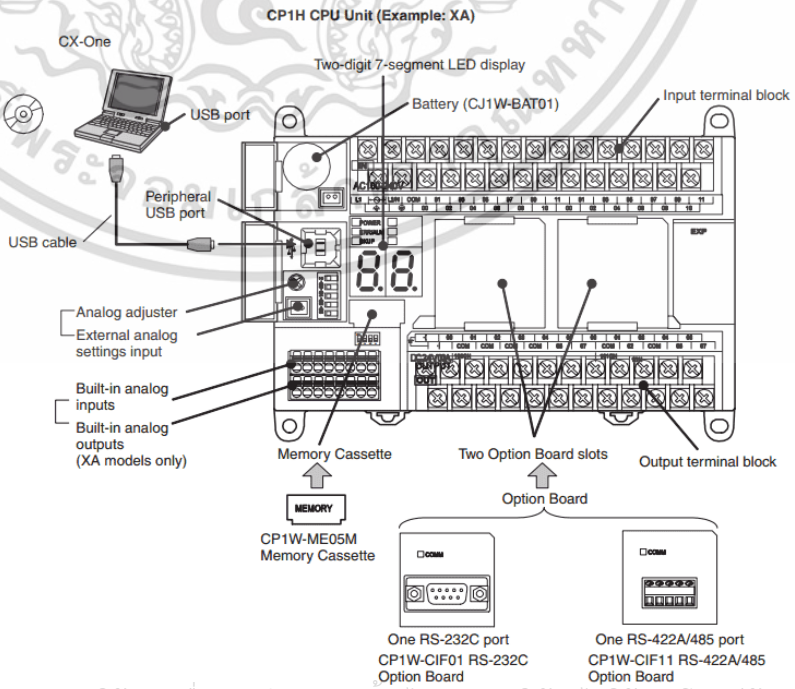
2.1.3.7 Battery (CJ1W-BAT01) เป็นตัวให้พลังงานนำไปสำรองข้อมูลเมื่อขาดไฟเลี้ยง

2.1.3.8 Two-digit 7-segment LED display แสดงผลตัวเลข LED 2 ตำแหน่ง

2.1.3.9 Memory Cassette เป็นหน่วยความจำของ PLC

2.1.3.10 Analog adjuster

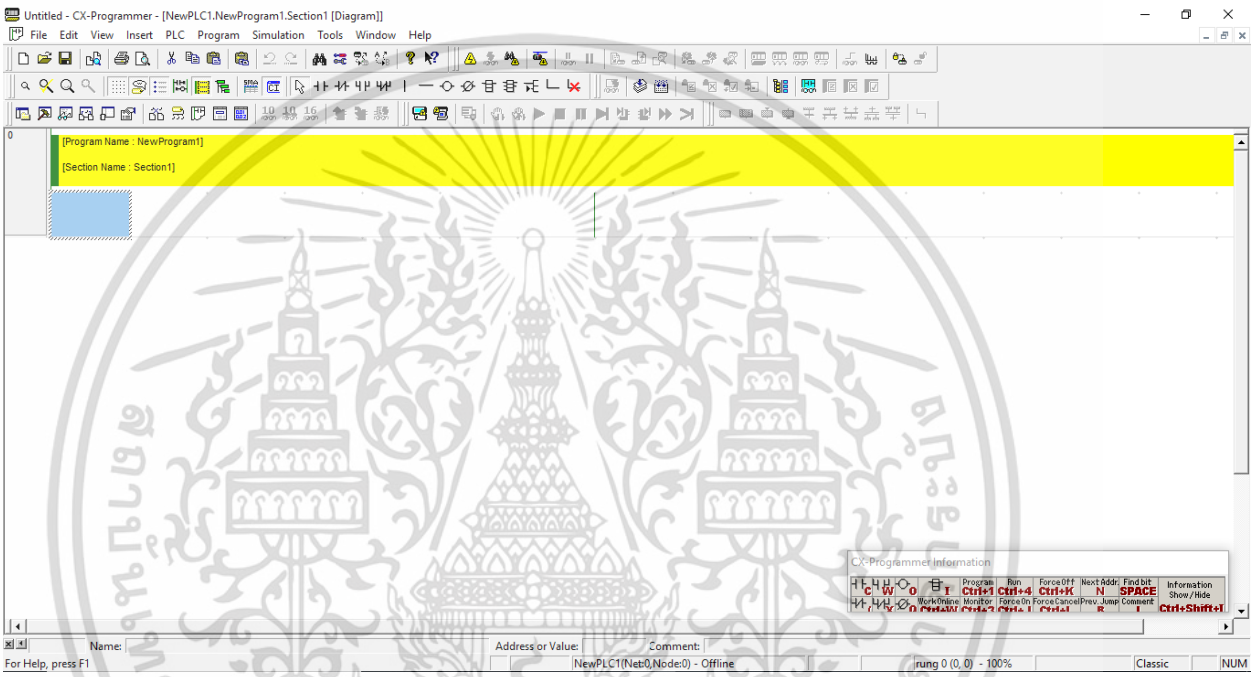
2.1.3.11 External analog settings input



รูปที่ 2.2 ส่วนประกอบ PLC CP1H-XA

2.2 CX-Programmer

CX-Programmer เป็นโปรแกรมที่ใช้สำหรับการติดต่อระหว่าง PLC กับคอมพิวเตอร์ ในการควบคุมคำสั่ง การกำหนดสัญญาณ Input และOutput การเรียงลำดับความสำคัญ การกำหนดลักษณะการตั้งค่า และการกำหนดพื้นที่หน่วยความจำ



รูปที่ 2.3 แสดงโปรแกรม CX-Programmer

2.3 Human Machine Interface (HMI)

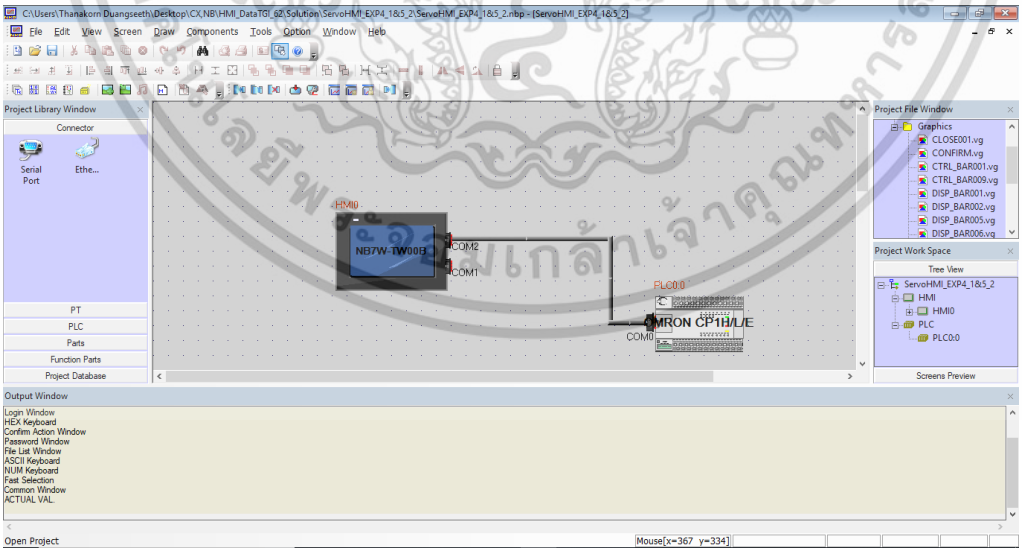
2.3.1 Human Machine Interface (HMI) เป็นอินเทอร์เฟซหรือแดชบอร์ดที่เชื่อมต่อบุคคลกับเครื่องจักร, ระบบหรืออุปกรณ์ และสามารถใช้งานร่วมกับหน้าจอใดก็ได้ ซึ่งจะทำให้ผู้ใช้โต้ตอบกับอุปกรณ์ต่างๆได้ HMI ถูกใช้บ่อยที่สุดในกระบวนการอุตสาหกรรม

2.3.2 Human Machine Interface (HMI): NB7W-TW00B

ในโครงการนี้จะเลือกใช้เป็น HMI NB7W-TW00B ซึ่งเป็นของบริษัท Omron (Specification: 7 inch, TFT LCD, Color, 800 × 480 dots)

รูปที่ 2.4 หน้าจอ HMI ของ Omron

2.4 NB-Designer



รูปที่ 2.5 แสดงโปรแกรม NB-Designer

NB-Designer เป็นโปรแกรมที่ใช้ในการสร้างหน้าจอแสดงผลให้กับหน้าจอ HMI ที่ใช้ในการต่อกับ PLC โดยโปรแกรม NB-designer มีความสามารถในการสร้างปุ่มกดบนจอ touchscreen และกำหนดเป็นสัญญาณ Input ให้แก่ PLC ได้ และยังสามารถกำหนดค่าให้แก่หน่วยความจำใน PLC ได้ และทั้งสามารถแสดงผล Output เป็น bit และค่าตัวเลขได้

2.5 กล้อง Pixy

2.5.1 กล้อง Pixy คืออะไร

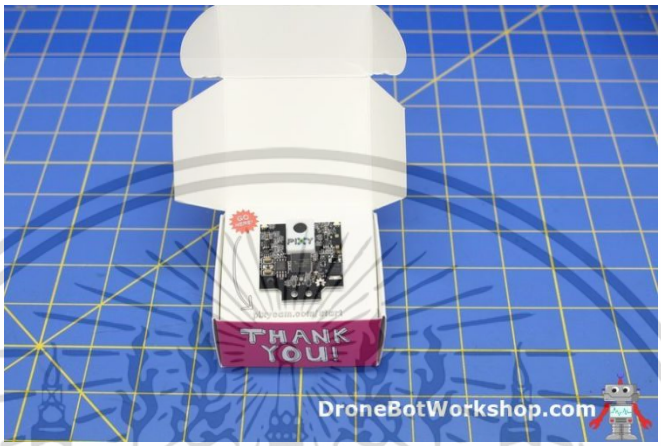
รูปที่ 2.6 กล้อง Pixy 2

Pixy เป็นกล้องที่มีเซนเซอร์ในการตรวจจับเร็วมากโดยส่วนมากจะใช้เกี่ยวกับการสร้างหุ่นยนต์หรือโครงการอื่นๆ ที่คล้ายกันและเราสามารถที่จะสอนให้ Pixy จำลักษณะของวัตถุโดยแค่กดปุ่มที่อยู่บนตัว Pixy โดยเราสามารถใช้ Pixy ในการใช้งานอื่นๆ ได้อีกมากมายไม่เฉพาะที่เกี่ยวกับหุ่นยนต์ โดยมันสามารถตรวจจับและติดตามวัตถุได้ในระดับร้อยชิ้นในเวลาเดียวกันและมันก็สามารถส่งข้อมูลที่ตรวจจับได้ในทันที

2.5.2 กล้อง Pixy 2

Pixy2 เป็นกล้องขนาดเล็กที่ถูกออกแบบมาเพื่อใช้ในการตรวจจับวัตถุ การตรวจจับเส้นทางเดินทาง และ การอ่านบาร์โค้ดอย่างง่าย โดย Pixy2 สามารถตรวจจับความแตกต่างของวัตถุได้ถึง 7 ลักษณะโดยมันจะขึ้นอยู่กับรูปร่างและโทนสีของวัตถุที่เราตรวจจับ โดยแต่ละวัตถุจะมีเอกลักษณ์ของตัวเองซึ่งเราจะเรียกว่า “Signature“ โดยกล้องยังมีอัลกอรึทึมสำหรับติดตามทางเดิน ซึ่ง Pixy2 จะแตกต่างจากกล้องตัวอื่นโดยสามารถที่จะตรวจจับ สิ่งที่อยู่ข้างหน้าเพื่อกำหนดเส้นทางการเดิน (หรือเรียกว่า เวกเตอร์) ซึ่งคล้ายกับการใช้ชีวิตประจำวันของเราเช่นเวลาที่เราเดิน ปั่นจักรยานหรือแม้กระทั่ง

การขับขี่ ซึ่งเราก็ต้องมองไปข้างหน้าเพื่อที่จะตัดสินใจว่าจะเลี้ยวหรือจะหยุดเพื่อที่จะตัดสินใจกับ Pixy2 ก็เช่นกัน

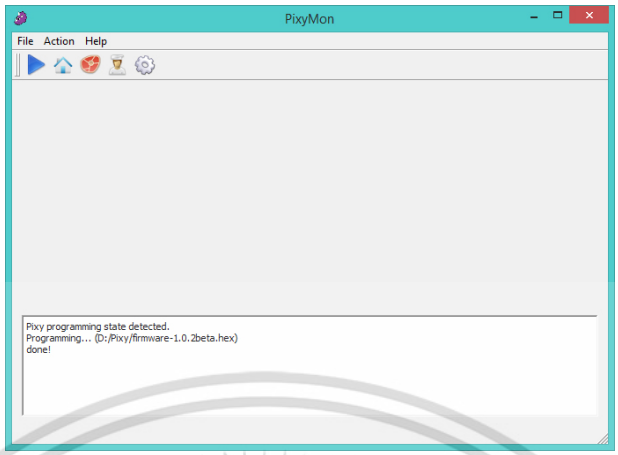


รูปที่ 2.7 กล่องแพคเกจกล้อง Pixy 2

2.6 PixyMon

2.6.1 PixyMon

เป็นซอฟแวร์ที่ใช้งานร่วมกับกล้อง Pixy ที่เราสามารถดาวน์โหลดได้เป็นฟรีซอฟแวร์ เป็นสิ่งจำเป็นอันดับแรกๆ ที่เราต้องมีหลังจากแกะกล่องกล้อง Pixy ออกมาก็ว่าได้โดยการทำงานของมันสามารถที่จะแก้ไขโปรแกรมของเราหรือสั่งการให้กล้องสามารถตรวจจับวัตถุชนิดใดบ้าง ควบคุมโหมดการทำงาน หรือแม้แต่ตั้งค่าพื้นฐานของกล้อง โดยโปรแกรม PixyMon สามารถใช้ได้ใน Windows , Linux และ Mac OS X.



รูปที่ 2.8 แสดงโปรแกรม PixyMon

2.6.2 การใช้งานโปรแกรม PixyMon เบื้องต้น

การใช้งานกล้อง Pixy นั่นเราจำเป็นต้องตั้งค่าการตรวจจับวัตถุ ขนาด และสี ภายในโปรแกรม PixyMon โดยมีขั้นตอนดังต่อไปนี้

2.7 Arduino Uno

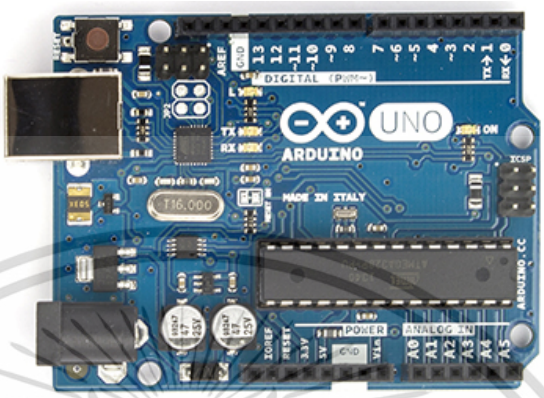
2.7.1 Arduino คืออะไร

Arduino คือ โครงการที่นำชิปไอซีไมโครคอนโทรลเลอร์ตระกูลต่างๆ มาใช้ร่วมกันในภาษา C ซึ่งภาษา C นี้เป็นลักษณะเฉพาะ คือมีการเขียนไลบารี่ของ Arduino ขึ้นมาเพื่อให้การสั่งงานไมโครคอนโทรลเลอร์ที่แตกต่างกัน สามารถใช้งานโค้ดตัวเดียวกันได้ โดยตัวโครงการได้ออกบอร์ดทดลองมาหลายๆรูปแบบ เพื่อใช้งานกับ IDE ของตนเอง สาเหตุหลักที่ทำให้ Arduino เป็นนิยมมาก เป็นเพราะซอฟแวร์ที่ใช้งานร่วมกันสามารถโหลดได้ฟรี และตัวบอร์ดทดลองยังถูกแจกแปลน ทำให้ผู้ผลิตจีนนำไปผลิตและขายออกตลาดมาในราคาที่ถูกมากๆ โดยบอร์ดที่ถูกที่สุดในตอนนี้คือบอร์ด Arduino ที่มีราคาเพียง 120 – 150 บาทเท่านั้น

2.7.2 Arduino Uno

คำว่า Uno เป็นภาษาอิตาลี ซึ่งแปลว่าหนึ่ง เป็นบอร์ด Arduino รุ่นแรกที่ออกมา มีขนาดประมาณ 68.6x53.4 mm เป็นบอร์ดมาตรฐานที่นิยมใช้งานมากที่สุด เนื่องจากเป็นขนาดที่เหมาะสำหรับการเริ่มต้นเรียนรู้ Arduino และมี Shields ให้เลือกใช้งานได้มากกว่าบอร์ด Arduino รุ่นอื่นๆ ที่

ออกแบบมาเฉพาะมากกว่า โดยบอร์ด Arduino Uno ได้มีการพัฒนาเรื่อยมา ตั้งแต่ R2 R3 และรุ่นย่อยที่เปลี่ยนชิปไอซีเป็นแบบ SMD

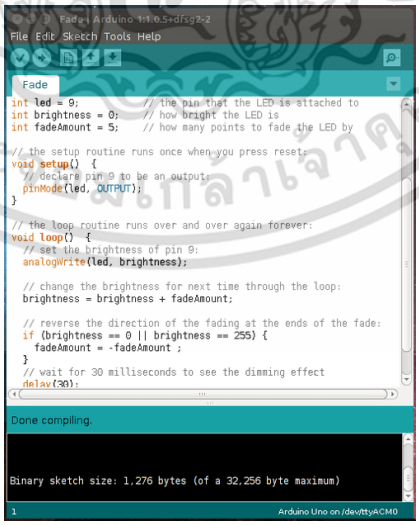


รูปที่ 2.9 Arduino Uno

2.8 Arduino ide

2.8.1 Arduino ide

เครื่องมือการเขียนโปรแกรมที่ใช้งานกับ Arduino ได้ทุกรุ่น โดยภายในจะมีเครื่องมือสำหรับติดต่อ Arduino เช่น การค้นหาArduino ที่ติดต่อกับเครื่องคอมพิวเตอร์ การเลือกรุ่น Arduino ที่ต่ออยู่เพื่อตรวจสอบว่าขนาดของโปรแกรมที่เขียน หรือไรบรารี่ต่างๆซับพอร์ตกับ Arduino รุ่นนั้นๆไหม อีกทั้งยังมีโปรแกรมติดต่อผ่านซีเรียลโดยตรงสำหรับคอมพิวเตอร์



รูปที่ 2.10 แสดงโปรแกรม Arduino ide

2.9 Servo Motor

2.9.1 Servo Motor

เซอร์โวมอเตอร์ (Servo Motor) เป็นมอเตอร์ที่มีการควบคุมการเคลื่อนที่ของมัน (State) ไม่ว่าจะเป็นระยะ ความเร็ว มุมการหมุน โดยใช้การควบคุมแบบป้อนกลับ (Feedback control) เป็นอุปกรณ์ที่สามารถควบคุมเครื่องจักรกล หรือระบบการทำงานนั้นๆ ให้เป็นไปตามความต้องการ เช่น ควบคุมความเร็ว (Speed) ควบคุมแรงบิด (Torque) ควบคุมแรงตำแหน่ง (Position) ระยะทางในการเคลื่อนที่(หมุน) (Position Control) ของตัวมอเตอร์ได้ ซึ่งมอเตอร์ทั่วไปไม่สามารถควบคุมในลักษณะงานเบื้องต้นได้ โดยให้ผลลัพธ์ตามความต้องการที่มีความแม่นยำสูง

รูปที่ 2.11 Servo Motor

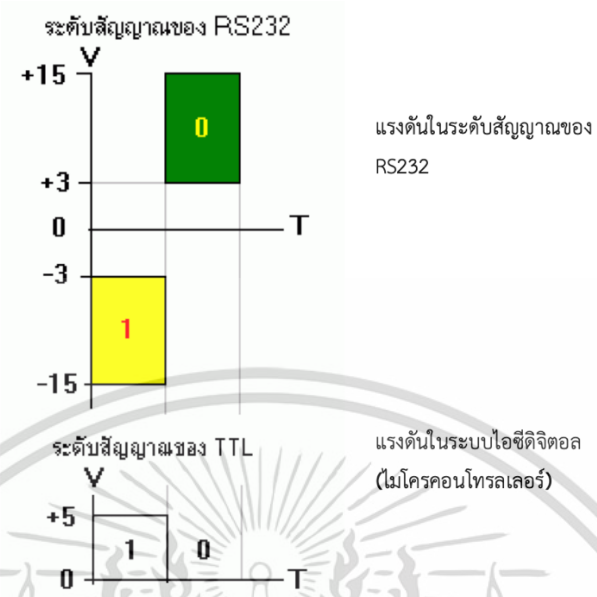
2.9.2 Servo Drive

Servo Drive คือ อุปกรณ์อุตสาหกรรมที่ได้รับสัญญาณคำสั่งมาจาก controller หรือระบบควบคุมสัญญาณ และส่งกระแสไฟฟ้าเพื่อเชื่อมต่อไปยัง Servo Drive เป็นตัวผ่านการสั่งการไปยัง Servo Motor ด้วยการใช้โปรแกรมของตัวงานนั้นๆ ซึ่งจะทำการควบคุมการเคลื่อนที่ ความเร็วของการหมุน ระยะทางที่มอเตอร์หมุน และกำลังที่จะใช้หมุน (กรณีที่เครื่องจักรอุตสาหกรรมต้องใช้กำลังการทำงานสูง)

รูปที่ 2.12 Servo Drive

2.10 การสื่อสารข้อมูลผ่านพอร์ต อนุกรม RS232

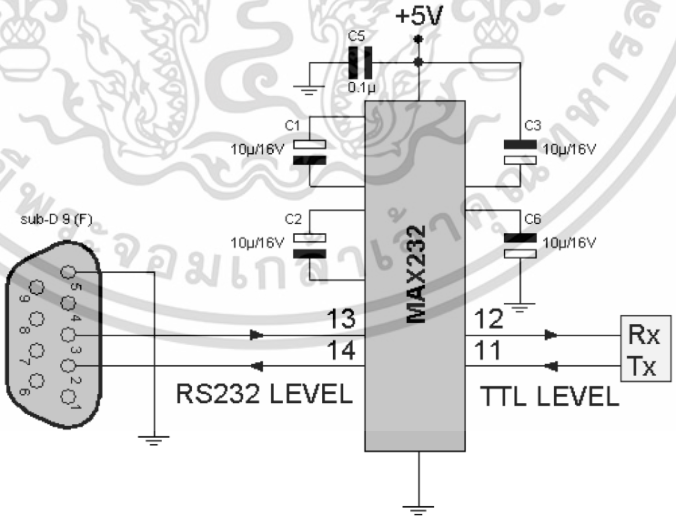
การกำหนดมาตรฐานการเชื่อมต่อแบบอนุกรม EIA RS-232 (x) เป็นมาตรฐานอุตสาหกรรมโดยคณะกรรมการสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association) ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบ อะซิงโครนัส 2 ทิศทาง เพื่อให้มีการใช้งานในการเชื่อมต่อที่สอดคล้องกันระหว่างอุปกรณ์คอมพิวเตอร์ต่างๆ การรับส่งสัญญาณจะมีระดับ สัญญาณตั้งแต่ 3 โวลต์ จนถึง 15 โวลต์สำหรับลอจิก "0" และมีระดับแรงดันที่ -3 โวลต์ จนถึง -15 โวลต์ สำหรับลอจิก "1" ดังนั้นสังเกตได้ว่าจะมีระดับแรงดันที่ใช้ในสถานะลอจิก "0" และ ลอจิก "1" แตกต่างออกไปจากระบบไอซีดิจิตอลทั่วไป การต่อใช้งานกับวงจรไอซีดิจิตอลจึงต้องมีอุปกรณ์ที่ทาหน้าที่ปรับเปลี่ยนระดับแรงดันจาก 3 - 15 โวลต์ ให้มีระดับแรงดัน 0 - 5 โวลต์ ในภาคการส่งข้อมูล ส่วนในภาคของการรับข้อมูลจะต้องเปลี่ยนระดับแรงดัน 0 - 5 โวลต์ จากไมโครคอนโทรลเลอร์ให้เป็นระดับแรงดันที่สูงกว่า +3 หรือต่ำ กว่า –3 โดยจะมีไอซีสำเร็จรูปพร้อมใช้งาน หรืออาจจะต่อวงจรจากทรานซิสเตอร์ก็ได้



รูปที่ 2.13 แรงดันในระดับสัญญาณของ RS-232

ไอซี MAX232

ไอซี MAX232 เป็นไอซีที่แปลงระดับสัญญาณจากระดับ TTL ไปเป็นระดับของ RS-232 และในทำนอง เดียวกันก็รับระดับสัญญาณจาก RS-232 เพื่อแปลงเป็นระดับสัญญาณจากระดับ TTL ให้กับไมโครคอนโทรลเลอร์ได้ในโปรเจคของเราใช้ในการแปลงสัญญาณจาก Arduino to PLC



รูปที่ 2.14 ภาพแสดงตำแหน่งขาของ MAX232 และการเชื่อมต่อ

2.10.1 RS-232

ในโครงงานนี้เราใช้สาย RS-232 ในการรับส่งข้อมูลระหว่าง PLC กับ Arduino และ PLCกับหน้าจอแสดงผล HMI แล้วใช้สาย USB2.0 ต่อจาก PLC เข้าคอมพิวเตอร์ เพื่อตั้งค่าArduino ผ่าน Software Arduino IDE และ เพื่อสร้างหน้าจอแสดงผล HMI ผ่านโปรแกรม NB-Designer

มาตรฐาน RS-232 เป็นมาตรฐานที่รับ/ส่งข้อมูลแบบ Full duplex หรือจะให้พูดง่ายๆ คือ สามารถรับและส่งข้อมูลได้พร้อมกันทั้งคู่ในเวลาเดียวกัน โดยการรับ/ส่งข้อมูลนั้นจะใช้สายไฟทั้งหมด 3 เส้น ได้แก่

- Tx (Transmit data) คือ สายส่งข้อมูล ซึ่งสายเส้นนี้จะมีหน้าที่ในการส่งข้อมูลเท่านั้น
- Rx (Receive data) คือ สายรับข้อมูล ซึ่งสายเส้นนี้จะมีหน้าที่ในการรับข้อมูลเท่านั้น
- GND (Signal ground) คือ สายกราวด์ เป็นสายเทียบหรืออ้างอิงแรงดันไฟฟ้า 0V

รูปที่ 2.15 การรับส่งข้อมูลกันระหว่างคอมพิวเตอร์กับArduinoและHMI

จากรูปที่ 2.13 เป็นตัวอย่างการเชื่อมต่อแบบ RS-232 ของเครื่องมือวัดอุตสาหกรรมกับคอมพิวเตอร์ เพื่อตั้งค่าเครื่องมือวัดผ่าน Software โดย

- Tx (เครื่องวัด) จะถูกต่อเข้าที่ Rx (คอม) เพื่อส่งข้อมูลจากเครื่องวัดไปยังตัวรับของคอมพิวเตอร์
- Rx (เครื่องวัด) จะถูกต่อเข้าที่ Tx (คอม) เพื่อรับข้อมูลที่ถูกส่งมาจากคอมพิวเตอร์
- GND (เครื่องวัด) จะถูกต่อเข้าที่ GND (คอม) เพื่อเทียบสัญญาณแรงดัน 0V

ข้อดีของสัญญาณ RS232

จากที่กล่าวมาข้างต้นการสื่อสารแบบ RS232 ถูกคิดค้นมาตั้งแต่ปี 1960 ซึ่งถือว่านานมาก จากการถือกำเนิดมาอย่างยาวนานนั้นก็ทำให้ข้อดีเหลือน้อยลงไปทุกทีเพราะมีการสื่อสารรูปแบบใหม่ที่ถูกพัฒนาให้ดีกว่าเกิดขึ้นอยู่ทุกวัน แต่ถึงกระนั้น RS232 ก็ยังพอมีข้อดีหลงเหลืออยู่ซึ่งจะขออธิบายเป็นข้อๆดังนี้

RS232 เป็นระบบที่ถูกคิดค้นมาตั้งแต่ปี 1960 และเป็นที่นิยมในยุคแรกซึ่งมีข้อดีคือ มีอุปกรณ์ที่รองรับเยอะ การสื่อสารแบบ RS232 เป็นการสื่อสารที่มีอยู่ในเมนบอร์ดคอมพิวเตอร์แทบทุกรุ่น ซึ่งคนทั่วไปจะรู้จักกันในชื่อ Serial port ซึ่งทำให้การสื่อสารแบบ RS232 ไม่จำเป็นต้องใช้ Converter (ตัวแปลงสัญญาณ) ในการเชื่อมต่อกับคอมพิวเตอร์ ซึ่งต่างจากมาตรฐานใหม่อย่าง RS422, RS485 ที่ถึงแม้จะมีข้อดีที่มากกว่าแต่ก็ต้องใช้ Converter ในการแปลงสัญญาณอยู่ดี แต่ข้อดีข้อนี้อาจอยู่ได้อีกไม่นาน เพราะปัจจุบันเมนบอร์ดรุ่นใหม่ๆได้นำ Serial port ออกจากเมนบอร์ดและเพิ่ม Port การสื่อสารน้องใหม่ที่กำลังเป็นที่นิยมเข้าไปแทนนั้นคือการสื่อสารแบบ USB ซึ่งทำให้การสื่อสารรุ่นเก่าอย่าง RS232 ค่อยๆเลือนหายไปตามกาลเวลา

2.10.2 Host link protocol

ในโปรเจคของเราใช้เครื่อง PLC ของ OMRON รุ่น CP1H ซึ่งในการส่งข้อมูลระหว่างเครื่อง PLC รุ่น CP1H กับเครื่องคอมพิวเตอร์จะใช้การสื่อสารแบบอนุกรมภายใต้มาตรฐาน RS-232C

Command Block

@	Unit Number	Hearder	Text	FCS	*	CR

- @ ใช้ขึ้นต้นคำสั่งหรือ Command
- Unit Number จำนวน 2 หลักเลขฐานสิบ (x*101+x*100)
- เพื่อใช้เลือกเครื่องควบคุมที่ XX ในระบบเครือข่าย Host Link
- Header ต้องการอ่าน หรือ เขียนกับหน่วยความจำส่วนใดของ PLC
- Text เป็นส่วนของข้อมูลที่ได้จากการอ่าน หรือ เขียน
- FCS (Frame Check Sequence) เป็นส่วนควบคุมความผิดพลาดของข้อมูลซึ่งได้จากการคำนวณ
- * (Terminate) ใช้ในการปิดท้ายคำสั่ง
- CR (Caries Return) เพื่อให้ข้อความที่ตามมาขึ้นบรรทัดใหม่

การคำนวณหา FCS

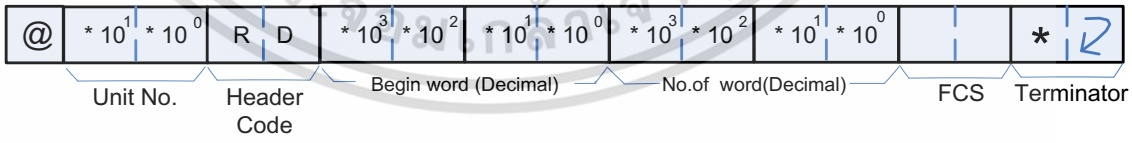
อักขระ	รหัส (ASCII BINARY)	(HEX)
@	0100 0000	[40]
0	0011 0000	[30]
0	0011 0000	[30]
R	0101 0010	[52]

R	0101	0010	[52]
0	0011	0000	[30]
0	0011	0000	[30]
3	0011	0011	[33]
0	0011	0000	[30]
0	0011	0000	[30]
0	0011	0000	[30]
0	0011	0000	[30]
2	0011	0010	[32]
FCS	0100	0001	[41]

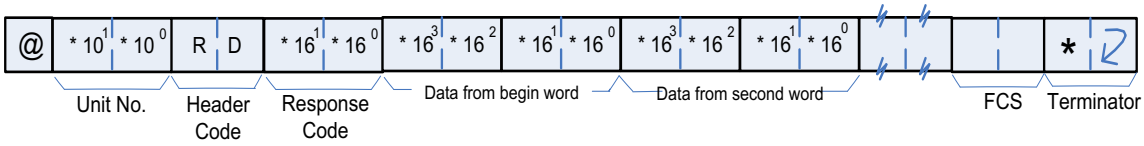
ตารางที่ 2.2 ตารางการคำนวณหาค่า FCS

DM Read Command & Response Command

DM READ
Command Format

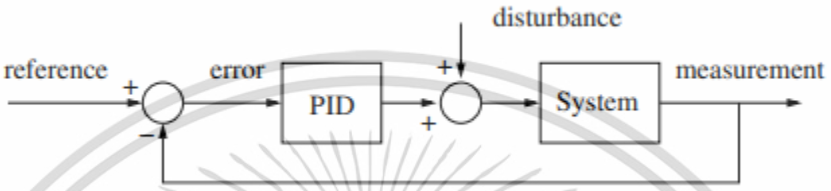


DM
Response Command



2.11 PID Controller

เป็นกระบวนการควบคุมอย่างหนึ่งที่นิยมนำมาใช้ในอุตสาหกรรม ในระบบควบคุมมีตัวควบคุมหลายชนิด ตัวควบคุมส่วนใหญ่ที่ใช้ในการควบคุมกระบวนการเป็นแบบ PID โดยต่ออนุกรมกับระบบที่ต้องการควบคุม ดังแสดงในรูปที่ 2.14 สัญญาณออกจากตัวควบคุม PID สามารถบรรยายได้ดังนี้



รูปที่ 2.16 ตัวควบคุม PID ที่ต่อเข้าในระบบแบบอนุกรม

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right)$$

โดย *u(t)* คือสัญญาณควบคุม *e(t)* คือค่าความคลาดเคลื่อนของสัญญาณออกจากค่ากำหนด ตัวควบคุม PID ประกอบไปด้วยเทคนิคการควบคุมพื้นฐาน 3 แบบ 1. แบบสัดส่วน (Proportional หรือ P) 2. แบบอินทิกรัล (Integral หรือ I) และ3.แบบอนุพันธ์(Derivative หรือ D) แต่ละแบบสามารถนำมาประกอบกันเพื่อให้ได้ตัวควบคุมที่ต้องการตัวควบคุมมีพารามิเตอร์ 3 ตัว คือ ค่าอัตราขยายแบบสัดส่วน (K_p) ค่า integral time (T_i) และderivative time (T_d) ซึ่งรายละเอียดของแต่ละแบบมีดังนี้

2.11.1 Proportional Action

การควบคุมแบบสัดส่วนเป็นเทคนิคที่ง่ายที่สุด หลักการคือสัญญาณควบคุม u(t) จากตัวควบคุมที่ส่งไปปรับกระบวนการมีค่าเป็นสัดส่วนกับความคลาดเคลื่อน ซึ่งสามารถเขียนได้ในรูป

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

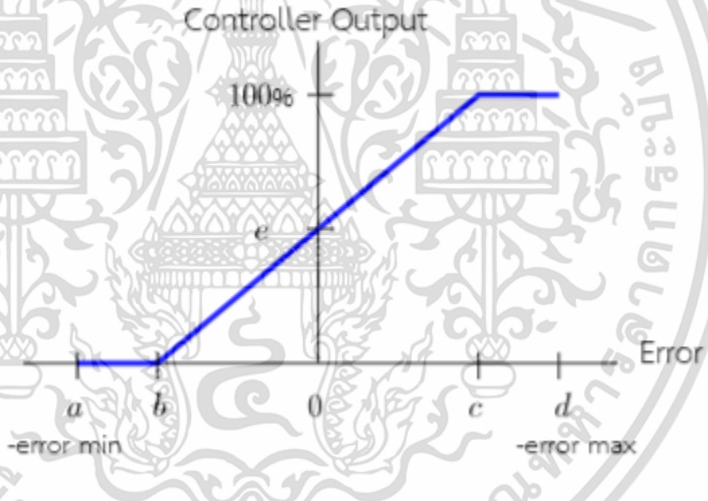
โดยที่ K_p คือค่าอัตราขยายและ

e(t) = ความคลาดเคลื่อน = ค่ากำหนด – ค่าวัด

ตัวควบคุมบางตัวสัญญาณเข้าและสัญญาณออกอาจมีหน่วยต่างกัน เช่นการเปลี่ยนแปลงของอุณหภูมิที่ทำให้เกิดการเปลี่ยนแปลงความดัน เพื่อหลีกเลี่ยงการแปลงหน่วย ความสัมพันธ์ระหว่างสัญญาณออกและสัญญาณเข้าของตัวควบคุมอาจแสดงเป็นแถบสัดส่วน (Proportional Band หรือ %PB) โดยที่แถบสัดส่วนคือพิสัยของสัญญาณเข้าที่ทำให้ตัวควบคุมปฏิบัติงานเต็มพิสัยการทำงาน หรือถ้ามองจากตัวควบคุม แถบสัดส่วนคือช่วงความคลาดเคลื่อนที่ทำให้สัญญาณออกของตัวควบคุมเปลี่ยนแปลงจากค่าสูงสุดไปต่ำสุด โดยแสดงเป็นเปอร์เซ็นต์ของพิสัยสัญญาณเข้าตัวควบคุมความสัมพันธ์ระหว่างอัตราขยายและเปอร์เซ็นต์ แถบสัดส่วนคือ

$$K_P = \frac{100}{\%\,PB}$$

ลักษณะสมบัติของการควบคุมแบบสัดส่วนแสดงไว้ในรูปที่ 2.15

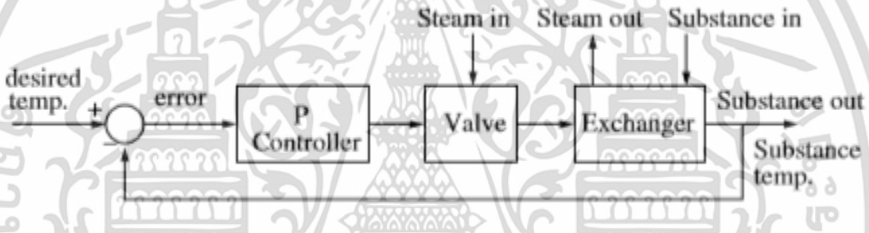


รูปที่ 2.17 ลักษณะสมบัติของตัวควบคุมแบบสัดส่วน

ค่าเปอร์เซ็นต์แถบสัดส่วน (%PB) คือระยะ bc แม้ความคลาดเคลื่อนเป็นศูนย์ ยังมีสัญญาณค่าหนึ่ง ออก จากตัวควบคุมที่ป้อนให้กับกระบวนการค่านี้ทำหน้าที่เป็นไบแอส (bias) ทำให้ระบบทำงานที่จุดทำงานต่อไปได้โดยทั่วไป สัญญาณค่านี้มักจะถูกตั้งให้เท่ากับ 50% ของสัญญาณขาออกสูงสุดของตัวควบคุมนั่นคือ

$$\text{สัญญาณออก} = \frac{\%\ \text{ความคลาดเคลื่อน}}{\%\ \text{แถบสัดส่วน}} + 50\%$$

นอกจากนี้ตัวควบคุมมีย่านทำงานที่เป็นเชิงเส้นช่วงหนึ่ง โดยทำหน้าที่เป็นตัวขยาย (amplifier) แต่ถ้าความคลาดเคลื่อนมีมากเกินระดับหนึ่ง ตัวขยายจะอิ่มตัวทำให้สัญญาณออกมีค่าคงที่ การควบคุมแบบสัดส่วนนี้สามารถควบคุมระบบได้ดีพอสมควร เหมาะสมกับกระบวนการที่ต้องการผลตอบสนองรวดเร็วและยอมให้เกิดความคลาดเคลื่อนขนาดคงที่ขนาดหนึ่ง อย่างไรก็ตาม หากในกระบวนการเกิดมีการเปลี่ยนแปลงพารามิเตอร์ อาจทำให้เกิดปัญหา เช่น มีค่าความคลาดเคลื่อนในสภาวะอยู่ตัว (steady-state error) หรือที่เรียกว่า ออฟเซต (offset) ตัวควบคุมแบบสัดส่วนไม่สามารถแก้ไขให้หมดได้ตัวอย่างในกรณีนี้ก็ได้แก่ระบบในรูปที่ 2.18



รูปที่ 2.18 ตัวอย่างระบบควบคุมเครื่องแลกเปลี่ยนความร้อนแบบใช้ไอน้ำ

จากรูปที่ 2.18 ระบบเป็นเครื่องแลกเปลี่ยนความร้อนแบบใช้ไอน้ำ โดยมีตัวควบคุมแบบสัดส่วนทำหน้าที่ปรับวาล์วควบคุมปริมาณไอน้ำที่ผ่านไปให้ความร้อนกับสารที่ต้องการทำความร้อน ในสภาวะอยู่ตัว ถ้าประสิทธิภาพการแลกเปลี่ยนความร้อน ของเครื่องแลกเปลี่ยนความร้อนต่ำลง (เนื่องจากไอน้ำหรือสารที่ต้องการทำความร้อนทำให้เกิดสนิมหรือตะกรันบนผิวที่ใช้แลกเปลี่ยนความร้อนหรือเนื่องจากสาเหตุอื่น) ผลที่ตามมาคืออุณหภูมิของสารที่ได้จะไม่ตรงกับค่าที่ตั้งไว้ดังนั้นเพื่อให้ระบบเข้าสู่สภาวะอยู่ตัวอีกครั้ง สัญญาณออกของตัวควบคุมจะต้องมีค่าเพิ่มขึ้นจากเดิม เมื่อพิจารณาจากสมการของสัญญาณควบคุมข้างต้น สัญญาณออกจะมีค่าเพิ่มขึ้นได้ความผิดพลาดจะต้องมีค่าเพิ่มขึ้นด้วย จึงทำให้เกิดความแตกต่างของระหว่างอุณหภูมิ สารกับค่าที่ตั้งไว้ที่สภาวะสมดุลใหม่

แนวทางการแก้ไขปัญหาที่เกิดขึ้นทำได้ 2 วิธีคือ วิธีแรก คือ เพิ่มอัตราขยาย (gain) ของตัวควบคุมเพื่อเพิ่มผลของความคลาดเคลื่อนที่มีต่อระบบ ถึงแม้ความคลาดเคลื่อนที่เกิดขึ้นจะมีค่าน้อยลงแต่ก็จะทำให้สัญญาณออกจากตัวควบคุมที่เหมาะสมกับกระบวนการขณะนั้นได้ อย่างไรก็ตามการเพิ่มผลของ

ความคลาดเคลื่อนมากเกินไปก็อาจทำให้ระบบแกว่งได้เนื่องจากระบบมีความไว วิธีที่สอง คือปรับค่าไบแอสของตัวควบคุมใหม่ด้วยมือ ซึ่งทำให้ตัวควบคุมเลื่อนจุดทำงานไปยังจุดที่ให้สัญญาณออกที่เหมาะสมกับกระบวนการในขณะนั้นได้ ปัญหาของวิธีหลังอยู่ตรงที่ต้องปรับค่าไบแอสของตัวควบคุมทุกครั้งที่มีการเปลี่ยนแปลง พารามิเตอร์ของกระบวนการ

2.11.2 Integral Action

ผลตอบของการควบคุมแบบสัดส่วนรวมกับการควบคุมแบบอินทิกรัล สามารถอธิบายได้ในสมการ

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau\right)$$

เมื่อ K_p คืออัตราขยาย *e(t)* คือ ความคลาดเคลื่อน และ T_i คือ integral time (วินาที)

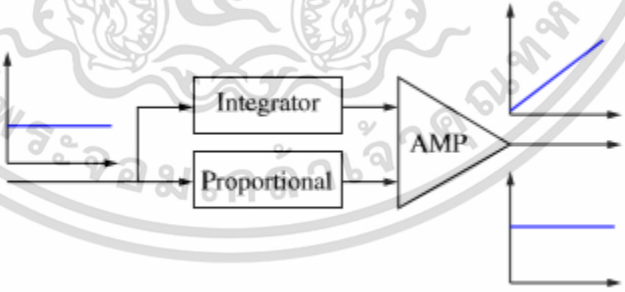
เมื่อเปรียบเทียบกับสมการของตัวควบคุมแบบสัดส่วน ความแตกต่างอยู่ตรงที่เทอมไบแอสนั่นคือตัวควบคุมแบบสัดส่วนถูกจำกัดด้วยส่วนไบแอสเป็นค่าคงที่ ส่วนการควบคุมแบบอินทิกรัลมีการสะสมความคลาดเคลื่อนในการปรับแต่งไบแอส (นั่นคือ ทำหน้าที่เป็นตัวอินทิกรัล) และจะหยุดสะสมเมื่อความคลาดเคลื่อนของระบบเป็นศูนย์ เมื่อผลตอบเข้าที่สมบูรณ์แล้วเทอมไบแอสของระบบจะมีค่ามากน้อยเพียงใดขึ้นอยู่กับลักษณะของการรบกวน(disturbance) การทำงานในลักษณะเช่นนี้ มีลักษณะคล้ายกับฟังก์ชันรีเซตด้วยมือ (manual-reset function) ดังนั้นในบางครั้งจึงเรียกตัวอินทิกรัลว่าฟังก์ชันรีเซต (reset function)

คุณสมบัติของตัวอินทิกรัลในการกำจัดความคลาดเคลื่อน (หรือออฟเซต) เป็นข้อดีอย่างมากจึงเป็นที่นิยมใช้กับระบบ ควบคุมป้อนกลับ อย่างไรก็ตาม ตัวอินทิกรัลก็มีข้อเสีย นั่นคือทำให้เกิดการล้าหลัง (capacity-like lag) และทำให้ช่วงเวลา ของการแกว่งยาวนานขึ้น โดยทั่วไประบบแบบสัดส่วนรวมกับอินทิกรัลจะมีช่วงเวลาของการแกว่งนานกว่าระบบเชิงสัดส่วนอย่างเดียว 50% หรือ T_{pi} = 1.5 T_p สำหรับระบบที่มีค่าคงตัวเวลา (time constant) น้อย (เช่น ระบบควบคุมอัตราการ ไหล) ปัญหานี้จะไม่มีผลมากนัก แต่สำหรับระบบที่มีค่าคงตัวตัวเวลามาก (เช่น ระบบควบคุมระดับ) ปัญหานี้อาจมีผลมากจนทำให้ระบบเข้าสู่จุดวิกฤติที่ไม่สามารถยอมรับได้

การควบคุมแบบอินทิกรัลมีลักษณะเช่นเดียวกับการควบคุมสัดส่วนตรงผลกระทบของการเพิ่มอัตราขยายของตัวควบคุมหากอัตราขยายมีค่ามากเกินไปจะทำให้ผลตอบของระบบมีการแกว่ง โดยทั่วไป Integral time (T_i = 1/Ki secโดยที่ Ki =repeats/sec) เป็นตัวแสดงว่า อัตราการตอบสนองของกระบวนการต่อสัญญาณการควบคุม ค่า T_i ที่น้อยกว่า จะทำให้ตัวควบคุมมีการตอบสนองที่เร็วกว่าในระยะเริ่มต้นโดยที่ความคลาดเคลื่อนยังเป็นค่าบวกอยู่ ดังนั้นกว่าความคลาดเคลื่อนจะเป็นศูนย์ (ซึ่งทำให้เทอม $\int_0^t e(t)dt$ หยุดทำงาน) เทอมไบแอสก็จะมีค่าสูงกว่าที่ต้องการดังนั้นผลตอบสนองจึงเกิดส่วนพุ่งเกิน (overshoot) สูงกว่าค่ากำหนด เป็นผลให้ตัวอินทิกรัลทำหน้าที่ปรับให้ความคลาดเคลื่อนมีค่าลดลง การใช้ตัวอินทิกรัลในการควบคุมควรระวังในเรื่องของความคลาดเคลื่อนขนาดใหญ่ เช่น เกิดการเปลี่ยนแปลงค่ากำหนดขนาดใหญ่) เพราะจะทำให้เกิดปัญหา integral windup ถึงแม้ว่า T_i มีค่าถูกต้องในสภาวะการทำงานธรรมดา แต่สัญญาณควบคุมอาจถึงจุดอิ่มตัวขณะผลตอบเกิดส่วนพุ่งเกิน

ข้อสรุปของตัวควบคุมอินทิกรัล

- ทำหน้าที่คล้ายรีเซตด้วยมือ (manual reset) เพื่อกำจัดความคลาดเคลื่อน
- มีปัญหาการล้าหลัง ยังทำให้ผลเกิดการหักล้างทางเวลาในตัวควบคุมจึงไม่เหมาะกับระบบที่มีค่าคงตัวเวลายาวนาน
- ทำให้ช่วงเวลาในการแกว่งยาวนานขึ้น



รูปที่ 2.19 แผนภาพกรอบแสดงลักษณะตัวควบคุมแบบ PI

ในระบบควบคุม ค่าที่วัดได้และค่ากำหนดควรเป็นค่าเดียวกันหรือกล่าวอีกนัยหนึ่ง ค่าความคลาดเคลื่อนในสภาวะอยู่ตัวควรเป็นศูนย์ ถ้ามีความคลาดเคลื่อนในสภาวะอยู่ตัว สัญญาณที่ออกจากอินทิเกรเตอร์ (เพิ่มขึ้นด้วยอัตราคงที่ เมื่อสัญญาณเข้ามีค่าคงที่) ส่งต่อให้กับวงจรขยาย

ดังแสดงในรูปที่ 2.19 สังเกตว่า ความคลาดเคลื่อนเป็นสัญญาณเข้าของตัวควบคุมทั้งสัดส่วน และอินทิกรัลโดยสัญญาณออกจะมารวมกันที่วงจรขยายและส่งสัญญาณไปควบคุมระบบตัวควบคุมจะทำให้ค่าที่วัดได้ เพิ่มขึ้นจนเท่ากับค่ากำหนด นั่นคือทำให้ความคลาดเคลื่อนในสภาวะอยู่ตัวเป็นศูนย์ อย่างไรก็ตามหาก T_i มีค่าน้อย ผลตอบอาจเกิดการแกว่งได้

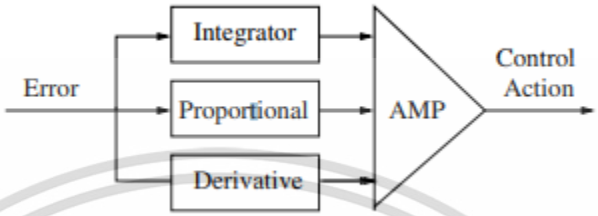
2.11.3 Derivative Action

ตัวควบคุมแบบสัดส่วนและแบบปริพันธ์ต่างก็มีข้อจำกัดอยู่ที่ความคลาดเคลื่อนขนาดใหญ่ซึ่งเป็นปัญหาต่อการควบคุมกระบวนการแต่ความคลาดเคลื่อนขนาดใหญ่นี้สามารถรู้ได้ล่วงหน้าโดยพิจารณาจากแนวโน้มของความคลาดเคลื่อนหรืออัตราการเปลี่ยนแปลงของสัญญาณนั่นเอง ตัวอนุพันธ์มีหลักการทำงาน คือ ตัวควบคุมตอบสนองต่ออัตราการเปลี่ยนแปลงของความคลาดเคลื่อน ถึงแม้ว่าความคลาดเคลื่อนมียังค่าเล็กอยู่ สัญญาณออกของตัวอนุพันธ์ไม่ได้สัมพันธ์กับขนาดของความคลาดเคลื่อนแต่ขึ้นอยู่กับการเปลี่ยนแปลงของความคลาดเคลื่อนถ้าความคลาดเคลื่อนมีค่าคงที่ตัวอนุพันธ์จะให้สัญญาณออกเป็นศูนย์คุณลักษณะข้อนี้มีผลดีคือ ตัวควบคุมจะมีผลตอบสนองที่เกิดก่อนที่ความคลาดเคลื่อนจะเพิ่มมากขึ้น และทำให้ระบบมีผลตอบสนองที่เร็วขึ้นตัวควบคุมแบบอนุพันธ์สามารถเขียนได้ดังนี้

$$u(t) = K_p\left(e(t) + T_d\frac{de(t)}{dt}\right)$$

โดย derivative time (T_d) เป็นเวลาที่แสดงถึงผลตอบสนองเนื่องจากตัวอนุพันธ์ การเพิ่ม T_d จะทำให้ผลตอบสนองของตัวอนุพันธ์มีค่ามากขึ้น เนื่องจากตัวอนุพันธ์มีความไวต่อการเปลี่ยนแปลงมาก ดังนั้นจึงนิยมใช้กับค่าที่วัดได้เท่านั้น แต่ไม่ใช้กับค่ากำหนด เพราะการเปลี่ยนค่ากำหนดมักจะเป็นแบบขั้น (step) ทำให้ผลตอบสนองของตัวอนุพันธ์เป็นพัลส์ และทำให้เกิดการกระแทก (bump) ของอุปกรณ์ในกระบวนการสำหรับค่ากำหนดใช้เฉพาะกับตัวควบคุมสัดส่วนและอินทิกรัล ตัวอนุพันธ์คือตัวควบคุมที่ก่อให้เกิดผลตรงข้ามกับตัวอินทิกรัล ดังนั้นจึงใช้ในการปรับปรุงกระบวนการที่มีการล้าหลังทางเวลา (timelag) มากๆ ทำให้ผลตอบสนองรวดเร็วขึ้น และช่วงเวลาการแกว่งที่สั้นลง ข้อเสียของตัวอนุพันธ์ คือ มีความไว ต่อสัญญาณรบกวนเป็นอย่างมาก เพราะมีผลตอบสนองโดยตรงต่ออัตราการเปลี่ยนแปลงของสัญญาณที่วัดได้ ดังนั้นแม้สัญญาณรบกวนจะมีขนาดเล็กแต่ก็อาจก่อให้เกิดการเปลี่ยนแปลงต่อสัญญาณออกของตัวควบคุม

จึงเป็นไปไม่ได้ที่จะใช้ตัวอนุพันธ์ในการควบคุมผลของสัญญาณรบกวน ยิ่งไปกว่านั้นระบบใดที่มีสัญญาณรบกวนมาก จะไม่สามารถใช้ตัวอนุพันธ์ ในวงการอุตสาหกรรมส่วนใหญ่นิยมใช้เพียงตัวควบคุม PI เท่านั้น



รูปที่ 2.20 แผนภาพกรอบแสดงลักษณะตัวควบคุมแบบ PID

บทสรุปของการตัวอนุพันธ์

• เหมาะสำหรับกระบวนการที่ล้าหลังทางเวลามาก ทำให้การควบคุมถึงจุดที่ต้องการเร็วขึ้น

• ถ้า T_d มากเกินไป ผลของตัวอนุพันธ์จะทำให้ผลตอบสนองไวขึ้น จนกระทั่งระบบอาจขาดเสถียรภาพได้

• ไม่เหมาะกับระบบที่มีตัวแปรกระบวนการเปลี่ยนแปลงได้ง่าย หรือมีการล้าหลังทางเวลาน้อย เพราะจะทำให้ระบบขาดเสถียรภาพ (เช่นระบบควบคุมอัตราการไหล)

• ไม่ควรใช้กับระบบที่มีสัญญาณรบกวนมาก

• ใช้ชดเชยการล้าหลังที่เกิดจากตัวปริพันธ์ด้วยการนำหน้า (lead) ในตัวอนุพันธ์

จะได้ผลรวมเทอมสัดส่วน ปริพันธ์ และอนุพันธ์ จะนำมารวมกันเป็นสัญญาณขาออกของการควบคุมแบบ PID กำหนดให้ เป็นสัญญาณขาออก สมการสุดท้ายของวิธี PID คือ

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

เมื่อมีการเปลี่ยนแปลงค่าที่กำหนดทันที ความคลาดเคลื่อนจะมีค่าเปลี่ยนแปลงอย่างทันที และส่งผลต่อผลตอบสนองของระบบ ถ้านำอนุพันธ์ของความคลาดเคลื่อน นั่นคือ อัตราการเปลี่ยนแปลงของความคลาดเคลื่อน แล้วไปรวมกับสัญญาณที่ได้จากตัวควบคุมแบบสัดส่วนและปริพันธ์

ดังแสดงในรูป 5 จะทำให้การทำงานของระบบดีขึ้น การควบคุมเชิงอนุพันธ์ไม่มีผลต่อความคลาดเคลื่อนในสภาวะอยู่ตัว แต่จะลดช่วงเวลาเข้าที่ (settling time) โดยลดการแกว่งลง

จากรายละเอียดที่กล่าวมาจะพบว่าตัวควบคุม PID ยังคงมีจุดอ่อนบ้าง ดังนั้นในการใช้งานจริงจึงมีการต่อวงจรเพิ่มเติม สำหรับแก้จุดอ่อน ได้แก่

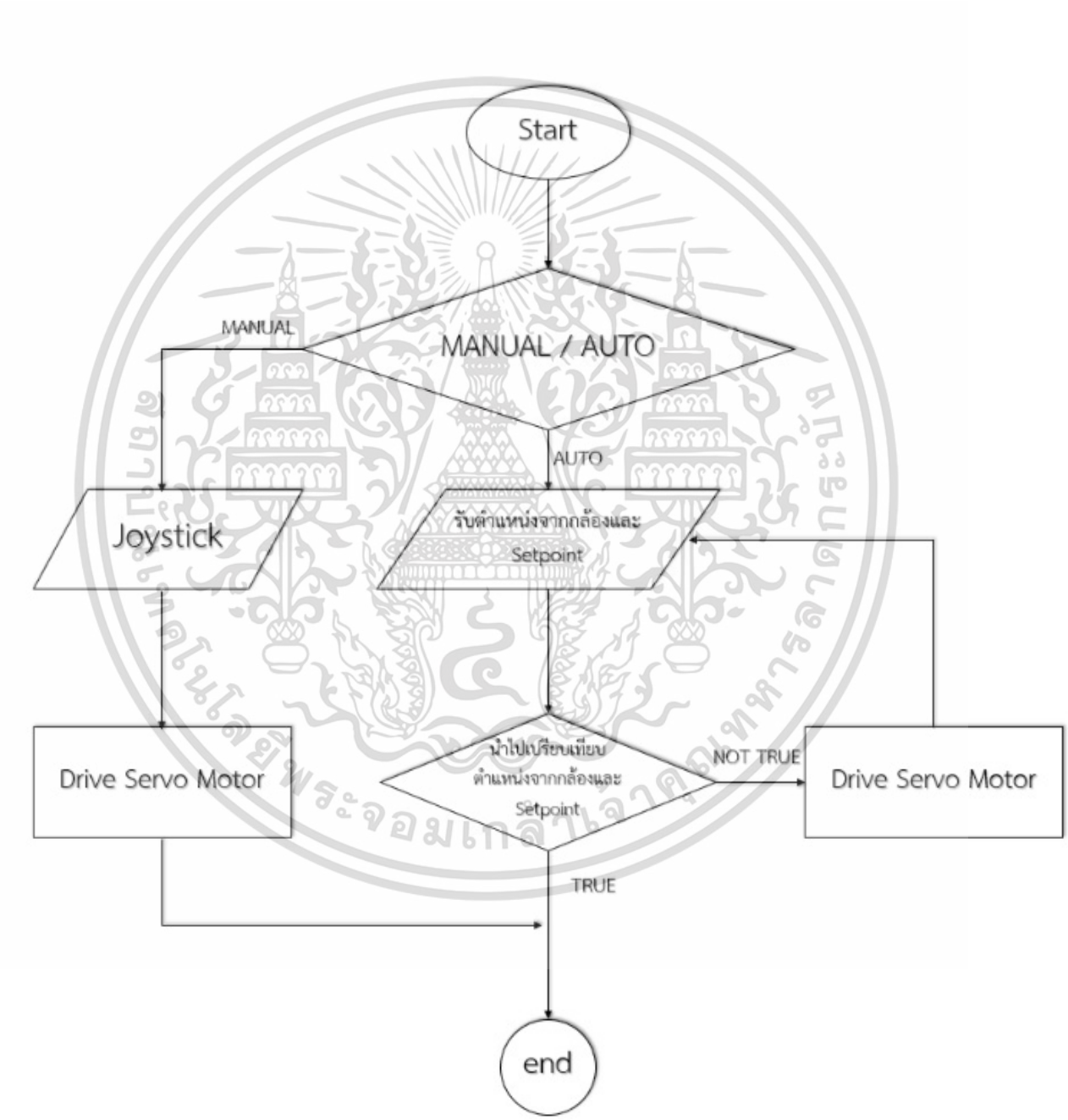
- วงจรสำหรับป้องกัน integral windup ที่เกิดจากตัวควบคุมแบบอินทิกรัล
- วงจรกรอง (filter) สำหรับลดผลเนื่องจากสัญญาณรบกวนที่มีกับตัวควบคุมแบบอนุพันธ์
- ปรับโครงสร้างให้ตัวควบคุมเชิงอนุพันธ์รับสัญญาณออกของระบบเท่านั้น เพื่อป้องกันการเปลี่ยนแปลงที่มีค่าเกินกว่าที่รับได้ (derivative overrun)

นอกจากปัญหาที่เกิดจากการควบคุมทั้ง 3 แบบแล้ว ยังมีปัญหาที่เกิดจากฟังก์ชันการทำงาน คือ ตัวควบคุมส่วนมากจะมีโหมดการทำงาน 2 โหมด คือ การควบคุมด้วยมือ (manual) และการควบคุมอัตโนมัติ (automatic) ในโหมดการควบคุมด้วยมือ สัญญาณที่ส่งออกจากตัวควบคุมจะขึ้นกับการปรับโดยตรงของผู้ใช้หากมีการเปลี่ยนโหมดการทำงานกลับมาที่โหมดการควบคุมอัตโนมัติ ตัวควบคุมทำหน้าที่ส่งสัญญาณออกจากตัวควบคุมอาจเกิดปัญหาการกระแทก (bump) ขึ้นได้ เนื่องจากการเปลี่ยนแปลงสัญญาณควบคุมที่ออกจากตัวควบคุมอย่างเฉียบพลัน ดังนั้นในตัวควบคุม PID ส่วนมากจึงต้องมีวงจรลดการกระแทก (bumpless transfer) สำหรับแก้ปัญหานี้ไว้ด้วย

บทที่ 3

วิธีการดำเนินงาน

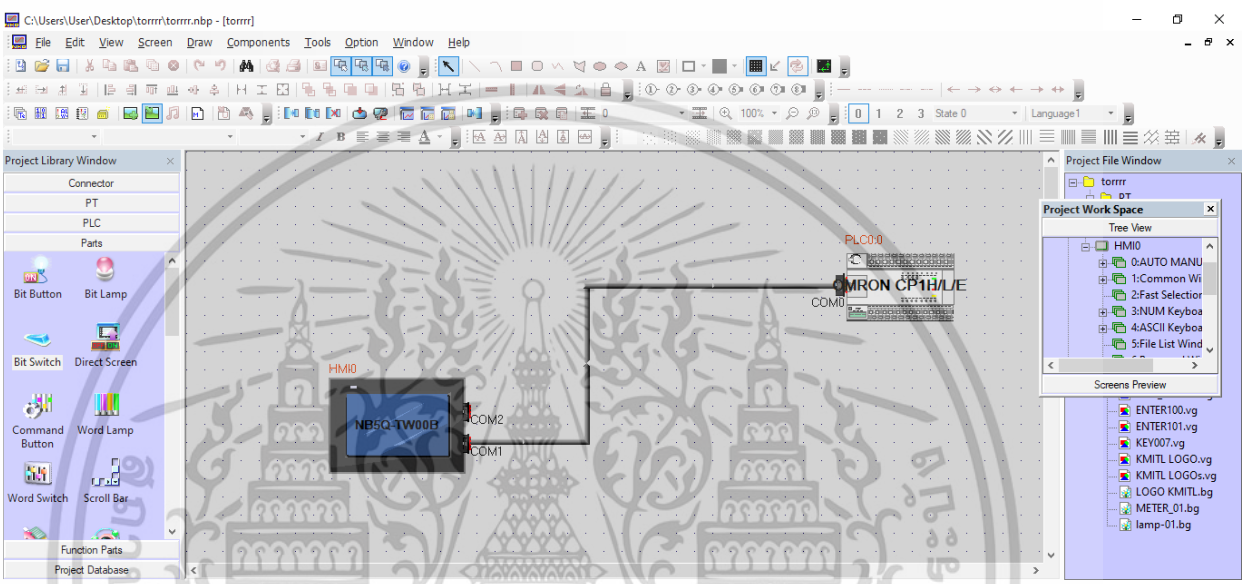
3.1 แผนผังการดำเนินงาน



รูปที่ 3.1 Flowchart แสดงหลักการทำงาน

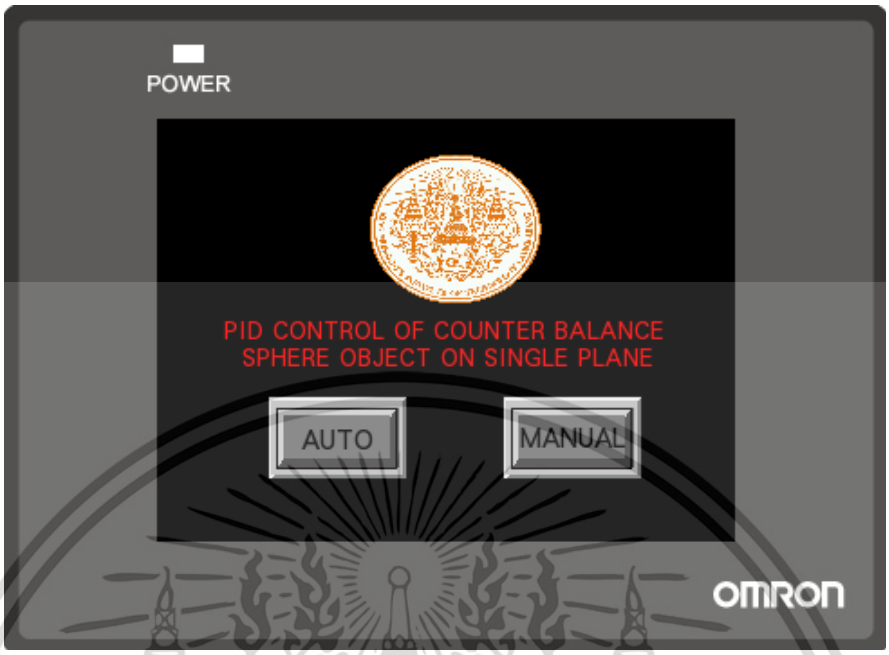
3.2 การดำเนินงานในส่วน HMI

Software ที่ใช้ในการทำจอแสดงผล HMI คือโปรแกรม NB-Designer โดยเราใช้ HMI เพื่อกำหนดค่า Set point และตั้งค่า PID Control ในโหมด AUTO และยังสามารถควบคุมโดยใช้ Joystick ในโหมด MANUAL



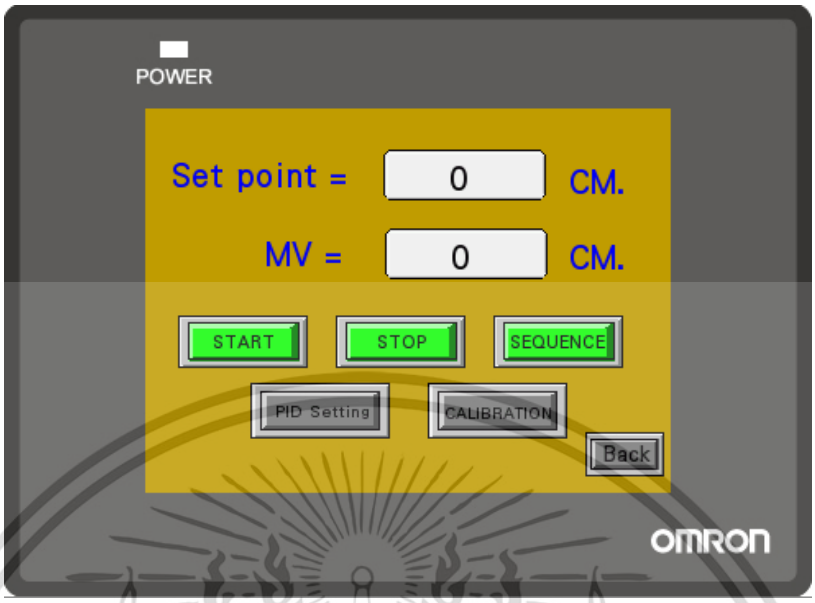
รูปที่ 3.2 การเชื่อมต่อ HMI และ PLC

ขั้นแรกทำการเลือกรุ่นของ PLC และ HMI ให้ตรงกับที่เราใช้งาน ทางคณะผู้จัดทำใช้เป็น PLC CP1H/L/E และ HMI NB5Q-TW00B เชื่อมต่อกันด้วย Serial Port ดังแสดงในรูปที่ 3.2



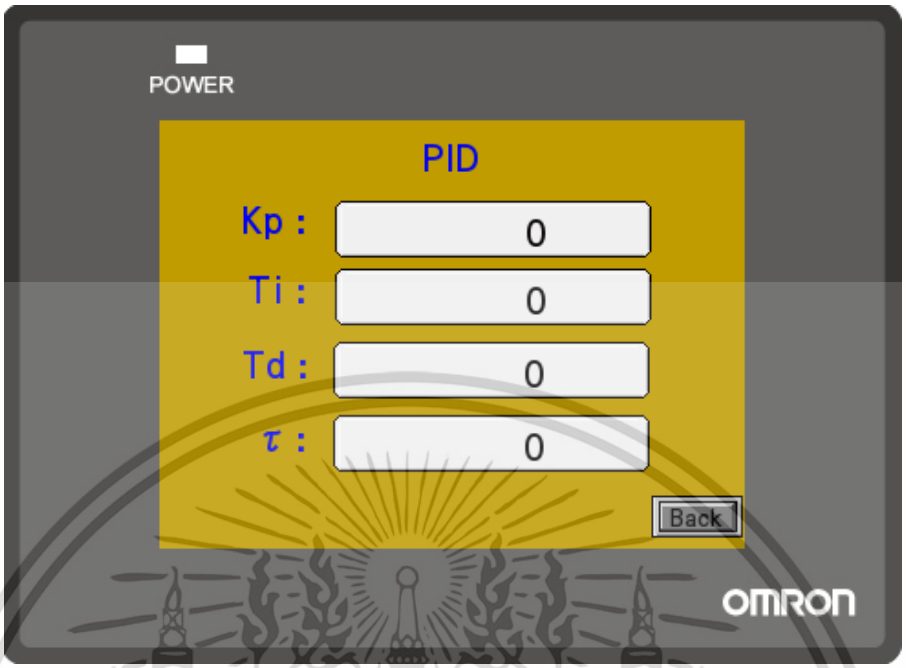
รูปที่ 3.3 หน้าจอแสดงผล หน้าเลือก AUTO/MANUAL

หน้าแรกของจอแสดงผลจะมีให้เลือกโหมดในการควบคุม 2 โหมดคือ AUTO และ MANUAL โดยเมื่อกดปุ่ม AUTO ระบบจะทำงานในโหมด AUTO และหน้าจอแสดงผลจะเปลี่ยนไปยังรูปที่ 3.4 (Set Point) เมื่อกดปุ่ม MAUAL ระบบจะทำงานในโหมด MANUAL และหน้าจอแสดงผลจะเปลี่ยนไปยังรูปที่ 3.6 (Manual Setup)



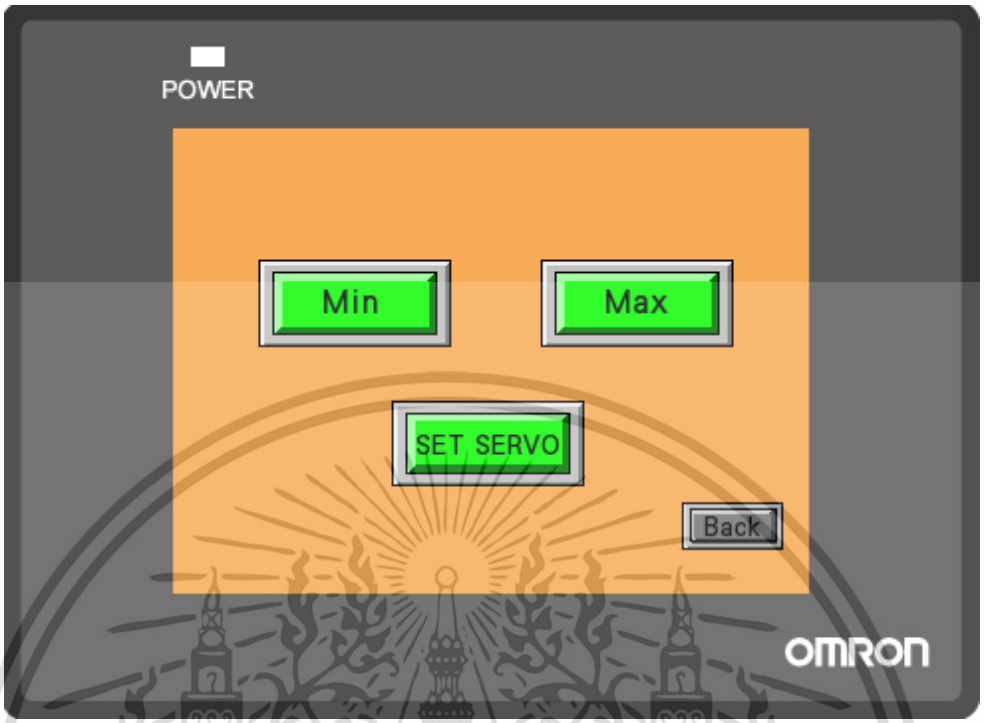
รูปที่ 3.4 หน้าจอแสดงผล หน้าเลือก Setpoint

หลังจากกดปุ่ม AUTO ในรูปที่ 3.3 จะสามารถตั้งค่า Set point 0 ถึง 40 เซนติเมตร และสามารถอ่านค่า Manipulate Value ได้ ในหน้าที่ 3.4 จะมีปุ่ม START (เมื่อกดแล้วระบบจะเริ่มทำงานโดยการ load ลูกบอลขึ้นมาบนคาน), STOP (เมื่อกดแล้วจะเป็นการ unload ลูกบอลกลับไปยัง basket), SEQUENCE (เมื่อกดแล้ว basket จะ load ลูกบอลขึ้นมาบนคาน จากนั้นลูกบอลจะวิ่งเข้าสู้ set point 3 จุดที่ถูกตั้งไว้ คือ 10 20 และ 30 ตามลำดับ โดยจะเปลี่ยนไปสู้ set point ต่อไปได้ก็ต่อเมื่อลูกบอลสามารถเข้าสู่ set point นั้นได้แล้ว) และเมื่อกดปุ่ม PID Setting หน้าจอแสดงผลจะเปลี่ยนไปยังรูปที่ 3.5 เมื่อกดปุ่ม CALIBRATION หน้าจอแสดงผลจะเปลี่ยนไปยังรูปที่ 3.6 เมื่อกดปุ่ม Back หน้าจอแสดงผลจะเปลี่ยนกลับไปยังรูปที่ 3.3



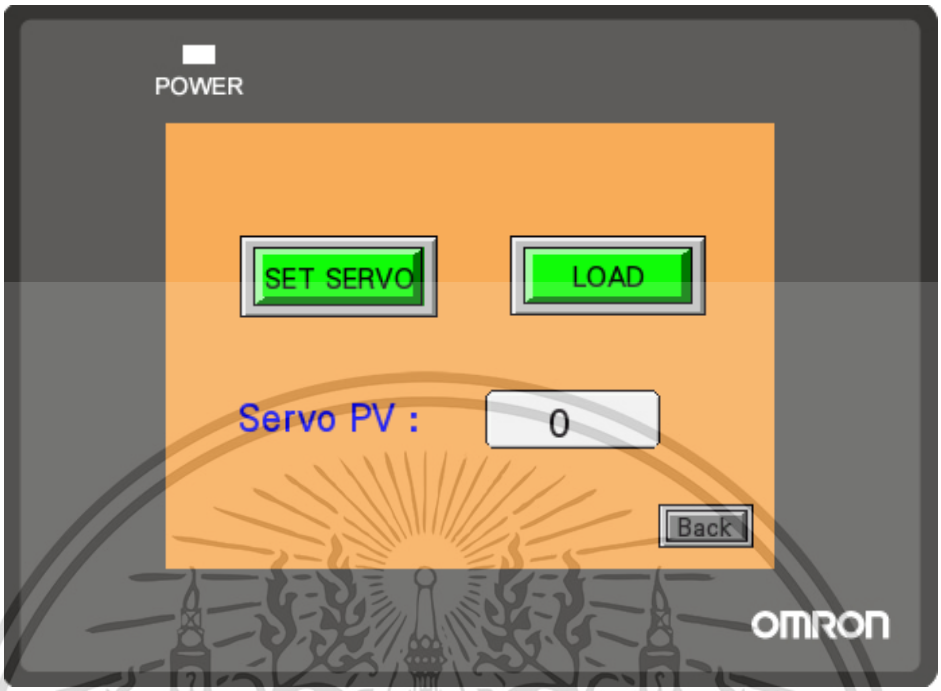
รูปที่ 3.5 หน้าจอแสดงผล หน้า PID Setting

ในหน้า PID Setting ดังแสดงในรูปข้างต้น จะสามารถกำหนดค่า K_p, T_i, T_d และ τ เมื่อกดปุ่ม Back จะกลับไปยังหน้า Set point (รูปที่ 3.4)



รูปที่ 3.6 หน้าจอแดงผล หน้า Calibration

หลังจากกดปุ่ม Calibration ในหน้า Set point (รูปที่ 3.4) จะสามารถสอบเทียบ data ที่รับมาจากกล้อง โดยการสอบเทียบทำได้โดยนำลูกบอลวางที่ตำแหน่ง 40 เซนติเมตรบนคานแล้วกดปุ่ม Max และนำลูกบอลวางที่ตำแหน่ง 1 เซนติเมตรบนคานแล้วกดปุ่ม Min เมื่อกดปุ่ม SET SERVO จะทำให้ระนาบของคานมีความชันเป็น 0° เมื่อกดปุ่ม Back จะกลับไปยังหน้า Set point (รูปที่ 3.4)



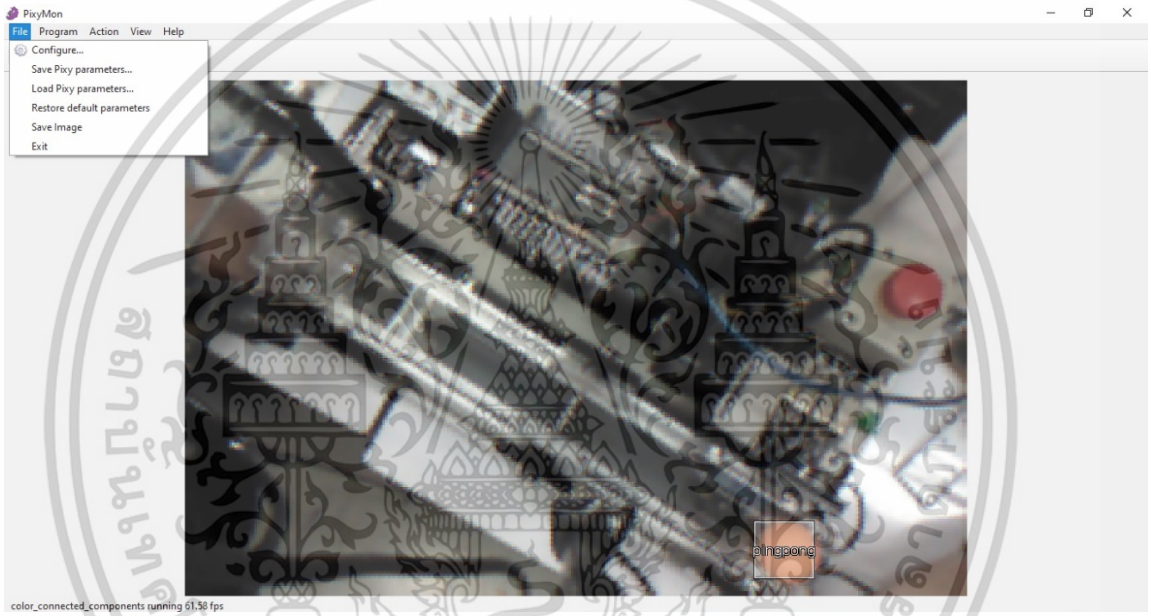
รูปที่ 3.7 หน้าจอแสดงผล หน้า Manual

เมื่อกดปุ่ม MANUAL ในรูปที่ 3.3 จะเข้าสู่โหมด MANUAL และเข้าสู่จอแสดงผลรูปที่ 3.7 ในหน้านี้สามารถกดปุ่ม LOAD เพื่อปล่อย Object และกดปุ่ม SET SERVO เพื่อทำให้ระนาบของคานมีความชันเป็น 0° โดยในโหมด MANUAL จะใช้ joystick เป็นตัวควบคุม ซึ่งสามารถอ่านค่า Present Value ของ Servo ที่ควบคุมด้วย joystick ได้อีกด้วย เมื่อกดปุ่ม Back จะกลับไปยังหน้าเลือก AUTO/MANUAL (รูปที่ 3.3)

3.3 การทำงานของ Vision camera และ Arduino Code

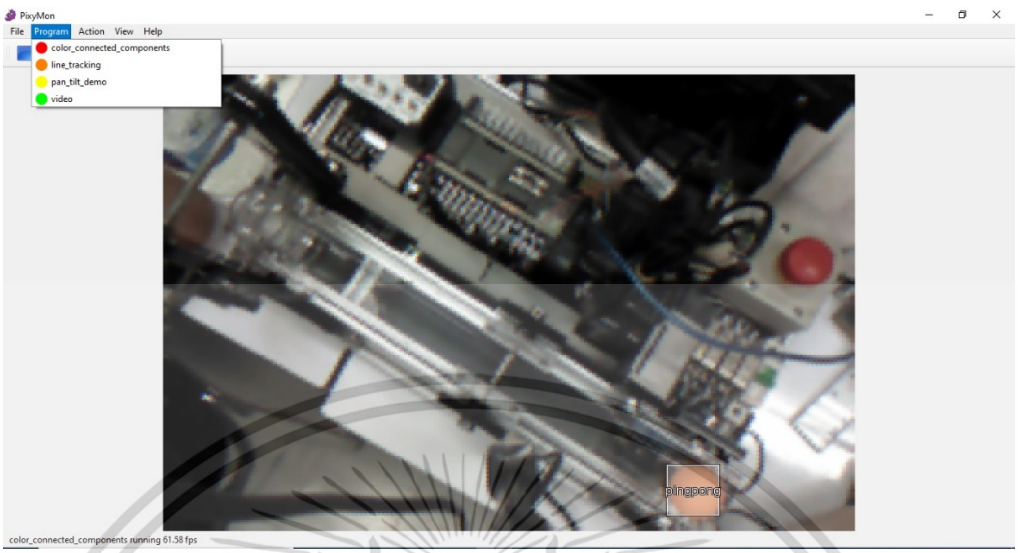
3.3.1 การใช้งานโปรแกรมสแกนวัตถุ

การใช้งานวัตถุต้องเชื่อมสายเชื่อมต่อระหว่าง Arduino และ Pixy2 เพื่อจ่ายไฟเลี้ยงให้ตัวกล้องหรืออีกวิธีเราสามารถใช้สาย USB เชื่อมต่อตัวกล้องเข้ากับคอมพิวเตอร์ได้โดยตรงเราสามารถตั้งค่าการทำงานหรือสอนให้กล้องรู้เรียนการจำแนกแยกสีโดยโปรแกรม Pixymon



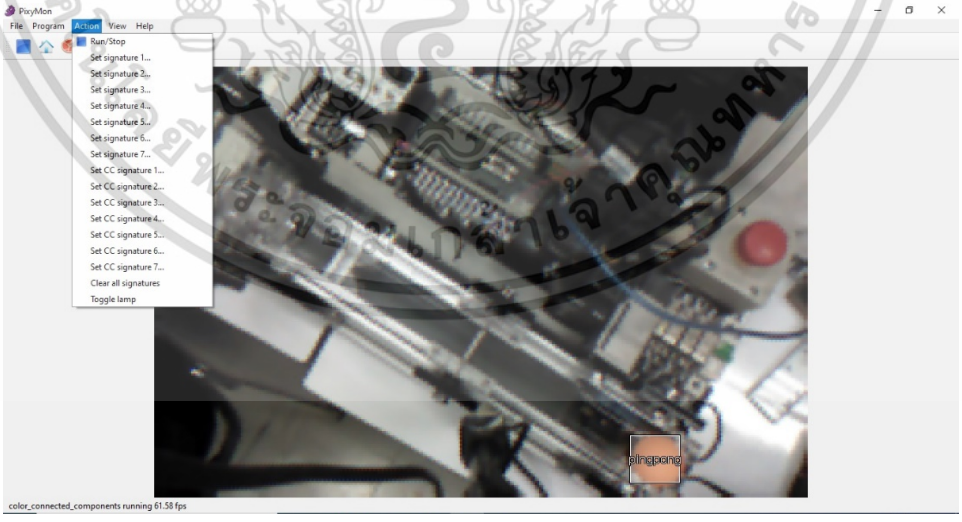
รูปที่ 3.8 หน้าตาโปรแกรม pixymon ส่วนที่ 1

ในส่วนแรก คือส่วนของ File สามารถตั้งค่าต่างๆของกล้องได้จากส่วนนี้ เช่น ค่าพารามิเตอร์ของกล้อง ค่าความกว้างของวัตถุที่ต้องการตรวจจับ หรือแม้กระทั่งตั้งชื่อของวัตถุที่เราตรวจจับได้เช่นในตัวอย่างนี้เราได้ตั้งค่าลูกปิงปองที่จับได้ว่า “pingpong”



รูปที่ 3.9 หน้าตาโปรแกรม pixymon ส่วนที่ 2

ในส่วนที่สองของโปรแกรมเราสามารถเลือกได้ว่าต้องการใช้ option ไหนของตัวกล้องซึ่งกล้อง pixy2 มีทั้งหมด 4 options ให้เลือก โดยในโปรเจคของเราเราได้เลือกใช้ color_connected _componernts (ccc) หรือจะเรียกกว่า ความสามารถในการตรวจจับและจำแนกแยกสีของกล้อง



รูปที่ 3.10 หน้าตาโปรแกรม pixymon ส่วนที่ 3

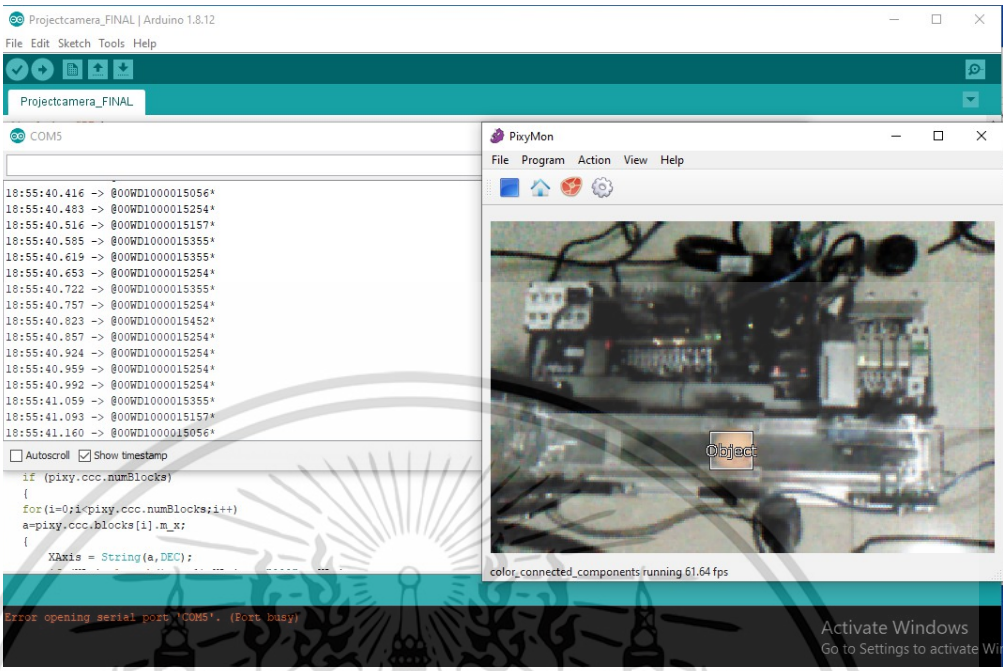
ส่วนที่ 3 ของโปรแกรมจะเกี่ยวข้องกับการกำหนดตั้งค่าสีที่ต้องการให้ Pixy ตรวจจับได้ (ตัวอย่างตรวจจับสีส้ม) โดย Pixy สามารถตรวจจับได้ 7 สี (แดง ส้ม เหลือง เขียว ฟ้า น้ำเงิน ม่วง) ถ้าต้องการตรวจจับเพียงสีเดียว ให้ทำการเคลียร์ค่าทั้งหมดก่อน ไปที่ action > clear all signatures หลังจากนั้นไปที่ เมนู Action > Set signature 1 เสร็จแล้วให้คลิกเมาส์ลากเพื่อเลือกสี วัตถุที่ต้องการ

3.3.2 การใช้ Libraries Pixy บน Arduino

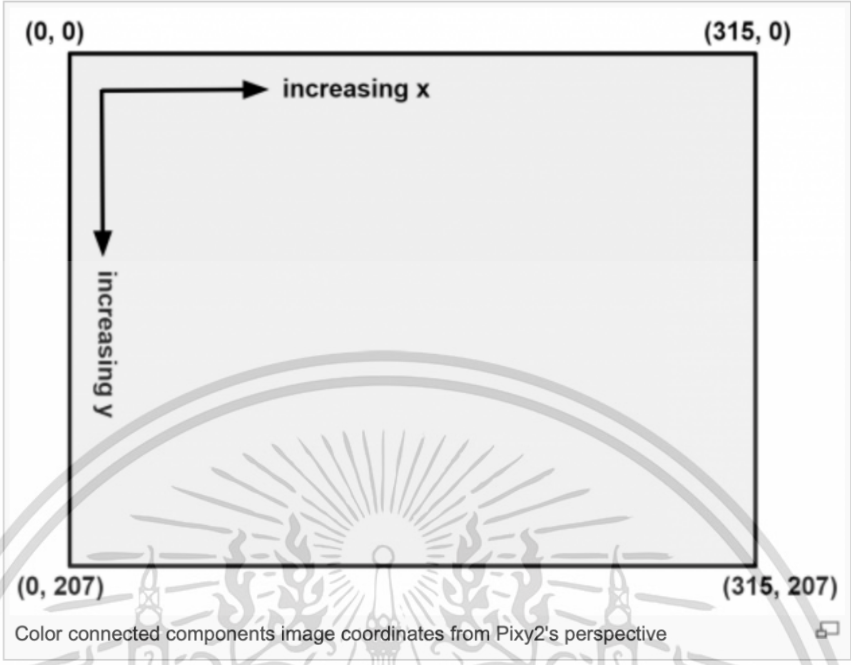
กล้อง pixy2 จะมี libraries ที่สามารถใช้งานร่วมกันระหว่างกล้องและตัวบอร์ด Arduino โดยเป็น free software ซึ่งเราสามารถเข้าไป download ได้ที่ https://pixycam.com/downloads-pixy2/ แตกไฟล์แล้วไปไว้ใน Folder ที่ติดตั้ง Arduino > Libraries. หลังจากนั้นเปิดโปรแกรม Arduino เลือก example > Pixy > hello_world แล้ว burn ลงบอร์ดที่ใช้งาน

รูปที่ 3.11 โปรแกรม IDE

ในโปรเจคของเราได้นำตัวอย่างมาปรับแก้เพื่อตรวจจับค่าของตำแหน่งที่วัตถุทรงกลมของเราเคลื่อนที่ไปโดยค่าที่แสดงออกมาคือค่าที่กล้องสามารถตรวจสอบได้ใน 1 block (ความกว้างสูงสุดที่กล้องสามารถตรวจจับได้) 0 ถึง 315



รูปที่ 3.12 แสดงค่าที่ส่งให้ PLC ทาง Serial monitor คู่กับตรวจจับว่าวัตถุปัจจุบันนั้นอยู่ตำแหน่งไหน



รูปที่ 3.13 ที่มาของค่าใน blocks ที่ pixy ตรวจจับได้ในแกน x และ y

3.3.3 Code Arduino

ในส่วนของโค้ดที่ใช้ในการเขียนโปรแกรม Arduino สำหรับกล้อง Pixy2 นั้นจะมีรายละเอียดต่างๆดังนี้

```
#include <SPI.h>
#include <Pixy2.h> //ประกาศฟังชั่นในการเรียกงานใช้กล้อง pixy2
Pixy2 pixy; //ประกาศฟังชั่นในการเรียกงานใช้กล้อง pixy2
int a,b; //กำหนดตัวแปร a และ b
void setup()
{
  Serial.begin(38400); //การประกาศความเร็วในการรับส่งข้อมูล
  Serial.begin(38400,SERIAL_8N1); //กำหนดพอร์ทสื่อสารกับ PLC
```

```
  Serial.print("Starting...\n"); //การสั่งพิมพ์ตัวอักษรที่มีชื่อว่า Starting...  ให้แสดงออกผ่านทาง Serial Monitor
  pixy.init(); //การเรียกใช้ ฟังชั่นร่วม init()
}
void loop()
{  int i; //กำหนดตัวแปร I เป็นเลขจำนวนเต็ม
  String XAxis;
  String message;
  String FCSStr;
  byte FCS;
  pixy.ccc.getBlocks(); //เรียกใช้ฟังชั่นให้กล้อง pixy ตรวจจับสีได้ทั้งหมดใน blocks
  if (pixy.ccc.numBlocks) //ถ้าตรวจเจอให้แสดงค่าออกมา
  {
    for (i=0;i<pixy.ccc.numBlocks;i++) //ใช้คำสั่งเพื่อให้ค่าที่ตรวจจับเพิ่มขึ้นหรือลดลงตามแนวแกน
    a=pixy.ccc.blocks[i].m_x; //กำหนดตัวแปร a เท่ากับค่าที่กล้องตรวจจับได้ในแกน x
    {
      XAxis = String(a,DEC);
      if (XAxis.length() == 1) XAxis = "000" + XAxis;
      if (XAxis.length() == 2) XAxis = "00" + XAxis;
      if (XAxis.length() == 3) XAxis = "0" + XAxis;
      message = "@00WD1000" + XAxis;
```

กำหนดตัวแปรชื่อต่างๆ โดยมี String , byte เป็นประเภทของตัวแปร

ส่วนของการส่งข้อมูลจาก Arduino ไปยัง PLC โดยใช้ WD1000 ของ PLC ส่งข้อมูลไปยัง Data memory ที่ 1000 ของ PLC

```
FCS = message[0];
for (byte j=1; j <= message.length()-1; j++)
{
  FCS = FCS ^ message[j];   }
FCSStr = String(FCS,HEX);
FCSStr.toUpperCase();
if (FCSStr.length() == 1) FCSStr = "0" + FCSStr;
message = message + FCSStr + "*" + char(13);
Serial.println(message); }}}
```

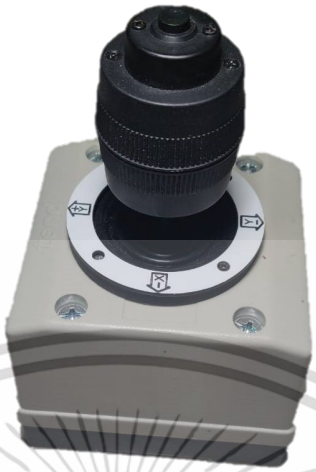
ส่วนของการส่งข้อมูลจาก Arduino ไปยัง PLC โดยใช้ WD1000 ของ PLC ส่งข้อมูลไปยัง Data memory ที่ 1000 ของ PLC

3.4 การควบคุมสมดุลของวัตถุด้วย PLC และโปรแกรม Cx-Programmer

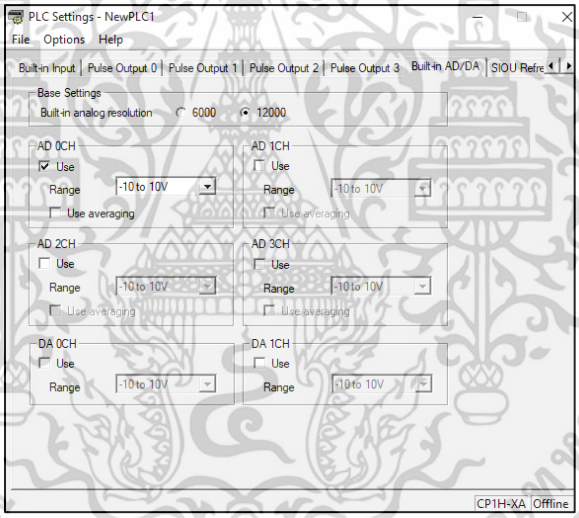
PLC ทำหน้าที่ควบคุมสมดุลของวัตถุในระนาบโดยการควบคุมด้วยมือและอัตโนมัติ โดยใช้ Cx-Programmer เป็นซอฟแวร์การเขียนโปรแกรมให้กับ PLC โดยใช้การเขียนภาษา Ladder Diagram และ Structure Text

3.4.1 การควบคุมสมดุลของวัตถุด้วย PLC โดยการควบคุมด้วยมือ (Manual Mode)

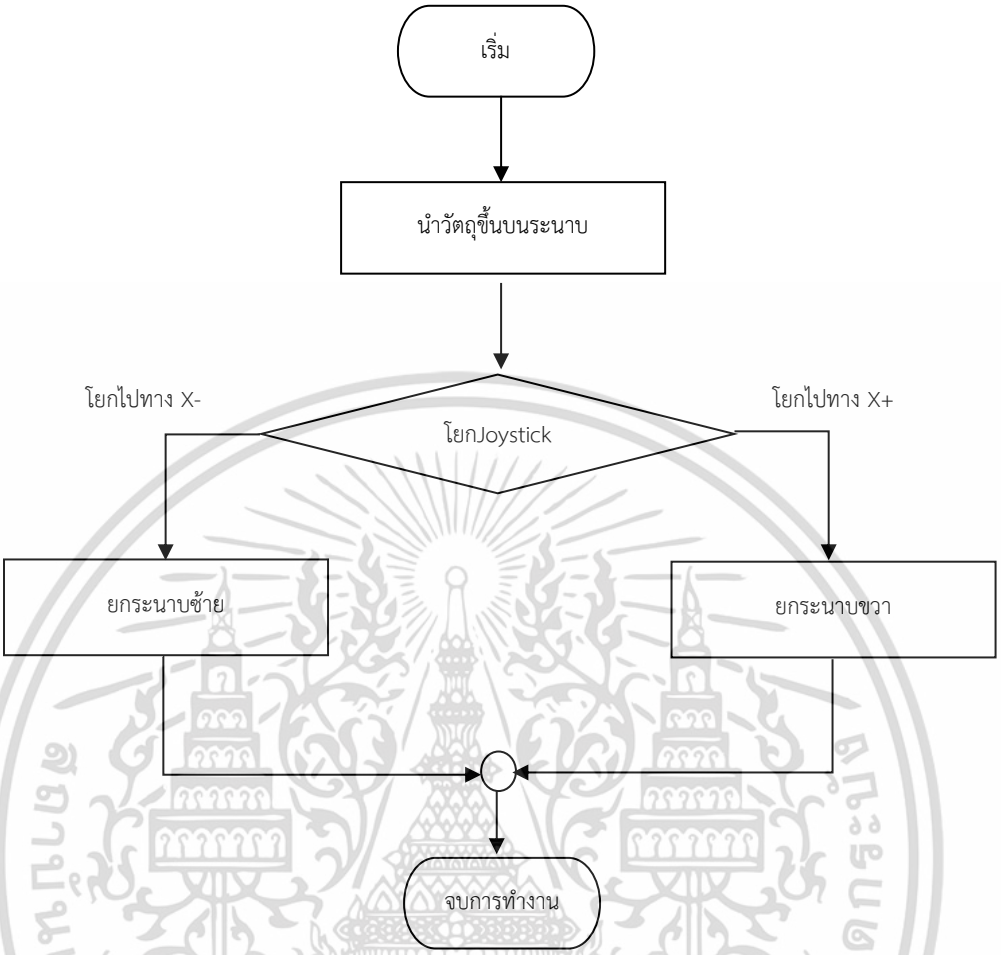
3.4.1.1 หลักการทำงานการควบคุมสมดุลของวัตถุด้วย PLC โดยการควบคุมด้วยมือ(Manual Mode) สำหรับการควบคุมด้วยมือผ่าน Joystick ในการควบคุมสมดุลวัสถุในระนาบเดียว โดยการโยก Joystick ในแกน X จะได้ค่าแรงดันไฟฟ้าตั้งแต่ -10 ถึง +10 โวลต์ และแปลงเป็นสัญญาณ Analog (-6000 ถึง 6000) แก่ PLC โดยถ้าโยก Joystick ในทางแกน X+ จะได้แรงดันไฟฟ้า 0 ถึง +10 โวลต์ และโยก Joystick ในแกน X- จะได้แรงดันไฟฟ้า 0 ถึง -10 โวลต์ เพื่อนใช่แรงดันไฟฟ้าเป็น Input ในการควบคุมสมดุลของวัตถุโดยการยกระนาบซ้าย (ในแรงดันไฟฟ้า 0 ถึง -10 โวลต์) และขวา (ในแรงดันไฟฟ้า 0 ถึง -10 โวลต์) โดยระดับการยกระนาบเป็นอัตราส่วนโดยตรงกับแรงดันไฟฟ้า



รูปที่ 3.14 Joystick



รูปที่ 3.15 Settings Analog Input (แรงดันไฟฟ้าจาก Joystick)



รูปที่ 3.16 แผนผังการควบคุมด้วยมือ (Manual Mode)

3.4.1.2 การใช้ Cx-Programmer สำหรับ PLC ในการควบคุมด้วยมือ (Manual Mode)

ในส่วน Cx-Programmer สำหรับ PLC ในการควบคุมด้วยมือ (Manual Mode) ใช้คำสั่งรับค่าจาก Joystick และขับเคลื่อน Servo Motor โดยใช้การเขียน Ladder Diagram ในการรับค่าแรงดันไฟฟ้าจาก Joystick และใช้ในการหมุน Servo Motor ด้วย Pulse & Speed และใช้ Structure Text ในการทำอัตราส่วนของแรงดันไฟฟ้าเป็น Pulse ในการหมุนของ Servo Motor

Variable Type	Name	Data Type	Retained	AT	Initial Value	Comment
Inputs	EN	BOOL	No		FALSE	Controls execution of the Function Block.
Outputs	ENO	BOOL	No		FALSE	Indicates successful execution of the Function Block.
Outputs	JOY_MAN	DINT	No		0	
Internals	AD0	DINT	No	D16		

```
[Function Block Name : Joy_man]

IF (AD0 > -5150) AND (AD0 <5150) THEN
        JOY_MAN := AD0 + 5150 ;
        JOY_MAN := JOY_MAN *1772 ;
        JOY_MAN := JOY_MAN / 10300 ;
        JOY_MAN:= JOY_MAN+154;
END_IF;
IF (AD0 >5150) THEN
        JOY_MAN := 1926;
END_IF;
IF(AD0 <-5150) THEN
        JOY_MAN := 154;
END_IF;
```

รูปที่ 3.17 Structure text การทำอัตราส่วนแรงดันไฟฟ้าเป็น Pulse ในการหมุน Servo Motor

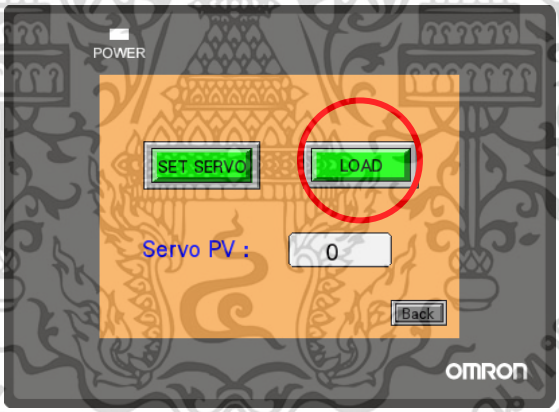
รูปที่ 3.18 Ladder Diagram ในการหมุน Servo Motor ด้วย Pulse&speed ใน Manual Mode

3.4.1.3 การใช้งานการควบคุมสมดุลวัตถุบนระนาบด้วยมือ (Manual Mode)

- กดปุ่ม Manual Mode บนหน้าจอ HMI

รูปที่ 3. 19 การเลือก Manual Mode

- ถ้าเกิดคานไม่เป็นระนาบในขณะไม่โยก Joystick ให้กดปุ่ม SET SERVO เพื่อปรับระดับระนาบ
- กดปุ่ม Load เพื่อนำวัตถุขึ้นบนระนาบ

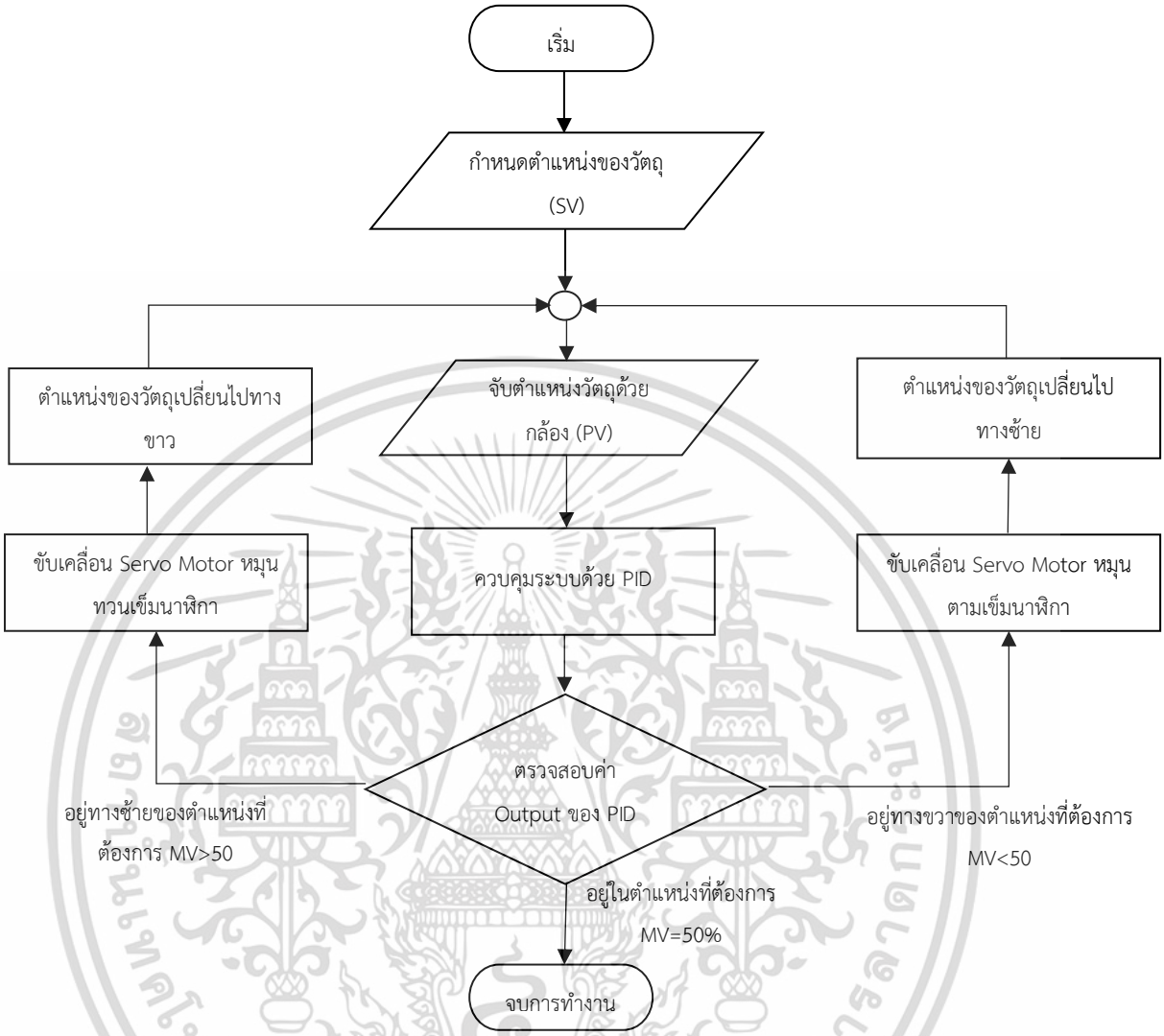


รูปที่ 3.20 ปุ่ม Load

- โยก Joystick เพื่อควบคุมวัตถุในอยู่ในตำแหน่งที่ต้องการ
- Servo PV มีไว้เพื่อแสดงตำแหนงการเคลื่อนที่ของระนาบ

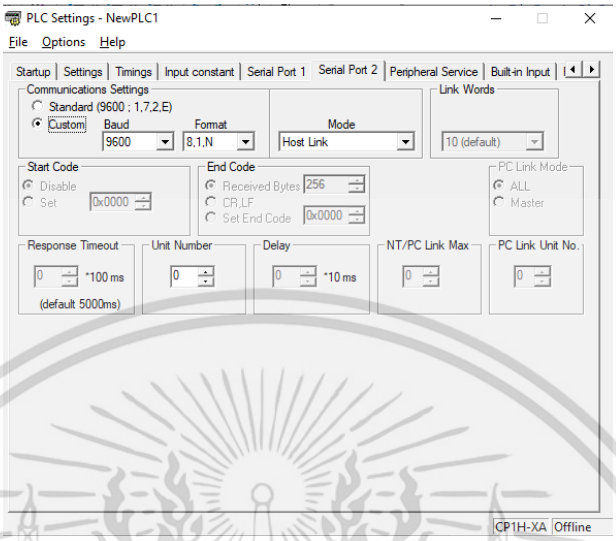
3.4.2 การควบคุมสมดุลของวัตถุด้วย PLC โดยการควบคุมแบบอัตโนมัติ(Auto Mode)

3.4.2.1 หลักการทำงานการควบคุมสมดุลของวัตถุด้วย PLC โดยการควบคุมแบบอัตโนมัติ(Auto Mode) สำหรับการควบคุมแบบอัตโนมัติ จะใช้กล้องเป็นตัวจับตำแหน่งของวัตถุเพื่อใช้ในการควบคุมแบบ PID โดยตำแหน่งวัตถุที่ได้จากกล้อง (PV) มาคิดกับค่า Setpoint ที่เรากำหนดจาก HMI (SV) เป็นค่าเป็นหมายของการควบคุม ผ่านการควบคุม PID เพื่อนำ Output (MV 0% ถึง 100%) มาขับเคลื่อน Servo Motor ให้ปรับระดับระนาบเพื่อให้วัตถุเข้าสู่สมดุล โดยตั้งค่า Output ที่ตำแหน่งต้องการไว้ MV =50% และเมื่อวัตถุอยู่ทางซ้ายของตำแหน่งที่กำหนด MV มากกว่า 50% ให้ทำการขับเคลื่อน Serve Motor หมุนทวนเข็มนาฬิกาเพื่อให้ยกระนาบซ้ายเป็นอัตราส่วนโดยตรงกับ MV และเมื่อวัตถุอยู่ทางซ้ายของตำแหน่งที่กำหนด MV น้อยกว่า 50% ให้ทำการขับเคลื่อน Serve Motor หมุนทวนเข็มนาฬิกาเพื่อให้ยกระนาบขวาเป็นอัตราส่วนโดยตรงกับ MV เพื่อให้วัตถุเข้าสู่ตำแหน่งที่ต้องการ

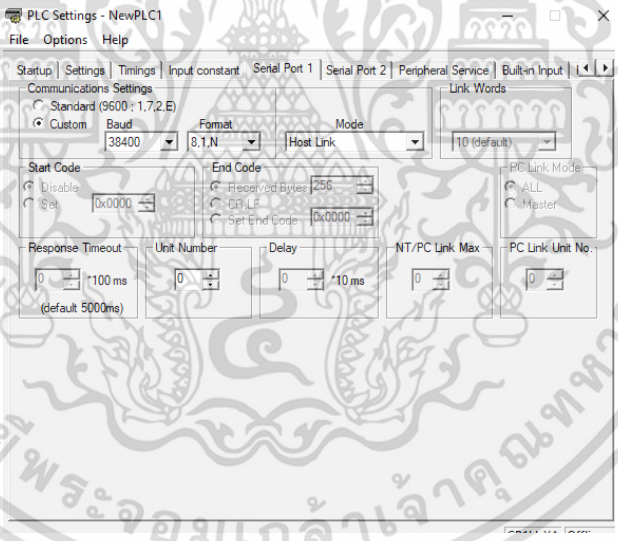


รูปที่ 3.21 แผนผังการควบคุมแบบอัตโนมัติ(Auto Mode)

3.4.2.2 การใช้ Cx-Programmer สำหรับ PLC ในการควบคุมแบบอัตโนมัติ (Auto Mode) จะรับค่า Setpoint จากหน้าจอ HMI ผ่านสาย RS-232 โดยตั้งค่าที่ Serial port 2 เป็น Baud 9600 ,Format 8,1,N and Mode Host link ตามรูปที่ 3.21 และรับค่าตำแหน่งของวัตถุจากกล้องผ่าน Arduino ผ่านสาย RS-232 โดยตั้งค่าที่ Serial port 1 เป็น Baud 38400 ,Format 8,1,N and Mode Host link ตามรูปที่ 3.21 จากนั้นนำค่าที่ได้มาทำการควบคุมแบบ PID โดยให้ค่า Setpoint ที่ได้จากจอ HMI เป็น SV และให้ค่าตำแหน่งที่ได้จากกล้องเป็น PV และได้ Output เป็นค่า MV และทำการปรับอัตราส่วน MV ให้กลายเป็นค่าที่นำมาขับเคลื่อนให้ Servo Motor หมุน โดยการตอบสนองของ Output จะขึ้นกับการปรับตัวแปร PID เพื่อให้การตอบสนองของ MV ดีที่สุด



รูปที่ 3.22 Settings Serial Port2 สำหรับหน้าจอ HMI



รูปที่ 3.23 Settings Serial Port1 สำหรับ Arduino

Variable Type	Name	Data Type	Retained	AT	Initial Value	Comment
Inputs	EN	BOOL	No		FALSE	Controls execution of the Function Block.
Inputs	MV	DINT	No		0	
Outputs	ENO	BOOL	No		FALSE	Indicates successful execution of the Function Block.
Outputs	Plus	DINT	No		963	
Internals	Real_MV	REAL	No		0.0	
Internals	Real_Plus	REAL	No		0.0	
Internals	Real_M1	REAL	No		0.0	
Internals	Real_M2	REAL	No		0.0	
Internals	Real_OS1	REAL	No		0.0	
Internals	Real_OS2	REAL	No		0.0	
Internals	Real_Min	REAL	No		154.00	
Internals	Real_Mid	REAL	No		1040.0	
Internals	Real_Max	REAL	No		1926.0	

```
[Function Block Name : SERVOAUTO]

Real_MV:= DINT_TO_REAL(MV);
Real_OS1:= Real_Mid-Real_Min;
Real_OS2:= Real_Max-Real_Mid;
Real_M2:= Real_OS2/-2000.00;
Real_M1:= Real_OS1/-2000.00;
IF (MV <4000) AND (MV >2000) THEN
        Real_Plus := Real_MV -2000.00;
        Real_Plus := Real_Plus*Real_M1;
        Real_Plus := Real_Plus+Real_Mid;
END_IF;
IF (MV <2000) AND (MV >0) THEN
        Real_Plus := Real_MV -0.00;
        Real_Plus := Real_Plus*Real_M2;
        Real_Plus := Real_Plus+Real_Max;
END_IF;
IF (MV <0) THEN
        Real_Plus := Real_Max;
END_IF;
IF(MV > 4000) THEN
        Real_Plus := Real_Min;
END_IF;
Plus:= REAL_TO_DINT(Real_Plus);
```

รูปที่ 3.24 Structure text การทำอัตราส่วน MV ของ PID เป็น Pulse ในการหมุน Servo Motor

รูปที่ 3.25 Ladder Diagram ในการหมุน Servo Motor ด้วย Pulse & speed ใน Auto Mode

Variable Type	Name	Data Type	Retained	AT	Initial Value	Comment
Inputs	EN	BOOL	No		FALSE	Controls execution of the Function Block.
Inputs	Input_min	DINT	No		0	
Inputs	Input_max	DINT	No		0	
Inputs	Output_min	DINT	No		0	
Inputs	Output_max	DINT	No		0	
Inputs	Input	DINT	No		0	
Outputs	ENO	BOOL	No		FALSE	Indicates successful execution of the Function Block.
Outputs	Output	DINT	No		0	
Internals	Span_input	REAL	No		0.0	
Internals	Span_output	REAL	No		0.0	
Internals	Slope	REAL	No		0.0	
Internals	Real_Input_max	REAL	No		0.0	
Internals	Real_Input_min	REAL	No		0.0	
Internals	Real_Output_max	REAL	No		0.0	
Internals	Real_Output_min	REAL	No		0.0	
Internals	Real_Input	REAL	No		0.0	
Internals	Real_Output	REAL	No		0.0	

[Function Block Name : Scale]

```
Real_Input := DINT_TO_REAL(Input);
Real_Input_max := DINT_TO_REAL(Input_max);
Real_Input_min :=  DINT_TO_REAL(Input_min);
Real_Output_max := DINT_TO_REAL(Output_max);
Real_Output_min :=  DINT_TO_REAL(Output_min);
Span_input := Real_Input_max-Real_Input_min;
Span_output := Real_Output_max-Real_Output_min;
Slope := Span_output/Span_input;
Real_Output := (Slope*(Real_Input-Real_Input_min))+Real_Output_min;
Output := REAL_TO_DINT(Real_Output);
```

รูปที่ 3.26 Structure text การทำอัตราส่วนค่าที่ได้จากกล้อง เป็นตำแหน่ง

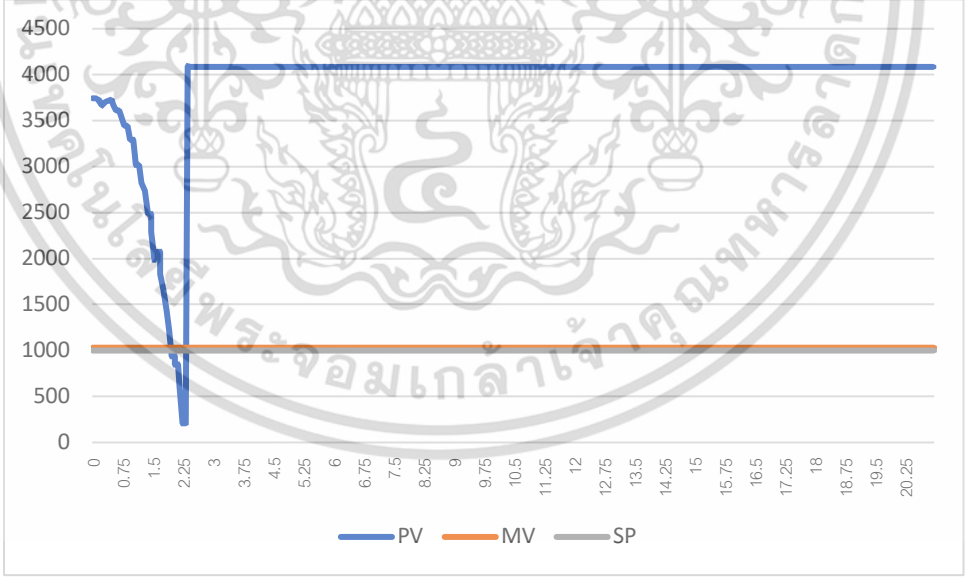
บทที่ 4

ผลการทดลอง

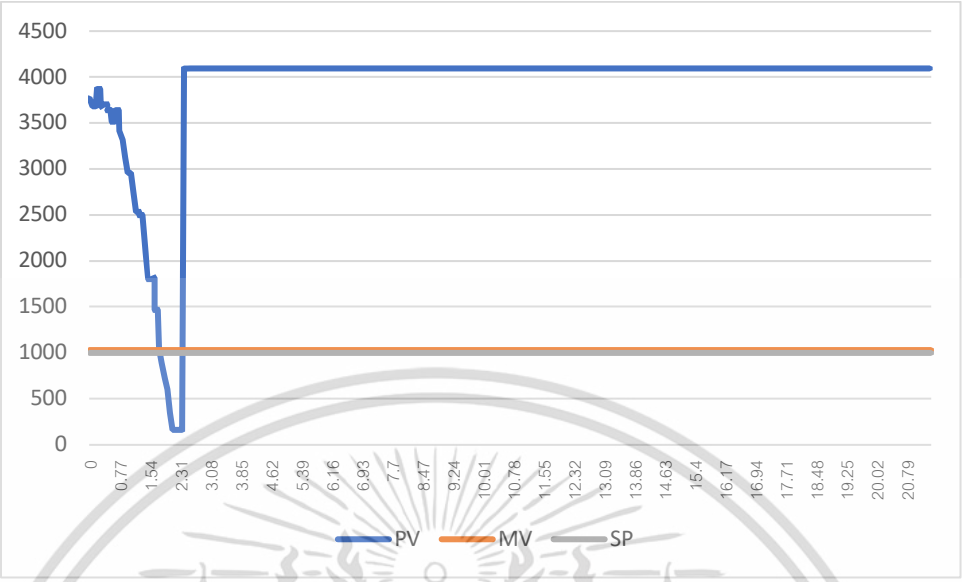
จากการดำเนินงานในส่วนของการควบคุมโดยใช้ joystick ใน Manual Mode ทำให้นักศึกษาสามารถเข้าใจถึงการเคลื่อนที่ของคานซึ่งเคลื่อนที่โดย Servo Motor นำไปสู่การหาค่าพารามิเตอร์ของ PID Control ใน Auto Mode ซึ่งขั้นแรกของการทดลองนักศึกษาได้ทำการปรับ Tp=1000 และทำการปิดค่า I และD แล้วศึกษาการเคลื่อนที่ของลูกบอลบนคาน จากนั้นจึงเริ่มทดลองใส่ค่าพารามิเตอร์ต่างๆ โดยเราได้ทำการทดลองที่ Setpoint 3 จุดเป็นหลักนั่นคือ 10, 20 และ 30 เซนติเมตร ซึ่งผลที่ได้จะแสดงดังต่อไปนี้

4.1 การทดลองโดยการปรับค่า Tp = 1000 และปิดค่า I (Ti = 9999) และ D (Td = 0) แต่ละ Setpoint 10, 20 และ 30 เซนติเมตร

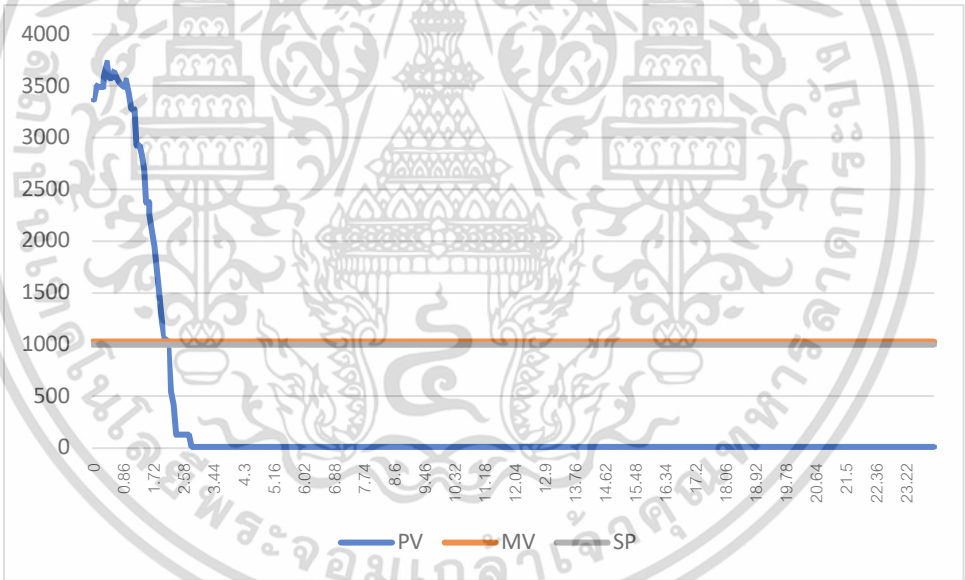
4.1.1 ที่ Setpoint = 10 โดยการปรับค่า Tp = 1000 และปิดค่า I (Ti = 9999) และ D (Td = 0)



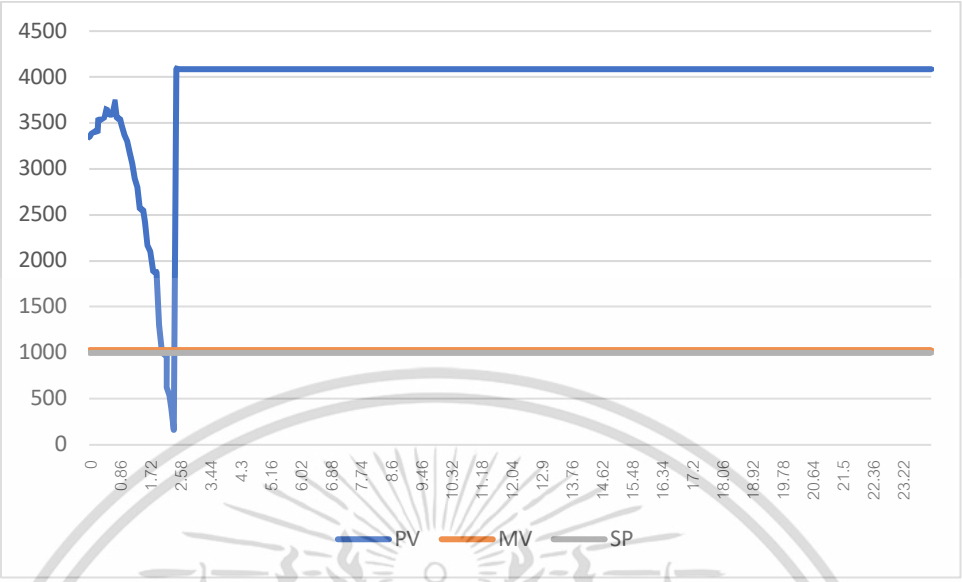
รูปที่ 4.1 Setpoint = 10 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 1



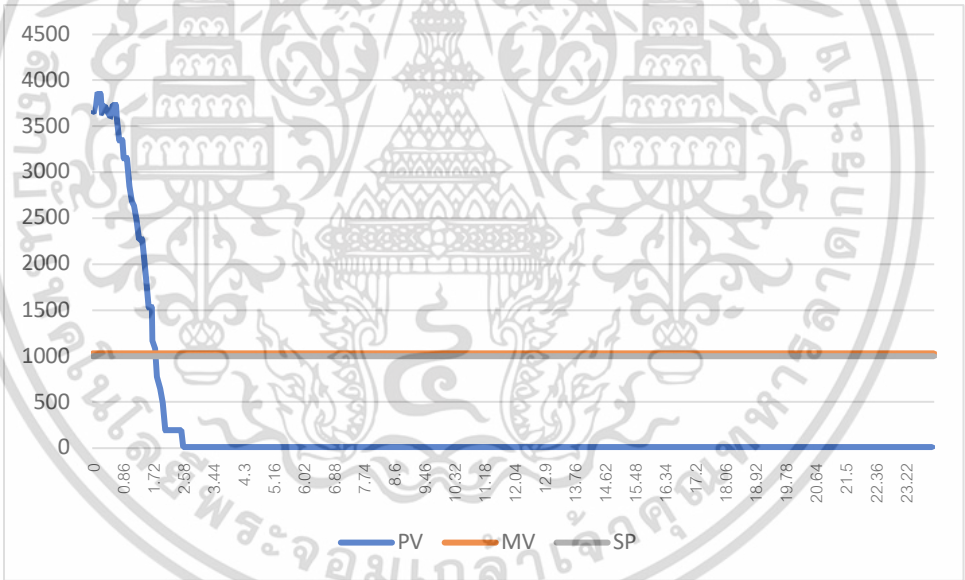
รูปที่ 4.2 Setpoint = 10 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 2



รูปที่ 4.3 Setpoint = 10 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 3



รูปที่ 4.4 Setpoint = 10 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 4

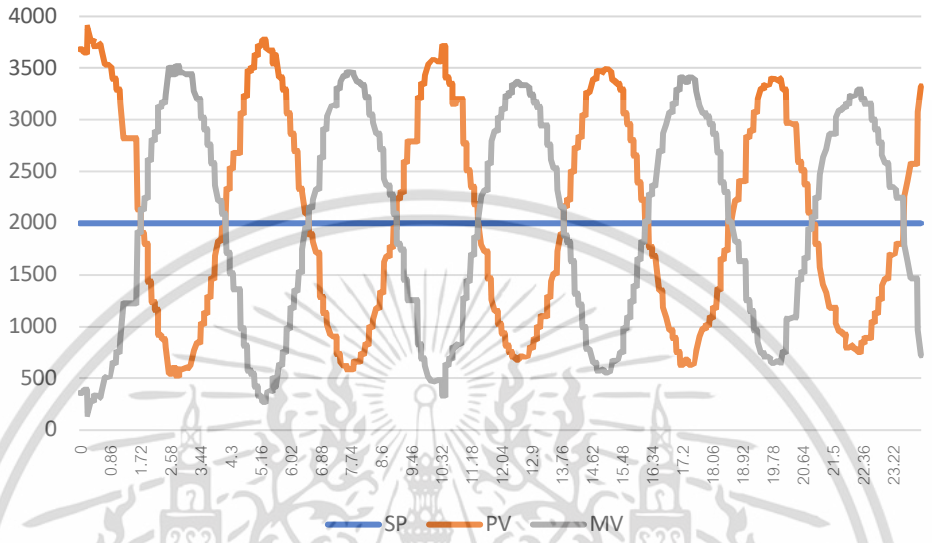


รูปที่ 4.5 Setpoint = 10 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 5

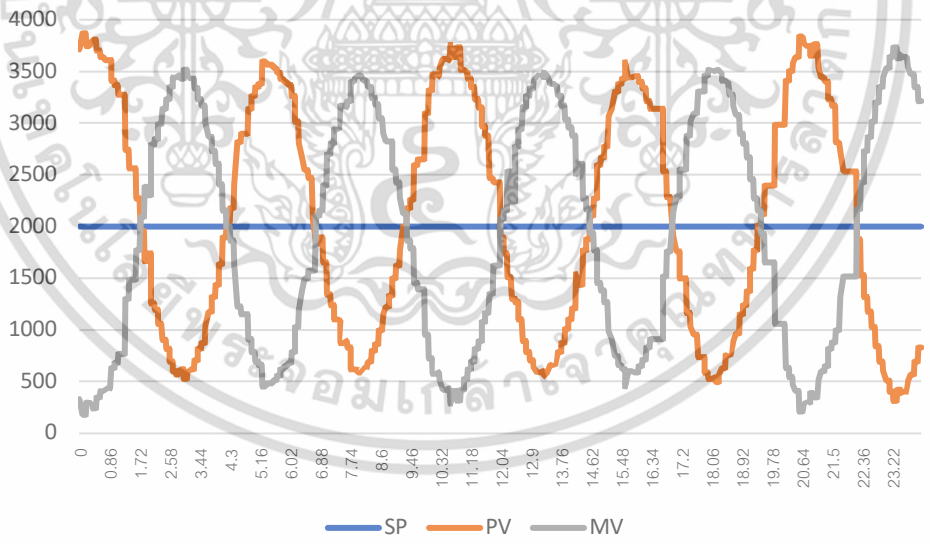
<u>สรุปผลการทดลอง</u>

จากการทดลองที่ Setpoint = 10 โดยการปรับค่า Tp = 1000 และปิดค่า I (Ti = 9999) และ D (Td = 0) จะเห็นว่าไม่สามารถเข้าสู้ Setpoint ได้ และค่า Overshot เกินขอบเขตของการทดลองทำให้วัตถุทรงกลมหลุดออกจากระนาบ

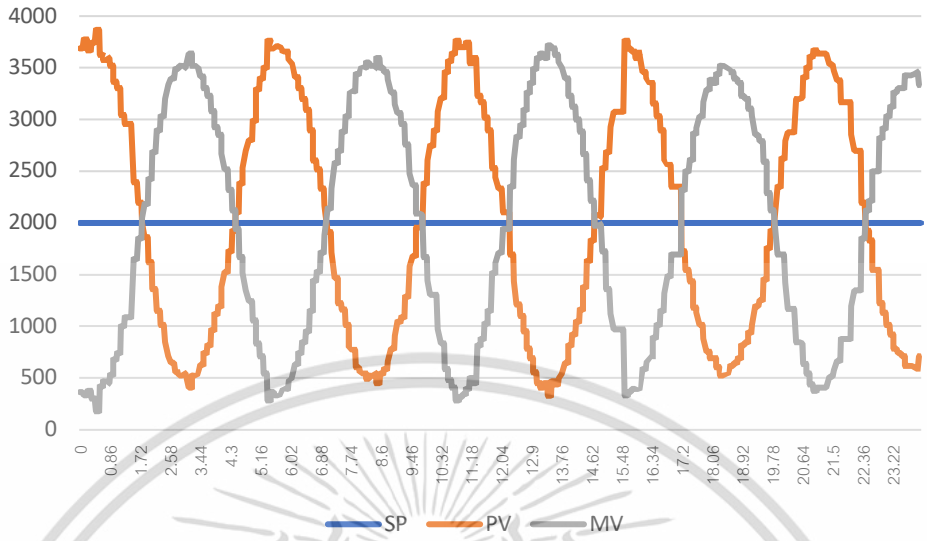
4.1.2 ที่ Setpoint = 20 โดยการปรับค่า Tp = 1000 และค่า I (Ti = 9999) และ D (Td = 0)



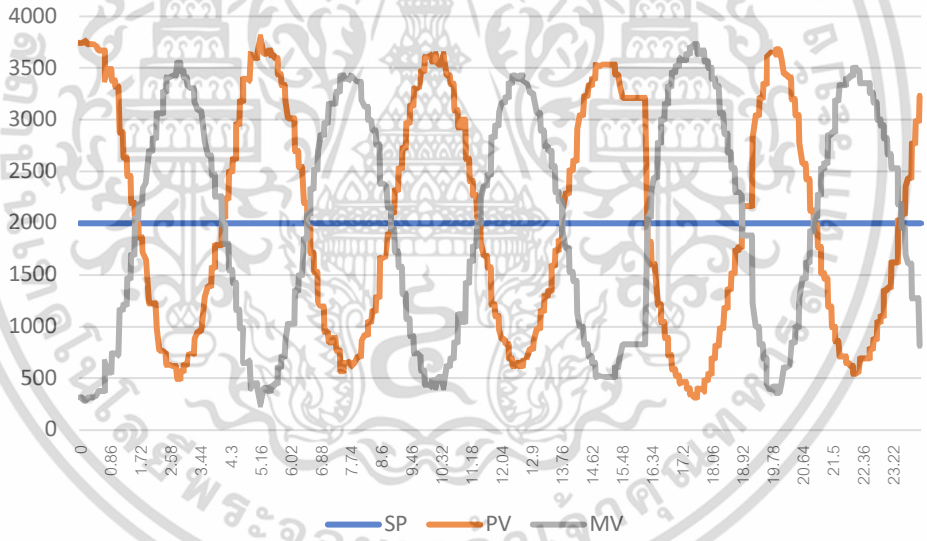
รูปที่ 4.6 Setpoint = 20 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 1



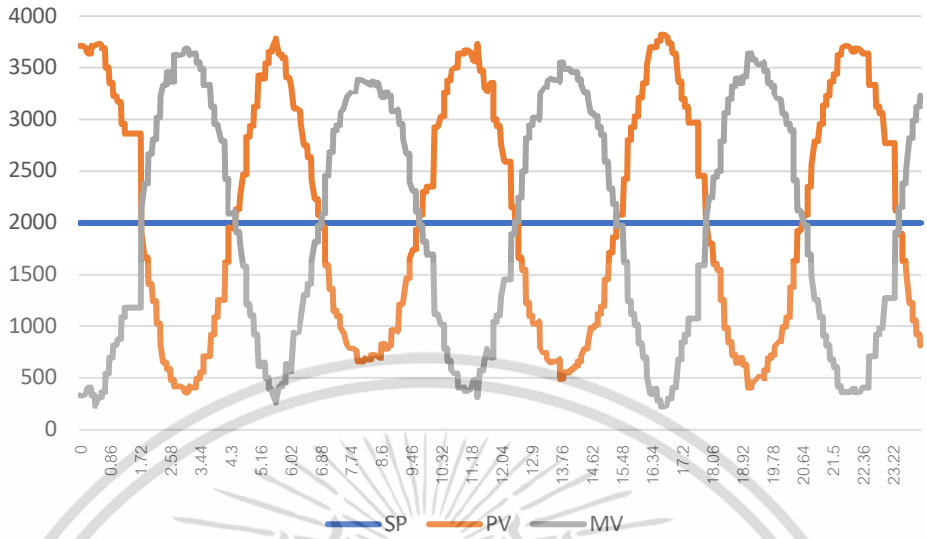
รูปที่ 4.7 Setpoint = 20 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 2



รูปที่ 4.8 Setpoint = 20 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 3



รูปที่ 4.9 Setpoint = 20 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 4

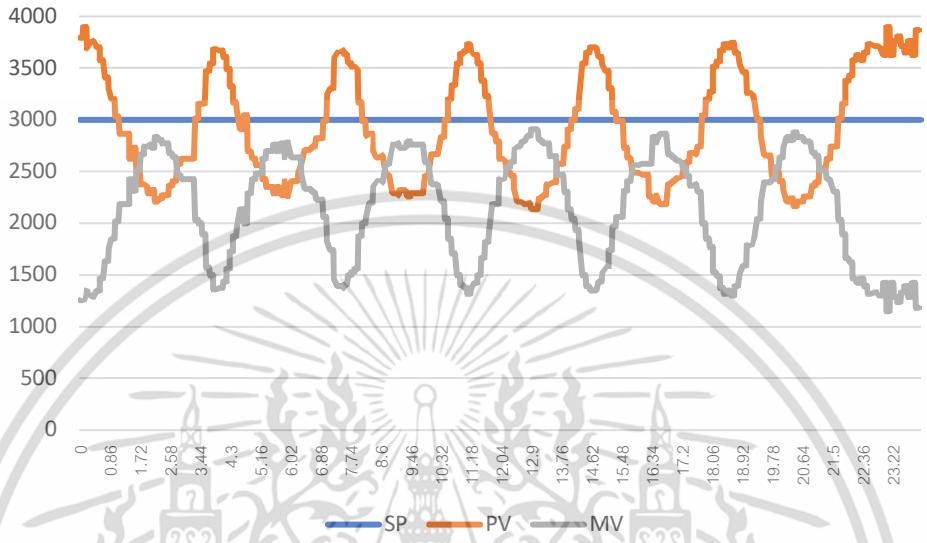


รูปที่ 4.10 Setpoint = 20 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 5

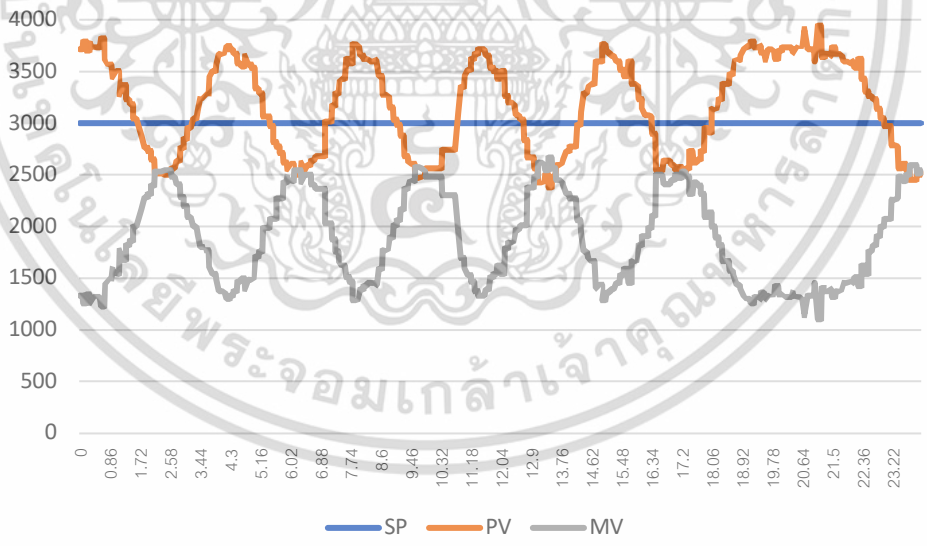
<u>สรุปผลการทดลอง</u>

จากการทดลองที่ Setpoint = 20 โดยการปรับค่า Tp = 1000 และค่า I (Ti = 9999) และ D (Td=0) จะเห็นว่าไม่สามารถเข้าสู้ Setpoint ได้ และมีเกิดการแกว่งรอบ Setpoint เช่นเดียวกับที่ Setpoint = 10 เนื่องจากมี Overshoot ที่สูงเกินไป

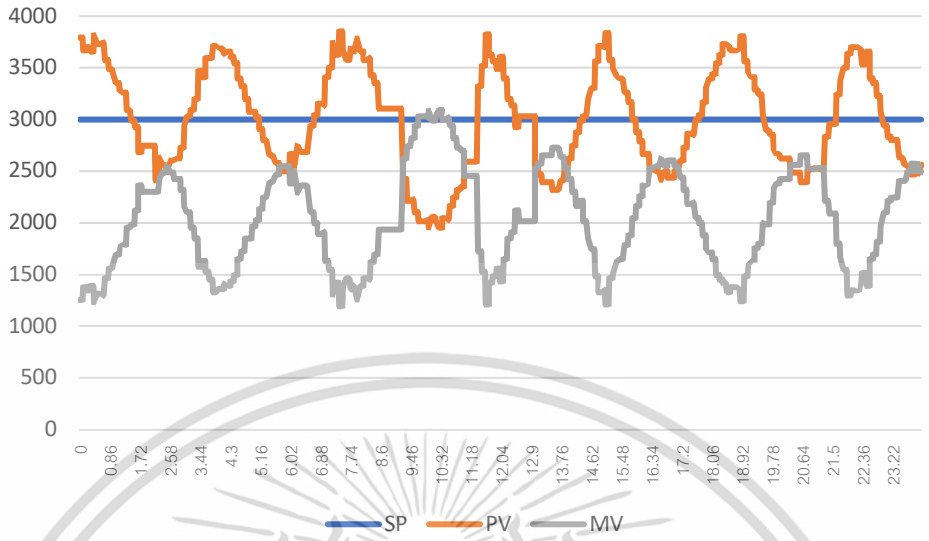
4.1.3 ที่ Setpoint = 30 โดยการปรับค่า Tp = 1000 และค่า I (Ti = 9999) และ D (Td = 0)



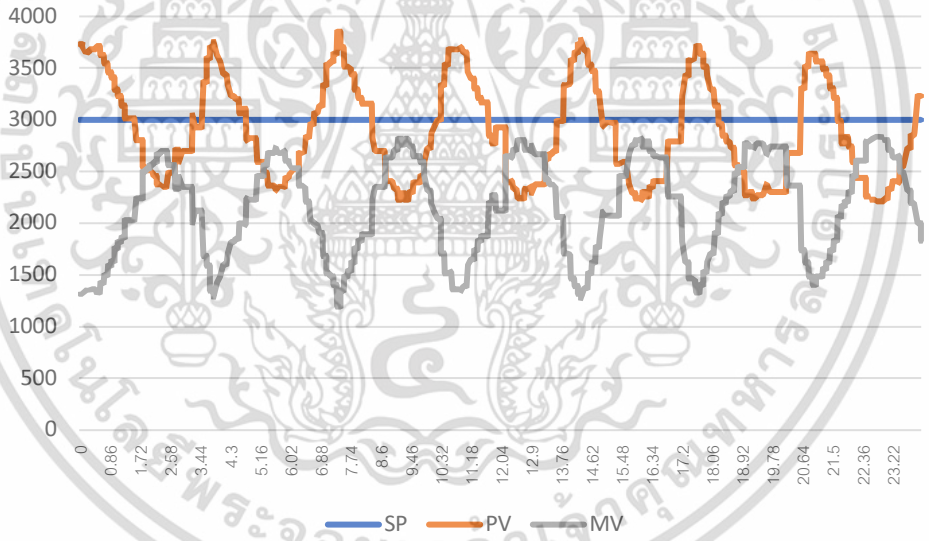
รูปที่ 4.11 Setpoint = 30 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 1



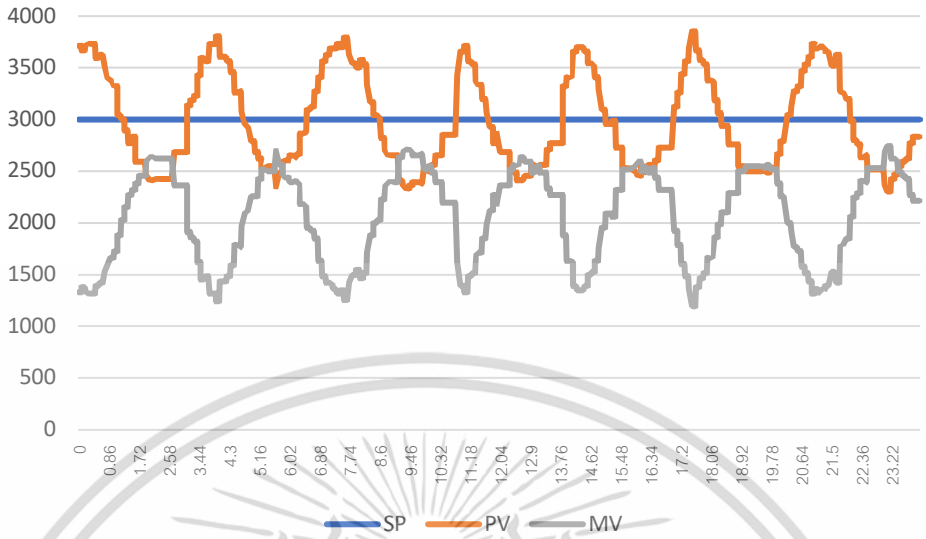
รูปที่ 4.12 Setpoint = 30 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 2



รูปที่ 4.13 Setpoint = 30 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 3



รูปที่ 4.14 Setpoint = 30 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 4



รูปที่ 4.15 Setpoint = 30 โดยการปรับค่า Tp = 1000 และปิดค่า I และ D ครั้งที่ 5

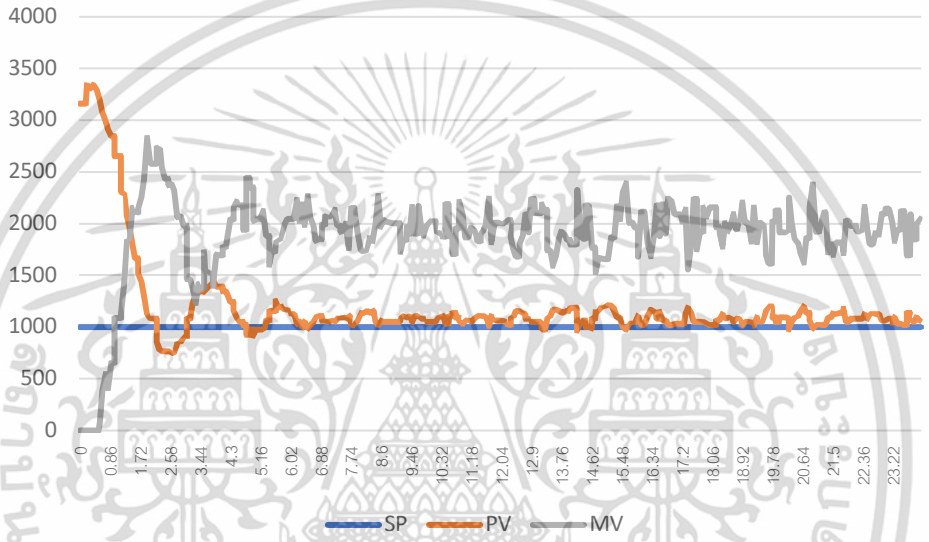
<u>สรุปผลการทดลอง</u>

จากการทดลองที่ Setpoint = 30 โดยการปรับค่า Tp = 1000 และค่า I (Ti = 9999) และ D (Td=0) จะเห็นว่าไม่สามารถเข้าสู้ Setpoint ได้ และมีเกิดการแกว่งรอบ Setpoint เช่นเดียวกับที่ Setpoint = 10 และ Setpoint = 20 เนื่องจากมี Overshoot ที่สูงเกินไป

4.2 การทดลองค่า P, I และ D ต่างๆด้วย Tp, Ti และ Td ณ แต่ละ Setpoint 10, 20 และ 30 เซนติเมตร

จากการทดลองโดยการปรับค่า Tp = 1000 และค่า I (Ti = 9999) และ D (Td = 0) จะเห็นได้ว่าเกิดค่า Overshoot ที่มากเกินไปเราจึงทำการปรับค่าพารามิเตอร์ D (Td = 45) เพื่อนำไปลดค่า Overshoot

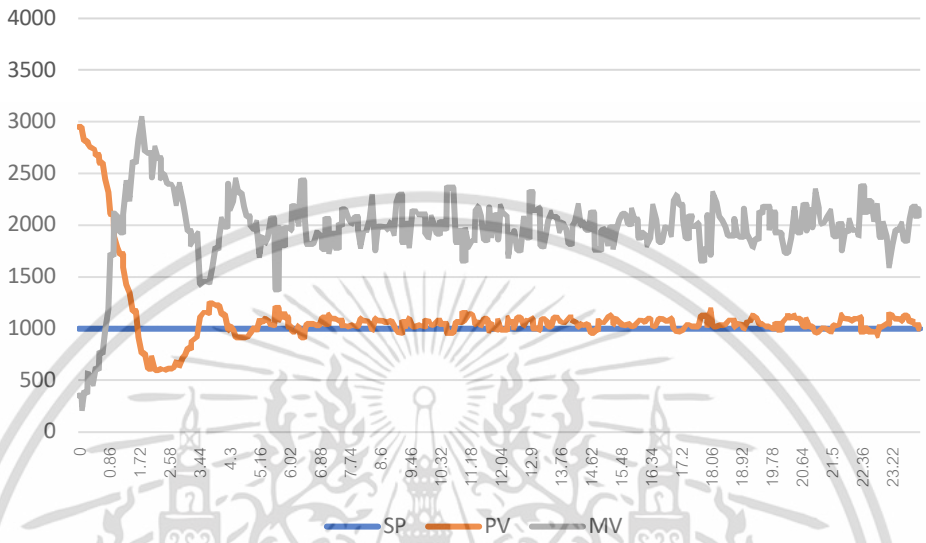
4.2.1 ที่ Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45



รูปที่ 4.16 Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 1

Setpoint	8.46s	Error <1cm
Peak Time	2.61s	
Rise Time	2.21s	
Overshoot	8.36667%	

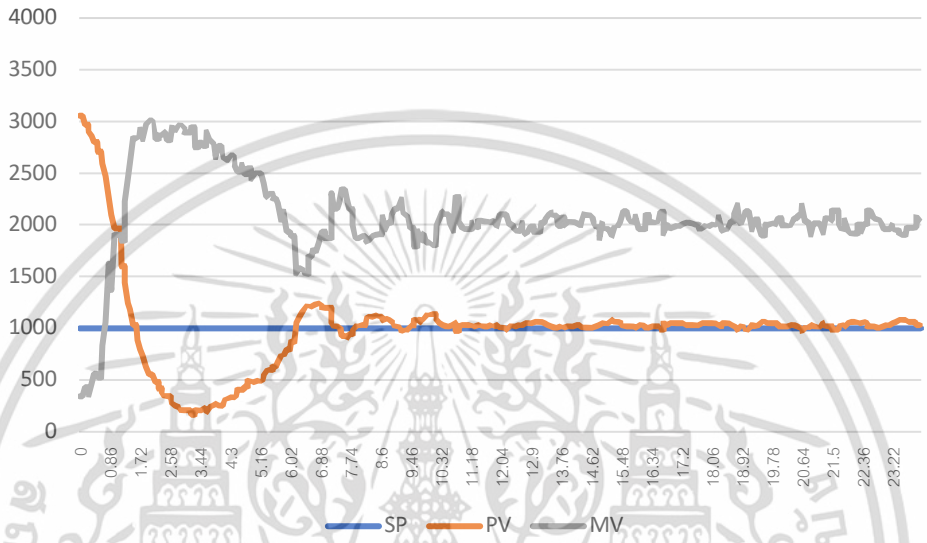
ตารางที่ 4.1 Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 1



รูปที่ 4.17 Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 2

Setpoint	4.27s	Error <1cm
Peak Time	2.19s	
Rise Time	1.72s	
Overshoot	13.3333%	

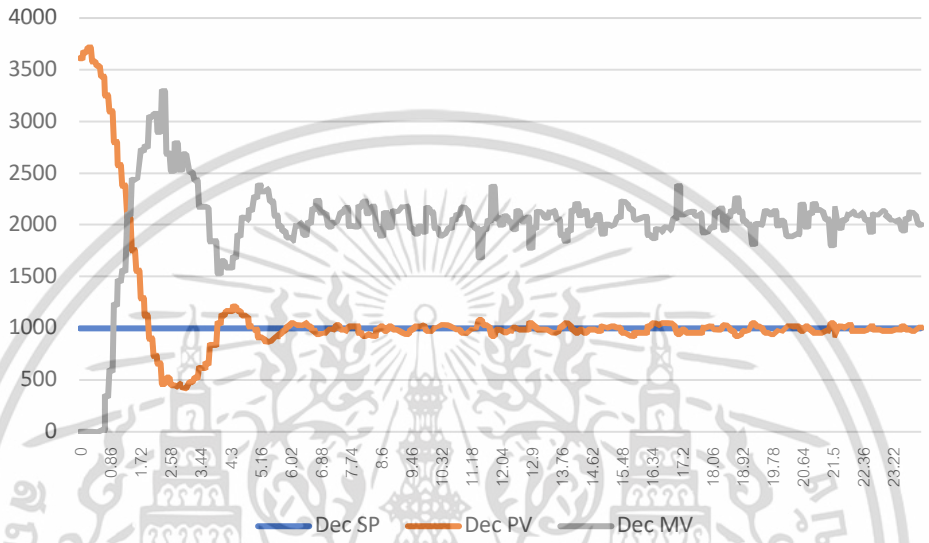
ตารางที่ 4.2 Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 2



รูปที่ 4.18 Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 3

Setpoint	5.03s	Error <1cm
Peak Time	2.15s	
Rise Time	1.71s	
Overshoot	12.8333%	

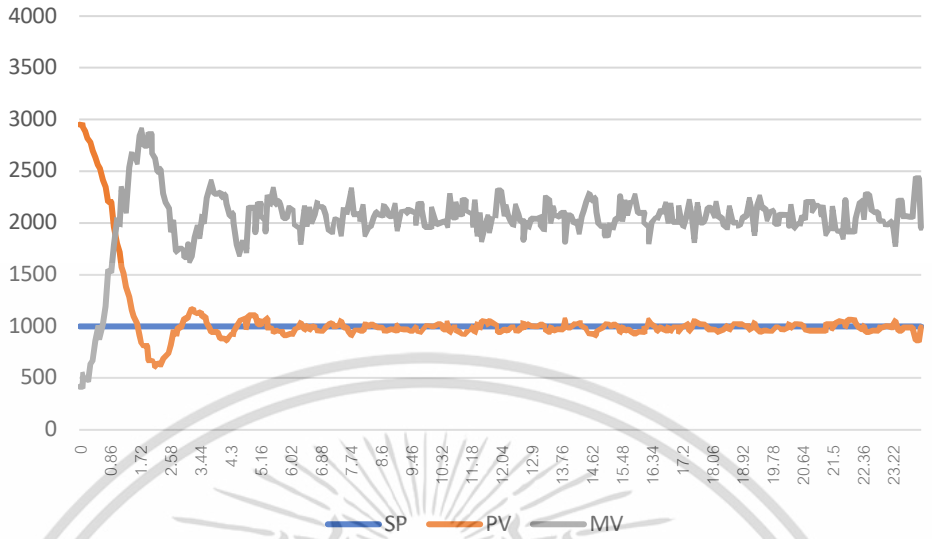
ตารางที่ 4.3 Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 3



รูปที่ 4.19 Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 4

Setpoint	5.57s	Error <1cm
Peak Time	2.96s	
Rise Time	1.99s	
Overshoot	19.3333%	

ตารางที่ 4.4 Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 4



รูปที่ 4.20 Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 5

Setpoint	5.03s	Error <1cm
Peak Time	2.15s	
Rise Time	1.71s	
Overshoot	12.8333	

ตารางที่ 4.5 Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 5

จากการทดลองที่ Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ทั้ง 5 ครั้ง ได้ค่าเฉลี่ยดังต่อไปนี้

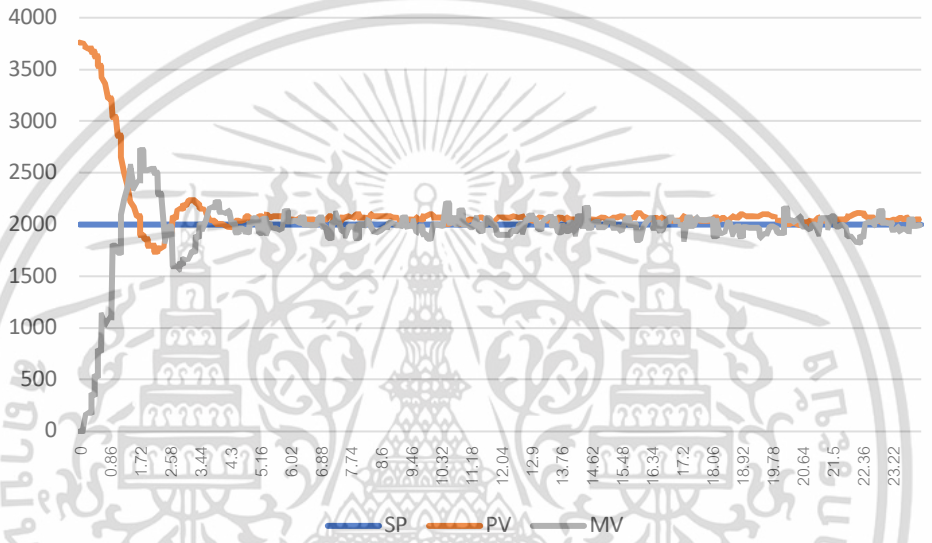
average	setpoint	6.706 s
	Peak time	2.622 s
	Rise Time	1.858 s
	Overshoot	13.9498%

ตารางที่ 4.6 Average ที่ Setpoint = 10 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ทั้ง 5 ครั้ง

<u>สรุปผลการทดลอง</u>

จากการทดลองที่ Setpoint = 10 โดยการปรับค่า Tp = 1000 ค่า I (Ti = 9999) และ D (Td=45) จะเห็นว่าสามารถเข้าสู่ Setpoint ได้ ที่ Average time constant = 6.706 s และมี Average overshoot ที่ 13.9498% ซึ่งมีค่าลดลงจากการทดลองที่ปิดค่า P และ D ทำให้ค่าสามารถเข้าสู่ Setpoint ได้

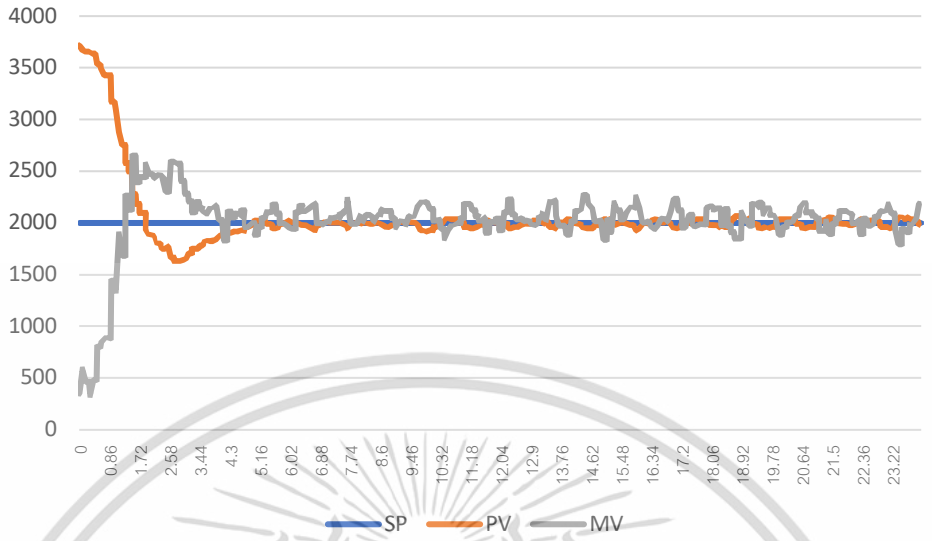
4.2.2 ที่ Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45



รูปที่ 4.21 Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 1

setpoint	3.67s	at error <1cm.
Peak Time	2.14s	
Rise Time	1.73s	
Overshoot	13.1%	

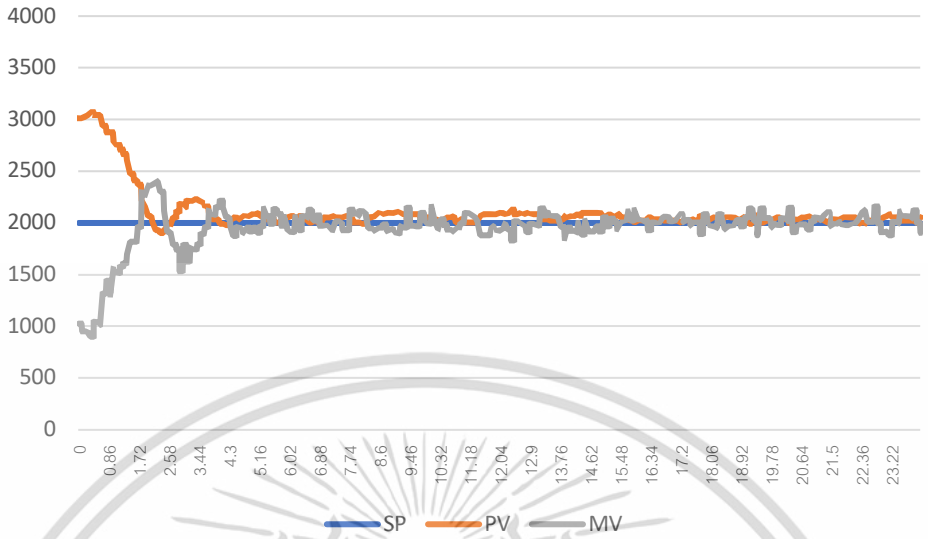
ตารางที่ 4.7 Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 1



รูปที่ 4.22 Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 2

setpoint	4.13s	at error <1cm.
Peak Time	2.71s	
Rise Time	1.90s	
Overshoot	18.35%	

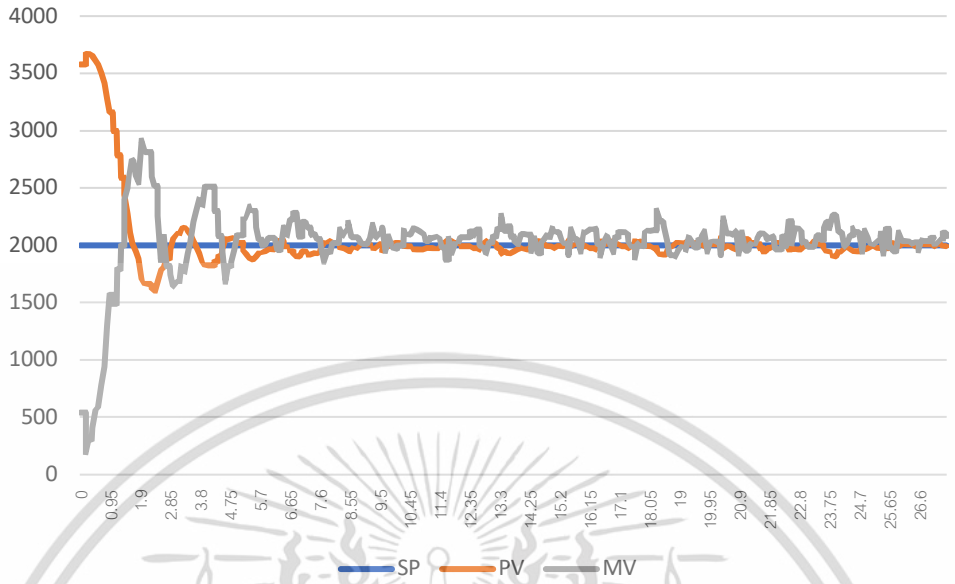
ตารางที่ 4.8 Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 2



รูปที่ 4.23 Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 3

setpoint	3.72s	at error <1cm.
Peak Time	2.34s	
Rise Time	2.21s	
Overshoot	9.7%	

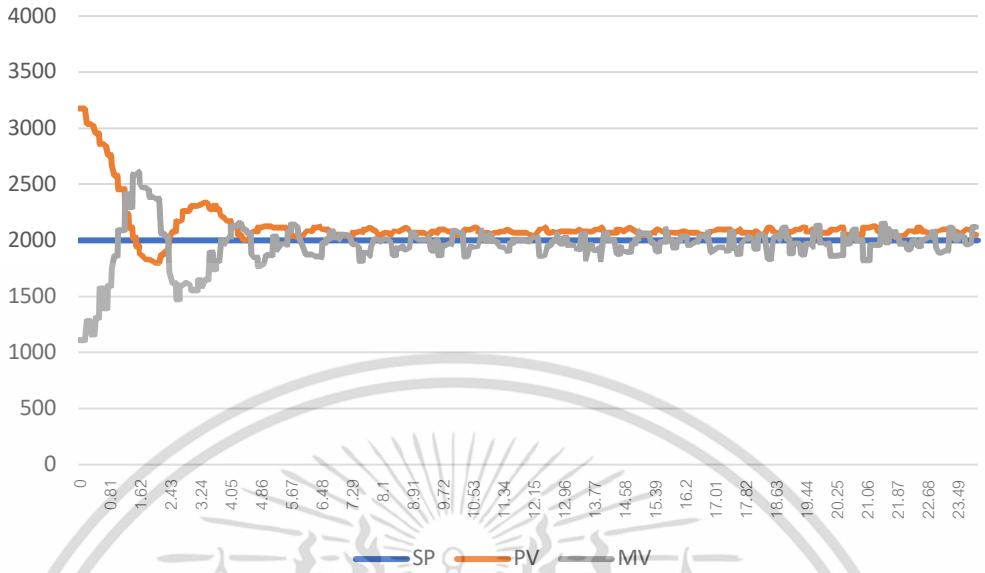
ตารางที่ 4.9 Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 3



รูปที่ 4.24 Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 4

setpoint	5.58s	at error <1cm.
Peak Time	2.36s	
Rise Time	1.77s	
Overshoot	19.85%	

ตารางที่ 4.10 Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 4



รูปที่ 4.25 Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 5

setpoint	6.51s	at error <1cm.
Peak Time	2.00s	
Rise Time	1.54s	
Overshoot	20.2%	

ตารางที่ 4.11 Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 5

จากการทดลองที่ ที่ Setpoint = 20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ทั้ง 5 ครั้งได้ค่าเฉลี่ยดังต่อไปนี้

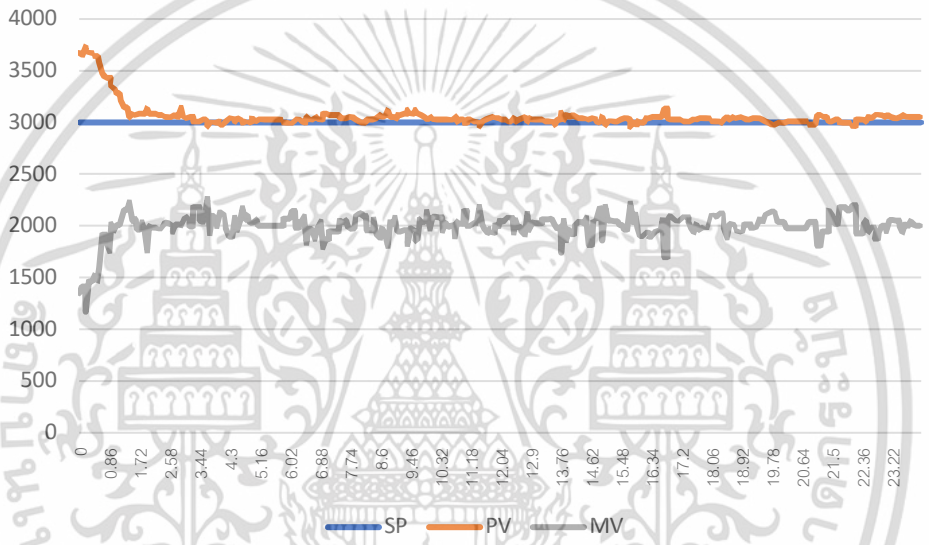
average	setpoint	4.722 s
	Peak Time	2.328 s
	Rise Time	1.83 s
	Overshoot	16.24%

ตารางที่ 4.12 Average ที่ Setpoint=20 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ทั้ง 5 ครั้ง

สรุปผลการทดลอง

จากการทดลองที่ Setpoint = 20 โดยการปรับค่า Tp = 1000 ค่า I (Ti = 9999) และ D (Td=45) จะเห็นว่าสามารถเข้าสู่ Setpoint ได้ ที่ Average time constant = 4.722 s และมี Average overshoot ที่ 16.24% จะเห็นได้ว่ามีการเข้าสู่ Setpoint ที่ 20 เร็วกว่า Setpoint ที่ 10 เพราะมีระยะทางเริ่มต้นที่ใกล้กว่า

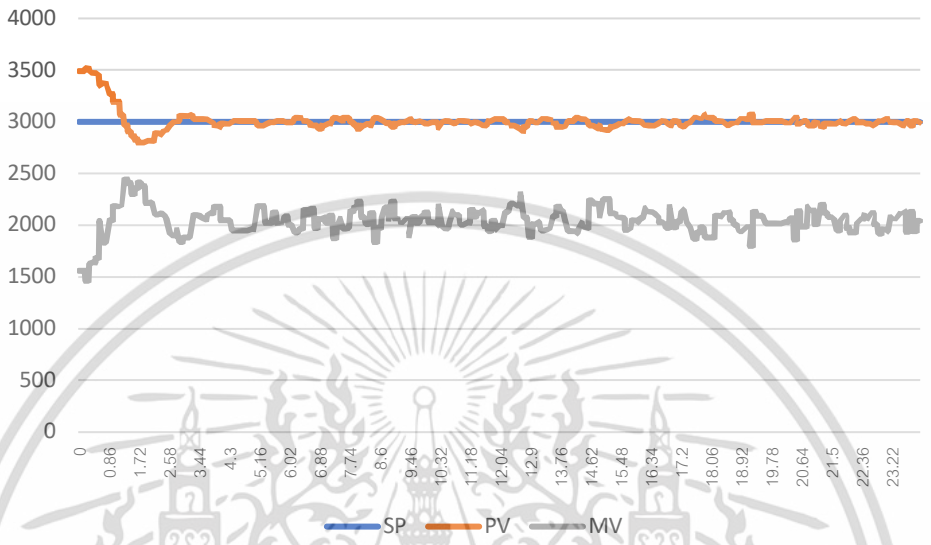
4.2.3 ที่ Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45



รูปที่ 4.26 Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 1

Setpoint	2.98s	Error <1cm
Peak Time	none	
Rise Time	none	
Overshoot	0%	

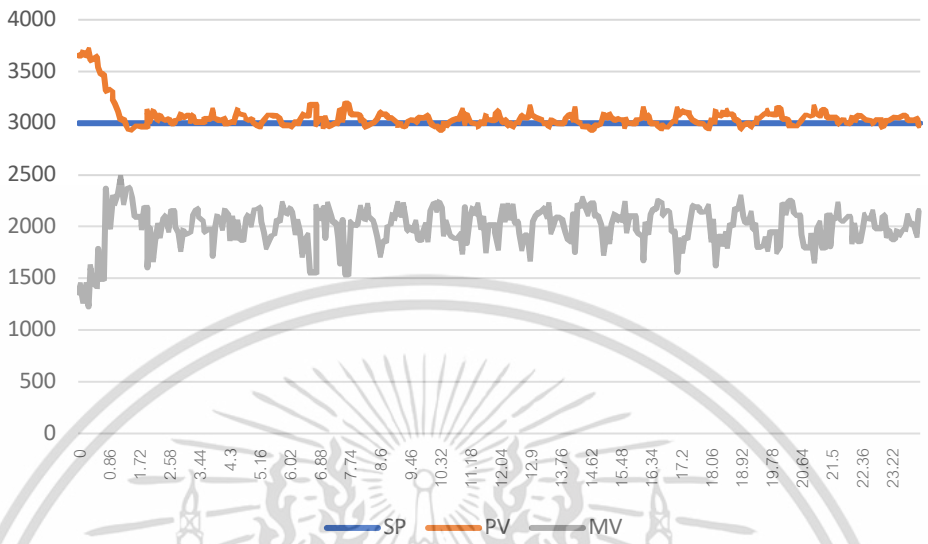
ตารางที่ 4.13 Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 1



รูปที่ 4.27 Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 2

Setpoint	2.42s	Error <1cm
Peak Time	1.70s	
Rise Time	1.31s	
Overshoot	19.9%	

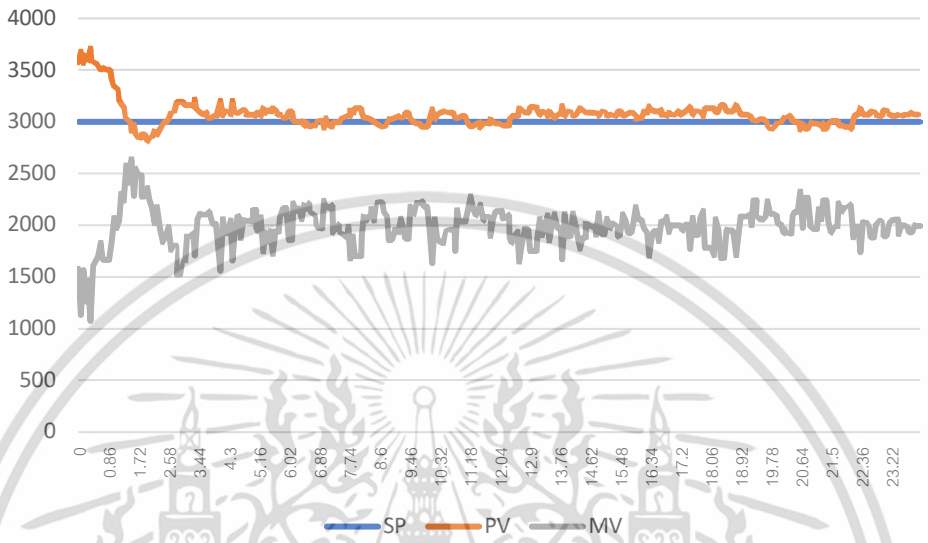
ตารางที่ 4.14 Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 2



รูปที่ 4.28 Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 3

Setpoint	2.19s	Error <1cm
Peak Time	1.45s	
Rise Time	1.37s	
Overshoot	6.4%	

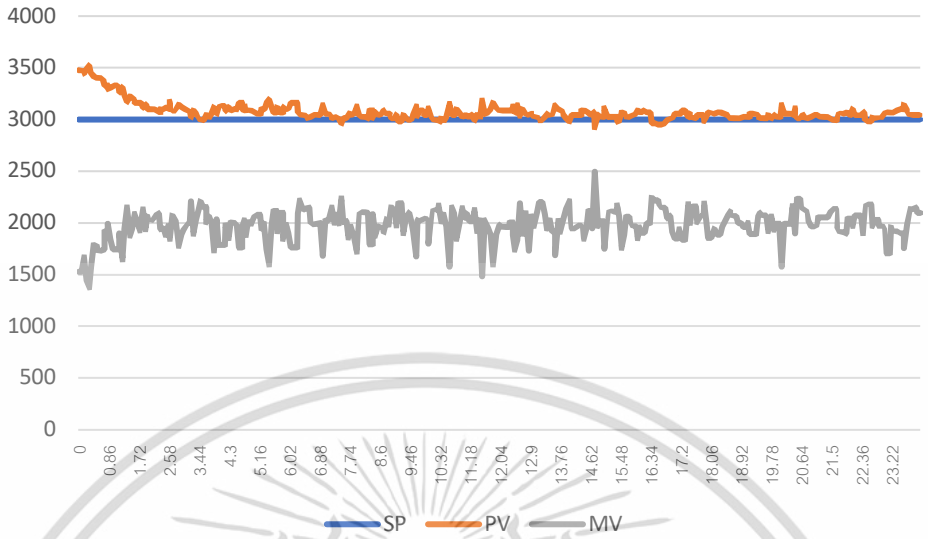
ตารางที่ 4.15 Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 3



รูปที่ 4.29 Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 4

Setpoint	5.69s	Error <1cm
Peak Time	1.97s	
Rise Time	1.51s	
Overshoot	18.4%	

ตารางที่ 4.16 Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 4



รูปที่ 4.30 Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 5

Setpoint	3.08s	Error <1cm
Peak Time	none	
Rise Time	none	
Overshoot	0%	

ตารางที่ 4.17 Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ครั้งที่ 5

จากการทดลองที่ ที่ Setpoint = 30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ทั้ง 5 ครั้งได้ค่าเฉลี่ยดังต่อไปนี้

Average	Time constant	3.272 s
	Rise Time	1.572 s
	Overshoot	8.94%

ตารางที่ 4.18 Average ที่ Setpoint=30 โดยใส่ค่า Tp = 1000, Ti = 9999 และ Td = 45 ทั้ง 5 ครั้ง

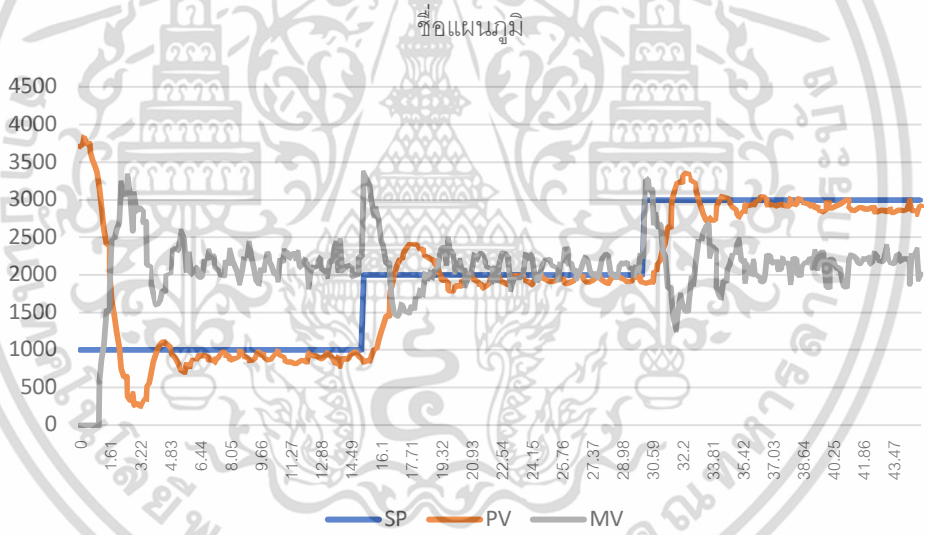
<u>สรุปผลการทดลอง</u>

จากการทดลองที่ Setpoint = 30 โดยการปรับค่า Tp = 1000 ค่า I (Ti = 9999) และ D (Td=45) จะเห็นว่าสามารถเข้าสู้ Setpoint ได้ ที่ Average time constant = 3.272 s และมี Average overshoot

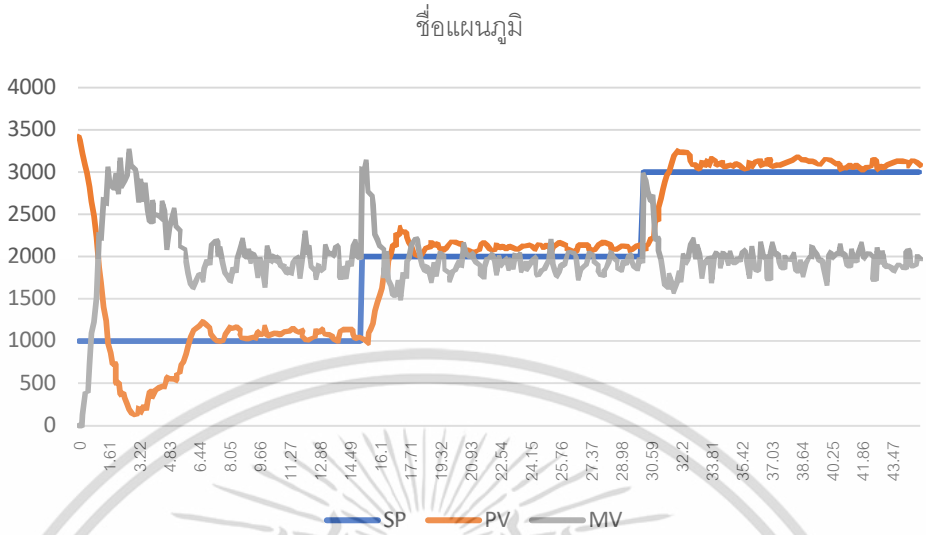
ที่ 8.94% จะเห็นได้ว่ามีการเข้าสู่ Setpoint ที่ 30 เร็วกว่า Setpoint ที่ 10 และ 20 เพราะมีระยะทางเริ่มต้นที่ใกล้กว่า

4.3 การทดลองแบบ Sequence โดยระบบจะทำการปรับค่า set point auto เมื่อเข้าสู่จุดสมดุลของแต่ละ set point (10,20,30)

ระบบ Sequence ใน Auto Mode จะเป็นการควบคุมวัตถุทรงกลมให้สามารถวิ่งเข้าสู่ Setpoint สามจุดตามลำดับได้แก่ 10,20 และ 30 โดยวัตถุทรงกลมจะเปลี่ยน Setpoint ได้ก็ต่อเมื่อสามารถเข้าสู่ Setpoint นั้นๆ ได้เสียก่อน กราฟต่อไปนี้จะแสดง Peak Time, Rise Time และ Overshoot ของระบบ Sequence



รูปที่ 4.31 Sequence ใน Auto Mode ครั้งที่ 1



รูปที่ 4.32 Sequence ใน Auto Mode ครั้งที่ 2

<u>สรุปผลการทดลอง</u>

การทดลองในโหมดการทำงานแบบ Sequence จากกราฟจะเห็นได้ว่าการทำงานในโหมดนี้สามารถทำงานได้อย่างถูกต้องและเวลาในการเข้าสู่ Setpoint เป็นที่น่าพอใจเป็นไปตามการทดลองที่ 4.2

บทที่ 5

สรุปผลและวิจารณ์ปริญญานิพนธ์

5.1 สรุปผลการทดลอง

ในการทดลองพบว่าระบบสามารถควบคุมได้ในโหมด manual โดยใช้ joystick ในการควบคุม Servo motor และสามารถควบคุมได้ในโหมด Auto โดยระบุค่า setpoint ผ่านทาง HMI และ Vision camera จะตรวจจับตำแหน่งของวัตถุทรงกลมและส่งข้อมูลไปยัง PLC ผลการทดลองในโหมด Auto สังเกตุเห็นว่าทุกครั้งหลังจากที่ Load ลูก ball ลงบนคานแล้วจะให้ลูกบอลเคลื่อนที่มาที่ตำแหน่ง 40 cm ก่อน แล้วจึงค่อยเข้าหา Set Point กรณีที่ Setpoint 30 cm ระยะการเคลื่อนของ Object จะสั้นคือ 10 cm เท่านั้น การคุมเข้าหาตำแหน่งเข้าได้เร็ว เมื่อ เทียบกับ Setpoint 10 cm ซึ่งไกลออกไปถึง 3 เท่า ทำให้เกิดการแกว่งและเข้าหา Steady State ของ PV ที่มากกว่าสำหรับการควบคุม กรณีที่ Setpoint 20 cm จะเกิดการสมดุลของ PV และ MV เพราะเป็นการควบคุมที่ 50% ประกอบกับตำแหน่งนี้เป็นตำแหน่งที่จุดหมุนของคานพอดี จึงทำให้การควบคุมที่ 20 cm. มีเสถียรภาพที่สุด

5.2 ปัญหาที่พบในระหว่างดำเนินงาน

5.2.1 จากการทดลองสภาพแวดล้อมมีผลต่อภาพที่ Vision camera สามารถบันทึกได้ คือ เรื่องของแสงหากมีแสงมากเกินไปหรือน้อยเกินไป จะทำให้กล้องไม่สามารถจับวัตถุได้กล้องจะไม่สามารถจดจำสีของวัตถุได้ทำให้ค่าที่ส่งไปยัง PLC มีความคาดเคลื่อน

5.2.2 เนื่องจากบางโปรแกรมที่ได้ใช้ในการทดลองมาไม่มีความรู้ลึกซึ้งก่อนที่จะมาทำ ทำให้ไม่มีความชำนาญในการใช้โปรแกรมทำให้ช่วงแรกยังทำงานได้ไม่เต็มที่นักและยังมีความไม่เข้าใจในชุดเครื่องมือหรือคำสั่ง ต่อมาเมื่อสามารถปรับตัวและได้รับคำแนะนำจากอาจารย์ที่มีเชี่ยวชาญในเรื่องนี้ จึงทำงานได้ดีขึ้นตามลำดับ

5.2.3 การเขียนออกแบบโปรแกรมให้ได้ผลเป็นไปตามที่ต้องการ ต้องใช้เวลาในการตรวจสอบ ทดลอง แก้ไข และพัฒนาคำสั่ง เพื่อจัดการแก้ปัญหาและข้อผิดพลาด

5.2.4 ปัญหาอันเนื่องมาจาก สถานการณ์การแพร่ระบาดของ โควิด-19 เนื่องจากปริญญานิพนธ์ชิ้นนี้ทางคณะผู้จัดทำได้มีการวางขอบเขตการทำงานไว้เรียบร้อยแล้วและขอบเขตการทำงานจำเป็นต้องมีการทำงานร่วมกัน ทำให้การทำงานในช่วงท้ายดำเนินการได้อย่างล่าช้า

5.2.5 ข้อจำกัดของการส่งข้อมูลไปยัง PLC โดย Arduino ต้องใช้โมดูล MAX3232 ในการแปลง port ในการส่งข้อมูลและค่าของ Arduino ตอนแรกเป็นค่าที่ PLC ไม่สามารถรับรู้ได้จำเป็นต้องเขียนโปรแกรมสำหรับการสื่อสารข้อมูลที่เรียกว่า Host link Protocol ให้กับ Arduino เพื่อส่งค่าต่ำแหน่งการเคลื่อนที่ของบอลไปยัง PLC ให้ทำการควบคุมต่อไป

5.3 แนวทางในการพัฒนา

การควบคุมการเคลื่อนที่วัตถุทรงกลมแบบสมดุลของแรงในระนาบเดียวนั้น ยังสามารถนำไปพัฒนาต่อยอดหรือไปประยุกต์ใช้ได้ในอีกหลายสถานการ์ณเช่น เราสามารถนำการทดลองนี้ไปเป็นตัวอย่างหรือเป็นพัฒนาแบบอย่างในการเรียนรู้และประยุกต์ใช้การควบคุม PLC โดยใช้ PID Controller แบบเห็นภาพหรือ Hardware โดยตรง เนื่องจากประสบการ์ณโดยตรงของผู้ทำการทดลองพบว่าในหลักสูตรที่ได้สำเร็จจบมาได้เรียนรู้เกี่ยวกับวิชา PLC และ วิชา Control system ซึ่งมีประโยชน์ต่อการทดลองนี้มากซึ่งเมื่อเราจัดทำโครงการการทดลองนี้ขึ้นมา ทำให้นักศึกษารุ่นต่อไปหรือรุ่นน้องของเราสามารถมาศึกษาต่อยอดได้จากโครงการของเราและยังเป็นประโยชน์ต่อภาควิชาและคณะ ทำให้นักศึกษารุ่นต่อไปได้มีความรู้มากขึ้น

เอกสารอ้างอิง

[1] Omron Co., Ltd, **“CP1H CPU Unit OPERATION MANUAL”**, Inc.2010, Pp.50-287.

[2] Factory Consultant Co., Ltd, **“Wind Plate Control”**, Pp 24-42.

[3] Browne, Michael E. (July 1999). **“Schaum's outline of theory and problems of physics for engineering and science (Series: Schaum's Outline Series).”** McGraw-Hill Companies. Pp. 58. ISBN 978-0-07-008498-8

[4] **“Hooking up Pixy to a Microcontroller (like an Arduino),”** [ออนไลน์]: https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:color_connected_components

[5] **“ทดลอง serial,”** [ออนไลน์]: https://kong-arduino-th.blogspot.com/2014/09%20/serial.html?fbclid=IwAR3-rbpdx0XbG6xum3Q-5M7B5%20tRQy HVPT21tUjWswoeiIEdZ1WgRrYCfVaY

[6] **“What is HMI?”** [ออนไลน์]: https://www.youtube.com/watch?v=kujHQgK352o&t=14s.

[7] **“HMI,”** [ออนไลน์]: https://www.youtube.com/user/sharmalalit96/videos

[8] **“Pixy Camera Color Connected Components”** [ออนไลน์]: https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:color_connected_components

[9] “PID”

[ออนไลน์]: https://www.primusthai.com/primus/Knowledge/info?ID=142

[10] D.E. Seborg, T.F. Edgar and D.A. Mellichamp, **“Process Dynamics and Control”**, John Wiley, New York, 1989.

[11] G. Stephenapoulos, **“Chemical Process Control”**, Prentice-Hall, Englewood Cliffs, 1984.